TECHNISCHE UNIVERSITÄT DRESDEN

FAKULTÄT ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Diplomarbeit

Investigation of Real-Time Scheduling for Broadband PLC Networks

Bibiana Torres Mulase
(Geboren am 14. Februar 1982 in Barcelona)

zur Erlangung des akademischen Grades

Diplomingenieur
(Dipl.-Ing.)

Betreuer	Dipl.-Ing. Le Phu Do
Verantwortlicher Hochschullehrer	Prof. Dr.-Ing. R. Lehnert
Ort und Tag der Einreichung	Dresden, den 4. November 2008

Selbständigkeitserklärung

Hiermit erkläre ich, daß ich die von mir am heutigen Tage dem Prüfungsausschuß der Fakultät Elektrotechnik und Informationstechnik eingereichte Diplomarbeit zum Thema

Investigation of Real-Time Scheduling for Broadband PLC Networks

vollkommen selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 4. November 2008

Bibiana Torres Mulase

Abstract

Broadband Power line Communications (B-PLC) is an access technology which utilizes the low-voltage cables network to offer telecommunication services to the subscribers. In each PLC network cluster, one base station is designed to serve all the connected subscribers, which share the same channel to access the network. The access to the network is controlled by the scheduler implemented in the base station. This scheduler has to allocate the available resources to the users requiring a connection in the form of an allocation table. This allocation table has to take into account which kind of service is requested to provide the resources to the subscribers in accordance with the QoS requirements. In addition, the scheduler has to operate in real time. This task is especially important when the network capacity is not enough to fulfill the requirements from all the users. Therefore, the performance of the network strongly depends on scheduling strategy used to do the bandwidth allocation and the connection management.

In this thesis, the strategies for scheduling based on the evaluation of the benefit of the solutions are studied. An Evolutionary Algorithm, NSGA-II, is used to maximize two confronted objectives, benefit and fairness, managing the constrained network capacity. In addition, a method to improve the execution time is included. The method implemented uses the previous allocation table to find the allocation for the new service requests, and several procedures to do this updating are studied in some different scenarios.

Acknowledgements

This Master Thesis is the culmination of my Degree in Electrical Engineering. During these years, I have acquired a lot of knowledge, not only technical, but also related to life. I have learned what hard work and sacrifice mean, not always with reward; I have learned to be more patient, braver and better person; I have also learned what is to lose the faith, but keep on fighting. But I could not have had the strength to learn all these things without the people who have been by my side during all this time. It is not possible to mention them all now, but they will remain in my mind.

Firstly, I would like to express my gratitude to my supervisor, Le Phu Do, for letting me work in this project and helping me with the support and advice necessary to build this Master Thesis. I am also very grateful to the Chair for Telecommunications for giving me the opportunity of finishing my studies at Technische Universität Dresden.

I would also like to thank to all the people who have made these six months in Dresden an unforgettable experience. Thanks to Eugenio, Nacho, Miguel, Edu, Esther, Carlos, Javi, Dani, Rebeca, Gloria, Michela and many others for all the laughs and good moments lived together.

I don’t want to forget my college fellows and friends in Barcelona, with whom I have spent such an important stage of my life. Thanks to Carmen and Eulàlia for always believing in me, Ana and Pili for the long conversations, and thank you Francis, Costa, Juanca, Nando, Silvia, Jeni, Mosqui, Lluis, Cesar, Willy and the other people for the funny moments lived in Polimenu.

I also remember my colleagues of CBN, with whom I have spent so many good moments not only at work, but also in the dinners, in the parties, in the bolos... After more than two years working with them, some have become good friends.

Of course, I have to thank to my very best friends Marina, Anna, Cris, Sara and Mercè. Most of the good things that I have become are thanks to them. Thanks for being patient with me, for understanding me, for not judging me, for being by my side in the good and in the bad moments.

At last, but not least, I would like to sincerely thank to my family the unconditional support that they have given to me, not only during these years, but in an entire life. I am especially grateful to my mother, Mercedes, and my father, Xavi, who had always taken care of me and have given me everything; the life, the love and what I am.

Table of Contents

Abbreviations

ABR	Available Bit Rate
ACK	Acknowledge
BE	Best Effort
BPL	Broadband Power Line
CAC	Connection Admission Control
CBR	Constant Bit Rate
CPE	Customer Premises Equipment
EA	Evolutionary Algorithm
FTP	File Transfer Protocol
FMN	Flow Master Node
GA	Genetic Algorithm
HE	Head End
HF	High Frequency
IP	Internet Protocol
ISO	International Organization for Standardization
LLC	Logical Link Control
LV	Low Voltage
MAC	Medium Access Control
MOEA	Multi Objective Evolutionary Algorithm
MPDU	MAC Protocol Data Unit
MV	Medium Voltage
NSGA-II	No dominated Sorting Genetic Algorithm II
OFDM	Orthogonal Frequency Division Multiplexing
OPERA	Open PLC European Research Alliance
PHY	Physical layer
PPDU	Physical Protocol Data Unit
QC	QoS Controller
QoS	Quality of Service
REP	Repeater
RX	Reception

SOT	Start of Transmission
TCP	Transfer Control Protocol
TDD	Time Division Multiplexing
TDMA	Time Division Multiple Access
TDR	Time Division Repeater
TS	Time Slot
TX	Transmission
UDP	User Datagram Protocol
URD	Uniform Resource Distribution
VBR	Variable Bit Rate
VLAN	Virtual Local Area Network
VoIP	Voice over IP

Definitions

Active slave - Slave which regularly gets transmission opportunities via the reception of data tokens from its master.

Burst - MAC Service Data Unit passed to the MAC layer by the LLC layer, which includes one or several complete or fragmented packets destined to the same port.

Channel Estimation Frame - MPDU which does not contain any data payload. It starts with a Token Announce followed by a specific sequence of symbols used for channel estimation. It is not terminated by a token.

CPE - BPL unit which only behaves as a slave node.

FDR - BPL unit which combines a HE and either a CPE or a TDR.

Flow - Unidirectional data stream exchanged between two BPL units and carried over the same service class.

FMN - Node that guarantees the QoS of a certain traffic flow. The QC shall be always the FMN of the highest service class of the BPL cell.

Frame - MPDU passed by the MAC layer to the PHY layer. It can be regular or channel estimation MPDU.

HE - BPL unit which only behaves as a master node. It is the root of a BPL cell.

Idle slave - Slave that does not get any transmission opportunities from its master. However, it is regularly polled by its master.

Master - Node responsible for controlling access of its slaves to the network.

Mode - It is defined by a symbol type, a carrier centre frequency and an optional power mask.

MPDU - MAC Protocol Data Unit delivered by the MAC layer to the PHY layer. The kinds of MPDUs are: regular and channel estimation MPDUs (See definition of Channel Estimation Frame). A regular MPDU starts with a Token Announce delimiter, contain zero, one or several bursts addressed to one or several ports and is terminated by a Token delimiter. There are 6 types of regular MPDUs: data, polling, access, access reply, silence and non-returnable. The type of the regular MPDU corresponds to the type of the Token terminating the regular MPDU.

Packet - LLC Service Data Unit passed by the Convergence layer to the LLC layer. Each Ethernet frame received from the Convergence layer is converted into one single packet.

BPL cell - Set of units composed by a HE and the units under the direct or indirect control of that HE. It is operating under the same symbol type and the same carrier centre frequency.

BPL subcell - Set of units composed by a master and the units under the direct control of that master.

BPL sub-tree - Set of units composed by a TDR and the units under the direct or indirect control of that TDR.

Polling - Process under which a master regularly checks the status of its idle slaves, which can be CPEs or TDRs. There are four types of polling identified by the type of the polling token:

1. Alive polling token:
    a. The master can detect if a slave is not connected anymore.
    b. The TDR can detect if its own slave is not connected anymore.
2. Active polling token:
    a. The master can detect if a slave is requiring transmission opportunities of a service class different from 7.
    b. The TDR can detect if its own slave is requiring transmission opportunities of a service class different from 7.

QC - It is the node that distributes the access to the channel among all the nodes present in a PLC cell, and it can be defined as the master of the PLC cell. Any node can become a QC, but it cannot be fixed by the user, it is decided by the nodes. The QC shall be the FMN of the high priority data traffic of the BPL cell.

REP - Refers to either TDR or FDR.

Slave - Node for which transmission opportunities are controlled by its master. These transmission opportunities are specified as Validity Periods carried in some specific tokens delivered by the master. Slaves which are part of a TDR can delegate these transmission opportunities to their associated internal master

SOT - Specific PHY-signal which starts all PPDUs. It is also used as a positive acknowledgement of a polling request.

Spatial reuse - Feature which enables simultaneous node transmissions within non interfering BPL sub-trees of a single BPL cell.

TDR - BPL unit made of both an internal master part and a slave part which never operate at the same time. It behaves either as a master or as a slave at different time periods. Contrarily to a HE, the master part of a TDR is not a permanent master; the right to communicate of the master part of a TDR depends on the Validity Period assigned to its associated slave part.

Token - MAC delimiter which ends a regular MPDU. There are ten types of tokens: data and distributor token are used to propose transmission opportunities to one or more distant node; access/access reply tokens are used to discover unregistered slaves; polling and TDR token are used for the slave and TDR states polling respectively; non-returnable token is used for allocating simultaneous transmission opportunities to several slaves (spatial reuse); silence token is used for sending a regular MPDU and keeping control of the medium after the transmission of this MPDU; CSMA token allow the priorized contention for the following transmission to every node that demodulate it; and clock token allow the synchronization in time of every node in the BPL cell.

Unregistered slave - Slave which is not managed by any master. A master can detect such slaves via the exchange of an access frame and an access reply frame.

List of Figures

List of Tables

1. Introduction

1.1. Motivation

The unparalleled growth of the Internet in the last years has produced a big increase in the number and kind of services offered by Internet providers using IP as networking protocol. Most of these services require a bandwidth that was unreachable for the user only some years ago and that is why technologies such as DSL have been so successful in the past and, in fact, they are still very successful. Nevertheless, this kind of technologies is linked to the possession of a subscriber loop, nowadays controlled by the dominant operator. PLC can be a competitor in this segment and contribute to a real liberalization of the subscriber loop. With power lines, a convenient and inexpensive communication medium for control signalling and data transmission is formed. Houses and industrial buildings are coupled to the power grid, and they have power outlets available in virtually all rooms. These power outlets can be used as communication terminals using a simple standard interface in the form of a wall socket plug. In addition, with the technological advancements of VLSI and digital signal, speeds of up to 200 Mb/s can be reached, and with the QoS mechanism, different bandwidth and latency can be guaranteed for different users and different services. To schedule these traffic flows in order to maximize the overall performance of the BPLC network is the motivation of this Master Thesis.

1.2. Scheduling in Broadband PLC. State of the Art

The Broadband PLC access network is a master-slave system that supports multi-users with multiple services. Each service class requires a minimum resource to achieve its QoS constraints. These service classes can be classified into real-time and non real-time traffic. The main difference between this two kinds of service class is that, unlike non real-time traffic, real-time traffic not only requires a minimum resource to perform correctly but also it is sensitive to the delay experimented by the information. Because there is a maximum transmission capacity available, when there are not enough resources for all the requests from all the users, scheduling algorithms have to be applied. The scheduler has to maximize the overall benefit, has to be fair among all users

and has to assure that the QoS constraints of the different services are guaranteed. So a trade-off solution between fairness and benefit has to be found. Furthermore, this solution must not require a very long calculation time in order to be apt for a real-time system. Thus, the allocation with constrained resources problem is formulated as an optimization problem. In this Thesis, both benefit and fairness will be optimized using an Evolutionary Algorithm, with which a trade-off solution will be found, assuring a maximal execution time suitable for a real-time system.

However, some other solutions have been proposed in order to schedule the traffic flows in a multi-user and multi-service network.

In [1], an efficient cell scheduling algorithm called Dynamic Weighed Round Robin (DWRR) is proposed, which schedules the traffic and preserves the claimed throughput in the outgoing link for each VBR or CBR connection. When there is no arrival of QoS resources, the BE traffic is scheduled.

Some other QoS control algorithms, such as Weighted Fair Queuing [2] and Self Clocked Fair Queuing [3], make resource reservations based on worst-case analysis and maintain these reservations throughout the entire lifetime of the transmission. However, these schemes provide service guarantees at the expense of low resource utilization, making the overall performance of the system to be diminished.

On the perspective of optimization problem specifically applied to PLC, a heuristic algorithm is proposed in [4], in which the benefit is maximized in order to find the best allocation. The achieved benefit and the required execution time of this algorithm are compared with an exhaustive search method, a uniform resource distribution method and a best fill algorithm.

2. The Broadband PLC Network

2.1. General Description of the Electrical Network

PLC uses the electrical distribution network for data transmission. The electrical energy arrives to the users in a low frequency (50 or 60 Hz). PLC uses the high frequency carriers to transport the data, usually between 1 and 34 MHz. But the electrical channel has very high noise levels and narrow band interferences; in consequence, a robust codification against them has to be used: OFDM.

The electrical networks, from generation to subscriber, can be divided in several trams.

- A first tram of Medium Voltage (between 15 and 50 kV), from the energy generation central to the first transformer.
- A Transport tram or High Voltage Tram (between 220 and 400 kV) that leads the energy to the transport substation.
- A Medium Voltage Tram (from 66 to 132 kV) between the transport substation to the distribution substation.
- A last tram of Medium Voltage (between 10 and 50 kV) from the distribution substation to the distribution centre.
- The Low Voltage network (between 220 and 380 V) which distributes the energy to the urban centres, for domestic, industrial and commercial use.

The application of PLC is in the last two trams, in the Medium Voltage and the Low Voltage. The Low Voltage tram is used as the access network, while the Medium Voltage one is used as distribution network [5]. This work focuses on the "last mile" problem, so the access network will be described with more detail.

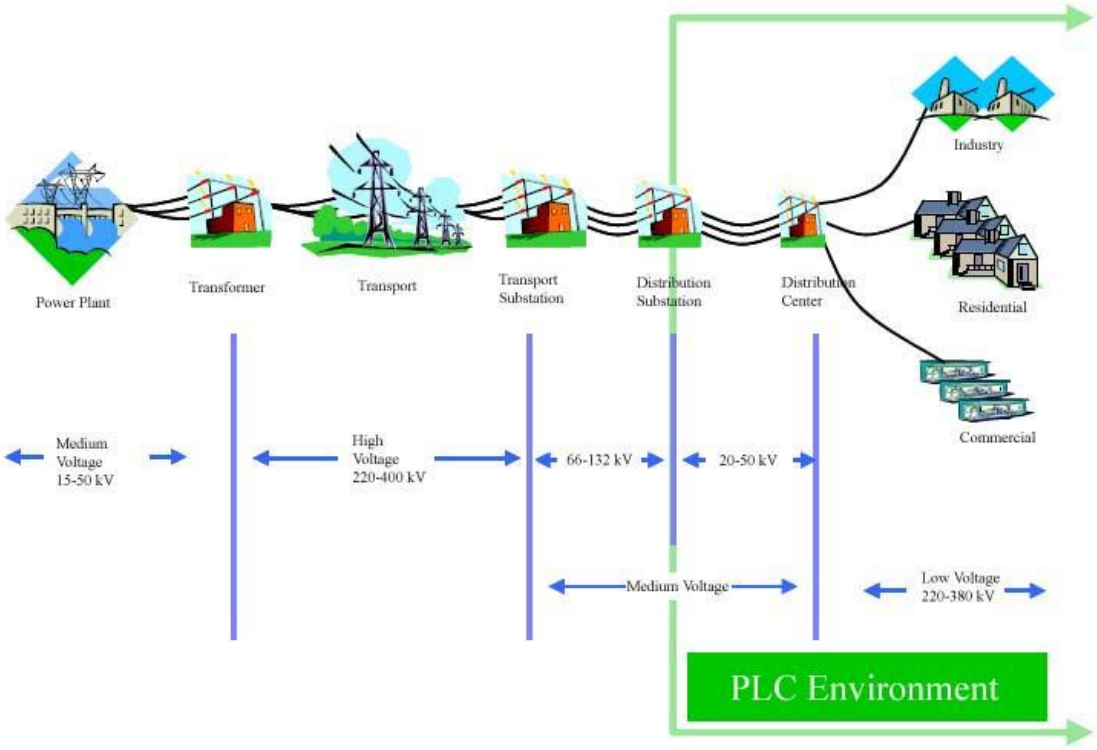

Figure 1. The electrical network and PLC [5]

2.2. The Access Network

The network structure proposed by the OPERA specification [6] consists in a cellular network, where each BPLC cell consists of a number of user terminals (CPEs: Customer Premises Equipment) that transmit/receive traffic in a shared medium to/from a centralized station (HE: Head End). If the signal is too attenuated to reach all CPEs from the same HE, repeaters (REP) can be inserted in the network in order to retransmit the signal and thus increase the coverage. Repeaters can be either TDR (Time Division Repeater) or FDR (Frequency Division Repeater).

The cells can be connected to the backbone network via an optical link or via PLC over medium voltage lines.

From this description, the types of topologies that are usually found in BPL access networks are tree-like topologies. In this type of topologies, the HE, which is the central node, concentrates all of the upstream and downstream traffic. However, other topologies, as the ring shape topology, are also common. All kinds of structures in Medium Voltage (MV) and Low Voltage (LV) can be reduced to a HE plus Repeaters structure.

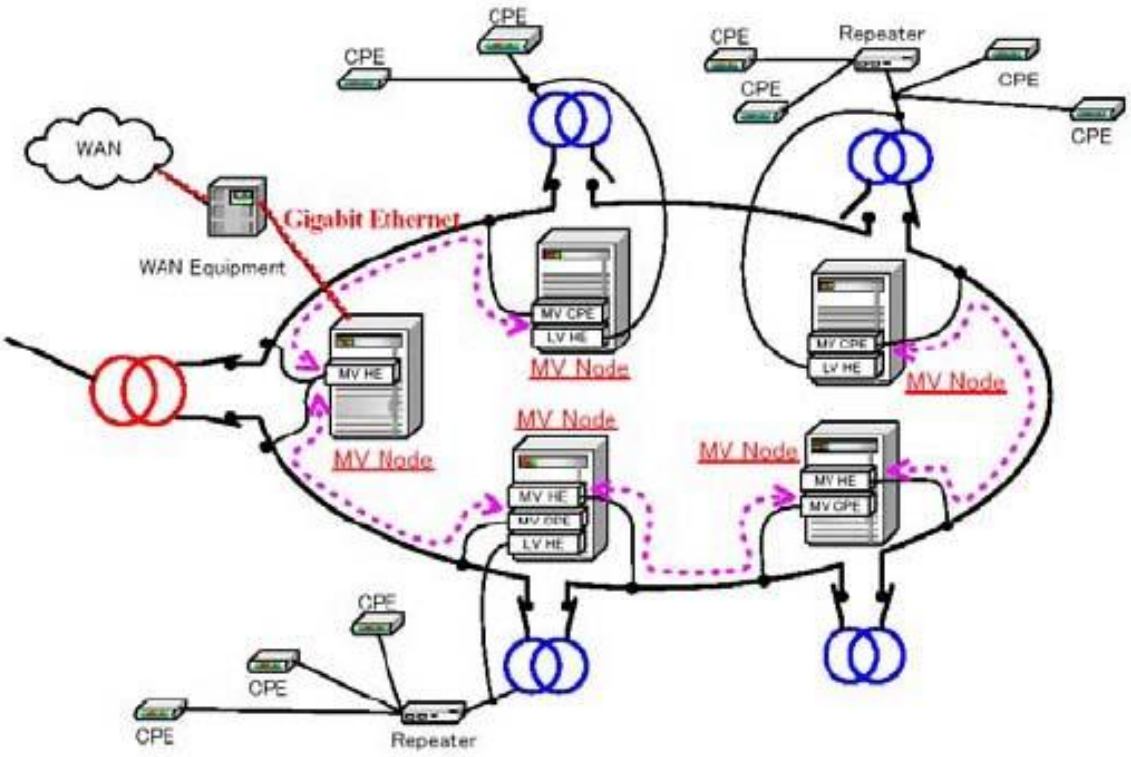

Figure 2. Distribution network in Medium Voltage [5]

2.3. Nodes

A BPL cell is composed, from a Medium Access Control point of view, by a HE and the set of units (CPEs or TDRs) under the direct or indirect control of that HE operating under the same symbol type and the same carrier frequency. Its composition is:

1. One HE.

2. From zero to several TDRs.

3. One (or several) CPE(s).

To guarantee the required performance in terms of Quality of Service, the nodes can also be divided in three types, depending on which is their function to assure the Quality of Service: QoS Controllers (QCs), assumed to be a HE in a typical BPL cell, Flow Master Nodes (FMN), that are the nodes that assure the resources needed by a certain class of service traffic flow, and End Points that have no management tasks regarding QoS assurance. More details about QoS in the BPL cell will be described in section 2.5.

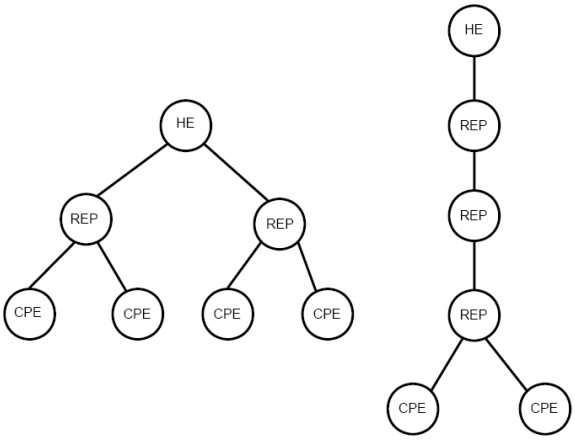

Figure 3. Typical access scenario [6]

2.3.1. Head End

The HE is the central node that controls the entire BPL cell. It assigns resources to all nodes of the BPL cell through the use of the token, according to the QoS requirements of the flows circulating on the BPL cell. The HE will always be the master of any node directly connected to it.

The HE injects the signal from the data backbone in the MV network. Its configuration can differ, depending on the type of backbone. If the network in which it is connected is a "classic" network, the HE will have to use an Ethernet interface to be able to connect to the switch or router connected to the backbone. On the other hand, if the HE is connected to a MV network, it will integrate a PLC modem of MV which will make the connection to the HE of MV.

2.3.2. Time Division Repeater

A TDR is used to increase the coverage in areas too far from the cell's HE. TDRs are connected to the HE or to other TDRs that act as their master node. TDRs share the channel allocated to them by their master node and distribute it among their slave nodes according to the traffic flows and service classes in the BPL cell and the origin and destination of them. The TDR will be the slave of the HE or of another TDR, and will be the master of its slaves.

2.3.3. Customer Premises Equipment

CPEs are BPL units installed in the customer's household. A CPE must subscribe to the network before being able to access the channel. Subscribing to the network means selecting a master node that assigns channel access time. In order to subscribe to the network, a validation process is run that acknowledges that the CPE is valid. After being accepted into the network, the CPE automatically downloads a file (auto-configuration process) detailing the parameters to use, such as the user profile. The CPE is always a slave.

The CPE is usually composed of a device called electric coupler and a PLC modem. The electric coupler separates the high frequency data signal from the alternating current and delivers them to the modem. It is a passive device which, basically, consists on a Low Pass Filter, which separates the electric current, and a High Pass Filter to extract the HF. The electric coupler also injects the AF signal to the electrical network. The modem receives the HF signal and is the responsible of demodulating it, extracting and delivering the data to the backbone.

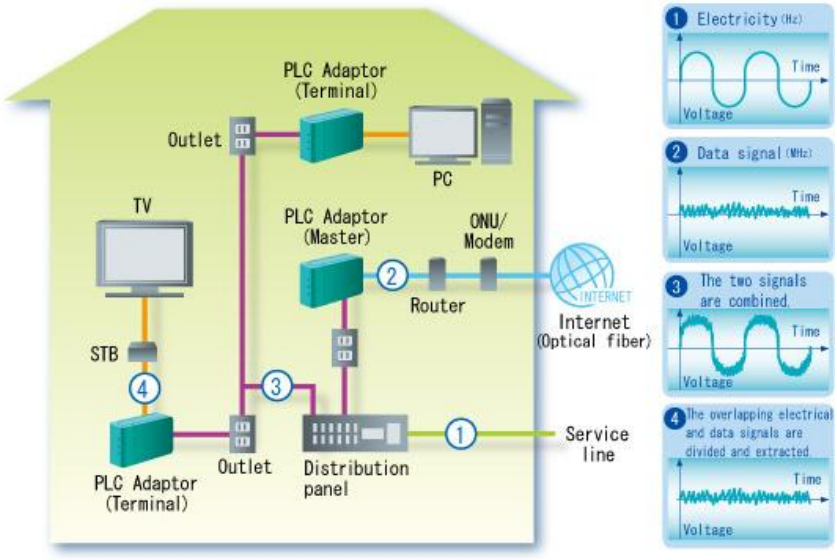

Figure 4. In-home PLC network [7]

2.4. MAC

According to the draft proposed for the IEEE P1901, the access to the channel should be managed in a hierarchical way. The HE should schedule the use of the channel

depending on the current traffic flows and the QoS requirements. The access to the channel can be managed using two main procedures.

1. First procedure consists on sending individually to each of the slaves a data or distributor frame. This frame contains data from the HE to the slaves and it is finished by a token that contains the next slave or list of slaves that shall use the channel when the transmission is finished. The amount of channel time assigned by the HE can be fixed or not. If a node does not use the entire assigned time, the right to transmit is immediately given to the following node in the list or returned to the HE.
2. The second procedure to manage the access to the channel consists on sending a CSMA token that opens a priorized contention among the nodes that receive it. The winner of the contention obtains the right to transmit for a given amount of time. Once this node has finished its transmission, the right to transmit is returned to the HE. By means of the first procedure, a timing interval can be programmed where different slaves can allocate their transmissions.

2.5. QoS

Several mechanisms [6] to assure the QoS of the traffic flows in the BPL cell are provided. Traffic flows are session oriented and tagged with certain service class. Belonging to a service class implies certain requirements in terms of latency and bandwidth, jointly with a requirement in resource reservation. A slave which requires transmitting certain type of traffic shall first classify it in one of the eight available service classes, each one with a pre-programmed latency, bandwidth and level of assurance of the required resources that can be Variable Bit Rate (VBR), Constant Bit Rate (CBR), Available Bit Rate (ABR) or Best Effort. The mapping from Service Classes to Bandwidth, Latency and Type of Traffic is common and known by all nodes in the BPL cell. Then, by means of the CAC protocol, the requirements to transmit the traffic flow are solicited to the Flow Master Node of this service class, and can be accepted and a session ID shall be assigned to it, or rejected. In the case where the maximum capacity of the channel is reached, several congestion policies can be applied by the QC and FMNs to admit, reject or drop current sessions in the BPL cell.

2.5.1. Service Class Definition

Up to eight service classes can be defined within a BPL cell. Services classes are globally defined over a BPL subcell. Four service parameters are necessary to fully describe a service class: priority, subcell access time, resource reservation type and service reliability.

- Priority: It uniquely identifies a service class. The priority values must be between 0 and 7, being 0 the lowest priority. The packets with lower priority will be dropped before the higher priority ones when there is lack of resources. When there is data transmitted to the same destination node, the higher priority packets will be transmitted earlier than the lower priority ones.

- Max Subcell Access Time: It corresponds to the maximum duration for a flow to access the channel on the subcell level. It is a latency requirement for the scheduler to be configured within any master of a BPL cell. There are four values for the Max Subcell Access Time, where a minimum value is used as reference step from which the other three values will be obtained by multiplying it by 2, 4, or 8.

- Resource Reservation Type: This parameter is related to the claimed guarantees in terms of node and time resources. The resource reservation types are:

Best Effort: There aren't guarantees neither in node nor in time resources. It is a non connected flow, and there is a bandwidth limitation in upstream and downstream flows.

ABR: A minimum bandwidth can be guaranteed, with a peak over this minimum, which can be limited by the network capacity or by the node limit bandwidth.

VBR: An average bandwidth can be guaranteed with a maximum variation around this average. There is no bandwidth limitation with this kind of traffic flows.

CBR: Individual guarantees in terms of node and time resources are asked, and the claimed bandwidth is guaranteed. Like VBR traffic, no bandwidth limitation can be applied.

- Service reliability: This parameter defines whether the ACK mode is enabled or not.

The service class definition shall be the same for all nodes in the BPL cell, and it must determine univocally the relationship between priority, maximum subcell access time, resource reservation and service reliability. This service class definition must be used by QoS to determine the priorities between flows and resources reservation in the network, and will provide criteria to allow or deny new flows or to drop existent flows to allow new flows which higher service class. In Table 1, the relationship between the different QoS parameters can be seen, as well as some examples of different services.

