An ant-colony approach for the design of optimal chunk scheduling policies in live peer-to-peer networks

Pablo Romero*, Franco Robledo and Pablo Rodríguez-Bocca

Departamento de Investigación Operativa,
Instituto de Computación,
Facultad de Ingeniería,
Universidad de la República,
Julio Herrera y Reissig 565, Montevideo, Uruguay
E-mail: promero@fing.edu.uy
E-mail: frobledo@fing.edu.uy
E-mail: prbocca@fing.edu.uy
*Corresponding author

Abstract: Peer-to-peer networks are self-organised communities over the internet infrastructure, in which peers are both clients and servers. The global resources of a peer-to-peer network increase proportionally with the population, promoting scalability. Peers are organised covering neighbouring-strategies and chunk-scheduling policies that determine the success of the cooperation scheme. In this paper, we address the design of chunk-scheduling policies in a cooperative scenario, assuming a complete mesh-topology under regime. All users wish to display a video channel with no cuts and low buffering times. We propose an in-depth analysis of this cooperative system, and develop the best chunk scheduling policy so far, found via a sophisticated ant-colony-based exploration. We introduce the new policy into a real platform. There, users wait five seconds to start watching following our new policy (versus minutes in previous policies), with acceptable number of cuts.

Keywords: combinatorial optimisation problem; COP; travelling salesman problem; ant colony optimisation; ACO; peer-to-peer; chunk-scheduling policy.

Reference to this paper should be made as follows: Romero, P., Robledo, F. and Rodríguez-Bocca, P. (2013) 'An ant-colony approach for the design of optimal chunk scheduling policies in live peer-to-peer networks', *Int. J. Metaheuristics*, Vol. 2, No. 2, pp.101–122.

Biographical notes: Pablo Romero received his MSc in Mathematical Engineering and degree in Electrical Engineer in 2008 from Universidad de la República, Montevideo, Uruguay. He is an Associate Professor at both the Computer Science Institute and Mathematics and Statistics Institute, at Facultad de Ingeniería, Montevideo, Uruguay. Currently, he is a PhD student of Computer Science. His research interests include mathematical analysis and design of telecommunication networks.

Franco Robledo is a Full Professor at the Computer Science Institute and Mathematics and Statistics Institute, at Facultad de Ingeniería, Montevideo, Uruguay. He holds a PhD in Computer Science from University of Rennes 1, INRIA Rennes, France in 2005, and Computer Systems Engineer degree from

Universidad de la República, Uruguay in 1997. His research interests include operations research techniques (meta-heuristics, graph theory and combinatorial optimisation) applied to topological network design models.

Pablo Rodríguez-Bocca is a Professor at the Computer Science Institute and Mathematical and Statistical Institute. He holds a PhD in Computer Science from University of Rennes 1, INRIA Rennes, France in 2008, Computer Systems Engineer degree from Universidad de la República, Uruguay in 2001, and Electrical Engineer degree from Universidad de la República, Uruguay in 2001. He has a professional activity as Manager in Added Value and Contents, at Administración Nacional de Telecomunicaciones (ANTEL), Uruguay. His research interests include operations research techniques (meta-heuristics, combinatorial optimisation and probability models). His main application areas are networks design, quality evaluation (QoS, QoE), and multimedia applications and systems.

1 Introduction

Client server systems offer a highly-predictable service and performance. However, their upload-capacity is constant and easily exceeded. As a consequence, a common alternative approach is the deployment of peer-to-peer systems. Peers are clients as well as servers, and their uploading bandwidth is part of the network resources. Peers cooperate basically in a three-level policy. First, peers discover other peers interested in the same content: this is called *swarming*. Then, peers select few other neighbours from their swarm in order to cooperate, called *peer selection policy*. Finally, peers must cooperate sending data-pieces to each other according to a *chunk scheduling policy*.

In order to distribute video contents via the internet there are three video streaming modes, which are quite similar from the users viewpoint but extremely different when considering a peer-to-peer network design. The first one is file sharing, where the video content must be completely downloaded before its playback. The second one is video on-demand, where the video is distributed as progressive download to end-users. In progressive download protocols, the video is downloaded in a best effort mode, because the protocols do not include any consumer-producer synchronisation (i.e., real-time constrains). The third mode is live-video streaming, which imposes the most strict requirements. In this case, a streaming protocol is used, and there is an explicit synchronisation between consumers and producer. Therefore, end-users must be synchronised with time, and the system is highly sensitive to latency.

By efficiency and network-adaptation issues, the video content must be chopped into pieces, called chunks. As a consequence, a chunk scheduling policy must be defined. In its most basic and structural way, a chunk policy represents the order in which chunks should be requested, trying to assure the highest video quality of experience to end-users. There are two reasons that press peers to contribute with the system, namely, idle uploading capacities and chunk urgencies. In push-based systems, peers are pressed to send chunks whenever they have idle uploading resources. So, push-based systems inspire the chunk scheduling design in idle uploading resources. The suppliers choose a neighbour and send a missing chunk, without their explicit request. A different approach is defined in pull-based systems, which is pressed by requests: downloaders choose a

neighbour and request a selected chunk. The reader can find a recent survey on the design of live video streaming in Abboud et al. (2011).

In this work, we focus on a simplified pull-based model for the steady state of a live streaming mode. This paper is structured as follows. Section 2 describes related work and our motivation. Section 3 details a simplified cooperative model for a live streaming service. This model is originally defined in Zhou et al. (2007), and is focused on the design of optimal *chunk scheduling policies*, in a structural way. We also define in this section a combinatorial optimisation problem (COP) associated with the search of the best chunk scheduling policy. Section 4 contains a preliminary analysis of the optimisation problem and a rough solution. Section 5 presents a resolution of the problem using ant-colony optimisation, based on its seminal concept of ants-system. In a first stage, a suitable re-formulation of the COP into an asymmetric travelling salesman problem (ATSP) takes effect. In a second stage, we introduce an ant-colony-based heuristic to solve the ATSP, which has an interpretation in the design of chunk scheduling policies. Section 5 shows the applicability of our results in a real scenario. Finally, Section 6 contains the main conclusions of this work.

2 Related work

Normally, the development of experiments in real peer-to-peer systems is expensive, and prohibitive by reputability reasons. A fail has a disappointing effect in final users. An alternative is to use network overlays that offer a performance evaluation for distributed systems, for instance, M-Lab and PlanetLab (Chun et al., 2003). By scalability and bandwidth reasons, PlanetLab is not naturally suitable for live streaming scenarios with hundreds of nodes. On the other hand, emulations can be carried out to measure performance aspects of live video streaming. However, it turns out to be a hard task when many realistic elements must be addressed. As a consequence, the main comprehension of peer-to-peer systems is based on mathematical modelling and simulations. Despite, the first articles in this area were essentially experimental. Napster and Gnutella (Zeinalipour-Yatzi and Folias, 2002) inspired the new cooperative paradigm in a centralised fashion. However, flooding and non-altruistic behaviour of users determined bad performance (Napster in fact closed for legal aspects of audio distribution). The creation of BitTorrent by Cohen (2003) had an enormous influence in current peer-to-peer systems. Diverse mathematical models try to understand the behaviour and scalability of BitTorrent-based systems, including Markov chains (Zhao et al., 2009a), fluid models (Qiu and Srikant, 2004), branching processes (Veciana and Yang, 2003; Yang and Veciana, 2004) and marginal probabilities (Zhou et al., 2007; Zhao et al., 2009b), just to name a few. Veciana and Yang (2003), and Yang and Veciana (2004) propose a branching process to describe the service capacity of peer-to-peer systems. They prove that chunk-subdivision (normally known as pipelining) combined with peer-cooperation outperform exponentially the capacity of a raw-server. They studied the *life* of a content via three phases. The first one is the flash-crowd phase, in which users arrive massively, and the system is exposed to a danger of extinction, if the capacity is not enough. The second is the stationary state, where users arrive and depart the system defining a balanced phenomenon with nearly-constant capacity, usually modelled as a closed network. Finally, during the *death* of the file-life, peers depart with practically no

arrivals (the content finally looses its popularity). A comprehensive analysis of BitTorrent-based system is presented in Qiu and Srikant (2004). Their results promote the applicability of BitTorrent systems for file sharing services. Several models were developed to understand the trades-off between continuity and delay in peer-to-peer data-driven systems. A push-based model and analysis of four different forwarding schemes are discussed by Chatzidrossos et al. (2010), under both outdated and full information. They outstand the strength of random packet forwarding over fresh-data first (what we here call the Rarest First policy). The model includes node connectivity degree, outdated information and dynamics. A pull-based cooperative system is analysed in Veciana and Yang (2003), and Yang and Veciana (2004), which illustrates the playback-delay trade-off in a streaming context. They introduce a mixture strategy between Greedy and Rarest First, showing improvements. The same authors revisit the problem in Zhao et al. (2009b), finding asymptotically optimal set of strategies when the storage capacity of peers tends to infinity.

