12

Peer-to-Peer Neighbor Selection Using Single and Multi-objective Population-Based Meta-heuristics

Hongbo Liu[1,2], Ajith Abraham[3], and Fatos Xhafa

[1] School of Computer Science and Engineering, Dalian Maritime University, 116026 Dalian, China
[2] Department of Computer, Dalian University of Technology, 116023 Dalian, China
lhb@dlut.edu.cn
[3] Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
ajith.abraham@ieee.org
http://www.softcomputing.net
[4] Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
C/Jordi Girona 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.edu

Summary. Peer-to-peer (P2P) topology has significant influence on the performance, search efficiency and functionality, and scalability of the application. In this Chapter, we introduce the problem of neighbor selection in peer-to-peer networks using two population based meta-heuristics: Particle Swarm Optimization (PSO) algorithms and Genetic Algorithms (GAs). Both a single objective and a multi-objective problem are formulated, and then the P2P neighbor selection problem is defined. We present the neighbor selection strategy based on PSO and GA algorithm. Each particle encodes the upper half of the peer-connection matrix through the undirected graph, which reduces the search space dimension. We also discuss the characteristics of ergodicity during particle swarm searching process. We also illustrate the algorithm performance and trace its feasibility and effectiveness with the help of some examples.

Keywords: P2P computing, Neighbor selection, Multi-objective optimization, Population-based meta-heuristics, Genetic Algorithms, Particle Swarm Optimization.

12.1 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].

F. Xhafa, A. Abraham (Eds.): Meta. for Sched. in Distri. Comp. Envi., SCI 146, pp. 323–340, 2008.
springerlink.com

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, 4, 5, 6, 7].

Kurmanowytsch et al. [8] developed the P2P middleware systems to provide an abstraction between the P2P topology and the applications that are built on top of it. 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. [9] presented a framework and an implementation technique for a flexible management of peer-to-peer overlays. 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.

Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection. GAs have been widely studied, experimented and applied in many fields in engineering worlds. Finding optimal parameters for many real world problems prove difficult for traditional methods but is suitable for GAs [10]. PSO (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 [11, 12]. 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 [13, 14]. In this chapter, we introduce the P2P neighbor-selection problem based GA and PSO for P2P networks.

This chapter is organized as follows. We formulate the problem in Section 12.2. The considered approaches based on GAs and PSO algorithms are presented in Section 12.3. In Section 12.4, experiment results and discussions are provided in detail, followed by some conclusions in Section 12.5.

12.2 Neighbor-Selection Problem in P2P Networks

Based on existing research [15, 16, 17, 18, 19, 20], we formulate the neighbor-selection problem for P2P networks. We introduce first the model of P2P networks, and then discuss metrics for measuring neighbor selection.

12.2.1 Modelling P2P Networks

P2P networks can be modelled by an undirected graph $G = (V, E)$ where the vertex set V represents units such as hosts and routers, and the edge set E represents physical links connecting pairs of communicating units. Further, $f : V \to \{1, \cdots, n\}$ is a labelling of its nodes, where $n = |V|$. For instance, G could model the whole or part of the Internet. Given an undirected graph $G = (V, E)$ modelling an interconnection network, and a subset $X \subseteq V(G)$ of communicating units (peers), we can construct a corresponding weighted graph $D = (V, E)$, where $V(D) = X$, and the weight of each $uv \in E(D)$ is equal to the length of a shortest path between peer u and peer v in G. Usually we start with a physical network G (perhaps representing the Internet), and then choose a set of communicating peers X. The resulting distance graph D is the basis for constructing a P2P graph $H = (V, E)$, which is done as follows. The vertex set $V(H)$ will be the same as $V(D)$, and edge set $E(H) \subseteq D(G)$. The key issue here is how to select $E(H)$. If $E = [e_{ij}]_{n\times n}$ is such that $e_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise, i.e., E is the incidence matrix of G, then the neighbor-selection problem is to find a permutation of rows and columns which brings all non-zero elements of E into the optimal possible interconnection around the diagonal.

12.2.2 Metrics

In P2P networks, specially for file sharing, an interested file is divided into many fragments. The size of each fragment ranges from several hundred kilobytes to several megabytes. When a new peer joins the network, it begins to download fragments from other peers. As long as it obtains one fragment of the file, the new peer can start to serve other peers by uploading fragments. Since peers are downloading and uploading at the same time, when the network becomes large, although the demands increase, the service provided by the network also increases [21]. Given N peers, a graph $G = (V, E)$ can be used to denote a 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 connectivity : $e_{ij} = 1$ 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 fragments, and $\{c_i \subseteq C, i = 1, \cdots, N\}$ denotes the collection of the content fragments each peer i shares. The disjointness of contents from peer i to peer j is denoted by $c_i \setminus c_j$, which can be calculated as:

$$c_i \setminus c_j = c_i - (c_i \cap c_j). \tag{12.1}$$

This disjointness can be interpreted as the collection of content fragments that peer i has but peer j does not. In other words, it denotes the fragments that peer i can upload to peer j. Note that the disjointness operation is not commutative, *i.e.*, $c_i \setminus c_j \neq c_j \setminus c_i$. Let $|c_i \setminus c_j|$ denote the cardinality of $c_i \setminus c_j$, which is the number of content fragments peer i can contribute to peer j. In order to maximize the disjointness of content, we maximize the number of content fragments each peer can contribute to its neighbors by determining the connections e_{ij}'s. Let

us 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:

$$f(x) = \max_{E} \sum_{j=1}^{N} \Big| \bigcup_{i=1}^{N} (c_i \setminus c_j) \cap \epsilon_{ij} \Big| \tag{12.2}$$

