MOPSO-based Data Scheduling Scheme for P2P Streaming Systems

Pingshan Liu^{1,2}, Yaqing Fan^{2*}, Xiaoyi Xiong^{2} , Yimin Wen^{2} and Dianjie Lu^{3}
^{1} Business School, Guilin University of Electronic Technology
[e-mail: ps.liu@foxmail.com]
^{2} Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China
[e-mail: fan.yaq@foxmail.com]
^{3}School of Information Science and Engineering, Shandong Normal University, Shangdong, China
[e-mail: ludianjie@sina.com]
*Corresponding author: Yaqing Fan

*Received November 25, 2018; revised April 1, 2019; accepted April 14, 2019;
published October 31, 2019*

Abstract

In the Peer-to-Peer (P2P) streaming systems, peers randomly form a network overlay to share video resources with a data scheduling scheme. A data scheduling scheme can have a great impact on system performance, which should achieve two optimal objectives at the same time ideally. The two optimization objectives are to improve the perceived video quality and maximize the network throughput, respectively. Maximizing network throughput means improving the utilization of peer's upload bandwidth. However, maximizing network throughput will result in a reduction in the perceived video quality, and vice versa. Therefore, to achieve the above two objects simultaneously, we proposed a new data scheduling scheme based on multi-objective particle swarm optimization data scheduling scheme, called MOPSO-DS scheme. To design the MOPSO-DS scheme, we first formulated the data scheduling optimization problem as a multi-objective optimization problem. Then, a multi-objective particle swarm optimization algorithm is proposed by encoding the neighbors of peers as the position vector of the particles. Through extensive simulations, we demonstrated the MOPSO-DS scheme could improve the system performance effectively.

Keywords: Peer-to-Peer, video streaming, data scheduling, multi-objective optimization, multi-objective particle swarm optimization

A preliminary version of this paper was presented at the 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD 2017). This research was supported by the National Natural Science Foundation (No.61762029, No.U1811264, No.61866007), Guangxi Natural Science Foundation (No. 2016GXNSFAA380011), and Guangxi Key Laboratory of Trusted Software (No. kx201726).

1. Introduction

With the wider employment in industries, Peer-to-Peer (P2P) video streaming has developed rapidly [1, 2]. Generally, the researches on P2P video streaming systems are mainly divided into two modules. The first module is overlay network construction, which decides how the peers are organized into overlay network and select their neighbors to share streaming data [3]. The other module is data scheduling, which determines how the requested streaming data are streamed among the peers.

In recent years, data scheduling has attracted many researchers [4-12]. Shen et al.[11] aimed at optimizing the perceived video quality by scheduling the streaming segment transmission. And two approximation algorithms for data scheduling problem are proposed. Zhang et al.[12] first formulated the data scheduling problem as a classic min-cost network flow problem. Then, they proposed a global optimal data scheduling algorithm and distributed heuristic algorithm to improve the network throughput. When the network throughput is improved, the peers' upload bandwidth are efficiently used. Therefore, improving the perceived video quality and maximizing the network throughput are two important optimal objectives of the data scheduling problem.

However, sometimes the two optimal objectives will conflict. To improve the network throughput, the rarest video segment should be transimitted first. The previous study [13] shows that the "rarest first" strategy is very effective in data scheduling, which can improve the network throughput effectively. Empirically, to improve the perceived video quality, the "most emergent" video segment should be transimitted first. Usually, a rarest video segment is not a most emergent video segment. In this case, the two optimal objectives conflict. Thus, when one optimal objective is achieved, the other optimal objective is very likely to be missed. Therefore, maximizing network throughput will result in a reduction in the perceived video quality, and vice versa.

Furthermore, as far as we know, the previous studies failed to optimize the two optimal objectives of the data scheduling problem simultaneously. In the paper, we optimize the two optimal objectives at the same time. To this end, the data scheduling problem is transformed into a multi-objective optimization problem. In the problem, we use the sum of weights of each segment received by the receiver to represent perceived video quality. The network throughput is indicated by the utilization rate of upload bandwidth of senders. We put forward a multi-objective particle swarm optimization data scheduling algorithm to solve the multi-objective optimization problem. The algorithm encodes the senders of a receiver as the position vector of the particles. The simulation results show that the performance of our algorithm is superior to other algorithms.

The structure arrangement of the paper is as follows: We introduce the related work in section 2. Then we described the P2P steaming system and the data scheduling scheme model in section 3. In section 4, we propose our data scheduling algorithm and analyze its computational time complexity. In section 5, the performance of MOPSO-DS is evaluate by simulation. In section 6, we draws the conclusions and give the future work.

2. Related Work

The previous works [5, 11, 12] have shown that the optimal data scheduling problem is a NP-complete problem. In order to solve the NP-complete optimal data scheduling problem, the previous works have proposed heuristic algorithms with approximate optimal solutions [5-8, 10, 11, 12]. Some most related works are as follows. For live video streaming, the authors of [5] propose a data-driven overlay network. Then, they used a simple heuristic algorithm to solve the data scheduling problem and proved its effectiveness. Chakareski et al. [6] proposed an iterative descent algorithm to optimize the perceived video quality. Their proposed algorithm has been proved to be effective. The authors of [7] presented an integer linear programming (ILP) formulation to solve the data scheduling problem and aimed at improving the perceived video quality. Then, they proposed a polynomial-time approximation algorithm. The authors of [11] extended the work of [7], and to simplify the data scheduling model, they proposed a simpler approximation algorithm. Liu et al. [10] proposed an event-driven high-priority first data scheduling algorithm, called EHPF algorithm. The algorithm can solve the problem of highly dynamic peers in P2P VOD streams. The EHPF scheme considers both the response processing of the receivers and the response processing of the senders. In the EHPF scheme, an event-driven piggyback mechanism is designed and a new priority calculation strategy is designed. The simulation results showed that the EHPF algorithm can effectively optimize the perceived video quality. Bideh et al. [8] considered the influence of each frame type. The frame scheduling scheme on the receiving side and a sender selection strategy are proposed. However, frames in video are interrelated. Due to the loss of some frames, the frames can cause decoding errors when transmitted. Then, the perceived video quality will decrease. To improve perceived video quality, which may reduce sender's utilization of upload bandwidth. Zhang et al. [12] first equated the data scheduling problem with the minimum cost problem. Then, they proposed a distributed heuristic algorithm to resolve the minimum cost problem, aimming at optimizing network throughput. However, the overhead of the proposed algorithm is expensive. Huang et al. [14] proposed a load balance scheme. They used the request migration algorithm to solve the load balance problem. The algorithm can improve the network throughput effectively. Therefore, these works failed to consider improving the perceived video quality and maximizing the network throughput at the same time.

Several other works are related to P2P streaming in mobile Adhoc networks (MANET) [15] or P2P live streaming [16-18]. Hu et al. [15] used a delay-sensitive segment scheduling strategy to provide timely P2P streaming services in mobile Adhoc networks. In their algorithm, they proposed a rate control method. The method uses the playback-rate of the service to transmit each part of the P2P VOD streaming service evenly to MANET. Their algorithm effectively improves bandwidth utilization of a MANET. The authors of [16] proposed a QoS-awareness peer coordination control strategy. The peer coordination control mechanism includes a ring buffer mechanism implementing the cyclic coordination, a content similarity peer selection algorithm and the shortest distance peer selection algorithm, as well as peers task assignment algorithm. Pal et al.[17] proposed a new start-up-based selection mechanism and scheduling scheme based on slack time. In their scheme, the start-up buffer location of the new peer is defined based on the start-up-based selection process, and the block and peer are selected simultaneously in the scheduling scheme. The authors of [18] proposed a Priority-based Scheduling Scheme (PrSS). The PrSS considers four factors at the same time, including a deadline, rarity and layer number for chunk selection as will as variable size chunk selection strategy. Their proposed scheme uses the crucial Chunk Selection Procedure (CSP) and Peer Selection Procedure (PSP) as the key parts of the algorithm. The PrSS considers two independent CSPs and a Bandwidth-Aware Peer Selection (BAPS). These works did not consider to design an optimal data scheduling scheme based on a scheduling model.