Service class	Resource reservation	Example	Max subcell access time (ms)	Service reliability
7	BE	Management messages	240	No
6	CBR	VoIP	30	No
5	VBR	Video, Games	60	Yes
4	VBR	Data high prio: E-commerce	120	Yes
3	VBR	Data high prio: Tele-services	120	No
2	ABR	Data high prio: HTTP	120	Yes
1	BE	Data low prio: E-mail	240	No
0	BE	Data low prio	240	No

Table 1. Service class definition

3. Scheduling – Problem Formulation

3.1. Introduction

The advent of high speed networking has introduced opportunities for new applications such as video conferencing, scientific visualization and medical imaging. These applications have stringent performance requirements in terms of throughput, delay, delay jitter, and loss rate. Current packet-switched networks (such as the Internet) offer only a best-effort service, where the performance of each session can degrade significantly when the network is overloaded. In this work, a scheduling strategy has been developed to control the interactions among the connections from the different users in order to assure a good quality in the transmissions, even when the network is overloaded. Since this Thesis is focalized in real time systems, the algorithm implemented in the scheduler should be fast enough to assure that the new allocation of resources is calculated before the schedule updates the requests. The scheduler is supposed to be updated every Time Slot, so an algorithm faster than 60 ms should be implemented.

3.2. Estimation of Traffic Requirements

When a CPE wants to transmit, it must send a request to its HE. In this request, information about the kind of service that is required has to be included. If the request is accepted, the HE passes it to the scheduler, and then the bandwidth broker allocates additional bandwidth for the new service. Then, the CPE is informed about the new allocation, and the transmission begins.

In this work, a simple cell in the multiple cell structure is considered, with a specific transmission capacity to spend for transmissions of its devices. The HE of the cell has several CPEs directly connected to it, which send transmission requests of several types of services. The bandwidth broker in the HE has to divide the available resource to fulfil the requirements for the different services from the different users. As the system is TDMA, the bandwidth assigned for a service is defined as follows [4]:

$$\beta_{ij} = ES_{Ui} \cdot \frac{T_{ij}}{T_F}$$

Where T_{ij} is the given time duration (Time duration for the service *j* from user *i*), T_F is the frame duration and ES_{Ui} is the effective data rate from the HE to the CPE_i.

An example of this kind of configuration is depicted in Figure 5. It can be observed that the resource is the time assigned for each traffic flow, which is proportional to the bandwidth. There are several possible bandwidth assignments, but the services that require a predetermined amount of resources (in this case, time resources) should be allocated first in order to assure the transmission in case the resources are not enough. The services classified as Best Effort, such as HTTP, are allocated then, because they don't need a minimum guaranteed resource to be transmitted.

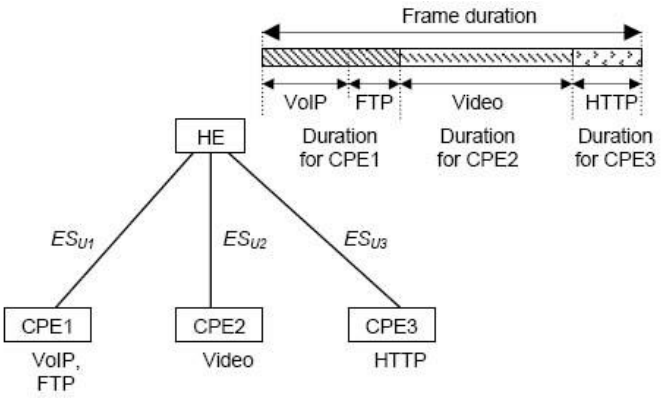

Figure 5. Example of resource allocation in the HE [4]

This allocation policy assures a high benefit, because services such as VoIP or Video, with high bandwidth requirements, achieve high benefit levels. But when the resources are very limited, the opportunities to best effort services can be diminished and this situation can cause that all the low benefit traffic flows were not able to transmit and consequently, make the resource allocation very unfair. Therefore, to solve the allocation problem two main parameters have to be taken into account: Benefit and Fairness. The scheduler has to find the highest benefit and most fair solution to allocate the resources from the different services and users of the system. But we need to make a clear definition of these parameters before we find the algorithm to make this optimization.

3.2.1. Benefit of a schedule

The benefit achieved for a user is determined by his individual service assignment. This means that a granted VoIP connection gets a higher benefit than a data connection. If we assume that the benefit is the utility of bandwidth granted to a service, the higher bandwidth allocated for a service class, the higher the benefit it achieves.

As in [4], we consider a BPLC system with a base station and N users from U_1 to U_N. Each user can provide S service classes, decreasingly sorted by their benefits, from S_1 to S_S. In Table 2, the resource requirement and benefit achieved by the different service classes used in this Thesis are depicted. In the simulation analysis, the resource unit is normalized to simplify the calculation.

Service Class	Throughput	Benefit b_j	Example
S1	64 kbps	4	VoIP
S2	128 kbps	8	Video, Game
S3	32 kbps	2	HTTP
S4	1 kbps	1	e-mail, data low priority

Table 2. Service and resource requirements

Thus a definition of the individual benefit for each user depending on the service class is done. To make a definition of the overall benefit achieved by all the system, it is needed to define the request matrix and the granted matrix.

The request matrix is defined as:

$Y = \{y_{ij}\}$ where y_{ij} is the number of requests for service j from user i and

$$0 \leq y_{ij}, y_{ij} \in \mathbb{N}$$

And the granted matrix:

$X = \{x_{ij}\}$ where x_{ij} is the number of granted requests of service j to user i and

$$0 \leq x_{ij} \leq y_{ij}, x_{ij} \in \mathbb{N}$$

The figure below shows an example of a request matrix and its granted matrix. It can be seen that the values of the granted matrix have to be lower than its associated request matrix values. The values of each row correspond to the requests of a user, while the values of the columns represent the requests associated to a determined

$$Y = \begin{array}{c} \\ U_1 \\ U_2 \\ \vdots \\ U_N \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 2 & 1 \\ 2 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 2 & 0 & 1 \end{bmatrix} \end{array}$$

$$X = \begin{array}{c} \\ U_1 \\ U_2 \\ \vdots \\ U_N \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Figure 6. Request and granted matrix

With the definition of these matrices, we can make a calculation of the global benefit obtained by the system for a given resource allocation. We firstly define the benefit obtained by a user, $B_{Ui,}$ as:

$$\boldsymbol{B}_{Ui} = \sum_{j=1}^{S} \boldsymbol{x}_{ij} \cdot \boldsymbol{b}_j$$

Where b_j is the associated value of the benefit for the service class j. In the same way, a definition of the total benefit achieved for a given granted matrix is:

$$\boldsymbol{B} = \sum_{i=1}^{N} \boldsymbol{w}_i \cdot \sum_{j=1}^{S} \boldsymbol{x}_{ij} \cdot \boldsymbol{b}_j = \sum_{i=1}^{N} \boldsymbol{w}_i \cdot \boldsymbol{B}_{Ui}$$

Where w_i is the weight associated to the user i, which depends on the importance of the user, which can be measured by, for example, how much he pays. In this study, for simplicity reasons, all the users are supposed to have the same weight, thus, in a system with N users, the weight of each user is $w_i = \frac{1}{N}$, consequently, and the total benefit is:

$$\boldsymbol{B} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \cdot \sum_{i=1}^{N} \boldsymbol{B}_{Ui}$$

In this definition for the benefit, it can be observed that, in the case all the users have the same weight; the total benefit for a concrete bandwidth allocation is an average of all the benefits achieved by all the users. Further discussion about the semantic description of the benefit will be exposed in the simulation study.

3.2.2. Fairness among users

The fairness of a system, the users in a system, or of a scheduling algorithm is not usually expressed in terms of a quantitative value. Generally, a system is deemed to be fair or unfair based on whether or not the system meets certain criteria. In this work, a quantitative definition for the fairness is needed, as it is one of the functions, jointly with the benefit, that we need to optimize in order to find an allocation solution. We will use the definition and quantitative expression of fairness shown in [8] when all the users are equal weighted.

We define the self-fairness of a user *i* as:

$$\mathfrak{F}_i = \frac{-log(p_i)}{-log(\frac{1}{N})} = \frac{-log(p_i)}{log(N)}$$

Where p_i is the proportion of resource allocated to user *i*. This means that p_i is the result of dividing the amount of resource given to user *i* per the total amount of resource available in the system. That is:

$$p_i = \frac{\sum_{j=1}^{S} R_{ij} \cdot x_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{S} R_{ij} \cdot x_{ij}} = \frac{R_{Ui}}{R_{TOT}}$$

Where R_{ij} is the resource allocated for user *i* for granted service *j*, R_{Ui} is the total resource allocated for user *i*, and R_{TOT} is the total resource allocated for all the users and all the services. For simplicity, the resource allocated will only depend on the type of service; this means that all the users will achieve the same amount of resources when they get a connection of a certain service. Thus the proportion of resource allocated to user *i* is:

$$p_i = \frac{\sum_{j=1}^{S} R_j \cdot x_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{S} R_j \cdot x_{ij}} = \frac{R_{Ui}}{R_{TOT}}$$

Some interesting properties of the self-fairness are:

1. When a user consumes exactly its fair share of resources (in the case of equal weighted users, this share would be *1/N*), the value of the self-fairness becomes the unity.
2. As the amount of resources a user consumes increases, it takes an increasing amount away from the other users in the system. Thus, that user becomes less fair and consequently, the value of self-fairness for that user decreases.
3. Conversely, as a user consumes fewer resources, it is giving up resources in favor of the other users in the system. Hence, that user becomes fairer and correspondingly, the value of self-fairness for that user increases.
4. In the limit as one user consumes all the available resources, all the other users in the system will be starved. In this situation, there is no possible way for the greedy user to be less fair. Thus, logically, the value for the self-fairness for that user should be the lowest possible value, zero. Having zero as a minimum makes sense, as negative values of fairness do not really have much meaning from a semantic point of view.
5. Similarly, in the limit when a user consumes no resources, that user gives up all its deserved resources in favor of other users. Thus, there is no possible way for that user to be fairer. This is reflected in the resulting values of infinity for the self-fairness.

The proposed definition for the average fairness of a system with *N* users is:

$$\bar{\mathfrak{F}} = \sum_{i=1}^{N} p_i \cdot \mathfrak{F}_i = -\sum_{i=1}^{N} p_i \cdot \frac{log(p_i)}{log(N)}$$

The value for the average fairness of the system of *N* users ranges from 0 to 1 inclusive. Furthermore, the maximum value of unity is only achieved when all users in the system consume exactly their fair share of the resources (i.e. $p_1 = p_2 = ... = p_N = 1/N$). The minimum of zero occurs in the limit when one user consumes all the allocated resources while the other *N-1* users are starved.

3.3. Problem Formulation

The BPLC network is a multi-user and multi-service network. We use the *Y* matrix and *X* matrix to express the requests of different services for the different users, and the amount of accepted requests of these users and services, respectively. Several

strategies can be adopted to decide which the requests that are accepted are. The strategy selected has to guarantee the highest benefit and the fairest allocation, having into account that the resources available are limited. In addition, the strategy adopted has to be fast enough to satisfy the time requirements of the real time connections. Thus we can formulate our problem as follows:

Find the granted matrix $X = \{x_{ij}\}$ which:

Maximizes the overall benefit

$$B_{\mathrm{i}} = \frac{1}{N} \cdot \sum_{i=1}^{N} B_{\mathrm{Ui}}$$

And maximizes the average fairness of the system

$$\overline{\mathfrak{F}} = \sum_{i=1}^{N} p_i \cdot \mathfrak{F}_i = -\sum_{i=1}^{N} p_i \cdot \frac{\log(p_i)}{\log(N)}$$

Such that $0 \leq x_{ij} \leq y_{ij}$

And

$$R_{TOT} = \sum_{i=1}^{N} \sum_{j=1}^{S} R_{ij} \cdot x_{ij} = \sum_{i=1}^{N} R_{Ui} \leq R_{MAX}$$

Where R_{MAX} is the total available resource of the system.

4. Multi Objective Optimization

4.1. General description

Although some real world problems can be reduced to a single objective, very often it is difficult to define all the aspects in terms of a single objective. Defining multiple objectives often gives a better idea of the task. This multi objective optimization has been available for about two decades, and its application to solve real world problems is continuously increasing. While a big research have been done to develop single objective optimization techniques, relatively few work have been done for multi objective optimization. In single objective optimization, the search space is often well defined. As soon as there are several possibly contradicting objectives to be optimized simultaneously, there is no longer a single optimal solution but a whole set of possible solutions which have equivalent quality.

In this work, two contradicting objectives have to be maximized: benefit and fairness. The benefit increases faster when data from higher priority (and thus, higher benefit) are scheduled, rather than data with low priorities. But if only traffic flows with high priorities are scheduled, the system becomes unfair because there are not opportunities for the low benefit transmissions. That is why it is necessary to find an algorithm which takes into account both benefit and fairness. When several objectives have to be optimized at the same time, the search space becomes partially ordered. To obtain the optimal solution, there will be a set of optimal trade-offs between the conflicting objectives.

A multi objective optimization problem is defined by a function *f* which maps a set of constraint variables to a set of objective values. As shown in Figure 7 a solution could be best, worst and also indifferent to other solutions (neither dominating nor dominated) with respect to the objective values. Best solution means a solution not worst in any of the objectives and at least better in one objective than the other. An optimal solution is the solution that is not dominated by any other solution in the search space. Such an optimal solution is called Pareto optimal and the entire set of such optimal trade-offs solutions is called Pareto optimal set. As evident, in a real world situation a decision

making (trade-off) process is required to obtain the optimal solution. Even though there are several ways to approach a multi objective optimization problem, most work is concentrated on the approximation of the Pareto set [9].

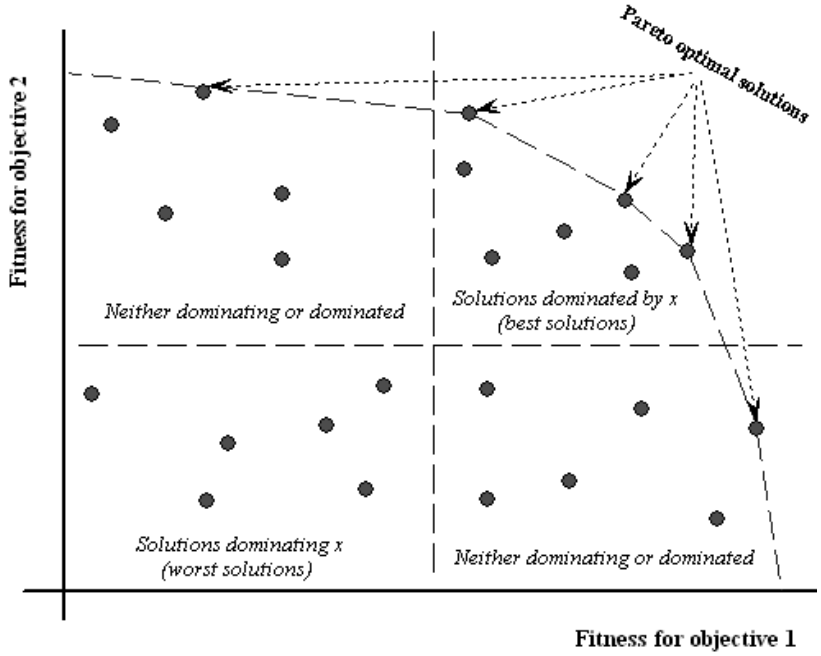

Figure 7. Concept of Pareto optimality [9]

4.2. Evolutionary Algorithms

Evolutionary algorithms have become the method at hand for exploring the Pareto-optimal front in multi objective optimization problems that are too complex to be solved by exact methods, such as linear programming and gradient search. This is not only because there are few alternatives for searching intractably large spaces for multiple Pareto-optimal solutions. Due to their inherent parallelism and their capability to exploit similarities of solutions by recombination, they are able to approximate the Pareto-optimal front in a single optimization run. The numerous applications and the rapidly growing interest in the area of multi objective EAs take this fact into account.

Evolutionary algorithms include evolutionary strategies, genetic algorithms and differential evolution.

All evolutionary algorithms aim to improve the existing solution using the techniques of *recombination*, *mutation*, and *selection*. The general paradigm is as follows [10]:

1. Initialization: The initial population consisting of μ members (parents) is chosen randomly.

2. Recombination: The μ parent vectors randomly recombine with each other to produce $\lambda \geq \mu$ child vectors.

3. Mutation: After recombination, the λ child vectors undergo mutation where a random deviation is added to each child vector.

4. Selection: The two most commonly used selection strategies are (μ, λ) and $(\mu, \mu + \lambda)$ selection strategies. In (μ, λ) strategy, the best μ child vectors replace the existing μ parent vectors to become parents in the next generation, whereas in $(\mu, \mu + \lambda)$ strategy, the best μ vectors from the child and parent populations become parents in the next generation.

5. Termination: The number of iterations (generations) performed depends on the convergence criterion chosen.

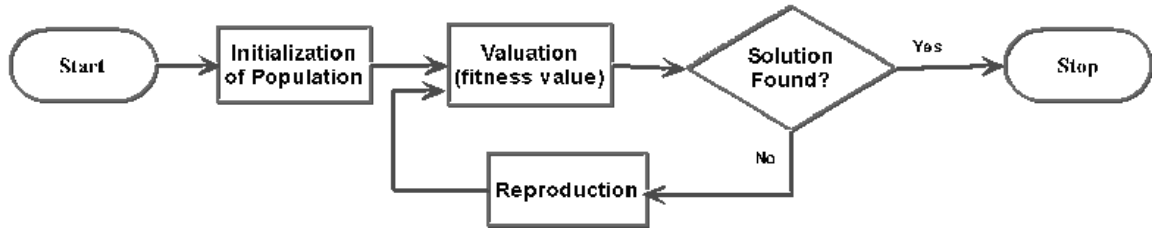

Figure 8. Flowchart of evolutionary algorithm iteration [9]

Although, evolutionary strategies and genetic algorithms are categorized as evolutionary algorithms, they have an important difference: Evolutionary strategies encode parameters as floating point numbers and then manipulate them by using arithmetic operators whereas genetic algorithms encode parameters as bit strings and then manipulate them by using logical operators. So, evolutionary strategies are suitable for continuous optimization while genetic algorithms are more suitable in combinatorial optimization.

4.2.1. Definitions

In contrast to fully ordered scalar search spaces, multidimensional search spaces are only partially ordered, i.e. two different solutions are related to each other in two possible ways: either one dominates the other or none of them is dominated.

Definition 1: *Consider without loss of generality the following multi objective optimization problem with m decision variables x (parameters) and n objectives y:*

$$Maximize\ \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m))$$
$$where\ \boldsymbol{x} = (x_1, \dots, x_m) \in X$$
$$\boldsymbol{y} = (y_1, \dots, y_n) \in Y$$

and where **x** *is called* decision (parameter) vector, *X* parameter space, **y** objective vector *and Y* objective space.

A decision vector $\boldsymbol{a} \in X$ *is said to* dominate *a decision vector* $\boldsymbol{b} \in X$ (*also written as* $\boldsymbol{a} \succ \boldsymbol{b}$) *if and only if:*

$$\forall i \in \{1, \dots, n\}: fi(\boldsymbol{a}) \geq fi(\boldsymbol{b})$$
$$\forall j \in \{1, \dots, n\}: fj(\boldsymbol{a}) > fj(\boldsymbol{b})$$

Additionally, we say **a** covers **b** ($\boldsymbol{a} \succcurlyeq \boldsymbol{b}$) *if and only if* $\boldsymbol{a} \succ \boldsymbol{b}$ *or* **f** (**a**) = **f** (**b**).

Based on this convention, we can define non dominated, *Pareto-optimal* solutions as follows:

Definition 2: *Let* $\boldsymbol{a} \in X$ *be an arbitrary decision (parameter) vector.*

(*a*) *The decision vector* **a** *is said to be* non dominated *regarding a set* $X' \subseteq X$ *if and only if there is no vector in X' which dominates* **a***; formally:*

$$\nexists \boldsymbol{a}' \in X' : \boldsymbol{a}' \succ \boldsymbol{a}'$$

(*b*) *The decision (parameter) vector* **a** *is called* Pareto-optimal *if and only if* **a** *is non dominated regarding the whole parameter space X.*

If the set *X'* is not explicitly specified, the whole parameter space *X* is implied. Pareto-optimal parameter vectors cannot be improved in any objective without causing degradation in at least one of the other objectives. They represent in that sense *globally optimal* solutions. Note that a Pareto-optimal set does not necessarily contain all Pareto optimal solutions in *X*. The set of objective vectors $\boldsymbol{f(a')}$, $\boldsymbol{a}' \in X'$, corresponding to a set of Pareto-optimal parameter vectors $\boldsymbol{a}' \in X'$,is called *"Pareto-optimal front"* or *"Pareto-front"*.

4.2.2. Some of the most popular algorithms

In the last years, different EAs have been suggested to solve multi-objective problems. Daniel Kunkle [11] makes a short review of the most popular MOEA algorithms. The characteristics of some of them are briefly described below.

- **NPGA -** Niched Pareto Genetic Algorithm [12]

NPGA is one of the first algorithms which directly address the diversity of the approximation set.

The main difference between this algorithm and traditional GAs is in the selection mechanism. This modified tournament selection mechanism is called Pareto domination. In addition, it uses fitness sharing in the objective space in order to maintain a diverse population. Nevertheless, some studies [13] demonstrate that NPGA have a worse performance than most of the more recent MOEAs.

- **NPGA II** [14]

NPGA II was introduced by Erickson, Mayer and Horn as an upgrading of the original NPGA [12].

The main improvement of this algorithm over NPGA is the use of the degree of domination (which is the number of solutions in the current population that dominate it) as the determining score in tournament selection. This method is deterministic, in contrast with the probabilistic method used by NPGA, which can result in a "noisier" result. However, NPGA II, like NPGA, is very sensitive to the parameters controlling tournament selection and fitness sharing.

There are not many studies comparing NPGA II with other MOEAs, thus more research should be done before considering this algorithm superior to other MOEAs, even to NPGA.

- **NSGA -** Non-dominated Sorting Genetic Algorithm [15]

This genetic algorithm is very similar to the two algorithms mentioned before. It differs only with the ranking method used in tournament selection. The technique for sorting consists in assigning rank 0 to the current non dominated subset of population and removing it for consideration temporally. The remaining

population is then evaluated to determine another non dominated subset, which is given rank 1 and removed from consideration. This process continues until the entire population has been ranked.

Although NSGA includes some fundamental MOEA components, it is now surpassed by other state-of-the-art algorithms, as happens with NPGA.

- **NSGA II** [16]

This algorithm is significantly different from the original NSGA. It addresses the following drawbacks of NSGA:

- High computational complexity of non-dominated sorting: The non-dominated sorting algorithm from NSGA is $O(MN^3)$, where M is the number of objectives and N is the population size.
- Lack of elitism: Results show that elitism can speed up the performance of the GA significantly and can help prevent the loss of good solutions.
- Need for specifying the sharing parameter σ_{share}: The main problem with fitness sharing is that it requires the specification of the sharing parameter, which can significantly affect performance. A parameterless diversity mechanism solves this problem.

NSGA II is currently one of the most popular MOEAs.

- **PESA** - Pareto Envelope-based Selection Algorithm [17]

The main attraction of PESA is the integration of selection and diversity management into one technique (selection and archiving are based on the diversity maintaining crowding measure). This algorithm includes ideas form SPEA and PAES.

Comparison with other MOEAs demonstrated that PESA was the best in 3 of the 6 functions tested and tied for the best with SPEA in 2.

- **SPEA** - Strength Pareto Evolutionary Algorithm [18]

This algorithm is similar to MOEAs that came before it (such as NPGA) because it stores the Pareto-optimal solutions found so far externally (archiving); it uses Pareto-dominance to assign fitness values to individuals and performs clustering to reduce the number of non dominated solutions stored without

destroying the characteristics of the Pareto-optimal front. On the other hand, it has unique aspects such as the determination of the fitness of an individual, which is only from the archive of non dominated solutions; another characteristic is that all solutions in the archive participate in the selection, and that niching does not require any fitness sharing parameter.

This algorithm has been very successful and many improved algorithms, such as SPEA2, have resulted from it.

- **SPEA2** [19]

SPEA2 improves the fitness assignment scheme, which takes into account, for each individual, how many individuals it dominates and it is dominated by. In addition, a nearest neighbour density estimation technique is incorporated, which allows a more precise guidance of the search process. Besides this, a new archive truncation method guarantees the preservation of boundary solutions.

The main difference between SPEA II and other MOEAs, as SPEA or PESA, is in how it does the fitness assignment and selection.

The performance comparisons done in the original paper conclude that SPEA is much better than SPEA in all respects in all test problems, that it is better than PESA in both diversity and final quality of solutions, although PESA converged faster on some test problem. In the comparison with NSGA-II, SPEA2 has a better distribution but NSGA-II had a broader range of solutions, which means that it found solutions closer to the outlying edges of the Pareto-optimal front.

- **PAES** - Pareto Archived Evolution Strategy [20]

It is argued that PAES represents the simplest possible nontrivial algorithm capable of generating diverse solutions in the Pareto-optimal set. It has three forms, (1+1)-PAES, (1+λ)-PAES, and (μ+λ)-PAES. The notation refers to: (population size + number of new solutions per generation). While (1+1)-PAES and (1+ λ)-PAES perform only local search, i.e. they keep only one current solution, instead of a population of searches, the (μ +λ) version maintains a population.

PAES introduces a new crowding procedure, the adaptive grid algorithm. It is superior to previous niching methods in two ways: (1) Its computational cost is lower; (2) It is adaptive and does not require the critical setting of a niche-size parameter.

The comparable performance of this simple algorithm to other more complex ones is attributed to the use of an archive of non dominated solutions, which was uncommon at the time but has been incorporated into most state-of-the-art algorithms.

4.3. The NSGA-II Algorithm

```
NSGA-II() {
   generate P_0 at random.
   set P_0 = (F_1, F_2, ...) =non-dominated-sort(P_0)
   for all F_i ∈ P_0
      crowding-distance-assignment(F_i)
   set t = 0
   while(not done) {
      generate child population Q_t from P_t
      set R_t = P_t ∪ Q_t
      set F = (F_1, F_2, ...) =non-dominated-sort(R_t)
      set P_{t+1} = ∅
      set i = 1
      while |P_{t+1}| + |F_i| < N {
         crowding-distance-assignment(F_i)
         set P_{t+1} = P_{t+1} ∪ F_i
         set i = i + 1
      }
      sort F_i on crowding distances
      set P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)]
      set t = t + 1
   }
   return F_1
}
```

Figure 9. NSGA-II Algorithm [21]

Pseudo code for the NSGA-II is given in Figure 9. The algorithm is based on the idea of transforming the *M* objectives to a single fitness measure by the creation of a number of fronts, sorted according to non domination. During fitness assignment, the first front F_1 is created as the set of solutions not dominated by any solutions in the population. These solutions are given the highest fitness and are temporarily removed from the population. After this, a second non dominated front F_2 consisting of the solutions that are now non dominated is built, assigned the second-highest fitness, etc. This is repeated until all of the solutions have been assigned a fitness. After each front has been created, its members are assigned crowding distances (normalized distance to closest neighbors in the front in objective space) later to be used for niching. Selection is performed in tournaments of size two: The solution with the lowest front number wins. If

the solutions come from the same front, the solution with the highest crowding distance wins, since a high distance to the closest neighbors indicates that the solution is located at a sparsely populated part of the front. Reproduction occurs in generations. In each generation N new individuals are generated, where N is the population size. Of the $2N$ individuals, the best N individuals are kept for the next generation. In this way, a huge elite can be kept from generation to generation. The processing time used by the non-dominated-sort procedure as suggested in [16] is $O(MN^2)$, since the procedure involves comparing the objective values of every solution to the objective values of all other solutions in the population. The time used by the crowding-distance-assignment procedure is $O(MNlogN)$, since the procedure involves sorting the elements in fronts F_i one time for every objective, and since the front size is only limited by N. The time used by the rest of the steps in the algorithm can be expected to be in $O(N)$, and the overall running time of the algorithm is dominated by non dominated sort, and runs in time $O(GMN^2)$, where G is the number of generations [21].

5. Simulation Study

5.1. Introduction

This chapter is devoted to describe in detail the main elements of the BPLC system and scheduling algorithm that set up the framework of this thesis wherein several strategies for initializing the algorithm will be tested. These different strategies are described in detail in section 5.2. The description of a Uniform Resource Distribution scheduling technique will be made in 5.3, which will be compared with the main algorithm. On the other hand, section 5.3 is devoted to the definition of the performance parameters that will be taken into account when assessing the benefits of the proposed selection mechanisms. These results are presented in CHAPTER 6.

5.2. The proposed NSGA-II based scheduling algorithm

The algorithm used in this Thesis is based on the NSGA-II algorithm proposed and implemented by Kalyanmoy Deb in [22]. Several modifications have been done in order to adapt it to solve our problem. Besides this, some different strategies have been implemented and tested in order to improve the speed of the algorithm. In the following sections a detailed description of these strategies is given.

