Optimum Piece Selection Strategies for A Peer-to-Peer Video Streaming Platform

Pablo Romero, Franco Robledo Amoza, Pablo Rodríguez-Bocca

Laboratorio de Probabilidad y Estadística,
Facultad de Ingeniería, Universidad de la República
Julio Herrera y Reissig 565,
11300, Montevideo, Uruguay

Abstract

The Client-server architecture is still popular due to its high predictable service and performance. However, it is not bandwidth scalable. An alternative setup for Internet video-streaming is offered by the peer-to-peer architecture, in which peers are servers as well as clients. Peers basically communicate in a three-level based policy. First, they meet other peers with common interests: this is called *swarming*. Then, each peer selects a small number of them for cooperation, called the *peer selection strategy*. In the last step peers cooperate sending pieces, defining the *piece selection strategy*.

This paper is focused on piece selection strategies. We propose an in-depth analysis of a simple cooperative model. In this model the issue is to find the best order in which pieces should be obtained. In the first stage, we introduce a Combinatorial Optimization Problem (COP), which maximizes the average user experience for video streaming services, and has a permutation as the decision variable. Its hardness motivates us to approximately solve it via an Ant Colony Optimization-based heuristic.

The main theoretical contributions are twofold: the introduction of a new piece selection strategy with better results in contrast with the ones found in the literature, and a systematic way of computing new piece selection strategies with high quality. The practical contribution is the incorporation of a new piece selection strategy in a live peer-to-peer streaming platform, with remarkable performance in relation with classical strategies.

Keywords: Peer-to-peer, Piece selection strategies, COP

1. Introduction

Internet-based multimedia systems have many different architectures, depending on their sizes and on the popularity of their contents. The majority of them have a traditional CDN (Content Delivery Network) structure [9, 34], where a set of datacenters absorbs all the load, that is, concentrates the task of distributing the content to the customers. This is, for instance, the case of msnTV, YouTube, Jumptv, etc., all working with video content.

Another popular alternative consists in 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 [24, 16, 18]. 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 network. For this reason, P2P networks are said to *scale* well.

Nowadays P2P networks play an important role because of their popularity and their impact on Internet traffic. Some commercial P2P networks for live video distribution are available, all of them with proprietary source-codes and protocols. The most successful are PPlive [21, 15], SopCast [28], PPstream [22], TVAnts [31] and TVUnetwork [32].

On one hand, the dynamism and freedom of peers in a P2P network are attractive and powerful tools for the users. On the other hand they impose many challenges on architecture design and protocols to share information [26, 37]. The design of resilient peer-to-peer live streaming must cope with stringent timing constraints and node-churn, which is the unpredictable peer-arrival and departure. A recent survey on hints for a resilient design of streaming networks is [1]. A video frame has to reach its play-out time; otherwise the quality of experience is degraded. Moreover, many nodes only remain connected for a few minutes [29]. Several deployments and measures on P2P networks for live video distribution [27, 30, 2] confirm that the delay and play-out losses represent the most important factors in the quality of experience perceived by end users (see also [25, 19] for related details). Therefore, a well-designed piece selection strategy is essential in order to obtain high play-out continuity and low latency in a P2P streaming network [8].

There are three kinds of streaming services, that differ in generation, distribution and synchronization between peers, to know: file sharing, video on-demand and live-streaming. In file sharing, the file is available only after complete downloading. In video on-demand, the stream is distributed only when users demand it. The third service is live-streaming. Here, all users must be synchronized watching at the same instant, and the video stream is distributed and generated simultaneously. An inspirational system for file sharing and fast propagation is called BitTorrent [8]. There are many papers that show BitTorrent works for both off-line and on-demand services. However, it is not well-adapted for live-streaming requirements. An enormous effort of the scientific community is pointing to understand BitTorrent's deficiencies, and finally provide a full-scalable triple-play BitTorrent compatible with the three streaming modes. Diverse mathematical models try to understand the behavior and scalability of BitTorrent-based systems, including Markov Chains [38], Fluid Models [23], Branching Processes [33, 36] and Marginal Probabilities [39], among many others.

We developed an in-depth analysis of the model stated in [39] mainly because it is simple and captures two fundamental notions of live-streaming scalable architectures: cooperation and synchronization. There, the authors designed a cooperative pull process, where peers cooperate with each other in order to recover a video streaming delivered by a single source-node. The time is discretized, and requesting peers use a random peer selection policy to request for a new video piece. The aim is to find an optimal piece selection strategy, that dictates the order in which pieces must be downloaded to achieve high continuity and low buffering times. In this paper, we present a new strategy that has better results than previously considered proposals. For its design, we define a Combinatorial Optimization Problem (COP), translated into a suitable formulation of an Asymmetric Traveling Salesman Problem (ATSP). The latter is solved meta-heuristically following an Ant Colony Optimization (ACO) approach, which is inspired in the way ants find the shortest path between their nests and their food [4]. We refer the reader to [10, 12, 7, 11] for an in-depth analysis of this nature-inspired heuristic.

This paper is organized as follows. Section 2 describes a simple model of piece selection strategies proposed in [39]. Section 3 introduces a new family of piece selection strategies, its basic properties and an ideal approach. Sec-

tion 4 contains a combinatorial optimization problem (COP) whose decision variables are permutations. A methodology for its meta-heuristic resolution is also developed here. This process enables to obtain new piece selection strategies. Section 5 contrasts the new piece selection strategy with classical ones from both theoretical and empirical aspects. In the light of the mathematical model, we present analytical comparisons between the proposed strategies and previous ones. Then, the new piece selection strategy is applied into a real peer-to-peer platform called GoalBit [14, 6], showing the practical advantages of the new strategy proposed. Finally, Section 6 contains the main conclusions of this work.

2. Mathematical Model

The description of this mathematical model is taken from [39] and its most modern version, detailed in [38]. Consider a closed fully-connected P2P live-streaming system which needs to serve M identical peers. This P2P system has a logical server $\mathcal{S}$, which organizes the video content into a stream of pieces, sent in playback order. We model this large scale network as a discrete-time system. At each time slot, the server $\mathcal{S}$ uploads one video piece to one peer uniformly chosen at random. Each piece has a sequence number, starting from 1. Therefore, at time slot t, the server randomly selects one peer and uploads the video piece of sequence number t to this randomly selected peer.
Each peer needs to receive and buffer these video pieces from the P2P streaming system. To achieve this, each peer holds a local buffer $\mathcal{B}$, which can cache up to N video pieces. Position $\mathcal{B}_1$ stores the newest video piece that the server $\mathcal{S}$ is uploading in the current time slot, whereas position $\mathcal{B}_N$ is used to store the oldest video piece, that is currently being played back. In other words, when server $\mathcal{S}$ is uploading a piece with sequence number $k \geq N$, the video piece $k - N + 1$ is being played back by the peer (provided that the video piece is available in $\mathcal{B}_N$). At the end of each time slot, the video piece in $\mathcal{B}_N$ is discarded, and all pieces will be "shifted right" by one buffer position: video piece in $\mathcal{B}_i$ will be shifted to $\mathcal{B}_{i+1}$ for each $i = 1, \ldots, N-1$. A distortion is perceived on the screen if the peer could not obtain the current piece in time. We assume that all peers are synchronized in the buffer consumption. See Figure 1 for a graphical description.