In this work, we extend the mathematical analysis of the model presented in Zhou et al. (2007). This model is here chosen mainly for three reasons consistent with live video streaming: it assumes strict synchronisation between peers, the cooperation is essential and the quality offered by each chunk scheduling policy is measured considering the playback-delay trade-off. This model has been extensively analysed in Bertinat et al. (2009a, 2009b) and Romero et al. (2010). Particularly, Bertinat et al. (2009a) propose a control system which systematically improves continuity and latency. This ideal design has sub-optimal performance, and motivates further research. A single-objective video quality measure is defined related with video cuts and buffering times, and a new COP is introduced in Bertinat et al. (2009b). In Zhao et al. (2009b), an asymptotic analysis of the cooperative model is presented. The authors assure that a '*V*-shaped' chunk policy is asymptotically optimal, and study particular instances of the problem as well. Romero et al. (2010) propose a translation of this COP into a suitable ATSP, and an ant-colony optimisation resolution. In this article, we extend the analysis presented in Romero et al. (2010). We outperform the asymptotically optimum design of *V*-shaped chunk scheduling policies defined in Zhao et al. (2009b), with a meta-heuristic-based approach.

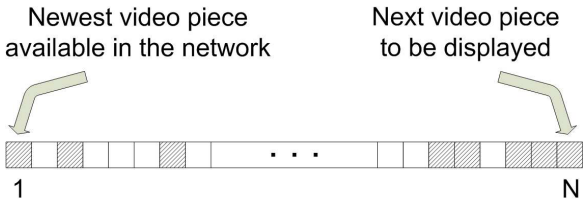
3 A simple cooperative model

3.1 Description (from Zhou et al., 2007)

Consider a closed fully-connected network with a swarm of *M* peers (normally several hundreds or thousands) and one server *S*, who stores and broadcasts an unlimited video flow (a TV-channel, for instance). The server *S* cuts the video into small pieces (called *chunks*) and, in each time slot, chooses only one peer uniformly at random to send the piece. Pieces are distributed in playback order. All peers have a buffer storage with a capacity of *N* video chunks, and every video chunk shifts one buffer-cell in each time slot. The server *S* injects chunks at buffer-cell 1, for only one benefited peer in each slot. All non-benefited peer can obtain (at most) one video chunk in a time slot by means of a request.

The request is a *pull-based* process, and works as follows. Peer A chooses a peer uniformly at random (peer B). Peer A chooses a permutation π_A of the natural subset $\{1, \ldots, N-1\}$, and checks if he owns the piece on buffer-cell $\pi(1)$. If not, peer A checks B's buffer-cell $\pi(1)$. If B has that video chunk, he sends to A a copy of that piece, and the request is successful. Otherwise, the process is repeated (peer A checks buffer cell $\pi(2)$ and so on). Every request finishes either in a success (if the requesting peer gets one video chunk) or in a failure. It is assumed that the whole request takes place during one time slot. Once the time slot is over, all video chunks are right-shifted one buffer-cell, closer to the playback deadline. Figure 1 illustrates the buffer structure.

Figure 1 Buffer structure for each peer (see online version for colours)



Note: Buffer-cell 1 has the newest video chunk of the system, whereas buffer-cell N contains the piece being currently played on the screen.

Peers are awarded with high video quality when they can fill buffer-cell N as many times as possible (high continuity), but having at the same time the lowest number of video chunks in the whole buffer (which means reduced buffering times).

Definition 3.1: The bit-map $(p_X(1), \ldots, p_X(N))$ is the probability that peer X owns a video chunk in the buffer-cell i. The number $p_X(N)$ is the playback-delivery ratio, or continuity.

Under regime, all peers will have the same bit-map (i.e., $p_X(i) = p_i$ for all i), and the buffering time can be measured (Zhou et al., 2007):

Definition 3.2: The buffering-time of the system can be found by $L = \sum_{i=1}^{N} p_i$, and represents the expected time a joining peer should wait in order to reach the regime bit-map p_i, starting with an empty buffer.

Either in a perfect cooperative scenario or assuming fair conditions, peers are encouraged to choose the same permutation policy, in order to have the same buffering times as well as equivalent continuity. The issue is thus to choose a permutation in order to get the *continuity* p_N as high as possible, and the expected number of video chunks or *buffering time*, measured by $L = \sum_{i=1}^{N} p_i$, as low as possible.

3.2 Simple chunk scheduling model

Considering symmetry and cooperation, we can state natural assumption for this simplified live streaming scenario. Fairness suggests that all peers should apply the same chunk policy, so $\pi_X = \pi$ for some permutation π of the set $\{1, \ldots, N-1\}$. As a consequence, the chosen permutation governs the logic of a request. In a stationary state, the bit-map p_i is the same for each peer [the existence of a stationary state is assured in

Zhao et al. (2009b)]. The conservation law of video chunks together with the requesting process imply that:

$$p_1 = 1/M, \tag{1}$$

$$p_{i+1} = p_i + (1 - p_i)\, p_i s_i, \quad i = 1, ..., N-1. \tag{2}$$

where s_i represents the probability to visit buffer-cell i during a request following the permutation policy π, named *strategic sequence*, for short. Equation (1) states that the probability of being selected by the server is one out of M. Equation (2) holds because buffer-cell $i + 1$ can be filled by two disjoint ways. The first one is by chunk-shift from buffer-cell i, which is filled with probability p_i (all video chunks are one step closer to the deadline after one time slot). The second one occurs when the requested peer selects video chunk at buffer-cell i in the previous time slot. This event has probability $(1 - p_i)p_i s_i$ and represents the cooperation term: the requested peer has the video chunk at buffer-cell i (probability p_i) but the requesting peer does not $(1 - p_i)$. Moreover, the requested peer visited buffer-cell i (with probability s_i). We refer the reader to Zhou et al. (2007) for further details.

The chunk policy π dictates an expression for the strategic sequence s_i. A peer is not-benefited from the server with probability $1 - \frac{1}{M}$. In order to reach buffer-cell $\pi(i)$ during a request, a partial fail must occur in all previous positions $\pi(1)$, $\pi(2)$, …, $\pi(i - 1)$. A fail in position $\pi(j)$ for a certain $j \in \{1, \ldots, i - 1\}$ has probability $[1 - p_{\pi(j)}(1 - p_{\pi(j)})]$ (it is not true that the requested peer has the video chunk in buffer-cell $\pi(j)$ and the requesting does not). As a consequence, the next expression for the strategic sequence s_i holds:

$$s_{\pi(i)} = \left(1 - \frac{1}{M}\right) \prod_{j=1}^{i-1} \left[1 - p_{\pi(j)} \left(1 - p_{\pi(j)}\right)\right], \forall i = 1, ..., N-1. \tag{3}$$

The next linear relations are obtained combining Expressions (2) and (3):

$$s_{\pi(i+1)} - s_{\pi(i)} = -s_{\pi(i)} \left(p_{\pi(i)} \left(1 - p_{\pi(i)}\right)\right) = p_{\pi(i)} - p_{\pi(i)+1}, \, i = 1, ..., N-2. \tag{4}$$

Summing up all the information, the bit-map probabilities $p = (p_1, \ldots, p_N)$ (and also the strategic sequence s_i) can be computed for each permutation strategy π, by solving the following non-linear system $NLS(\pi)$:

$$NLS(\pi): \begin{cases} p(1) = \frac{1}{M} \\ p(i+1) = p(i) + p(i)\left(1 - p(i)\right) s(i), \quad \forall i \in \{1, ..., N-1\} \\ s_{\pi(1)} = 1 - \frac{1}{M} \\ s_{\pi(i+1)} = s_{\pi(i)} + p_{\pi(i)} - p_{\pi(i)+1}, \quad \forall i \in \{1, ..., N-2\} \end{cases}$$

The non-linear system $NLS(\pi)$ can be accurately solved for every particular permutation π with the Newton-Raphson method, with quadratic order convergence.