5.2.1. Main description

In previous chapters we have seen how the requests from different services for the users are expressed in a *Y* matrix, wherein the values in the rows are the requests of different services for each user, and the columns, the values of certain type of service for all the users. In the same way, an allocation matrix has been defined, where the values of a certain user for a certain service cannot be higher than the corresponding value in the *Y* matrix. As we can see in Figure 10, in the reality, the value of *Y* matrix would be updated with all the new requests each time slot. Once this value is updated, the new *X* matrix could be calculated in order to make the new allocation of resources. Then, once the new resources are allocated, new requests come and the *Y* matrix has to be updated at the beginning of the following time slot and the process is repeated. On the other hand, in the simulation, there are no real requests, thus the *Y* matrix has to be created, then, *X*

matrix is calculated, and almost at the same time, the *Y* matrix is updated, as we don't have real requests to wait for.

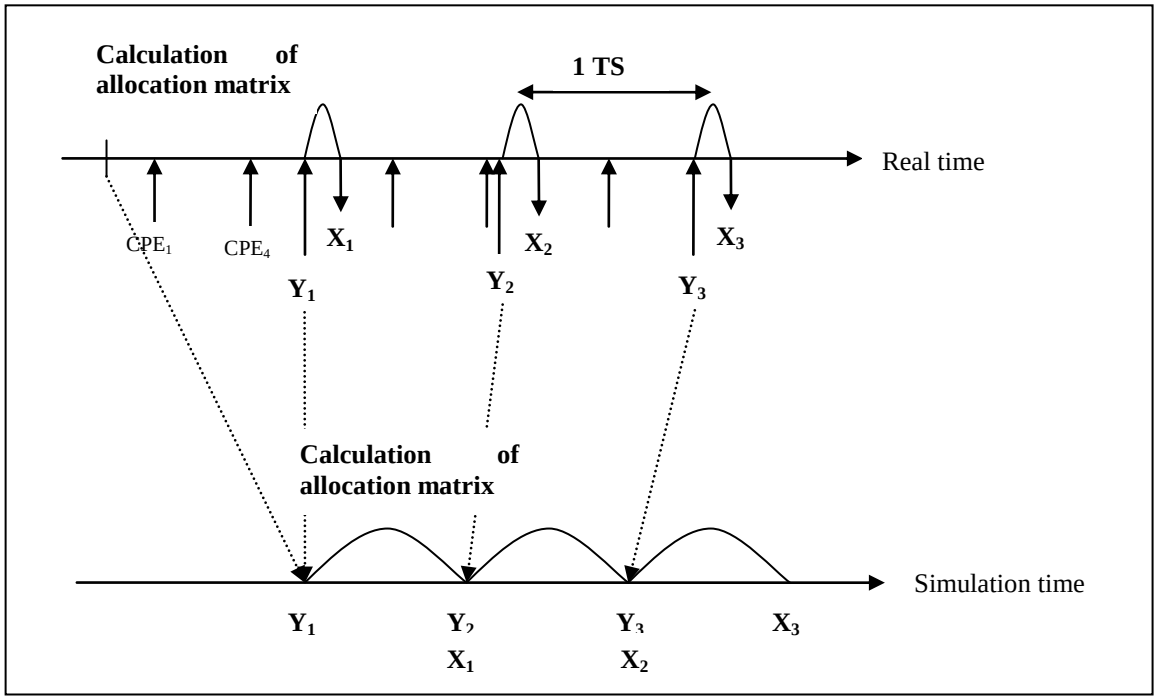

Figure 10. System modelling

In Figure 11, a pseudo-code of the algorithm can be seen. A more detailed description of the routines implemented to make the different initializations for the different strategies tested is given in Appendix.

```
/*NSGA-II based scheduling algorithm*/



create_input_file();  /*Creates a file containing the parameters of the NSGA-II algorithm: Population size, Number of generations, Crossover probability, Mutation probability*/

create_output_files(); /*Creates the files where the results will be printed*/

initialize_Y(); /*Creates the request matrix, Y, where each value of the matrix is an uniform distribution between 0 and a maximum number of requests*/

MAIN LOOP:

        initialize_algorithm_parameters();

        update_Y(); /*The Y matrix is updated*/

        read_input_file(); /*The data from the input file is read ans stored in the program*/

        initialize_population(); /*The individuals are initialized*/

        decode(); /*Decode the chromosomes to find the integer values of the population*/

        function_evaluation(); /*Makes fitness evaluation*/

        GENERATION LOOP:
                selection();
                crossover();
                mutation();
                decode();
                function_evaluation();
                selection();
                elitism();
                decode();
                rank_ratio_calculation();
                copy_new_to_old_population(); /*We copy the results for the next generation*/
                FOR THE LAST GENERATION:
                        FOR ALL FEASIBLE AND NON-DOMINATED SOLUTIONS:
                                find_max_benefit_solution(); /*Among all the pareto solutions, we choose the one with maximum benefit value*/
                                print_pareto_results();

        END OF GENERATION LOOP;

        parcial_results_calculation();

END OF MAIN LOOP;

final_results_calculation();
```

Figure 11. NSGA-II based scheduler algorithm pseudo-code

5.2.2. No Adaptation Strategy

The Figure 12 shows us the block diagram for this strategy. The information about the fitness functions to maximize (benefit and fairness) and the constraints are given to the NSGA-II algorithm. The specific parameters to make it run, such as the population size, the number of generations, the crossover probability and the mutation probability, are also given. These parameters are static; they don't change every time the scheduler makes a change in its allocation table. *Y* matrix is the input that will be updated in each time slot (in the case of the simulation, in each iteration of the main loop, here denoted by *k*), and that defines the constraints related to this matrix. On the

other hand, we have the allocation matrix X as the output of the algorithm, and once it has been found, the method goes on, a new Y matrix is processed, and so on.

In this strategy, the initialization of the NSGA-II algorithm is random. This means that for all the iterations of the main loop, the individuals are randomly created, and the results from the previous allocations are not taken into account.

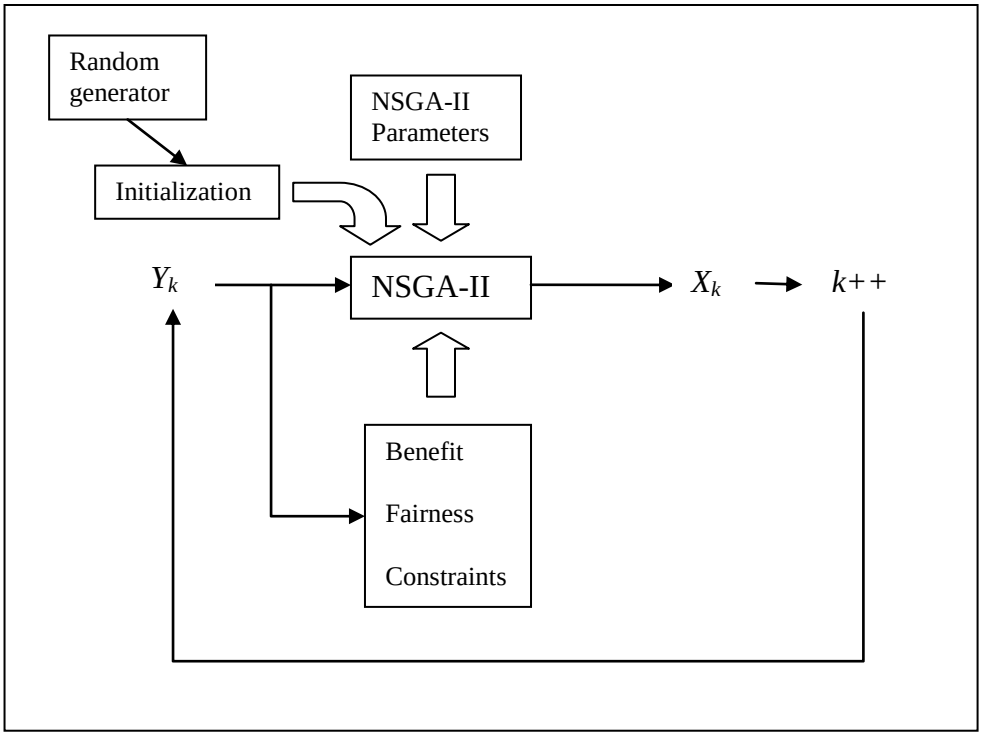

Figure 12. No Adaptation strategy diagram

The routine implemented to make this initialization of the algorithm is the one originally employed by Deb in [22]. The implementation of this function can be seen in Appendix.

5.2.3. Adaptation Strategy 1

In Figure 13 a diagram of this strategy is shown. The method is the same as the previous, except for the process for initializing the NSGA-II algorithm. With this strategy, only the first time the algorithm is initialized will be by generating a random population. Once an allocation solution has been found, the matching values of the X matrix will be stored and coded to use them in the following iteration of the main loop. The differences with the no adaptation strategy will be tested in the simulation results.

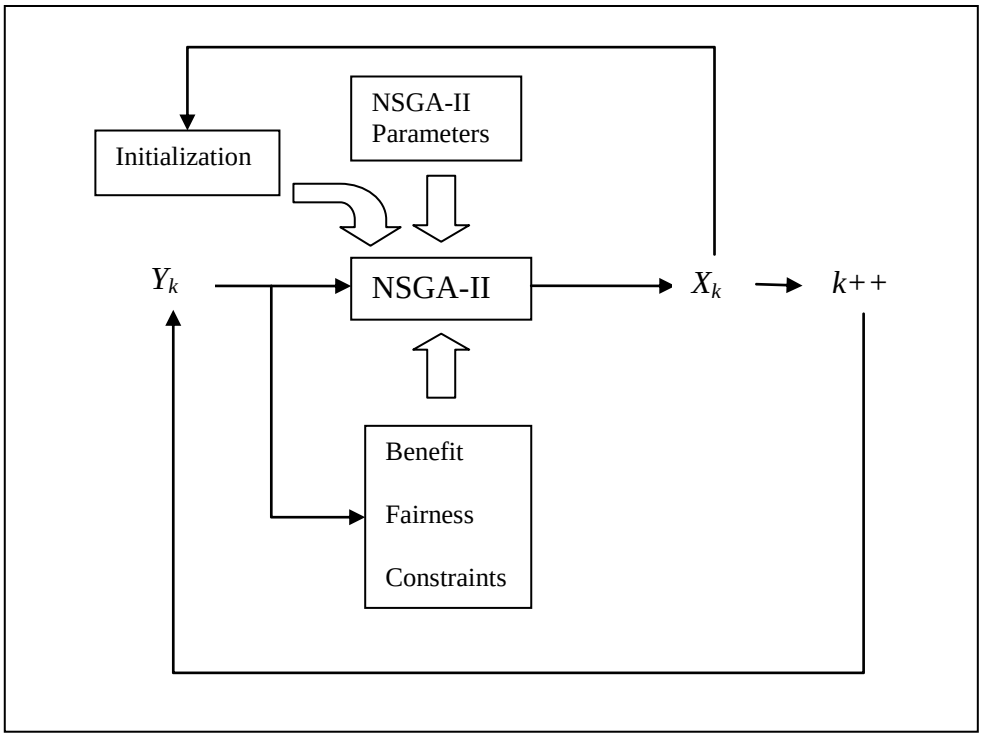

Figure 13. Adaptation strategy 1 diagram

5.2.4. Adaptation Strategy 2

The main procedure of this strategy is the same as in the two previous sections. The difference, again, is the initializing method. In this case, the results from the previous allocation matrix are used, but some random information is also added to the chromosomes of the initial population. Two different strategies have been used to add this random information. In the first one, the genes are toggled, one with the value from the previous allocation vector, one with a random value. This strategy will be called *Adaptation Strategy 2a*. The second strategy to add randomness to the initialization (*Adaptation Strategy 2b*) is to toggle, not the genes but the individuals in the population, so one individual codes the genes values from the previous results, and the following codes the genes randomly, putting a 0 or a 1, with 50% probability. The following diagram shows the procedure of this second adaptation strategy.

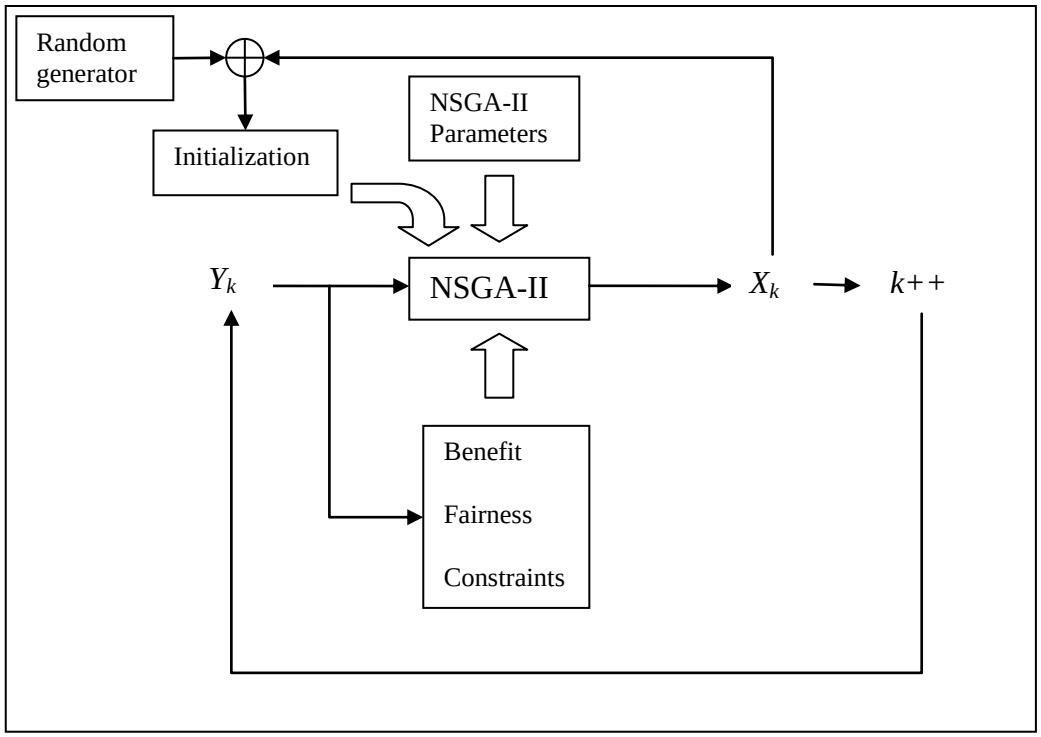

Figure 14. Adaptation strategy 2 diagram

5.3. Uniform Resource Distribution

The simulation results of the proposed NSGA-II based algorithm will be compared with a simple scheduling strategy, which is the Uniform Resource Distribution. The Figure 15 shows the pseudo-code implemented for this scheduler.

The proceeding for allocating the resources is as follows: The available resources are divided into k equal parts, assuming that *N* is the number of users. Then, for each user, starting for the highest benefit service class, the needed resources are allocated, until there is no more available bandwidth or no further requirement. This algorithm is very fast, but it can be unfair, because higher priority connections are served first, and this fact can cause the starving of the lowest priority data transmissions. In addition, with this method, the benefit achieved might not be a maximum.

```
/*Uniform Resource Distribution scheduling algorithm*/



create_output_files(); /*Creates the files where the results will be printed*/

initialize_Y(); /*Creates the request matrix, Y, where each value of the matrix is an uniform
distribution between 0 and a maximum number of requests*/
calculate_user_allocation(); /*The total resource available is divided among all the users*/

MAIN LOOP:

        initialize_user_allocation_matrix();  /*The  allocation  matrix  for  each  user  is
initialized with the user allocation value calculated before*/

        update_Y(); /*The Y matrix is updated*/

        uniform_resource_allocation(); /*The requests from Y are allocated, starting with the
highest benefit services for each user*/

        parcial_results_calculation();

END OF MAIN LOOP;

final_results_calculation();
```

Figure 15. Pseudo-code for the Uniform Resource Distribution scheduler

5.4. Performance Metrics

5.4.1. Benefit

The figure below shows the function that implements the benefit. This function is used as one of the fitness functions that have to be maximized.

```
      /*Calculation  of  the  total  benefit  for  all  users  and  all
services for a given X matrix*/

      float total_benefit(int x[N*S])
      {
            int i,j;
            float result = 0;
            for(i=0;i<N;i++)
            {
                for(j=0;j<S;j++)
                {
                    result += x[i*S+j]*b[j];
                }
            }
            result = result/N;
            return(result);
      }
```

Figure 16. Routine that implements the Total Benefit

The function implemented is the one based on the definition given in 3.2.1.

$$\boldsymbol{B} = \frac{\mathbf{1}}{\boldsymbol{N}} \cdot \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{B}_{Ui}$$

This definition, which is applicable for a network with *N* users each one of them with the same weight (*1/N*), has the same structure as the arithmetic mean of the benefit achieved by the system.

The definition of arithmetic mean is the sum of all the members of a list, divided by the number of items in that list. If we have a set of data $Z = \{z_1,z_2,z_3,...,z_n\}$, the arithmetic mean is calculated as follows:

$$\bar{z} = \frac{1}{N}\sum_{i=1}^{N} z_i$$

This definition is also valid when *Z* is a set of random numbers. The value obtained will be the probabilistic mean or expected value of the random number. The benefit achieved by each user B_{Ui} is a random number, as it depends on the obtained values of *X* matrix. Consequently, the benefit achieved by the system with the allocation provided by the scheduling algorithm, gives the expected value of benefit for a user.

5.4.2. Fairness

The code implemented to define the fairness, which is the second fitness function to evaluate in the NSGA-II algorithm, is showed in Figure 17. The routines used in this function are also shown in Figure 18 and Figure 19.

```
      /*Calculation of the total fairness for all users for a given X
matrix*/

      float average_fairness(int x [N*S])
      {
            float result = 0;
            int i;
            for(i=0;i<N;i++) {result += get_p(i,x)*selffairness(i,x);}
            return(result);
      }
```

Figure 17. Routine that implements the Average Fairness

```
      /*Calculation of the self-fairness for a given user for a given X
matrix*/

      float selffairness(int user,int x [N*S])
      {
            float result = 0;
            float aux;
            aux = get_p(user,x);
            if(aux==0) return(0);
            result = - log(aux) / log(N);
            return(result);
      }
```

Figure 18. Function that implements the Self Fairness

```
      /*Calculation of the proportion of resource allocated for a user
for a given X matrix*/

      float get_p(int user,int x[N*S])
      {
            float num = 0;
            float den = 0;
            int j,k;
            float result;
            for(j=0;j<S;j++) {num += R[j]*x[user*S+j];}
            den = allocated_resource(x);
            result = num/den;
            return(result);
      }
```

Figure 19. Function that implements the Proportion of Resource Allocated

As it has been seen in 3.2.2, the average fairness gives a value between 0 and 1. The value 0 is returned when only one user consumes all the resources available and the others starve. On the other hand, the highest value will be achieved when all the users have their fair proportion of allocated resource, which in the case of equal weighted users is *1/N*.

5.4.3. Execution Time

The code used to make the calculation of the execution time is showed in Figure 20.

```
/* returns "a - b" in seconds */
double timeval_diff(struct timeval *a, struct timeval *b)
{
  return
    (double)(a->tv_sec + (double)a->tv_usec/1000000) –
    (double)(b->tv_sec + (double)b->tv_usec/1000000);
}

/*Main program*/
int main()
{
  struct timeval t_ini, t_fin;
  double msecs;

  MAIN LOOP:
        /*More code*/
        gettimeofday(&t_ini, NULL);
        /* ...the algorithm... */
        gettimeofday(&t_fin, NULL);
        msecs = timeval_diff(&t_fin, &t_ini) * 1000.0;
        /*More code*/
  END OF MAIN LOOP;
/*More code*/
}
```

Figure 20. Implementation and application of the execution time calculation

The method *timeval_diff* for time calculation is only working with Linux O.S. As it can be seen, the *timeval* structure has two fields, elapsed seconds and microseconds (*tv_sec* and *tv_usec* respectively), thus this function offers an accuracy of microseconds. The routine *gettimeofday* belongs to the standard C library and gives the system's notion of the current Greenwich Time and the current time zone. The time is expressed in seconds and microseconds since midnight (0 hour), January 1, 1970. The resolution of the system clock is hardware dependent, and the time may be updated continuously or in "ticks".

The calculation of the execution time will be done for all the iterations of the main loop, i.e. it will be calculated each time a reschedule is done.

The execution time is very important in order to decide whether the chosen algorithm is suitable or not for our purpose, because we don't only look for a maximum benefit and fairness, but also a fast enough scheduler for a real time system.

5.4.4. Not Guaranteed Connections

Another useful metric to evaluate the efficiency of the selected algorithm is the percentage of not guaranteed connections. We say that a connection is no guaranteed when a request for a certain type of service has been made, but there are no resources guaranteed for it. The following figure shows the code to implement its calculation.

```
/*Calculation of the not guaranteed connections, given the request
matrix Y and the allocated matrix X*/
void calculate_not_guaranteed_conn(int Y[N][S], int X[N*S])
{
	int i,j,drops;
	for(i=0;i<N;i++)
	{
		for(j=0;j<S;j++)
		{
			if(Y[i][j] > X[i*S+j])
			{
				drops = Y[i][j] - X[i*S+j];
				not_guaranteed_conn += drops;
			}
		}
	}
}
int main()
{
  /*Main program code*/
  MAIN LOOP:
  /*More code*/
	pp = (not_guaranteed_conn * 100 )/ total_requests;
  /*More code*/
  END OF MAIN LOOP;
  /*More code*/
}
```

Figure 21. Implementation of the method for calculating the not guaranteed connections

When a request of a certain type of service from a user is not allocated in the granted matrix *X*, it goes to the not guaranteed queue, where the requests are treated in a BE basis. This means that when a request goes to this queue, the resources needed to do the transmission are not guaranteed, and it will obtain unspecified variable bit rate and delivery time, depending on the current traffic load. In fact, when a request from a hard real time connection, for example, from a VoIP connection, cannot be guaranteed, the connection should be dropped instead of queued, as it does not only require a specific bandwidth, but also a minimum tolerable delay. Nevertheless, we will assume that all connections that are not allocated in X matrix will be put in the BE queue, even if they are hard real time services.

6. Simulation Results

6.1. Introduction

In this chapter, the results obtained with several strategies in two different scenarios will be analyzed and compared with the results of a Uniform Resource Distribution. The performance metrics described in CHAPTER 5 will be under study, and conclusions will be extracted and exposed in CHAPTER 7.

6.2. Scenario 1

6.2.1. Description

As it has been explained before, the algorithm is applied to a network with N users that have the same weight ($1/N$). We have assumed that the resource allocated for each service is the same for all the users. Thus, we have four services classes, *S1*, *S2*, *S3* and *S4*, which require 1, 2, 1 and 0.01 normalized resource unit and achieve 8, 4, 2 and 1 units of benefit, respectively.

In this scenario, the number of requests is uniformly distributed between 0 and 2. This means that each user can requests a maximum of two sessions for the same service at an instant of time. To make the updating of the Y matrix each time slot, as it is very difficult to make a model of the behaviour of the requests for each service class, it will assumed that a change in the request matrix will occur with probability of $2 \cdot 10^{-5}$. This probability of changing a request for each time slot is equivalent to say that there is approximately one change in the requests for each user and for each type of service in one hour. This behaviour can be the model of, for example, a phone call. Obviously this is not a realistic modelling for other types of traffic, like http or mail traffic, but we will assume that they act in the same way.

To choose the NSGA-II parameters to run the simulations, the rule used has been to assure that the algorithm is apt for a real time system. The computational complexity of the NSGA-II is $O(GMN^2)$ [21], where G is the number of generations, M the number of objective functions and N the population size. The number of objectives is fixed by the problem, so the number of generations and the population size will have to be chosen in order to make as short as possible the execution time, and also assuring a good enough

result. In addition, several values of the mutation probability and cross-over probability have been tested in order to increase the quality of the obtained results. The obtained values for the execution time have been obtained running the simulations in a computer with a CPU running at 2 GHz and with a RAM of 1 Gb.

To select these parameters, a network with 128 users has been chosen, and the AS1 strategy has been used as the scheduling technique. The used parameters to do the simulations are shown in the following table.

Number of Generations	6
Population size	4
Mutation probability	0.01
Cross-over probability	0.9

Table 3. NSGA-II parameters for Scenario 1

This scenario will be studied, changing the number of users and maximum allocation, as showed in the table below. The simulations will be running during 1000 time slots; that means that 1000 new allocations will be calculated.

Number of Users	4	8	16	32	64	128
Maximum Allocation	8	16	32	64	128	256

Table 4. Users and Maximum Resource Allocation for 2 maximum requests

6.2.2. No Adaptation Strategy

The benefit, fairness, execution time and not guaranteed connections in Scenario 1 using the NSGA-II algorithm with No Adaptation Strategy (NAS), compared with the benefit achieved by the Uniform Resource Distribution (URD), are studied in the six different network configurations explained above.

As it has been explained in 5.2.2, this strategy consists in applying the NSGA-II algorithm each time a reschedule has to be done, without taking into account the allocation obtained in the previous TS. Thus, each allocation is independent from the previous and the next one.

The obtained results are showed then.

a) N=128, R_{MAX}=256:

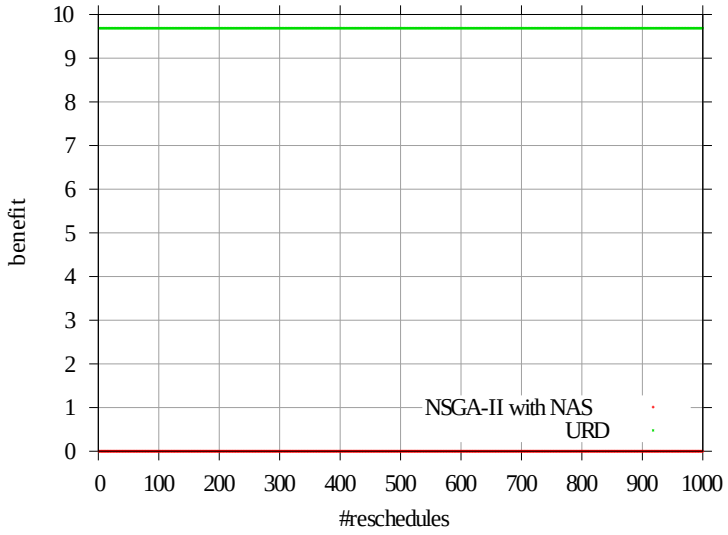

Figure 22. Benefit in Scenario 1 – NAS and URD (N=128, R_{MAX}=256)

With the NSGA-II parameters selected and this network configuration, NAS is not able to find a solution. For all the schedule iterations the benefit achieved is zero. Logically, if the algorithm is not able to find a solution for the benefit, it is neither able to find it for the fairness, as these two functions are the fitness functions used by NSGA-II.

As NAS cannot find a solution to allocate the resources, all the requests are sent to the not guaranteed queue, thus, in this case, we can say that the 100% of the requests are not guaranteed. They will be served in a best effort manner. With URD, the percentage of not guaranteed connections is 42.00%.

The average execution time needed to run the NSGA-II with this strategy is 55.45 ms, in contrast with the 22.27 μs needed to run the URD algorithm. Thus, NAS is fast enough to work in a real time system (given that the premise to select the NSGA-II parameters was to assure an execution time lower than 1 TS), but it clearly doesn't achieve neither an optimal benefit nor a fair system allocation.

b) N=64; R_{MAX}=128:

When the number of users decreases, the algorithm is able to find some solutions (in this case only the 0.90% of the solutions are found), and, as it can be seen in Figure 23, all these solutions get a lower benefit than the ones obtained by the URD algorithm.

The same happens with fairness, the few solutions found give a fairness lower than the 0.95 obtained by the URD allocation.

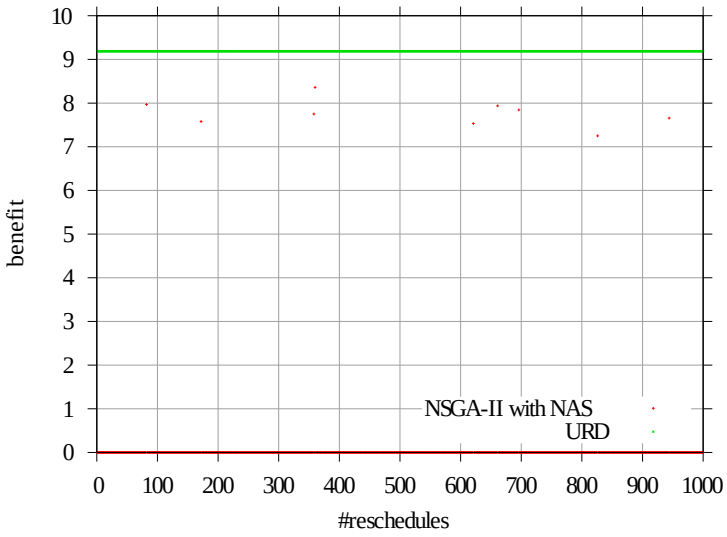

Figure 23. Benefit in Scenario 1 – NAS and URD (N=64, R_{MAX}=128)

In this case, at least the 99.10% of the connections are not guaranteed. This is because when a solution is not found, all the requests in that time slot go to the not guaranteed queue, and this situation happens in 991 of the 1000 reschedules. However, it has to be taken into account that when a solution is found, not all the connections have to be guaranteed, some can also be rejected because they don't accomplish the constraints. This fact is in contrast with the 61.78% guaranteed connections with URD.