It is desirable to select peers with the most mutually disjoint collection of content fragments as neighbors. However, downloading the file fragments between each peer pair would consume a lot of bandwidth and connection cost, etc. Let τ_{ij} denote the cost coefficient between peers i and j. The performance of the whole system can be expressed as follows. The neighbor selection strategy is expected not only to assure maximum content availability but also to minimize the downloading cost to improve the overall throughput of the system. Therefore we have the following multi-objective optimization problem:

$$f_1(x) = \max_{E} \sum_{j=1}^{N} \Big| \bigcup_{i=1}^{N} (c_i \setminus c_j) \cap \epsilon_{ij} \Big| \tag{12.3}$$

$$f_2(x) = \min_{E} \sum_{j=1}^{N} \sum_{i=1}^{N} \tau_{ij} |(c_i \setminus c_j)||\epsilon_{ij}| \tag{12.4}$$

In the network, every node is a potential neighbor of each other node since the network's topology is a logical one. So the full connection is an ideal solution for the peer's connectivity. For the networks, however, we have to consider some constraints [3, 20]:

- based on the underlying network characteristics, i.e., delay or capacity of actual links;
- based on location of data and services;
- based on the nodes's capabilities of managing peers, e.g., the number of direct neighbors a node can maintain; some peers are tied down since they could possess relatively more content fragments. Note that this resource constraint can be independent of the underlying network.

In the environment, the maximum number of each peer needs to be considered, i.e., each peer i will be connected to a maximum of d_i neighbors, where $d_i < N$. Therefore there are two constraints for each peer:

$$\begin{aligned} \sum_{j=1}^{N} e_{ij} &\leq d_i, \quad \text{for all } i \\ \sum_{i=1}^{N} e_{ij} &\leq d_j, \quad \text{for all } j \end{aligned} \tag{12.5}$$

Definition 1. *A neighbor-selection problem in P2P networks problem can be defined as* $\prod = (N, C, M, F, s)$*, in which* N *is the number of peers,* C *is the entire*

collection of content fragments, M is the maximum number of the peers, which each peer can connect steadily in the session, F is a single objective to optimize the number of swap fragments, or multi-objective to optimize the number of swap fragments, and to minimize the downloading cost; s denotes the environment constraints. The key components are operations, machines and data-hosts. A P2P state is determined by N, C and M, i.e. $S = (N, C, M)$. For the sake of simplify, the neighbor-selection problem can be also represented in triple $\prod = (S, F, s)$.

12.3 P2P Neighbor-Selection Strategy

GA and PSO algorithms share many similarities [22]. In 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 to the next generation. PSO algorithm is inspired by social behavior patterns of organisms that live and interact within large groups. It incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior. In this section, we discuss P2P neighbor selection strategy based on PSO and GA algorithms.

12.3.1 Particle Swarm Algorithm for Single Objective Neighbor Selection

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 swap fragments in the potential solution. Each particle has a position represented by a position-vector $\mathbf{p}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\mathbf{v}_i$. Each particle remembers its own best position so far in a vector $\mathbf{p}_i^{\#}$, and its j-th dimensional value is $p_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\mathbf{p}^*$, 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:

$$P(p_{ij}(t)=1)=f(p_{ij}(t-1),v_{ij}(t-1),p_{ij}^{\#}(t-1),p_{j}^{*}(t-1)), \tag{12.6}$$

where the probability function is

$$sig(v_{ij}(t))=\frac{1}{1+e^{-v_{ij}(t)}}. \tag{12.7}$$

At each time step, each particle updates its velocity and moves to a new position according to Eqs. (12.8) and (12.9):

$$\begin{aligned} v_{ij}(t)=wv_{ij}(t-1)+c_1r_1(p_{ij}^{\#}(t-1)-p_{ij}(t-1)) \\ +c_2r_2(p_{j}^{*}(t-1)-p_{ij}(t-1)) \end{aligned} \tag{12.8}$$

$$p_{ij}(t)=\begin{cases}1 & \text{if } \rho<sig(v_{ij}(t));\\ 0 & \text{otherwise.}\end{cases} \tag{12.9}$$

where c_1 is a positive constant, called coefficient of the self-recognition component, c_2 is a positive constant, called coefficient of the social component; r_1 and r_2 are random numbers in the interval [0,1]. The variable w is called as the inertia factor, whose value is typically setup to vary linearly from 1 to near 0 during the iterated processing and ρ is a random number in the closed interval [0, 1]. From Eq. (12.8), 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 12.1.

In multi-dimensional search space, the characteristics of ergodicity is of vital importance to an algorithm. We discuss them for the particle swarm optimization. Clerc and Kennedy have stripped the particle swarm model down to a most simple form [23, 24]. If the self-recognition component c_1 and the coefficient of the social-recognition component c_2 in the particle swarm model are combined into a single term c, *i.e.* $c=c_1+c_2$, the best position $\mathbf{p}_i$ can be redefined as follows:

$$\mathbf{p}_i \leftarrow \frac{(c_1\mathbf{p}_i+c_2\mathbf{p}_g)}{(c_1+c_2)} \tag{12.10}$$

Then, the update of the particle's velocity is defined by:

$$\mathbf{v}_i(t)=\mathbf{v}_i(t-1)+c(\mathbf{p}_i-\mathbf{x}_i(t-1)) \tag{12.11}$$

