Peer-to-Peer Networking and Applications (2019) 12:129–142
https://doi.org/10.1007/s12083-018-0644-2

Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm

Saied Asghari[1] · Nima Jafari Navimipour[1]

Received: 7 September 2016 / Accepted: 2 March 2018 / Published online: 15 March 2018

Abstract
In recent years, the attractiveness of Peer-to-Peer (P2P) networks has been grown rapidly due to the easiness of use. The P2P system is a decentralized relationship model in which every party has the analogous abilities and either party can start a relationship session. In these networks, due to the high number of users, the resource discovery process becomes one of the important parts of the P2P networks. But, in many previously proposed methods, there is a common problem that is called load balancing. If the balance of workload is inefficient, it reduces the resource utilization. Therefore, in this article, we propose the Inverted Ant Colony Optimization (IACO) algorithm, a variety of the basic Ant Colony Optimization (ACO) algorithm, to improve load balancing among the peers. In the proposed method, the effect of pheromone on the selected paths by ants is inverted. In this approach, ants start to traverse the graph from the requester peer and each ant chooses the best peer for moving. Then, requirements and pheromone amount are updated. Finally, we simulate the method and evaluate its performance in comparison to the ACO algorithm in different terms. The obtained results show that the performance of the IACO is better than the ACO algorithm in terms of load balancing, waiting time and resource utilization.

Keywords Peer-to-peer · Resource discovery · Inverted ant colony optimization · Load balancing · Waiting time

1 Introduction

Along the last decades, outstanding improvements in Information and Communication Technology (ICT) have permitted the human generation to create, process, and share a growing quantity of information [1–5]. Peer-to-Peer (P2P) networks are networks of interconnected peers where some services are provided whereas others want to get them [6]. P2P networks overcome the challenges of client/server systems, including scalability, points of failure, cost, and complexity [7]. These networks are widely applied and studied in recent years [8]. In these systems, unlike the traditional client-server models, every peer can at the same time operation as a server or a client [9]. Thus, the peer is allowed to share resources directly with other peers, which makes P2P networks quite famous [10]. Resource sharing, real-time data streaming, and cycle stealing are some popular representative services presented by P2P networks [11]. Other applications of P2P networks are the sharing of storage and content distribution [12, 13].

Resource discovery is one of the most important challenges in any distributed system like grid [14, 15], cloud [16–21], P2P network [9], and social network [22–24]. It is a key service to discover the system resources through a huge-scale distributed system [25]. Also, it is one of the important and interesting components of P2P systems and becomes more difficult in such multi-hop networks [26]. It can be considered in four fundamental categories: unstructured, structured, super-peer and hybrid techniques [27]. Unstructured P2P networks usually use flooding methods to search resources [28, 29]. They are widely used over the Internet and Gnutella is one of the practical implementations of unstructured P2P [30, 31]. But, unlike unstructured networks, structured P2P systems make bandwidth utilization more efficient by imposing limits on resource location and machine topology [32]. Structured P2P networks use distributed Hashtable to increase the efficiency, but they are inflexible under a dynamic environment [33, 34]. Compared to unstructured P2P networks, structured networks generate more overheads for locating famous content. The super to peer-based networks, as a novel method to facilitate the convergence of P2P networks, include a few peers in the overlay network as super-peers to perform more powerfully on behalf of the peers [35].

✉ Nima Jafari Navimipour
jafari@iaut.ac.ir

[1] Young Researchers and Elite Club, Tabriz Branch, Islamic Azad University, Tabriz, Iran

The super-peers have much greater resource capacities than other peers. Peers are assigned to at least one super-peer and they route all their queries via this super-peer such as searching and routing [36], like [37–42]. Hybrid P2P networks combine structured and unstructured strategies to overcome some weakness while retaining benefits and advantages of each approach, like [43, 44]. Furthermore, one of the major challenges in P2P networks is load balancing among the peers, however, some methods such as ACO algorithm [6] suffer from this problem.

Therefore, in this paper, we propose an IACO algorithm to improve the loads among the peers. In the presented method, updated pheromone in each iteration has an important effect in order to improve load balancing among the peers. After applying updated pheromone, ants try to select the new paths to traverse and satisfy requester requirements. The presented method follows the classical concept of an unstructured P2P system because this system is suitable for the proposed method. Then, we compare obtained results of the proposed method in terms of load balancing and waiting time in comparison with the ACO algorithm. Briefly, the main objectives of the presented method are as follows:

- Establishing load balancing among the peers in P2P networks.
- Reducing the waiting time of the requester peer for receiving requirements.
- Evaluating the execution time and the resource utilization of the proposed method.

The remainder of this paper is organized as follows. The related work is provided in the next section. In Section 3, the proposed method is discussed. Section 4 describes experimental dataset and the simulation results. Finally, in the last section, the conclusion and future work are provided.

2 Related work

In this section, some important methods for resource discovery in P2P networks along with the advantages and disadvantages of each are depicted and discussed.

Napster[1] as a famous type of unstructured network [45] includes a central server which stores the indicator of all resources shared by the peers. To find a resource, a user queries utilizing the name of the resource and as a result, the IP address of a peer is received. Then, a direct connection between the requester peer and the peer contains the resource being downloaded. For example, when peer *A* wants to search for some resource, it contacts the central server. The server returns some peers that hold the resource, for instance, peer *B*. Then peer *A* begins to download the file from peer *B*. Napster is simple and provides low response time but it is difficult to scale the central indicator server and it has a single point of failure.

Gnutella[2] implements searches via a flooding strategy with a fixed scope to determine a way in which servants communicate over the network. It comprises a set of descriptors used to communicate data among servants and a set of rules governing the inter-servant exchange of descriptors. For resource discovery, a flooding technique is used. Peer *A* wants to search for some file. It floods its search query to its neighbors. When a peer receives the query, it checks whether it holds the matching file itself. If not, it forwards the query to its neighbors. If a peer holds the file that peer *A* wants, it returns a response to the peer that sends it the query and that query continues forwarding the response to the query sender. Finally, peer *A* contacts the peer that hold the requested file to download [46]. This method will be efficient if it produces fewer numbers of redundant queries. Gnutella is highly effective for finding famous items. This approach has an enormous delay and false negative error. Furthermore, it suffers from extreme traffic due to utilizing flooding technique.

