A Particle Swarm Optimization Algorithm for Neighbor Selection in Peer-to-Peer Networks

Shichang Sun[1,3], Ajith Abraham[2,4], Guiyong Zhang[3], Hongbo Liu[3,4]
[1]School of Computer Science and Engineering, Dalian Nationalities University, Dalian 116600, China
[2]Centre for Quantifiable Quality of Service in Communication Systems
Faculty of Information Technology, Mathematics and Electrical Engineering,
Norwegian University of Science and Technology, N-7491 Trondheim, Norway
[3]Department of Computer Science, Dalian University of Technology, Dalian 116024, China
[4]School of Computer Science, Dalian Maritime University, Dalian 116026, China
ssc@dlnu.edu.cn, ajith.abraham@ieee.org, zgy197842@gmail.com, lhb@dlut.edu.cn

Abstract

Peer-to-peer (P2P) topology has significant influence on the performance, search efficiency and functionality, and scalability of the application. In this paper, we propose a Particle Swarm Optimization (PSO) approach to the problem of Neighbor Selection (NS) in P2P Networks. Each particle encodes the upper half of the peer-connection matrix through the undirected graph, which reduces the search space dimension. The results indicate that PSO usually required shorter time to obtain better results than Genetic Algorithm (GA), specially for large scale problems.

1. Introduction

Peer-to-peer computing has attracted great interest and attention of the computing industry and gained popularity among computer users and their networked virtual communities [1]. It is no longer just used for sharing music files over the Internet. Many P2P systems have already been built for some new purposes and are being used. An increasing number of P2P systems are used in corporate networks or for public welfare (e.g. providing processing power to fight cancer) [2]. P2P comprises peers and the connections between these peers. These connections may be directed, may have different weights and are comparable to a graph with nodes and vertices connecting these nodes. Defining how these nodes are connected affects many properties of an architecture that is based on a P2P topology, which significantly influences the performance, search efficiency and functionality, and scalability of a system. A common difficulty in the current P2P systems is caused by the dynamic membership of peer hosts. This results in a constant reorganization of the topology [3], [12], [13], [14], [15].

Kurmanowytsch et al. developed the P2P middleware systems to provide an abstraction between the P2P topology and the applications that are built on top of it [4]. These middleware systems offer higher-level services such as distributed P2P searches and support for direct communication among peers. The systems often provide a pre-defined topology that is suitable for a certain task (e.g., for exchanging files). Koulouris et al. presented a framework and an implementation technique for a flexible management of peer-to-peer overlays [5]. The framework provides means for self-organization to yield an enhanced flexibility in instantiating control architectures in dynamic environments, which is regarded as being essential for P2P services to access, routing, topology forming, and application layer resource management. In these P2P applications, a central tracker decides about which peer becomes a neighbor to which other peers. Koo et al. [6] investigated the neighbor-selection process in the P2P networks, and proposed an efficient neighbor-selection strategy based on Genetic Algorithm (GA).

Particle Swarm Optimization (PSO) algorithm is inspired by social behavior patterns of organisms that live and interact within large groups. In particular, PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the Swarm Intelligence(SI) paradigm has emerged [7, 8]. It could be implemented and applied easily to solve various function optimization problems, or the problems that can be transformed to function optimization problems. As an algorithm, the main strength of PSO is its fast convergence, which compares favorably with many global optimization algorithms [9, 10]. In this paper, we explore the neighbor-selection problem based PSO for P2P

Networks.

This paper is organized as follows. We formulate the problem in Section 2. The proposed approach based on particle swarm algorithm is presented in Section 3. In Section 4, experiment results and discussions are provided in detail, followed by some conclusions in Section 5.