3.3 Single-score for each chunk-policy

The cooperative system is fully characterised by a scheduling policy π, which must be chosen regarding playback and buffering times. Equation (2) assures that the bit-map probabilities p_i are monotonically increasing. Then, there is a trade-off between the continuity p_N and the buffering time $L = \sum_{i=1}^{N} p_i$. The following analytical result will determine a natural single-objective measure for this cooperative system:

Proposition 3.3: Let π be a permutation of the natural set $\{1, \ldots, N-1\}$, and X_π the random variable that represents the number of steps in a successful request. Then, its expected value $E(X_\pi)$ is:

$$E(X_\pi) = \frac{M}{M-1} \sum_{i=1}^{N-1} \pi(i)\left(p_{i+1} - p_i\right).$$

Proof: Let α_i be the probability of having a successful request in step i. Then:

$$\begin{aligned} E(X_\pi) &= \sum_{i=1}^{N-1} i\alpha_i \prod_{j=1}^{i-1}\left(1-\alpha_j\right) = \sum_{i=1}^{N-1} i p_{\pi(i)}\left(1-p_{\pi(i)}\right) \prod_{j=1}^{i-1}\left[1-p_{\pi(j)}\left(1-p_{\pi(j)}\right)\right] \\ &= \frac{1}{1-\frac{1}{M}} \sum_{i=1}^{N-1} i p_{\pi(i)}\left(1-p_{\pi(i)}\right) s_{\pi(i)} = \frac{M}{M-1} \sum_{i=1}^{N-1} i\left(p_{\pi(i)+1} - p_{\pi(i)}\right) \\ &= \frac{M}{M-1} \sum_{i=1}^{N-1} \pi(i)\left(p_{i+1} - p_i\right). \end{aligned}$$

□

Then, $Q(\pi) = E(X_\pi)$ is a linear combination of the jumps $p_{i+1} - p_i$. Moreover, it is monotonically increasing with the continuity p_N, for its derivative with respect to p_N is $\pi(N-1) > 0$. Let us recall that all requests take place in one time slot, so long requests do not affect the time response of the system. Then, it is convenient to maximise the quality Q_π, which defines the score of a COP, where the chunk scheduling policy represents the decision variable.

Let S_{N-1} be the space of all permutation of the natural subset $\{1, \ldots, N-1\}$. In order to find the best chunk policy, the following COP must be solved:

$$\text{COP}: \max_{\pi \in S_{N-1}} E(X_\pi) \tag{5}$$

s.t.

$$\begin{cases} p_1 = \frac{1}{M} \\ p_{i+1} = p_i + p_i\left(1-p_i\right)s_i, \quad \forall i \in \{1, \ldots, N-1\} \\ s_{\pi(1)} = 1 - \frac{1}{M} \\ s_{\pi(i+1)} = s_{\pi(i)} + p_{\pi(i)} - p_{\pi(i)+1}, \quad \forall i \in \{1, \ldots, N-2\} \end{cases} \tag{6}$$

4 On the search of an optimal chunk scheduling policy

4.1 Preliminary analysis

The goal is to determine a permutation such that the continuity p_N is as high as possible, keeping the sum $L = \sum_{i=1}^{N} p_i$ low. We try here this ideal bi-objective goal. In this subsection, we will present a first approximation to the resolution of the playback-delay trade-off. We will give evidence of the weaknesses of this resolution. However, it is instructive and serves as a seed for the design of trails in a more sophisticated ant-colony-based heuristic.

Definition 4.1: The best bit-map vector $\hat{p}$ is defined by $\hat{p}_i = \frac{1}{M}$ whenever $1 \le i < N$ and $\hat{p}_N = 1$. It has the lowest buffering time $L = \frac{N-1+M}{M}$ and perfect continuity.

It is clear that we wish to find a permutation π whose bit-map is $\hat{p}$. However, the latter is impossible:

Lemma 4.2: Imperfect continuity:

$$p_N < 1, \quad \forall \pi \in S_{N-1}.$$

Proof: We will prove a stronger statement: $p_i < 1$, $\forall i \in \{1, \ldots, N\}$ by induction on i. The base step is evident, since $p_1 = \frac{1}{M} < 1$. Let us suppose now that $p_h < 1$ for some $h \in \{1, \ldots, N-1\}$. Then $p_{h+1} = p_h + (1 - p_h)p_h s_h < p_h + (1 - p_h) = 1$. □

Lemma 4.3: Increasing bit-map:

$$p_i < p_{i+1}, \forall i \in \{1, ..., N\}, \quad \forall \pi \in S_{N-1}.$$

Proof: Trivial: by induction over the set $\{1, \ldots, N\}$. □

It is worth noticing that the most spread peer-to-peer communication system in internet is BitTorrent. Its original protocol was developed for highly populated file sharing systems, and was successfully introduced in video on-demand applications as well. The chunk scheduling policy in BitTorrent is known as the *Local Rarest First*. Every peer checks the total number of video chunks in the neighbouring peers, and downloads the rarest replicated video chunk among them. The Increasing Bitmap Lemma guarantees that Rarest First policy is identified with the permutation $\pi(i) = i$.

Lemma 4.4: Decreasing composition: Given an arbitrary permutation $\pi \in S_{N-1}$, the composition $(s \circ \pi)(i) = s_{\pi(i)}$ is strictly decreasing.

Proof: Take an arbitrary permutation $\pi \in S_{N-1}$. Combining the Increasing Bitmap Lemma and (4), we get that $s_{\pi(i+1)} = s_{\pi(i)} + p_{\pi(i)} - p_{\pi(i)+1} < s_{\pi(i)}$. □

Proposition 4.5: 'Approximation strategy property' (ASP): Let $x = (x_1, \ldots, x_{N-1})$ be an injective real-valued sequence. Then, there exists a permutation $\pi \in S_{N-1}$ that respects the natural order of the vector x in the following sense: if $x_i > x_j$ then $s_i > s_j$, $\forall i, j \in \{1, \ldots, N-1\}$, where s is the strategic sequence associated with the permutation π.

Proof: For every injective real-valued sequence x there exists a permutation of indices π such that $x_{\pi(1)} > x_{\pi(2)} > \ldots > x_{\pi(N-1)}$. Using the Decreasing Composition's Lemma, the strategic sequence s respects the order $s_{\pi(1)} > s_{\pi(2)} > \ldots > s_{\pi(N-1)}$. The result follows comparing the latter chains of inequalities. □

The ASP leads us to design a strategy whose bitmap is similar to a desired one. Suppose we want to find the chunk policy π that approximates best an ideal input bit-map p^{id}. First, find the corresponding ideal strategic sequence s_i^{id} using equation (2):

$$s_i^{id} = \frac{p_{i+1}^{id} - p_i^{id}}{\left(1 - p_i^{id}\right) p_i^{id}}, \forall i \in \{1, ..., N-1\} \tag{7}$$

Then, we can approximate s_i^{id} via the ASP. Specifically, we obtain a permutation policy π whose strategic sequence s_i respects $s_i > s_j$ whenever $s_i^{id} > s_j^{id}$. The system is supported in the fact that similar strategic sequences are translated into similar bit-maps. The follower system summarises these ideas, and is illustrated in Figure 2.

