Chemical Reaction Optimization for Population Transition in Peer-to-Peer Live Streaming

Albert Y.S. Lam, *Member, IEEE*, Jialing Xu, *Student Member, IEEE*, and Victor O.K. Li, *Fellow, IEEE*

***Abstract*— Peer-to-peer (P2P) live streaming applications are very popular in recent years and a Markov open queueing network model was developed to study the population dynamics in P2P live streaming. Based on the model, we deduce an optimization problem, called population transition problem, with the objective of maximizing the probability of universal streaming by manipulating population transition probability matrix. We employ a chemical reaction-inspired metaheuristic, Chemical Reaction Optimization (CRO), to solve the problem. Simulation results show that CRO outperforms many commonly used strategies for controlling population transition in many practical P2P live streaming systems. This work also shows that CRO also demonstrates the usability of CRO to solve optimization problems.**

I. Introduction

Peer-to-peer (P2P) streaming has become one of the killer applications in the Internet. Many companies providing commercial P2P streaming services have been established, including UUSee [1], PPStream [2], SopCast [3], etc. For example, iResearch [4] shows that PPStream services have access rates of more than 10 million per month. The huge demand of P2P streaming services creates many research problems, including scheduling [5], topology control [6], and QoS support [7].

In addition to traditional video broadcasting, P2P technology provides a new platform for live streaming with less stringent geographical constraint and at lower costs. However, the stringent time requirement of P2P live streaming creates extra challenges. Xu and Li are the first to analyze the population dynamics for live streaming over P2P networks [8]. They propose a stochastic model and a new metric is introduced to evaluate the performance of different population transition strategies.

In this work, based on [8], we formulate the model as an optimization problem, entitled the Population Transition Problem (PTP). Solution of this problem requires exponentially increasing computation time with the problem scale. Due to its complexity, it can be considered as a black box and traditional optimization techniques are not effective. Therefore, we rely on evolutionary approaches. A newly proposed metaheuristic called Chemical Reaction Optimization (CRO) [9] is adopted to tackle PTP.

CRO is a chemical reaction-inspired general-purpose optimization method and it mimics the collisions of molecules in chemical reactions. Molecular interactions bring molecules from high to low energy states. This tendency allows us to explore the global optimum of the solution space. CRO has been shown to be effective in many applications [9], [10]. This motivates us to apply CRO to solve PTP. Simulation results show that CRO outperforms other commonly used strategies.

CRO is an evolutionary algorithm. Other evolutionary algorithms have been applied to P2P, e.g. [11] and [12], but they do not consider PTP. Instead, [11] focuses on constructing decentralized and self-organized P2P information system in computational grids and [12] addresses the packet scheduling issue for QoS support in P2P video streaming. Moreover, P2P has also been used as a platform to operate evolutionary algorithms, e.g. [13], [14], [15], but this is not the focus of this paper.

The rest of the paper is organized as follows. In Section II, we formulate the problem deduced from the model given in [8]. Section III describes the framework of CRO. We give the simulation results for performance evaluation in Section IV. Finally, we conclude the paper and suggest possible future work in Section V.

II. Problem Formulation

In this section, we firstly revise the P2P streaming systems and the population dynamics model. Then we formulate the problem in the form of maximization.

A. Peer-to-Peer Live Streaming Systems

A single-channel P2P live streaming system consists of a stream source and a group of peers. The stream source provides real time streaming data (e.g. live video broadcast) to the peers. The peers may connect to networks with different topologies and traffic patterns, and thus, their cumulative transmission delays (total delay from the moment when the source sends out a packet to when the peer receives it) vary. As pointed out in [8], measuring the exact locations of peers are infeasible. So we differentiate the peers according to their cumulative transmission delays. Two peers with smaller and larger delays are called upstream and downstream peers, respectively. Upstream peers receive streaming data from the source earlier than downstream peers, act like a source and provide streaming data to its downstream peers. More upstream peers help support more downstream peers. Therefore, the total capacity of the system can improve. This demonstrates the power of P2P system compared with the traditional client-server architecture [16].

Many systems have tracker servers which provide system information to the peers. The most important information is a list of addresses of upstream peers for a peer to connect

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China (email: {ayslam, jlxu, vli}@eee.hku.hk).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

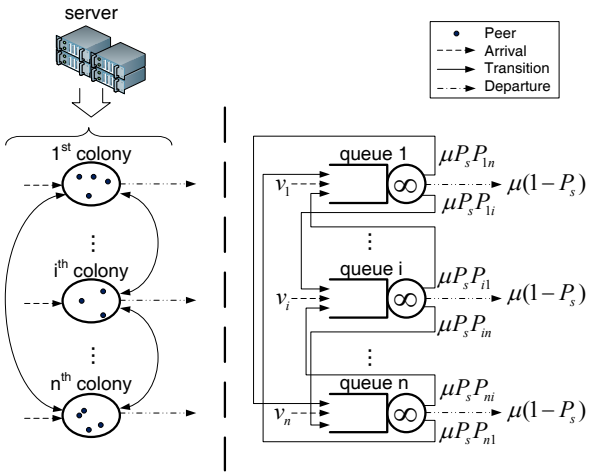


Fig. 1. Modeling colonies of peers as a Markov open queueing network

with. It is particularly useful when a peer joins a system. When a peer experiences degraded services, a tracker server may also provide it a list of recommended peers to re-attach to. Depending on the system implementation, this may be automatically completed by the streaming software. Thus, a system may control the topology of the P2P overlay network so as to improve the system performance.