Likewise, a strong and trust based search structure is presented by Mashayekhi and Habibi [47]. This technique keeps limited size routing indexes composing search and trust data to direct queries to most reputable peers. Each peer uses the semantic and trust information to create limited size indexes on its inks. Utilizing these indexes, the structure can guide query messages to the most famous peers that contain query search results. By dynamically choosing trustworthy peers as score managers, the plan tracks the reputation of participating peers. The presented method offers high robustness and dynamism, but it suffers from low response time and load balancing. Likewise, because of utilizing the Time To Live (TTL), the false negative error can happen.

Furthermore, Zaharia and Keshav [48] have proposed a search technique called Gossip Adaptive Hybrid (GAB) on hybrid P2P networks which are used gossip method to gather the network's statistics. It uses the concept of a Gnutella where end peers store the ultra-peers index content. It lets peers predict the best search technique for a query. Then this prediction is compared to an actual measurement of search effectiveness. Furthermore, the design of GAB does not depend on the choice of the DHT or the unstructured flooding network. It offers suitable response time and query bandwidth usage whereas it suffers from low reliability and load balancing.

Also, HybridFlood as a new technique in unstructured P2P networks has been proposed by Barjini, et al. [49]. This technique combines the merits of flooding and super peer and limiting their drawbacks. It divides flooding pattern into two phases. At the first phase, the algorithm follows flooding by a limited number of hops. In the second phase, it chooses nosey peers[3] in

[1] www.napster.com

[2] www.Gnutellaforums.com

[3] The nosey peers are peers which have the most links to others.

each searching horizon to maintain the data index of all clients. HybridFlood extends the search efficiency by reducing redundant messages in each hop. The authors have shown that HybridFlood decreases the redundant messages and saves up to 70% of searching traffic in comparison to flooding approach. But, the robustness and dynamicity of the technique are not evaluated.

Kumar, et al. [50] have proposed a mechanism for designing an efficient query routing. In this article, a query routing protocol that permits low bandwidth consumption during query forwarding utilizing a low-cost mechanism to build and keep data about nearby objects was presented. To obtain this goal, the proposed protocol maintains a lightweight probabilistic routing table at each node that offers the location of each object in the network. Following the corresponding routing table entries, a query can reach the destination in a small number of hops with high probability. A data structure called an Exponentially Decaying Bloom Filter (EDBF) was designed that encodes routing tables in a highly compressed manner and allows for the effective aggregation and propagation. The search primitives presented by the proposed system can be utilized to search for single or multiple keywords. The analytical modeling of the proposed design forecasts the improvements in search efficiency and is verified through extensive simulations. This method requires low bandwidth, low cost and offers high reliability and trust. But, due to using TTL, it suffers from false negative error and also it suffers from imbalanced load problem.

Finally, Chawathe, et al. [51] have modified the Gnutella approach to include flow control, dynamic topology adaptation, and node heterogeneity. Retaining Gnutella's simplicity was advocated while presenting new techniques that greatly boost its scalability. Several modifications were proposed to Gnutella's design that dynamically adapt the overlay topology and the search algorithms to accommodate the natural heterogeneity present in most peer-to-peer systems. The proposed design was tested through simulations and the results illustrate three to five orders of magnitude improvement in system capacity. This method retains significant robustness and scalability. It improves system download capacity, but, due to using TTL, it suffers from false negative error and also it is not suitable for file sharing.

3 Proposed method

In order to improve load balancing among the peers in P2P networks, we use an IACO algorithm. In this section, at first, the ACO algorithm is briefly introduced. In the following, the network model is described and then an IACO algorithm is proposed for resource discovery in P2P networks. Finally, a case study is described and evaluated.

3.1 ACO algorithm

ACO is a swarm intelligence technique to problem-solving which is proposed by Dorigo, et al. [52]. The core idea of ACO is to utilize a swarm of basic and stochastic automata to solve complex issues. Such a communication method has shown to provide interesting results, especially with the emphasis on finding the shortest path or paths optimizing a given function [53, 54]. The automata, or agents, in ACO, are called ants. Each ant has the simple task of locating the needed resource (search section) and bringing it back to its nest (returning phase). ACO for resource discovery in P2P networks is provided in [6]. According to our observations, in this method there is not a central server; therefore, this method is a decentralized one according to the proposed method. Furthermore, in this method, Hashtable is not used for searching resources. Also, there is not any solution about the placement of resources. Therefore, this method is an unstructured method according to the proposed method.

3.2 Network model

P2P network has emerged as a famous way to share data across a large peer population and offers several advantages over the centralized methods [28]. P2P system comprises a dynamically changing set of peers with symmetric roles connected through the Internet [55, 56]. Formally, a P2P network is defined as a non-directed graph $G = (P, E)$, where P is a non-empty set of peers in the network and E demonstrates the set of edges in the graph, such that $e = \{p_i, p_j\} \in E$ represents an edge between peers p_i and p_j. A sample of P2P network consisting of six peers that each of them can act as a client and server is shown in Fig. 1. Some definitions of P2P networks is shown in the following.

Peer: a peer in the P2P network is usually a computing node that performs as a node for sharing files within the group.

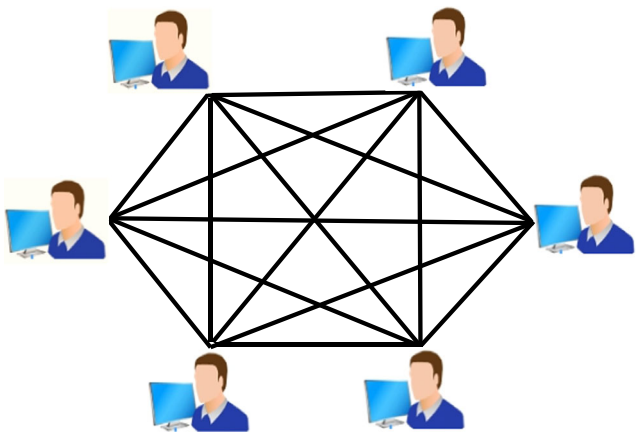

Fig. 1 A sample of P2P networks with all possible connections.