Peers need to collaborate with each other to minimize their chance of losses and delays. As a consequence, pieces can also be obtained from other

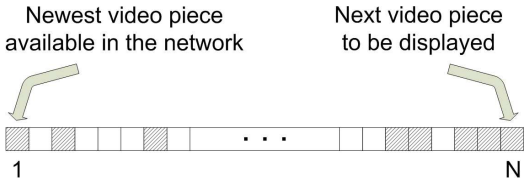


Figure 1: Buffer model for each peer. Position $\mathcal{B}_1$ has the newest video piece in the buffer, and $\mathcal{B}_N$ the piece being displayed at the screen.

peers in a pull-based process (i.e. pressed by downloader needs). Peer A randomly selects another peer B within the network. In general, and following some specific strategy, peer A will look at a given position in its buffer. If it is empty, it will request peer B for this missing piece. If the initial buffer position is already filled, the requesting peer shifts to some other position and the same iteration is repeated until it finds an empty one. If B does not have the corresponding piece, then A can ask for another piece. This scheme leads either to a success, when A finally gets a video piece from B, or to a failure, when there is no empty position in A's buffer that can be filled by a piece coming from B. The *extension* of the query is the number of buffer positions that A needs to examine in order to get a new piece. It is assumed that the whole query lasts no more than one time slot, and every peer obtains no more than one piece during a time slot (the peer chosen by the server does not ask for more video pieces).

Let us call p_i the probability that a peer has the correct video piece in $\mathcal{B}_i$. By symmetry, if all peers use the same strategy, p_i is independent of the peer. In stationary state, it does not depend on time either. Consider that a particular peer A selected a peer B to download a piece. Using a specific piece selection strategy, suppose that at some time, the piece corresponding to A's buffer position $\mathcal{B}_i$ is missing, so, it is desired by peer A and owned by peer B. The probability of this event is denoted by s_i (for the reasons mentioned above, it only depends on i).

Definition 2.1. *The* buffer-map $(p_1, \ldots, p_N)$ *is the probability that peer owns a video chunk in the buffer-cell* $\mathcal{B}_i$*. The number* $C = p_N$ *is the playback-delivery ratio, or* continuity*.*

In symmetric conditions, peers will have the same buffer-map p_i, and the buffering time can be measured ([39]):

Definition 2.2. *The* buffering-time *or latency for a peer can be found by*

$L = \sum_{i=1}^{N} p_i$, *and represents the expected time (measured in slots) a joining peer in the system should wait in order to reach the buffer-map* p_i*, starting with an empty buffer.*

The function mapping $i \in \{1, \ldots, N-1\}$ into $s_i \in [0, 1]$ is called the *selection* function of the strategy. It is proved in [39] that:

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

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

This has a clear interpretation: the probability of being selected by the server is one out of M. There are two mutually disjoint and exhaustive ways to get position $\mathcal{B}_{i+1}$ in time slot t_{x+i+1}: by shifting with time (the peer already had the piece at position $\mathcal{B}_i$ in time slot t_{x+i}) or via a query (the peer did not have that piece, but could obtain it in time slot t_{x+i}). The first event has probability p_i, and the second $(1 - p_i)p_i s_i$.

Observe that the buffer-map p_i is increasing in the playback direction: $p_1 < p_2 < \ldots < p_N$). We will measure the performance of the system with the playback delivery ratio and the start-up latency or *buffering time*.

It is worth noticing that the start-up latency can be tuned as desired in a real platform. It is not recommended to start playback before reaching the stationary state, because the user will experience many video gaps (poor continuity). Additionally, it is not useful to start lately after the stationary state. Indeed, it carries additional delays and the playback delivery ratio does not improve. As a consequence, the start-up latency is typically configured as the time needed to reach the stationary state.
This paper is focused on proposing a new piece selection strategy with high continuity C, and at the same time, low latency L.

3. General Permutation-Based Strategies

3.1. Definition

A natural way to obtain strategy-diversity is to use an arbitrary permutation to decide the order of a query. Let us consider the set of permutations Π_{N-1} of the first $N-1$ buffer positions:

$$\Pi_{N-1} = \{\pi : \{1, \ldots, N-1\} \mapsto \{1, \ldots, N-1\},\, \pi(i) \neq \pi(j)\, \forall i \neq j\}.$$

For each permutation $\pi \in \Pi_{N-1}$ we can associate the following piece selection strategy. Let us call A the requesting peer and B the requested one. In the first step, A examines its buffer at position $\pi(1)$. If that piece is missing and B has it, the download is performed. Otherwise (either because peer A owns that piece or B does not), A shifts to position $\pi(2)$ and the process is repeated (until there is either success or failure, the two only possible final results). This scheme leads to $(N-1)!$ different strategies. We call them *permutation-based* strategies. For the strategy associated with permutation $\pi \in \Pi_{N-1}$, the corresponding strategy function is related with the buffer-map:

$$s_{\pi(i)} = (1 - \frac{1}{M}) \prod_{j=1}^{i-1} \Big(1 - p_{\pi(j)}(1 - p_{\pi(j)})\Big). \tag{3}$$

Equation (3) can be interpreted in this way: peer A will ask for position $\pi(i)$ only if it was not selected by the server and every buffer position $\mathcal{B}_i$ such that $j < i$ meets one of the next two conditions: peer A already owns the piece at position $\pi(i)$, or neither A nor B owns it. It is interesting to notice that other strategies could be considered, for example using random variables to decide which position of the buffer to ask for next. In this paper we will restrict the attention to deterministic strategies, and we suspect that under this model, permutation-based strategies capture all of them. Moreover, some permutation-based strategies outperform strategies previously proposed, as we will confirm in the final results of our proposal.

The Greedy notion for this problem is always to ask for the nearest piece of the playback. Contrarily, the Rarest First strategy (as its name predicts) always asks for the rarest piece first (starting at position 1 and then increasing towards the playback direction), trying to balance the quantity of rarest pieces in the network. Rarest First is widely adopted by BitTorrent, with great success for downloading purposes. We shall refer to Greedy and Rarest First as *classical strategies*. Both strategies and a Mixture of them were proposed in [39]. A brief analysis given in Appendix 7 evidences a poor performance of the Rarest First strategy for streaming applications, tending to exhibit unacceptable large latencies. Moreover, the Greedy strategy presents poor continuity as emulations of a real platform from Section 5 indicate. Now, let us analyze some properties of the Permutation-Based strategies, which will guide us to outperform classical strategies and their Mixture.

3.2. Properties

The family of permutation-based strategies Π_{N-1} enjoys many useful properties, which guide us to define algorithms to find efficient ones. The first is that they are a super-set that includes previous classical strategies (and their mixture). Observe that the identity permutation $\pi(i) = i$ and the reverse one ($\pi(i) = N - i$) define the Rarest First and Greedy strategies respectively. For any given index $m : 1 \leq m \leq N - 1$, there is a Mixture between Greedy and Rarest first, captured with the following permutation π_m:

$$\pi_m(i) = i, \quad i = 1, \ldots, m;$$
$$\pi_m(i) = N - (i - m), \quad i = m + 1, \ldots, N - 1.$$

Clearly, all mixtures of the classical strategies are permutation-based strategies. This trivial fact confirms that the quality of the best permutation strategy will not decay. In fact, better strategies can be found.

Lemma 3.1. Degradation in the Selection
The sequence $s_{\pi(i)}$ is strictly monotone decreasing.

Proof. By (3) we have that $s_{\pi(i+1)} = s_{\pi(i)}[1 - p_{\pi(i)}(1 - p_{\pi(i)})] < s_{\pi(i)}$. □

Definition 3.2. *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.*

Let $x = (x_1, \ldots, x_{N-1})$ and and $y = (y_1, \ldots, y_{N-1})$ be injective real-valued vectors. There are unique permutations π_x and π_y such that $x_{\pi_x(1)} > x_{\pi_x(2)} > \ldots > x_{\pi_x(N-1)}$ and $y_{\pi_y(1)} > y_{\pi_y(2)} > \ldots > y_{\pi_y(N-1)}$

Definition 3.3. *The Cayley pseudo-distance between the vectors x and y is $d(x, y) = d(\pi_x, \pi_y)$.*

Corollary 3.4. Approximation Strategy Property
For every injective real-valued sequence $(x_1, \ldots, x_{N-1})$, there is only one permutation π whose selection strategy s verifies that $d(x, s) = 0$.