In the paper, the data scheduling problem is formulated as a multi-objective optimization problem. We propose a new multi-objective particle swarm optimization data scheduling algorithm to resolve this problem, called MOPSO-DS algorithm, which can simultaneously achieve high perceived video quality and high network throughput.

3. Description of Data Scheduling Problem

In the chapter, we introduce the P2P streaming system model and data scheduling problem in detail. For convenience, we listed the symbols in **Table 1**, which are used in this paper.

3.1 System Model

The architectures of the P2P systems can be classified into two classification: structured and unstructured. The main structured approach is tree-based, which including single tree structure and multiple tree structure. In this approach, a parent peer pushes video sources along the tree path to its child peers according to a predetermined schedule. Therefore, the overlay network of our proposed data scheduling scheme is a tree-based overlay network structure. However, this approach cannot adapt to the high dynamics of the large-scale network. In the paper, we focuses on the data scheduling algorithm in the mesh-based overlay network model, which has robustness , scalability and simplicity.

Table 1. List of symbols used in the paper

Symbol	Description	Symbol	Description
M	number of each receiver's neighbors	d_j	playback deadline of segment j
Q	the swarm in which the peers share a video	$P^j_{emergency}$	emergency of segment j
v_f	frame rate of the video	$t_{playback}$	current playback time
N_{gop}	number of frames in each segment	$PSNR_j$	quality of segment j
N	number of segments in each video	w_j	weight of segment j
i	serial number of sender	r_q	ratio of the quality property
j	serial number of segment	r_e	ratio of the emergency property
b_i	upload bandwidth of the sender i	$N^j_{neighborCount}$	number of neighbors who holds segment j
$a_{i,j}$	availability of segment j on the sender i	x_{ij}	variable for sender $\boldsymbol{i}$ to transmit segment j
δ	length of the sliding window	s_j	size of segment j
S	the wanted segment set of a receiver	τ	length of scheduling window
k	serial number of request peer	N_i	the set of receivers of sender i

Fig. 1 is the general system model diagram**.** The system is consisted of a tracker, a streaming source server and many peers. As shown in **Fig. 1**, the streaming source server divides a video into segments by the H.264 coding standard [19]. Each video segment has a serial number for identification, and consists of video frames. The frame size varies with the coding scheme. Therefore, the segment size is varied too.

When a peer starts playing a video, it first contacts the tracker, and the tracker provides some neighbor peers which are playing the same video. Then, the peer joins the swarm formed by its neighbor peers. The swarm is dynamic, so the peers can join or leave the swarm at any time. When a peer is ready to join a swarm, the peer first chooses its neighbors according to its playback point. Then, the peer can periodically request video segments from its neighbors and cache the received video segments. Following, the peer can contribute its upload bandwidth to share its cached video segment. Usually, the peers in a swarm have different bandwidth, and cache different video segments. In order to achieve message communication, a peer in a swarm exchanges its upload bandwidth information and buffer mapping with its neighbor peers periodically. The buffer map information is encoded as a binary string to represent the availability of the video segment. For brevity, the neighbor peer that provides video segments to some other peers is called a sender, and the peer that requests some video segments is called a receiver.

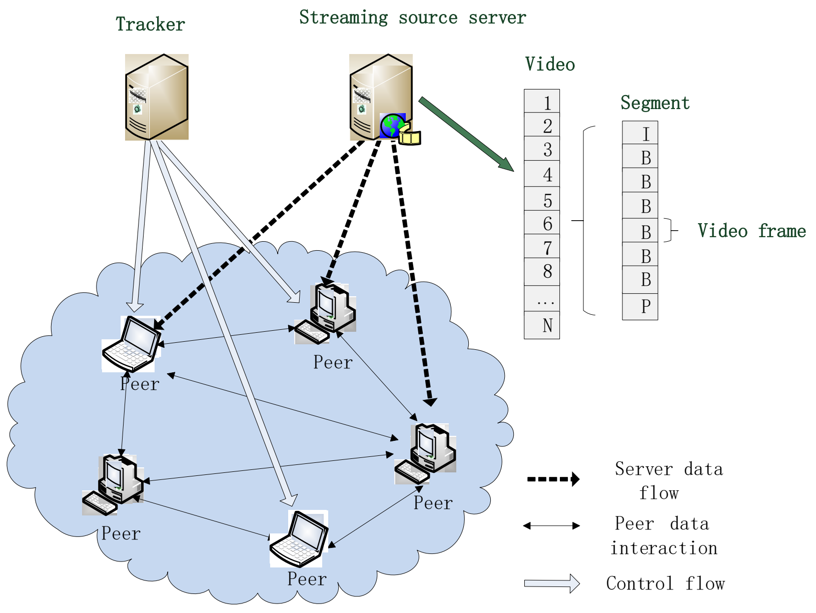


Fig. 1. System model

3.2 Data Scheduling Problem Statement

The goal of the paper is to compute an approximately optimal data scheduling to imporve the perceived video quality and network throughput. In this part, the data scheduling problem is introduced.

The Data scheduling problem: In our proposed P2P streaming system, each peer with M neighbors and they are regarded as a swarm Q, in this swarm they share a video segment with each other. We let v_f denote the video's frame rate. A video consists of N segments, and each video segment is given a unique number j ($1 \leq j \leq N$). Because the content of each segment is different after encoding, the segment quality of each segment is different. Let b_i represent the upload bandwidth of peer i. Let a_{ij} represent whether segment j is available for a peer i. a_{ij}=1 indicates that the sender i ($1 \leq i \leq M$) has a available segment j, and a_{ij}=0 otherwise. b_i and a_{ij} of a peer in the swarm are exchanged periodically. When a peer starts watching the video, its missing segments are kept in a sliding window. The length of the sliding window is δ seconds. We let S denote the set of the serial numbers of the missing segments. According to the current playing time of the segment, we can calculate the decoding deadline of each segment in the sliding window. The formula of decoding deadline is as follows:

$$d_j = \frac{(j-1) \times N_{gop}}{v_f} \tag{1}$$

4. Solution of the Multi-Objective Optimizing Problem

In the chapter, we analyze the data scheduling problem in detail and use an efficient MOPSO-DS algorithm to solve the problem.

The perceived video quality and the network throughput will be affected by the segments of a video, and different segments will bring different impacts. The previous study[12] shows that a rarer segment should be transmitted first to improve the network throughput. However, to improve the perceived video quality, a more urgent segment should be transimitted first. A rare segment may not be an urgent segment. For example, a segment j is not a rare segment, but it is an urgent segment. If the segment j is not transmitted first in the next scheduling period, the user will suffer a playback disrupt. To gurantee the perceived video quality, the segment j is transimitted. But the network throughput will not be improved. Therefore, optimizing perceived video quality may conflict with optimizing network throughput. To achieve the two optimal objectives, in our algorithm, data scheduling problem is considered as a multi-objective optimization problem. Then, we use a multi-objective partical swarm optimization data scheduling algorithm to solve the problem.

4.1 The formulation for optimizing the perceived video quality

The perceived video quality is affected by video segments, and different video segments have different effects. To optimize the perceived video quality, the total weight of the segments requested by the receiver should be maximized in each data scheduling period. We design a calculation strategy for segment weight, which is controlled by two characteristics of each segment: emergency and segment quality.