Instead of having a central server to perform as a shared drive, a peer performs as the server for the files stored on it. Query: the original mechanism for searching the distributed network that is sent via a requester (a peer that request to meet their needs) to other peers. Other peers receive a query descriptor and then respond to the query message. Request sender (requester): every peer in P2P networks contains a number of resources. But, sometimes the available resources of each peer, could not satisfy its requirements. Therefore, this peer decides to send a request to neighbor peers to obtain needed requirements. After that, this peer waits to receive the responses from neighbor peers and based on the received responses decides to select one or more of peers in order to satisfy requirements. In this paper, we define this peer as a request sender or requester.

3.3 Resource discovery using an IACO

In this section, the proposed method that is presented in the next paragraph is described. The proposed approach acts through four steps. Based on the proposed method, there are pieces of information related to pheromone and heuristic. These features are essential in the proposed method. Therefore, we need to define these features along with the equations used by them. Especially the heuristic information is defined according to the system conditions. In the equations used by the pheromone and heuristic, there are different parameters that determine the selected paths by the ants. Therefore, these parameters will be defined and the reason why they were used will be given. The flowchart used by the proposed method will be described and the steps will be shown in order. Finally, we are interested in evaluating the effectiveness of the proposed method by solving an example. We will describe the mentioned goals separately and present the used equations with the description about them below.

An IACO algorithm is applied to improve the load balancing among the peers. The IACO algorithm is a variation of the basic ACO algorithm that converts its logic by inverting the attraction of ants towards pheromones into a repulsion effect [57]. The pheromone scent in this method will create a repulsion effect instead of an attraction effect for the other ants [58]. The proposed resource discovery process uses this algorithm through four steps shown as follows:

Step 1: At first, a message that is known as a query is sent from the requester to the neighbor peers that are connected to the requester by the direct edge. The requester is a peer that wants to satisfy its requirements.

Step 2: The neighbor peers that receive the query message start to respond the query and the response messages are sent to the requester. The response message contains enough information about the peer condition and whether it contains the target resources or not.

Step 3: The requester starts to connect with one of the neighbor peers which has more probability of being selected. The requester selects from among peers that contain target resources. The requester selects the neighbor peer to move through the use of Eq. (1).

Step 4: After the first ant moves to the selected peer, requirements are updated and the pheromone amount of the traversed edge is updated according to Eq. (2).

In the proposed method, there is a graph in the form $G = (P, E)$ in which P represents the peers and E represents the edges. Ants start the graph traversal from the requester and each ant chooses its path according to the pheromone and heuristic information on the edges of the path according to Eq. (1). The pheromone and heuristic information are primary features of the IACO algorithm. These features have a significant effect on the chosen paths by the ants. Therefore, these features should be used in the basic equation (Eq. (1)). The best peer is selected to move by the first ant and the pheromone is updated according to Eq. (2) causing the algorithm to establish load balancing among the peers. Therefore, the traveled route by the previous ant might not be followed by the second ant because the available pheromone in the edge traveled by the previous ant has caused aversion. In each iteration, ants select the next peer among peers that have more probability. If the calculated probability for a peer is zero, the peer will not be selected by any ant. If two peers available in the similar condition, one of them will be chosen randomly. If all of the available edges of the requester that contain requirements for reaching neighbor peers are traversed to satisfy the requirements, the remaining ants choose the traversed paths from the first ant in order. Figure 2 shows a scenario of the proposed method that is used for the resource discovery employing the four steps mentioned above.

Equation (1) is the probability formula that is used to select the traversed paths by the ants. This formula consists of two values: $\tau_{kj}(t)$ is the pheromone on edge e_{kj} that has an initial value. This initial value cannot be zero because all obtained probabilities will be zero. Therefore, at first, the pheromone value for all edges is one. $\eta_{kJ}(t)$ is the heuristic information on edge e_{kj}. The obtained probability must be between [0, 1], therefore, the basic equation or Eq. (1) must have a denominator. This equation is shown below where t is the current iteration and α and β are values that show the amount of the impression of the pheromone and heuristic information respectively. It is assumed that the value of these parameters is one.

$$p_{kj}^{i}(t) = \frac{\left[\tau_{kj}(t)^{\alpha}\right]\left[\eta_{kj}(t)^{\beta}\right]}{\sum_{l=1}^{n} p_{kl}^{i}(t)\left[\tau_{kl}(t)^{\alpha}\right]\left[\eta_{kl}(t)^{\beta}\right]} \quad (1)$$

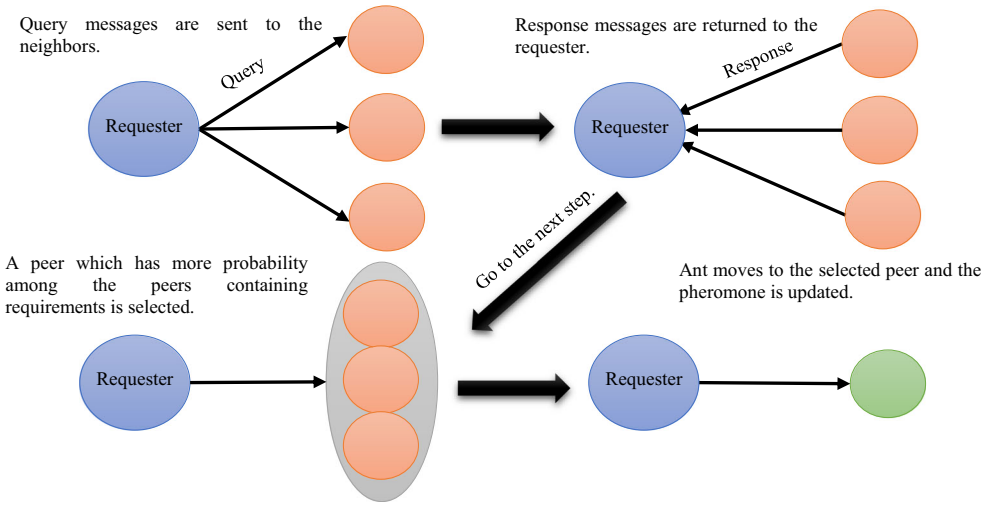

Fig. 2 Resource discovery during four steps using the IACO algorithm

The pheromone is a chemical substance produced and released into the earth by an ant, influencing the conduct or physiology of ants when they need to move for food. Equation (2) shows the updated pheromone amount in each iteration. k is the number of visited peers that increments 1 value in each iteration. Every iteration moves one peer to another peer. k must increase in each iteration that the ants could establish load balancing.