Proof. By Lemma 3.1 the evidence is the only permutation π that complies:

$$x_{\pi(1)} > x_{\pi(2)} > \cdots > x_{\pi(N-1)}$$

□

So far, we know how to approximate a given injective vector with a feasible selection strategy, choosing an appropriate permutation strategy. However, we want to have a full comprehension of the relation between our permutation π and the buffer map $(p_1, \ldots, p_N)$, which determines both the delivery ratio p_N and buffering times by $\sum_{i=1}^{N} p_i$.

Definition 3.5. *The ideal buffer-map p' has the lowest buffering time, subject to perfect delivery ratio: $p'_i = \frac{1}{M}$, $\forall i \in \{1, \ldots, N-1\}$ and $p'_N = 1$.*

The next lemma is simple but pessimistic. It assures that there is no strategy that can achieve perfect delivery ratio. Therefore, the ideal buffer-map is not achievable, and users will eventually have a cut in the video sequence.

Lemma 3.6. *Unless the network has exactly one peer, the strict inequality $p_N < 1$ holds.*

Proof. We will prove a stronger statement: $p_i < 1, \quad \forall i \in \{1, \ldots, N\}$. By (1) we have that $p_1 = \frac{1}{M} < 1$, given that there is more than one peer inside the net. This proves the base step.
Let us suppose now that $p_h < 1$ for some $h : 1 \leq h < N$, and s an arbitrary piece selection function. Then, Equation (2) provides the next recursion: $p_{h+1} = p_h + (1 - p_h)p_h s_h$. We must recall that by its definition s is a probability, so $s_h \leq 1$. Thus: $p_{h+1} \leq p_h + (1 - p_h)p_h < p_h + (1 - p_h) = 1$. □

A simple link between the buffer-map and selection strategy is offered by Equation (2).

Definition 3.7. *Given a desired buffer-map p, the corresponding* ideal selection strategy *s^{id} can easily be obtained with a restatement of* (2)*:*

$$s_i^{id} = \frac{p_{i+1} - p_i}{(1 - p_i)p_i}, \forall i \in \{1, \ldots, N-1\}. \tag{4}$$

Let us recall that we search for permutations which achieve high continuity and low latency. The Approximation Strategy Property leads us to invert the search order, following these stages:

1. Choose an ideal buffer-map $p = (p_1, \ldots, p_{N-1})$.
2. Find the corresponding ideal strategy s^{id} with Equation (4).
3. Find the only permutation π such that $s^{id}_{\pi(1)} > s^{id}_{\pi(2)} > \ldots > s^{id}_{\pi(N-1)}$.

Thanks to the Approximation Strategy Property, we have that the strategic sequence s associated with permutation π verifies that $d(s, s^{id}) = 0$. Hence, we are able to "imitate" the ideal vector s^{id} with a feasible strategic vector s. Nevertheless, this imitation does not assure that the respective buffermaps are similar. In order to solve the puzzle of searching for high quality permutations, it is necessary to *evaluate the continuity and latency of a given permutation*:

Definition 3.8. *For every permutation π, the Non-Linear System $NLS(\pi)$ consists in the $N-1$ equations* (2)*, the $N-2$ equations given by* (3) *and the unknowns $\{p_i\}_{i=2,\ldots,N} \cup (\{s_i\}_{i=1\ldots,N-1} - \{s_{\pi(1)}\})$:*

$$NLS(\pi): \begin{cases} p_1 & = \frac{1}{M} \\ p_{i+1} & = p_i + (1-p_i)p_i s_i, \quad \forall i = 1, \ldots, N-1 \\ s_{\pi_1} & = 1 - \frac{1}{M} \\ s_{\pi(i+1)} & = s_{\pi(i)}(p_{\pi(i)} + (1-p_{\pi(i)})^2) \quad \forall i = 1, \ldots, N-2 \end{cases}$$

Solving the non-linear system $NLS(\pi)$ (for example with the Newton-Raphson method), it is possible to evaluate the performance for any particular permutation.

In a first attempt to find an optimal permutation, we consider the control system illustrated in Figure 2. This system shows the *reverse design* recently stated: the input is the desired probability vector p_i, and the output is its best approximation p_i^*. We call this serial blocks the *Follower System*.

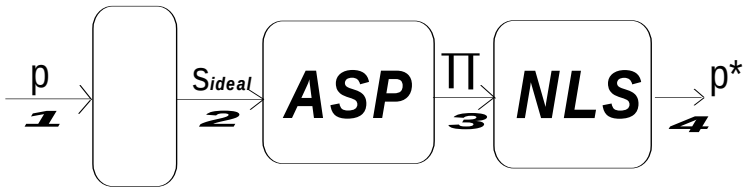


Figure 2: Follower System: Applies the Approximation Strategy Property (ASP) and constructs a permutation π that follows the corresponding strategy s^{id} (that would get the exact probability of occupation of the input).

Several inputs were injected into the Follower System to test its performance. An ambitious design of the input vector p is an exponential sequence

with unit playback continuity, because of its regularity and low values of latency:

$$p_{\epsilon_i} = M^{\frac{N-i}{1-N}}, \quad i = 1, \ldots, N. \tag{5}$$

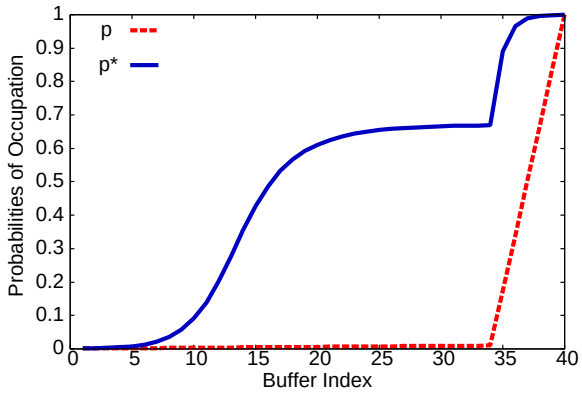
However, the output can be analytically found in this case. Note that $p_{\epsilon_{i+1}} = M^{\frac{1}{N-1}} p_{\epsilon_i} > p_{\epsilon_i}$, and the corresponding ideal selection strategy s^{id} respects the following identity:

$$\begin{aligned} \frac{s^{id}_{i+1}}{s^{id}_i} &= \frac{p_{\epsilon_{i+2}} - p_{\epsilon_{i+1}}}{(1 - p_{\epsilon_{i+1}})p_{\epsilon_{i+1}}} \frac{(1 - p_{\epsilon_i})p_{\epsilon_i}}{p_{\epsilon_{i+1}} - p_{\epsilon_i}} \\ &= \frac{1 - p_{\epsilon_i}}{1 - p_{\epsilon_{i+1}}} > 1, \quad \forall i \in \{1, \ldots, N-2\} \end{aligned}$$

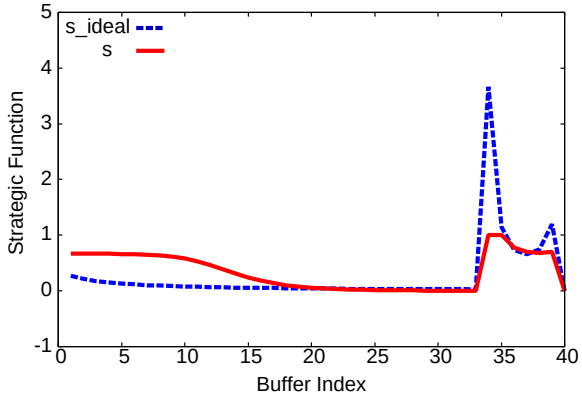
This means that the strategy to simulate is increasing, and the output permutation is $\pi(i) = N - i$. The output falls into the Greedy strategy, something not desirable.
Then we input the Ramp Vector shown in Figure 3(a) for the case $M = 1000$, $N = 40$. The corresponding ideal selection strategy is not feasible, because its magnitude exceeds the unit (it is not a probability). However, the output shows a key element of this system. The two pairs of functions p, p^* and s and s^{id}, are contrasted in Figure 3(a) and Figure 3(b) respectively.