Figure 2 Follower system: receives a desired bit-map p and returns a feasible bit-map p^*, with the nearest result to the input

Note: Observe that it permits to obtain the permutation strategy π that achieves p^* (in Step 3).

In order to understand the effectiveness or deficiencies of the follower system, we applied some chosen inputs. If the output strategy achieves a bit-map p^* very close to that of the input p, the proposed system would help us to choose any desired vector and approximate it (for instance, the best bit-map $\hat{p}$). Obviously, if this follower system worked perfectly, there would be no reason to find the highest score by solving the previous COP, since the playback continuity p_N and buffering-times L could be chosen.

For the sake of simplicity, we first introduced the exponential bit-map $p_i^{id} = M^{\frac{N-i}{1-N}}$, $i \in \{1, \ldots, N\}$, which has perfect continuity, is monotonically increasing and complies with the constraint $p_1^{id} = \frac{1}{M}$. We can find analytically the output permutation for this case. Observe that $p_{i+1}^{id} = M^{\frac{1}{N-1}} p_i$, and:

$$\frac{s_{i+1}^{id}}{s_i^{id}} = \frac{p_{i+2}^{id} - p_{i+1}^{id}}{\left(1 - p_{i+1}^{id}\right) p_{i+1}^{id}} \frac{\left(1 - p_i^{id}\right) p_i^{id}}{p_{i+1}^{id} - p_i^{id}} = \frac{1 - p_i^{id}}{1 - p_{i+1}^{id}} > 1. \tag{8}$$

Figure 3 Segmented bit-map and its output (see online version for colours)

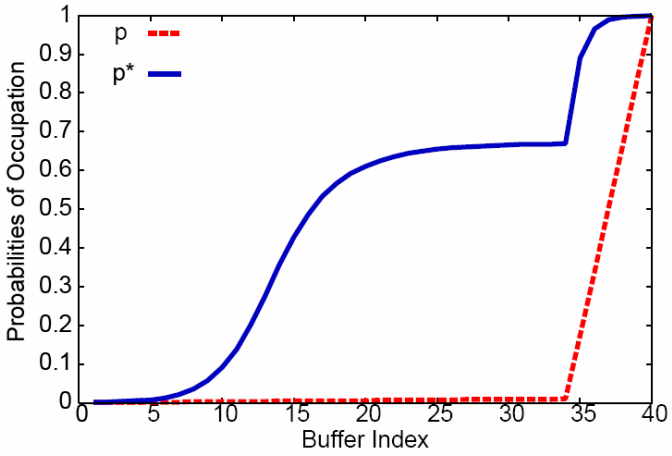
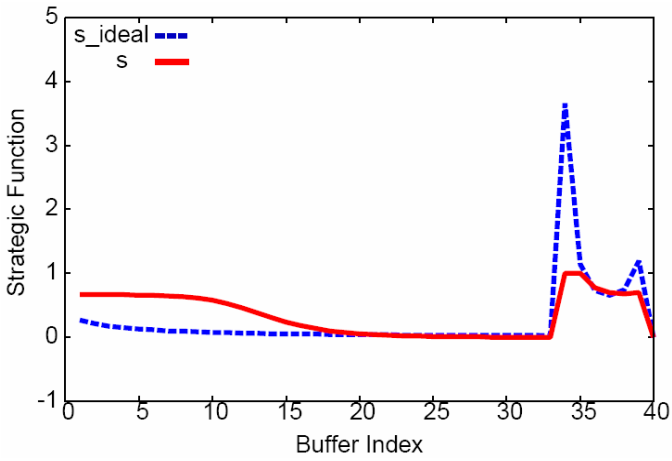


Figure 4 Ideal sequence s^{id} and feasible sequence s (see online version for colours)



This means that s^{id} is an increasing sequence, hence the ASP returns the permutation $\pi_i = N - i$, $i \in 1, \ldots, N-1$. Surprisingly, this strategy is the *Greedy* notion for this cooperative model: ask first for the video chunks that are closer to the deadline, trying not to fill the rest of the buffer. If we input the perfect bitmap $\hat{p}$ the output is again the Greedy strategy. Further studies show that Greedy is not the optimum strategy. Moreover, it achieves a very poor continuity (Bertinat et al., 2009a).

A second test was carried out, now entering a ramp vector with perfect continuity for the case of $M = 1{,}000$ peers and a storage capacity of size $N = 40$. The vectors p^{id}, s^{id}, π and p are represented in Figures 3 and 4. The first observation is that this follower system fails again when following a ramp input (the bit-maps p^{id} and p^* are clearly different). However, a relative maximum or *peak* in the ideal sequence s^{id} generates a big jump in the bit-map (change in slopes of p^{id}). The exact place of the peak leads us to choose where is the change in behaviour of the bit-map p^*, controlling in this way the playback-delay trade-off. The position of the peak plays a critical role, because if it is

near the deadline the continuity will be poor. If not, a higher buffering time L will be experimented. Summarising, the ideal approach does not reflect the behaviour of chosen inputs. Rather, it works like a *dirty mirror*. However, the experience gained with chosen inputs outstands the subset of permutations whose strategic sequence presents a peak (relative maximum). We call it the subfamily of *W*-shaped policies [see Bertinat et al. (2009a) for details]:

Definition 4.6: For each pair of naturals (I, J): $I + J \leq N - 1$, there is one permutation of the *W*-shaped policies, that can be expressed in the following way:

$$\pi(i) = N - i, \quad i = 1, ..., I,$$
$$\pi(I + j) = j, \quad j = 1, ..., J$$
$$\pi(I + J + k) = \left\lfloor \frac{N + J - I}{2} \right\rfloor + \left\lceil \frac{k}{2} \right\rceil (-1)^{k+1},$$
$$k = 1, ..., N - I - J - 1.$$

The reader can check that these policies present a *W*-shaped buffer-priority. The I buffer-cells nearest-to-the-deadline have the highest priority, whereas the J faraway follow in priority. Then, a zig-zag priority is defined in the middle of the buffer (the $\wedge$-part of the *W* priority). Curiously, Zhao et al. (2009b) analysed the same model via Markov-chains in a continuity-driven fashion, and suggest another sub-family of chunk scheduling policies, called *V*-shaped policies:

Definition 4.7: Let $k \in \{1, \ldots, N - 1\}$ be the buffer-cell with the lowest priority. Then a permutation member is *V*-shaped if the priority increases as the position moves away from k.

The authors prove that the *V*-shaped policies contain the asymptotically optimal policy when the buffer size tends to infinity. The number of *V*-shaped policies is exponential with the buffer capacity N. Hence, an exhaustive search among the *V*-shaped policies is computationally prohibitive for large buffer sizes. Note that the SubFamily of *W*-shaped policies $C_2^{N+1} = \frac{N(N+1)}{2}$, polynomial in the buffer capacity N.

4.2 A polytime translation into an ATSP

From now on, we will solve the COP stated in (3.3), trying to find the best chunk scheduling policy π. In this section, we will translate the COP into a suitable ATSP. The latter is solved heuristically following an ant colony optimisation (ACO) approach, which is inspired in the way ants find the shortest path between their nests and their food (Beckers et al., 1992). The reader can find a deep analysis of this nature-inspired meta-heuristic in Dorigo et al. (2006), Dorigo and Stützle (2004), Dorigo and Gambardella (1997), and Blum (2005). A complete graph of N nodes allows to get a bijection between a Hamiltonian tour and a given permutation. We translate the COP into an instance of the well-known ATSP (Oncan et al., 2009)) using the following bijection.

Proposition 4.8: There is a bijection between the space of permutations S_{N-1} and the set of directed Hamiltonian tours of an N-clique.