The execution time with NAS and 64 users is 18.04 ms. This value is lower than in the last case, as the number of users is also lower, and consequently, the number of variables of the problem. However, the execution time for URD is 14.09 µs, more than 1000 times faster than NAS algorithm.

c) N=32; R_{MAX}=64:

Figure 24 and Figure 25 show the benefit and fairness values, respectively, obtained with NAS and URD for this configuration. It can be observed that the algorithm is able to find more solutions than in the lasts configurations, but it still only finds a 2.7% of the solutions, making an average benefit of 0.98 and an average fairness of 0.11. It is in contrast with the average benefit achieved by the URD which is 8.69, and the average fairness of 0.94.

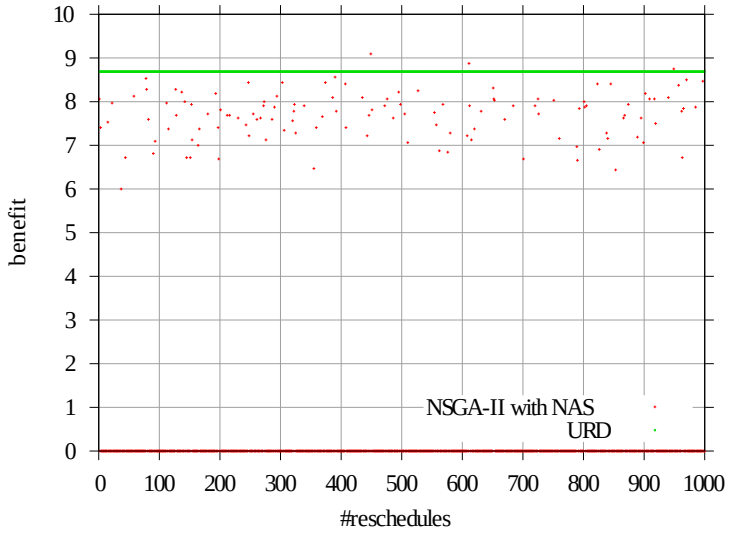

Figure 24. Benefit in Scenario 1 – NAS and URD (N=32, R_{MAX}=64)

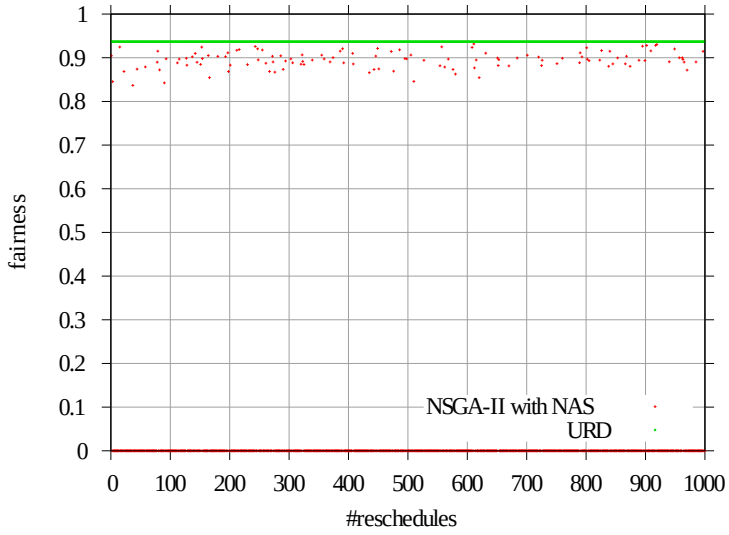

Figure 25. Fairness in Scenario 1 – NAS and URD (N=32, R_{MAX}=64)

As only the 12.7% of the solutions are found, more than the 87.3% of the connections will not be guaranteed. With URD the 60.56% of the connections are guaranteed.

The execution time for this configuration for NAS is 6.53 ms, again lower than for the previous configuration due to the lower number of variables implied in the process. The average execution time for the URD is 8.24 µs.

d) N=16; R_{MAX}=32:

When the number of users decreases to 16, Figure 26 shows that some more solutions are found and that some of them achieve a higher benefit value than the 8.13

value for the URD. Specifically, the 74.7% of the solutions are found. The fairness results shown in Figure 27 are analogue.

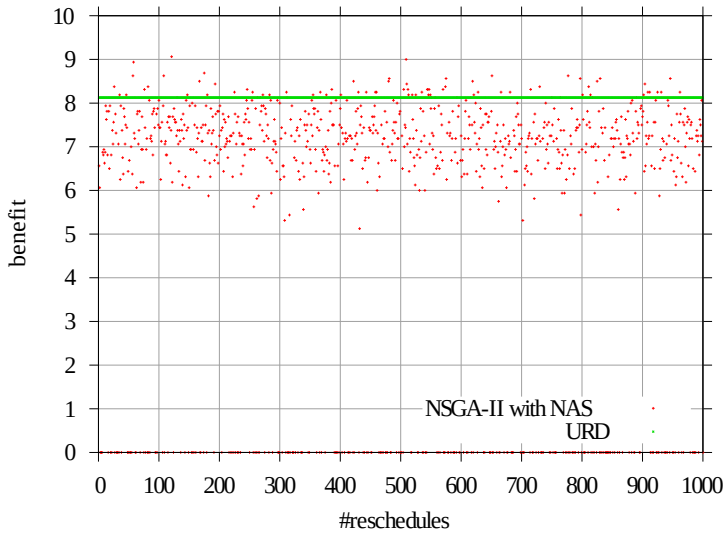

Figure 26. Benefit in Scenario 1 – NAS and URD (N=16, R_{MAX}=32)

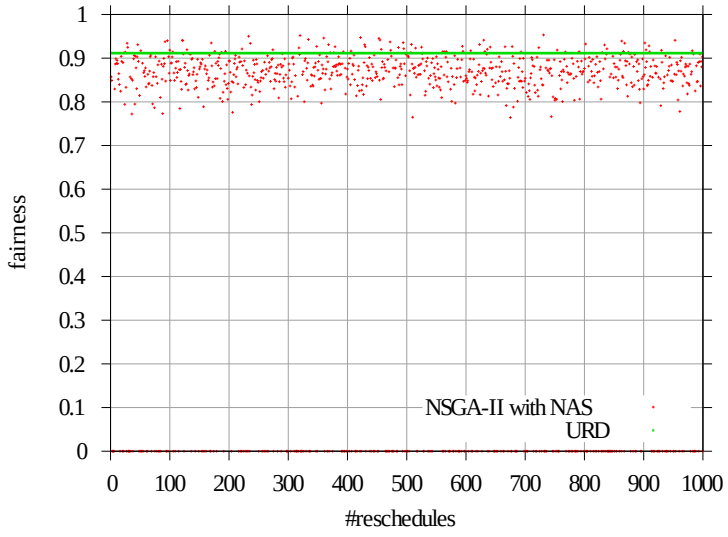

Figure 27. Fairness in Scenario 1 – NAS and URD (N=16, R_{MAX}=32)

The average execution time for NAS is 2,71 ms, while for URD is 5,11µs.

e) N=8; R_{MAX}=16:

The figures below show us, again, the lack of efficiency of the NAS algorithm to find a good solution for the allocation of resources. Although in almost all the cases (96.60%) a solution is found, this is far from being an optimal solution, as the URD is able to find better solutions in almost all the reschedule iterations. The average benefit

achieved by NAS is 8.13 and the average fairness is 0.80, while these values are 10.5 and 0.92, respectively, for the URD.

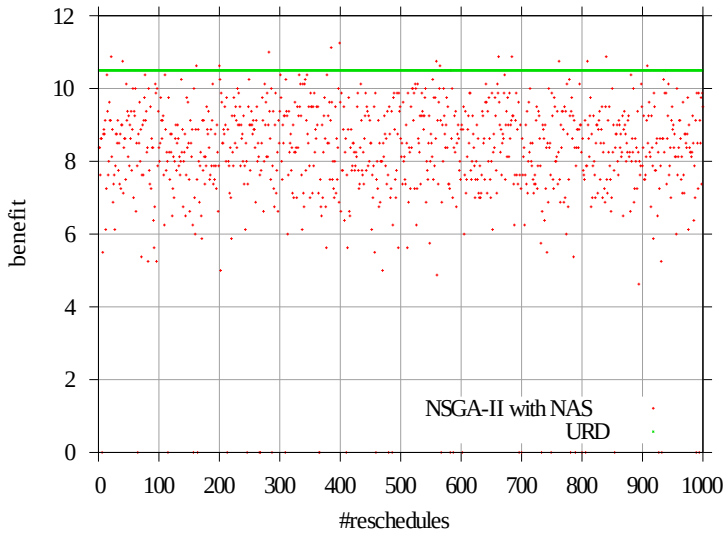

Figure 28. Benefit in Scenario 1 – NAS and URD (N=8, R_{MAX}=16)

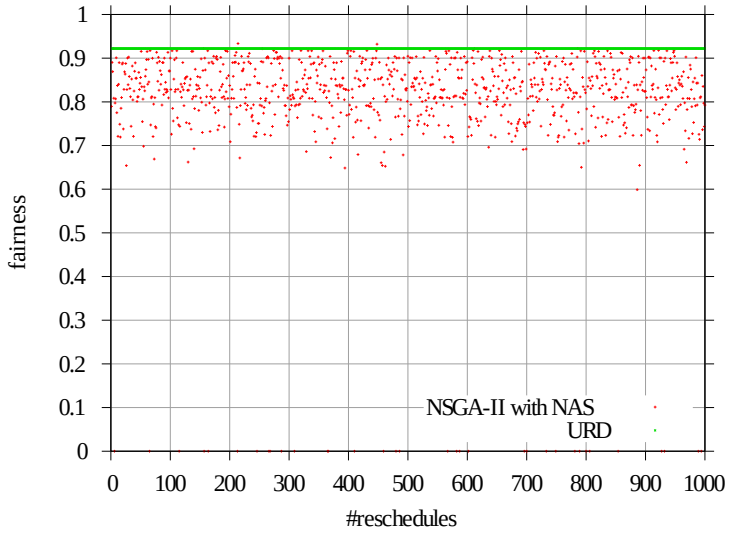

Figure 29. Fairness in Scenario 1 – NAS and URD (N=8, RMAX=16)

The execution time for NAS with this network configuration is 1.43 ms. The execution time for URD is 3.63 µs.

f) N=4; R_{MAX}=8:

Figure 30 and Figure 31 show the benefit and fairness achieved in a network with 4 users, having a maximum allocation of 8 normalized units. The average values calculated are 5.29 and 0.55 for the benefit and fairness, respectively. As in the other cases, NAS does not find all the solutions, and the solutions found are worse than the

ones obtained by URD. Specifically, it is not able to find more than the 68.00% of the solutions, thus, at least, this percentage of requests cannot be guaranteed. In addition, the range of solutions found is very wide, so NAS is not able to find a stable allocation for the users.

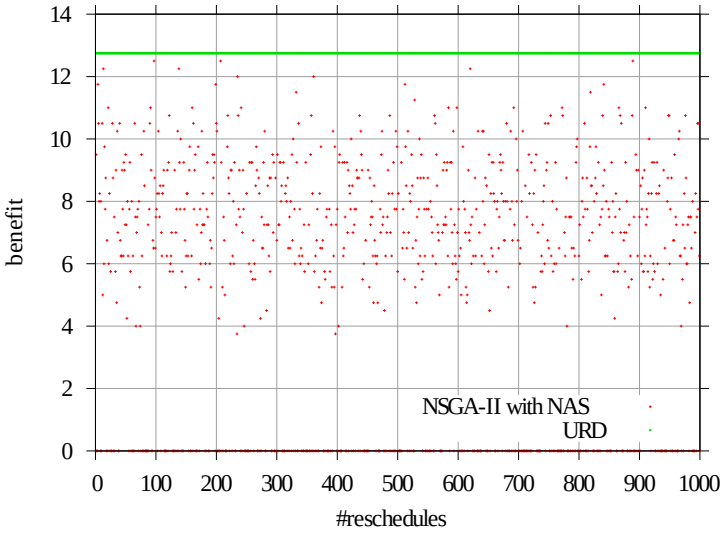

Figure 30. Benefit in Scenario 1 – NAS and URD (N=4, R_{MAX}=8)

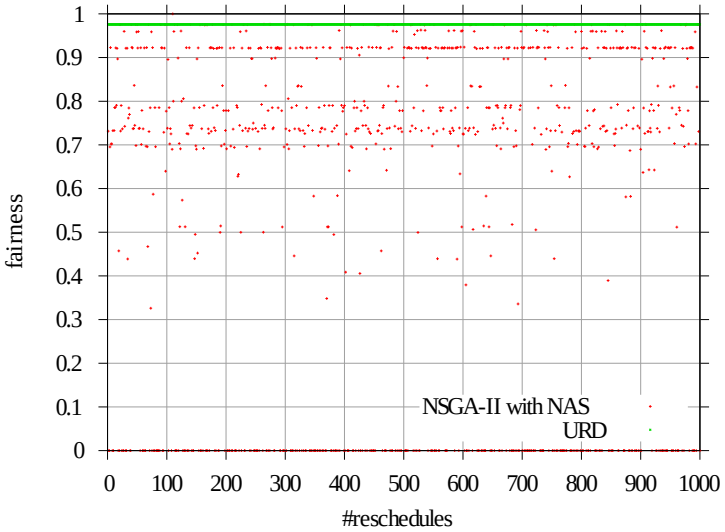

Figure 31. Fairness in Scenario 1 – NAS and URD (N=4, R_{MAX}=8)

The above results show that, with the selected parameters to run the NSGA-II algorithm, the NAS strategy is not a good strategy to solve our problem, as it is not able to find, not only the highest benefit solution, but also no solution when the number of users is high. URD performs a better allocation in all cases, which means that achieves higher benefit, higher fairness and lower execution time.

6.2.3. Adaptation Strategy 1

The Adaptation Strategy 1 (AS1), as it has been explained before, consists in using the result obtained by the previous allocations to initialize the NSGA-II algorithm. In this strategy, the initial individuals in one TS are exactly the same that the individuals of the solution obtained in the previous one. Only the first schedule initialization is done randomly. The results are now exposed.

a) N=128, R_{MAX}=256:

The figure below shows the benefit obtained with AS1 compared with URD for a network with 128 users supporting a maximum allocation of 256 units. For the 5 firsts iterations, the algorithm is not able to find a solution. Nevertheless, as the initialization takes into account the previous results, in the 6th it is possible to find a solution and begin with the allocation of resources. It can be seen that for the 100th first TS, the AS1 increases quite fast, and around the 60th iteration overcomes the benefit achieved by URD. Then, the benefit value increases slowly until it reaches what seems to be the optimum value, which is 10.73.

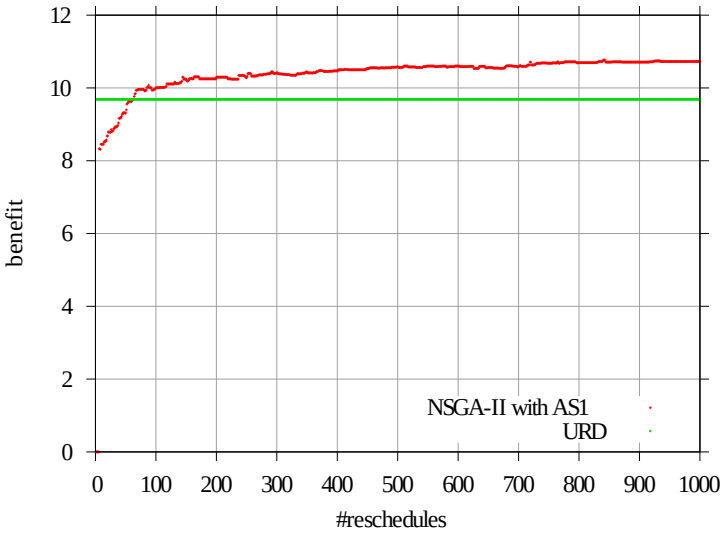

Figure 32. Benefit in Scenario 1 - AS1 and URD (N=128, R_{MAX}=256)

Figure 33 shows the fairness achieved. The results are similar to the obtained for the benefit. It can also be seen that for the 100th firsts iterations, the fairness increases faster than for the rest of the TS, where the increasing is slower, until it reaches a constant value of 0.98.

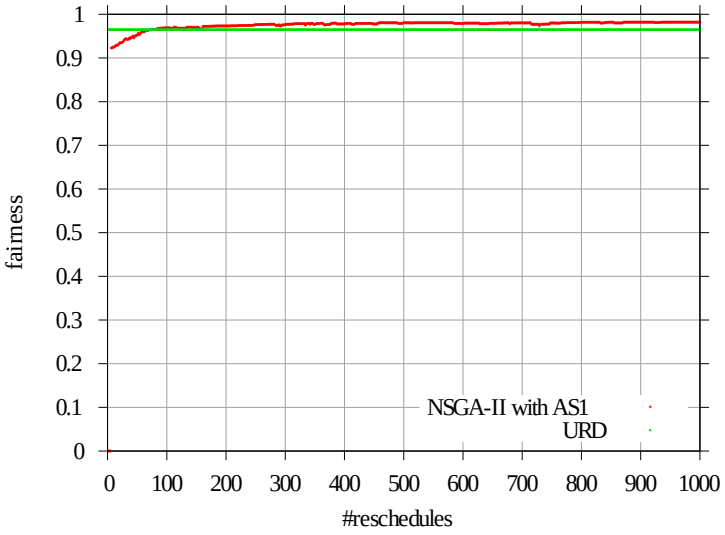

Figure 33. Fairness in Scenario 1 - AS1 and URD (N=128, R_{MAX}=256)

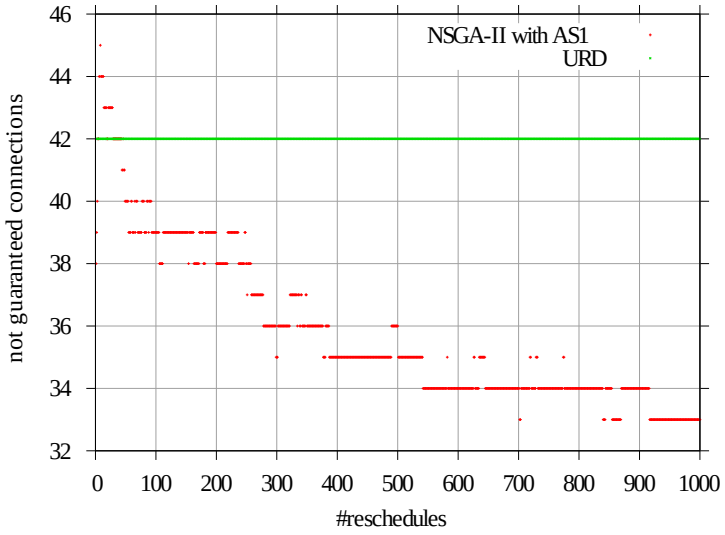

Figure 34. Not Guaranteed Connections (%) in Scenario 1 - AS1 and URD (N=128, R_{MAX}=256)

Figure 34 shows the evolution of the not guaranteed connections along the simulation time. While the value for the URD is stuck in 42.00%, NAS1, making different allocations, achieves to decrease the rate of the not guaranteed connections achieving a value of 33.00%, which means that only a third of the connections cannot be guaranteed.

The average time needed to find an allocation with AS1 is 55.60 ms. The parameters for the simulation were chosen to assure an execution time lower than 60 ms (1TS), so we can conclude saying that this strategy, for 128 users and a maximum resource allocation of 256 normalized units configured for a real time system, performs better than URD, although this last is faster.

b) N=64, R_{MAX}=128:

Figure 35 shows the benefit achieved with this configuration. In this case, the first 3 schedules don't have an allocation solution, but in the 4th the algorithm is able to find it. For the first 150th TS the solutions achieve a fast increasing benefit, and then, it can be observed that the benefit increases very slowly and achieves a value of 10.37. From the 20th TS, the allocation with AS1 overcomes the benefit obtained by URD.

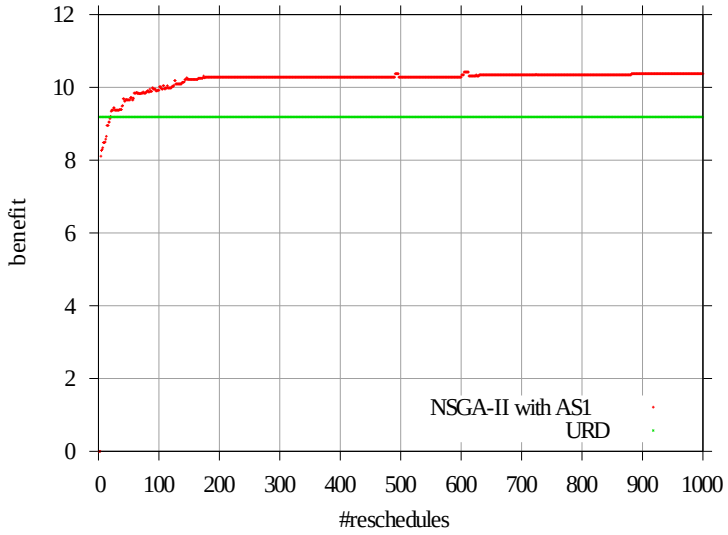

Figure 35. Benefit in Scenario 1 - AS1 and URD (N=64, R_{MAX}=128)

In the figure below, the fairness for this configuration shows similar characteristics to the benefit graphic showed above. We can also see that the convergence of the fairness is faster than the convergence for the benefit, but it gets stuck before in a stable value, which is 0.98.

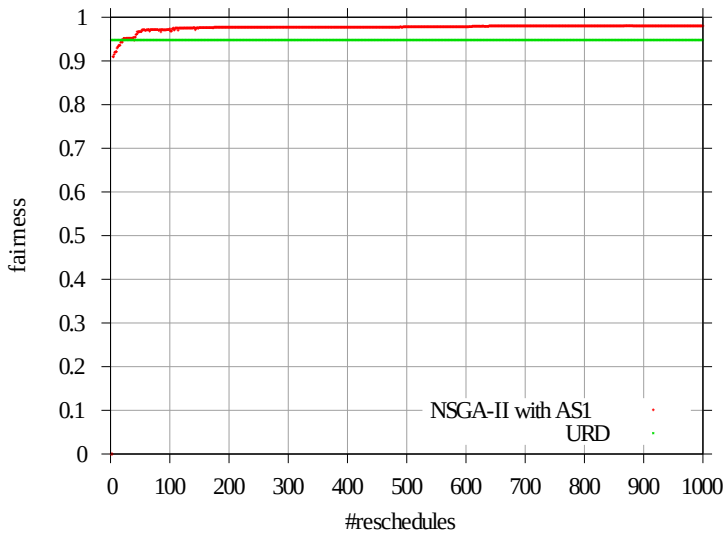

Figure 36. Fairness in Scenario 1 - AS1 and URD (N=64, R_{MAX}=128)

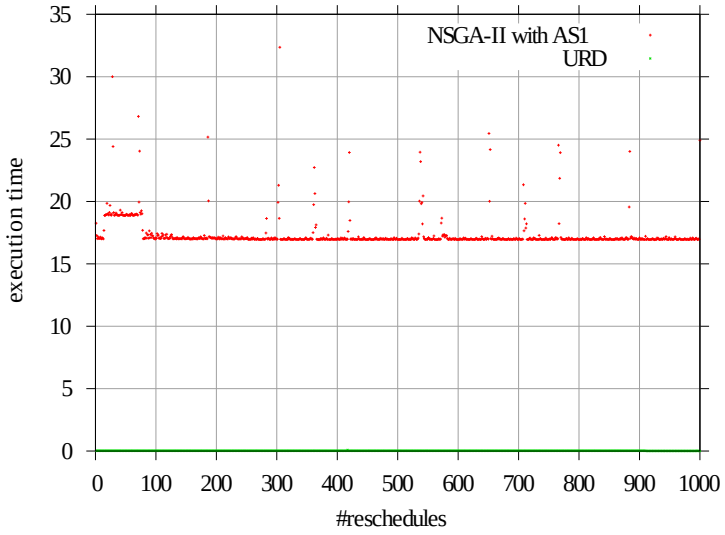

Figure 37. Execution Time (ms) in Scenario 1 - AS1 and URD (N=64, R_{MAX}=128)

This figure above shows the execution time of AS1 compared with URD. For URD this time is of about µs, however, for AS1 it is around the 15 ms. We can observe some fluctuations in this execution time, specially at the beginning, and then during all the simulation time. These changes in the execution time can be produced by random fluctuations in the efficiency of the processor. The slightly higher execution time in the first iterations can be caused because the algorithm is converging faster.

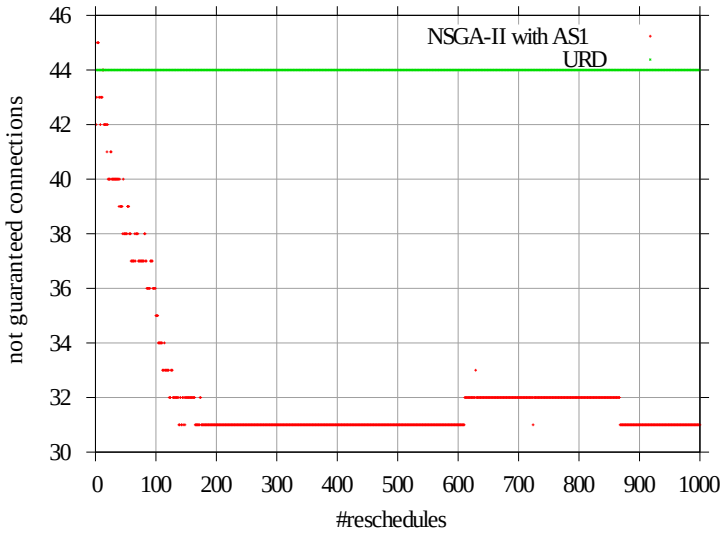

Figure 38. Not Guaranteed Connections (%) in Scenario 1 – AS1 and URD (N=64, R_{MAX}=128)

In Figure 38, we can observe the drop of the percentage of the not guaranteed connections for the 150th first TS, and then how this percentage remains stable. While the 44.00% of the connections cannot be guaranteed with URD, with AS1, almost the 70.00% of the requests can be guaranteed.

c) N=32, R_{MAX}=64:

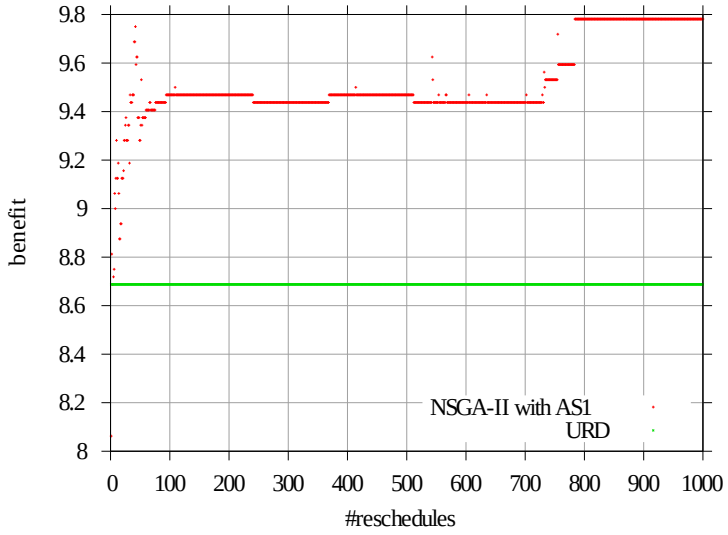

Figure 39. Benefit in Scenario 1 - AS1 and URD (N=32, R_{MAX}=64)

In the figure above the values obtained for the benefit are represented. We can observe that a solution is found since the first TS, and during approximately the following 100 iterations, the benefit increases since it gets a stable value of 9.44. From the 730th to the 783th TS there is another increasing in the benefit value done in some steps, until it reaches the maximum value of 9.78. We can also observe that in all cases the benefit achieved by AS1 is higher than the achieved by URD.

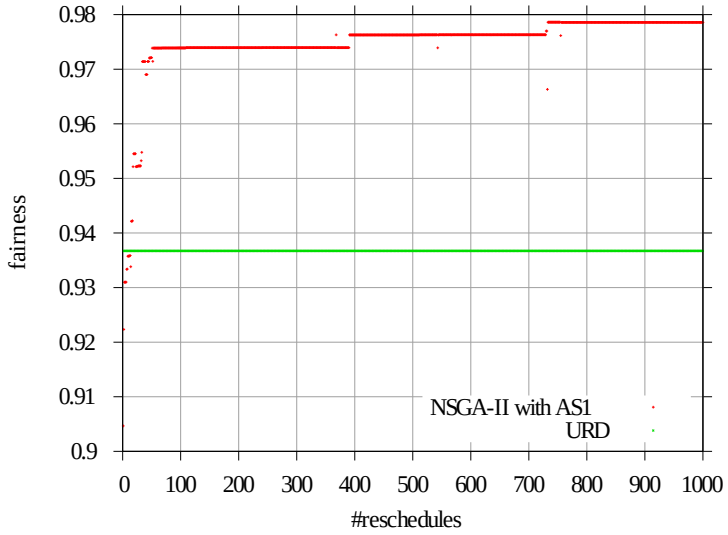

Figure 40. Fairness in Scenario 1 - AS1 and URD (N=32, R_{MAX}=64)

Figure 40 shows the evolution of fairness during all the allocations. As it happens with the benefit, there is a rise in the value of the fairness for the first 50th TS, and then the value remains constant, making two steps, one a little bit before the 400th TS, and the

other in, approximately, the 730th TS, reaching the highest value of 0.978. During almost all the simulation time the fairness of the system with AS1 is higher than with URD, which achieves a value of 0.937.