B. Population Dynamics Model

We divide the peers into several colonies according to their cumulative transmission delays. Each peer joins a particular colony when participating in the streaming system and those with similar delays are put into the same colony. A peer may leave its colony and attach to another upon sensing its accumulative transmission delay has changed[1]. It may leave the system forever if it is not interested in the channel anymore.

The system is modeled as a Markov open queueing network (see Fig. 1). A peer is just like a customer in the network of queues. A peer participating in a colony, sojourning within a colony, shuttling between colonies, and quitting the system is tantamount to a customer entering a queue, getting served, transiting to other queues, and leaving the network, respectively. Here we consider a special case of the model with Poisson arrivals and exponential service times. We assume that the queues are $M/M/\infty$ as a peer immediately gets served without waiting once it joins a colony. We also assume that there are n colonies and peers join the i-th colony from outside the system as a Poisson process with arrival rate v_i for $i = 1, \ldots, n$. Peers stay at any colony for an exponentially distributed time period with mean $\frac{1}{\mu}$. After a peer has been served, it may stay in or quit the system with probability P_s and $(1 - P_s)$, respectively. If it stays, the probability of moving from the i-th to the j-th colony is P_{ij}. By *Theorem 2* of [8], the system has one and only one equilibrium distribution if there exist positive α_i's satisfying

$$\alpha_i \mu = v_i + \mu \sum_{j=1}^{n} \alpha_j P_s P_{ji}, \quad i = 1, \ldots, n, \tag{1}$$

where α_i represents the average number of peers in the i-th colony. Let x_i[2] be the population size (i.e. number of peers) of the i-th colony. By *Theorem 3* of [8], the probability of having a population size x_i for the i-th colony, $\pi_i(x_i)$, are independent of each other in the equilibrium state, and given by

$$\pi_i(x_i) = e^{-x_i} \frac{\alpha_i^{x_i}}{x_i!}, \quad i = 1, \ldots, n. \tag{2}$$

Thus, the probability of the whole network in equilibrium state $s = (x_1, \ldots, x_n)$ is given by

$$\pi(s) = \prod_{i=1}^{n} \pi_i(x_i). \tag{3}$$

A P2P streaming system is considered to be in universal streaming when all peers in the system can receive sufficient streaming data. To make the universal streaming condition tractable, we suppose that peers in the i-th colony can only get data supply from those in the same colony and their immediate upstream colony, i.e. the $(i-1)$-th colony. Moreover, the maximum number of peers with sufficient data supply in the i-th colony is determined by the population size in the $(i-1)$-th colony multipled by a sufficient provision ratio b[3], i.e.,

$$x_i \leq \lfloor x_{i-1} \cdot b \rfloor.$$

Let B be the capacity of the stream source. The probability of universal streaming is defined as

$$P_{us} = \Pr\{x_1 \leq B, x_2 \leq \lfloor x_1 \cdot b \rfloor, \ldots, x_n \leq \lfloor x_{n-1} \cdot b \rfloor\}.$$

With the consideration of system state probability given by (3), P_{us} can be further expressed as

$$P_{us} = \sum_{s \in \Phi} \pi(s), \tag{4}$$

where Φ is the set of all system states satisfying universal streaming, i.e.,

$$\begin{aligned} \Phi = \{&(x_1, \ldots, x_n) | x_1 \leq B, \\ &x_2 \leq \lfloor x_1 \cdot b \rfloor, \ldots, x_n \leq \lfloor x_{n-1} \cdot b \rfloor\}. \end{aligned}$$

Here we only give an overview of the population dynamics model. For more information about the model, interested readers may refer to [8].

[1]A peer which participates in an inappropriate colony may degrade the overall system performance. For example, a peer with high delay is not suitable to act as a source for transmitting data to peers with lower delay.

[2]x_i must be a non-negative integer.

[3]b usually has a value between one and two in practice.

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \mu \begin{bmatrix} (1-P_sP_{11}) & -P_sP_{21} & \cdots & -P_sP_{n1} \\ -P_sP_{12} & (1-P_sP_{22}) & \cdots & -P_sP_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ -P_sP_{1n} & -P_sP_{2n} & \cdots & (1-P_sP_{nn}) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad (6)$$

C. Population Transition Problem

The system parameters B, b, n, P_s, μ, and $v_i, i = 1, \ldots, n$, characterize the P2P streaming system. Recall that the system can control the topology of the overlay P2P network for providing better service. When a peer needs to switch from a colony to another, the system can determine to which colony it should connect (by controlling the list of available upstream peers provided to the peer). One simple implementation is to control the population transition strategy, i.e. the population transition probability matrix $[P_{ij}]_{1 \le i,j \le n}$ (or $[P_{ij}]$ in short), so as to maximize the probability of universal streaming P_{us}.