If a segment can be sent to a receiver, it must satisfy the following conditions: 1) The segment must be owned by at least one sender. 2) The segment should be sent before the decoding deadline of the receiver. First, we should focus on the emergency of a segment. When a video is played by peers, the segments with shorter decoding time and shorter playback intervals are easier to lose. So data scheduling algorithms should give priority to these segments. Therefore, the formula of emergency of segment j is defined as (2):

$$P^j_{emergency} = \frac{\delta - (d_j - t_{playback})}{\delta} \tag{2}$$

In (2), d_j is playback deadline of segment j, $t_{playback}$ stands for the current playback time, δ represents the size of a sliding window, i.e. the time of a sliding window. Then, d_j-$t_{playback}$ is the available remaining time of segment j, between the current playback time and the playback deadline.

Another attribute we consider is segment quality. The segment quality is defined by the peak signal-to-noise ratio (PSNR) [20]. Various video frames are formed into a video segment, and each video frame has different PSNR value. The segment quality is represented by the average PSNR of frames. The average PSNR value of a segment is also used in [11]. Therefore, we use the following formula to measure segment quality:

$$P_{quality}^{j} = \frac{(PSNR_j - PSNR_{min})}{(PSNR_{max} - PSNR_{min})} \tag{3}$$

In (3), $PSNR_j$ presents the quality of the segment j. $PSNR_{min}$ means the minimum quality of all video segments. $PSNR_{max}$ means the maximum quality of all video segments.

Therefore, we summarize the two properties in formula (2) and (3) as the weight formula of segment j, which is as follows:

$$w_j = r_e \cdot p_{emergency}^{j} + r_q \cdot p_{quality}^{j} \tag{4}$$

In (4), r_e is the ratio of the emergency and r_q is the ratio of the segment quality , in addition $r_e + r_q = 1$. Then, we discuss the values of r_e and r_q in the following two scenarios.

1) $N^{j}_{neighborCount}=0$.

$N^{j}_{neighborCount}$ is used to denote the number of neighbors with segment j. Network failure or other personal reasons may lead to peers leave the system at any time. This situation may result in no neighbor owning the requested segment. In this case, it is useless to compute the values of weight function, because the senders cannot send the segments they do not have. Therefore, the weight of segment j is set to 0, the formula is:

$$w_j = 0 \tag{5}$$

2) $N^{j}_{neighborCount} > 0$.

According to our simulation which is shown in Section 5.2, we set r_e to 0.7 and r_q to 0.3. The proportion of segment emergency is high, its priority is high. Therefore, in order to maintain fluency, when approaching the current playback time, the segment will play first. Due to video encoding errors, some segments with lower *PSNR* value may appear. It will reduce the quality of segment quality. Therefore, we should reduce the weight of the segments with poor quality. In this part, we give the formula:

$$w_j = 0.7 \cdot p_{emergency}^{j} + 0.3 \cdot p_{quality}^{j} \tag{6}$$

Finally, the perceived video quality optimization formula is formulated as the following formula:

$$Maximize: F_1 = \sum_{i \in Q} \sum_{j \in S} w_j x_{ij} \tag{7}$$

Where x_{ij} denotes that sender i scheduling segment j to the receiver.

4.2 The formulation for optimizing the network throughput

In the part, we discuss the difference in bandwidth utilization between senders in each scheduling period. The higher the upload bandwidth utilization of senders, the higher the network throughput. In order to improve network throughput, we define the following formulas:

$$Maximize: F_2 = \sum_{i \in Q} R_i \tag{8}$$

In (8), R_i means the upload bandwidth utilization of sender i. The formula for calculating R_i is as follows:

$$R_i = \sum_{j \in S} \sum_{k \in N_i} \frac{s_j^k x_{ij}}{b_i \tau} \tag{9}$$

In formula (9), N_i denotes the set of receiver of sender i; s_j^k denotes the receiver k requets segment j; x_{ij} shows that sender i scheduling segment j to the receiver; b_i denotes the upload bandwidth; τ denotes the length of scheduling window.

Then, we can get a summarized optimization formula (10):

$$Maximize: F_2 = \sum_{i \in Q} \sum_{j \in S} \sum_{k \in N_i} \frac{s_j^k x_{ij}}{b_i \tau} \tag{10}$$

4.3 The multi-objective optimization formulation

In the part, we use the multi-objective optimization problem to represent the data scheduling problem, according to the two optimization formula (7) and (10). The set of formula is defined as follows:

$$Maximize: F_1 = \sum_{i \in Q} \sum_{j \in S} w_j x_{ij}$$

$$Maximize: F_2 = \sum_{i \in Q} \sum_{j \in S} \sum_{k \in N_i} \frac{s_j^k x_{ij}}{b_i \tau} \tag{11}$$

$$\sum_{i \in Q} x_{ij} \le 1, \forall j \in S \tag{a}$$

$$\sum_{j \in S} \sum_{k \in N_i} x_{ij} s_j^k \le b_i \tau, \forall i \in Q \tag{b}$$

$$s_j^k \in \{0,1\}, \forall j \in S, k \in N_i \tag{c}$$

$$x_{ij} \le a_{ij}, \forall i \in Q, j \in S \tag{d}$$

$$x_{ij} \in \{0,1\}, \forall i \in Q, j \in S \tag{e}$$

In the set of formula (11), the objective function F_1 is used to calculate the maximum total weight of all on-time video segments. The function F_2 is used to calculate the maximize sum of the upload bandwidth utilizations of all the senders. Constraint (a) ensures that each segment can be sent by at most one sender. M denotes the number of each receiver's neighbors. Constraint (b) guarantees that each sender doesn't overload. We let s_j^k denote that the receiver k requests segment j. We let τ denotes the request period. Constraint (c) shows that the receiver k can request several segments from the sender i in a period. s_j^k=1 means the receiver k requests the segment j from the sender i and s_j^k=0 otherwise. Following, constraint (d) means a sender only sends its cached segments. a_{ij}=1 means that there is a segment j on the sender i, otherwise a_{ij}=0. Finally, the value range of variable x_{ij} is restricted by constraint (e).

4.4 The multi-objective particle swarm optimization algorithm

The particle swarm optimization(PSO) algorithm was proposed by Kennedy et al. [21, 22] in 1995. It is an algorithm simulating the natural behavior of bird foraging. By observing the behavior of particle swarm, the problem can be solved by sharing information among groups from disorder to order. PSO was initially used to solve single objective optimization problem. With the expansion and deepening of research on PSO, researchers have proposed a multi-objective particle swarm optimization (MOPSO) algorithm. It can achieve better performance in solving multiple target problems [23].

The formula for calculating the velocity of each particle is as follows:

$$v_{i+1} = wv_i + c_1r_1(pbest_i - x_i) + c_2r_2(gbest_i - x_i) \tag{12}$$

In formula (12), we let w denote inertia weight, which can avoid particles entering local optimization setting. r_1 and r_2 are random decimal number with values ranging between 0 and 1. Let *Pbest* represent the local best location, that is to say, particles are at their best in their history. Let *Gbest* represent global best location, that is to say, particles are at their best in the whole particle swarm. c_1 and c_2 are learning factors, which are two constants to measure the weight of *Pbest* and *Gbest*.

In the data scheduling period of our proposed algorithm, a receiver has many neighbors who hold the requested segment. In each data scheduling period, each segment selects a sender from the receiver's neighbors. In the paper, the serial numbers of the neighbors is transformed into the decision variables of the particle swarm. In our algorithm, the solution set of the classical PSO algorithm is transformed from continuous to discrete. The equation of position is as follows:

$$x_{i+1} = \lfloor x_i + v_i \rfloor \tag{13}$$

In (13), for a particle in the *i*th dimension, the *x* means that the *i*th segment is sent by the *x*th neighbor.