2. Neighbor-selection problem in P2P networks

Kooa et al. model the neighborhood selection problem using an undirected graph and attempted to determine the connections between the peers [6]. Given a fixed number of N peers, we use a graph $G = (V, E)$ to denote an overlay network, where the set of vertices $V = \{v_1, \cdots, v_N\}$ represents the N peers and the set of edges $E = \{e_{ij} \in \{0, 1\}, i, j = 1, \cdots, N\}$ represents their connectivities : $e_{ij} = 1$ if peers i and j are connected, and $e_{ij} = 0$ otherwise. For an undirected graph, it is required that $e_{ij} = e_{ji}$ for all $i \neq j$, and $e_{ij} = 0$ when $i = j$. Let C be the entire collection of content pieces, and we denote $\{c_i \subseteq C, i = 1, \cdots, N\}$ to be the collection of the content pieces each peer i has. We further assume that each peer i will be connected to a maximum of d_i neighbors, where $d_i < N$. The disjointness of contents from peer i to peer j is denoted by $c_i \setminus c_j$, which can be calculated as:

$$c_i \setminus c_j = c_i - (c_i \cap c_j). \tag{1}$$

where $\setminus$ denotes the intersection operation on sets. This disjointness can be interpreted as the collection of content pieces that peer i has but peer j does not. In other words, it denotes the pieces that peer i can upload to peer j. Moreover, the disjointness operation is not commutative, i.e., $c_i \setminus c_j \neq c_j \setminus c_i$. We also denote $|c_i \setminus c_j|$ to be the cardinality of $c_i \setminus c_j$, which is the number of content pieces peer i can contribute to peer j. In order to maximize the disjointness of content, we want to maximize the number of content pieces each peer can contribute to its neighbors by determining the connections e_{ij}'s. Define ϵ_{ij}'s to be sets such that $\epsilon_{ij} = C$ if $e_{ij} = 1$, and $\epsilon_{ij} = \emptyset$ (null set) otherwise. Therefore we have the following optimization problem:

$$\max_E \sum_{j=1}^{N} \Big| \bigcup_{i=1}^{N} (c_i \setminus c_j) \cap \epsilon_{ij} \Big| \tag{2}$$

Subject to

$$\sum_{j=1}^{N} e_{ij} \leq d_i \text{ for all } i$$

3. Particle swarm heuristic for NS

Given a P2P state $S = (N, C, M, f)$, in which N is the number of peers, C is the entire collection of content pieces, M is the maximum number of the peers which each peer can connect steadily in the session, f is to goal the number of swap pieces, i.e. to maximize Equation (2). To apply the particle swarm algorithm successfully for the NS problem, one of the key issues is the mapping of the problem solution into the particle space, which directly affects its feasibility and performance. Usually, the particle's position is encoded to map each dimension to one directed connection between peers, i.e. the dimension is $N * N$. But the neighbor topology in P2P networks is an undirected graph, i.e. $e_{ij} = e_{ji}$ for all $i \neq j$. We set up a search space of D dimension as $N * (N - 1)/2$. Accordingly, each particle's position is represented as a binary bit string of length D. Each dimension of the particle's position maps one undirected connection. The domain for each dimension is limited to 0 or 1.

The particle swarm model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the D-dimension problem space to search the new solutions, where the fitness f can be measured by calculating the number of condition attributes in the potential reduction solution. Each particle has a position represented by a position-vector $\vec{p}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\vec{v}_i$. Each particle remembers its own best position so far in a vector $\vec{p}_i^{\#}$, and its j-th dimensional value is $p_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\vec{p}^*$, and its j-th dimensional value is p_j^*. When the particle moves in a state space restricted to zero and one on each dimension, the change of probability with time steps is defined as follows:

$$P(p_{ij}(t) = 1) = f(p_{ij}(t-1), v_{ij}(t-1), p_{ij}^{\#}(t-1), p_j^*(t-1)). \tag{3}$$

where the probability function is

$$sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}}. \tag{4}$$

At each time step, each particle updates its velocity and moves to a new position according to Eqs.(5) and (6):

