A GRASP algorithm using RNN for solving dynamics in a P2P live video streaming network

Marcelo Martínez Alexis Morón Franco Robledo Pablo Rodríguez-Bocca
Héctor Cancela
Instituto de Computación
Facultad de Ingeniería de la Universidad de la República
Julio Herrera y Reissig 565, 11300, Montevideo, Uruguay
{marcelo.martinez,alexis.moron}@tcs.com,
{frobledo,prbocca,cancela}@fing.edu.uy

Gerardo Rubino
IRISA/INRIA, Campus de Beaulieu Rennes 35042 CEDEX, France
rubino@irisa.fr

Abstract

In this paper, we present an algorithm based on the GRASP metaheuristic for solving a dynamic assignment problem in a P2P network designed for sending real-time video over the Internet. In a highly dynamic P2P topology, the frequent connections and disconnections of nodes are the main obstacle we face when trying to offer a high Quality-of-Experience (QoE) to clients. We first introduce the P2P network architecture where this node dynamics occurs. This architecture employs a multi-source streaming approach where the stream is decomposed into several flows sent by different peers to each client, including some level of redundancy, in order to cope with the fluctuations in network connectivity. Then, we present the GRASP-based algorithm developed in order to tackle the problem of maintaining connectivity in presence of node dynamics by periodically reassigning network connections; these assignments are performed so as to maximize the global expected QoE, calculated using the recently proposed PSQA methodology. Additionally, we provide a variation of the GRASP-based algorithm, based on the Random Neural Network model. Finally, we show the results obtained when these algorithms are applied to a case study based on real life data.
Keywords: GRASP, RNN, P2P, QoE, PSQA methodology.

1. Introduction

Multimedia systems are now ubiquitously present in the Internet, and employ many different architectures, depending on their sizes and on the popularity of their contents. Most of them have a traditional CDN (Content Delivery Network) structure [3, 11], where a set of datacenters concentrates the task of distributing the content to the customers[1]. Another popular alternative consists of using the often idle capacity of the clients to share the video distribution with the servers, through the present mature Peer to Peer (P2P) systems. These are virtual networks developed at the application level over the Internet infrastructure. The nodes in the network, called peers, offer their resources to the other nodes, basically because they all share common interests. As a consequence, as the number of customers increases, the same happens with the global resources of the P2P network. In this paper, we are interested in some aspects related to the use of a P2P architecture to distribute live video. Using a P2P infrastructure for video distribution looks like a good idea due to the high requirements in terms of bandwidth of these applications. The main problem is how to provide good quality levels in a context where this quality depends on other clients that are delivering the stream, and given the fact that users connect and disconnect very frequently in an autonomous and completely asynchronous way. In a typical situation, there are a lot of peers that have enough bandwidth to serve other peers. Each peer has its particular behavior of connection/disconnection; therefore an intelligent assignment, between source peers and target peers, has to be done to maximize the quality in the overall network (mitigating the loss effect of peer disconnections). The number of possible assignments grows exponentially with the

[1]This is the case of msnTV, YouTube and Jumptv.

number of nodes in the network. To determine the best assignment is a complex combinatorial optimization problem.

Multi-source Streaming. Since we are considering a P2P network with bandwidth-limited clients which must also act as servers, we consider the following architecture for the video stream distribution: the original stream, which can be seen as a sequence of frames, is split into several substreams, each one containing only some of the frames of the original one. In this way, each substream requires less bandwidth to be transmitted, and upstream-limited clients are able to send it. However, by using this distribution method, each client that receives the video must obtain the different substreams from various sources, and then recombine them in order to obtain the original stream to be visualized. Then, a client that receives a certain substream can act as a server and send it to other clients if it has enough upstream bandwidth. A client can visualize the content even if it is not receiving *all* of the substreams, but in this case, some frames[2] will be missing and therefore video quality will be impaired. This multi-source streaming technique was designed for an optimal Quality-of-Experience using real dynamics peers' behavior [2], and a full implementation is available [10].

Quality Measurements. The Pseudo Subjective Quality Assessment (PSQA) [8] is a technique allowing to automatically approximate the value obtained from a subjective test. The idea is to have several distorted samples evaluated subjectively, and then to use the results of this evaluation to train a specific learning tool (in PSQA the best results come from using Random Neural Networks [4]) in order to capture the relation between the parameters that cause the distortion and the perceived quality. In this work, the affecting quality parameters considered are the proper reception of each substream.