Algorithm 12.1. 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 stopping 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. Calculate $f = f + \left| \bigcup_{i=1}^{N} (c_i \setminus c_j) \cap \epsilon_{ij} \right|$;
14. $\mathbf{p}^* = argmin_{i=1}^{n}(f(\mathbf{p}^*(t-1)), f(\mathbf{p}_1(t)),$
15. $f(\mathbf{p}_2(t)), \cdots, f(\mathbf{p}_i(t)), \cdots, f(\mathbf{p}_n(t)))$;
16. For $s = 1$ to n
17. $\mathbf{p}_i^{\#}(t) = argmin_{i=1}^{n}(f(\mathbf{p}_i^{\#}(t-1)), f(\mathbf{p}_i(t))$;
18. For $d = 1$ to D
19. Update the d-th dimension value of $\mathbf{p}_i$ and $\mathbf{v}_i$
20. according to Eqs. (12.8) and (12.9);
21. End While

The system can be simplified even further by using $\mathbf{y}_i(t-1)$ instead of $\mathbf{p}_i - \mathbf{x}_i(t-1)$. Thus, the reduced system is then:

$$\begin{cases} \mathbf{v}(t) = \mathbf{v}(t-1) + c\mathbf{y}(t-1) \\ \mathbf{y}(t) = -\mathbf{v}(t-1) + (1-c)\mathbf{y}(t-1) \end{cases}$$

This recurrence relation can be written as a matrix-vector product, so that

$$\begin{bmatrix} \mathbf{v}(t) \\ \mathbf{y}(t) \end{bmatrix} = \begin{bmatrix} 1 & c \\ -1 & 1-c \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}(t-1) \\ \mathbf{y}(t-1) \end{bmatrix}$$

Let

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{v}_t \\ \mathbf{y}_t \end{bmatrix}$$

and

$$A = \begin{bmatrix} 1 & c \\ -1 & 1-c \end{bmatrix}$$

we have an iterated function system for the particle swarm model:

$$\mathbf{P}_t = A \cdot \mathbf{P}_{t-1} \tag{12.12}$$

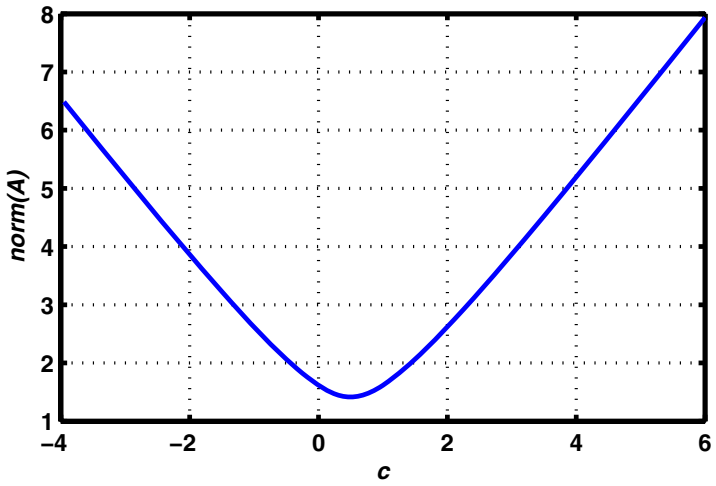

Fig. 12.1. Norm of A

Thus, the system is completely defined by A. Its norm $\|A\|_2$ (also written $\|A\|$) is determined by c. The relationship of A and its dependence on c is illustrated in Fig. 12.1.

Note that it is possible to find different trajectories of the particle for various values of c. Fig. 12.2(a) illustrates the system for a torus when c=2.9; Fig. 12.2(b), a hexagon with spindle sides when c=2.99; Fig. 12.2(c), a triangle with spindle sides when c=2.999; Fig. 12.2(d) and a simple triangle when c=2.9999. As depicted in Fig. 12.2, the iteration time step used is 100 for all the cases. Another system sensitivity instance is illustrated in Fig. 12.3. It is to be noted that Figs. 12.2 and 12.3 illustrate only some 2-dimensional representations of the iterated process. A comparison between 2D and 3D is illustrated in Fig. 12.4.

12.3.2 Genetic Algorithm for Multi-objective Neighbor Selection

Multi-objective GAs are very popular multi-objective techniques, which normally exhibit good overall performance. Many multi-objective optimization techniques using evolutionary algorithms have been proposed in recent years [22, 25, 26]. Given a P2P state S, the multi-objective neighbor selection is not only to maximize Eq. (12.3) but also to minimize Eq. (12.4) with the constraint in Eq. (12.5).

Similarly, we adopt the upper-half-triangle encoding representation in our genetic algorithm for the NS problem. We also set up a search space of D dimension as $N*(N-1)/2$. Accordingly, each individual is represented as a binary bit string of length D. The pseudo-code for our P2P neighbor selection method is illustrated in Algorithm 12.2.