Here we try to demonstrate how to compute P_{us} corresponding to a matrix $[P_{ij}]$. Recall that P_{ij} is the probability of a peer transiting from the i-th to the j-th colony. Therefore, we have

$$\sum_{j=1}^{n} P_{ij} = 1, \quad i = 1, \ldots, n, \quad (5)$$

and it is a constraint of PTP. From (1), we have

$$v_i = \mu(\alpha_i - \sum_{j=1}^{n} \alpha_j P_s P_{ji}), \quad i = 1, \ldots, n,$$

which in matrix form gives (6) or

$$\boldsymbol{v} = \mu(\boldsymbol{I} - P_s \boldsymbol{P}^T)\boldsymbol{\alpha}, \quad (7)$$

where $\boldsymbol{v}$ is the arrival rate vector $[v_1, \ldots, v_n]^T$, $\boldsymbol{I}$ is an identity matrix of size n, $\boldsymbol{P}$ is the transition matrix of the Markov chain $[P_{ij}]$, and $\boldsymbol{\alpha}$ equals $[\alpha_1, \ldots, \alpha_n]^T$. We need to solve the system of linear equations (7) to get $\boldsymbol{\alpha}$. Let M and $\tilde{M}$ be the matrix $\mu(\boldsymbol{I} - P_s\boldsymbol{P}^T)$ and the augmented matrix of M, respectively, i.e.,

$$\tilde{M} = \left[\begin{array}{cccc|c} \mu(1-P_sP_{11}) & -\mu P_sP_{21} & \cdots & -\mu P_sP_{n1} & v_1 \\ -\mu P_sP_{12} & \mu(1-P_sP_{22}) & \cdots & -\mu P_sP_{n2} & v_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\mu P_sP_{1n} & -\mu P_sP_{2n} & \cdots & \mu(1-P_sP_{nn}) & v_n \end{array}\right]$$

By the Kronecker-Capelli Theorem [17], (7) has a solution for $\boldsymbol{\alpha}$ provided that

$$rank(M) = rank(\tilde{M}). \quad (8)$$

Therefore, (8) is also a constraint of PTP. We use (7) to compute $\boldsymbol{\alpha}$ from $\boldsymbol{P}$.

For a particular state $s = (x_1, \ldots, x_n)$, we calculate its probability with (2) and (3). We determine P_{us} by summing the probabilities of all states satisfying universal streaming with (4). Consider an example when $n = 2$, $B = 5$, and $b = 1.5$. The feasible states (x_1, x_2) include:

$$\begin{array}{lllll} (5,0), & (4,0), & (3,0), & (2,0), & (1,0), \quad (0,0), \\ (5,1), & (4,1), & (3,1), & (2,1), & (1,1), \\ (5,2), & (4,2), & (3,2), & (2,2), & \\ (5,3), & (4,3), & (3,3), & (2,3), & \\ (5,4), & (4,4), & (3,4), & & \\ (5,5), & (4,5), & & & \\ (5,6), & (4,6), & & & \\ (5,7). & & & & \end{array}$$

By inspection, the size of the feasible state space Φ is expressed as

$$\sum_{i_1=0}^{\lfloor i_0 b \rfloor} \sum_{i_2=0}^{\lfloor i_1 b \rfloor} \cdots \sum_{i_{n-1}=0}^{\lfloor i_{n-2} b \rfloor} (1 + \lfloor i_{n-1} b \rfloor), \quad (9)$$

where $\lfloor i_0 b \rfloor$ is equal to B. This shows that the computation of P_{us} grows exponentially with n, B, and b.

A possible solution ω of the problem is a probability transition matrix $[P_{ij}]$ and the corresponding objective function value y is its P_{us}. The above process, denoted by f, shows how to compute P_{us} for a particular $[P_{ij}]$, i.e.,

$$P_{us} = f([P_{ij}]). \quad (10)$$

It is the objective function which we need to maximize. (9) reveals that f is very computationally intensive when the scale of the P2P streaming system grows. To summarize, PTP is mathematically represented by

$$\max \quad P_{us} = f([P_{ij}])$$

subject to

$$\sum_{j=1}^{n} P_{ij} = 1, \qquad i = 1, \ldots, n,$$
$$rank(M) = rank(\tilde{M}).$$

Different system characteristics (e.g. values of μ and v) imply different problem instances of PTP, and thus, the optimal solutions corresponding to different instances may vary. In other words, we need to solve the PTP whenever the system parameters change. According to [18], the system characteristics follow similar daily patterns. Therefore, we can construct several instances of PTP with representative system parameter settings corresponding to such patterns, and the computed $[P_{ij}]$'s can be utilized to optimize the system performance. We will present an algorithm to solve PTP in the next section.

```
procedure InitialSolnGen(n)
  for i := 1 to n
    Randomly generate P_{i1}, ..., P_{in} ∈ [0,1]
    sum := P_{i1} + ... + P_{in}
    for j := 1 to n
      P_{ij} := P_{ij}/sum
    end for
  end for
  ω := [P_{ij}]_{1≤i,j≤n}
  return ω
end procedure
```

Fig. 2. Pseudocode of initial solution generator

```
procedure Neighbor(ω)
  ω' := ω
  Randomly generate an integer i ∈ [1,n]
  Randomly generate two distinct integers
j,k ∈ [1,n]
  sum := P_{ij} + P_{ik}
  Randomly generate a real number t ∈ [0,1]
  P'_{ij} := sum × t
  P'_{ik} := sum − P'_{ij}
  return ω'
end procedure
```

Fig. 3. Pseudocode of neighborhood solution generator

III. Algorithm Design

There are not many optimization problems which have right stochastic matrices as solutions. Thus, we introduce several operators to deal with the solutions satisfying the constraints posed on PTP. Then we discuss how CRO works.