In our algorithm, the particles search for the best position in a certain boundary space. When a particle flies out of the boundary, it automatically returns to the space and then continues to fly.

Algorithm 1 The Proposed Data Scheduling Algorithm (MOPSO-DS)

```
1   Procedure
2   Calculate all $w_j$ and $NeighborCount_j$ for each segment in the sliding window
3   Add the neighbor peers with the segment $j$ to a set $neighbor_j$
    //Initialise the parameters in the particle swarm
4   For each $particle_k$ in the particle swarm do
5       For each dimension $j$ in the $particle_k$ do
6           Set the location range: $0<location_j<NeighborCount_j$
7           $velocity_j=0$
8           $location_j$=random positional coordinate value in the $boundary_j$
9       End for
10      evaluate $particle_k$
```

```
11      $P_{bestk}$=$particle_k$
12  End for
13  Store the nondominated particles in the external archive EA and initialise $t_{iteration}$=0
14  Calculate the values of crowding distance $d_{crowd}$ for each particle in EA
      // $d_{crowd}$ shows the density of each location in pareto front
15  Listed the particles in EA in descending order by $d_{crowd}$ values
16  Repeat
17    For each $particle_k$ in the particle swarm do
18      $Gbest_k$=$particle_x$ ($particle_x$ represents a random particle in EA, and x represents a random
        number under tenth of $S_{EA}$)
19      Recalculate the velocity of the $particle_k$ according to the formula (12)
20      Recalculate the location of the $particle_k$ according to the formula (13)
21      Appraise $particle_k$
22      Update $Pbest_k$
23    End for
24    Store the nondominated particles to EA and delete the dominated particles from the EA.
25    Calculate the value of crowding distance $d_{crowd}$ for each particle in EA and list $d_{crowd}$ in
      descending order
26    While $S_{EA}$ > maxArchiveSize do
27      Delete the particle with the biggest $d_{crowd}$ from EA
28    End while
29  Until $t_{iteration}$ =maxIterationTime
30  Select a optimum particle from EA, the location vector of the particle is the solution vector
    of the problem
31  For each segment j in the sliding window do
32    the selected sender =$neighbor_j[location_j]$
33    usedTime =sender.usedBandwidth/sender.uploadBandwidth
34    $transmissionTime_j$ =usedTime+$s_j$/sender.uploadBandwidth
35    If $transmissionTime_j \le d_j$ do
36      Add <i,j,usedTime> to the schedule of the algorithm
37      Recalculate usedBandwidth of the sender
38    End if
39  End for
40  End procedure
```

Algorithm 1 lists the specific steps of our proposed algorithm. There are seven steps in *Algorithm 1*. First, all the weights of the segments in the sliding window of the receiver is calculated by the algorithm in line 1, and then adds the neighbors with the segment *j* to a collection *neighbor$_j$* in line 2. Second, the algorithm initialises the particle swarm with a loop from line 4 to line 12. Third, the algorithm copies all nondominated particles into the external archive *EA* in line 13. A particle is nondominated, which means that the particle improves an optimal objective and does not let down other optimal objectives. The fourth step is from line 14 to line 15. In this step, the algorithm calculates the crowding distance(d_{crowd}) of all the particles in *EA*, and sorts them in descending order according to d_{crowd}. The fifth step includes two loops, which is from line 16 to line 29. The first loop is to update the optimum for each particle, which is from line 17 to line 23. According to the updating value of each particle, the *EA* is updated in line 24. Then, in line 25, the crowding distance of the particles in *EA* is calculated. According to the updating crowding distance, the algorithm sort the particles in descending order. The second loop is to remove the particles with the biggest d_{crowd} one by one until the size of *EA* is smaller than the maximum size of *EA*, which is from line 26 to line 28. The above process from line 18 to line 28 is iterated to the maximum iteration times. The sixth step selects a particle from *EA* as the solution, which is in line 30. The specific selection scheme is explained in detail at the end of this section. The last step determines the scheduling scheme with a loop, which is from line 31 to line 39. In this step, the usage time of the senders transmitting segment *j* and the transmission time of segment *j* are calculated first. Then, a scheduling descision is made, and related information is updated.

In this algorithm, the nondominated solutions are stored in an external archive. With the operation of the algorithm, the external archive stores more and more nondominated solutions constantly. To maintain the diversity and size of external archive, the algorithm removes redundant nondominated solutions from *EA*. Therefore, the crowding distance of particles in *EA* is defined, which is the distance between the particle and its adjacent two particles in *EA*. Then, the algorithm deletes the nondominated solutions with the biggest crowding distance value, to maintain the diversity of *EA*. The crowding distance is calculated according to *Algorithm 2*.

Algorithm 2 Crowding Distance Computation

```
1   List the particles in descending by $fitness_1$ values
2   List the particles in ascending by $fitness_2$ values
3   Let the value of $d_{crowd}$ of the first and last particle be infinite
4   For each particle in the EA (except the first one and the last one) do
5     $distance_1$ =Euclidean distance between the particle and its last one
6     $distance_2$ =Euclidean distance between the particle and its next one
7     $d_{crowd}=distance_1+distance_2$
8   End for
```

The performance of the algorithm is influenced by two important modules. One module is fitness evaluation and constraint handling. The search performance of the particle swarm is affected by this module. The other module is solution selection, which chooses a solution from *EA* after the MOPSO algorithm completes the iteration.

1) Fitness Evaluation and Constraint Handling

In this part, the fitness evaluation and the constraint handling is described in detail. First, the *usedBandwidth* of each sender is initialised to 0. Second, the position information of all particles is iterated. Fitness evaluation has two constraints. One constraint is the transmission time constraint, which ensures that the requested segments can arrive at the receivers before the playback deadline. The other constraint is the upload bandwidth constraint, which guarantees that each sender doesn't overload. In the process of evaluating a particle, only when the solution satisfies both constraints, the $fitness_1$ will add to the weight of the segment.

The $fitness_2$ is the sum of the sender's upload bandwidth utilization. *Algorithm 3* shows the pseudo code of this module.

Algorithm 3 Fitness Evaluation And Constraint Handling

```
1   Set the usedBandwidth to 0 for each sender
2   Set the fitness value to 0 for each objective
3   For each dimension $j$ of the location of the particle do
4     The selected sender=$neighbor_j[location_j]$
5     $sender.usedBandwidth+=s_j$
6     $transmissionTime_j=(sender.usedBandwidth+s_j)/sender.uploadBandwidth$
7     Boolean $feasible_1=transmissionTime_j \le d_j$
8     Boolean $feasible_2=sender.usedBandwidth \le sender.uploadBandwidth * \tau$
9     If (!$feasible_1$ || !$feasible_2$) do
10        $sender.usedBandwidth -= s_j$
11        continue
12    End if
13    $fitness_1 += w_j$
14  End for
15  $fitness_2$= the sum of utilization of the each sender's upload capacity
```

2) Solution Selection

The solution selection module selects a particle in *EA* as a solution to determine the data scheduling algorithm. The goal of solution selection is to find an optimal scheduling scheme, which can optimize both the perceived video quality and network throughput. These two objectives are considered in *Algorithm 1*, which can be nearly achieved by the MOPSO algorithm. The particles in *EA* contain two indications denoted the two objectives. In order to improve QoE (Quality of Experience), the perceived video quality should be considered first. Therefore, the particle in *EA* with the maximum perceived video quality should be chosen preferentially.

4.5 The computational complexity of the algorithm

The time complexity of the proposed algorithm (MOPSO-DS) is $O(NM + \lambda NK + \lambda MK + \lambda K\log K)$, as is shown in *Algorithm 1*. We let N denote the number of segments in a video, let M denote the number of neighbors of a receiver, let K denote the number of particle in the particle swarm, let λ denote the iteration times for finding optimal solutions in particle swarm.