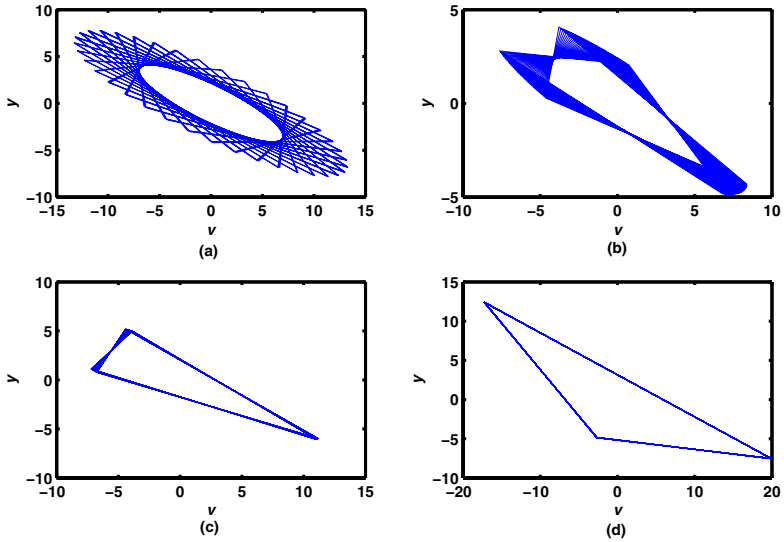

Fig. 12.2. Trajectory of the particle (a) $c = 2.9$, (b) $c = 2.999$, (c) $c = 2.999$, (d) $c = 2.9999$

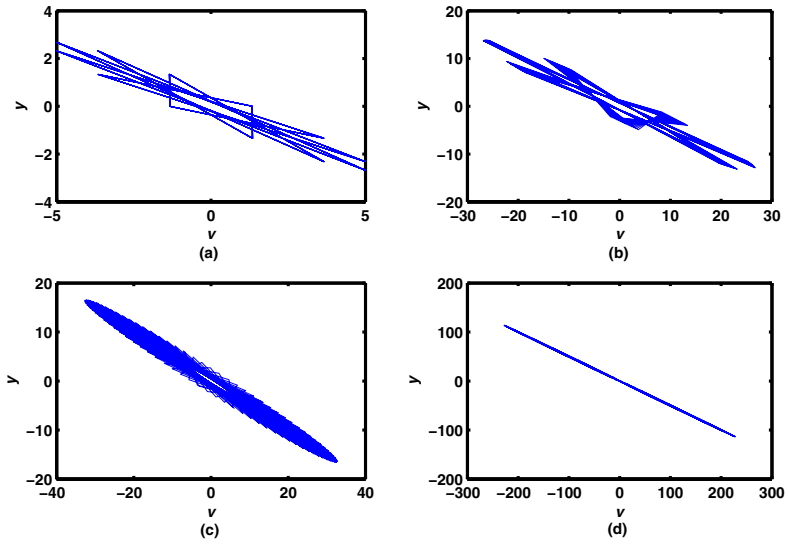

Fig. 12.3. Trajectory of the particle (a) $c = 3.7321$, (b) $c = 3.8$, (c) $c = 3.9$, (d) $c = 3.999$

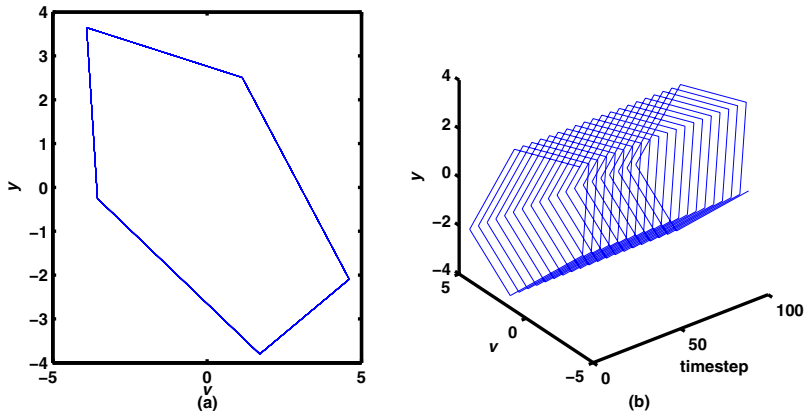

Fig. 12.4. 2D versus 3D (a) 2D: $c = 1.3820$, (b) 3D: $c = 1.3820$

Algorithm 12.2. Neighbor Selection Algorithm Based on Genetic Algorithm

01. Initialize the population, and other parameters.
02. While (the stopping criterion is not met) do
03. Evaluate();
04. for $i = 1$ to N
05. for $j = 1$ to N
06. if $j == i$, $e_{ij} = 0$;
07. else if $j < i$, $a = j; b = i$;
08. else if $j > i$, $a = i; b = j$;
09. $e_{ij} = p_{[a*N+b-(a+1)*(a+2)/2]}$;
10. If $e_{ij} = 1$, calculate $c_i \setminus c_j$;
11. Calculate $f_2 = f_2 + \tau_{ij}|(c_i \setminus c_j)|$;
12. Next j
13. calculate $f_1 = f_1 + \left| \bigcup_{i=1}^{N} (c_i \setminus c_j) \cap \epsilon_{ij} \right|$;
14. Rank();
15. If nondomCtr> MaxArchiveSize, maintenance-archive();
16. Generate-new-pop();
17. Crossover();
18. Mutation();
19. $t++$;
20. If rank == 1 output the fitness;
21. End While

12.4 Algorithm Performance Evaluation

To illustrate the effectiveness and performance of the considered algorithms, we illustrate the neighbor-selection process and results through some test problems. The specific parameter settings of the algorithms are described in Table 12.1.