A. Operators

1) Initial Solution Generator: This operator is used to generate the initial solutions for CRO to manipulate. Each call to the operator will generate a right stochastic matrix of order n. It is done by firstly generating a matrix of order n with element $P_{ij} \in [0,1]$ randomly. Then we transform the elements so that the sum of each row is equal to one. For example,

$$\begin{bmatrix} 0.79 & 0.04 & 0.68 \\ 0.96 & 0.85 & 0.76 \\ 0.66 & 0.93 & 0.74 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.53 & 0.02 & 0.45 \\ 0.37 & 0.33 & 0.30 \\ 0.28 & 0.40 & 0.32 \end{bmatrix}$$

Its pseudocode is given in Fig. 2.

2) Neighborhood Solution Generator: This operator is used to generate a new solution ω' in the neighborhood of a given solution ω. It is done by randomly choosing a row i $(1 \le i \le n)$ of ω first. Then we select two distinct elements P_{ij} and P_{ik} $(1 \le j,k \le n)$ in row i. Next we divide the sum of P_{ij} and P_{ik} into two random portions and assign each of them to P'_{ij} and P'_{ik} of ω', respectively. The rest of the elements in ω' are just the same as those in ω. For example, P_{12} and P_{13} are chosen in the following:

$$\begin{bmatrix} 0.53 & \mathbf{0.02} & \mathbf{0.45} \\ 0.37 & 0.33 & 0.30 \\ 0.28 & 0.40 & 0.32 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.53 & \mathbf{0.16} & \mathbf{0.31} \\ 0.37 & 0.33 & 0.30 \\ 0.28 & 0.40 & 0.32 \end{bmatrix}$$

Its pseudocode is given in Fig. 3.

3) Decomposition: This operator is used to produce two new solution ω'_1 and ω'_2 from a given solution ω. For each row of ω, we assign it to the same row of either ω'_1 or ω'_2 randomly. For those rows in ω'_1 or ω'_2 which have not been assigned with values, we make them rows of random positive values whose total is equal to one. This mechanism is similar to that used in the initial solution generator to produce a random row of numbers. For example,

```
procedure Decompose(ω)
  for i := 1 to n
    Generate a random number t ∈ [0,1]
    if t < 0.5
      [P'_{(1)i1}, ..., P'_{(1)in}] := [P_{i1}, ..., P_{in}]
      Randomly generate P'_{(2)i1}, ..., P'_{(2)in} ∈ [0,1]
      sum := P'_{(2)i1} + ... + P'_{(2)in}
      for j := 1 to n
        P'_{(2)ij} := P'_{(2)ij}/sum
      end for
    else
      [P'_{(2)i1}, ..., P'_{(2)in}] := [P_{i1}, ..., P_{in}]
      Randomly generate P'_{(1)i1}, ..., P'_{(1)in} ∈ [0,1]
      sum := P'_{(1)i1} + ... + P'_{(1)in}
      for j := 1 to n
        P'_{(1)ij} := P'_{(1)ij}/sum
      end for
    end if
  end for
  return ω'_1 and ω'_2
end procedure
```

Fig. 4. Pseudocode of decomposition

$$\begin{bmatrix} \mathbf{0.53} & \mathbf{0.02} & \mathbf{0.45} \\ \underline{0.37} & \underline{0.33} & \underline{0.30} \\ \mathbf{0.28} & \mathbf{0.40} & \mathbf{0.32} \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} \mathbf{0.53} & \mathbf{0.02} & \mathbf{0.45} \\ 0.34 & 0.62 & 0.04 \\ \mathbf{0.28} & \mathbf{0.40} & \mathbf{0.32} \end{bmatrix} \textbf{ AND } \begin{bmatrix} 0.84 & 0.04 & 0.12 \\ \underline{0.37} & \underline{0.33} & \underline{0.30} \\ 0.13 & 0.22 & 0.65 \end{bmatrix}$$

Its pseudocode is given in Fig. 4.

4) Synthesis: This operator is used to produce a new solution ω' from two solutions ω_1 and ω_2. For each row of ω', we assign it a corresponding row from either ω_1 or

```
procedure Synthesize(ω1, ω2)
  for i := 1 to n
    Generate a random number t ∈ [0, 1]
    if t < 0.5
      [P'_i1, ..., P'_in] := [P_(1)i1, ..., P_(1)in]
    else
      [P'_i1, ..., P'_in] := [P_(2)i1, ..., P_(2)in]
    end if
  end for
  return ω'
end procedure
```

Fig. 5. Pseudocode of synthesis

ω_2 randomly. For example,

$$\begin{bmatrix} 0.53 & 0.02 & 0.45 \\ 0.37 & 0.33 & 0.30 \\ \mathbf{0.28} & \mathbf{0.40} & \mathbf{0.32} \end{bmatrix} \textbf{AND} \begin{bmatrix} \underline{0.34} & \underline{0.62} & \underline{0.04} \\ \underline{0.84} & \underline{0.04} & \underline{0.12} \\ 0.13 & 0.22 & 0.65 \end{bmatrix}$$
$$\Rightarrow \begin{bmatrix} \underline{0.34} & \underline{0.62} & \underline{0.04} \\ \underline{0.84} & \underline{0.04} & \underline{0.12} \\ \mathbf{0.28} & \mathbf{0.40} & \mathbf{0.32} \end{bmatrix}$$

Its pseudocode is given in Fig. 5.