The paper is organized as follows. Section 2 introduces the assignment problem, which contemplates the P2P dynamics and tries to maximize the expected overall perceived quality. A GRASP metaheuristic is presented in Section 3 to solve the problem and in Section 4 an algorithmic improvement based on an RNN model is presented. In Section 5 some experimental results are introduced along with the main contributions of this work.

2. P2P Robust Assignment Model

The considered system broadcasts a single video stream over a population of terminals, in a P2P manner.

[2]Not all frames have the same relevance in the stream, the absence of some frames could pass unnoticed by the viewer, while other frames missing could cause artifacts or other kinds of problems in the video.

Time. The system is reconfigured at discrete points in time, every Δt. Let us use Δt as unit of time, and denote by I_n the nth interval $(t_n, t_{n+1}]$ for each n.
Distribution scheme. The stream is originally sent to the clients (or terminals) by a broadcaster node s. Some clients act also as servers, relaying streams to other clients. For this purpose, each node v has an available output bandwidth BW_v^{out}. The system distributes a single stream of live video by means of K sub-streams denoted by $\sigma_1, \sigma_2, ..., \sigma_K$. Each substream σ_k is sent with a constant bandwidth bw_k.
Dynamics. The evolution of the system from t_n to t_{n+1} is as follows: some nodes leave in I_n, possibly disconnecting other clients in some substreams; at the same time, some nodes enter the network requesting for connection; they remain isolated from the rest of the nodes until t_{n+1} when the new reconfiguration action is performed. The goal of the reconfiguration is to reconnect the disconnected nodes and to connect the new arrivals to the network. The connection scheme always builds trees of peers. At time $(t_{n+1})^-$, just before the reconfiguration, the general situation is the following. For each substream σ_k there is a *main tree*, $\mathcal{P}_k$, containing (at least) the source s; all its nodes receive substream σ_k. There are also $M_k \geq 0$ other trees, disjoint between them and with $\mathcal{P}_k$, denoted by $\tau_{k,1}, \tau_{k,2}, \cdots, \tau_{k,M_k}$; their nodes do not receive σ_k. The set of trees associated with substream σ_k is called *plane* k. A *perfect network* is a system where for each substream σ_k there is only one directed tree ($\mathcal{P}_k$, the main one), meaning that $M_k = 0$.
Optimization. The reconfiguration action will try to search among the huge number of possible perfect networks, in order to build a robust one. For this purpose, we keep statistics about the nodes' behavior allowing us to evaluate their expected departure times from the network. Specifically, we maintain an estimate p_i of the probability that node i remains connected in the current period, when it is connected at the beginning.
Formal Model. In [1], an Integer Mathematical Programming Model which describes this assignment problem is introduced.

3. Algorithmic solution based on GRASP

GRASP [9] is a well known metaheuristic that has been successfully used to solve many hard combinatorial optimization problems. It is an iterative process which operates in two phases: in the *Construction Phase* an initial feasible solution is built by means of a greedy algorithm. This algorithm must also have a random component, so that different executions lead to different results. The neighborhood of this solution is then explored in the *Local Search Phase*.

Next, we present an algorithm based on a customized GRASP heuristic to solve our problem. As we have seen in the formal model description, the time line is divided into intervals of the form $I_n = [t_n, t_{n+1})$. In this way, we simplify the problem by considering only a discrete number of points $(t_0, t_1, t_2, ...)$ on the time line.