Table 12.1. Parameter settings for the algorithms

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

12.4.1 Single Objective Neighbor Selection

We first illustrate an execution trace of the algorithm for the NS problem. A file of size 7 MB is divided into 14 fragments (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 fragments is as follows:

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

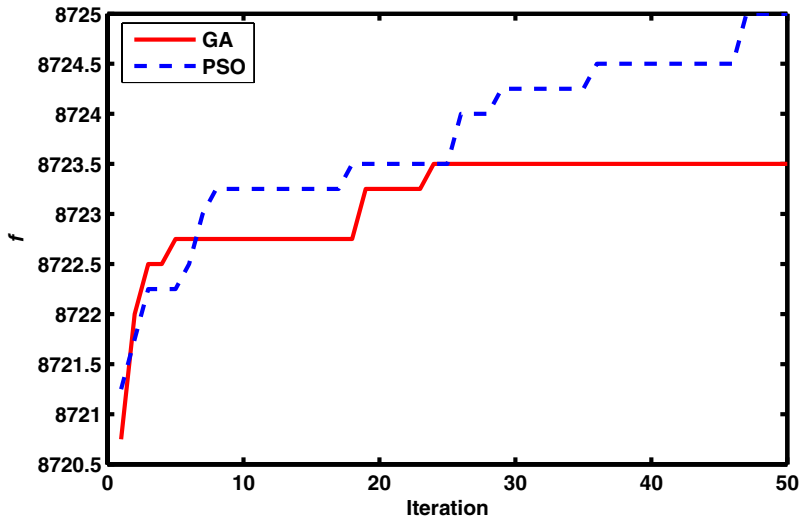

Fig. 12.5. Performance for the NS $(25, 1400, 12)$

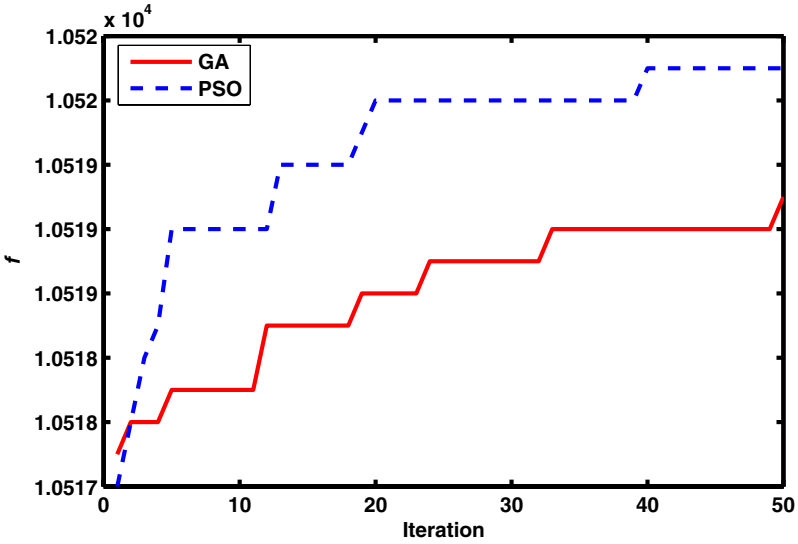

Fig. 12.6. Performance for the NS $(30, 1400, 15)$

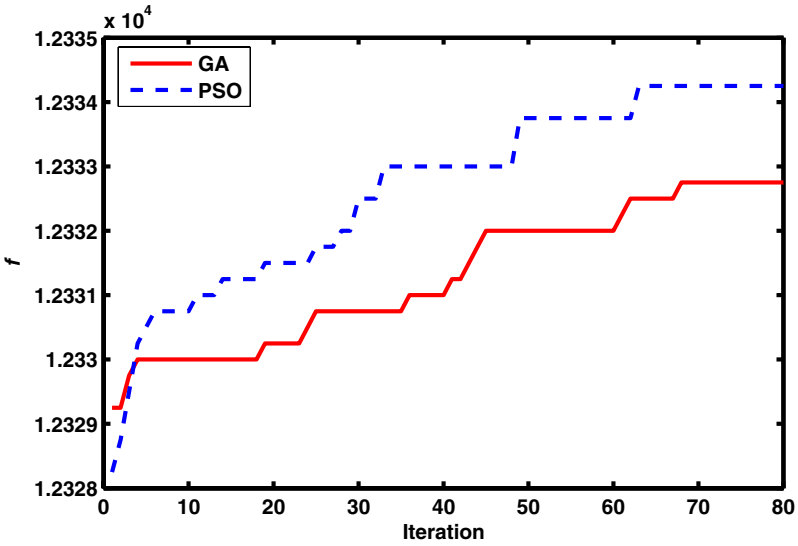

Fig. 12.7. Performance for the NS $(35, 1400, 17)$

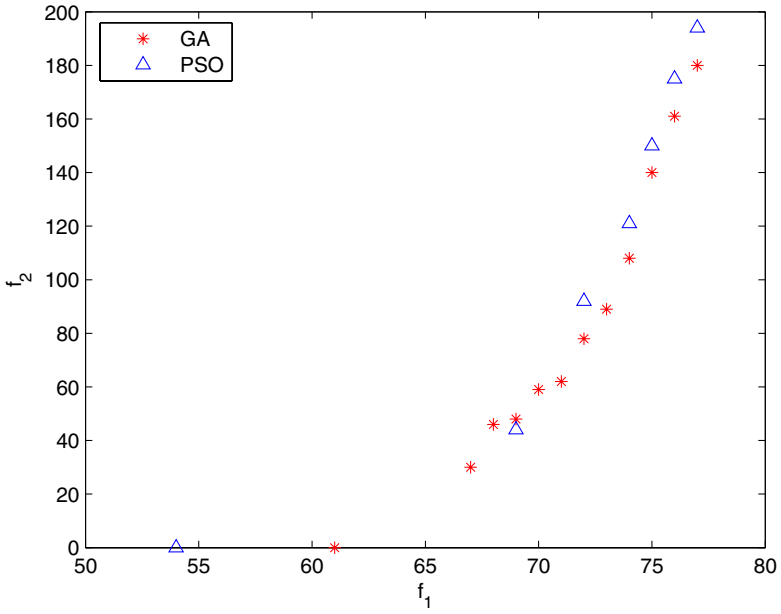

Fig. 12.8. Performance for the NS $(6, 60, 3)$

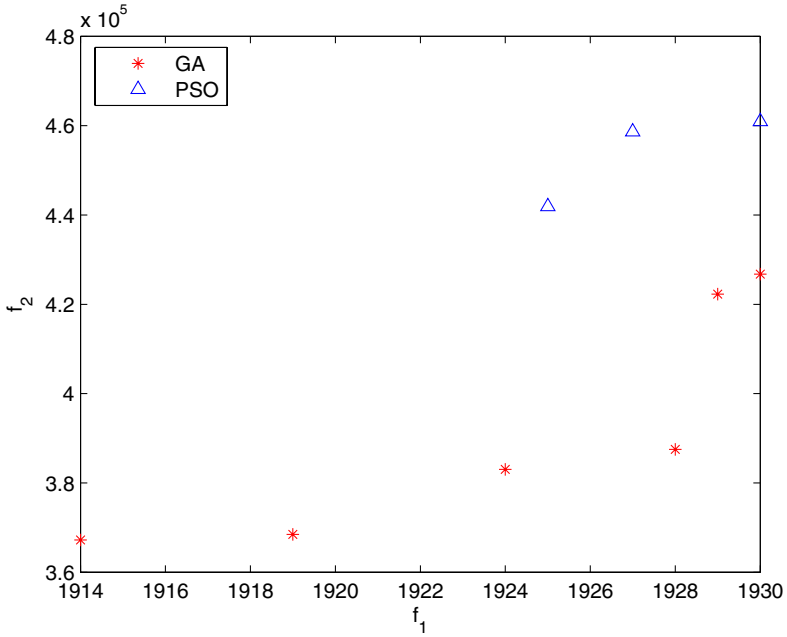

Fig. 12.9. Performance for the NS $(25, 300, 12)$

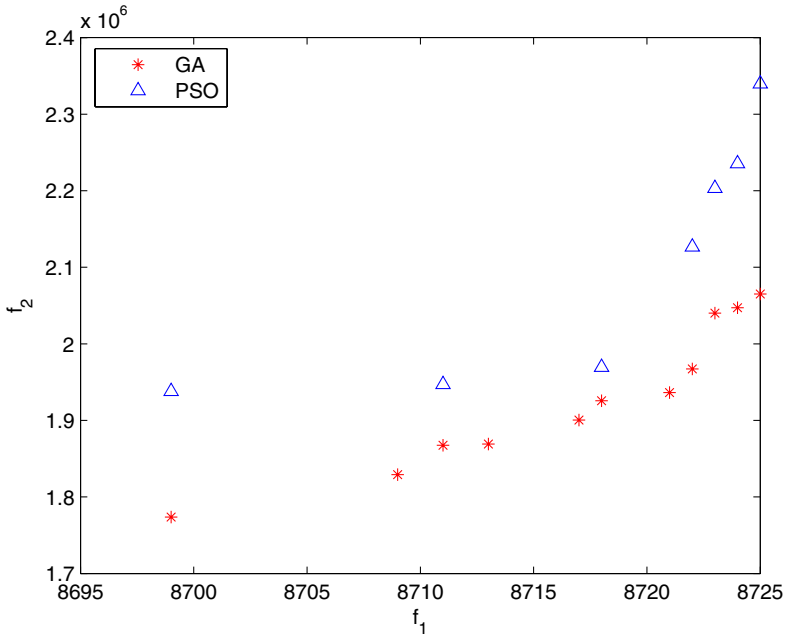

Fig. 12.10. Performance for the NS $(25, 1400, 12)$

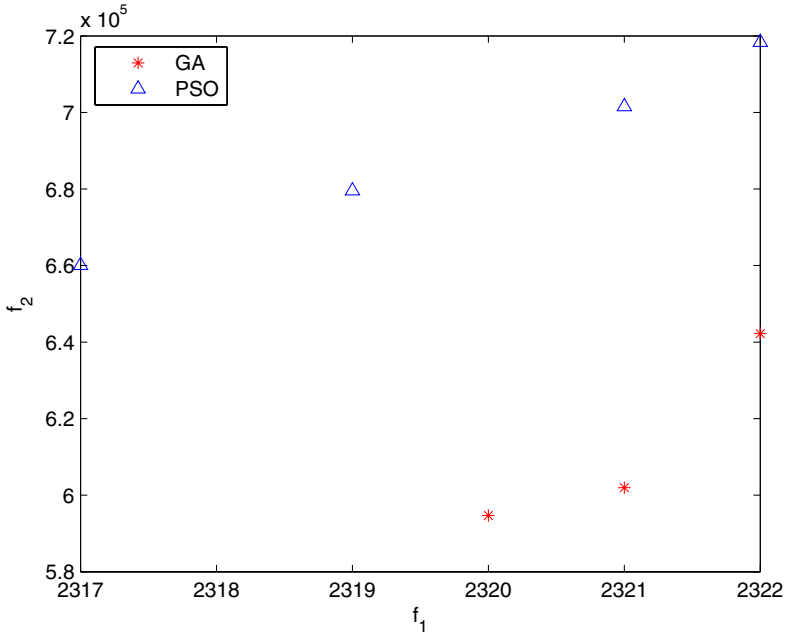

Fig. 12.11. Performance for the NS $(30, 300, 15)$

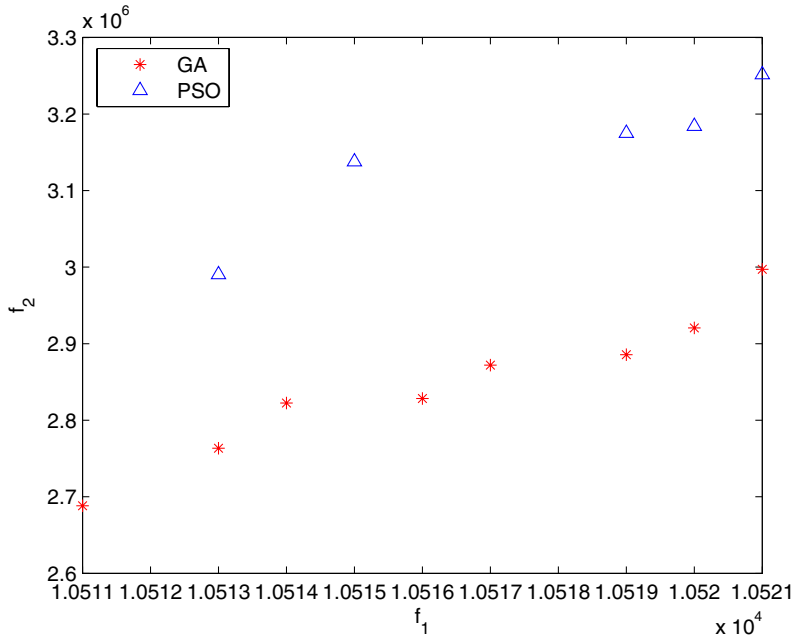

Fig. 12.12. Performance for the NS $(30, 1400, 15)$

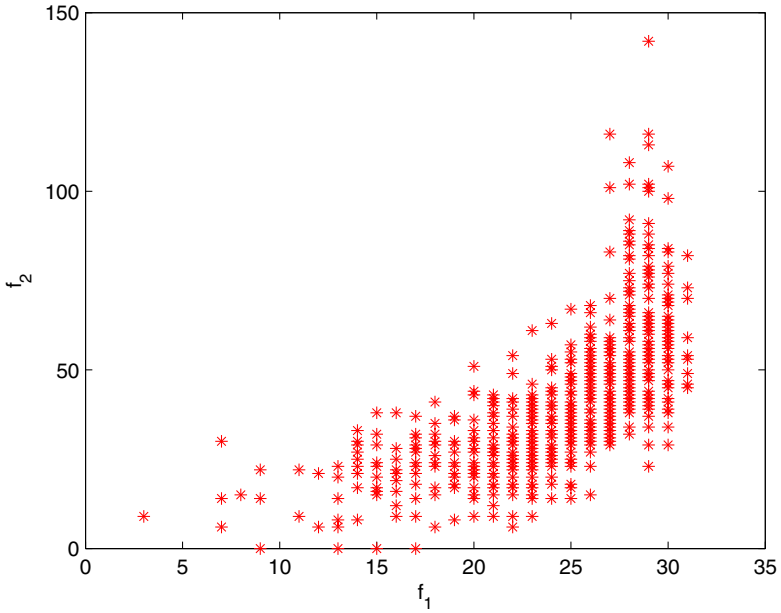

Fig. 12.13. Performance for the NS $(6, 60, 3)$