Proof : In *Algorithm 1*, The first step is in line 2, which takes time $O(N)$. The second step is in line 3, which takes time $O(NM)$. The third step is from line 4 to line 12, which initializes all the particles in the particle swarm. It takes time $O(NK)$. The fourth step is from line 13 to line 15. The time complexity of line 13 is $O(1)$, because the size of *EA* is 30. The time complexity of line 14 and line 15 is $O(K\log K)$, due to the particles in *EA* have been sorted twice according to the crowding distance in *Algorithm 2*. Therefore, the fourth step takes time $O(K\log K)$. The fifth step is from line 16 to line 29, which iterates all the particles in the particles swarm. This step is divided into four parts: part a, part b, part c, and part 4.

The part a is from line 17 to line 23, which iterates all the *N* segments and *M* neighbors in the particle swarm. Thus, the part a takes time O(NK+MK), for there are *K* particles in the particle swarm. The part b is in line 24, which takes time *O(K)*. The nondominated particles are copied to *EA* and the dominated particles are deleted from the *EA*. The part c is in line 25, which takes time *O(KlogK)*, which computes crowding distance. The part d is from line 26 to line 28, which takes time O(K), for the number of increased particles in *EA* in an iteration is not bigger than *K*. Furthermore, the number of iterations for finding optimal solutions is λ in the fifth step. Thus, the time complexity of the fifth step is $O(\lambda NK+\lambda MK+\lambda KlogK)$ in sum. The last step is from line 30 to line 39, which calculates the transmission time schedule for all the segments in a period. The last step takes time *O(N)*. So the time complexity of the proposed algorithm is $O(NM+\lambda NK+\lambda MK+\lambda KlogK)$ in sum.

5. Evaluation

In the chapter, we demonstrate the performance of MOPSO-DS algorithm through simulation. First, we introduce the setup of simulation. Then, we introduce the metrics of the simulation. Finally, the MOPSO-DS algorithm is compared with other classical data scheduling algorithms.

5.1 Simulation Setup

In simulation, the simulator we use is an event-driven simulator proposed by Shen [11]. The simulator is coded in Java, and the overlay network structure of simulation platform is a mesh-based overlay network structure. The simulation program runs the five algorithms mentioned in this paper. The five algorithms are SSTF(Serialized shortest transmission-time first) and WSS(Weighted segment scheduling) in [11], LRF(Local rarest first) in [5], Min-cost in [12], and the proposed MOPSO-DS. SSTF implements an non-weighted data scheduling algorithm, which gives priority to the minimum transmission time. WSS is an algorithm that focuses on segment quality. LRF is one of the most popular data scheduling algorithms widely used in PPTV[1], CoolStreaming [5], and other practical systems. LRF schedules peers with the largest remaining bandwidth and chooses peers with larger bandwidth as senders. Finally, the min-cost method is a heuristic algorithm, which solves the data scheduling problem as a minimum cost flow model.

In the simulator, a tracker, a streaming source server and 2000 peers are deployed. The simulator selects 1% of all peers as seeds, which are used to propagate video segments in the beginning. **Table 2** shows the distribution of upload bandwidths of peers, which is used in [24]. The upload bandwidth capacity of the tracker is set to 10Mb, and the download bandwidth is the same as the upload bandwidth. We set the number of the neighbors to 10. The simulation time for each algorithm is 24 hours. In order to simulate the dynamic real system, each peer joins the system randomly, and some of the peers leave randomly in the simulation process. The peers exchange their buffer mapping information, which is used to indicate whether their segments are available. The available upload bandwidth of peers is updated periodically. When a peer starts to play a video, it holds a sliding window containing its missing segments. In the simulation, the time of sliding window is set to 10 seconds. For the parameters of MOPSO-DS, we use the same parameters as literature [26]. The specific parameter settings are as follows. The inertia weight, the c_1 and the c_2 are set to be 0.8, 0.3, and 0.7 respectively. In the simulation, the size of particle swarm is set to 100, and we set the size of the external archive to 30. In order to obtain the optimal scheduling scheme, particle swarm optimization iterates 300 times in each scheduling period. Then, the segments is sent by the senders to receivers according to the schedule.

Table 2. Peer upload bandwidth distribution

Distribution (%)	10	14.3	8.6	12.5	2.2	1.4	6.6	28.1	16.3
Total Bandwidth	256	320	384	448	512	640	768	1024	>1500
Contributed Bandwidth	150	250	300	350	400	500	600	800	1000

The simulator runned on a normal personal computer.The personal computer with a 2.3-GHz Intel CPU and 8GB memory. We use two different high-resolution video for simulation, which from the Arizona State University video trace library [20, 25]. Table 3 shows the parameters of the video traces.

Table 3. The parameter of the video traces

Video Name	From Mars To China	Fugitive
Resolution	HDTV(1920*1080)	HDTV(1920*1080)
Frame Rate(fps)	30	24
Number of Frames	51715	86368
Group of Pictures(GOP)	12	16
Quantization Parameter(QP)	28-28-30	24, N/A, 32
Frame Size(bits):min/max/mean	80/326905/20207.12	81/235891/12970
PSNR Value (dB): min/max/mean	43.314/68.458/47.94675	35.194/76.171/43.560
Mean Bitrate(kbps)	4849.71	2490.24

Referring to the metrics used in references [5] and [11], we use four key performance metrics. They are listed as follows.

(1) Average perceived video quality

The first metric is the average perceived video quality, which is calculated as follows.

$$\alpha = \sum_{n=1}^{N} \frac{q_n u_n}{N} \tag{14}$$

Where the PSNR value of the segment n is expressed as q_n. We let u_n denotes whether segment n will arrive on time. The high value of the PSNR means the better the perceived video quality.

(2) Continuity index

The second metric is the continuity index, which is calculated as follows.

$$\beta = \frac{\sum_{n=1}^{N} u_n s_n}{\sum_{n=1}^{N} s_n} \tag{15}$$

The continuity index is the total size of on-time segments among all segments. It is used to indicate the fluency of the watched videos. The high value of the continuity index means the better the viewing experience.

(3) Load balancing factor

The third metric is the load balancing factor. To calculate the load balancing factor of a sender, the load of sender should be computed first, which is calculated as follows.

$$l_i = \sum_{j \in Schedule_i} \frac{s_j}{\delta b_i} \tag{16}$$

The load of the sender i is measured by the utilization of upload bandwidth of the sender i. In (16),We use $\sum_{j \in Schedule_i} s_j$ to represent the total size of the segments of the sender i in a scheduled period.

Then, we can compute load balancing factor as follows.

$$\gamma = \sqrt{\frac{\sum_{p_i=1}^{P_i} \left(l_{ip_i} - \overline{l} \right)^2}{p_i}} \tag{17}$$

We use the standard deviation of the sender’s load during all scheduling periods to express the load balancing factor r. Load balancing performance is better when the value of r is small. In (17), the number of scheduling periods of sender i is represented as P_i in the simulation.

(4) Execution time

The forth metric is the execution time of a data scheduling algorithm. The shorter the execution time of a data scheduling algorithm, the better the performance of the data scheduling algorithm.

5.2 Simulation Results

First, we run our algorithm on the simulator and find the optimum parameters of the proposed algorithm. Then we run each algorithm on the simulator, and record the performance metrics mentioned above for each algorithm. In order to clearly demonstrate the performance of the five algorithms mentioned above, we calculated and plotted the cumulative distribution function (CDF) curves of each algorithm's performance. Finally, we give a general overview of the simulation results. For convenience, the MOPSO-DS is denoted by MOPSO in this part.