At the beginning of each interval I_n (we will call this instant t_n^-), the state of the network is the result of all the changes which have occurred in the previous interval, I_{n-1}. During this period, some nodes may have left the network (willingly or not), while new clients may have requested connection in order to receive the video stream. Therefore, new disconnected trees may have appeared. At this instant t_n our GRASP-based algorithm is executed in order to connect the new nodes and disconnected trees to the main tree. This has to be done in such a way that the expected quality is maximized, taking into account the upstream bandwidth restrictions of each node. We assume an instantaneous execution of the algorithm at t_n, so that at instant t_n^+ we will have our reconfigured system[3]. Then, new connections and disconnections will occur during the interval I_n, and the algorithm will be executed again at the start of I_{n+1}. The output of the algorithm is a set of *assignments*, where each assignment indicates to which node of the main tree we will attach a disconnected tree (or a new node[4]) in order to enable transmission of a certain substream to that tree. An assignment can then be seen as a triplet (parent, child, substream), where the *child* is the root of the disconnected tree which will be attached to a node of the main tree (the *parent*) to receive the corresponding substream. As stated before, the algorithm should produce a solution which maximizes the global perceived quality. However, we have seen that this is an extremely difficult problem, so the algorithm will try to construct a solution as close to an optimal one as possible.

In the following subsections, we describe the GRASP-based algorithm which is used to reassign connections in the network after a certain interval of time.

3.1 Construction phase

The *Construction phase* of the algorithm, outlined in Figure 1, consists of using a greedy and randomized procedure to construct an *initial solution* for the problem.

When the procedure $\mathsf{GRASP_{p2p}}$ is executed with the initial state of the network g_0, i_{max} initial solutions are constructed (lines 1 - 3) using a procedure $\mathsf{Construction_{p2p}}$ (this procedure has a random component, therefore n executions will lead to n different results). Then, from the i_{max} solutions obtained, we choose the one which gives the network the better global perceived quality in the following interval (lines 4 - 7). To evaluate the global perceived quality in the following interval for each solution, we use the procedure $\mathsf{PSQA_{expected}}$, which uses statistical data about the nodes in order to "predict" the connections and disconnections in the future interval and see how a given solution is affected by these events. We will now examine the procedure used to construct each one of these initial solutions, $\mathsf{Construction_{p2p}}$: first of all, a list of all possible assignments is determined given the current state of the network: that is, a list of all possible (parent, child, substream) triplets is constructed to determine which trees need to be reconnected to the main tree and which nodes have enough upstream bandwidth available in order to transmit a certain substream to a disconnected tree (line 1). Then, a *Restricted Candidate List* or RCL is generated using the last procedure of Figure 1. First, this procedure selects the *greedy* function AIC (*Assignment Improvement Criteria*) (line 1) that compares candidates. Then, we compute the AIC function for all candidates (lines 3-6) and finally a subset of size $cfg.rclSize$ is returned which contains those candidates with higher computation values of the AIC function. In this paper, we developed three AIC functions: *BW* AIC which evaluates the amount of available bandwidth that one assignment will provide to the *main* tree for future assignments, *Current* AIC which evaluates the value of quality increase of an assignment at the current time, and *Future* AIC which evaluates the value of the expected quality increase of an assignment for the next iteration[5]. Once the RCL is generated, one of these candidate assignments is randomly selected from the RCL (line 5) and added to a set A (line 6), this constitutes the *random* component of the algorithm. The list of candidates is then updated (line 7), since the tree corresponding to the selected assignment will no longer be disconnected, and the upstream bandwidth of the node to which that tree will be connected will have decreased. This process is repeated until the set A satisfies that all the disconnected trees have been reassigned to the main tree for each substream or no node in the main tree has enough bandwidth to transmit a substream to a disconnected tree.

The set A of assignments is then our *initial solution.*

3.2 Local search phase

Our customized GRASP algorithm does not contain the traditional local search phase. In the local search phase, the solution built in the construction phase is improved by exploring its neighborhood in order to find a local optimum. In the construction phase of our algorithm, a feasible solution is built applying successives individual *as-*

[3]In the current implementation, this "instantaneous execution" means that the execution time of the algorithm has to be negligible compared to the duration of each interval I_n.

[4]New nodes can be seen as disconnected trees with only one element, so there is no need to make a distinction.