The optimal result search by the proposed algorithm is 31, and the neighbor selection solution is shown in the matrix 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 also tested three other representative instances (problem (25,1400,12), problem (30,1400,15) and problem (35,1400,17)). In our experiments, the algorithms used for comparison were GA and PSO.

The PSO/GA algorithms were repeated 4 times with different random seeds. Each trial had a fixed number of 50 / 80 iterations. Other specific parameter settings of the algorithms are described in Table 12.1. 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. Figs. 12.5, 12.6 and 12.7 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.

12.4.2 Multi-objective Neighbor Selection

We demonstrate an execution trace of the algorithm for the first NS problem in last subsection, i.e., $(6, 14, 3)$ problem. In this problem, the network cost is considered; the corresponding cost matrix is as follows:

$$\begin{bmatrix} 0 & 5 & 2 & 4 & 1 & 0 \\ 5 & 0 & 3 & 0 & 2 & 2 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 5 & 2 \\ 1 & 2 & 0 & 5 & 0 & 10 \\ 0 & 2 & 0 & 2 & 10 & 0 \end{bmatrix}$$

The PSO/GA algorithms were repeated 3 times with different random seeds. Each trial had a fixed number of 200 iterations. The average fitness values of the best (rank = 1) solutions throughout the optimization run were recorded. The performance output is illustrated in Fig. 12.13 by the proposed algorithm. We also tested other five representative instances (problem (6,60,3), problem (25,300,12), problem (25,1400,12), problem (30,300,15), problem (30,1400,15)) further. Figs. 12.8, 12.9, 12.10, 12.11 and 12.12 illustrate the GA/PSO performance during the search processes for the NS problem. As evident, GA usually obtained better results than PSO.

12.5 Conclusions

In this chapter, we introduced the problem of neighbor selection in peer-to-peer networks using a Particle Swarm Optimization and Genetic Algorithms. We first introduced the model of Peer-to-Peer networks and discussed measuring metrics for P2P neighbor selection. Both a single and a multi-objective formulations are given, and then the P2P neighbor selection problem is defined. In the considered approaches, we presented an upper-half-triangle encoding representation method. The particle/individual was encoded by 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 algorithms. The results indicate that PSO usually required shorter time than GA, specially for large scale problems. PSO could be an ideal approach for solving the single objective NS problem, while GA usually obtain better results than PSO in the multi-objective NS problems.