We can also observe that in this case, with 32 users, the curves of the benefit and the fairness with AS1 are not very smooth and become more abrupt than with 64 and 128 users. This fact means that NSGA-II stagnates at the points where there is a step because it hasn't found a better solution, and when it finds it, it moves over. In order to avoid these steps and find a smoother curve, different parameters from the selected ones should be chosen, as the ones selected are suitable for networks with a larger number of users.

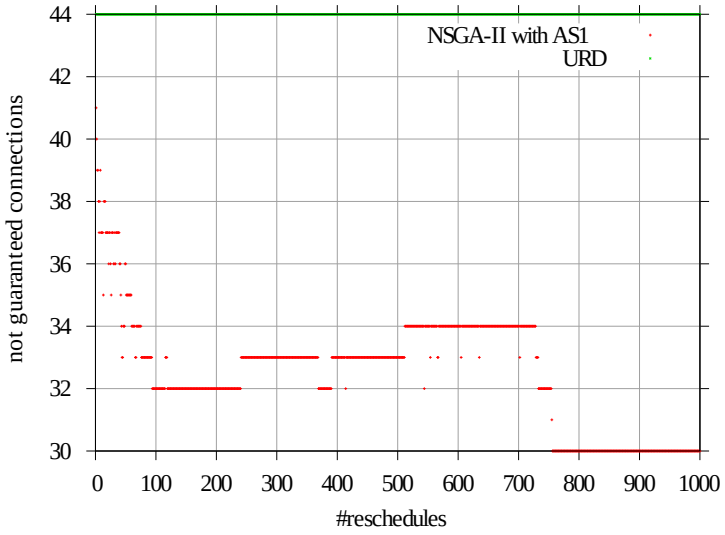

Figure 41. Not Guaranteed Connections (%) in Scenario 1 - AS1 and URD (N=32, R_{MAX}=64)

Figure 41 shows the evolution of the not guaranteed connections. We observe that the percentage of these connections is lower than with URD. However, we can also see that the curve is not strictly decreasing, but has several steps in which this percentage gets higher and lower, until one point in which it decreases and then remains constant in the 30.00%.

Figure 42 shows the execution time for both AS1 and URD, and we can observe that, except for some fluctuations, the execution time for AS1 remains in the 6ms, higher than the μs of the URD.

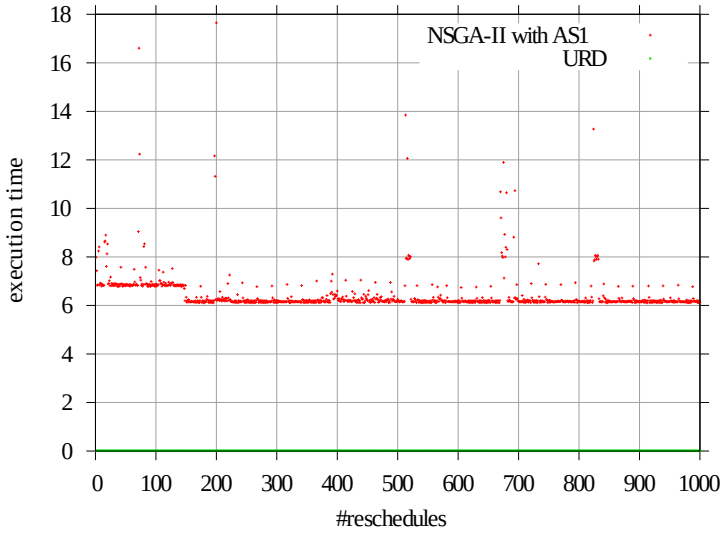

Figure 42. Execution Time (ms) in Scenario 1 - AS1 and URD (N=32, R_{MAX}=64)

d) N=16, R_{MAX}=32

Figure 43 shows us that with 16 users the AS1 algorithm is not able to find a solution with higher benefit than URD. The benefit value increases and decreases by steps until it gets stagnated in the value of 7.875, which is lower than the 8.125 of the URD.

In Figure 44 the steps in the curve for AS1 can be observed, although the fairness reaches higher values (0.95) than for the URD (0.911).

As with 32 users, optimizing the parameters for this configuration, smoother curves for benefit and fairness could be generated.

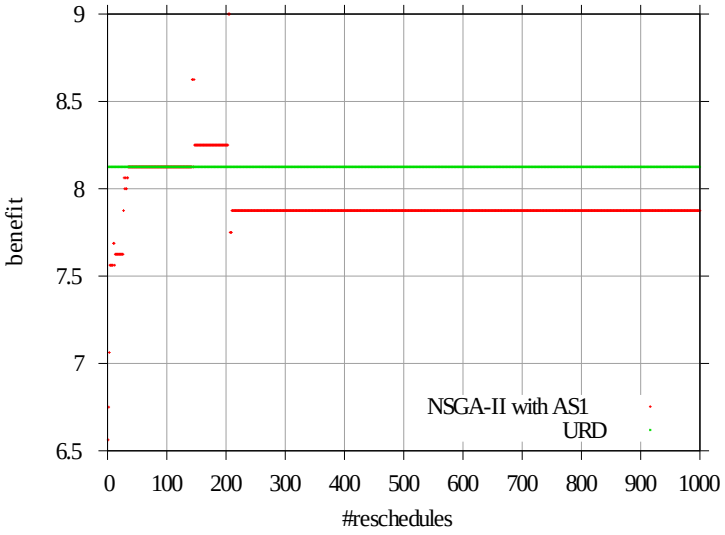

Figure 43. Benefit in Scenario 1 - AS1 and URD (N=16, R_{MAX}=32)

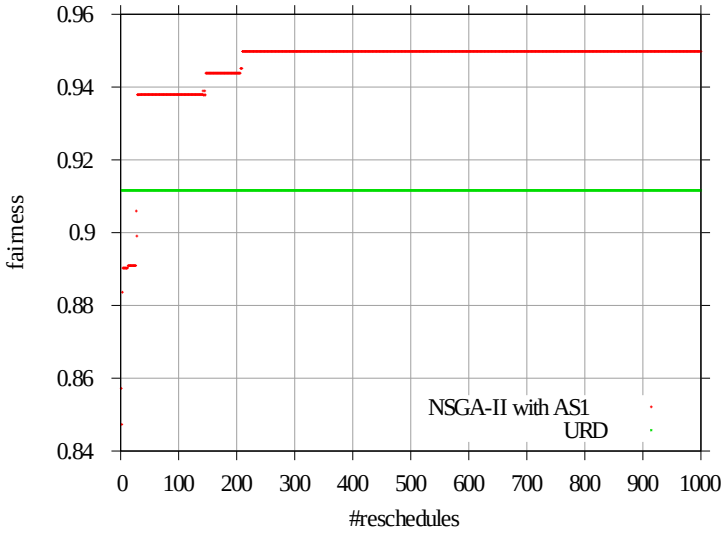

Figure 44. Fairness in Scenario 1 - AS1 and URD (N=16, R_{MAX}=32)

Figure 45 illustrates that for AS1, from the 142th TS, the percentage of not guaranteed connections remains constant to a value of 31.00%, lower than the 40.00% of not guaranteed connections for URD strategy.

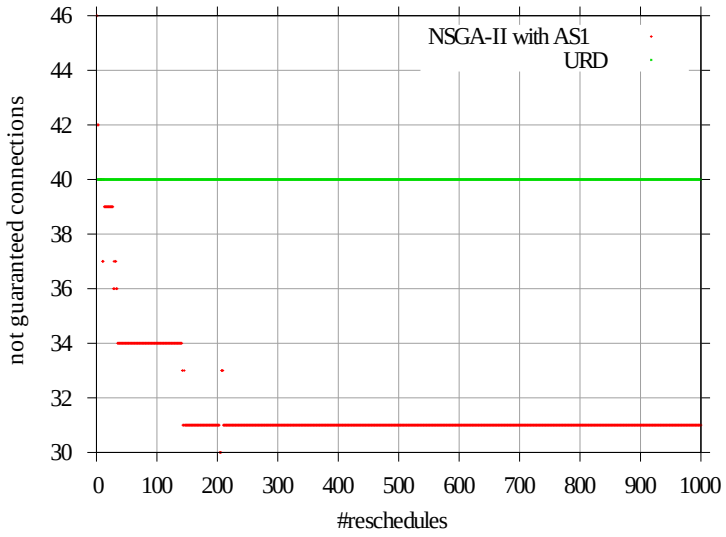

Figure 45. Not Guaranteed Connections (%) in Scenario 1 - AS1 and URD (N=16, R_{MAX}=32)

The average execution time for AS1 is 2.82 ms, while for URD is 5.11 µs.

Although with this network configuration the benefit achieved by AS1 is lower than the benefit achieved by URD, in both fairness and guaranteed connections, AS1 obtain better results.

e) N=8, R_{MAX}=16:

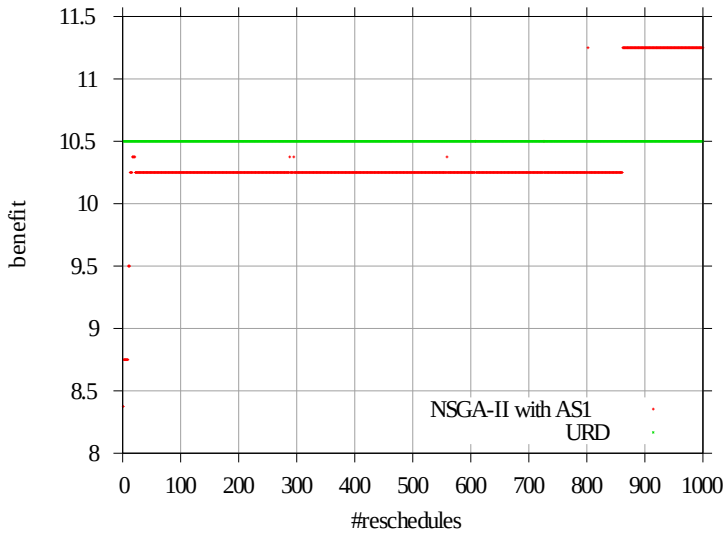

Figure 46. Benefit in Scenario 1 - AS1 and URD (N=8, R_{MAX}=16)

Figure 46 exemplifies how the benefit curve has a big step in approximately the 850th TS, in which its value increases from 10.25 to 11.25, overcoming the benefit value obtained by URD, 10.5.

In Figure 47, the fairness remains constant in almost all the time, and lower than the achieved by URD.

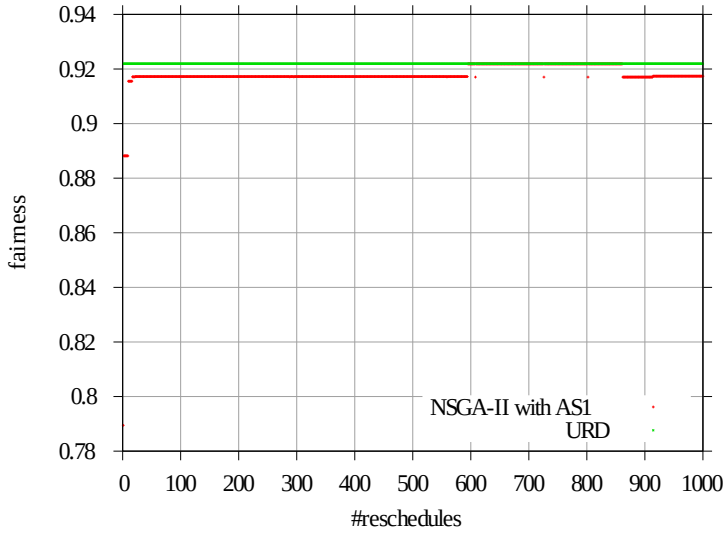

Figure 47. Fairness in Scenario 1 - AS1 and URD (N=8, R_{MAX}=16)

Figure 48 and Figure 49 show not guaranteed connections and execution time, respectively. In the first one we can observe that the curve from AS1 shows a better

behaviour, in which the percentage for not guaranteed connections decreases to the 27.00%. In the second one an execution time lower than 1.5 ms is illustrated.

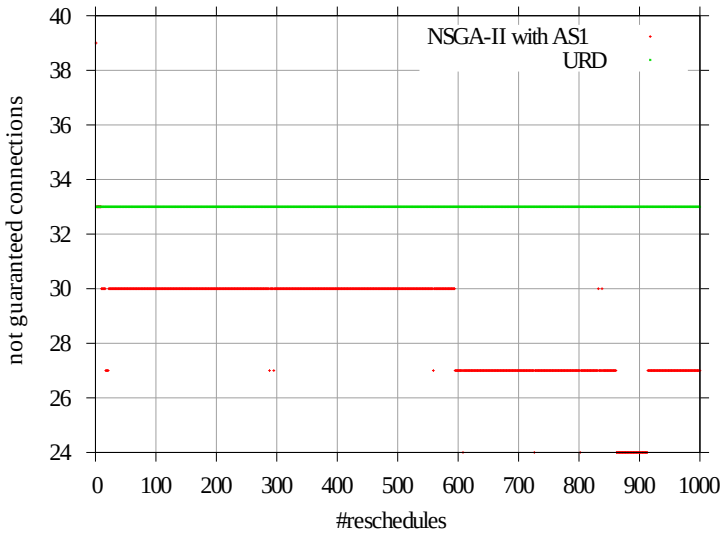

Figure 48. Not Guaranteed Connections (%) in Scenario 1 - AS1 and URD (N=8, R_{MAX}=16)

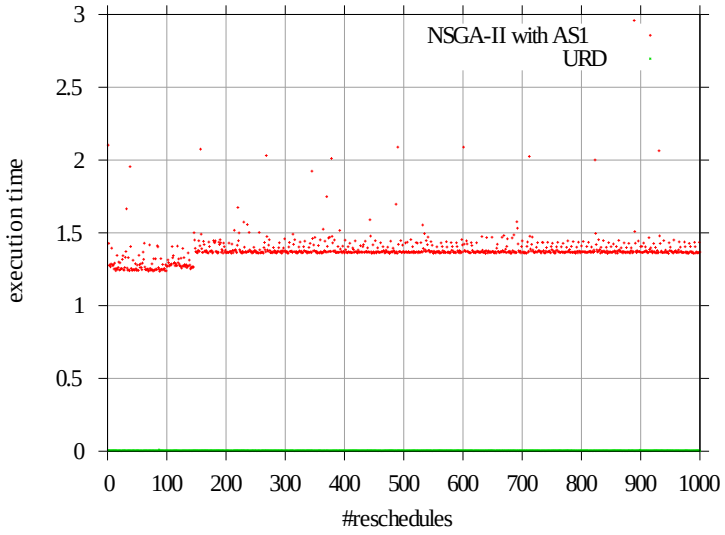

Figure 49. Execution time (ms) in Scenario 1 - AS1 and URD (N=8, R_{MAX}=16)

f) N=4, R_{MAX}=8:

The results exposed in Figure 50 and Figure 51 demonstrate that NSGA-II with AS1 performs worse than URD in terms of benefit and fairness. In addition, Figure 52 shows that the percentage of not guaranteed connections is the same with both strategies. The execution time showed in Figure 53 point up that with AS1 strategy, the average needed time to calculate the resource allocation of the requests is 0.8 ms, while it takes only µs for URD to make this calculation.

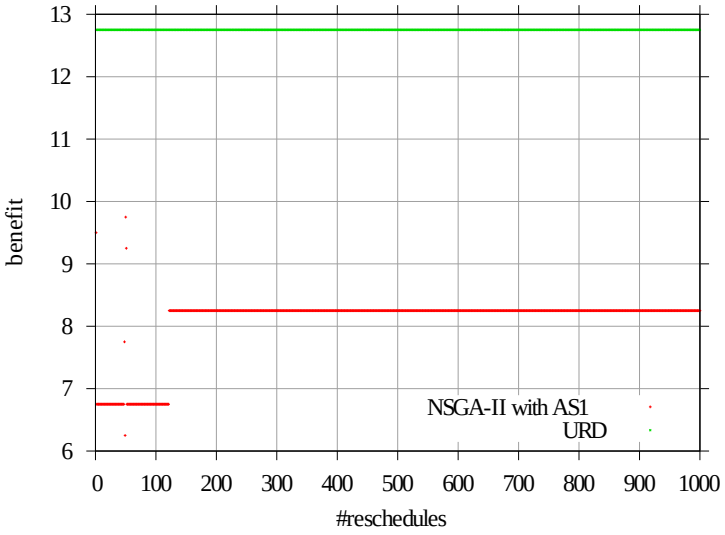

Figure 50. Benefit in Scenario 1 - AS1 and URD (N=4, R_{MAX}=8)

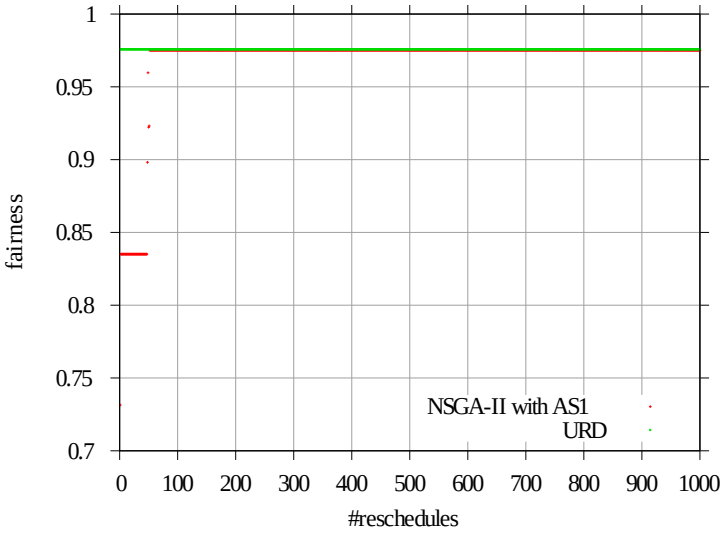

Figure 51. Fairness in Scenario 1 - AS1 and URD (N=4, R_{MAX}=8)

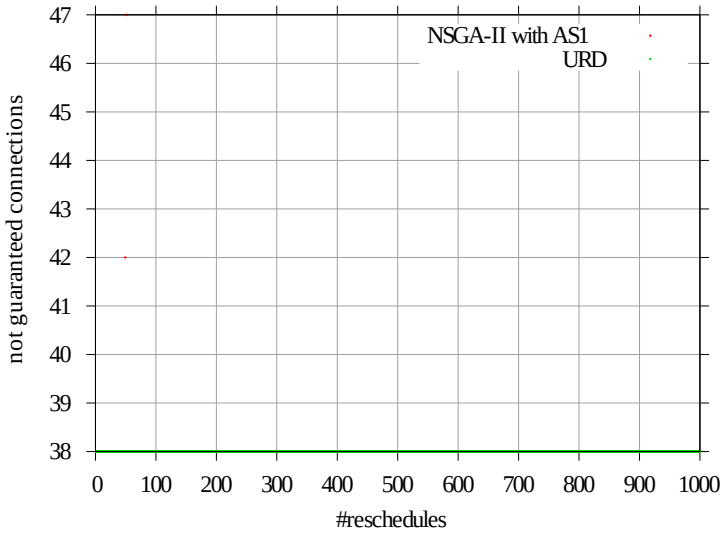

Figure 52. Not Guaranteed Connections (%)in Scenario 1 - AS1 and URD (N=4, R_{MAX}=8)

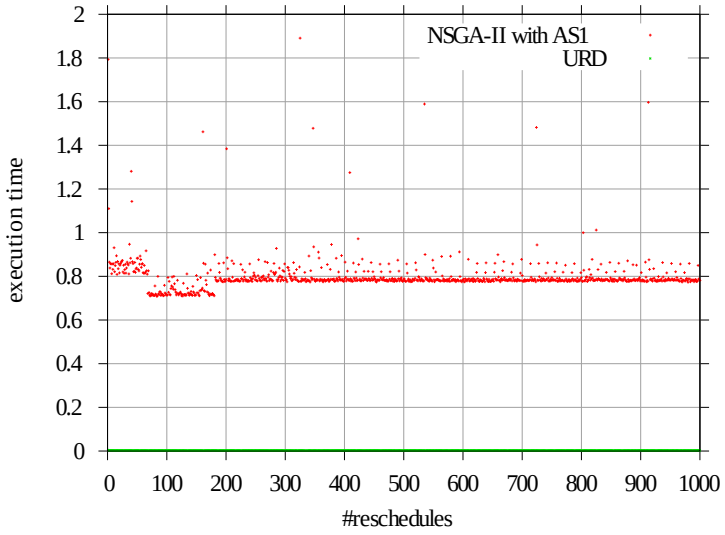

Figure 53. Execution Time in Scenario 1 - AS1 and URD (N=4, R_{MAX}=8)

As a conclusion, we can say that NSGA-II with AS1 strategy performs better than URD in almost all the cases, understanding that a better performance implies higher benefit, fairness and guaranteed connections percentage. When the number of users in the network decreases the differences between both algorithms are narrower, until URD outperforms AS1. The execution time is always lower for URD than for AS1, as the method for the uniform allocation is much simpler. However, AS1 is suitable for a real time system as it can calculate an allocation faster than 1 TS, even when the number of users is 128.

6.2.4. Adaptation Strategy 2

With this second adaptation strategy, as it has been explained in the simulation study, not only the previous results, but also some random information is used to initialize the NSGA-II algorithm. We have added this random information in two different ways, calling these strategies AS2a and AS2b, as described in 5.2.4. We have to remind that the difference between these two initialization methods was that the random information was toggled in the genes, for the AS2a strategy, and to the individuals, to the AS2b strategy.

The results obtained with both strategies, compared with the uniform resource distribution are now presented with the six different network configurations told before.

a) N=128, R_{MAX}=256:

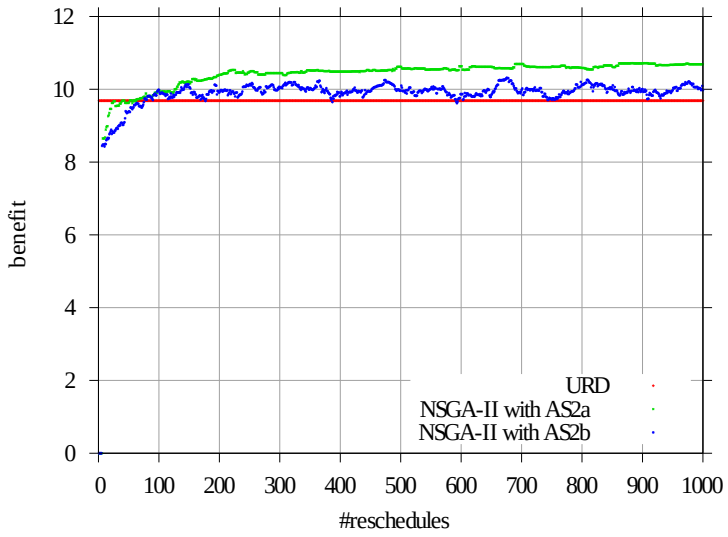

Figure 54. Benefit in Scenario 1 – AS2a, AS2b and URD (N=128, R_{MAX}=256)

In the figure above we can observe the curves for the benefit. For AS2a, this curve is smoother than for AS2b, as we can see that there are more fluctuations. While AS2a curve increases for the first 200th TS and then remains stable in the average value of 10.7, AS2b curve has a higher variation in the benefit value, and its average value is slightly lower, approximately 10. Nevertheless, both strategies achieve a higher benefit value than URD before the first 100 allocations have been done.

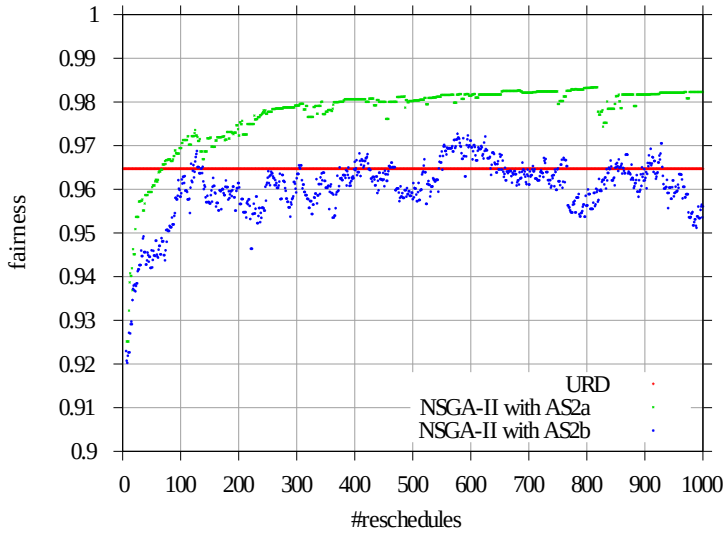

Figure 55. Fairness in Scenario 1 – AS2a, AS2b and URD (N=128, R_{MAX}=256)

Figure 55 illustrates the evolution of the fairness along the time. It can be observed that strategy AS2a achieves higher values than AS2b and that the convergence and stabilization of these values is faster. On the other hand, AS2b curve shows

fluctuations and doesn't seem to find a stable value. The average fairness value achieved by AS2a is 0.97, while for AS2b is 0.955. The fairness achieved by URD is 0.965.

The percentage of not guaranteed connections, when the strategy applied is AS2a, has a clear tendency to decrease, until a value of 32.00% is reached. However, AS2b has not this tendency, as the percentage of not guaranteed connections oscillates between the 38.00 and the 43.00%. Nevertheless, in almost all the cases, these values are lower than the rate achieved by URD, which is the 42.00%.

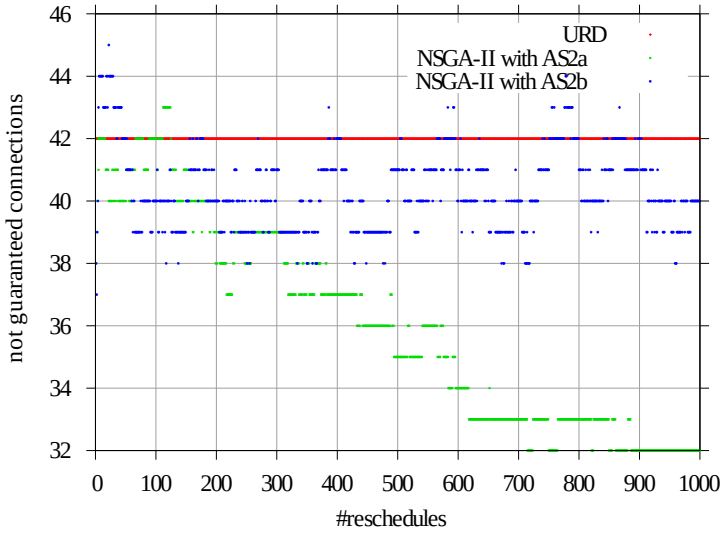

Figure 56. Not Guaranteed Connections (%) in Scenario 1 – AS2a, AS2b and URD (N=128, R_{MAX}=256)

Average Execution Time		
AS2a	**AS2b**	**URD**
57.26 ms	57.46 ms	22.27 µs

Table 5. Execution Time in Scenario 1 – AS2a, AS2b and URD (N=128, R_{MAX}=256)

The table above shows the average execution time for each allocation. As the premise for selecting the parameters was a execution time smaller than 1 TS, both AS2a and AS2b require less than 60 ms to calculate the allocations. URD is much faster, as it has been seen before, having an execution time of about microseconds.

b) N=64, R_{MAX}=128:

Figure 57 and Figure 58 illustrate that the two NSGA-II strategies under study perform better than URD, when not more than 100 allocations (much earlier in the case of fairness) have been done. In terms of benefit, when AS2a is stable, the curve can achieve a value of 10.5, and AS2b, although has not a stable value, can reach an average

benefit of 9.86, slightly higher than the 9.18 achieved by the URD. In terms of fairness, AS2a seems to have a stable value of 0.98 in the 700th iteration, while with AS2b the network reaches an average fairness of 0.97, both values higher than the ratio obtained by URD, 0.95.