B. Chemical Reaction Optimization

CRO is a population-based metaheuristic, inspired by chemical reactions. Imagine that we have a system of some unstable molecules in a closed container. They are unstable because they have excess energy. The energy needs to be released so that the system becomes more stable. Thus they interact with each other through collisions and change from high to low energy states. The initial and final molecules are called reactants and final products, respectively. The whole process is what we call a chemical reaction.

Each molecule has a molecular structure with two kinds of energy, i.e. potential energy and kinetic energy. We mimic a solution of an optimization problem as a molecular structure ω. The potential energy (PE_ω) is the objective function value corresponding to ω while the kinetic energy (KE_ω) can be interpreted as the tolerance of about how much worse (i.e. more rather less energy) the molecule can change[4]. Suppose that the molecule gets changed from ω to ω'. It is allowed only when the molecule have sufficient energy to support the change, i.e. $PE_\omega + KE_\omega \geq PE_{\omega'}$. In other words, the change is always accepted if $PE_\omega > PE_{\omega'}$, but it is rejected when $PE_{\omega'} - PE_\omega > KE_\omega$. This explains the solution acceptance rule of CRO for a change of a single molecule. There are changes involving two or more molecules. The rule works similarly but more vigorous changes are possible as more energy is engaged.

Molecules collide either on the walls of the container or with each other. The fraction of collisions corresponding to the second kind is *CollMole*. We define four types of elementary reactions, i.e. on-wall ineffective collision, decomposition, intermolecular collision, and synthesis, to characterize different collisions. The former two can be triggered when a molecule hits on a wall. The latter two can take place when molecules collide with each other (we assume only two molecules are involved in the simulation). We follow the framework described in [9] to implement the algorithm. Interested readers can refer to [9] to comprehend how the whole algorithm works. To make it suitable for solving PTP, we have several modifications to the operators used in CRO. We adopt the initial solution generator described in Section III-A.1 to produce *PopSize* number of molecules at initialization. Their initial KEs are assigned a value equal to *InitialKE*. The neighborhood solution generator given in Section III-A.2 is utilized in on-wall and inter-molecular ineffective collisions to generate new solutions in the neighborhoods of existing ones. We apply the operators from Section III-A.3 and Section III-A.4 to perform decomposition and synthesis, respectively.

[4]Energy must be of non-negative real numbers.

IV. Simulation

In this section, we firstly explain how we handle the constraints and then compare our solution with a set of common benchmarks. Next we describe the configuration of the simulation and finally discuss the results.

A. Constraints Handling

We have two types of constraints, i.e. (5) and (8). We try to confine the search to the feasible solution space. In other words, we keep generating solutions without violating any constraints in both initialization and iterations of CRO. To do this, we handle the two types of constraints separately. For (5), the initial solution generator ensures that all solutions it produces are right stochastic matrices. Moreover, the neighborhood solution generator, decomposition and synthesis operators are specially designed to transform right stochastic matrices to right stochastic matrices as well. For (8), with the possible values (non-negative real numbers) assigned to $\boldsymbol{v}$, μ, P_s, and $\boldsymbol{P}$, it is uncommon to generate a $[P_{ij}]$ which makes the system of linear equations (7) inconsistent[5]. If it does happen, we can simply ignore that $[P_{ij}]$ and generate a new one instead.

B. Benchmark Problems

To have fair comparisons of performance over various strategies, we need a set of common benchmark problems in the simulation. Let V_T be the overall external arrival rate to the system, i.e. $V_T = v_1 + v_2, \ldots, v_n$. We randomly generate 60 problem instances with $n = 4$ and $V_T = 2, 4, \ldots, 20$. For each V_T, there are 6 problem instances, each of which is generated by randomly dividing V_T into four parts, representing v_1, v_2, v_3, and v_4, respectively. Each instance is represented by a number formatted in the form

[5]We seldom encounter $[P_{ij}]$ making (7) inconsistent in the mass simulation.

of "V_T_case number". For example, "20_5" means that it is the fifth instance with V_T equal to 20.

C. Implementation Details

The simulation codes are programmed in MATLAB because MATLAB has powerful and efficient operators to handle matrices and to solve (7) [19]. In computing (10), we follow [8] to set the system parameters, i.e. $B = 15$, $b = 1.5$, $P_s = 0.6$, and $\mu = 0.5$. For all the dynamic strategies (introduced later), the stopping criterion is when the maximum number of function evaluations (FEs), set to 1000, is reached. After several trial runs of the simulation, the parameter values of CRO are set as follows: *PopSize* $= 15$; *KELossRate* $= 0.2$; *MoleColl* $= 0.3$; *InitialKE* $= 10$; $\alpha = 100$; and $\beta = 0.5$.