Proof: Let K_N be an N-clique with labelled nodes $\{1, \ldots, N\}$, and N an auxiliary node, from which all directed Hamiltonian tours T start. Both sets are finite with cardinal $(N-1)!$. By counting, it suffices to find an injection $\varphi : T \mapsto P$. Let us define for every directed tour $t = \{N, v_1, v_2, \ldots, v_N - 1, N\}$ the permutation $\pi(i) = v_i$, $i = 1, \ldots, N-1$. Function $\varphi(t) = \pi$ is one-to-one, and the result holds. □

4.3 A neighbourhood structure

Consider the composition law in the permutation space S_{N-1}. Some trivial facts on the group of permutations $(S_{N-1}, \cdot)$ will be very useful in order to define a neighbourhood structure in the space S_{N-1}, that will be introduced in a meta-heuristic resolution. All permutations are generated by a product of a finite number of transpositions. There are several metrics adopted for permutation spaces. For an overview of distances on permutation groups we refer the reader to Arvind and Joglekar (2008).

Definition 4.9: The Cayley distance $d(\pi_1, \pi_2)$ between permutations π_1 and π_2 is the minimum number of transpositions needed to obtain π_2 from π_1.

Definition 4.10: A distance d on S_{N-1} is called graphic if $d(\pi_1, \pi_2)$ is the length of a shortest path joining π_1 and π_2 in the simple graph with vertex set S_{N-1} and edge set (π_1, π_2): $d(\pi_1, \pi_2) = 1$.

Lemma 4.11: The Cayley distance is graphic.

Proof: Take two arbitrary permutations $\pi_x \neq \pi_y$, and call $l = d(\pi_1, \pi_2) > 0$. By its definition, there exists transpositions $t_1, \ldots, t_l$ such that $\pi_y = t_l t_{l-1} \ldots t_1 \pi_x$. Define recursively the family of adjacent intermediate permutations $\pi_0 = \pi_x$ and $\pi_{i+1} = t_{i+1}\pi_i$, such that $\pi_y = \pi_l$. Then clearly $d(\pi_i), \pi_{i+1}) = 1$, and there exists a path of length l between the nodes π_x and π_y. If there were a shorter path of length $r < l$, the r-path from π_x to π_y would provide intermediate permutations that differ in only one transposition, obtaining that $r \geq d(\pi_1, \pi_2) = l$, a contradiction. Hence, $\pi_x = \pi_0, \pi_1, \ldots, \pi_l = \pi_y$ is the shortest path between π_x and π_y, and the Cayley distance is graphic. □

The previous lemma can be strengthened. In fact, every integer-valued distance on any set X is graphic if and only if $d(a, b) > 1$ implies $d(a, c) + d(c, b) = d(a, b)$ for some c (see Deza and Huang, 1998).

Definition 4.12: A neighbourhood structure for the COP with feasible set S_{N-1} is a collection $\{\mathcal{N}_\pi\}_{\pi \in S_{N-1}}$ of subsets of S_{N-1} such that:

1 The resulting hypergraph is connected: given $\pi_x, \pi_y \in S_{N-1}$, there exists a sequence of solutions $\pi_x = \pi_0, \pi_1, \ldots, \pi_l = \pi_y$ such that $\pi_i \in \mathcal{N}_{\pi > -\infty}$ for all $i = 1, \ldots, l$.

2 For every $\pi \in S_{N-1}$ we can decide in polytime whether there exists a better neighbour $\pi' \in \mathcal{N}_\pi$ such that $Q(\pi') > Q(\pi)$, whenever it exists.

The previous algebraic objects are useful to define a neighbourhood structure in our space of permutations S_{N-1}. Consider for each permutation π the unit ball $\mathcal{N}_\pi = \{\pi' \in S_{N-1} : d(\pi, \pi') = 1\}$.

Proposition 4.13: The set of unit balls $\mathcal{B} = \{\mathcal{N}_\pi, \pi \in S_{N-1}\}$ is a neighbourhood structure for the COP.

Proof: By Lemma 4.11 the Cayley distance is graphic, and consequently $\mathcal{B}$ generates a connected hypergraph. Suppose that we reach a solution π for the COP $\max_{S_{N-1}} Q(\pi)$. There are exactly $|\mathcal{N}_\pi| = C_2^{N-1}$ neighbours, a quadratic polynomial in N. In order to evaluate the quality of a neighbour the non-linear system $NLS(\pi)$ must be solved. We can decide if a neighbouring solution is better than π in polytime, because the Newton-Raphson method is a polynomial approximation scheme, with quadratic order of convergence to the solution. Hence, $\mathcal{B}$ is a neighbourhood structure for the maximisation problem $\max_{S_{N-1}} Q(\pi)$. □

All these tools are used to define an ACO-based algorithm, which finds high competitive policies for the COP.

5 A High-quality policy built by ant-workers

We propose and ant-based algorithm that returns high-quality chunk scheduling policies. The main algorithm can be studied in four blocks (see Algorithm 1). In the first block (Line 1), a weighted network is defined, translating in this way the original COP into a suitable ATSP. A non-negative cost is assign to each edge when Function *Edges* is called, with a learning mechanism based on ant exploration. The second block prepares the ant-colony application calling Function *Pheromones*, which will allow to trace high quality tours. The subfamily of *W*-Sshaped policies from Definition 9 will be considered here as a seed for the pheromone trails. The third block (Line 3) is the *AntWorkers* application itself, which returns a strategy π. Finally, a local improvement is introduced exploiting the neighbourhood structure of the permutation group by means of Function *LocalSearch*.

Functions *Edges*, *Pheromones*, *AntWorkers* and *LocalSearch* will be presented in the following subsections.

Algorithm 1 π = *Main algorithm*(M, N, d, τ, α, β, ρ, *ants*, *iterations*)

1: $d(E)$ = *Edges*(M, N, *ants*)
2: $\tau(E)$ = *Pheromones*(M, N, *ants*, *SubFamily*)
3: π = *AntWorkers*(M, N, d, τ, α, β, ρ, *ants*)
4: π_{out} = *LocalSearch*(π, M, N, *iterations*)
5: **return** π_{out}

5.1 Edges

The whole solution is designed following an ant-worker's philosophy. The basic idea is that several artificial ants start a tour in the auxiliary node N, and measure distances (or leave traces) according to the quality of the recently visited tour.

During *Edges*, the translation from the COP to an ATSP takes place. It returns a non-negative matrix $d(E)$ whose entries $d(i, j)$ contain the edge-costs of a directed N-clique. A unit-cost is assign to all edges in Line 1, and the Greedy strategy ($\pi_i = N - i$)

is considered as a reference score in Line 2. In Lines 3 to 6 several ant-tours are built in order to update costs for edges. Each ant stochastically chooses the next node to visit with Tabu-nodes (in order not to visit the same node twice). During Function *VisitCycle*, ants choose the next step according to the following probability distribution:

$$p\left(x_{j+1}\right)=\frac{d\left(x_j, x_{j+1}\right)^{-1}}{\sum_{i \in NoCycle} d\left(x_j, x_i\right)^{-1}} \tag{9}$$

This means that shorter tours are desirable. In Line 5, the matrix d is updated. *UpdateCost* finds the best strategy so far. Then, all edges inside the tour π are updated according to its scores:

$$d\left(\pi(j), \pi(j+1)\right)=10(N-j) \times \frac{Q_{\max}}{Q(\pi)},$$

where $Q_{\max}$ is the best score obtained so far. The additional ladder-factors $N-j$ avoids re-visiting a cycle several times.

Algorithm 2 $d(E) = Edges(M, N, ants)$

1: $d(E) = 1$
2: $Quality = Greedy$
3: **for** $i = 1$ **to** $ants$ **do**
4: $\quad \pi = VisitCycle(d(E))$
5: $\quad d(E) = UpdateCost(\pi(1), \ldots, \pi(i))$
6: **end for**
7: **return** $d(E)$

5.2 Pheromones

The main ingredient of Function *Pheromones* is that tours are deterministic, given by the subfamily of *W*-shaped policies from Definition (9), exploiting the *smell* of high quality that provides this SubFamily. In this way, the preliminary analysis is incorporated as a seed in this sophisticated algorithm. It returns a matrix τ with the trail of pheromones for each edge. The notion of pheromones helps ants in the main block called *AntWorkers* to build tours with better quality. See Algorithm 3 for details.