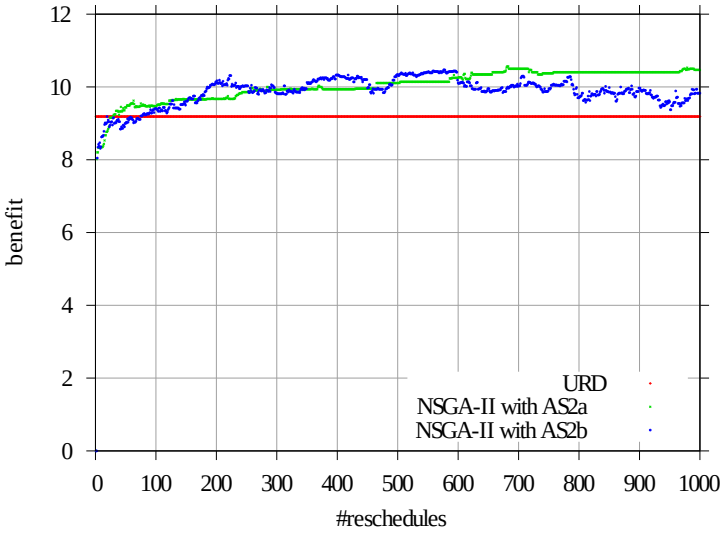

Figure 57. Benefit in Scenario 1 – AS2a, AS2b and URD (N=64, R_{MAX}=128)

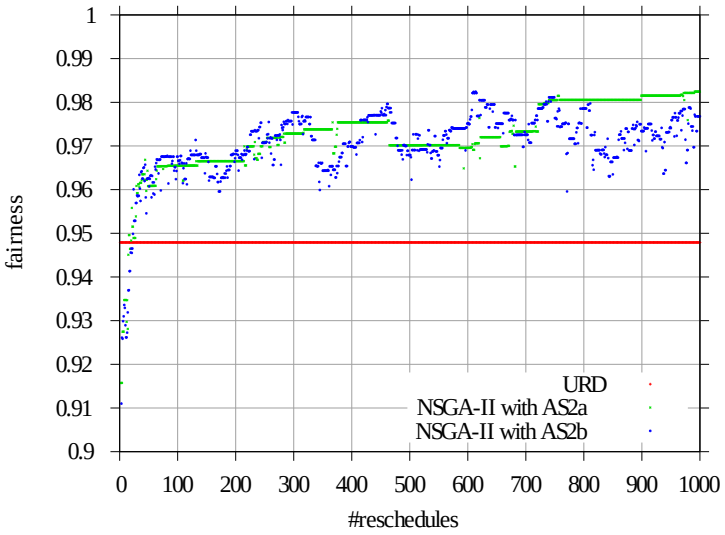

Figure 58. Fairness in Scenario 1 – AS2a, AS2b and URD (N=64, R_{MAX}=128)

The figure below also shows the better behaviour of the NSGA-II based algorithms when assuring resources, as during almost all the time, AS2a and AS2b have more guaranteed connections than URD. AS2a has a decreasing curve, with some small fluctuations and keep an average percentage of not guaranteed connections value higher than with AS2a. Oscillations in the AS2b curve are bigger. Both average values, 34.50% and 37.00% for AS2a and AS2b, respectively, are better than the 44.00% for URD.

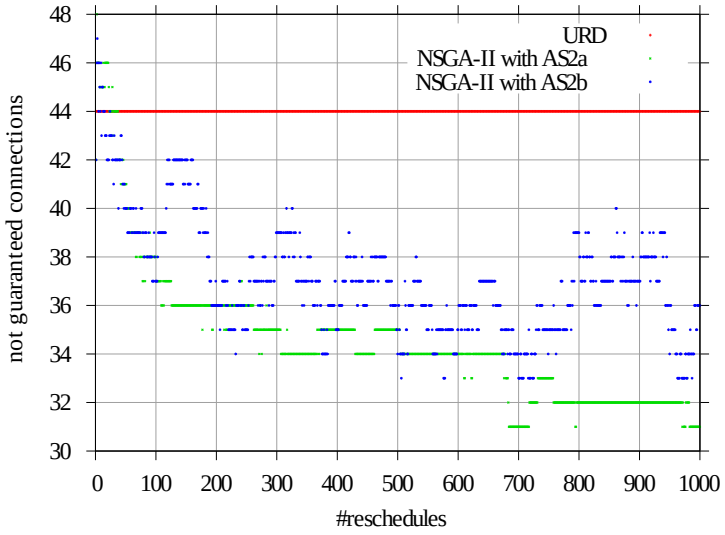

Figure 59. Not Guaranteed Connections (%) in Scenario 1 – AS2a, AS2b and URD (N=64, R_{MAX}=128)

Average Execution Time		
AS2a	**AS2b**	**URD**
18.27 ms	17.90 ms	14.09 µs

Table 6. Execution Time in Scenario 1 – AS2a, AS2b and URD (N=64, R_{MAX}=128)

As the number of users decreases, the number of variables involved in the process also reduces, making the algorithm computationally simpler, thus, faster. The difference between AS2a and AS2b are circumstantial.

c) N=32, R_{MAX}=64:

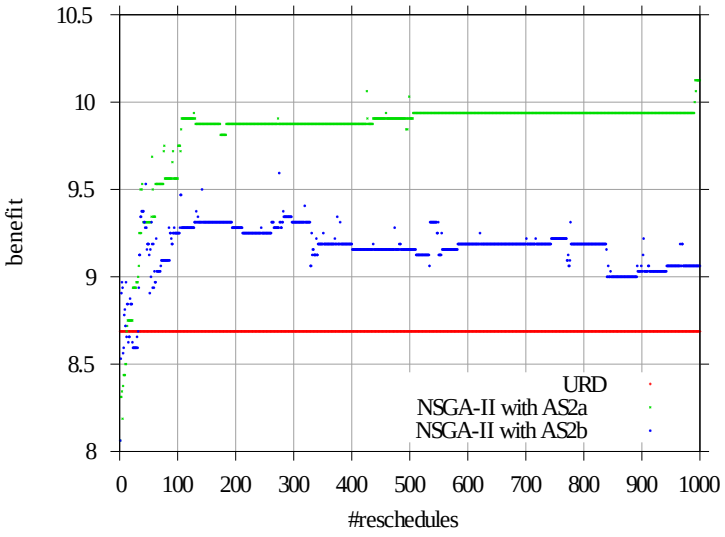

Figure 60. Benefit in Scenario 1 – AS2a, AS2b and URD (N=32, R_{MAX}=64)

As in the other cases, with 32 users and a maximum allocation of 64 normalized units, the implemented strategies reach a higher benefit and fairness value than URD, as it can be seen in Figure 60 and in Figure 61. For AS2a strategy, the benefit curve increases during the first 100th iterations, and then is kept in an almost constant value of 9.94. In the last 10th TS there is a jump in this value until it reaches 10.125; with a longer simulation could be checked if this is the maximum possible value. The fairness also increases dramatically in the first 100th allocations, and then the increasing is produced by several steps until it reaches a value of 0.978. The behaviour of AS2b strategy is very similar. The increasing is even faster than for AS2a and the final reached value is slightly higher, 0.979. The average fairness for AS2a and AS2b, respectively, is 0.970 and 0.976. The URD fairness value is 0.937. As happened with AS1 strategy, we can observe how the benefit and fairness lose its smoothness when the number of users decreases. As it was explained, this fact could be avoided selecting NSGA-II parameters more suitable for smaller networks.

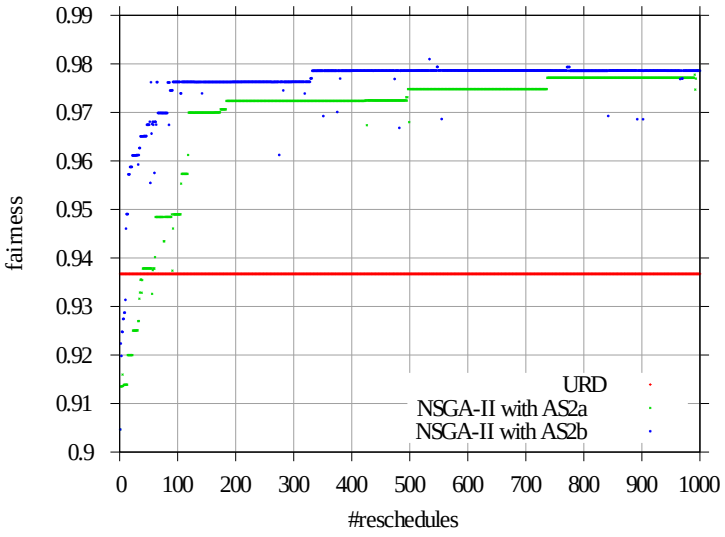

Figure 61. Fairness in Scenario 1 – AS2a, AS2b and URD (N=32, R_{MAX}=64)

The not guaranteed connections, in both AS2a and AS2b, have a decreasing behaviour, with some oscillations, as we can observe in Figure 62. We can see that for AS2b the lowest value achieved is 30.00%, while for AS2a is 32.00%. On the other hand, both values are better than the 44.00% of not guaranteed connections for URD.

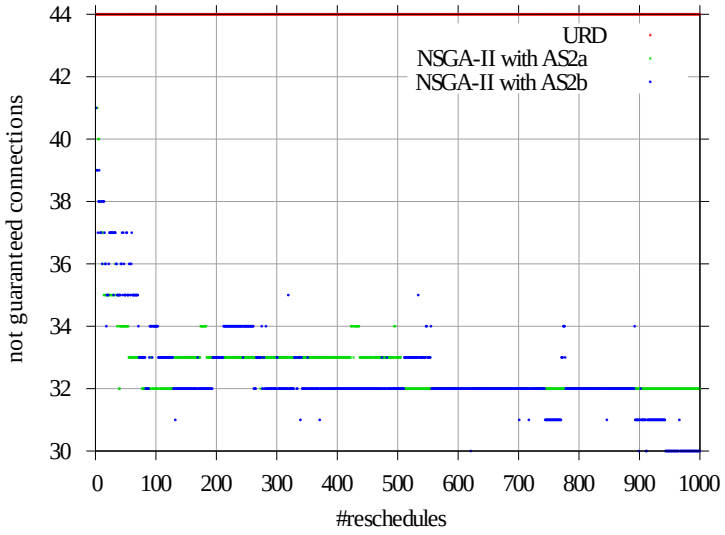

Figure 62. Not Guaranteed Connections (%) in Scenario 1 – AS2a, AS2b and URD (N=32, R_{MAX}=64)

The table below shows the execution time for each allocation. Logically, the values are lower than for a larger number of users, and are suitable for a real time system.

Average Execution Time		
AS2a	**AS2b**	**URD**
6.72 ms	6.28 ms	8.24 µs

Table 7. Execution Time in Scenario 1 – AS2a, AS2b and URD (N=32, R_{MAX}=64)

d) N=16, R_{MAX}=32:

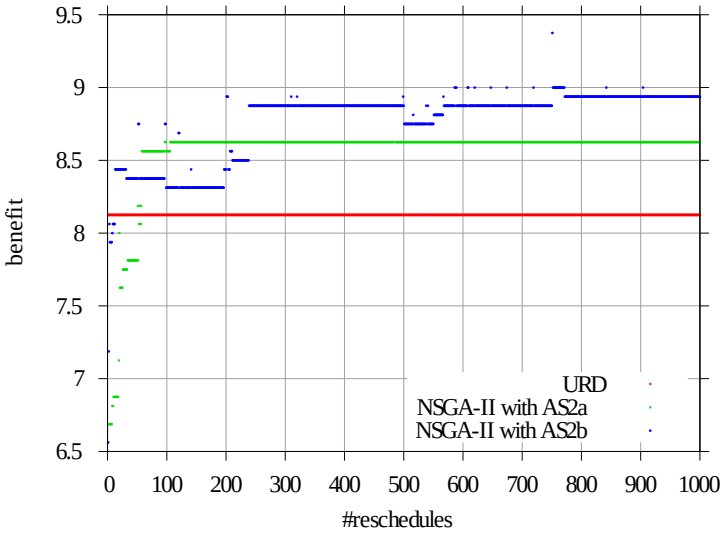

Figure 63. Benefit in Scenario 1 – AS2a, AS2b and URD (N=16, R_{MAX}=32)

Figure 63 and Figure 64 show that in both benefit and fairness, AS2b technique achieves better results than AS2a, and URD as well. While, in terms of benefit, AS2a increases during the first 100th TS and then remains constant in 8.625, AS2b has more oscillations, but remains in a higher value of benefit, 8.94. In terms of fairness, both behaviours are very similar, the difference, again, is in the final value achieved, which is 0.950 for AS2a and 0,964 for AS2b.

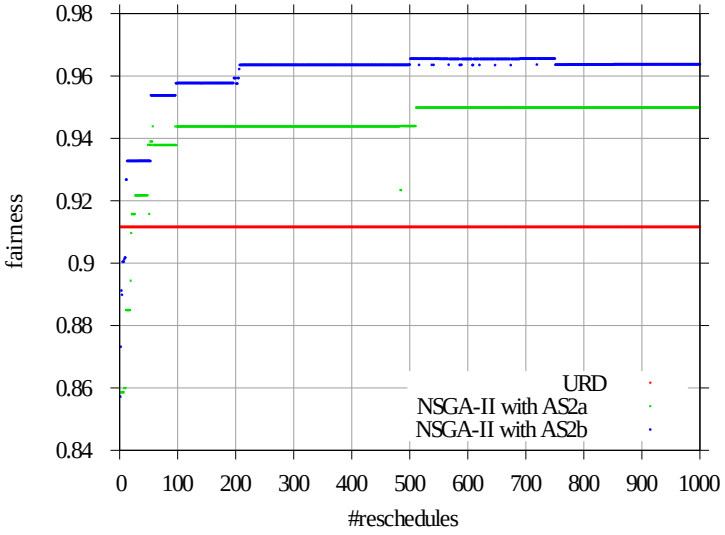

Figure 64. Fairness in Scenario 1 – AS2a, AS2b and URD (N=16, R_{MAX}=32)

Figure 65 shows the evolution of the not guaranteed connections. While AS2a achieves a minimum (30.00%) value quite fast, AS2b makes some oscillations, and achieves a final value of 33.00%. Both values are better than the 40.00% of not guaranteed allocations for URD.

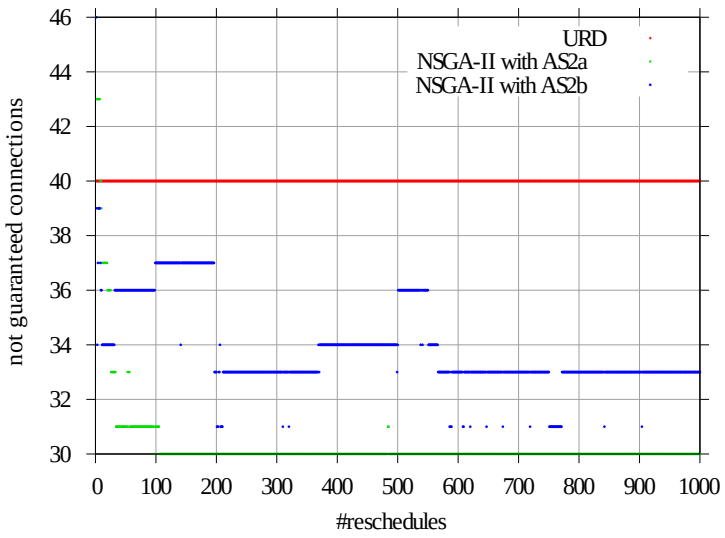

Figure 65. Not Guaranteed Connections (%) in Scenario 1 – AS2a, AS2b and URD (N=16, R_{MAX}=32)

The execution times in this case are the ones shown below.

Average Execution Time		
AS2a	**AS2b**	**URD**
2.94 ms	2.83 ms	5.11 µs

Table 8. Execution Time (ms) in Scenario 1 – AS2a, AS2b and URD (N=16, R_{MAX}=32)

e) N=8, R_{MAX}=16:

In Figure 66 we can observe that both AS2a and AS2b benefit curves have oscillations, achieving an average value of 11.02 and 10.85 respectively, slightly higher than the10.5 achieved by URD. In terms of fairness, the three curves are very similar, as the two NSGA-II methods have very small changes in their curves, reaching average values of 0.915 and 0.919. The value obtained by URD 0.922, a slightly better value for the fairness. We can observe these results in Figure 67.

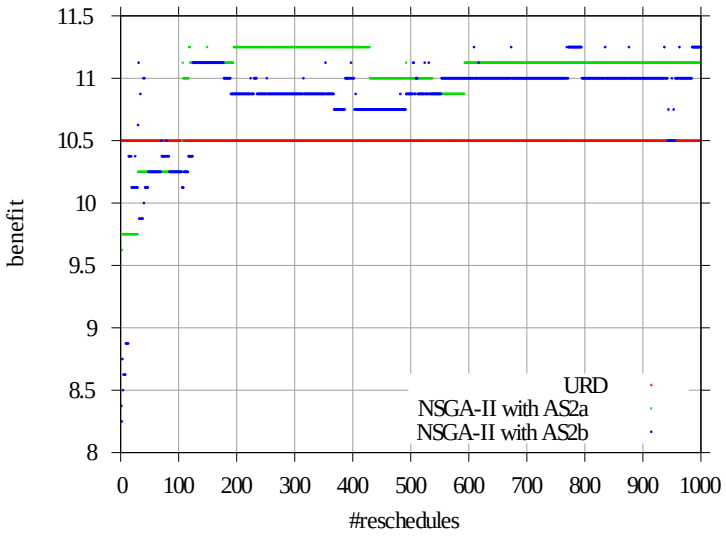

Figure 66. Benefit in Scenario 1 – AS2a, AS2b and URD (N=8, R_{MAX}=16)

In Figure 68 it is also obvious that the NSGA-II parameters should be modified in order to get better results, as the curves for the not guaranteed connections don't have a decreasing tendency, but have fluctuations, and in the case of AS2b, performs worse than URD in some allocations. Nevertheless the average values for the not guaranteed connections for both AS2 are better than URD. AS2a achieves 26.61% of average not guaranteed connections, AS2b 32.03%and URD 33.00%.

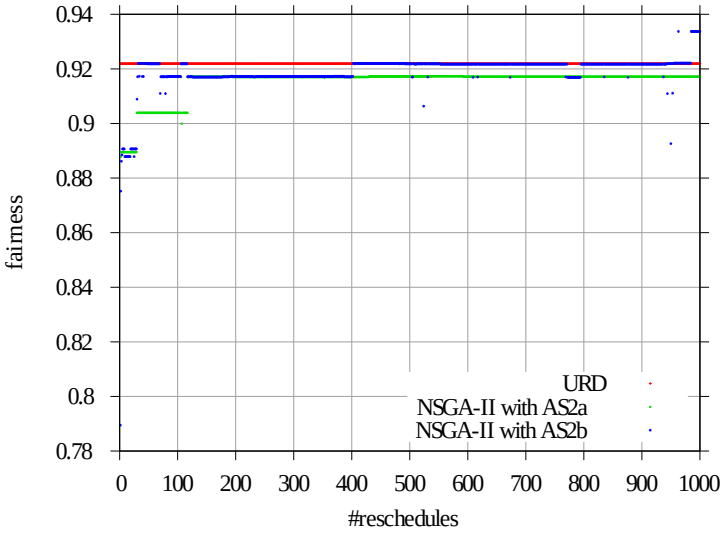

Figure 67. Fairness in Scenario 1 – AS2a, AS2b and URD (N=8, R_{MAX}=16)

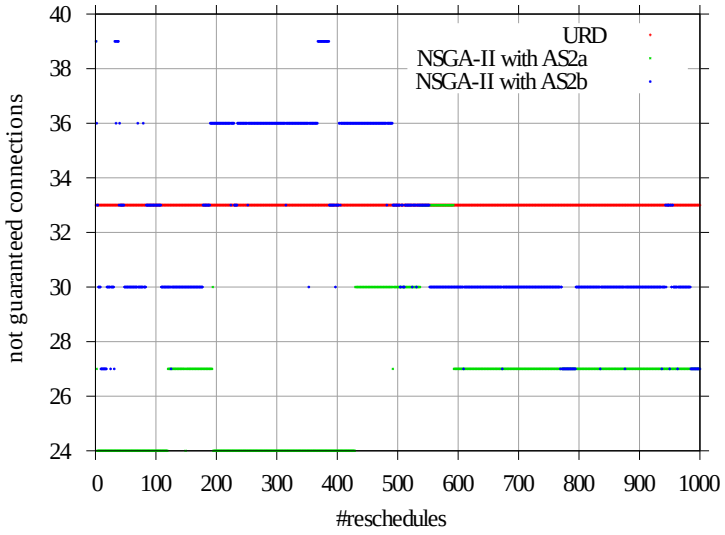

Figure 68. Not Guaranteed Connections (%) in Scenario 1 – AS2a, AS2b and URD (N=8, R_{MAX}=16)

The table below shows the execution times for the three strategies.

Average Execution Time		
AS2a	**AS2b**	**URD**
1.99 ms	1.43 ms	4.05 µs

Table 9. Execution Time in Scenario 1 – AS2a, AS2b and URD (N=8, R_{MAX}=16)

f) N=4, R_{MAX}=8:

Figure 69 demonstrates that AS2a and AS2b perform worse than URD in terms of benefit for this network configuration. In addition it also obvious that the parameters used are not suitable, as the changes in the curve are not smooth and the highest value

obtained is far from being an optimum. Figure 70 shows that the values obtained, in AS2a, AS2b and in URD as well, are very close.

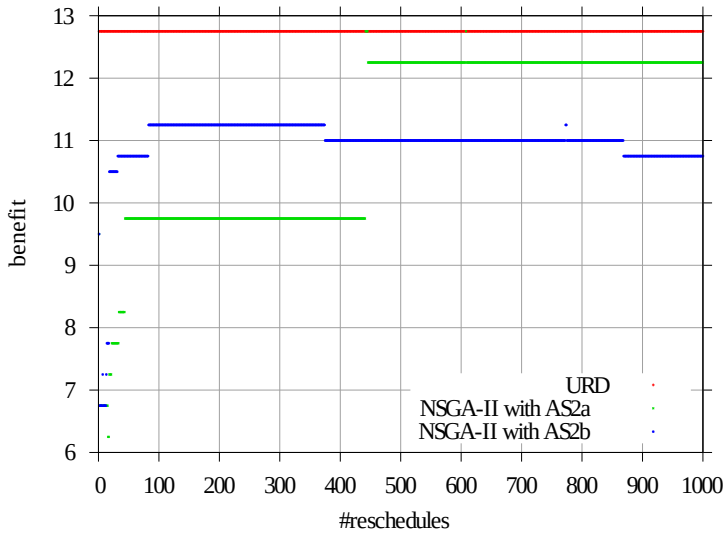

Figure 69. Benefit in Scenario 1 – AS2a, AS2b and URD (N=4, R_{MAX}=8)

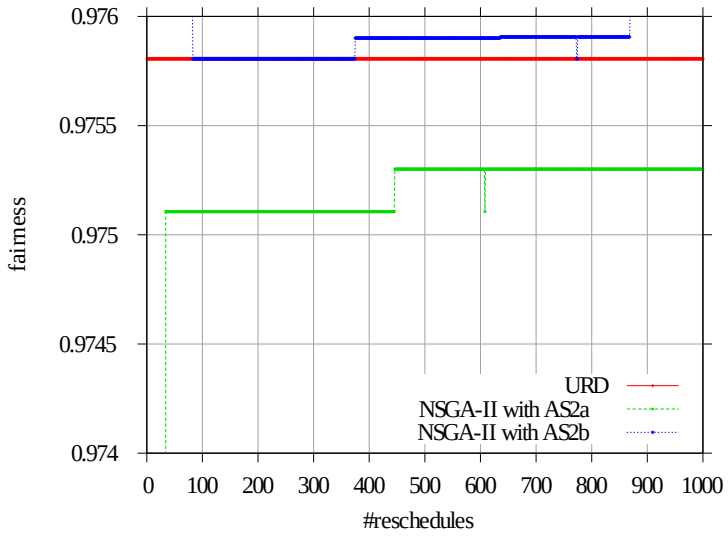

Figure 70. Fairness in Scenario 1 – AS2a, AS2b and URD (N=4, R_{MAX}=8)

The not guaranteed connections graphic also shows that in this case the performance of URD is better, as the average values obtained are 38.18%, 41.77% and 38.00% for AS2a, AS2b and URD, respectively.

Average Execution Time		
AS2a	**AS2b**	**URD**
0,74 ms	0,84 ms	3,10 µs

Table 10. Execution Time in Scenario 1 – AS2a, AS2b and URD (N=4, R_{MAX}=8)

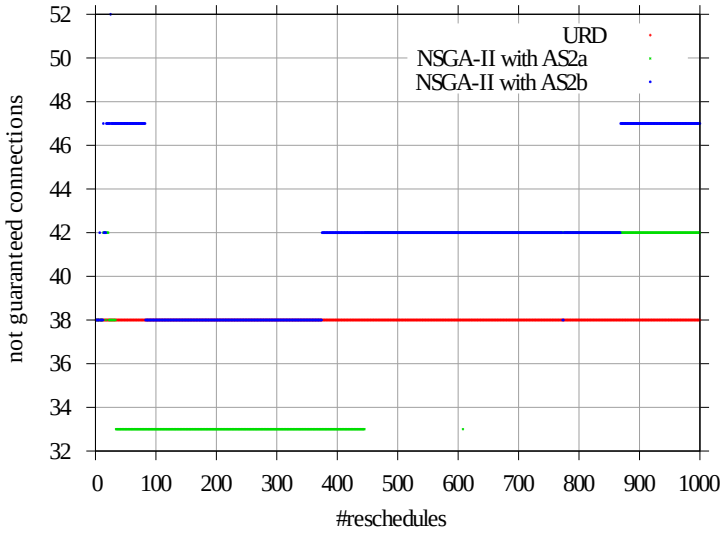

Figure 71. Not Guaranteed Connections in Scenario 1 – AS2a, AS2b and URD (N=4, R_{MAX}=8)

We can conclude that both strategies AS2a and AS2b achieve better results in terms of benefit, fairness and not guaranteed connections than URD in all cases, except when the number of users is small (N=4). We can also observe that the curve for AS2b has more oscillations when the number of users is high, and that both strategies have less smooth curves when the number of users is low. As with AS1, the execution times are higher for the NSGA-II based strategies rather than URD.

6.2.5. Comparison

In the last sections, three NSGA-II based strategies have been studied and compared with a Uniform Distribution Resource scheduling algorithm, in a scenario in which 2 maximum connections for each service type can be requested for each user. The NSGA-II parameters have been selected in order to assure that the allocation can be calculated in less than one Time Slot, i.e., the execution time has to be lower than 60 ms. The results obtained for the benefit and fairness are summarized in Table 11.

N	128		64		32		16		8		4	
	B	F	B	F	B	F	B	F	B	F	B	F
NAS	-	-	0.07	0.008	0.97	0.11	5.43	0.65	8.13	0.8	5.29	0.55
AS1	10.36	0.97	10.19	0.97	9.52	0.97	7.92	0.95	10.37	0.92	8.08	0.97
AS2a	10.37	0.97	10.03	0.97	9.84	0.97	8.56	0.94	11.02	0.92	11.05	0.97
AS2b	9.86	0.96	9.85	0.97	9.17	0.98	8.77	0.96	10.85	0.92	10.95	0.97
URD	9.68	0.96	9.19	0.95	8.69	0.94	8.13	0.91	10.5	0.92	12.75	0.97

Table 11. Benefit and Fairness in Scenario 1

It can be seen that NAS strategy is not suitable to solve our problem, as when the number of users is high no solutions are found, and when the number of users decreases, the values obtained are far from being optimal. To achieve better results, the NSGA-II parameters, such as the number of generations and/or the population size, should be increased, with the subsequent increase in the execution time. Thus, the algorithm would be too slow for a real time system.

On the other hand, when the results of previous allocations are used to initialize the NSGA-II algorithm, the results are significantly better. Both strategies AS1 and AS2a achieve very similar values when the number of users is high (for example, for 128 or 64 users), but when the number of users is reduced, AS2a performs better. AS2b strategy obtains also similar values to AS2a and AS1 when the network has an intermediate number of users (32 and/or 16). However, we have seen that the fluctuations along the time of this strategy make it unstable and quite unpredictable.

A long time simulation (100.000 TS) has been done in order to see the convergence characteristic of the different strategies studied. The benefit has been calculated for AS1, AS2a, AS2b and URD, with the parameters of this scenario, and for a network with 128 users and a maximum allocation of 256.

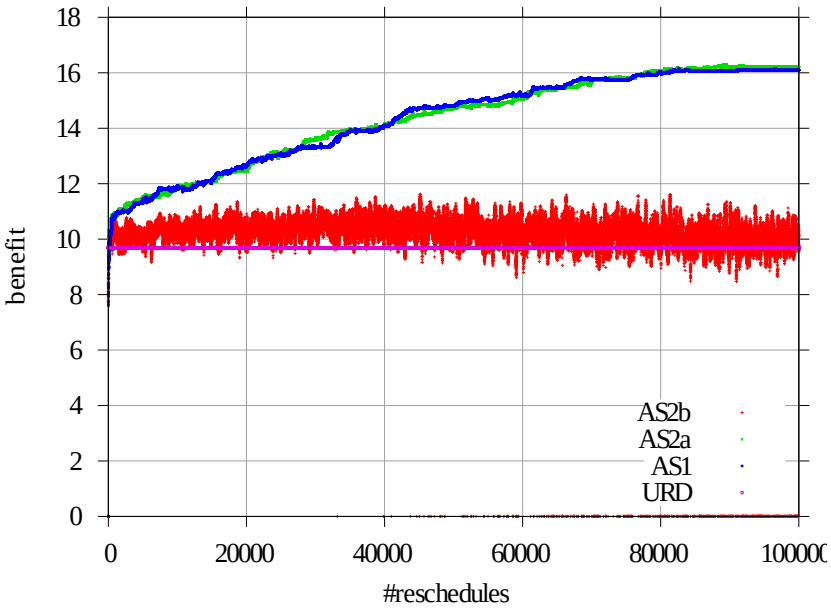

Figure 72. Convergence of Benefit in Scenario 1 (N=128, R_{MAX}=256)

The figure above shows how AS1 and AS2a have nearly the same behaviour. Their curves are increasing until they reach what seems to be the maximum value for the

benefit, in this case, 16.09 by AS1 and 16.21 by AS2a. Observe that, for URD the benefit remains constant in a value inferior to 10.