The first observation from Figures 3(a) and 3(b) is that this Follower System fails again when trying to follow the segmented buffer-map, and shows that the first idea does not work as desired. However, the experience with the Follower System shows us how a direct peak in the permutation strategy (see Figure 3(b)) generates an abrupt change in the buffer-map (change in slopes of Figure 3(a)). The position of the peak plays a critical role, because if it is next to the playback the continuity will be poor (as in the case of Greedy). On the other hand, a high latency will be carried out whenever the peak is chosen far away from the playback (for instance, with the Rarest First strategy). Absolute maximums are avoided in order not to get high latencies. These observations motivate us to introduce the following:

Definition 3.9. *For each pair of naturals* $(I, J) : I + J \leq N - 1$*, there is*



(a) Segmented buffer-map and its output.



(b) Ideal strategy s_{id} and feasible strategy s.

Figure 3: Ideal vs. feasible buffer-maps and strategies for the case $M = 1000$, $N = 40$.

one permutation of the W-Shaped Policies*, that can be expressed as follows:*

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

The reader can check that these permutation-based strategies present a W-shaped buffer-priority. The I buffer-cells nearest-to-the-deadline have the highest priority, whereas the J far-away follow in priority. Then, a zig-zag priority is defined in the middle of the buffer (the $\wedge$-part of the W priority).

Curiously, in [38] the authors analyzed the same model via Markov-Chains in a continuity-driven fashion, and suggest another sub-family of V-Shaped strategies:

Definition 3.10. *Let $k \in \{1, \ldots, N-1\}$ be the buffer-cell with the lowest priority (i.e. $\pi(N-1) = k$). 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 strategies 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 number of W-Shaped members is the cardinal $\{(I, J) : I + J \leq N - 1\}$, or the cardinal of natural solutions to the equality $I + J + K = N - 1$. By elementary combinatorics, this number is $C_2^{N+1} = \frac{N(N+1)}{2}$, polynomial in the buffer capacity N.

4. A Combinatorial Optimization Problem

In this section we define a measure of optimality for each feasible permutation strategy, and an algorithm for its resolution, which returns outstanding strategies with high quality. Let X_π be the random variable that counts the number of steps needed to obtain a piece in a successful query, with the permutation-based strategy π.

Definition 4.1. *The quality of the strategy associated with permutation π is the expected extension of a successful query: $E(X_\pi)$.*

It is desirable that this expected value is as large as possible, given that an extension means that the peers have many filled buffers (the needed time is not a problem because it is assumed that at worst, the longest one is shorter than a time slot). The goal of the combinatorial problem here proposed is to maximize $E(X_\pi)$ among all possible permutation-based strategies, subject to the constraints given in the Non-Linear System 3.8. Specifically:

$$\max_{\pi \in \Pi_{N-1}} \quad E(X_\pi) \tag{6}$$

$$s.t. \quad p_i \quad \texttt{Complies with} \quad NLS(\pi) \tag{7}$$

Finally, the following result provides a method to evaluate $E(X_\pi)$.

Proposition 4.2. *The expected extension of a query needed to obtain a piece with the strategy associated with permutation $\pi \in \Pi_{N-1}$ is:*

$$E(X_\pi) = \frac{M}{M-1} \sum_{i=1}^{N-1} i\big(p_{\pi(i)+1} - p_{\pi(i)}\big). \tag{8}$$

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

$$\begin{aligned}
E(X_\pi) &= \sum_{i=1}^{N-1} i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \\
&= \sum_{i=1}^{N-1} ip_{\pi(i)}(1-p_{\pi(i)}) \prod_{j=1}^{i-1}\big(1-p_{\pi(j)}(1-p_{\pi(j)})\big) \\
&= \frac{1}{1-\dfrac{1}{M}} \sum_{i=1}^{N-1} ip_{\pi(i)}(1-p_{\pi(i)})s_{\pi(i)} \\
&= \frac{M}{M-1} \sum_{i=1}^{N-1} i\big(p_{\pi(i)+1} - p_{\pi(i)}\big).
\end{aligned}$$

□

The Cayley distance turns to be useful in the design of a local search:

Corollary 4.3. *For every $\pi \in \Pi_{N-1}$, define $\mathcal{N}(\pi) = \{\pi^* : d(\pi, \pi^*) = 1\}$ as the neighborhood set for the permutation π. Then, $\{\mathcal{N}(\pi)\}_{\pi\in\Pi_{N-1}}$ is a neighborhood structure for the maximization problem $\max_{\pi\in\Pi_{N-1}} E(X_\pi)$.*

4.1. Translation of the Problem

An N-clique allows to get a bijection between a cycle and a given permutation. We translate the COP into an instance of the well-known Asymmetric Traveling Salesman Problem (ATSP [20]) using the next bijection.

Proposition 4.4. *An $N-$clique permits to obtain a bijection between a directed cycle that visits all its nodes and a permutation of $\{1, \ldots, N-1\}$.*

Proof. If $\{v_1, \ldots, v_N\}$ is the set of nodes of an N-clique and $\pi \in \Pi_{N-1}$, then π is in bijection with the cycle $v_N, v_{\pi(1)}, \ldots, v_{\pi(N-1)}$, where v_N is arbitrarily fixed as a starting node. □

4.2. Ants-Algorithm

Ants-Algorithm can be divided into four blocks (one line per each). The issue of Lines 1 and 2 is the translation of the problem, exploiting the bijection stated in Proposition 4.4 translating the proposed COP into the search of a minimum cost directed tour. In Line 1, Function $InitializeEdges$ constructs a weighted graph in a fully connected network with N nodes. In Line 2, the pheromones of each edge are initialized with Function $InitializePheromones$. The idea is to explore the space of permutations (cycles) via an Ant Colony Optimization approach (ACO), in which ants are placed in an auxiliary node N, and look for the cheapest tour. Line 3 contains precisely the ACO implementation (Function *ApplyACO*), that returns the most visited cycle π by these artificial ants. Finally, a concept of LocalSearch is introduced: π is replaced iteratively by its best neighbor (the best *swap*) until we find a local optimum or a number of iterations is reached. All functions are explained bellow.

Ants-Algorithm 1 Works in four steps. Functions *InitializeEdges* and *InitializePheromones* translate the COP into an ATSP. Functions *ApplyACO* and *LocalSearch* solved the translated problem.

Inputs:
Sub-family of permutations: $SubFamily$
Number of ants: $ants$
Number of iterations in ACO: $iterations$
Maximum number of iterations in the local search: n
Output:
Permutation π

1: $d = InitializeEdges(ants)$
2: $\tau = InitializePheromones(SubFamily)$
3: $\pi = ApplyACO(d, \tau, iterations)$
4: $\pi_{out} = LocalSearch(\pi)$
5: **return** π_{out}

4.2.1. Function *InitializeEdges*

Once we know the similarity between an oriented cycle in an $N-$clique and a permutation, the first stage of *Ants-Algorithm* consists in initializing the edges of the clique, which is implemented by Function $InitializeEdges$:

Function *InitializeEdges* **2** Initialization of the distances between the nodes of the graph. Each ant makes a stochastic tour with a Tabu-list in order not to visit the same node twice.