$$\tau(t+1) = \tau(t) \times \frac{1}{k} \tag{2}$$

Equation (3) is referred to as the heuristic relation. A *heuristic* is any way to deal with critical problems, learning, or discovery that utilizes a functional technique not ensured to be optimal or perfect, but sufficient for the immediate goals. The heuristic information in the proposed method consists of the profit amount and the Quality of Service (QoS) parameters. QoS elaborates the non-functional features. QoS of peers can be provided by providers, computed based on the execution and monitored by other peers. Two sorts of QoS criteria can be recognized: positive and negative. The high values for positive criteria are beneficial for requesters, such as reliability and reputation, but and the low values of negative criteria are beneficial for requesters such as cost and execution time. In this method, four basic QoS criteria are considered. P depicts the positive quality of services and N depicts the negative quality of services.

$$\eta_{kj}(t) = Q \times \left(\frac{P}{N}\right) \tag{3}$$

Q is described as the number of required resources for $i-th$ ant in the next selection among peers that is shown in Eq. (4) [59]. p_j is the next peer that will be selected and the available resources of this peer compare with all needed resources to satisfy requirements where are available in $s_r(i)$. The profit amount is considered to know that in the next peer how much of needed requirements could be satisfied.

$$Q_j^i(t) = \left|p_j \cap s_r(i)\right| \tag{4}$$

The positive quality of services is shown in Eq. (5) that consists of the availability and reputation. These criteria are very important when one peer among all peers is supposed to be selected. These criteria are described in Table 1 [60].

$$P = availability + reputation \tag{5}$$

The negative quality of services that consists of the price and response time is shown in Eq. (6). These criteria like positive criteria are very important when one peer among all peers is supposed to be selected. These criteria are described in Table 2 [60].

$$N = \text{price} + response\ \ time \tag{6}$$

Another parameter considered in comparing algorithms is called the waiting time. The waiting time refers to the time that each requester demand spends to gather requirements and it is assumed that 1 ms time is needed to use the resources of each peer that contain requirements. It is assumed that if a common peer is needed to respond the requester demands, this peer will be provided to the requester demands in order; thus other requester demands will wait for 1, 2, 3 and … ms, respectively. Therefore, to calculate the waiting time, the number of

Table 1 Positive QoS criteria

QoS criterion	Unit	Description
Reputation	Percent	A measure of service trustworthiness.
Availability	Percent	The probability that a service is accessible.

Table 2 Negative QoS criteria

QoS criterion	Unit	Description
Price	Dollar	The money that the requester has to pay to the service provider for the use of service.
Response time	Millisecond	The execution duration between the moment when a request arrives and the moment when the result is obtained.

common demands is important and crucial. It is assumed that each ant performs a requester demand.

The waiting time for each requester demand is calculated according to Eq. (7) in which n is the number of peers, t is the time that each requester demand to spend to take control over other peers and WE is the waiting time for each requester demand.

$$WE = \sum_{i=1}^{n} t_{P_i} \tag{7}$$

Equation (8) describes the total time that the requester demands have spent to meet their needs in which T is the number of demands, WE_i is the obtained waiting time for each requester demand and WA is the waiting time for all requester demands.

$$WA = \sum_{i=1}^{T} WE_i \tag{8}$$

A flowchart of the proposed method is shown in Fig. 3.

3.4 A case study

In this section, an example will be solved to show the performance of the proposed method. In this example, there are six peers which are produced randomly and $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$. A multiple peer environment with considered values for the QoS parameters for each peer is shown according to Table 3. The QoS parameters and the considered values for them are based on [60]. The parameters in the order of reputation, availability, price, and response time are also described in Tables 1 and 2. Also each of the peers has resources in the form $\{R1, R2, R3, R4, R5$ and $R6\}$ that is shown in Table 4. A request is given to the system to satisfy requirements by peer 2 or p_2. Assume that p_2 needs the resources $\{R4, R5\}$ to satisfy requirements. Also, it is assumed that M =6 (the number of ants). A graph of P2P network for this example is shown in Fig. 4.

In this example, node p_2 is considered as a requester. All ants (six ants) start the graph traversal from this peer and satisfy the needed requirements. The next peer will be selected among the peers that include required resources. $S_c(i)$ is the solution constructed by the $i-th$ ant and $S_c(i) = \{\}$ at first. When the ant selects a peer to move, the related peer is added to $S_c(i)$. Also, $S_r(i)$ is a set of needed resources that have not been meted by the $i-th$ ant and at first is equal with the needed resources that in this example $S_r(i) = \{R4, R5\}$. When $S_r(i) = \{\}$, in fact the graph traversal is finished by the $i-th$ ant. The first ant tries to choose the best peer to move from the other peers (p_3, p_4, p_5 and p_6) according to Eq. (1). The first ant chooses node p_4 because it has more probability to be selected according to Eq. (1) and $S_c(1) = \{p_4\}$. After that the first ant moves to node p_4, the pheromone amount of edge e_{14} is updated according to Eq. (2) and due to the satisfaction of p_2 requirements, the first ant leaves the system. When the first ant leaves the system, $S_r(1) = \{\}$, since all of the needed