PTP is a maximization problem. However, CRO is originally designed to solve minimization problems and the objective function value is interpreted as energy, which is of non-negative value. The typical way of converting a maximization problem f to a minimization one (i.e. letting $-f$ as the objective function) may not be appropriate. Instead, we can consider $f' = \mathit{offset} - f$, where *offset* is supposed to be large enough to make every possible f' non-negative. After minimizing f', we can compute the corresponding f by $f = \mathit{offset} - f'$. In this simulation, as P_{us} is upper bounded by one, we set the *offset* equal to one accordingly. This also demonstrates a technique on how to employ CRO to solve maximization problems.

D. Results

We compare the performance of CRO with other strategies which also manipulate $[P_{ij}]$ as a solution, i.e. random search (Random), inert staying (IS), shorter delay (SD), and random walk (RW). CRO and Random are considered as dynamic approaches as they produce $[P_{ij}]$ with respect to the problems, i.e. different $[P_{ij}]$ are produced according to the values of $\{v_1, \ldots, v_n\}$. Random is introduced to serve as a reference for the dynamic approaches. IS, SD, and RW (introduced in [8]) are regarded as static strategies since they always use the same $[P_{ij}]$ for different problems. Despite this, they still have practical use [8], [20], [21]. All simulations are performed on the same PC with Intel Core 2 Quad Processor Q9650 and 3GB of RAM. Each run of CRO and Random takes approximately 8.45s and 8.20s, respectively, and thus CRO suffers a little increase in computation time. As IS, SD, and RW always retain the same solution, one FE is required and they consume around $\frac{8.20}{1000}$s of computation time in each simulation run.

The simulation results for all problem instances are shown in Fig. 6. Each data point of CRO and Random is the mean of the results of 50 runs[6] while the plus and minus error bars indicate the best and worst results of all 50 runs, respectively. The upper and lower rows of numbers in each chart show the standard deviations of the 50 runs for CRO and Random, separately. For some instances (e.g. 8_6, 10_6, 12_1, etc.), CRO performs almost the same as Random because these instances are particularly hard to solve, and thus, CRO does not show superiority over Random on the average. We can see that, in general, dynamic approaches can produce better results than the static ones. Moreover, CRO is superior to the others in most cases with less fluctuations across the instances of the same V_T. This reveals that CRO is more reliable to generate good solutions when solving new problem instances.

[6]CRO and Random are stochastic methods and the result produced in different runs may vary.

TABLE I
AVERAGE % IMPROVEMENT OF CRO OVER OTHER METHODS

V_T	Random	IS	SD	RW	MSS
2	37.03	234.82	120.65	131.69	1.61
4	31.21	640.37	175.51	197.60	13.84
6	15.14	1421.87	175.82	186.66	24.28
8	6.29	1167.58	81.67	102.77	24.04
10	10.01	1004.38	90.29	140.53	16.84
12	1.42	664.41	9.37	101.11	19.89
14	6.40	12948.37	92.18	596.64	19.10
16	45.31	5604870.26	123.75	898.28	-0.42
18	22.57	25371.29	70.47	1221.48	19.26
20	51.33	96047.50	169.53	6372.97	13.48

We also study the average percentage (%) improvement of CRO over other strategies, which is computed by

$$\frac{\mathit{Result}_{CRO} - \mathit{Result}_{\xi}}{\mathit{Result}_{\xi}} \times 100\%,$$

where ξ is any algorithm other than CRO. In Table I, each value is computed with the mean for all instances with respect to the same V_T. Although the "more streaming supply" (MSS) strategy [8] does not manipulate $[P_{ij}]$ and thus it does not fit into the framework of PTP, we also include it in the table for completeness, as MSS is similar to the coolstreaming scheme [22]. CRO is obviously much better than IS and RW and it also outperforms the rest on the average.

For the dynamic approaches, we also give their convergence curves in Fig. 7, which shows the traces of best solutions found in a particular run. We only show the curves for the cases where CRO has best (20_5), median (2_4), and worst (12_1) performance over Random. We record one data point every 50 FEs. At the beginning (from 1 to 100 FEs, CRO may be inferior to Random because the performance at this stage highly depends on the initial solution. However, CRO gradually improves the solutions and get better ones after 100 FEs. Fig. 7 also shows that the stopping criterion used in the simulation is appropriate.