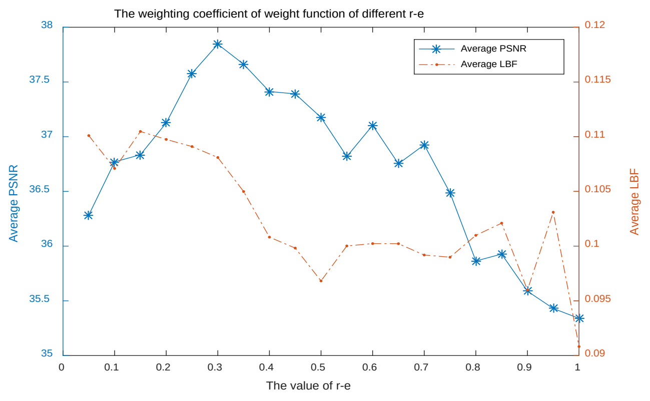


Fig. 2. The weighting coefficient of weight function of different r_e

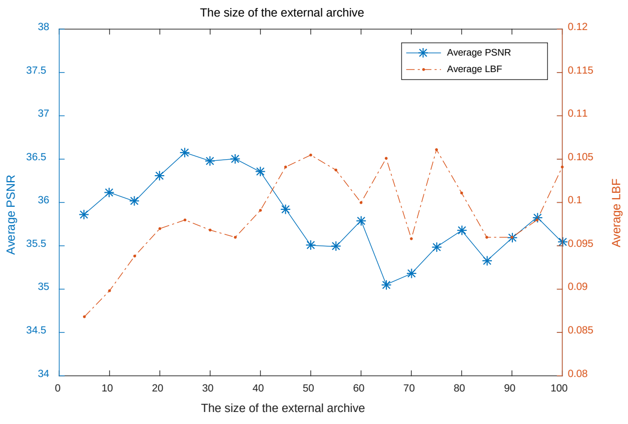


Fig. 3. The size of the external archive

First, we tested the average value of PSNR and LBF, which were generated by different weight values of weight function. As shown in **Fig. 2**, when r_e =0.3, the average value of PSNR of received video is the largest, and the LBF value is relatively stable and small. The comprehensive performance of the algorithm is optimal when r_e =0.3. At the same time, **Fig. 3** shows that the overall performance of the algorithm is better when the size of external archive is 30. Therefore, r_e and r_q are set to 0.3 and 0.7 respectively. Then we set the size of external archive to 30.

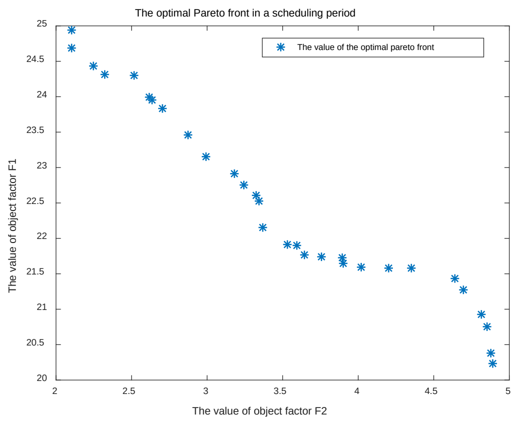
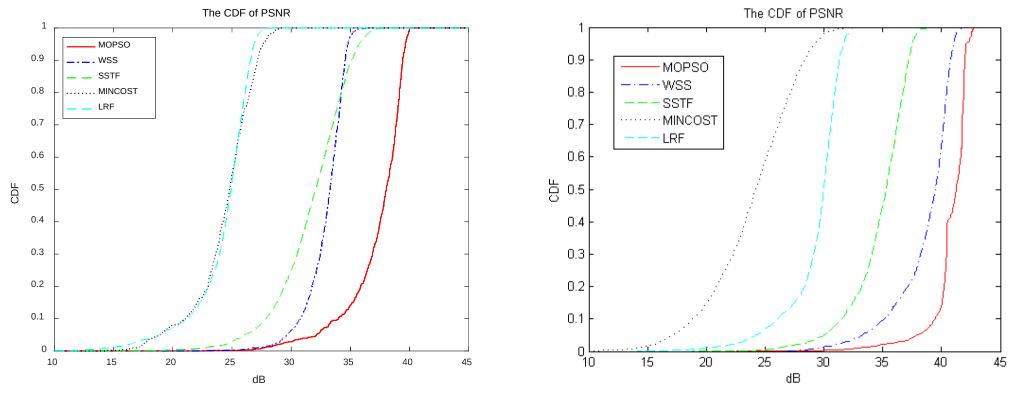


Fig. 4. The optimal pareto front in a scheduling period

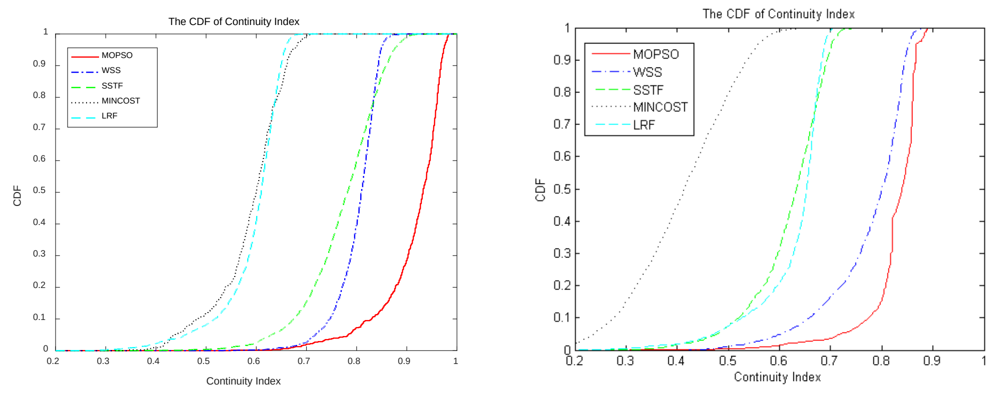
Fig. 4 shows the distribution of noninferior optimal solutions of objective function F_1 (maximize perceived video quality) and objective function F_2 (maximize network throughput) in a scheduling period. **Fig. 4** shows that each solution is independent at the Pareto front.



(a) From Mars To China (b) Fugitive

Fig. 5. CDF of PSNR.

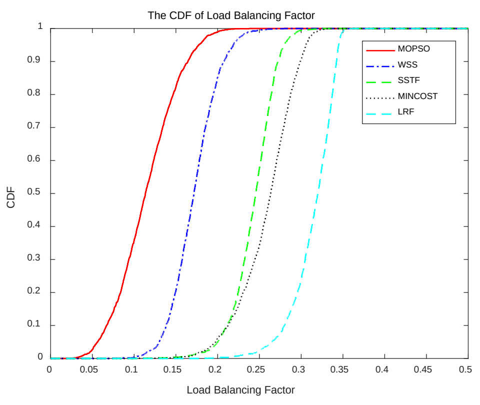
Fig. 5 shows the CDF curves of PSNR value**.** The two pictures represent different video traces with different video characteristics. **Fig. 5** clearly shows that the performance of MOPSO algorithm is significantly better than the other four algorithms. In **Fig. 5(a)**, around 80% of peers in the overlay network with MOPSO algorithm have a PSNR value between 36 dB and 40 dB, while around 80% of peers in the overlay network with WSS algorithm, SSTF algorithm, MINCOST algorithm and LRF algorithm have a PSNR value between 33 dB and 35 dB, between 29 dB and 35 dB, between 23 dB and 26 dB, between 23 dB and 28 dB, respectively. In **Fig. 5(b)**, around 90% of peers in the overlay network with MOPSO algorithm have a PSNR value between 40 dB and 42 dB, while around 90% of peers in the overlay network with WSS algorithm, SSTF algorithm, MINCOST algorithm and LRF algorithm have a PSNR value between 35 dB and 40.5 dB, between 31 dB and 37 dB, between 26 dB and 31 dB, and between 19 dB and 30 dB. Therefore, most of peers in the system with MOPSO algorithm can obtain a higher perceived video quality. Furthermore, most of peers in the overlay network with SSTF algorithm can get an acceptable quality close to WSS, while most of peers in the system with the other two algorithms MINCOST and LRF cannot get acceptable perceived video quality.