On the other hand, AS2b curve stays, in average, in the same benefit value, although with a strong oscillation, and when the number of time slots increases, it suffers degradation, as some of the solutions are not obtained (benefit zero points). This fact makes the average benefit weaken.

Thus, it is clear, that strategies AS1 and AS2a perform much better than AS2b and URD. The fairness values obtained are, respectively, 0.9983 and 0.9976.

In terms of execution time, the time needed to calculate the optimal allocation is, for AS1, 49.44 ms, while for AS2a is 51.06 ms. The percentage of not guaranteed resources is 46.6% for AS2a and 47.3% for AS1.

6.3. Scenario 2

6.3.1. Description

The characteristics of Scenario 2 are almost the same that Scenario 1. The main difference between them is that in this scenario, the requests from the users to the different types of services are uniformly distributed between 0 and 5. The maximum resource allocation has been also slightly increased, as it is showed in Table 13. The probability of changing in the request matrix is the same as in Scenario 1. The following table show the new NSGA-II parameters used to do the simulations.

Number of Generations	2
Population size	10
Mutation probability	0,005
Cross-over probability	0,8

Table 12. NSGA-II parameters for Scenario 2

Number of Users	4	8	16	32	64	128
Maximum Allocation	10	20	40	80	160	320

Table 13. Users and Maximum Resource Allocation for 5 maximum requests

6.3.2. No Adaptation Strategy

The algorithm is not able to find a solution when the initialization is completely random. With the parameters selected to run the simulations in Scenario 2, NAS strategy

doesn't work in any of the network configurations, even when the number of users is low. Figure 73, which illustrates the benefit for a network with 4 users, is a prove of this fact.

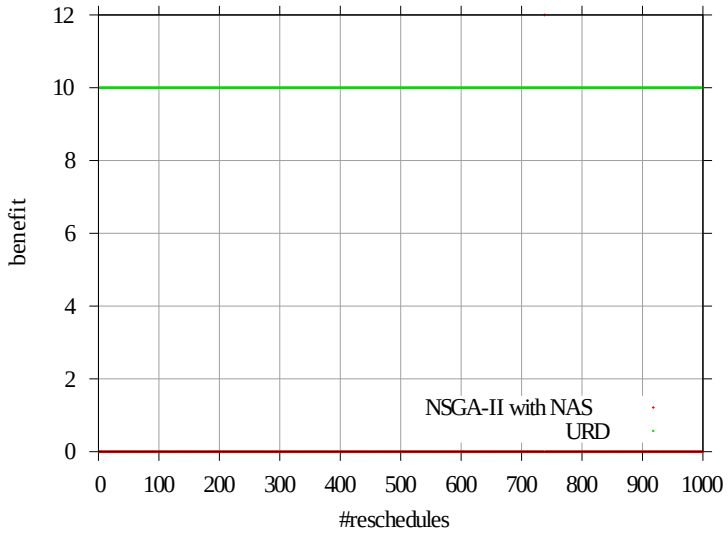

Figure 73. Benefit in Scenario 2 – NAS and URD (N=4, R_{MAX}=10)

6.3.1. Adaptation Strategy 1

AS1 is now under study in Scenario 2, when the maximum number of requests is 5.

a) N=128, R_{MAX}=320:

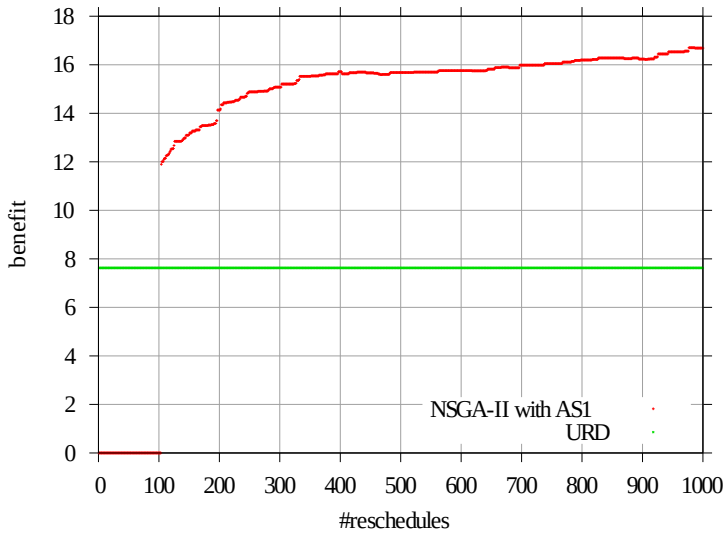

Figure 74. Benefit in Scenario 2 – AS1 and URD (N=128, R_{MAX}=320)

The figure above shows a fast increasing curve for AS1 strategy until the benefit reached is around 16, and then, a slower increasing. It can also be observed that the algorithm is not able to find the solutions for the first 100th TS.

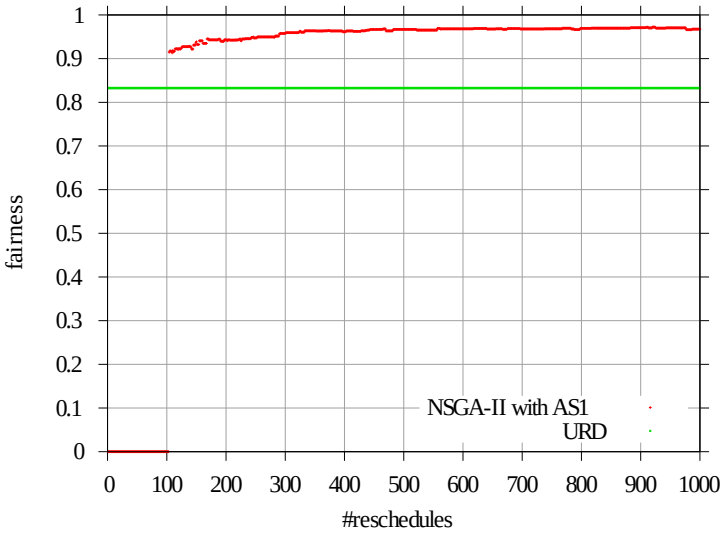

Figure 75. Fairness in Scenario 2 – AS1 and URD (N=128, R_{MAX}=320)

Figure 75 shows a similar tendency for the fairness, where the increasing is not so fast. In both benefit and fairness, the value obtained by AS1 is higher than for URD.

The average not guaranteed connections for AS1 is around 57%, while for URD is 67.00%. The execution time for the strategy under study is 58.85 ms, in contrast with the 14.35 µs needed by URD to do the allocation of resources. Although the execution time is much higher for AS1 than for URD, it still can guarantee a proper working of a real time scheduling.

b) N=64, R_{MAX}=160:

The results showed below, demonstrate that the strategy used is better than URD in terms of benefit and fairness, for this network configuration. Specifically, we can see that the benefit achieved by AS1 is the double than the one achieved by URD, when the number of TS is close to 1000. As in the previous configuration, the algorithm is not able to find the first solutions, but the adaptation strategy for initialization allows the method to find a solution in the 60th TS.

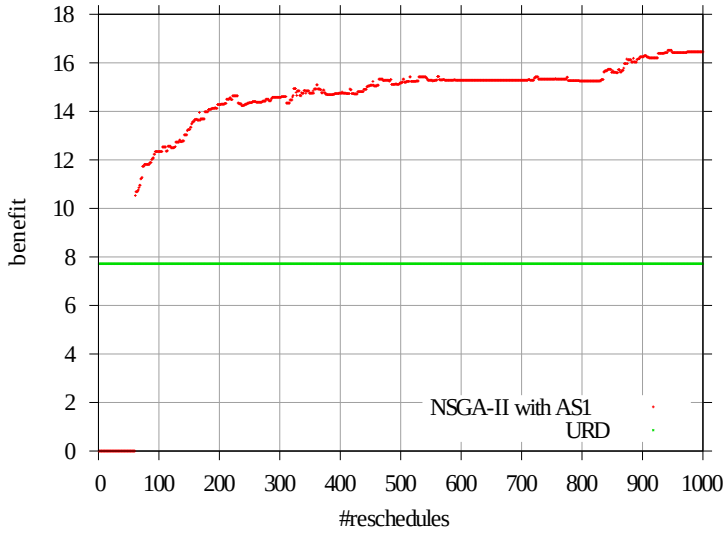

Figure 76. Benefit in Scenario 2 – AS1 and URD (N=64, R_{MAX}=160)

Not guaranteed connections percentage is around the 55% for AS1 and 66% for URD. While the execution time for URD is 8 µs, the average reached by AS1 is 20.64 ms.

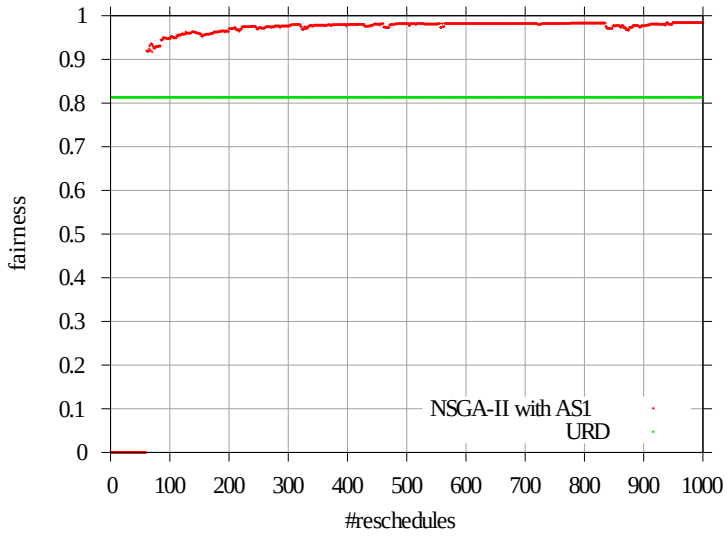

Figure 77. Fairness in Scenario 2 – AS1 and URD (N=64, R_{MAX}=160)

c) N=32, R_{MAX}=80:

The behaviour of both curves of benefit and fairness is very similar to the ones with a 64 user network, performing clearly better than URD. We can also see that it takes several TS to find the first solution.

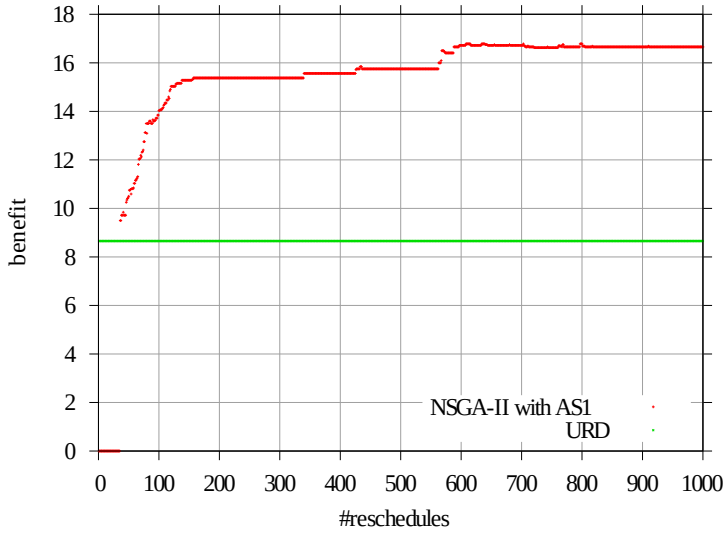

Figure 78. Benefit in Scenario 2 – AS1 and URD (N=32, R_{MAX}=80)

In this case, the percentage of not guaranteed connections is around the 51% for AS1 and 62% for URD. The execution times are 8.17 ms and 4.83 µs for AS1 and URD, respectively.

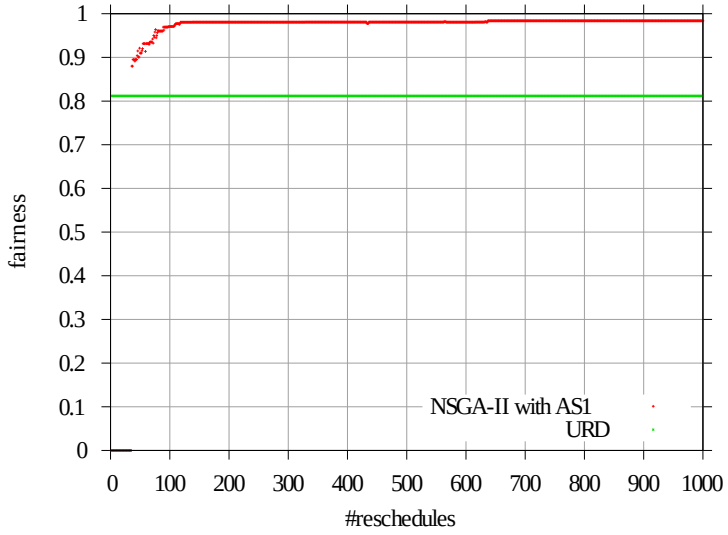

Figure 79. Fairness in Scenario 2 – AS1 and URD (N=32, R_{MAX}=80)

d) N=16, R_{MAX}=40:

The figure below shows that AS1 strategy is still better than URD in terms of benefit, even when the number of users decreases to 16. Nevertheless, we can see that the benefit gets stuck in a lower value than with 32 or more users. This problem would be probably overcome if the parameters of NSGA-II were adapted to this configuration.

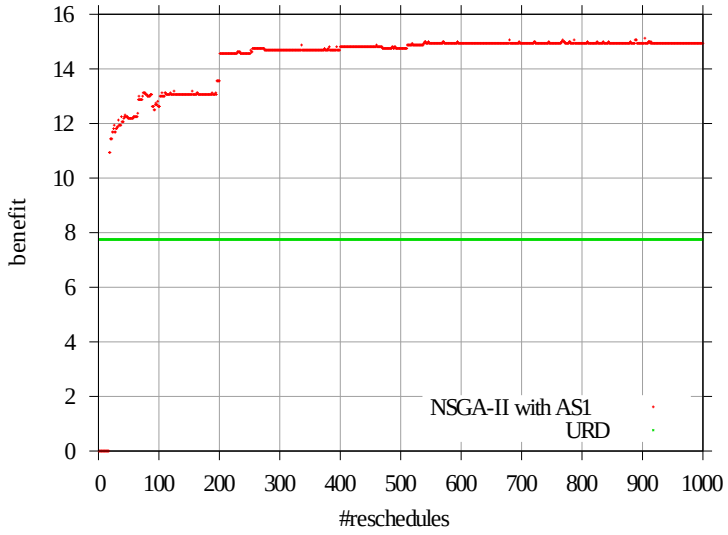

Figure 80. Benefit in Scenario 2 – AS1 and URD (N=16, R_{MAX}=40)

The next figure shows the results obtained with the fairness. It can also be observed the need of adapting the parameters in order to obtain better results.

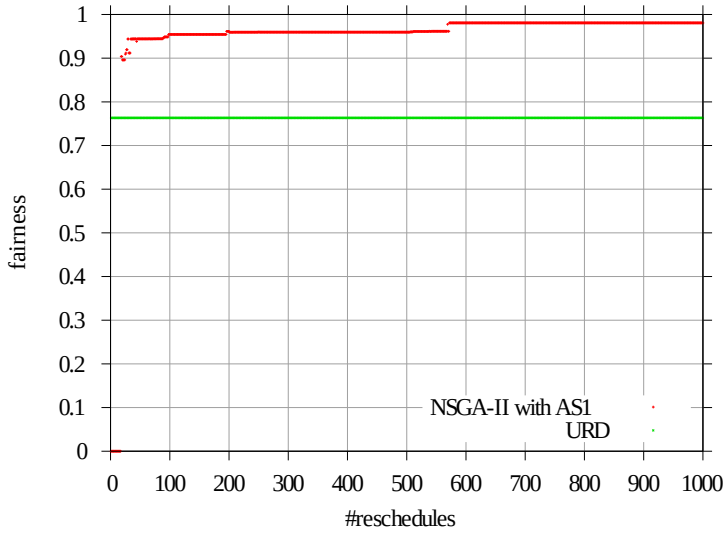

Figure 81. Fairness in Scenario 2 – AS1 and URD (N=16, R_{MAX}=40)

In this case, the not guaranteed connections are kept in 55.14% for AS1 and in 62% for URD. We notice that the difference between both percentages gets smaller when the number of users decreases. The execution times are 3.35 ms and 3.30 μs for AS1 and URD, respectively.

e) N=8, R_{MAX}=20:

Figure 82 and Figure 83 show, again, that the curves have several steps; they get stuck until it is possible to find a better solution for benefit and fairness, respectively. However, the results are still better than URD.

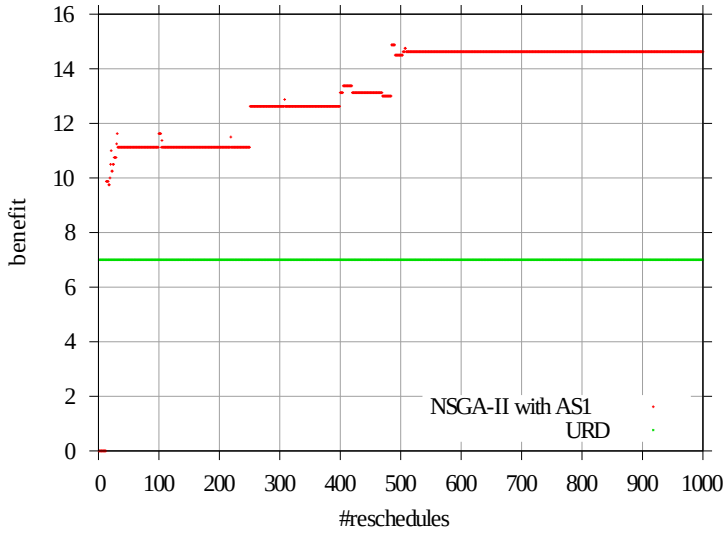

Figure 82. Benefit in Scenario 2 – AS1 and URD (N=8, R_{MAX}=20)

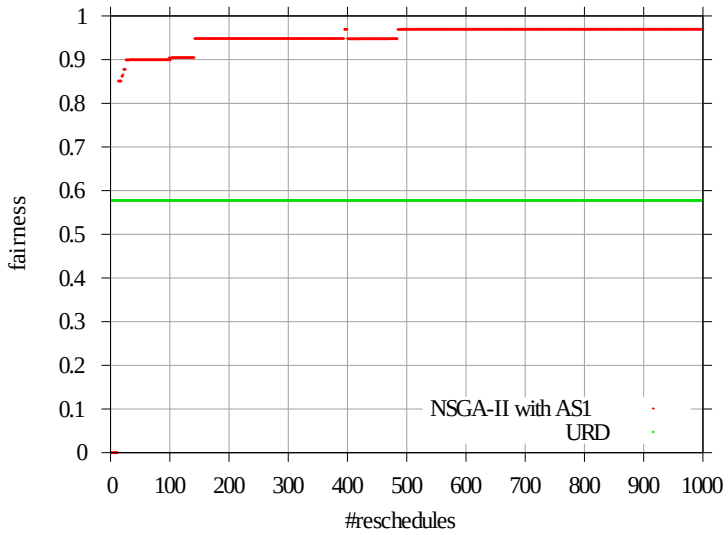

Figure 83. Fairness in Scenario 2 – AS1 and URD (N=8, R_{MAX}=20)

Also the not guaranteed connections percentage is better for AS1 rather than for URD, with 50.7% versus 60%. The execution times are 1.94 ms and 2.62 µs.

f) N=4, R_{MAX}=10:

Similar results are obtained with a network of 4 users. The following figures show the characteristics before mentioned. The percentage of not guaranteed connections

for AS1 is 52.68% in average, while for URD is 62%. The execution times in this case are 1.04 ms and 2.0 µs.

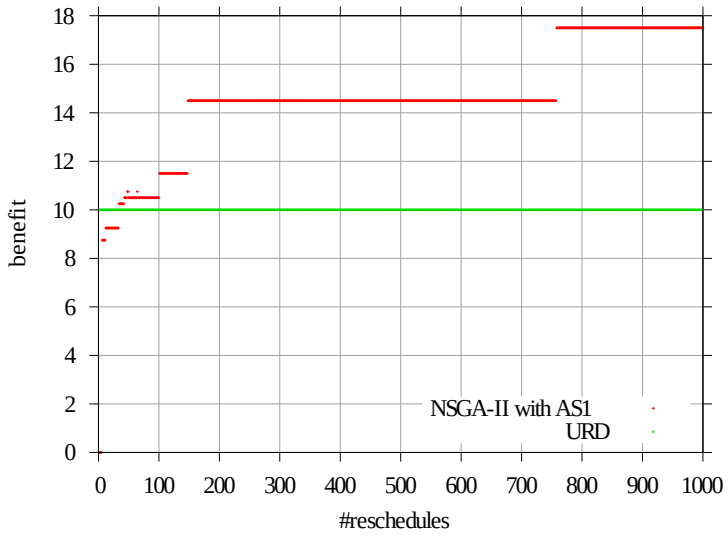

Figure 84. Benefit in Scenario 2 – AS1 and URD (N=4, R_{MAX}=10)

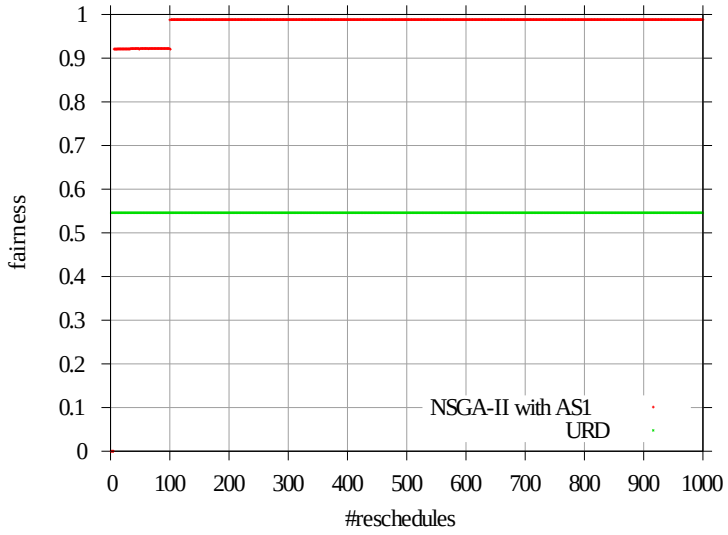

Figure 85. Fairness in Scenario 2 – AS1 and URD (N=4, R_{MAX}=10)

6.3.2. Adaptation Strategy 2

The same performance metrics are now analyzed when the strategies applied to do the scheduling are AS2a and AS2b, both using half a part of the previous results and half a random distribution to initialize the NSGA-II algorithm, as explained in 5.2.4.

a) N=128, R_{MAX}=320:

Both Figure 86 and Figure 87 show that the benefit and fairness have an increasing curve, when the strategy adopted is AS2a, while the curve for AS2b is

irregular and some points cannot be found. Nevertheless, both strategies perform higher values than URD.

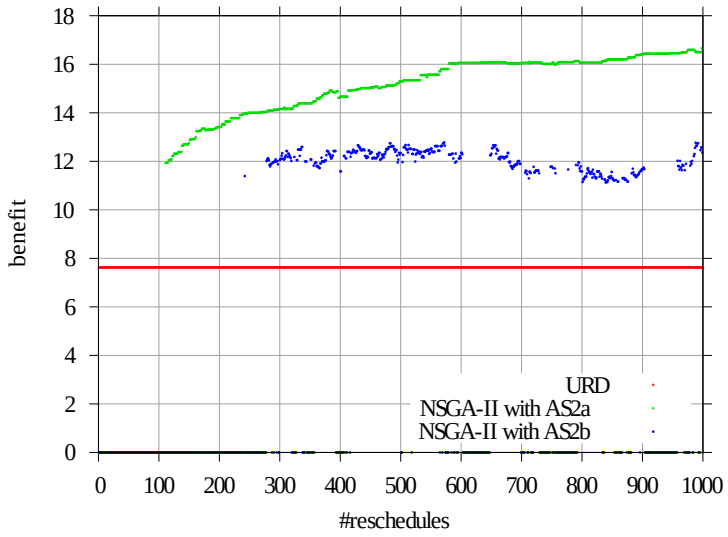

Figure 86. Benefit in Scenario 2 – AS2a, AS2b and URD (N=128, R_{MAX}=320)

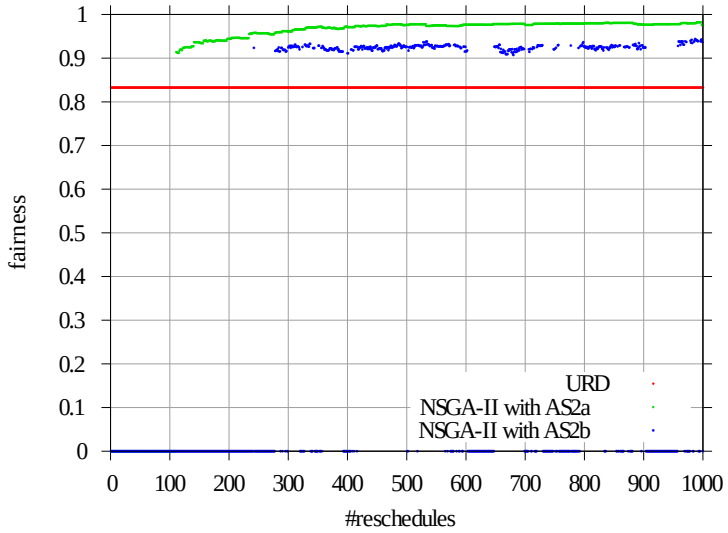

Figure 87. Fairness in Scenario 2 – AS2a, AS2b and URD (N=128, R_{MAX}=320)

While the average percentage of not guaranteed connections for URD is 67%, for AS2a and AS2b is, respectively, 57.67% and 59.93%. The execution times are 14.35 µs for URD, 59.77 ms for AS2a and 60.37 ms for AS2b.

b) N=64, R_{MAX}=160:

When the number of users decreases, the behaviour of AS2a and AS2b is very similar than for the last case. The main difference is that AS2b is able to find all the solutions once the algorithm is able to find the first one, in the 100th TS, and that it has a less smooth curve. The following figures show these facts.

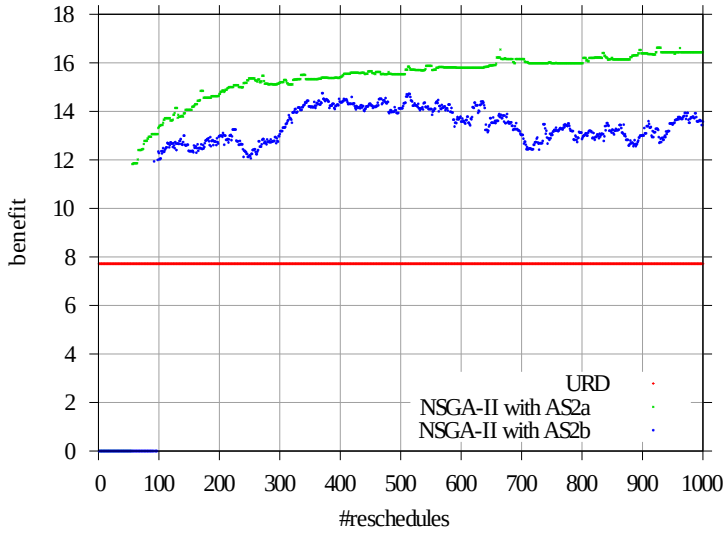

Figure 88. Benefit in Scenario 2 – AS2a, AS2b and URD (N=64, R_{MAX}=160)

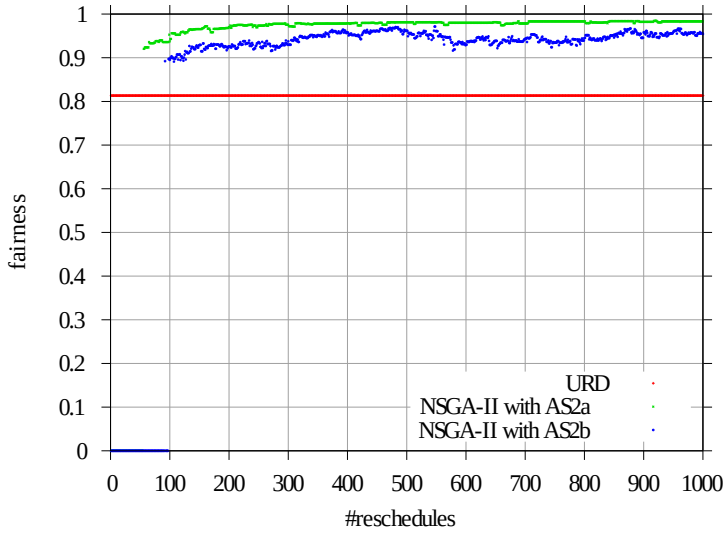

Figure 89. Fairness in Scenario 2 – AS2a, AS2b and URD (N=64, R_{MAX}=160)

The execution time is 21.62 ms for AS2a, 20.67 ms for AS2b and 8µs for URD. The values for not guaranteed connections are 54.72%, 60.03% and 66%, respectively.

c) N=32, R_{MAX}=80:

For this network configuration, both NSGA-II strategies perform similarly, and better than URD in terms of benefit and fairness, as it can be seen in the figures below.