$$\begin{aligned} v_{ij}(t) = wv_{ij}(t-1) + c_1 r_1 (p_{ij}^{\#}(t-1) - p_{ij}(t-1)) \\ + c_2 r_2 (p_j^*(t-1) - p_{ij}(t-1)) \end{aligned} \tag{5}$$

$$p_{ij}(t) = \begin{cases} 1 & \text{if } \rho < sig(v_{ij}(t)); \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Where c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. r_1 and r_2 are the random numbers in the interval [0,1]. The variable w is called as the inertia factor, which value is typically setup to vary linearly from 1 to near 0 during the iterated processing. ρ is random number in the closed interval [0, 1]. From Eq.(5), a particle decides where to move next, considering its current state, its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm. The particle has a priority levels according to the order of peers. The sequence of the peers will be not changed during the iteration. Each particle's position indicates the potential connection state. The pseudo-code for the particle swarm search method is illustrated in Algorithm 1.

Algorithm 1 Neighbor Selection Algorithm Based on Particle Swarm

```
01.Initialize the size of the particle swarm $n$, and other parameters.
02.Initialize the positions and the velocities for all the particles randomly.
03.While (the end criterion is not met) do
04.   $t = t + 1$;
05.   For $s$ = 1 to $n$
06.       For $i$ = 1 to $N$
07.           For $j$ = 1 to $N$
08.               If $j == i$, $e_{ij} = 0$;
09.               If $j < i$, $a = j; b = i$;
10.               If $j > i$, $a = i; b = j$;
11.                   $e_{ij} = p_{[a*N+b-(a+1)*(a+2)/2]}$;
12.               if $e_{ij} = 1$, calculate $c_i \setminus c_j$;
13.           Next $j$
14.           calculate $f = f + \left|\bigcup_{i=1}^{N}(c_i \setminus c_j) \cap \epsilon_{ij}\right|$;
15.       Next $i$
16.   Next $s$
17.   $\vec{p}^* = argmin_{i=1}^{n}(f(\vec{p}^*(t-1)), f(\vec{p}_1(t)),$
17.        $f(\vec{p}_2(t)), \cdots, f(\vec{p}_i(t)), \cdots, f(\vec{p}_n(t)))$;
18.   For $s$ = 1 to $n$
19.       $\vec{p}_i^{\#}(t) = argmin_{i=1}^{n}(f(\vec{p}_i^{\#}(t-1)), f(\vec{p}_i(t))$;
20.       For $d$ = 1 to $D$
21.           Update the $d$-th dimension value of $\vec{p}_i$ and $\vec{v}_i$
22.             according to Eqs.(5) and (6);
23.       Next $d$
24.   Next $s$
25.End While.
```

4. Algorithm performance demonstration

To illustrate the effectiveness and performance of the particle swarm optimization algorithm, we illustrate an execution trace of the algorithm for the NS problem. A file of size 7 MB is divided into 14 pieces (512 KB each) to distribute, 6 peers download from the P2P networks, and the connecting maximum number of each peer is 3, which is represented as $(6, 14, 3)$ problem. In some session, the state of distributed file pieces is as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 4 & 0 & 6 & 7 & 8 & 0 & 10 & 0 & 12 & 0 & 14 \\ 0 & 0 & 0 & 4 & 5 & 0 & 7 & 0 & 9 & 0 & 11 & 0 & 13 & 0 \\ 0 & 2 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 11 & 12 & 0 & 14 \\ 0 & 2 & 3 & 4 & 0 & 6 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 7 & 8 & 0 & 10 & 0 & 12 & 0 & 14 \\ 1 & 2 & 0 & 0 & 5 & 0 & 0 & 0 & 9 & 10 & 11 & 0 & 13 & 14 \end{bmatrix}$$

The optimal result search by the proposed algorithm is 31, and the neighbor selection solution is illustrated below:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

We test other three representative instances (problem (25,1400,12), problem (30,1400,15) and problem (35,1400,17)) further. In our experiments, the algorithms used for comparison were GA (Genetic Algorithm) and PSO (Particle Swarm Optimization). The GA and PSO algorithms share many similarities [11].