V. CONCLUSION

P2P live streaming is one of the killer applications in the Internet and the increasing size of P2P live streaming systems creates many challenging problems. Thus a Markov open queueing network model was developed to study the

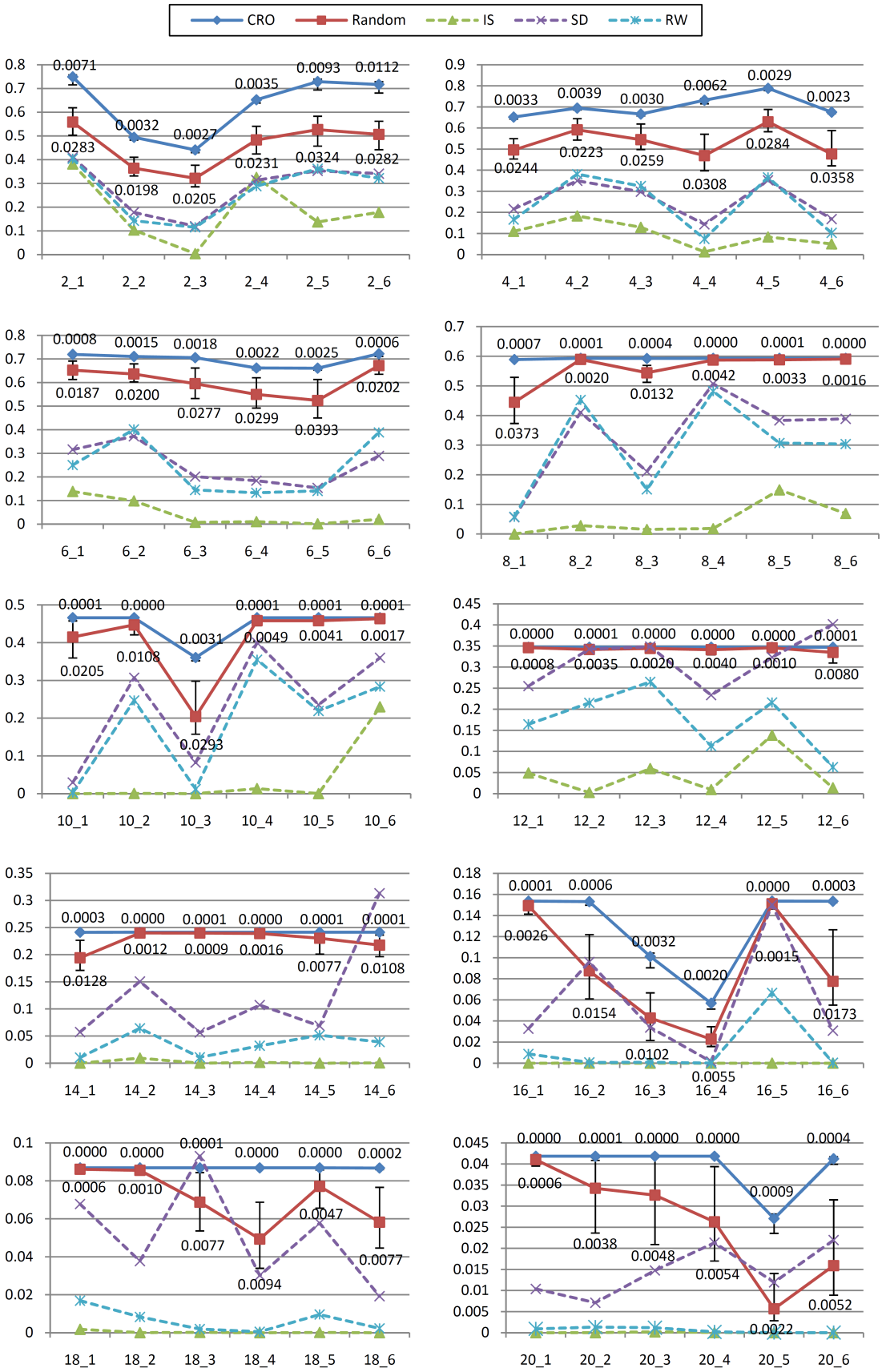
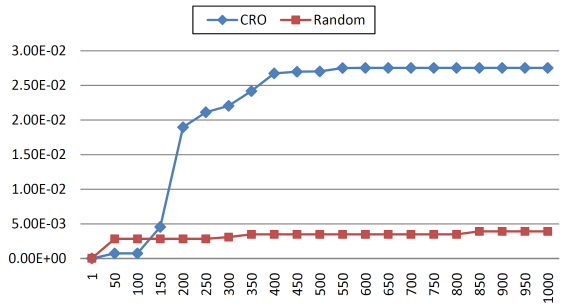
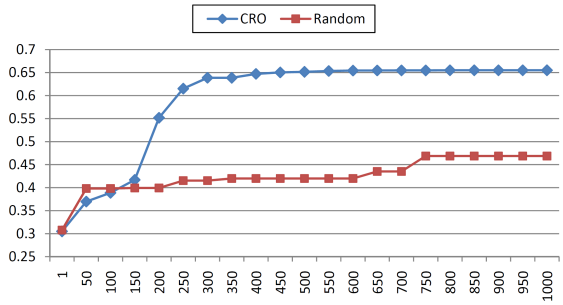