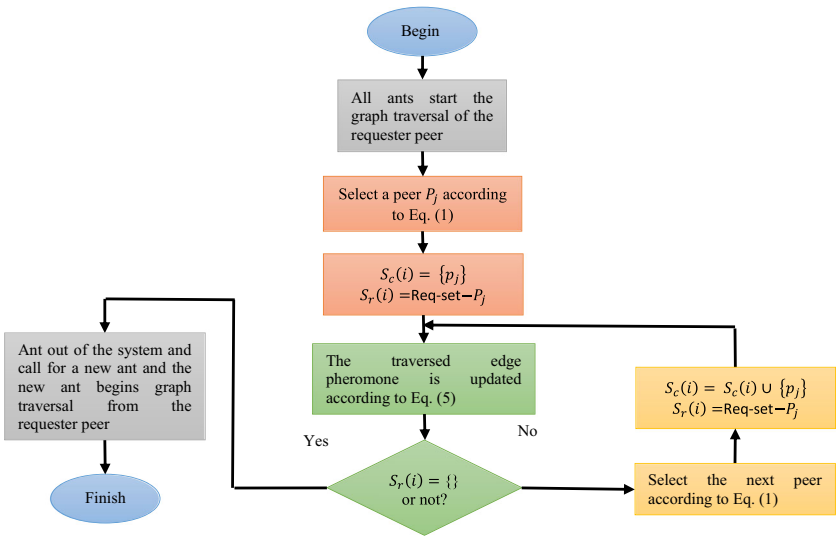

Fig. 3 Flowchart of the proposed method

Table 3 One multiple peer environment with QoS parameters

MPE QoS	p_1	p_2	p_3	p_4	p_5	p_6
Reputation	0.8	0.8	0.9	0.9	0.7	0.8
Availability	0.98	0.96	1	0.99	0.97	0.97
Price	3	4.5	2	2.5	4	3.5
Response time	80	75	100	90	70	80

resources are meted by the first ant. Then, the second ant will begin the traversal from the source node and chooses the next peer that has more probability according to Eq. (1). The second ant selects at first p_5 and $S_c(2) = \{p_5\}$. Now, $R4$ is met and because $S_r(2) = \{R5\}$, it continues graph traversal until requirements are satisfied. In the following, the second ant chooses p_6 according to Eq. (1) and $S_c(2) = \{p_5, p_6\}$ and $S_r(2) = \{\}$. The selected peers are in the order of $m_3 = \{p_6, p_4\}$, $m_4 = \{p_4\}$, $m_5 = \{p_3, p_6\}$, $m_6 = \{p_5, p_4\}$ for the third, fourth, fifth and sixth ants respectively. The obtained conclusions show that the proposed method improves load balancing among the peers that is elaborated in the next section.

4 Evaluation and analysis

In this section, we compare the presented approach with the ACO mechanism [6]. At first, we describe the simulation tool and then propose the used datasets. After that, we present comparative parameters and explain each of them. Finally, we evaluate the simulation results.

4.1 Simulation tool and environment

The Java programming language is used for the simulation in the eclipse environment. The Eclipse is an Integrated Development Environment (IDE) utilized in programming and is the most broadly utilized Java IDE. A set of standard metrics is utilized to measure the effectiveness of the proposed method in comparison with the ACO algorithm. The standard metrics are as follows: load balancing, waiting time, resource utilization and execution time. The experimental environment is shown in Table 5. Tests will be run on Intel Pentium 4630 at 3.00 GHz with 6 GB of RAM on a 64 bits Windows 7 machine. We explain a summary paragraph about how to simulate.

We use two methods for receiving the number of the ants and the number of the peers separately. The Scanner class is used to receive the number of the ants and the peers. We used the HashMap structure to save the available resources of each peer. The HashMap structure consists of two elements, Key and Value. The Key is a String and the Value is an ArrayList String for saving the available resources of each peer. We use an ArrayList String for saving the target resources and the target resources are added to the ArrayList by the add command.

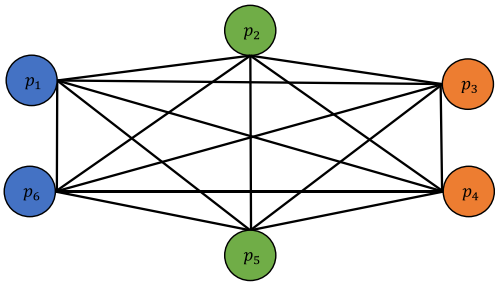

Fig. 4 P2P graph for the supposed example

4.2 Experimental datasets

We use three groups of experimental datasets to evaluate the proposed method in comparison with the ACO method. We propose each group of experimental datasets separately. These groups are called the first group or original dataset, the second group or random dataset and the third group or huge dataset. The proposed method is compared with the ACO method in all experimental groups.

4.2.1 Original dataset

At first, the mentioned dataset in [59] is considered to compare the existing methods as shown in Table 6. This group called a first or original group of experimental datasets. Suppose that p_3 requires the resource $\{R3\}$ to satisfy the requirements. We consider $M = 5$ (number of ants) and will be started the graph traversal. Heuristic information for four peers is presented according to Table 3.

4.2.2 Random dataset

The second group of the experimental dataset wheraie six peers are randomly generated is called the random dataset which is illustrated in Section 3.4. Tables 3, 4 and Fig. 4 contain the related information that shows the second group of experimental dataset.

Table 4 A group of peers with available resources

Set of peers	p_1	p_2	p_3	p_4	p_5	p_6
Resources	$\{R1, R6\}$	$\{R2, R3\}$	$\{R2, R4\}$	$\{R2, R4, R5\}$	$\{R1, R4\}$	$\{R1, R3, R5\}$

Table 5 The experimental environment

Experiment variables	Values (range)
Number of peers	4, 6, 100, 1000, 5000, 1,000,000
Number of ants	4, 6, 100, 1000, 5000, 1,000,000
Number of available peers from each peer (neighbor peers)	3,4, …, 99
Number of available resources in each peer	1, 2, 3, 4, 5
Number of needed target resources	1,2,3, …, 333,333

4.2.3 Huge dataset

The third group of the experimental dataset is called huge dataset where it is conducted in an environment with 100, 1000, 5000 and 1,000,000 peers. It is assumed that the number of the target resources for 100, 1000 and 5000 peers based on $N \times \frac{1}{2}$ is 50, 500, 2500 respectively. Also, the number of the target resources for 10^6 peers based on $N \times \frac{1}{3}$ is almost 333,333. Furthermore, the obtained results for 4 and 6 peers will be shown in this group. Table 7 shows the experimental dataset for the third group of the experimental dataset. When the number of peers is 100, the considered graph is complete; but in other cases, the considered graph is incomplete.