Inputs:
Number of ants: $ants$
Buffer size: N
Output:
Distances between nodes: $Distance$

1: $Distance = 1_{N \times N}$
2: $Q = E(X_{RarestFirst})$
3: **for** $i = 1$ to $ants$ **do**
4: $\quad C(i) = VisitedCycle(Distances)$
5: $\quad Distance = UpdateCost(Distances, C(i))$
6: **end for**
7: **return** $Distances$

Distances is a square matrix of size N that stores the directed-cost from node i to node j in its entry (i, j). In Line 1, all edges are initialized with unit cost. Then, a quality reference (the one of Rarest First) initializes Q_{max}. In Lines 3 to 6 an ant exploration block takes place. Each ant visits one cycle, that is interpreted as one solution by means of the bijection proved in Proposition 4.4. All cycles start by the auxiliary node N. Each ant stochastically chooses the next node with a Tabu-List in order not to visit the same node twice (Line 4). If an ant visited a partial path $x_1, \ldots, x_{j-1}$, then the jump-probability distribution for x_j is:

$$p_{x_j} = \frac{d(x_{j-1}, x_j)^{-1}}{\sum_{k \geq j} d(x_{j-1}, x_k)^{-1}}, \tag{9}$$

where $d(i, j) = Distances(i, j)$ for short, and the jump-rule $k \geq j$ is the Tabu-List. Finally, in Line 5 the distance of each recently visited edge is updated regarding the recent quality $E(X_\pi)$ and the best quality so far Q_{max}:

$$d((x_{j-1}, x_j)) = \frac{10(N - i)Q_{max}}{E(X_\pi)} \tag{10}$$

The cost is defined inversely proportional to the quality of the cycle. In this way, cheaper cycles are desirable. Let us also observe that factor $10(N - i)$

allocates higher weights to the first edges of each cycle. In this way, the cycles with different quality are more distinguishable.

Now, let us count the minimum number of ants needed to initialize distances between all edges. Each ant makes one cycle, and the $N-$clique has $C_2^N = \frac{N(N-1)}{2}$ edges. Each ant visits a total of $N-1$ edges (the last edge (x_{N-1}, N) of the cycle is not updated). Thus, at least $\frac{N}{2}$ ants are necessary to visit all edges. Given that ants can repeat the visit of edges with biased walks, it is normal to take a number of ants not lower than $2N$.

4.2.2. Function *InitializePheromones*

Once we have the clique with asymmetric distances defined between every pair of nodes i and j, *Ants-Algorithm* proceeds calling $InitializePheromones$, that assigns pheromones to each edge of the clique. The goal is to prepare the graph for a later ACO application, in which ants trace cycles with more quality. In order to initialize the pheromone of each edge, the sub-family of W-Shaped permutations is chosen as a seed, because of its richness and variable quality. The initialization block is identical to the one of $InitializeEdges$, and involves Lines 1 and 2. Next, the block of ants' cycles and pheromone update is also similar to the process of initialization of distances: the main difference is that now the cycles are chosen deterministically. Moreover, after each cycle the pheromones are updated according to the quality of the cycle. More explicitly, in Line 5 pheromones are updated with the following expression:

$$\tau(\pi(i), \pi(i+1)) = \frac{10(N-i)Q}{Q_{max}}. \tag{11}$$

As a consequence, the links with more pheromones are more likely to be visited by ants during ACO application. The ladder-factors $10(N-j)$ were fixed during Function $InitializeEdges$ as well. In this way we do not give priority to pheromones or distances beforehand.

4.2.3. ACO Implementation

Once we have defined a network with costs and pheromones of all edges (by Functions $InitializeEdges$ and $InitializePheromones$) the next step is to solve the corresponding ATSP by an ACO-based heuristic. Given that the exploration mechanism of the problem is similar to that applied by ants,

Function *InitializePheromones* **3** The pheromones for the later ACO application are initialized in accordance with the experience obtained from the Sub-family of permutations.

Inputs: Sub-Family of permutations: $SubFamily(I, J)$
Buffer Size: N
Output:
Pheromones of each edge: $\tau(i, j)$

1: $\tau \leftarrow 1_{N \times N}$
2: $Q_{max} = Q_{RarestFirst}$
3: **for** $\pi \in SubFamily$ **do**
4: $\quad Q = Quality(\pi)$
5: $\quad \tau = UpdateCost(\pi, Q)$
6: **end for**
7: **return** τ

its great success for ATSP from its beginnings [5], and its reduced computational effort and simplicity, the design of a meta-heuristic based on ACO mixed with a local search is here chosen.
Before studying the details of our particular definition of Function *ApplyACO*, it is worth considering the classical parameters of the first Ant-System implementation in TSP, as it can be found in the specialized literature [13, 3, 10, 11]. In all of them, the next four basic parameters are defined. One is the number of ants to use, always present in ACO implementations. Two parameters named α and β are the trail importance and visibility of the path respectively. These are positive values that permit either to give more weight to the pheromone trails or shorter steps. More specifically, if x_j is the actual node of an ant, it chooses the next node x_{j+1} according to the next probabilities [11]:

$$p_{x_{j+1}} = \frac{\tau(x_j, x_{j+1})^{\alpha} d(x_j, x_{j+1})^{-\beta}}{\sum_{y \in NotCycle\ y} \tau(x_j, y)^{\alpha} d(x_j, y)^{-\beta}} \tag{12}$$

where $\tau(x_i, x_j)$ is the matrix with pheromones for each edge, $d(x_i, x_j)$ the distance between nodes x_i and x_j and α and β the priority parameters to distances and pheromones respectively. Observe that an exploration with complete visibility and discarding the trail (i.e. $\beta > 0$ and $\alpha = 0$) is the *Greedy* heuristic for the TSP. In this way, Ant-System proposes a trade-off

between greediness and the *smell* of ants, based on pheromones. A concept of trail evaporation is considered as well. It plays the role of a trade-off between the preservation of the historical information of pheromones, and the inclusion of the new information: with $\rho = 1$, the whole historical information of pheromones is replaced by the new one (complete evaporation). On the contrary, $\rho = 0$ means no changes in pheromones. 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 minimization problem:

$$\tau(x_i, x_j)^{t+n} = \rho\tau(x_i, x_j)^t + \sum_{k=1}^{m} \frac{Q}{L_k} 1_{(x_i,x_j)\in Cyclek}, \tag{13}$$

where L_K is the length of the tour visited by ant $k \in \{1, \ldots, m\}$. 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 [11] for an overview of the Ant-System design, and its performance with respect to other classical meta-heuristics for the TSP.

In order to address the nature of our problem, we apart from their original approach in four aspects:

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 4.4.
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.

Function *ApplyACO* 4 The ACO implementation is based on the original Ant-System meta-heuristic. It introduces massive populations starting from an attractor nest, and weighted tours exploiting the previous experience of a sub-family of W-shaped permutations.

Inputs:
Cost of each edge: $d(i, j)$
Pheromones of each edge: $\tau(i, j)$
Number of ants: $ants$
Priority to pheromones: α
Priority to distances: β
Evaporation parameter: ρ
Output:
Most visited cycle: π

1: $Q_{max} = \frac{E(X_{RarestFirst})}{LatencyRarestFirst}$
2: **for** $i = 1$ to $ants$ **do**
3: $\quad \pi(i) = TabuTour(\tau, d, \alpha, \beta)$
4: $\quad \tau = NewPheromones(\rho, \tau, Q, Q_{max})$
5: **end for**
6: **return** $\pi = MostVisitedEdge(\pi_1, \ldots, \pi_{ants})$

Now let us attend Function $ApplyACO$, in which the previous concepts are introduced. The input parameters are the pheromones and distances matrices (obtained by $InitializeEdges$ and $InitializePheromones$) and the parameters $ants$, α, β and ρ previously explained. The output is a permutation π. Line 1 takes as a reference the score of the Rarest First strategy, normalized by its latency. Observe this normalization, that is justified because of the priority to the continuity in $E(X_\pi)$: this artifice is expected to provide solutions with more reduced latencies. Each time an ant finishes its cycle, the pheromone trace of all recently visited edges is updated, by using the quality normalized by latency. This is chosen because the sub-family of W-shaped permutations has an important margin to improve latency but continuity (this observation is confirmed by a polynomial revision of the sub-family). The cycles are stochastic, weighting costs and pheromones according to Equation (12). After each cycle completed by an ant, the pheromone's update takes place. The pheromone trails are updated regarding the quality

$Q = E(X_\pi)$ of the latest tour and Q_{max}:

$$\tau((x_j, x_{j+1})) = (1-\rho)\tau((x_j, x_{j+1})) + \rho\frac{10(N-j)Q}{Q_{max}\sum_{i=1}^{N} p_i}$$

The trail adaptation occurs only after the end of each tour. 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. Finally, the best cycle is obtained in a greedy fashion (Line 6, Function $MostVisitedEdge$), taking the most visited edge in each opportunity without making cycles.