In a GA, a population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying reproduction operators (mutation and crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation.

The PSO/GA algorithms were repeated 4 times with different random seeds. Each trial had a fixed number of 50 or 80 iterations. Other specific parameter settings of the algorithms are described in Table 1. The average fitness values of the best solutions throughout the optimization run were recorded. The average and the standard deviation were calculated from the 4 different trials. Figures 1, 2 and 3 illustrate the PSO/GA performance during the search processes for the NS problem. As evident, PSO obtained better results much faster than GA, especially for large scale problems.

5. Conclusions

In this paper, we investigated the problem of neighbor selection in peer-to-peer networks using a particle swarm optimization approach. In the proposed approach, the particle encodes the upper half matrix of the peer connection through the undirected graph, which reduces the dimension of the search space. We evaluated the performance of the proposed PSO algorithm with GA. The results indicate that PSO usually required shorter time to obtain better results

Table 1. Parameter settings for the algorithms.

Algorithm	Parameter name	value
	size of the population	$int(10 + 2sqrt(D))$
GA	Probability of crossover	0.8
	Probability of mutation	0.08
	Swarm size	$int(10 + 2sqrt(D))$
	Self coefficient c_1	2
PSO	Social coefficient c_2	2
	Inertia weight w	0.9
	Clamping Coefficient ρ	0.5

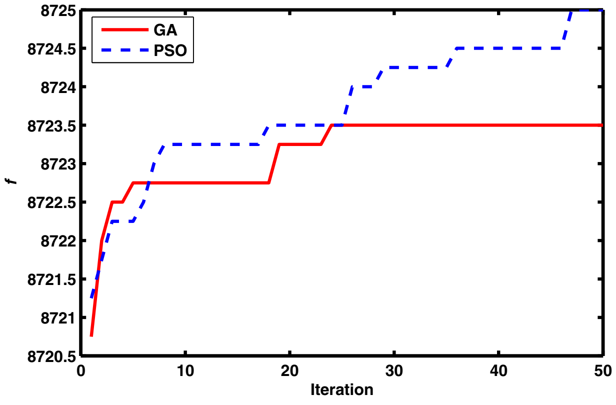

Figure 1. Performance for the NS $(25, 1400, 12)$

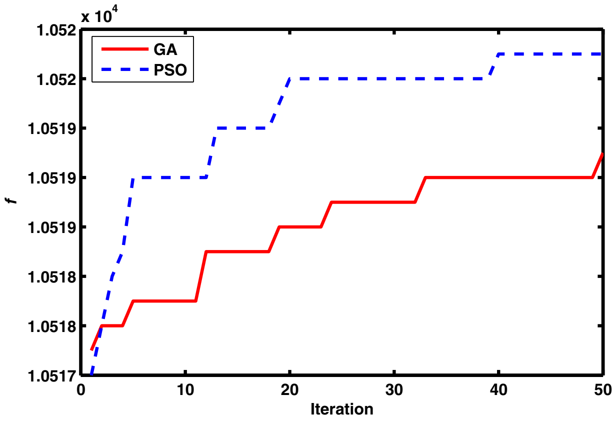

Figure 2. Performance for the NS $(30, 1400, 15)$

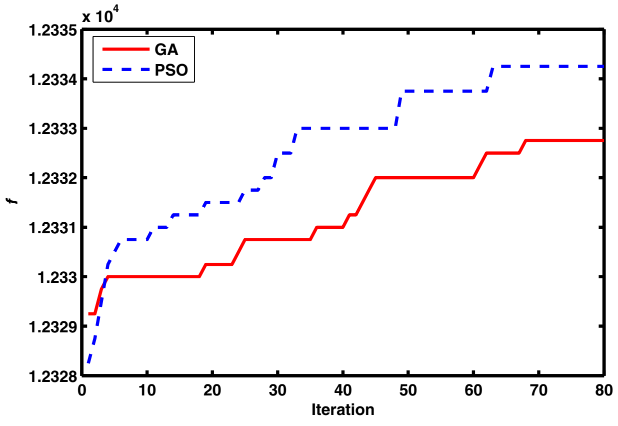

Figure 3. Performance for the NS $(35, 1400, 17)$

than GA, specially for large scale problems. The proposed algorithm could be an ideal approach for solving the NS problem.