4.3 Comparative parameters

In this section, the comparative parameters including the load balancing, waiting time, resource utilization and execution time are elaborated separately.

4.3.1 Load balancing

In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid the overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a domain name system server process. Load balancing is calculated according to Eq. (9) in which LB is obtained load balancing for each environment and PU is the amount of used peers in the related environment. More explanations will be mentioned in the following sections.

$$LB = \frac{PU}{100} \tag{9}$$

4.3.2 Waiting time

This parameter was defined in Section 3.3. The waiting time refers to the time that each requester demand spends to gather requirements and it is assumed that 1 ms time is needed to use the resources of each peer that contain requirements. It is assumed that if a common peer is needed to respond the requester demands, this peer will be provided to the requester demands in order; thus other requester demands will wait for 1, 2, 3 and … ms, respectively. Therefore, to calculate the waiting time, the number of common demands is important and crucial. It is assumed that each ant performs a requester demand. The waiting time for each requester demand and all requests are calculated according to Eqs. (7) and (8) that in Section 3.3 was described.

4.3.3 Execution time

The execution time or running time is the time during which a program is executing or running.

4.3.4 Resource utilization

Eventually, we decide to calculate the average resource utilization for the proposed method in comparison with the ACO algorithm. For the resource utilization, there are many definitions in different aspects. But, we define resource utilization in computer aspect. Resource utilization is a way to track how busy various resources of a computer system are when

Table 6 The first group of the experimental dataset

Set of peers Set of environments	p_1	p_2	p_3	p_4
Environment 1	$\{R1, R3, R4\}$	$\{R3\}$	$\{R2, R4\}$	$\{R3, R5\}$
Environment 2	$\{R2, R4\}$	$\{R1, R2\}$	$\{R1\}$	$\{R3, R4, R5\}$
Environment 3	$\{R1, R2, R3\}$	$\{R1, R5\}$	$\{R1, R2, R4\}$	$\{R3, R5\}$
Environment 4	$\{R1, R2, R4\}$	$\{R3, R5\}$	$\{R1, R5\}$	$\{R1, R2, R3, R4\}$

Table 7 The third group of the experimental dataset

The number of peers	100	1000	5000	1,000,000
Available resources	1, …, 5	1, …, 5	1, …, 5	1, …, 5
The number of target resources	50	500	2500	333,333
The number of neighbors (complete and incomplete graph)	99	Random	Random	Random

running a performance test. In other words, the resource utilization is a process that has been planned to use different resources for satisfying a certain goal instead of using one resource. In the proposed method, we try to utilize the resource efficiency to satisfy requirements with the help of the IACO algorithm. Equation (10) calculates the resource utilization in which RU is resource utilization, TR is the total percentage of used peers in all environments and E is the number of environments. More explanations will be mentioned in the next sections.

$$RU = \frac{TR}{E} \tag{10}$$

4.4 Comparative analysis

In this section, the obtained results of executing the proposed method and the ACO on the different groups of datasets are shown. We evaluate different groups of datasets separately and at first show the obtained results for the first group of experimental datasets.

4.4.1 The obtained results for the original dataset

In this section, the effectiveness of the proposed method and the ACO method in terms of the load balancing, waiting time, execution time and resource utilization are evaluated.

The obtained results in Fig. 5 show that the proposed method in three environments of Table 6 uses more peers to satisfy requirements. Since in $E1$, $E3$, and $E4$ of Table 6 there is more than one peer that contains the target resource ($R3$), therefore, the proposed method uses different peers to satisfy requester (p_3) requirements. For example, in $E1$ environment, the first ant selects p_4 to satisfy requester requirements, the second ant chooses p_2 and the third ant selects p_1. While the ACO method for all environments uses only one of the peers to satisfy requester requirements, it can be seen that, the proposed method has a better load balancing in comparison with the ACO algorithm. The performance of the proposed method and the ACO are equal in $E2$ because the target resource is only one peer. The performance of the proposed method is more efficient in $E3$ and $E4$ because the target resource is available on two different peers.

The existing results in Fig. 6 show that in the proposed method in three cases the requests are processed in less time in comparison with the ACO algorithm. In other words, in three cases the performance of the proposed method is better than that of the ACO algorithm in terms of the waiting time. The performance of the proposed method and the ACO are equal in the case of $E2$ because like the ACO method the proposed method uses only one peer to satisfy requirements. It is assumed that five common requests are given to the system by the requester or p_3 to calculate the waiting time.

The comparisons of the execution time of the proposed method and the ACO method for the mentioned dataset in [59] is shown in Fig. 7. The obtained results show that the execution time of the ACO method is less than that of the IACO method. Only in $E2$, the effectiveness of the proposed method and the ACO method is equal because there is only one peer that contains requirement resources. Unfortunately, the effectiveness of the presented method is weaker than that of the ACO method in terms of the execution time.

In the following, the average resource utilization for the proposed method and the ACO method is calculated. The obtained results show that the proposed method has a better average resource utilization in comparison with the ACO

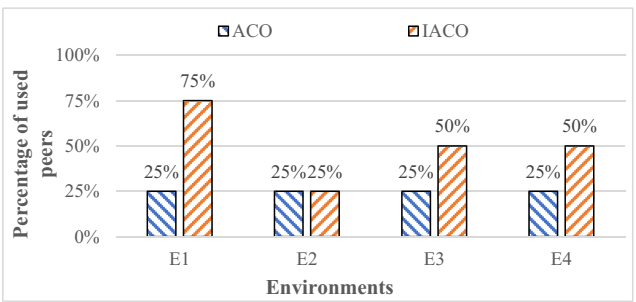

Fig. 5 The comparison of the algorithms in terms of the percentage of used peers in the original dataset