4.2.4. Computational Effort

In order to understand the trade off between the computational effort and the quality of the solutions that *Ants-Algorithm* offers, let us count the quantity of operations, taking the number of evaluations of $E(X_\pi)$ as the basic operation. The next result summarizes the block that imposes the largest computational effort, and the convergence order with respect to the input N.

Theorem 4.5. *Denote N the buffer size and $T(N)$ the average time for evaluating the quality $E(X_\pi)$. If we assume that the number of ants and the maximum number of iterations have order $O(N)$, then the average total time for running* Ants-Algorithm *is:*

$$\tau = O(N^3 T(N)) \tag{14}$$

Proof. Let us study the number of evaluations for each of the next blocks: $InitializeEdges$, $InitializePheromones$, $ApplyACO$ and $LocalSearch$.

In $InitializeEdges$, all ants make a cycle, and there is one evaluation for each. Considering an order $O(N)$ of ants, this block is linear with the input.

Let us focus now on $InitializePheromones$. This function applies as many evaluations as the members of the W-Shaped Subfamily, which coincides with the cardinality of natural solutions for the inequality: $I + J \leq N - 1$. Defining $K = N - J - I - 1$, it is the same to count the natural solutions to all possible values for the rest $I + J + K = N - 1$, that equals

$C_2^{N+1} = \frac{N(N+1)}{2}$. Hence, Function *InitializePheromones* has a quadratic number of evaluations with respect to the input N.

The same argument holds for counting the computational effort during *ApplyACO*, with quadratic order.

Finally we have *LocalSearch*. The number of neighbors for a given permutation is $C_2^{N-1} = \frac{(N-1)(N-2)}{2}$. In order to find the best neighbor, it is necessary to evaluate all neighbors in each iteration. If the number of iterations is linear with N , then this block imposes the largest computational effort, whose order is N^3. If the average time for an evaluation is $T(N)$, then the total average time is $\tau = O(N^3 T(N))$. □

5. Results

5.1. Comparative Results

Let us determine *Ants-Algorithm* performance versus classical strategies [39]. The adaptation of ACO's parameters is based on [13], and then tuned in accordance with our specific problem. By fixing 100 ants (which is higher than $2N$ and trades computational effort) and tuning the other three parameters (searching in $0 < \rho < 1$, $0 < \alpha < 1$, $1 < \beta < 2$), we obtained highly competitive results implementing *Ants-Algorithm* with $\alpha = 0.4, \beta = 1.5, \rho = 0.5$ and $ants = 100$.

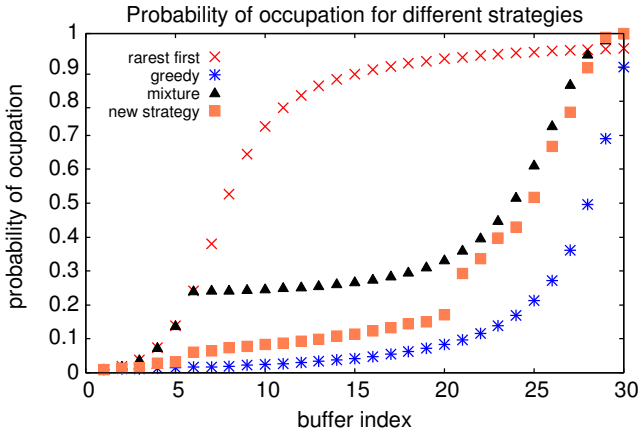


Figure 4: Comparison between different strategies

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
Ants-Algorithm	**0.9998**	**7.9821**

Figure 4 shows the result of a comparison between classical strategies versus *Ants-Algorithm* strategy for the common-network parameters $N = 30$ and $M = 100$. The best Mixture can be found in this case when $m = 10$, under consideration here. We carried-out an additional comparison between our new permutation strategy and the subfamily of V-shaped strategies introduced by the same authors which proposed the cooperative model [38]. We include in Table 1 the mixture with the highest continuity as well. The Average V-Shaped refers to the average performance of one-hundred V-Shaped policies randomly picked. Over those one-hundred samples, our permutation-policy has better play-out continuity 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 strategy. This results confirm a highly competitive trade-off between playback continuity and buffering-times of our proposal. The V-Shaped members define both high playback but high buffering times as well. In fact, the authors [38] focus the design on playback continuity only.

5.2. Empirical results

5.2.1. A Real Peer-to-Peer Platform

The real P2P platform considered for the test performance of our new piece selection strategies is GoalBit [14, 6]. BitTorrent's success for content downloading is well known. However, it does not comply the requirements of video streaming applications. GoalBit maintains BitTorrent's philosophy, mixing the tit-for-tat strategy with optimistic unchoking, and extending the success in the peer selection process (which is a key element in the design of protocols for cooperation [8]). The clear weakness of BitTorrent for streaming applications is its peer selection strategy: Rarest First. The analysis here

developed shows its unacceptable latencies. We refer to [14, 6] for details of the GoalBit protocol.

5.2.2. Results in a Real-Life Scenario

We carried-out real-life experiments based on a GoalBit emulator. First, we take 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. In the non-urgent size of the buffer, three different strategies were considered to analyze their performance: the Rarest First strategy, the Greedy strategy and the W-Shaped member specified in Definition 3.9 with $I = 16$ and $J = 1$. The test case considers $N = 40$ and 45 peers entering the network. Figures 5, 6 and 7 show respectively the start-up latency, number of re-bufferings and mean buffering time.

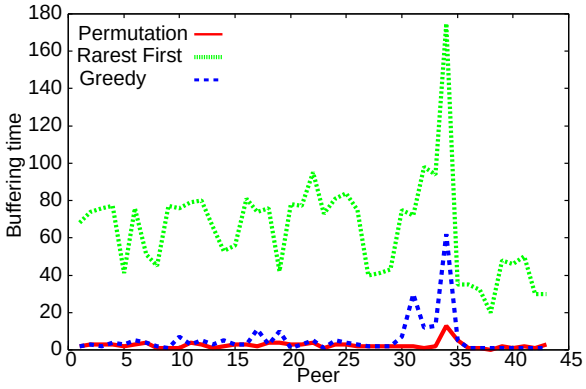


Figure 5: Start-up latency for different piece selection strategies. The figure also shows the iteration where each peer arrived at the network.

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 on average to start playing the video content following the Rarest First strategy. The new policy is competitive in relation with Greedy, having more reduced start-up latencies than Greedy for most of the peers.

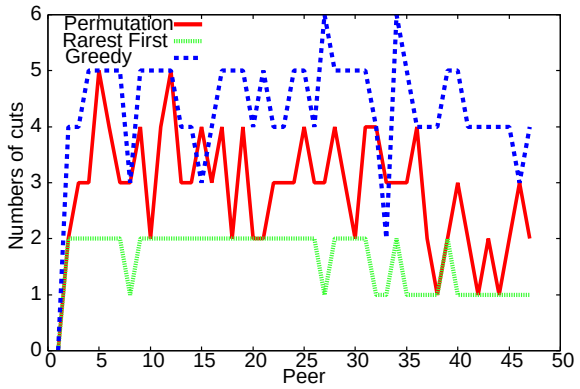


Figure 6: Number of cuts presented in the video for different strategies.