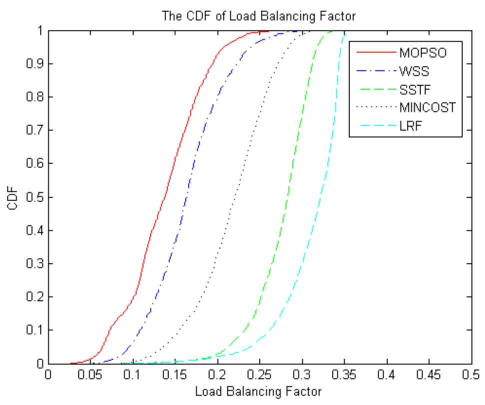


(a) From Mars To China (b) Fugitive

Fig. 6. CDF of continuity index.

The continuity index of each algorithm is shown in **Fig. 6**. As is shown in **Fig. 6**, MOPSO can achieve the best performance among the five algorithms obviously. In **Fig. 6(a)**, we can observe that around 70% of the peers in the overlay network with MOPSO algorithm can have a continuity index over 0.9, while around 70% of the peers in the overlay network with WSS algorithm, SSTF algorithm, MINCOST algorithm and LRF algorithm can have a continuity index between 0.79 and 0.89, between 0.74 and 0.9, between 0.59 and 0.68, and between 0.57 and 0.7 respectively. In **Fig. 6(b)**, we can observe that around 80% of the peers in the overlay network with MOPSO algorithm can have a continuity index over 0.8, while around 80% of the peers in the overlay network with WSS algorithm, SSTF algorithm, MINCOST algorithm and LRF algorithm can have a continuity index between 0.71 and 0.87, between 0.59 and 0.7, between 0.57 and 0.71, and between 0.32 and 0.61 respectively.





(a) From Mars To China (b) Fugitive

Fig. 7. CDF of load balancing factor

Fig. 7 shows the CDF curves of the load balancing factor. MOPSO performance is better than the other four algorithms in this performance metrics. Furthermore, compared with SSTF and MINCOST, LRF has higher load balancing factor. In **Fig. 7 (a)**, with MOPSO algorithm, the maximum load balancing factor value is only 0.18, and more than 90% of peers have a load balancing factor value below 0.15. In **Fig. 7 (b)**, with MOPSO algorithm, the maximum load balancing factor value is only 0.21, and more than 90% of peers have a load balancing factor value below 0.18.

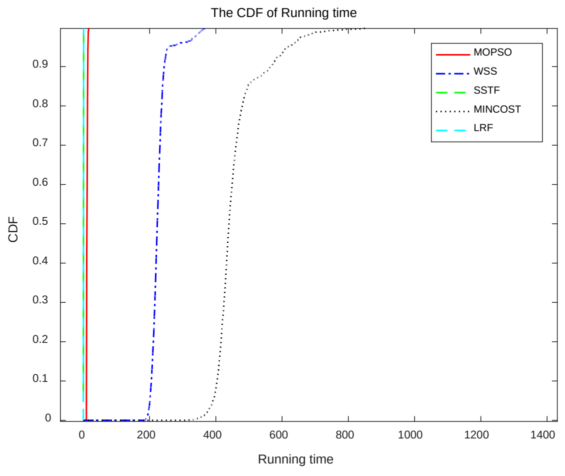


Fig. 8. CDF of execution time

We can see that the CDF curves of the execution time of the algorithms in **Fig. 8**. The execution time of MOPSO, LRF and STF are very small, while WSS and MINCOST cost a longer time. Therefore, due to the low overhead of MOPSO, it can widely used in most real-time P2P video streaming systems.

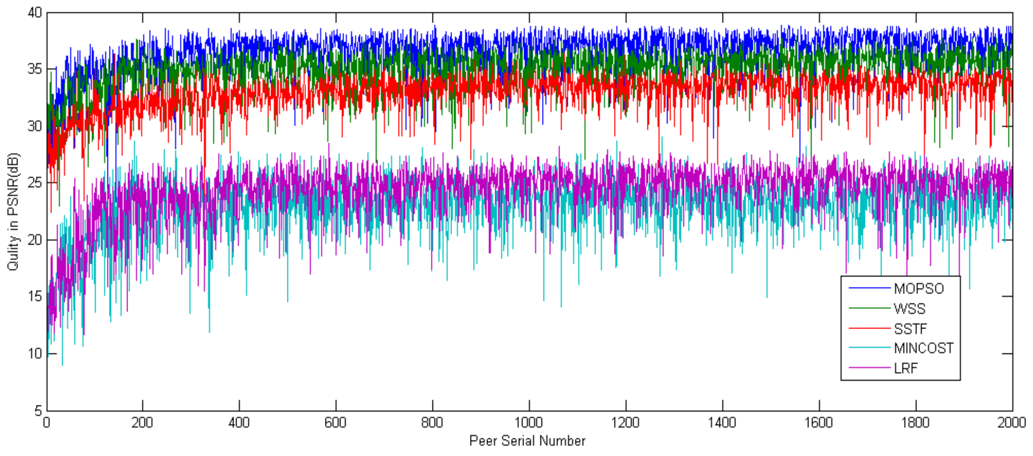


Fig. 9. Overall peer quality comparison among different algorithms. Sample results from From Mars To China

Finally, according to the order in which peers join the network, the quality map of each peer is drawn in **Fig. 9**. From **Fig. 9**, we can observe that the performance of MOPSO is obviously superior to other algorithms. Furthermore, MOPSO and WSS can maintain a high level of quality during the whole simulation process. Besides, the SSTF also can achieve an acceptable quality. The other two algorithms (MINCOST and LRF) perform poorly on the quality metric, and the highest quality value they can achieve is less than 30 dB. From **Fig. 9**, we can also see that the qualities of the peers of the five algorithms are low at the beginning. Then, the qualities increase gradually with the running of the system. After about 300 peers joined the network, the qualities tend to be stable.

In summary, through simulation experiments, we can prove that the performance of the proposed MOPSO-DS algorithm is obviously better than other algorithms. The following conclusions can be drawn: 1) The proposed MOPSO-DS algorithm can improve perceived video quality and network throughput at the same time. 2) The MOPSO-DS algorithm is a light-weight algorithm, which can be executed fast by an ordinary PC.

6. Conclusions and Future Work

In the paper, we proposed a grid-based distributed data scheduling algorithm for P2P video streaming systems. Specifically, for optimizing the perceived video quality and the network throughput at the same time, we transform the data scheduling problem into a multi-objective optimization problem. In order to solve this problem effectively, we proposed a multi-objective particle swarm optimization data scheduling (MOPSO-DS) algorithm. The algorithm codes the neighbors of peers as the positions of particles, and then obtains a external archive. Finally, the algorithm selects a nearly optimal scheme to schedule the video segments requested by the receiving peers. To verify the performance of the MOPSO-DS algorithm, we developed a P2P video stream simulator based on event-driven. Then, the other four classic algorithms are implemented in the simulator. The simulation results show that the MOPSO-DS algorithm can significantly optimize the perceived video quality and network throughput at the same time. In addition, the MOPSO-DS algorithm can run fast on an ordinary PC.