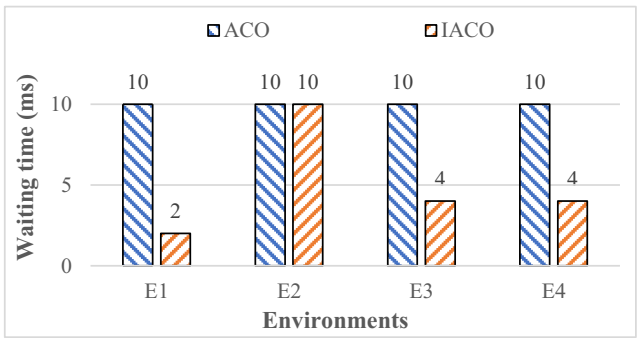

Fig. 6 The comparison of the algorithms in terms of the waiting time in the original dataset

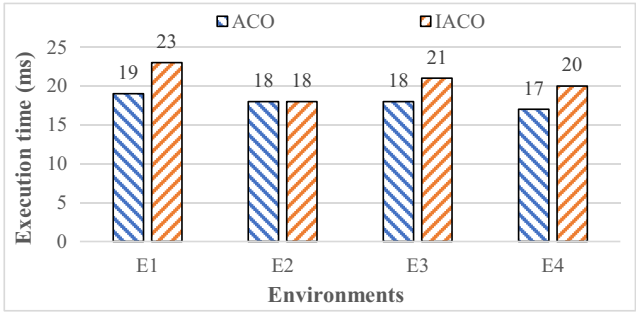

Fig. 7 The comparison of the algorithms in terms of the execution time in the original dataset

method. Table 8 shows that the proposed method of 50% peers and the ACO of 75% peers have not been utilized.

4.4.2 The obtained results for the random dataset

In this subsection, the second group of the experimental dataset that refers to the mentioned example in Section 3.4 is considered. This group of the dataset was produced randomly. As it is evident in Fig. 8, the percentage used by each peer in the graph traversal by six ants' shows that the performance of the proposed method is better than that in the ACO algorithm in terms of load balancing among the peers. The IACO method uses four peers to respond p_2 requirements, whereas the ACO algorithm uses only one peer to respond p_2 requirements. Figure 9 shows the general percentage used peers for M=6 (number of ants). As it is clear, the IACO algorithm has a better load balancing than that of the ACO algorithm.

Figure 10 shows that the proposed method takes less time in comparison with the ACO algorithm for processing requests. In other words, the proposed method spends less time to respond the requester requirements. It is assumed that six common requests are given to the system by p_2 to calculate the waiting time.

Table 8 The obtained results in average resource utilization in the original dataset

Algorithms	Resource utilization
ACO algorithm	25%
IACO algorithm	50%

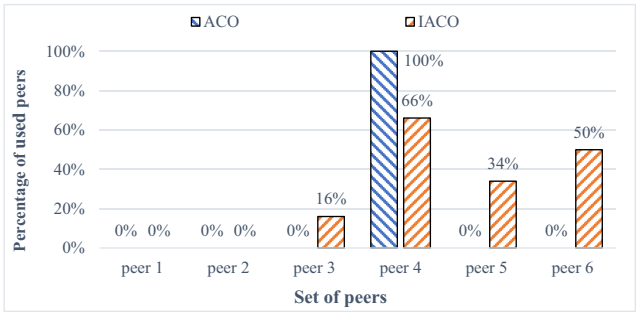

Fig. 8 The comparison of the algorithms in terms of the percentage of used peer (by peer) in the random dataset

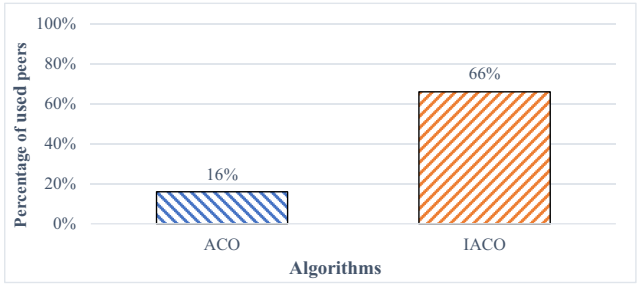

Fig. 9 The comparison of the algorithms in terms of percentage of used peers (by algorithms) in the random dataset

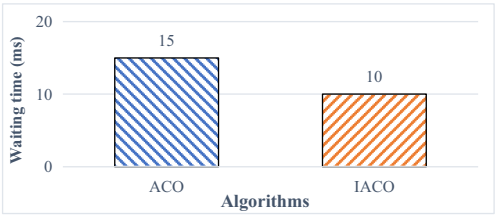

Fig. 10 The comparison of the algorithms in terms of the waiting time in the random dataset

The comparisons of the execution time of the proposed method and the ACO for the mentioned example in Section 3.4 are shown in Fig. 11. The obtained results show that the execution time of the ACO method is less than that of the IACO method. A comparison of the results of the two algorithms also shows that the execution time in the proposed method increases and that is considered as a disadvantage of the proposed method.

Furthermore, we evaluate the average resource utilization for the proposed method and the ACO method. The obtained results show that the proposed method has a better average resource utilization in comparison with the ACO method. Table 9 shows that the proposed method of 34% peers and the ACO of 84% peers have not been utilized.

4.4.3 The obtained results for the huge dataset

Final experiments were conducted in an environment with 100, 1000, 5000 and 1,000,000 peers. The obtained results for load balancing, waiting time, execution time and resource

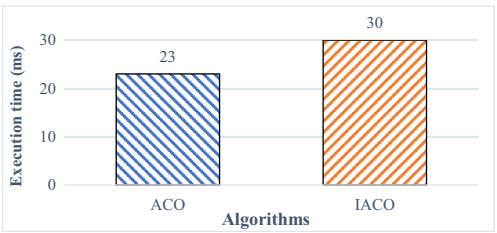

Fig. 11 The comparison of the algorithms in terms of the execution time in the random dataset

Table 9 The obtained results in the average resource utilization in the random dataset

Algorithms	Resource utilization
ACO algorithm	16%
IACO algorithm	66%

utilization have been shown in Fig. 12, Table 10, Table 11 and Table 12 respectively. The obtained results show that with the increase in the number of the peers, the proposed method performance gets more efficient in terms of load balancing among the peers in comparison with the ACO algorithm. In the IACO algorithm, as the number of the peers increases, the number of the visited nodes also increases and therefore, the waiting time decreases in comparison with the ACO algorithm. When the number of peers increases, the number of available peers of each peer (neighbor peers), the number of resources for each peer and the number of target resources also increases in a certain scope according to the considered values in Table 5. Therefore, the obtained results may be different in each iteration (program run). The number of target resources is evaluated based on certain conditions described in Section 4.2.3. If the number of target resources to satisfy requester requirements is 50, the maximum number of peers that is needed to satisfy the requirements is 50 for the ACO method. But in the proposed method, due to the updated pheromone, it can be different. These results are a basis to calculate the next experiments. The percentage of used peers is shown in Fig. 12. As the number of the peers increases, the performance of the proposed method in terms of load balancing among peers would certainly be better in comparison with the ACO algorithm.