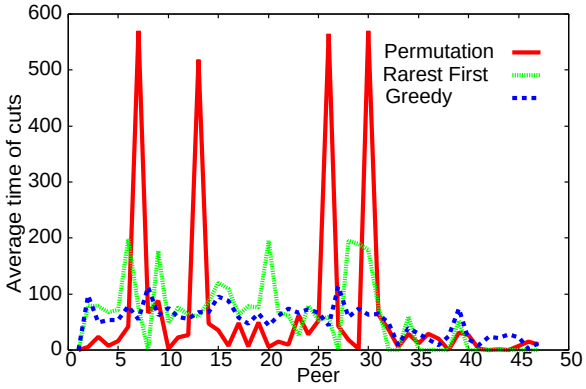


Figure 7: Average cut time for different strategies.

These latencies last no more than five seconds, which is a reasonable waiting time for end-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 not suitable for live streaming purposes because of its start-up latencies in the order of minutes. When GoalBit follows the new permutation-based policy, peers experience 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 experienced longer cuts when our permutation strategy is introduced. However, the performance of

our Permutation strategy is higher in the rest of the peers, with respect to both classical policies.

6. Conclusions and Future Work

This paper defines a new family of piece selection strategies based on permutations of the linear order of piece requests. This new strategy improved qualitatively the most frequently used piece selection strategies, named Greedy and Rarest First, in the real P2P platform GoalBit. It achieves the best continuity and at the same time, latency competitive with Greedy (whose continuity is poor).

Our deterministic design of piece selection strategies consists of an ideal and a feasible approach. The ideal approach proposes a Follower System that tries to follow a desired buffer-map. It does not work properly, but illustrates the playback-buffering trade-off. The feasible approach captures this trade off with a Combinatorial Optimization Problem. The methodology for its resolution is carefully elaborated, and includes a translation into an Asymmetric Traveling Salesman Problem (ATSP), an Ant Colony exploration and local search phase.

The weights of the ATSP were defined in accordance with biased ants tours. The final results show that the initial COP was correctly captured via the ATSP model, and that the Ant Colony Optimization led to better results than the ones obtained with classical strategies. Moreover, the results were better than those obtained by the members of the Sub-family of strategies, which served as a seed to initialize the pheromones for the ant colony optimization approach. Our future work is focused on the optimization of routing protocols in P2P streaming networks, complemented with the design of piece selection strategies here proposed. This should be directly applied to video streaming networks, and particularly to the GoalBit platform.

Acknowledgments

This work was partially supported by project “Sistema eficiente de distribución de video y TV en tiempo real” of the national Uruguayan telephony operator ANTEL, the French DGE project “P2Pim@ges”, and the French-Uruguayan ECOS project “Réseaux sans-fil de type mesh et applications multimédia P2P: outils pour la garantie de la qualité d’expérience”.

7. Appendix

7.1. Classical Strategies and a Mixture

There are two classical piece selection strategies, named Rarest First and Greedy, and a third one defined as a mixture of them. Rarest First enjoys a prestigious place nowadays, because of its origins with the popular BitTorrent system [17]. With a Rarest First strategy the peer chooses to download the pieces that are locally rarest, in order to guarantee a high piece diversity. In a streaming context, it basically consists in selecting a missing piece that the contacted peer has been searching initially far away from the playback (for rare pieces, because of monotonicity of p), that is, starting from the beginning of the buffer. This leads to the following expression for s_i [39]:

$$s_i = (1 - \frac{1}{M}) \prod_{j=1}^{i-1} \Big(1 - p_j(1 - p_j)\Big). \tag{15}$$

In other words, the server did not send the piece to the requesting peer (with probability $1-1/M$), and the event "the requesting peer does not have the piece and the contacted one does" with probability $p_j(1-p_j)$ is false from buffer positions 1 to $i-1$. The Greedy strategy is similar to the previous one, but first looks for the pieces nearest to the playback. This leads to the following expression for probability s_i:

$$s_i = (1 - \frac{1}{M}) \prod_{j=i+1}^{N-1} \Big(1 - p_j(1 - p_j)\Big). \tag{16}$$

It is interesting to notice that the piece selection functions s_i can be widely simplified in both strategies, as stated bellow:

Proposition 7.1. *The piece selection function for Rarest First complies with the equality:*

$$s_i = 1 - p_i, \quad \forall i \in \{1, \ldots, N-1\} \tag{17}$$

Proof. Direct by induction. □

Replacing (17) in (2) the next simplified recursion for Rarest First holds:

$$\begin{aligned} p_1 &= \frac{1}{M} \\ p_{i+1} &= p_i + p_i(1 - p_i)^2 \end{aligned} \tag{18}$$

In a similar way it can be proved a simplified expression for Greedy:

Proposition 7.2. *The following equality holds for Greedy:*

$$s_i = 1 - p_N + p_{i+1} - p_1, \quad \forall i \in \{1, \ldots, N-1\} \tag{19}$$

Figure 8 shows the Rarest First and Greedy strategies graphically. A mixture of these two strategies is possible, simply by dividing the buffer into two parts, from positions 1 to some index value $m : 1 \leq m \leq N$, and from $m+1$ to N, and applying the Rarest First strategy in the first section and Greedy in the second one. This combination offers a more reduced latency than Rarest First and still possesses good continuity [39]. However, the final results of this work show that the Mixture strategy can be improved by the new piece selection strategies introduced here.

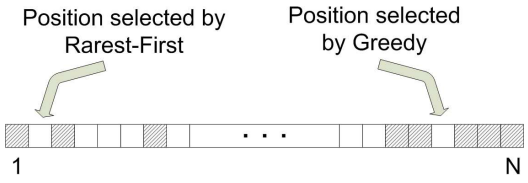


Figure 8: Rarest First and Greedy buffer strategies

In [39] and [35] it is stated that Greedy achieves low latency, but it is not as scalable as Rarest First, which presents unacceptable latencies. We will look at the performance of Rarest First. We will start showing the convergence properties in Rarest First, which describes its scalability. On the other hand, there is evidence that its latency makes its application to video streaming impractical. Greedy presents poor delivery ratio, as confirmed in the GoalBit platform.

The goodness of high continuities of Rarest First can be understood by the next two results:

Lemma 7.3. *In Rarest First:*

$$\lim_{N \to \infty} p_N = 1$$

Proof. By (2), sequence p_i is nondecreasing. Moreover, since it represents probabilities, it is clearly bounded by 1. Then its limit exists. If we denote $\alpha = \lim_{N \to \infty} p_N$ and take limits at both sides of (18), then:

$$\alpha = \alpha + (1 - \alpha)^2 \alpha \tag{20}$$

Consequently $\alpha = 0$ or $\alpha = 1$. But $\alpha \neq 0$ because $p_1 = \frac{1}{M}$ and p_i is nondecreasing. □

Theorem 7.4. *The convergence order in Rarest First is linear, and its convergence velocity is 1.*

Proof. By Lemma 7.3 we know that Rarest First converges to 1. Let us define the global error by $e_n = 1 - p_n$. Then, by using (18) we have that:

$$h_n = \frac{e_{n+1}}{e_n} = \frac{1 - p_n - (p_{n+1} - p_n)}{1 - p_n} = 1 - p_n(1 - p_n) \tag{21}$$

Lemma 3.6 assures that $e_n > 0, \forall n > 0$, so h_n is correctly defined. The result follows from taking limits on both sides, and using the fact that $\lim_{N\to\infty} p_N = 1$, as Lemma 7.3 guarantees. □

Moreover, given that $0 < p_n < 1, \quad \forall n > 0$ then $\frac{3}{4} \leq h_n < 1, \quad \forall n > 0$. By taking $r = \max_{n:1\leq n<N} \{h_n\} < 1$ we get that $e_N < r^{N-1} e_1 = r^{N-1}(1 - \frac{1}{M})$. This shows that the global error decreases at an exponential rate in Rarest First.
There is an absolute minimum for function $h_n = 1 - p_n(1 - p_n)$ in the index $i^* = arg\min_{1\leq i\leq N}\{|p_i - \frac{1}{2}|\}$. The biggest reduction in the global error occurs in the jump between p_{i^*} and p_{i^*+1}. The corresponding factor of the error reduction is near $\frac{3}{4}$ (or equal if $p_i^* = \frac{1}{2}$).