[5]This evaluation is done generating different possible future states of the network including the candidate (using p_i to draw which nodes will leave the network in the next iteration) and finally calculates the average of quality increase of all these possible network future states. The number of possible future networks generated is a configurable parameter.

```
Procedure GRASP$_{p2p}$
Input: $g_0$

1:  $q^* \leftarrow -1$
2:  for $i = 1$ to $i_{max}$ do
3:      $g_i \leftarrow$ Construction$_{p2p}(g_0)$
4:      $q_i \leftarrow$ PSQA$_{expected}(g_i)$
5:      if $(q_i > q^*)$ then
6:          $q^* \leftarrow q_i$
7:          $g^* \leftarrow g_i$
8:      end if
9:  end for

10: return $g^*$
```

```
Procedure Construction$_{p2p}$
Input: $g$

1:  $C \leftarrow$ FindAllAssignmentsCandidates$(g)$
2:  $A \leftarrow \emptyset$
3:  while $C \neq \emptyset$ do
4:      $C_{RCL} \leftarrow$ GenerateRCL$_{p2p}(g, C)$
5:      $a \leftarrow$ SelectRandom$(C_{RCL})$
6:      $A \leftarrow A \cup \{a\}$
7:      $C \leftarrow$ UpdateCandidates$(g, a, C)$
8:  end while
9:  $g_c \leftarrow$ ApplyAssignments2Graph$(g, A)$

10: return $g_c$
```

```
Procedure GenerateRCL$_{p2p}$
Input: $g, cfg, C$

1:  AIC $\leftarrow cfg.method$
2:  $A \leftarrow \emptyset$
3:  for each $a \in C$ do
4:      $val \leftarrow$ AIC$(g, cfg, a)$
5:      $A \leftarrow$ Insert$(A, a, val)$
6:  end for
7:  $R \leftarrow$ Subset$_{best}(A, cfg.rclSize)$

8:  return $R$
```

Figure 1. GRASP-based algorithm

signments to the initial network configuration. Let $\mathcal{T} = \{a_1, a_2, ..., a_k, ..., a_n\}$ be the set of assignments applied to build a feasible solution. In this case, for the local search phase, the choice of the neighborhood structure should be one where small variations on the assignments of $\mathcal{T}$ produce feasible solutions. The main problem here is that a particular assignment is strongly dependent on the previous assignment. In this sense, if we want to change the assignment a_k, there is a chance that next assignments $a_{k+1}, ..., a_n$ could not be applied. This is easily seen when, for example, the assignment a_{k+1} involves as a parent, one of the new available parent nodes added in the previous assignment a_k. A local search phase is not useful in this specific problem, because it introduces a new complex combinatorial problem. Therefore, we will not implement this phase.

In this paper, we propose an alternative way to improve the GRASP-based algorithm that consists on replacing the lines 4-5 of the procedure Construction$_{p2p}$, in a random fashion, with a new procedure based on the RNN model, which may find better solutions.

4. Algorithm improvement using RNN

The Random Neural Network (RNN) is a model introduced by Gelenbe [5], which has been applied with success to different optimization problems such as the *minimum vertex covering problem* [6] and the *traveling salesman problem* [7]. In the RNN model, signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation, and negative signals represent inhibition. Each neuron's state is a nonnegative integer called its *potential*, which increases when an excitation signal arrives to it and decreases when an inhibition signal arrives. Thus, an excitatory spike is interpreted as a "+1" signal at a receiving neuron, while an inhibitory spike is interpreted as a "-1" signal. Neural potential also decreases when the neuron fires. The neuron is *excited* if its potential is strictly positive, and then it fires after i.i.d.[6] exponentially distributed periods; firing means sending a signal (positive or negative) to another neuron, or outside. Signals coming from outside form Poisson processes. This model is parameterized by the following elements: the number n of neurons, the firing rate r_i of neuron i, the probability p^+_{ij} (resp. p^-_{ij}), for a signal sent by i, to go to j as a positive (resp. negative) one, the probability $d_i = 1 - \sum_j (p^+_{ij} + p^-_{ij})$ of the signal to go outside the network, and the exogenous rates α_i and β_i of the signal flows arriving at i. E. Gelenbe proved in [5] that the probability q_i that neuron i is excited, in steady state, is given by

$$q_i = \frac{\lambda^+_i}{r_i + \lambda^-_i} \tag{1}$$

where λ^+_i and λ^-_i are defined as $\lambda^+_i = \sum_{j=1}^n q_j w^+_{ji} + \alpha_i$, $\lambda^-_i = \sum_{j=1}^n q_j w^-_{ji} + \beta_i$, and the *weights* are $w^+_{ji} = r_i p^+_{ji}$, $w^-_{ji} = r_i p^-_{ji}$. Gelenbe also gave the stability conditions associated with this system. Observe that the information contained in the RNN is represented by the frequency at which the signals travel (the set of weights). The computation of the q_i's is thus a fixed point problem $F(q) = q$. During this computation, if we get a value $q_i > 1$ then we force $q_i = 1$ until the convergence (we say neuron i is *saturated*).