In future, there is still a lot of work to be done. First, to compared with PSNR, we will make a detailed research on the choice of the video quality metrics, such as VQM-VFD or VQM. Second, the MOPSO-DS algorithm can be redesigned in the real P2P streaming system with packet losses or frame freezing/stalling. Finally, the paper only studies single-layer video streaming system. In future work, we will research the multi-layer video streaming systems, and reconstruct the formulas and algorithms.

References

[1] PPTV. [Online]. Available: http://www.pptv.com. Article (CrossRef Link).
[2] PPS. [Online]. Available: http://www.pps.tv. Article (CrossRef Link).
[3] Y. Chu, S. Rao and H. Zhang, "A case for end system multicast," in *Proc. of Int. Conf. on Measurement and Modeling of Computer Systems*, pp. 1-12, June 17-21, 2000. Article (CrossRef Link).
[4] V. Pai, K. Kumar, K. Tamilmani and V. Sambamurthy, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. of 4th Int. Workshop on Peer-to-Peer Systems*, pp. 127-140, February 24-25, 2005. Article (CrossRef Link).
[5] XY. Zhang, JC. Liu and TSP. Yum, "Cool Streaming/ DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. of 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, pp. 2102-2111, March 13-17, 2005. Article (CrossRef Link).
[6] J. Chakareski and F. Pascal, "Utility-based packet scheduling in P2P mesh-based multicast," in *Proc. of SPIE Conf. on Visual Communications and Image Processing*, pp. 245-253, May 4-6, 2009. Article (CrossRef Link).
[7] Cheng Hsin and Mohamed Hefeeda, "Quality-aware segment transmission scheduling in peer-to-peer streaming systems," in *Proc. of the first annual ACM SIGMM conf. on Multimedia systems*, pp. 169-179, February 22-23, 2010. Article (CrossRef Link).
[8] MK. Bideh, B. Akbari and AG. Sheshjavani, "Adaptive content-and-deadline aware chunk scheduling in mesh-based P2P video streaming," *Peer-to-Peer Networking and Applications*, vol. 9, no. 2, pp. 436-448, March, 2016. Article (CrossRef Link).
[9] M. Efthymiopoulou, N. Efthymiopoulou and A.Christakidis, "Scalable playback rate control in P2P live streaming systems," *Peer-to-Peer Networking and Applications*, vol. 9, no. 6, pp. 1162-1176, November, 2016. Article (CrossRef Link).
[10] PS. Liu, GM. Huang and SZ. Feng, "Event-driven high-priority first data scheduling scheme for p2p vod streaming," *The Computer Journal*, vol. 56, no. 2, pp. 239-257, February, 2012. Article (CrossRef Link).
[11] Shen Yuanbin, Cheng Hsin and Hefeeda Mohamed, "Efficient algorithms for multi-sender data transmission in swarm-based peer-to-peer streaming systems," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 762-775, August, 2011. Article (CrossRef Link).
[12] Zhang Meng, Xiong Yongqing and Zhang Qing, "Optimizing the throughput of data-driven peer-to-peer streaming," *IEEE Transactions on Parallel and Distributed systems*, vol. 20, no. 1, pp. 97-110, January, 2009. Article (CrossRef Link).
[13] Bharambe Ashwin, Herley Cormac and Padmanabhan Venkata, "Analyzing and Improving a Bittorrent Networks Performance Mechanisms," in *Proc. of the 25th IEEE Int. conf. on computer communications*, pp. 2884-2895, April 23-29, 2006. Article (CrossRef Link).
[14] Huang Guimin, Li Chengshen and Liu Pingshan, "Load Balancing Strategy for P2P VoD Systems," *KSII Transactions on Internet & Information Systems*, vol. 10, no. 9, pp. 4207-4222, September, 2016. Article (CrossRef Link).

[15] Hu Chiacheng, Lai Chinfeng, Hou Jigong and Huang Yuehmin, “Timely scheduling algorithm for P2P streaming over MANETs,” *Computer Networks*, vol. 127, pp. 56-67, November, 2017. Article (CrossRef Link).

[16] J. Zhang and Y. Zhang, “QoS-awareness peer coordination control for topology-converging P2P live streaming,” *Multimed Tools and Applications*, vol. 76, no. 22, pp. 23834–23858, November, 2017. Article (CrossRef Link).

[17] Pal Kunwar , Govil Mahesh and Ahmed Mushtaq, “Slack time-based scheduling scheme for live video streaming in P2P network,” *International Journal of Communication Systems*, vol. 31, no. 2, pp. 1074-1090, January, 2018. Article (CrossRef Link).

[18] P. Kunwar, C. Mahesh, A. Mushtaq, “Priority-based scheduling scheme for live video streaming in peer-to-peer network,” *Multimed Tools and Applications*, vol. 77, no. 18, pp. 24427–24457, September, 2018. Article (CrossRef Link).

[19] T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, “Overview of the H.264/ AVC Video Coding Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July, 2003. Article (CrossRef Link).

[20] A. Vander, P. Geert and R. Martin, “Traffic and quality characterization of single-layer video streams encoded with the H. 264/ MPEG-4 advanced video coding standard and scalable video coding extension,” *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 698-718, January, 2008. Article (CrossRef Link).

[21] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proc. of the 6th Int. Symposium on Micor Machine and Human Science*, pp. 39-43, October 4-6, 1995. Article (CrossRef Link).

[22] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. of 1995 IEEE Int. Conf. on neural Networks*, pp. 1942-1948, November 27- December 1, 1995. Article (CrossRef Link).

[23] CR. Raquel and PC. Naval, “An effective use of crowding distance in multiobjective particle swarm optimization,” *Genetic and Evolutionary Computation Conference*, pp. 257-264, June 25-29, 2005. Article (CrossRef Link).

[24] Liu Pingshan, Feng Shengzhong and Huang Guimin, “Bandwidth-Availability-Based Replication Strategy for P2P VoD Systems,” *The Computer Journal*, vol. 57, no. 8, pp. 1211-1229, August, 2013. Article (CrossRef Link).

[25] Seeling Patrick, Reisslein Martin and Kulapala Beshan, “Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial,” *IEEE Communications Surveys and Tutorials*, vol. 6, no. 3, pp. 58-78, Third Quarter, 2004. Article (CrossRef Link).

[26] S. Shahrzad, M. Seyedali, and L. Andrew, “Enhanced multi-objective particle swarm optimisation for estimating hand postures,” *Knowledge-Based Systems*, vol. 158, pp. 175–195, October, 2018. Article (CrossRef Link).

Pingshan Liu, He received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, in 2014. Currently, he is an associate professor at Guilin University of Electronic Technology. His research interests include content delivery network, streaming media communication network and cloud computing.

Yaqing Fan, She is currently pursuing a master degree at the School of Computer Technology in Guilin University of Electronic Technology, Guilin, China. Her current research interests include the Content Distribution Network and Peer-to-Peer Network.

Xiaoyi Xiong, He received the B.S. degree from Hunan Agricultural University, Hunan, China, in 2015. He received the M.S. degree from Guilin University of Electronic Technology, Guilin, China, in 2018. His research interests include the Content Distribution Network and Peer-to-Peer Network.

Yimin Wen, He received the Ph.D degree from Shanghai Jiaotong University, Shanghai, China, in 2007. He is currently a professor in the School of Computer Science and Information Safety at Guilin University of Electronic Technology. His research interests include machine learning, pattern recognition, media and image mining, and educational data mining.

Dianjie Lu, He received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, Beijing, China, in 2012. He is currently an associate professor in School of Information Science and Engineering, Shandong Normal University, Jinan, China. His research interests include cognitive wireless network, heterogeneous cellular networks, and cloud computing.