Algorithm 3 $d(E) = Pheromones(M, N, ants, SubFamily)$

1: $d(E) = 1$
2: $Quality = Greedy$
3: **for each** $\pi \in SubFamily$ **do**
4: $\quad Q = Quality(\pi)$
5: $\quad \tau = UpdatePheromones(\pi(1), \ldots, \pi(i))$
6: **end for**
7: **return** τ

5.3 AntWorkers

Function *AntWorkers* has several similarities with ant-system, originally proposed by Dorigo and Gambardella (1997). There, the authors distribute m ants into n cities of a TSP instance, and each ant builds a tour using a Tabu list, in order not to visit twice the same node. Ants leave trails of pheromones, and select stochastic tours weighting both shorter paths and pheromones. The diversification of the meta-heuristic depends on a set of parameters:

1 the visibility of the path $\beta \geq 0$

2 the relative importance to the trail $\alpha \geq 0$

3 the trail persistence ρ: $0 \leq \rho \leq 1$ or *evaporation factor* $1 - \rho$

4 a constant Q related to the quantity of trail laid by ants.

Let $x_1, x_2, \ldots, x_j$ be the first j nodes visited by an ant. Each ant builds a biased tour according to the following jump-probability distribution:

$$p\left(x_j, x_{j+1}\right) = \frac{\tau\left(x_j, x_{j+1}\right)^{\alpha} d\left(x_j, x_{j+1}\right)^{-\beta}}{\sum_{i>j} \tau\left(x_j, x_i\right)^{\alpha} d\left(x_j, x_i\right)^{-\beta}}, \tag{10}$$

where $\tau(x_i, x_j)$ is the trail of pheromone for edge (x_i, x_j), and the constraint $i > j$ over the sum is the *Tabu list* of nodes, in order not to visit twice the same node. It is interesting to notice that under perfect visibility and discarding the trail (i.e., $\beta \to \infty$ and $\alpha = 0$), ant-system is exactly the *Greedy* heuristic for the TSP. In this way, the parameters α and β impose a trade-off between greediness and the *smell* of ants, based on pheromones. The updating of pheromones in its classical implementation is strictly based on the evaporation factor ρ: $0 \leq \rho \leq 1$ and the quality of each tour. More specifically, each visited edge receives, per ant, a trail proportional to the trail persistence and quality of the tour. Given that the TSP is a minimisation problem:

$$\tau\left(x_i, x_j\right)^{t+n} = \rho\tau\left(x_i, x_j\right)^{t} \sum_{k=1}^{m} \frac{Q}{L_k} 1_{(x_i, x_j) \in Cyclek}, \tag{11}$$

where L_K is the length of the tour visited by ant k. Note that only the edges visited by ants contribute to the sum (the indicator 1_X is one only if X is true), and the trails of the others receive a *trail evaporation*, by the factor $1 - \rho$. We refer the reader to Dorigo and Gambardella (1997) for an overview of the ant-system design, and its performance with respect to other classical meta-heuristics for the TSP. The book (Dorigo and Stützle, 2004) contains an in-depth analysis of the properties of this population-based meta-heuristics. Now, we will show our particular implementation of Function *AntWorkers*, paying particular attention to the differences with the original ant-system. Basically, our implementation introduces four differences, in order to address the nature of our problem and its translation:

1 *Attractor nest*: the network has an auxiliary node N that plays the role of an attractor nest. All ants start and finish the tour at this node. In fact, every tour has a corresponding permutation-based policy, in accordance with Proposition 4.8.

2 *Serial walk*: in the original ant-system implementation, all ants walk simultaneously. In this design, ants explore the network serially, and each ant updates the trails only after finishing its tour.

3 *Weighted tours*: in order to increase the diversification of the biased tours, we avoid ants to visit the same edges in their first step. This is an artificial guide to ants, so as to visit all edges of the network at least one. In this way, the trail of every edge is updated at least twice.

4 *Massive population*: the authors of the original implementation suggest to use n ants (i.e., one ant per node). Here, we will consider at least three times the number of nodes. In fact, the network has $N(N-1)/2$ edges, but each ant visit exactly N edges. The probability of visiting all edges is increasing with the population of serial ants in this single-run system.

AntWorkers is presented in Algorithm 4.

Algorithm 4 $\pi = AntWorkers(M, N, d, \tau, \alpha, \beta, \rho, ants)$

1: $Quality = Greedy(M, N)$
2: **for** $i = 1$ **to** $ants$ **do**
3: $\quad \pi_i = AntCycle(d, \tau, \alpha, \beta)$
4: $\quad \tau = NewPheromones(\rho, \tau, Q, Q_{\max})$
5: **end for**
6: **return** $\pi = MostVisitedCycle(\pi_1, \ldots \pi_{ants})$

A reference score is the Greedy policy (Line 1). The serial implementation covers Lines 2 to 5, where the nest-node is an attractor (all ants start and finish in this node). During Function *AntCycle* (Line 3), each ant builds a stochastic tour according to the probability vector defined in equation (10). A technical difference with respect to the original ants-system is defined in Line 4, where the trail update takes place. Specifically, only the trails of those edges visited by ant i are modified in Function *NewPheromones*. The new trail for the edge (x_j, x_{j+1}) is:

$$\tau(x_j, x_{j+1}) = (1-\rho)\tau(x_j, x_{j+1}) + \rho \times \frac{10(N-j)Q(\pi)}{Q_{\max}}, \tag{12}$$

where the factor $10(N-j)$ provides a similarity of magnitudes between pheromones and distances. Notice that the ladder-factors $10(N-j)$ were fixed during the network construction in Function *Edge* of the main algorithm. In this way, we do not give priority to pheromones or distances. Additionally, we are mainly interested in the shortest Hamiltonian path (the order in which the $N-1$ nodes are visited).

5.4 LocalSearch

The output permutation of *AntWorkers* is locally improved via a simple local search phase. It looks for the best permutation among their neighbours. If the maximum number of iterations is not reached, a local optimum strategy is returned. *LocalSearch* is very simple, and exploits the properties of the Cayley distance. See details in Algorithm 5.

Algorithm 5 $\pi_{out} = LocalSearch(\pi, M, N, iterations)$

```
Quality = Greedy
while iterations not reached and improves do
    π = BestNeighbor(π, M, N)
end while
return π
```

5.5 Computational effort

In order to understand the trade-off between the computational effort and the quality of the solutions that offers the main algorithm, let us count the quantity of operations, taking the number of score evaluations as the basic operation. The following result summarises the block that imposes the biggest computational effort, and the convergence order with respect to input N.

Proposition 5.1: Let N be the buffer size and $T(N)$ the mean computing-time for the quality $E(X_\pi)$. If ants and iterations have order $O(N)$, then the mean total time for running the main algorithm is $\mathsf{T} = O(N^3 T(N))$.

Proof: The local search phase imposes the largest computational effort. The number of neighbours of a given permutation is $C_2^{N-1} = \frac{(N-1)(N-2)}{2}$. In order to find the best neighbour, it is necessary to evaluate all those neighbours in each iteration. If the number of iterations is linear with N, then this block imposes the biggest computational effort, and is cubic in N. If the mean computing-time for a score evaluation is $T(N)$, the total running time is $\mathsf{T} = O(N^3 T(N))$.. □

6 Results

6.1 Comparison with historical policies

In P2P networks, two classical chunk scheduling policies are *Rarest First*: $\pi(i) = i$, and *Greedy*: $\pi(i) = N - i$. The former works properly in downloading, but not for streaming purposes. The same authors (Zhou et al., 2007) of the original model propose in Zhao et al. (2009b) a slight modification of the model, in which the server can offer video chunks to a randomly chosen fraction f of the M peers in the network. There, they find an asymptotic approximation to the optimal chunk scheduling policy when the buffer size increases. They measure quality only regarding playback continuity, and conjecture that the optimal policy is inside a subfamily of V-shaped policies, and becomes more greedy when f increases. Let k be the buffer-cell with the lowest priority. Then a permutation member is V-shaped if the priority increases as the position moves away from k. The exhaustive search among the V-shaped subfamily of policies is still computationally prohibitive (it has exponential size with N).