Next, we present the algorithm based on the RNN model to solve the problem of selecting a specific assignment during the iterative process of the GRASP construction phase.

We define a *neuron* as the pair (node, substream), denoted by v^k, which belongs to one of the following classes of neurons:

$$\mathcal{A} = \{v^k \mid v \in \mathcal{P}_k\ ,\ BW^{out}_v \geq bw_k\} \tag{2}$$

$$\mathcal{O} = \{v^k \mid v \in \{\tau_{k,1}, \ldots, \tau_{k,M_k}\}\ ,\ M_k \geq 0\} \tag{3}$$

$\forall k \in \{1, \ldots, K\}$. We use $\mathcal{A}^k$ and $\mathcal{O}^k$ to denote the subsets of neurons of $\mathcal{A}$ and $\mathcal{O}$ respectively, where the nodes of the neurons are contained in the substream σ_k. In addition, we define a *possible assignment* as the pair (i, j) where $i \in \mathcal{A}^k$ and $j \in \mathcal{O}^k$, $\forall k \in \{1, \ldots, K\}$. The excitatory and

[6]Independent and Identically Distributed.

```
Procedure Select_RNN_Assignment_p2p
Input: $C$, $\alpha$, $MaxIter$

 1:     for each $(i, j) \in C$ do
 2:         compute $w_{ij}^{+}$ {using the equation 4}
 3:     end for
 4:     $q = (q_1, \ldots, q_n) \leftarrow$ Initialize($n$) {$n$-orphans neurons vector}
 5:     compute $F(q)$ {using the equation 6}
 6:     while $||F(q) - q||_2 > \alpha$ and $iter < MaxIter$ do
 7:         $q \leftarrow F(q)$
 8:         compute $F(q)$ {using the equation 6}
 9:         $iter \leftarrow iter + 1$
10:     end while
11:     $j^* \leftarrow \operatorname{argmax}_j \{q_j \mid j = 1, \ldots, n\}$
12:     $i^* \leftarrow \operatorname{argmax}_i \{\overline{\mathsf{AIC}}_i \mid (i, j^*) \in C\}$ {AIC is the greedy function}
13: return $(i^*, j^*)$
```

Figure 2. RNN Algorithm

inhibitory weights of the neural network are set as follows

$$w_{ij}^{+} = \frac{AIC_{ij}}{\overline{AIC}_i}, \quad \text{if } (i,j) \text{ is a } \textit{possible assignment} \tag{4}$$

$$w_{ij}^{-} = 1, \quad \text{otherwise} \tag{5}$$

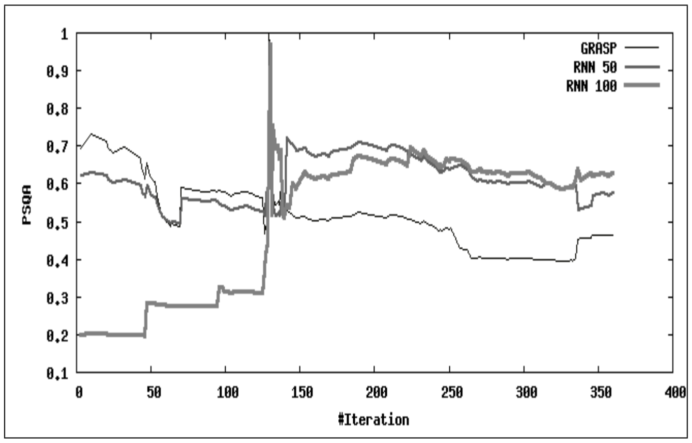
where AIC is the *"assignment improvement criteria"* function corresponding to the greedy criteria implemented in the same way it is done in GRASP, AIC_{ij} is the evaluation of the AIC function after the assignment (i, j) is applied and $\overline{AIC}_i$ is the average of all AIC function evaluations corresponding to all possible assignments containing the neuron i as a parent, which is formally defined as $\overline{AIC}_i = \sum_{j \in \mathcal{O}^k} AIC_{ij} / \left|\mathcal{O}^k\right|$ where $i \in \mathcal{A}^k, \forall k \in \{1, \ldots, K\}$. All other network parameters are set to zero. Using this setup, note that the firing rate is computed as $r_i = \sum_j (w_{ij}^{+} + w_{ij}^{-})$. The connection weights have been chosen in a way such that the neural network captures connectivity information about the possible assignments that can be done. When the neural network is set up in this manner, an excited neuron will have a greater excitatory effect on neurons which have high excitatory connections (i.e., higher assignment improvement) with it. Basically, the procedure in our RNN consists in artificially exciting the *parent* neurons setting $q_i = 1 \; \forall i \in \mathcal{A}$ and then computing the values q_j for all the *orphan neurons* in $\mathcal{O}$ iteratively solving the fixed point problem using Equation 1. According to the RNN set up presented previously, we simplify Gelenbe's equation as follows