Our future work is targeted to test more complicated instances in an online environment of P2P networks and involve more intelligent/heuristics approaches.

Acknowledgments

This work was supported by NSFC (60573124) and MOST (National Strategic Basic Research '973' Program: 2005CB321904). Ajith Abraham is supported by the Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence, appointed by The Research Council of Norway, and funded by the Research Council, NTNU and UNINETT.

References

[1] S. Kwok. "P2P searching trends: 2002-2004". *Information Processing and Management* 2006, 42, 237–247.

[2] T. Idris and J. Altmann. "A Market-managed topology formation algorithm for peer-to-peer file sharing networks". *Lecture Notes in Computer Science* 4033, 2006, pp. 61–77.

[3] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. "Load balancing in dynamic structured peer-to-peer systems". *Performance Evaluation* 2006, 63, pp. 217–240.

[4] R. Kurmanowytsch, E. Kirda, C. Kerer, and S. Dustdar. "OMNIX: A topology-independent P2P middleware". *Proceedings of The 15th Conference on Advanced Information Systems Engineering* (CAiSE'03), 2003.

[5] T. Koulouris, R. Henjes, K. Tutschku, and H. de Meer. "Implementation of adaptive control for P2P overlays". *Lecture Notes in Computer Science*, 2982, 2004, pp. 292–306.

[6] S.G.M. Koo, K. Kannan, C.S.G. Lee. "On neighbor-selection strategy in hybrid peer-to-peer networks". *Future Generation Computer Systems*, 2006, 22, pp. 732–741.

[7] J. Kennedy, and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, CA, 2001.

[8] M. Clerc. *Particle Swarm Optimization*. ISTE Publishing Company, London, 2006.

[9] A. Abraham, H. Guo, and H. Liu. “Swarm intelligence: foundations, perspectives and applications”. *Swarm Intelligent Systems, Studies in Computational Intelligence*, N. Nedjah, L. Mourelle (eds.), Springer Verlag, 2006, pp. 3–25.

[10] H. Liu, S. Sun, and A. Abraham. “Particle swarm approach to scheduling work-flow applications in distributed data-intensive computing environments”. *Proceedings of The Sixth International Conference on Intelligent Systems Design and Applications*, (ISDA’06), IEEE Computer Society Press, 2006, pp. 661–666.

[11] A. Abraham, Evolutionary Computation, Handbook for Measurement Systems Design, Peter Sydenham and Richard Thorn (Eds.), John Wiley and Sons Ltd., London, ISBN 0-470-02143-8, (2005) pp. 920-931.

[12] H. Duan, X. Lu, H. Tang, X. Zhou, Z. Zhao, Proximity Neighbor Selection in Structured P2P Network, Sixth IEEE International Conference on Computer and Information Technology (CIT’06), 2006, p. 52.

[13] S.G.M. Koo, K. Kannan, C.S.G. Lee, A genetic-algorithm-based neighbor-selection strategy for hybrid peer-to-peer networks, in: Proceedings of the 13th IEEE International Conference on Computer Communications and Networks, ICCCN04, Chicago, IL, October 2004, pp. 469474.

[14] R. Schollmeier, A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications, in: Proc. the First International August Conference on Peer-to-Peer Computing, P2P 01, Lingkoping, Sweden, August 2001, pp. 101102.

[15] D. Ghosal, B.K. Poon and K. Kong, P2P contracts: a framework for resource and service exchange, Future Generation Computer Systems 21 (2005), pp. 333347.