Acknowledgements

This work was partly supported by NSFC (60373095), DLMU (DLMU-ZL-200709). F. Xhafa acknowledges partial support by Projects ASCE TIN2005-09198-C02-02, FP6-2004-ISO-FETPI (AEOLUS) and MEC TIN2005-25859-E and FORMALISM TIN2007-66523.

References

1. Kwok, S.: P2P searching trends: 2002-2004. Information Processing and Management 42, 237–247 (2006)
2. Idris, T., Altmann, J.: A Market-managed topology formation algorithm for peer-to-peer file sharing networks. In: Stiller, B., Reichl, P., Tuffin, B. (eds.) ICQT 2006. LNCS, vol. 4033, pp. 61–77. Springer, Heidelberg (2006)
3. Surana, S., Godfrey, B., Lakshminarayanan, K., Karp, R., Stoica, I.: Load balancing in dynamic structured peer-to-peer systems. Performance Evaluation 63, 217–240 (2006)
4. Duan, H., Lu, X., Tang, H., Zhou, X., Zhao, Z.: Proximity neighbor selection in structured P2P network. In: Proceedings of Sixth IEEE International Conference on Computer and Information Technology, p. 52 (2006)
5. Koo, S., Kannan, K., Lee, C.: 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, pp. 469–474 (2004)
6. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: Proceedings of the First International August Conference on Peer-to-Peer Computing, pp. 101–102 (2001)
7. Ghosal, D., Poon, B.K., Kong, K.: P2P contracts: a framework for resource and service exchange. Future Generation Computer Systems 21, 333–347 (2005)
8. Kurmanowytsch, R., Kirda, E., Kerer, C., Dustdar, S.: OMNIX: A topology-independent P2P middleware. In: Proceedings of the 15th Conference on Advanced Information Systems Engineering (2003)

9. Koulouris, T., Henjes, R., Tutschku, K., de Meer, H.: Implementation of adaptive control for P2P overlays. In: Wakamiya, N., Solarski, M., Sterbenz, J.P.G. (eds.) IWAN 2003. LNCS, vol. 2982, pp. 292–306. Springer, Heidelberg (2004)
10. Quagliarella, D., Périaux, J., Poloni, C., Winter, G. (eds.): Genetic Algorithms in Engineering and Computer Science. John Wiley & Sons Ltd., Chichester (1997)
11. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
12. Clerc, M.: Particle Swarm Optimization. ISTE Publishing Company, London (2006)
13. Abraham, A., Guo, H., Liu, H.: Swarm intelligence: foundations, perspectives and applications. In: Nedjah, N., Mourelle, L. (eds.) Swarm Intelligent Systems. Studies in Computational Intelligence, pp. 3–25. Springer, Heidelberg (2006)
14. Liu, H., Sun, S., Abraham: A Particle swarm approach to scheduling work-flow applications in distributed data-intensive computing environments. In: Proceedings of The Sixth International Conference on Intelligent Systems Design and Applications, pp. 661–666 (2006)
15. Sen, S., Wang, J.: Analyzing Peer-to-Peer Traffic Across Large Networks. IEEE/ACM Transactions on Networking 12(2), 219–232 (2004)
16. Liu, Y., Xiao, L., Esfahanian, A., Ni, L.M.: Approaching Optimal Peer-to-Peer Overlays. In: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 407–414 (2005)
17. Belmonte, M.V., Conejo, R., Díaz, M., Pérez-de-la-Cruz, J.L.: Coalition Formation in P2P File Sharing Systems. In: Marín, R., Onaindía, E., Bugarín, A., Santos, J. (eds.) CAEPIA 2005. LNCS (LNAI), vol. 4177, pp. 153–162. Springer, Heidelberg (2006)
18. Koo, S., Kannan, K., Lee, C.: On neighbor-selection strategy in hybrid peer-to-peer networks. Future Generation Computer Systems 22, 732–741 (2006)
19. Ghanea-Hercock, R.A., Wang, F., Sun, Y.: Self-Organizing and Adaptive Peer-to-Peer Network. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 36(6), 1230–1236 (2006)
20. Wang, S., Chou, H., Wei, D., Kuo, S.: On the Fundamental Performance Limits of Peer-to-Peer Data Replication in Wireless Ad hoc Networks. IEEE Journal on Selected Areas in Communications 25(1), 211–221 (2007)
21. Qiu, D., Sang, W.: Global Stability of Peer-to-Peer File Sharing Systems. Computer Communications (2007) doi:10.1016/j.comcom.2007.08.012
22. Abraham, A.: Evolutionary computation. In: Sydenham, P., Thorn, R. (eds.) Handbook for Measurement Systems Design, pp. 920–931. John Wiley and Sons Ltd., London (2005)
23. Clerc, M., Kennedy, J.: The Particle Swarm-explosion, Stability, and Convergence in A Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
24. Liu, H., Abraham, A., Clerc, M.: Chaotic Dynamic Characteristics in Swarm Intelligence. Applied Soft Computing 7, 1019–1026 (2007)
25. Abraham, A., Jain, L.: Evolutionary Multi-objective Optimization. In: Abraham, A., Jain, L.C., Goldberg, R. (eds.) Evolutionary Multi-objective Optimization: Theoretical Advances and Applications, ch. 1, pp. 1–9. Springer, London (2005)
26. Srinivas, N., Deb, K.: Multi-objective optimization using nondominated sorting genetic algorithms. Evolutionary Computation 2(3), 221–248 (1994)