$$q_j = \frac{\sum_{i \in \mathcal{A}^k} w_{ij}^{+}}{2|\mathcal{A}| - |\mathcal{A}^k| + |\mathcal{O}| - 1 + \sum_{c \in \mathcal{O}, c \neq j} q_c} \tag{6}$$

$\forall j \in \mathcal{O}^k$ with $k = 1, \ldots, K$. Figure 2 presents the pseudocode of the RNN algorithm. First, we compute the excitatory weights for all the possible assignments (set C, candidates) (lines 1-3). This is necessary for the computation of $F(q)$ as can be seen in Equation 6 (lines 5 and 8). Before the iteration, we need to initialize the vector q (line 4); each neuron excitation probability is usually initialized with a near zero value because this way, the network iterates until convergence in a smaller number of iterations. The stop criterion depends on the threshold α, and in case the iteration does not converge, we limit the number of iterations to $MaxIter$. When convergence is reached, we choose the orphan neuron j with the highest value q_j (line 11) and then we compute the values $\overline{AIC}_i$ for all the neurons $i \in \mathcal{A}$ where (i, j) is a possible assignment. Finally, we choose the parent neuron i with the highest value $\overline{AIC}_i$ (line 12) and the assignment (i, j) is selected as the solution.

5 Experimental Results and Discussion

Network simulator. In order to test the implemented algorithms, we developed a program to simulate the behavior of the network during a certain period of time, divided in a discrete number of intervals. The algorithm to be executed at the end of each interval can be plugged into the simulator by implementing a specific function header. In our case, we implemented the GRASP / RNN based algorithm described in the previous sections, but it is important to note that *any* assignment algorithm can be used provided its interface can be adapted to the one required by the algorithm. This simulator was implemented in C language, using the GNU C Compiler (`gcc`). The tests were run on Intel Pentium 4 / AMD Athlon 64 machines with 1 GB of RAM, running under Ubuntu Linux 6.10.
Node dynamics. It is important that the behavior of the network is as close as possible to a real case; therefore, the dynamics of the peer connection/disconnection is based on a live-video service of a medium-size ISP, which gave us access to a set of logs of user's behavior. These logs were used to generate input files for the simulator, containing node connections and disconnections for each iteration, as well as general static data about the nodes, such as available upstream bandwidth.
RNN use percentage. On each iteration of the overall method, a parameter is used to determine how the connections are assigned: this parameter indicates the probability of using the RNN based algorithm for constructing solutions in that iteration, instead of the GRASP based algorithm.
Analysis of results. The tests were executed using data of a real network during a period of time divided into 360 time intervals; in this work, we present results for networks with averages of 30 and 60 nodes connected at a given time, and with the original video stream split into 1, 2, 4 and 10 substreams. The notation used to describe each one of these test cases is $NnKk$, where n is the average number of nodes connected at a given time, and k is the number of substreams. Therefore, we have the following tests: $N30K1$, $N30K2$, $N30K4$, $N30K10$, $N60K1$, $N60K2$,