The previous properties show the strength of Rarest First in achieving high continuities. However, this strategy carries high latency, as it is explained below.
Let us consider the polynomial that expresses the jump between two consecutive steps $(p_{i+1} - p_i)$ in Rarest First:

$$P(x) = x(1 - x)^2 \tag{22}$$

When restricted to the interval $[0, 1]$, it presents a global maximum in $x = \frac{1}{3}$. This is equivalent to say that when the probability of occupation reaches $p_{i'} \approx \frac{1}{3}$, the sequence p_i imposes a big jump, whose magnitude is near $P(\frac{1}{3}) = \frac{4}{27}$ time slots. Moreover, all indexes bigger than i' impose a great amount of latency: $L = \sum_{i=1}^{N} p_i$, because the magnitude of p increases.
This is a first insight of the unacceptable values of Rarest First, when it is considered in real time applications.

[1] O. Abboud, K. Pussep, A. Kovacevic, K. Mohr, S. Kaune, R. Steinmetz, Enabling resilient p2p video streaming: Survey and analysis, Multimedia Systems 17 (2011) 177–97.

[2] S. Alstrup, T. Rauhe, Introducing Octoshape - a new technology for large-scale streaming over the Internet, Technical Report, European Broadcasting Union (EBU), 2005.

[3] D.C. Asmar, A. Elshamli, S. Areibi, A Comparative Assessment of ACO Algorithms Within a TSP Environment, In the 4th International Conference on Engineering Applications and Computational Algorithms (2005).

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

[5] H. Bersini, M. Dorigo, S. Langerman, G. Seront, L.M. Gambardella, Results of the first international contest on evolutionary optimization., in: International Conference on Evolutionary Computation, IEEE Press, 1996, pp. 611–5.

[6] M.E. Bertinat, D.D. Vera, D. Padula, F. Robledo, P. Rodríguez-Bocca, P. Romero, G. Rubino, GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network, in: To appear in Proceedings of the 5th international IFIP/ACM Latin American conference on Networking, ACM, New York, USA, 2009.

[7] C. Blum, Ant colony optimization: Introduction and recent trends, Physics of life Reviews 2 (october 2005) 353–73.

[8] B. Cohen, Incentives Build Robustness in BitTorrent, www.bramcohen.com 1 (2003) 1–5.

[9] C.D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, J.E.V. der Merwe, C.J. Sreenan, Enhanced Streaming Services in a Content Distribution Network, IEEE Internet Computing 5 (2001) 66–75.

[10] M. Dorigo, M. Birattari, T. Stutzle, Artificial Ants as a Computational Intelligence Technique, Technical Report 23, Institut de Recherches Interdisciplinaires, Universit Libre de Bruxelles, 2006.

[11] M. Dorigo, L.M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation 1 (1997) 53–66.

[12] M. Dorigo, T. Stutzle, Ant Colony Optimization., MIT Press, 2004.

[13] H. Duan, G. Ma, S. Liu, Experimental Study of the Adjustable Parameters in Basic Ant Colony Optimization Algorithm, IEEE Congress on Evolutionary Computation 1 (2007) 149–56.

[14] GoalBit - The First Free and Open Source Peer-to-Peer Streaming Network. Home Page, http://goalbit.sf.net/, 2008.

[15] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System, in: In Proc. of IPTV Workshop, International World Wide Web Conference.

[16] S. Horovitz, D. Dolev, LiteLoad: Content unaware routing for localizing P2P protocols, in: Proceeding of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS'08), Miami, USA, pp. 1–8.

[17] A. Legout, G. Urvoy-Keller, P. Michiardi, Rarest first and choke algorithms are enough, in: IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2006, pp. 203–16.

[18] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, Peer-to-Peer Computing, Technical Report HPL-2002-57, HP Labs, 2002.

[19] S. Mohamed, G. Rubino, A Study of Real–time Packet Video Quality Using Random Neural Networks, IEEE Transactions On Circuits and Systems for Video Technology 12 (2002) 1071–83.

[20] T. Oncan, Y.K. Altynel, G. Laporte, A comparative analysis of several asymmetric traveling salesman problem formulations, Computers & Operations Research. 36 (2009) 637–54.

[21] PPLive Home page, http://www.pplive.com, 2007.

[22] PPStream home page, http://www.ppstream.com/, 2007.

[23] D. Qiu, R. Srikant, Modeling and performance analysis of bittorrent-like peer-to-peer networks, SIGCOMM Comput. Commun. Rev. 34 (2004) 367–78.

[24] P. Rodríguez-Bocca, Redes de Contenido: Taxonomía y Modelos de evaluación y diseño de los mecanismos de descubrimiento de contenido., Master's thesis, Universidad de la República, Facultad de Ingeniería, Intituto de Computación. ISSN 0797-6410 INCO-RT-05-13, Montevideo, Uruguay, 2005.

[25] P. Rodríguez-Bocca, Quality-centric design of Peer-to-Peer systems for live-video broadcasting, Ph.D. thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, 2008.

[26] S. Saroiu, P.K. Gummadi, S.D. Gribble, A Measurement Study of Peer-to-Peer File Sharing Systems, in: Multimedia Computing and Networking.

[27] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, S. Tewari, Will IPTV ride the peer-to-peer stream?, Communications Magazine, IEEE 45 (2007) 86–92.

[28] SopCast - Free P2P internet TV, http://www.sopcast.org, 2007.

[29] K. Sripanidkulchai, B. Maggs, H. Zhang, An analysis of live streaming workloads on the internet, in: IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM Press, New York, NY, USA, 2004, pp. 41–54.

[30] S. Tewari, L. Kleinrock, Analytical Model for BitTorrent-based Live Video Streaming, in: 4th IEEE Consumer Communications and Networking Conference (CCNC'07), Las Vegas, NV, pp. 976–80.

[31] TVAnts home page, http://cache.tvants.com/, 2007.

[32] TVUnetworks home page, http://tvunetworks.com/, 2007.

[33] G.D. Veciana, X. Yang, Fairness, incentives and performance in peer-to-peer networks, in: In the Forty-first Annual Allerton Conference on Communication, Control and Computing.

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

[35] C.J. Wu, C.Y. Li, J.M. Ho, Improving the Download Time of BitTorrent-like Systems, in: IEEE International Conference on Communications 2007 (ICC 2007), Glasgow, Scotland, pp. 1125–9.

[36] X. Yang, G. de Veciana, Service capacity of peer to peer networks, INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, volume 4, pp. 2242–2252 vol.4.

[37] D. Zeinalipour-Yatzi, T. Folias, A quantitative analysis of the gnutella network traffic, Technical Report, Department of Computer Science, University of California, Riverside, 2002.

[38] B.Q. Zhao, J.C.S. Lui, D.M. Chiu, Exploring the optimal chunk selection policy for data-driven p2p streaming systems, in: Peer-to-Peer Computing, pp. 271–80.

[39] Y. Zhou, D.M. Chiu, J. Lui, A Simple Model for Analyzing P2P Streaming Protocols, in: Proceeding of the IEEE International Conference on Network Protocols (ICNP'07), Beijing, China, pp. 226–35.