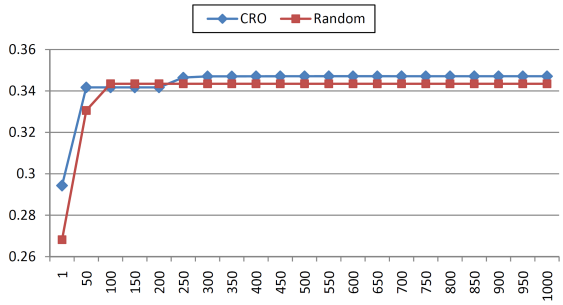
Fig. 6. Simulation results



(a) Best case : 20_5



(b) Median case : 2_4



(c) Worst case : 12_1

Fig. 7. Convergence curve

population dynamics [8]. From the model, we deduce an optimization problem, i.e. PTP, for maximizing the probability of universal streaming by manipulating the population transition probability matrix. Due to its complexity, we use evolutionary approaches to tackle PTP, instead of traditional optimization techniques. We employ a newly proposed chemical reaction-inspired metaheuristic CRO. Simulation results show that CRO outperforms many strategies for controlling population transition in existing P2P live streaming systems. Our contribution includes: 1) proposing PTP for P2P live streaming and 2) successfully applying CRO to solve the problem. This work also demonstrates the capability of CRO in solving optimization problems. In the future, we will design better operators for CRO and compare CRO with other evolutionary algorithms (e.g. genetic algorithm, simulated annealing, particle swarm optimization, etc) in solving PTP. We will also try to apply CRO to solve other problems in P2P systems.

ACKNOWLEDGMENT

This work is supported in part by the Strategic Research Theme of Information Technology of The University of Hong Kong.

REFERENCES

[1] UUSee. [Online]. Available: http://www.uusee.com/
[2] PPStream. [Online]. Available: http://www.pps.tv/
[3] SopCast. [Online]. Available: http://www.sopcast.com/
[4] China Internet Reserach Center. [Online]. Available: http://english.iresearch.com.cn/
[5] C. Liang, Y. Guo, and Y. Liu, "Is random scheduling sufficient in P2P video streaming?," *Proc. 28th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Beijing, China, June. 2008, pp. 53–60.
[6] A. Auvinen, T. Keltanen, and M. Vapa, "Topology management in unstructured P2P networks using neural networks," *Proc. IEEE Congress on Evol. Comput. (CEC)*, Singapore, Sept. 2007, pp. 2358–2365.
[7] M. Wang, L. Xu, and B. Ramamurthy, "Providing statistically guaranteed streaming quality for peer-to-peer live streaming," *Proc. 18th ACM Int. Workshop on Netw. and Operating Syst. Support for Digit. Audio and Video (NOSSDAV)*, Williamsburg, VA, Jun. 2009, pp. 127–132.
[8] J. Xu and V. O. K. Li, "A stochastic model of the population dynamics in P2P live streaming," submitted for publication. (Also available as Tech. Rep. TR-2010-001, Dept. of EEE, The Univ. of Hong Kong, Hong Kong, Jan. 2010)
[9] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, accepted for publication. (Also available as Tech. Rep. TR-2009-003, Dept. of EEE, The Univ. of Hong Kong, Hong Kong, Apr. 2009)
[10] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for the grid scheduling problem," *Proc. IEEE Int. Conf. Commun. (ICC)*, Cape Town, South Africa, May 2010.
[11] A. Forestiero and C. Mastroianni, "A swarm algorithm for a self-structured P2P information system," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 681–694, Aug. 2009.
[12] Y. H. Jung, H.-S. Kim, and Y. Choe, "Ant colony optimization based packet scheduler for peer-to-peer video streaming," *IEEE Commun. Lett.*, vol. 13, no. 6, pp. 441–443, Jun. 2009.
[13] I. Scriven, A. Lewis, and S. Mostaghim, "Dynamic search initialization strategies for multi-objective optimization in peer-to-peer networks," *Proc. IEEE Congress on Evol. Comput. (CEC)*, Trondheim, Norway, May 2009, pp. 1515–1522.
[14] J. L. J. Laredo, P. A. Castillo, A. M. Mora, and J. J. Merelo, "Exploring population structures for locally concurrent and massively parallel evolutionary algorithms," *Proc. IEEE World Congress on Comput. Intell. (WCCI)*, Hong Kong, Jun. 2008, pp. 2605–2612.
[15] I. Scriven, A. Lewis, D. Ireland, and J. Lu, "Decentralised distributed multiple objective particle swarm optimization using peer to peer networks," *Proc. IEEE World Congress on Comput. Intell. (WCCI)*, Hong Kong, Jun. 2008, pp. 2925–2928.
[16] A. S. Tanenbaum, *Computer Networks*, 4th ed. Upper Saddle River, NJ: Prentice Hall PTR, 2003, pp. 4.
[17] A. D. Polyanin and A. V. Manzhirov, *Handbook of Mathematics for Engineers and Scientists*, Boca Raton, FL: Chapman & Hall/CRC, 2007, pp. 198.
[18] C. Wu, B. Li, and S. Zhao, "Diagnosing network-wide P2P live streaming inefficiencies," *Proc. 29th IEEE Conf. Comput. Commun. (INFOCOM)*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2731–2735.
[19] Left or right matrix division - MATLAB. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/techdoc/ref/mrdivide.html/
[20] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast," *Proc. 14th IEEE Int. Conf. Netw. Protocols (ICNP)*, Santa Barbara, CA, Nov. 2006, pp. 2–11.
[21] Y.-H. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. 2000 ACM Int. Conf. Meas. and Modeling of Comput. Syst. (SIGMETRICS)*, Santa Clara, CA, Jun. 2000, pp. 1–12.
[22] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "Coolstreaming: design, theory, and practice," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1661–1671, Dec. 2007.