We tuned the parameters of the main algorithm inspired in Duan et al. (2007), and adapted to our particular problem. Our final implementation used the main algorithm with $\alpha = 0.4$, $\beta = 1.5$, $\rho = 0.5$ and 100 ants, for the common-network parameters $N = 30$

and $M = 100$. Table 1 shows that the obtained permutation achieves an excellent continuity and at the same time a latency comparable to the one reached by Greedy, outperforming classical policies as well as an average of randomly chosen *V*-shaped policies. The mixture is a set of policies introduced for the first time by Zhou et al. (2007). We include in Table 1 the mixture with the highest continuity. The average *V*-shaped refers to the average performance of one-hundred randomly chosen *V*-shaped policies. Over those 100 samples, our permutation-policy has better continuity of reproduction than 88 samples, and achieves lower latencies than 86 samples. Additionally, we could not find even one *V*-shaped sample with both better continuity and latency than our permutation-based policy. These results confirm a highly competitive trade-off between playback continuity and buffering-times of our proposal. The *V*-shaped members define both high playback policies but high buffering times as well. In fact, the authors Zhao et al. (2009b) focus the design on playback continuity only.

Table 1 Performance of different chunk policies

Strategies	*Continuity*	*Latency*
Rarest First	0.9571	21.0011
Greedy	0.9020	4.1094
Mixture	0.9970	14.4798
Average V-shaped	0.9670	17.6683
Main algorithm	*0.9998*	*7.9821*

6.2 Results in a real-life scenario

The new chunk scheduling policy was introduced into a real platform named *GoalBit*, which is the first open-source P2P network that widely offers live video streaming to final internet users Bertinat et al. (2009c). It is well known the BitTorrent's success for content downloading. However, it does not comply with the requirements of video streaming applications. GoalBit maintains the BitTorrent's philosophy mixing the tit-for-tat strategy with optimistic unchoking, extending the success in the peer selection process, that is a key element in the design of protocols for cooperation (Cohen, 2003). The clear weakness of BitTorrent for streaming applications is its chunk scheduling policy: Rarest First. The analysis of this paper shows its unacceptable latencies. We carried-out real-life experiments based on a GoalBit emulator, to figure-out the main characteristics of different chunk-scheduling policies. First, we took real traces from a previous GoalBit distribution of a football match. Therefore, we completely reproduce the real distribution (even with the identical protocol specification), but with a different piece selection strategy. The emulator reproduces all but network failures. The GoalBit protocol for cooperation keeps an urgent size of few pieces near the playback which are always asked for first, in order to attend the urgencies [for details of the GoalBit protocol, we refer the reader to Bertinat et al. (2009c)]. In the non-urgent size of the buffer, three different deterministic chunk scheduling policies were considered: Rarest First, Greedy

and the *W*-shaped member defined by $(I, J) = (16, 1)$. This test case considers a buffer capacity of $N = 40$ and 45 peers joining the network. In GoalBit, the video player halts when a video chunk is missing, and the player will skip frames. Therefore, users will have a re-buffering whenever a chunk is lost. Figures 5, 6 and 7 show respectively the first buffering time (or start-up latency), number of re-bufferings and their corresponding duration (in seconds), for each user.

Figure 5 Buffering-time for 45 peers using different chunk scheduling policies (see online version for colours)

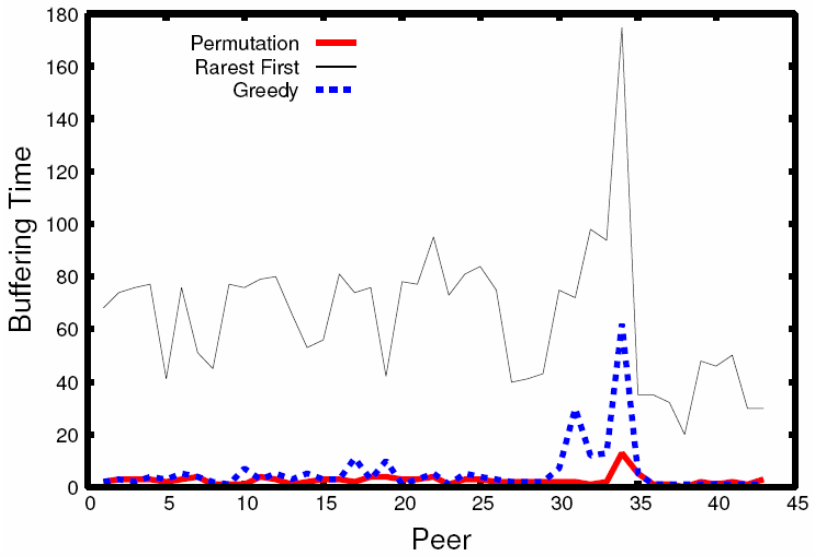


Figure 5 shows clearly that the Rarest First strategy has unacceptable start-up latencies for streaming purposes. In fact, users should wait more than one minute in average to start playing the video content following the Rarest First strategy. The new policy is competitive in relation with the Greedy strategy, having more reduced start-up latencies than Greedy for most of the peers. These latencies last no more than five seconds, which is a reasonable waiting time for users. Figure 6 illustrates the interruption of the video signal. The Greedy strategy clearly presents interruptions more often. Most of the peers live between four and six video interruptions when the Greedy policy is introduced. Rarest First trades-off video cuts with buffering times. However, this policy is discarded for live streaming purposes regarding start-up latencies in the order of minutes. When GoalBit follows the new permutation-based policy, peers live an intermediate number of video cuts, practically always lower than the Greedy policy (more specifically, only peers 15 and 35 had just one cut higher than Greedy following our permutation policy). Finally, Figure 7 shows that four peers experimented longer cuts when our permutation policy is introduced. However, the performance of the permutation policy is higher in the rest of the peers, with respect to both classical policies.

Figure 6 Number of re-bufferings (measure of playback continuity) for 45 peers using different chunk scheduling policies (see online version for colours)

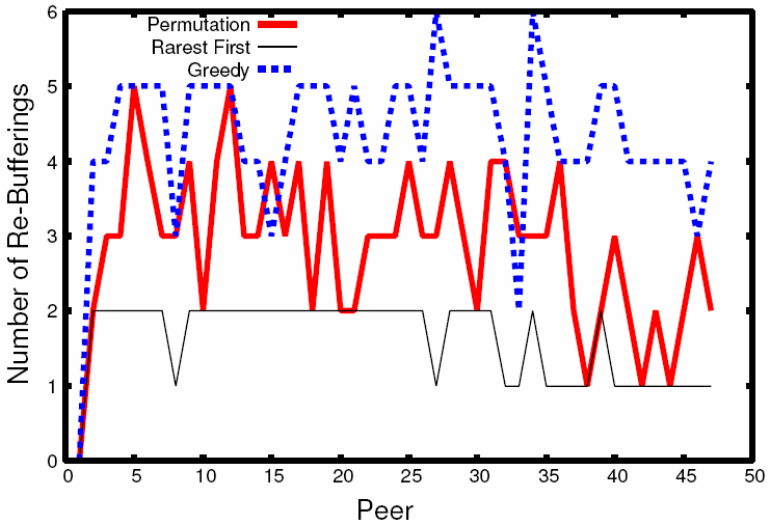
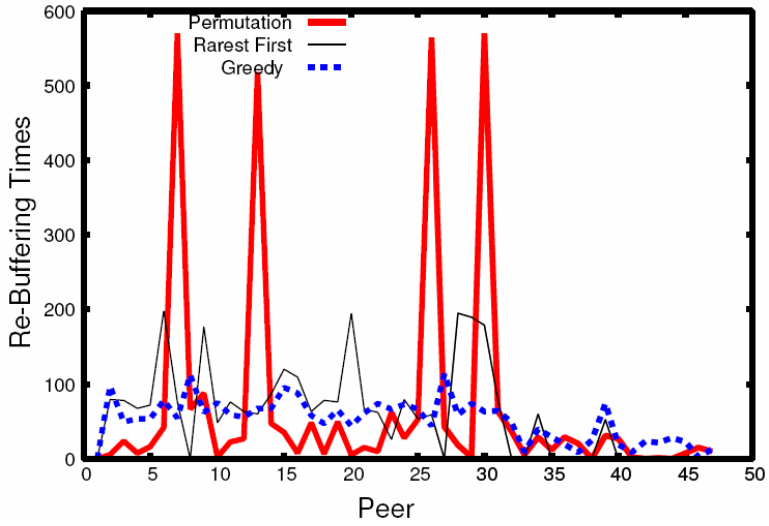