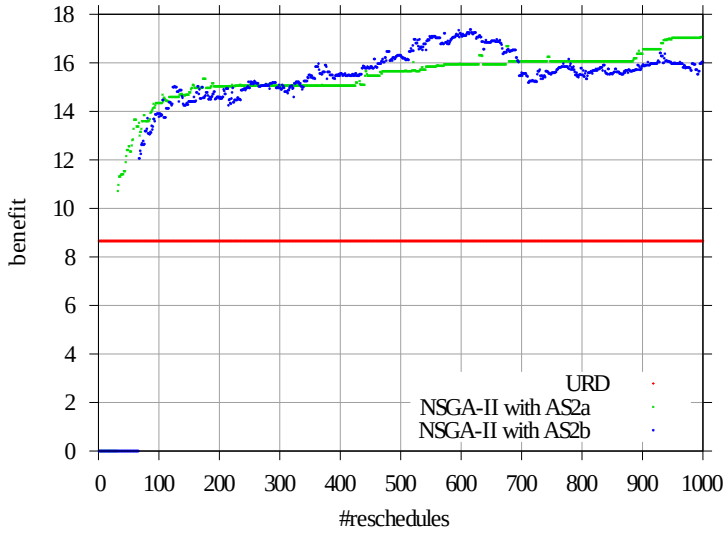

Figure 90. Benefit in Scenario 2 – AS2a, AS2b and URD (N=32, R_{MAX}=80)

The curves above show that the benefit achieved by AS2a and AS2b can take values which are twice the value obtained by URD.

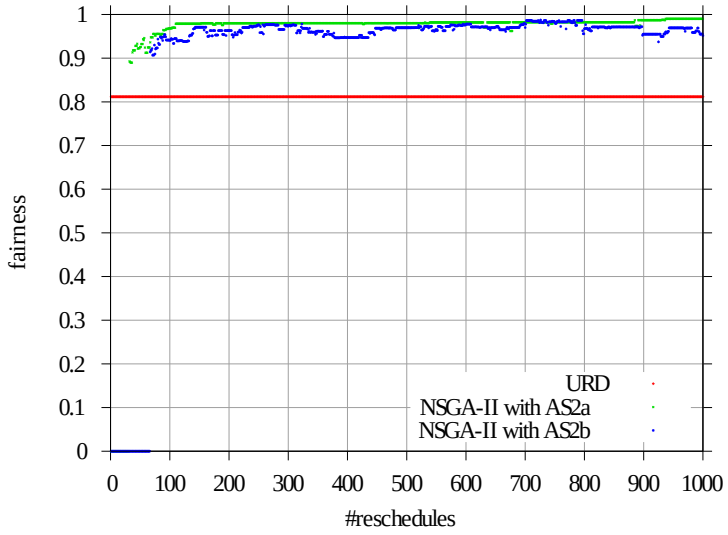

Figure 91. Fairness in Scenario 2 – AS2a, AS2b and URD (N=32, R_{MAX}=80)

Execution times and not guaranteed connections for AS2a, AS2b and URD are 8.29 ms and 52.60%, 8.36 ms and 55.64% and 4.80 µs and 62%, respectively.

d) N=16, R_{MAX}=40:

The same results are obtained in this case. We also see that the curves get stuck in a determined value and cannot increase, probably because the parameters of NSGA-II are not suitable. However, the values achieved are much better than the ones obtained by URD.

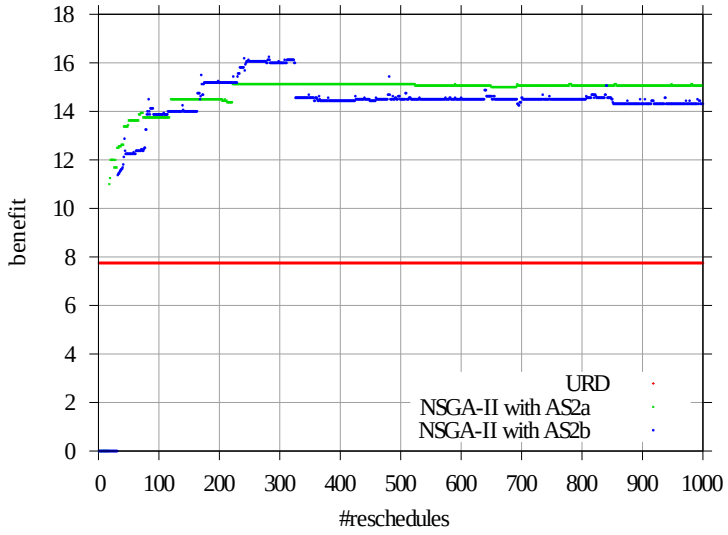

Figure 92. Benefit in Scenario 2 – AS2a, AS2b and URD (N=16, R_{MAX}=40)

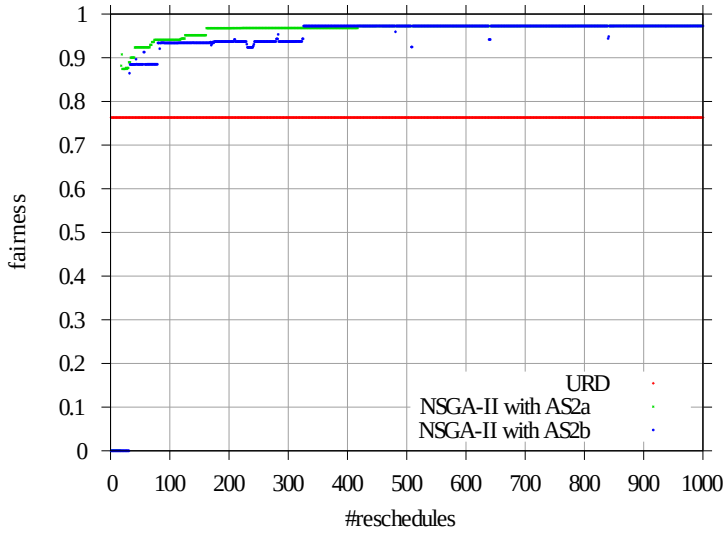

Figure 93. Fairness in Scenario 2 – AS2a, AS2b and URD (N=16, R_{MAX}=40)

The execution time for AS2a, AS2b are 3.41 and 4.3 ms, while for URD is 3.30 µs. Percentages of not guaranteed connections are 48.89, 51.85 and 62%.

e) N=8, R_{MAX}=20:

In this case, AS2a performs better than AS2b, and both, better than URD, in terms of benefit, fairness and not guaranteed connections, which take the value of 49.64, 50.72 and 60%, respectively.

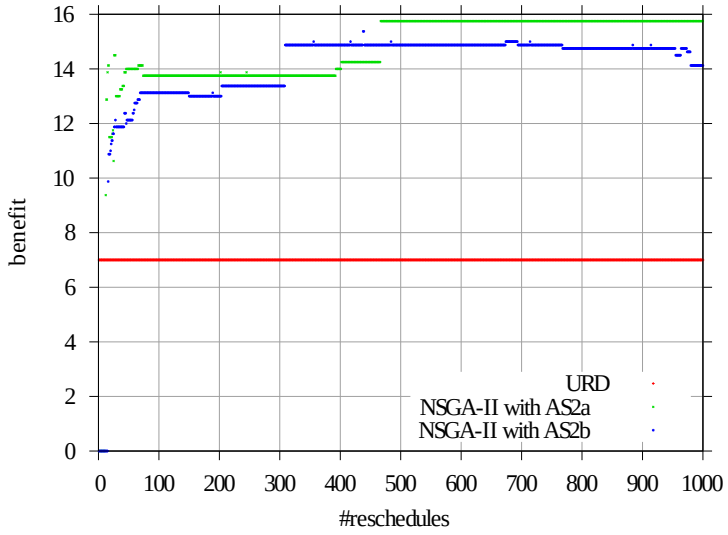

Figure 94. Benefit in Scenario 2 – AS2a, AS2b and URD (N=8, R_{MAX}=20)

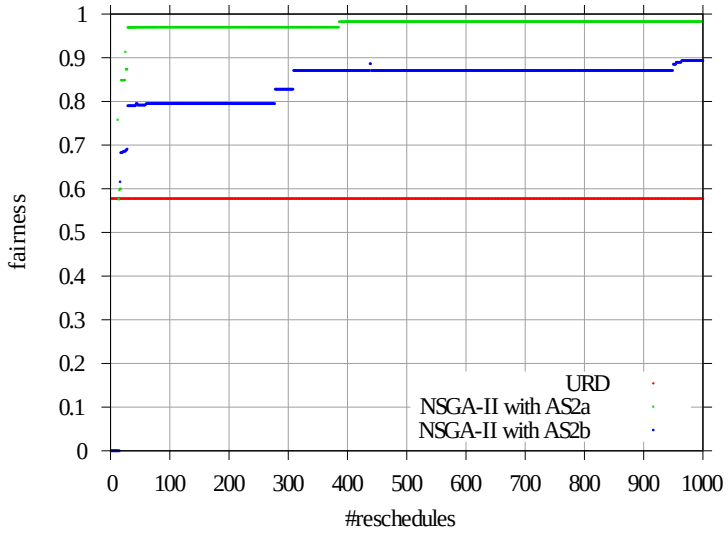

Figure 95. Fairness in Scenario 2 – AS2a, AS2b and URD (N=8, R_{MAX}=20)

The execution, as in the other cases, is much lower for URD, taking a value of 2.62 µs, versus the 1.88 ms for AS2a and the 1.74 ms for AS2b.

f) N=4, R_{MAX}=10:

In terms of benefit, AS2a performs better than AS2b, as it can be observed in the figure below. However, the fairness achieved is almost the same, maybe slightly higher for AS2a, although both are better than URD.

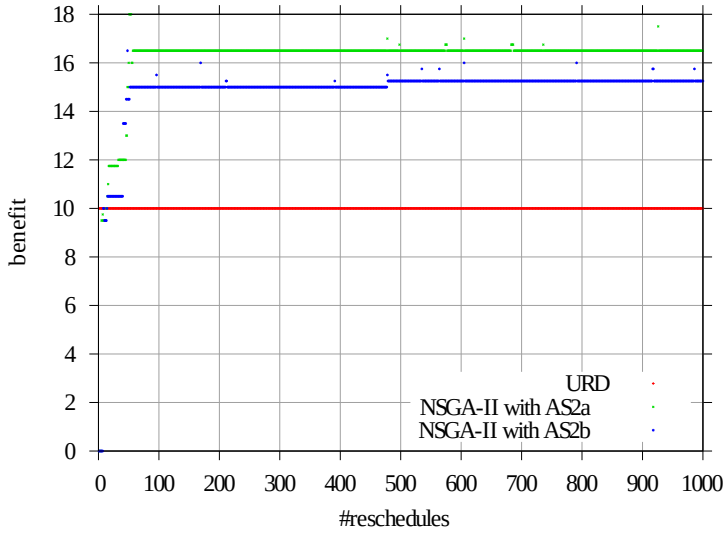

Figure 96. Benefit in Scenario 2 – AS2a, AS2b and URD (N=4, R_{MAX}=10)

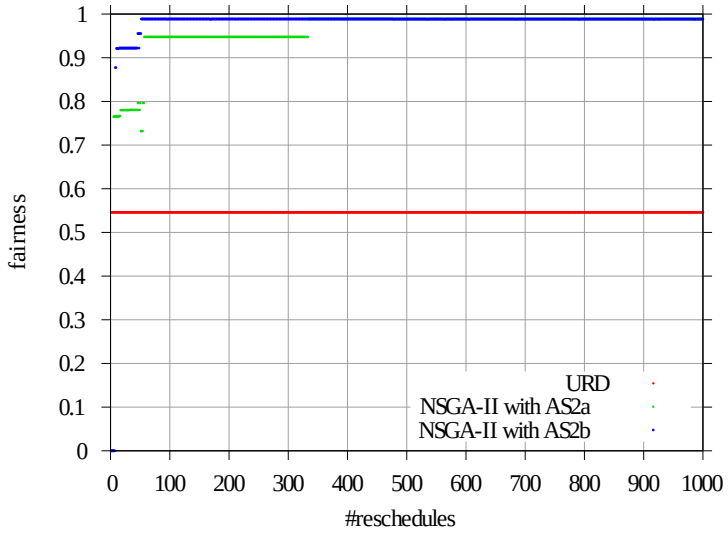

Figure 97. Fairness in Scenario 2 – AS2a, AS2b and URD (N=4, R_{MAX}=10)

The following table summarizes the values obtained for the execution time and not guaranteed connections.

Strategy	Average E. T.	Average N. G. C.
URD	2.00 µs	62.00%
AS2a	0.97 ms	49.64%
AS2b	1.14 ms	50.72%

Table 14. Execution Time and Not Guaranteed Connections in Scenario 2 – AS2a, AS2b and URD (N=4, R_{MAX}=10)

6.3.3. Comparison

As with Scenario 1, the simulation results obtained in the last sections are summarized in Table 15.

N	128		64		32		16		8		4	
	B	**F**	**B**	**F**	**B**	**F**	**B**	**F**	**B**	**F**	**B**	**F**
NAS	-	-	-	-	-	-	-	-	-	-	-	-
AS1	13.77	0.86	14.00	0.92	15.21	0.94	14.22	0.95	13.18	0.94	14.6	0.98
AS2a	13.53	0.86	14.62	0.92	15.02	0.95	14.59	0.95	14.68	0.96	16.22	0.96
AS2b	5.62	0.43	12.14	0.85	14.58	0.90	14.07	0.93	14.06	0.84	14.87	0.98
URD	7.62	0.83	7.72	0.81	8.65	0.81	7.75	0.76	7.00	0.58	10.00	0.54

Table 15. Benefit and Fairness in Scenario 2

When the number of maximum requests that can be done for each service class by each user is 5, the NAS strategy is clearly not valid. No solutions are found for none of the configurations. Better results could be obtained by changing the NSGA-II parameters (i.e., increasing number of generations and/or population size), but as happened with NAS in Scenario 1, the execution time would become too long to be suitable for a real time system.

In addition, the results show that, in this case, AS1, AS2a and also AS2b obtain better results in terms of benefit and fairness when the number of users is 64 or less. When the number of users is 128, only AS2b performs worse than URD, due to the difficulty of the strategy to find the first solution.

Figure 98, the convergence behaviour of these strategies can be observed. We see that strategy AS2b suffers such degradation that when the number of TS is approximately 20.000, the algorithm can no longer find a solution, and the benefit remains in zero. As happened in Scenario 1, both AS1 and AS2a behave in a very similar way. Their curves have an increasing performance, until reaching a value of 22.5 for AS2a, and 22.4 for AS1. The fairness values obtained are 0.9897 and 0.9839, respectively.

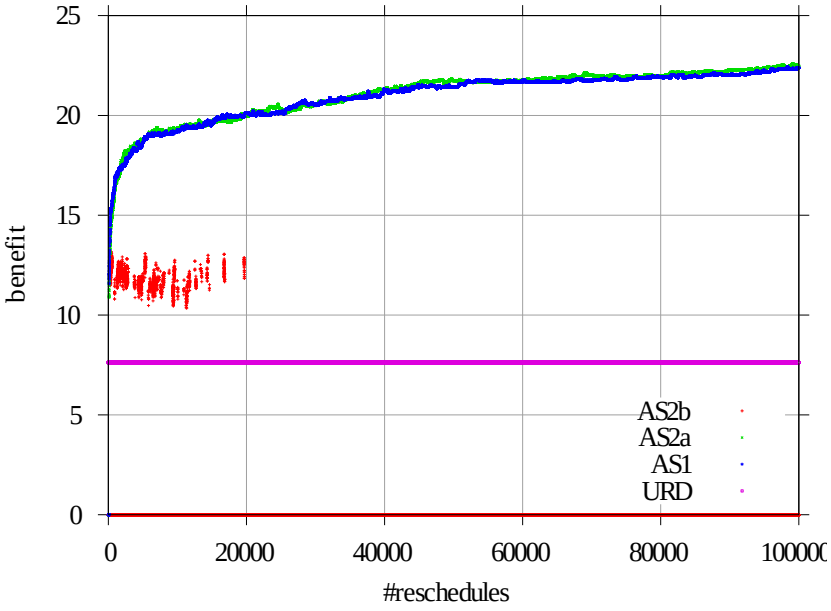

Figure 98. Convergence of Benefit in Scenario 2 (N=128, R_{MAX}=320)

The average time spent for each allocation is also very similar for both AS1 and AS2a. In the first case, 58.96 ms are necessary to calculate the X matrix for doing the allocation of resources, while in the second case, 59.17 ms are needed.

The average percentage of not guaranteed connections is, for AS1 of 60.8%, while for AS2a is 61.09%.

7. Conclusions and Future Work

In this Thesis, the NSGA-II algorithm has been used in order to solve the problem of multi objective optimization with constraints applied to PLC. Specifically, two objectives, the benefit, defined in 3.2.1, and the fairness, defined in 3.2.2, have been maximized in a multi user and multi service network in which the amount of resources is restricted. The solution obtained is the allocation of the different QoS requests from the different users that maximizes these objectives. The allocation of these requests should assure not only the required resources, but also enough speediness in its calculation in order to guarantee a right performance in real time. As the time complexity of NSGA-II is $O(GMN^2)$, the values of the number of generations G and the population size N (being M the number of objectives) have to be chosen in order to obtain the best solutions in a short enough execution time to be suitable for real time system.

The NSGA-II has been tested in two different scenarios, in which the maximum number of requests allowed for each service class by each user varies. In addition, three different strategies in the initialization of the NSGA-II algorithm have been tested.

No Adaptation Strategy (NAS), in which the initialization of the algorithm is random in every time slot, has showed not to be suitable to solve the problem. With this strategy, in most cases, the algorithm didn't have enough time to find a solution. The parameters of NSGA-II should be changed in order to find a solution, which means that even if the algorithm was capable of finding a solution, it would not be suitable for a real time system.

Adaptation Strategy 1 (AS1) and Adaptation Strategy 2 (with the variants AS2a and AS2b) use the solution obtained in the previous time slot to make the initialization of the NSGA-II algorithm for the current TS. The first strategy, as described in 5.2.3, only uses the values of the granted matrix X (after coding them into binary) to initialize the population, while the second one adds also some randomness when the initialization is done. Two different procedures to add this randomness are implemented, AS2a and AS2b, detailed in 5.2.4.

The results show that using the values from the previous allocations makes easier to NSGA-II algorithm to find a solution. This means that, for a given *G* and *N*, these adapted strategies achieve higher values of benefit and fairness, than the not adapted one, or even are able to find a solution after some TS, when the random initialization strategy is not able to.

In the simulations, it can be seen that AS1 and AS2a have very similar behaviour, in terms of benefit and fairness, and especially when the number of users is high, in both scenarios. These two strategies perform much better than AS2b, which is unstable and looses quality in the results obtained along the time.

However, when the number of users decreases, the performance of AS2a and, especially, of AS1 drops off, as the NSGA-II parameters were chosen to assure good results in a network of 128 users. In Scenario 1, we have seen that even the Uniform Resource Distribution strategy gets higher values of benefit and fairness when the number of users is lower than 8. To avoid this, new values for the number of generations, population size and probabilities of cross-over and mutation should be chosen depending on the configuration of the network. Increasing *G* and *N* when the number of users gets smaller, but assuring an execution time lower than 1 TS, could improve the performance of the AS1 and AS2a NSGA-II strategies in terms of benefit and fairness. Thus, further study could be done to find a method that adapts not only the initialization of NSGA-II, but also its parameters depending on the network configuration.

References

[1] Wu, Chiung-Shen, Ma, Gin-Kou y Lin, Bao-Shuh P., "A Cell Scheduling Algorithm for VBR Traffic in ATM Multiplexer." *Global Telecommunications Conference, IEEE, Vol.1, Page(s) 632-637.* November, 1995.

[2] Wang, Song, Wang, Yu-Chung y Lin, Kwei-Jay., "A Priority-Based Weighted Fair Queueing Scheduler for Real-Time Network." *Real-Time Computing Systems and Applications. Sixth International Conference. Page(s) 312 - 319.* December, 1999.

[3] Golestani, S.J., "A Self-Clocked Fair Queueing Scheme for Broadband Applications." *Networking for Global Communications, 13th Proceedings IEEE, Vol. 2, Page(s): 636-646.* June, 1994.

[4] Phu Do, Le and Lehnert, Ralf., "Scheduling Strategies for Service Admission in Powerline Communication Access Networks with QoS Support." *Power Line Communications and Its Applications, IEEE International Symposium, Page(s) 41-46.* April, 2008.

[5] Tecnocom., "Power Line Communications." 2005.

[6] The OPERA Consortium., "First draft of the OPERA specification version 2." June, 2007.

[7] , HD-PLC. *HD-PLC.* [Online] http://www.hd-plc.org/.

[8] Elliott, Robert., "A Measureof Fairness of Service for Scheduling Algorithms in Multiuser Systems." *Canadian Conference on Electrical and Computer Engineering, IEEE, Vol.3, Page(s) 1583-1588.* August, 2002.

[9] Jain, Lakhmi and Abraham, Ajith., "Evolutionary Multiobjective Optimization."

[10] Radhakrishnan, Alamelu., "Evolutionary Algorithms for Multiobjective Optimization with Applications in Portfolio Optimization." *Graduate Faculty of North Carolina State University.* March, 2007.

[11] Kunkle, Daniel., "A Summary and Comparison of MOEA Algorithms." *Northeastern University in Boston, Massachusetts.* May, 2005.

[12] Horn, Jeffrey, Nafpliotis, Nicholas and Goldberg, David E., "A Niched Pareto Genetic Algorithm for Multiobjective Optimization." *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence. Vol.1, Page(s): 82-87.* June, 1994.

[13] Zitzler, Eckart, Deb, Kalyanmoy and Thiele, Lothar., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results." *Evolutionary Computation, Vol. 8, Issue 2, Page(s) 173-195.* June, 2000.

[14] Erickson, Mark, Mayer, Alex and Horn, Jeffrey., "The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems." *Evolutionary Multi-Criterion Optimization.* s.l. : Springer Berlin / Heidelberg, 2001.

[15] Srinivas, N. and Deb, K., "Multiobjective function optimization using nondominated sorting genetic algorithms." *Evolutionary Computation, Vol. 2, Page(s) 221-248.* 1995.

[16] Deb, Kalyanmoy, et al., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation, Vol.6, Issue 2, Page(s) 182-197.* April, 2002.

[17] Corne, David W., Knowles, Joshua D. and Oates, Martin J., "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization." *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Page(s) 839-848.* September, 2000.

[18] Zitzler, Eckart and Thiele, Lothar., "An evolutionary algorithm for multiobjective optimization: The strength pareto approach." *Technical report, Computer Engineering and Communication Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH).* 1998.

[19] Laumanns, M., Zitzler, E. and Thiele, L., "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." *Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.* May, 2001.

[20] Knowles, Joshua D. and Corne, David W., "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy." *Evolutionary Computation, Vol. 8, Page(s) 149-172.* 2000.

[21] Jensen, Mikkel T., "Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms." *IEEE transactions on Evolutionary Computation, Vol. 7, Page(s) 503-515.* 2003.

[22] Deb, Kalyanmoy., "Kanpur Genetic Algorithms Laboratory." *Multi-objective NSGA-II code in C. Original Implementation (for Windows and Linux).* [Online] http://www.iitk.ac.in/kangal/codes/nsga2/nsga2code.tar.

Appendix

Initialization routines

Here are the routines used for doing the initialization of the NSGA-II algorithm used for the different strategies described in 5.2. Figure 99 shows the structs with the fields used to manage the individuals and the population. Figure 100 contains the original implementation for initialization. Figure 101, Figure 102 and Figure 103 show the implemented routines for the different adapted initialization strategies, AS1, AS2a and AS2b, respectively.

```
typedef struct /*individual properties*/
{
       int    genes[maxchrom],      /*binary chromosome*/
              rank,                 /*Rank of the individual*/
              flag;                 /*Flag for ranking*/
       float  xreal[maxvar], /*list of real variables*/
              xbin[maxvar];  /*list of decoded value of the chromosome */
       float  fitness[maxfun],      /*Fitness values */
              constr[maxcons],      /*Constraints values*/
              cub_len,              /*crowding distance of the individual*/
              error;                /* overall constraint violation for the
       individual*/
}individual;          /*Structure defining individual*/

typedef struct
{
       int    maxrank;              /*Maximum rank in the population*/
       float  rankrat[maxpop];      /*Rank Ratio*/
       int    rankno[maxpop];       /*Individual at different ranks*/
       individual    ind[maxpop],   /*Different Individuals*/
                     *ind_ptr;
}population ;  /*Population Structure*/
```

Figure 99. Individual and Population structs. Original implementation from [22]

```
void init(population *pop_ptr)
{
	int i,j,r;
	float d;
	pop_ptr->ind_ptr = &(pop_ptr->ind[0]);

	/*Loop Over the population size*/
	for (i = 0 ; i < popsize ; i++)
	{
		/*Loop over the chromosome length*/
		for (j = 0;j < chrom;j++)
		{
			/*Generate a Random No. if it is less than 0.5 it
generates a 0 in the string otherwise 1*/
			d = randomperc();
			if(d >= 0.5) pop_ptr->ind_ptr->genes[j] = 1;
			else pop_ptr->ind_ptr->genes[j] = 0;
		}
		pop_ptr->ind_ptr = &(pop_ptr->ind[i+1]);
	}
	pop_ptr->ind_ptr = &(pop_ptr->ind[0]);
	return;
}
```

Figure 100. Routine for NAS initialization. Original implementation from [22]

```
void init_G1(population *pop_ptr,population *last_pop_ptr)
{
	int i,j;
	float d;

	for(i=0;i<popsize;i++)
	{
		pop_ptr->ind_ptr = &(pop_ptr->ind[i]);
		last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[i]);
		/*Loop over the chromosome length*/
		for (j = 0;j < chrom;j++)
		{
			pop_ptr->ind_ptr->genes[j]   =   last_pop_ptr->ind_ptr-
>genes[j];
		}
	}
	pop_ptr->ind_ptr = &(pop_ptr->ind[0]);
	last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[0]);

	return;
}
```

Figure 101. Routine for AS1 initialization

```
void init_G2a(population *pop_ptr,population *last_pop_ptr)
{
	int i,j;
	float d;

	for(i=0;i<popsize;i++)
	{
		pop_ptr->ind_ptr = &(pop_ptr->ind[i]);
		last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[i]);
		/*Loop over the chromosome length*/
		for (j = 0;j < chrom;j++)
		{
			if((j%2)!=0)	pop_ptr->ind_ptr->genes[j]	=
last_pop_ptr->ind_ptr->genes[j];
			else
			{
				d = randomperc();
				if(d >= 0.5)
				{
					pop_ptr->ind_ptr->genes[j+1] = 1;
				}
				else
				{
					pop_ptr->ind_ptr->genes[j+1] = 0;
				}
			}
		}
	}
	pop_ptr->ind_ptr = &(pop_ptr->ind[0]);
	last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[0]);

	return;
}
```

Figure 102. Routine for AS2a initialization

```
void init_G2b(population *pop_ptr,population *last_pop_ptr)
{
	int i,j;
	float d;

	for(i=0;i<popsize;i++)
	{
		pop_ptr->ind_ptr = &(pop_ptr->ind[i]);
		last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[i]);
		if((i%2)!=0)
		{
			/*Loop over the chromosome length*/
			for (j = 0;j < chrom;j++)
				pop_ptr->ind_ptr->genes[j]    =    last_pop_ptr-
			>ind_ptr->genes[j];
		}
		else
		{
			for (j = 0;j < chrom;j++)
			{
				d = randomperc();
				if(d >= 0.5) pop_ptr->ind_ptr->genes[j+1] = 1;
				else pop_ptr->ind_ptr->genes[j+1] = 0;
			}
		}
	}
	pop_ptr->ind_ptr = &(pop_ptr->ind[0]);
	last_pop_ptr->ind_ptr = &(last_pop_ptr->ind[0]);

	return;
}
```

Figure 103. Routine for AS2b initialization