Criteria	BW		Current		Future	
Cases	$t(s)$	PSQA	$t(s)$	PSQA	$t(s)$	PSQA
$N30K1$	0.2	0.55	0	0.43	0.2	0.43
$N30K2$	0	0.79	0	0.68	1.8	0.66
$N30K4$	0.4	0.84	0.2	0.84	13.6	0.81
$N30K10$	16.6	0.95	20.8	0.92	4587.2	0.92
$N60K1$	0	0.61	0.4	0.39	0.8	0.36
$N60K2$	0.2	0.69	0.8	0.6	15.6	0.52
$N60K4$	1	0.72	2	0.53	100.8	0.47
$N60K10$	4.4	0.94	19.4	0.86	2108.2	0.89

Figure 3. Numerical results

$N60K4$ and $N60K10$. For each one of these tests, we ran the simulation using 3 different "improvement criteria": bandwidth, current PSQA and future PSQA (these criteria are described in Section 3.1). We also tried different values for the *RNN use percentage* parameter: 0% (GRASP only), 50% and 100% (RNN only). At the end of each iteration, after running the assignment algorithm, we measure the global perceived quality (PSQA) of the network, as well as the execution time of the algorithm. The table in Figure 3 shows the average quality of the network and execution time of the algorithm for the 360 iterations on each test case. As expected, average quality increases when more substreams are used, since this allows a better distribution of the content across the network. However, execution time also increases (and quite quickly), because there will likely be more choices on each step of the assignment algorithm. It is interesting to notice that the average quality of the network is quite good when using 4 substreams and excellent when using 10. These results can then be used when deciding into how many substreams the original video will be split up. Other observations include the fact that the best overall quality values are obtained using the bandwidth assignment improvement criterion, which also is the least computationally demanding. The graph in Figure 3 shows the evolution of PSQA during the simulation for different values of RNN use percentage. As we can see, while the use of RNN initially leads to weaker results, this soon improves and then the perceived quality stabilizes at a higher level than the GRASP only assignments. We also observed, for the "future" improvement criteria, a noticeable decrease in the execution time of the algorithm when using RNN. As a future work, it would be interesting to study this in detail to try to determine which situations are better suited for an RNN algorithm.

References

[1] H. Cancela, F. Robledo, P. Rodríguez-Bocca, G. Rubino, and A. Sabiguero. A robust P2P streaming architecture and its application to a high quality live-video service. *Electronic Notes in Discrete Mathematics*, 30C:219–224, 2008.

[2] A. P. Couto da Silva, P. Rodríguez-Bocca, and G. Rubino. Optimal Quality–of–Experience Design for a P2P Multi-source Video Streaming. In *IEEE International Conference on Communications (ICC 2008)*, Beijing, China, May 2008.

[3] C. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, and J. V. der Merwe. Enhanced streaming services in a content distribution network. *IEEE Internet Computing*, pages 66–75, 2001.

[4] E. Gelenbe. Random Neural Networks with Negative and Positive Signals and Product Form Solution. *Neural Computation*, 1(4):502–511, 1989.

[5] E. Gelenbe. Stability of the Random Neural Network model. *Neural Computation*, 5(2):239–247, 1990.

[6] E. Gelenbe and F. Batty. Minimum cost graph covering with the random neural network. *Computer Science and Operations Research*, pages 139–147, 1992.

[7] E. Gelenbe, V. Koubi, and F. Pekergin. Dynamical random neural network approach to the traveling salesman problem. In *Proc. Symp. Syst., Man., Cybern.*, pages 630–635, 1993.

[8] S. Mohamed and G. Rubino. A Study of Real–time Packet Video Quality Using Random Neural Networks. *IEEE Transactions On Circuits and Systems for Video Technology*, 12(12):1071–1083, 2002.

[9] M. G. C. Resende and C. C. Ribeiro. *Handbook of Methaheuristics*. Kluwer Academic Publishers, 2003.

[10] P. Rodríguez-Bocca, G. Rubino, and L. Stábile. Multi-Source Video Streaming Suite. In *7th IEEE International Workshop on IP Operations and Management (IPOM'07)*, San José, California, United States, October 2007.

[11] S. Wee, W. Tan, J. Apostolopoulos, and S. Roy. System design and architecture of a mobile streaming media content delivery network (msdcdn). Technical report, Streaming Media Systems Group, HP-Labs, 2003.