Figure 7 Average re-bufferings (in seconds) for 45 peers using different chunk scheduling policies (see online version for colours)



7 Conclusions and future work

In this paper, we address the chunk-scheduling policy of live streaming peer-to-peer networks, in a structural way. In the light of a pull-based cooperative system, we show that the BitTorrent chunk-policy, named Rarest First, is poor for live-streaming purposes. A counter-intuitive result is that the Greedy policy (request the nearest chunk to the

playback deadline) has unacceptable number of video-cuts. These results were stated from both theoretical and empirical aspects.

An in-depth analysis of a cooperative system for live streaming was presented, from both an ideal and feasible approach. The ideal approach fails, but gives an insight for the design of highly competitive chunk scheduling policies. Feasible policies were found via a sophisticated ant-colony-based meta-heuristic. It is inspired in the original ant-system design for the ATSP, and includes new ingredients selected for the nature of this problem. A local improvement was designed regarding the Cayley distance in the group of permutations. The result is a new chunk-policy which completely outperforms all possible mixtures of classical policies, as well as the average score of the *V*-shaped policies (which contains the asymptotic optimal for a single-playback score).

The new chunk scheduling policy was applied into a real P2P platform named GoalBit. The results were highly competitive, outperforming the classical Greedy and Rarest First policies.

This work reinforces the idea that Rarest First is not useful in its seminal BitTorrent concept for live streaming scenarios. The nature of the combinatorial problem here introduced could be correctly captured via an ATSP, and the ant-worker approach returned new chunk policies.

As a future work, we will consider node churn and heterogeneous peers in the network. We recently proposed an extension of this simple model for cooperation, including multiple classes of peers with heterogeneous capacities. This new model deserves a careful analysis, in order to design new policies for real-life peer-to-peer scenarios with no video cuts and high playback continuities.

Acknowledgements

We would like to thank the anonymous reviewers for their work and valuable comments that greatly improved the quality of our first manuscript. All their suggestions have been included.

References

Abboud, O., Pussep, K., Kovacevic, A., Mohr, K., Kaune, S. and Steinmetz, R. (2011) 'Enabling resilient P2P video streaming: survey and analysis', *Multimedia Syst.*, Vol. 17, No. 3, pp.177–197.

Arvind, V. and Joglekar, P.S. (2008) 'Algorithmic problems for metrics on permutation groups', *Lecture Notes in Computer Science*, Vol. 4910, pp.136–147, Springer, ISBN: 978-3-540-77565-2.

Beckers, R., Deneubourg, J. and Goss, S. (1992) 'Trails and U-turns in the selection of the shortest path by the ant Lasius Niger', *Journal of Theoretical Biology*, Vol. 159, pp.397–415.

Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P. (2009a) 'Systematic procedure for improving continuity and latency on a P2P streaming protocol', in *Proceedings of the 1st IEEE Latin-American Conference on Communications (LatinCom'09)*, IEEE Computer Society, Washington, DC, USA, pp.1–5.

Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P. et al. (2009b) 'A cop for cooperation in a P2P streaming protocol', in *Proceedings of the IEEE International Conference on Ultra Modern Telecommunications (ICUMT'09)*, IEEE Computer Society, Washington, DC, USA, pp.1–7.

Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P. et al. (2009c) 'GoalBit: the first free and open source peer-to-peer streaming network', in *Proceedings of the 5th International IFIP/ACM Latin American Conference on Networking*, ACM, New York, USA, pp.49–59.

Blum, C. (2005) 'Ant colony optimization: introduction and recent trends', *Physics of life Reviews*, October, Vol. 2, No. 4, pp.353–373.

Chatzidrossos, I., Dn, G. and Fodor, V. (2010) 'Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates', *Peer-to-peer Networking and Applications (PPNA)*, Vol. 3.

Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M. et al. (2003) 'Planetlab: an overlay testbed for broad-coverage services', *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 3, pp.3–12.

Cohen, B. (2003) 'Incentives build robustness in BitTorrent', in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*.

Deza, M. and Huang, T. (1998) 'Metrics on permutations, a survey', *Journal of Combinatorics, Information and System Sciences*, Vol. 23, pp.173–185.

Dorigo, M. and Gambardella, L.M. (1997) 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp.53–66.

Dorigo, M. and Stützle, T. (2004) *Ant Colony Optimization*, Bradford Company, Scituate, MA, USA, ISBN 0262042193.

Dorigo, M., Birattari, M. and Stützle, T. (2006) 'Artificial ants as a computational intelligence technique', Tech. Rep. 23, Institut de Recherches Interdisciplinaires, Universit Libre de Bruxelles.

Duan, H., Ma, G. and Liu, S. (2007) 'Experimental study of the adjustable parameters in basic ant colony optimization algorithm', *IEEE Congress on Evolutionary Computation*, Vol. 1, No. 1, pp.149–156.

Oncan, T., Altynel, Y.K. and Laporte, G. (2009) 'A comparative analysis of several asymmetric traveling salesman problem formulations', *Computers & Operations Research*, Vol. 36, No. 3, pp.637–654.

Qiu, D. and Srikant, R. (2004) 'Modeling and performance analysis of Bittorrent-like peer-to-peer networks', *SIGCOMM Comput. Commun. Rev.*, Vol. 34, No. 4, pp.367–378.

Romero, P., Robledo, F., Rodríguez-Bocca, P., Padula, D. and Bertinat, M.E. (2010) 'A cooperative network game efficiently solved via an ant colony optimization approach', in *Proceedings of the Seventh International Conference of Swarm Intelligence (ANTS'10), Lecture Notes in Computer Science*, Springer, London, UK, pp.336–343.

Veciana, G.D. and Yang, X. (2003) 'Fairness, incentives and performance in peer-to-peer networks', in the *Forty-first Annual Allerton Conference on Communication, Control and Computing*.

Yang, X. and de Veciana, G. (2004) 'Service capacity of peer to peer networks', in *INFOCOM 2004, Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, Vol. 4, pp.2242–2252.

Zeinalipour-Yatzi, D. and Folias, T. (2002) 'A quantitative analysis of the gnutella network traffic', Tech. Rep., Department of Computer Science, University of California, Riverside.

Zhao, B.Q., Lui, J.C.S. and Chiu, D.M. (2009a) 'Exploring the optimal chunk selection policy for data-driven P2P streaming systems', *P2P'09: IEEE International Conference Peer-to-Peer Computing*, pp.271–280.

Zhao, B.Q., Lui, J.C.S. and Chiu, D.M. (2009b) 'Exploring the optimal chunk selection policy for data-driven P2P streaming systems', in *Peer-to-Peer Computing*, pp.271–280.

Zhou, Y., Chiu, D.M. and Lui, J. (2007) 'A simple model for analyzing P2P streaming protocols', in *Proceeding of the IEEE International Conference on Network Protocols (ICNP'07)*, Beijing, China, pp.226–235.