Of course, when the number of peers is 4, 6 and 100, the considered graph to calculate load balancing is complete. But when the number of peers increases, or in other words, when the number of peers is 1000, 5000 and 1,000,000 peers, the considered graph to calculate load balancing is incomplete, because the scalability of P2P systems can only be achieved with incomplete graphs. It is clear that when the graph is incomplete, the effectiveness of the proposed method in terms of the load balancing is decreased, especially, when the number of the peers is low. Therefore, the effectiveness of the waiting time is decreased. In other words, the total waiting time of the requests to receive requirements is increased. Also, the effectiveness of the ACO algorithm is almost constant in terms of the load balancing and waiting time when the graph is complete or incomplete.

Table 10 The comparison of the algorithms in terms of the waiting time in the huge dataset

Algorithms The number of peers	ACO	IACO
4	10	4
6	15	10
100	240	180
1000	2700	2250
5000	15,000	12,300
1,000,000	1,999,998	1,320,000

It is assumed that four common requests are given to the system by the requester peer to calculate the waiting time for the proposed method and the ACO method for 100, 1000, 5000 and 1 million peers. As the number of the peers increases, the performance of the proposed method in terms of waiting time would certainly be better in comparison with the ACO algorithm. The obtained results are shown in Table 10.

But, in terms of the execution time, the proposed method acts weaker than the ACO algorithm. The obtained results are shown in Tables 11.

In the following, we attempt to get the average resource utilization for the huge dataset. The obtained result in Fig. 12 is considered for different numbers of the peers. It can be seen, there are six environments that consist of $N=4$, $N=6$, $N=100$, $N=1000$, $N=5000$ and N = 1,000,000. When the number of peers is 4, 6 and 100, the considered graph is complete. But when the number of peers is 1000, 5000 and 1,000,000

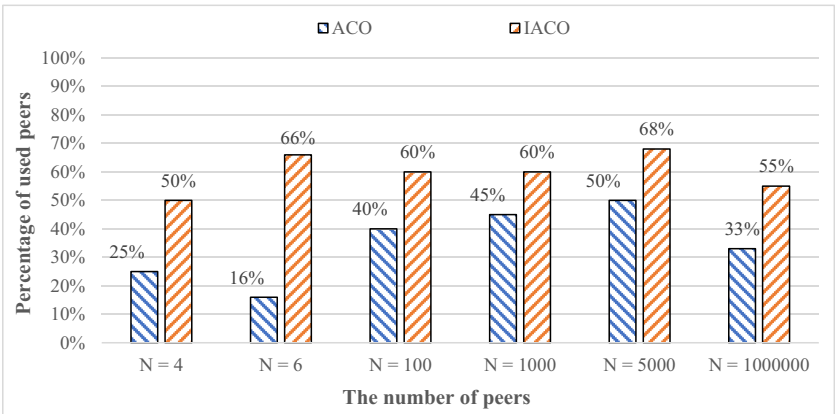

Fig. 12 The comparison of the algorithms in terms of the percentage of used peers in the huge dataset

Table 11 The comparison of the algorithms in terms of the execution time in the huge dataset

Algorithms The number of peers	ACO	IACO
4	19 ms	22 ms
6	23 ms	30 ms
100	2000 ms	2015 ms
1000	18,300 ms	18,650 ms
5000	78,850 ms	79,550 ms
1,000,000	150,400,000 ms	153,120,000 ms

peers, the considered graph is incomplete. Table 12 indicates the obtained results and, as has been shown, the performance of the IACO method is certainly better in comparison with the ACO algorithm. It is clear, almost 40% of peers is not used in the proposed method and 65% of peers is not used in the ACO method.

5 Conclusion and future work

In this paper, we proposed a method for the resource discovery in P2P networks using an IACO algorithm that consisted of four sections. In the first section, the query message of the requester was sent to the neighbor peers, neighbor peers tried to respond the query and the response message returned to the requester. In the third section, the optimal peer which contained the best value of probability was selected. Finally, the pheromone amount and requirements were updated causing the proposed method to improve load balancing among the peers. In the cases of completed and incomplete graphs, the performance of the proposed method was evaluated using Java. The obtained results showed that the proposed method had a better performance among different groups of experimental datasets in terms of load balancing, waiting time and average resource utilization in comparison with the ACO algorithm. The obtained results in all experiments depended on the number of the peers, the number of the target resources, the available resources in each peer, the number of the neighbor peers, the fullness of the graph and the number of ants. Therefore, it could not be said with certainty that the obtained results in the previous section were always stable and in each program execution, it could be different, especially when the number of peers increased. But it could be said with certainty that the IACO was superior to the ACO algorithm in terms of load balancing, waiting time and resource utilization. Also, since there wasn't a central server, the problem of a single point of failure didn't happen. Furthermore, since in the proposed method the flooding strategy was not used, the huge overhead problem did not happen due to the exponential increase of query packets.

Unfortunately, in the evaluation, parallelism and network latency were not considered. Therefore, in the future, using the peersim or equivalent simulation toolkit for parallelism and network latency calculation are very interesting. Also, the effectiveness of the proposed method in terms of the execution time was weaker than the ACO method. Considering the proposed method in different network topologies can be challenging. Furthermore, using the other meta-heuristic algorithms and evaluating the performance of them in terms of the load balancing, waiting time and execution time will be interesting.

Table 12 The obtained results for average resource utilization in the huge dataset

Algorithms	Resource utilization
ACO algorithm	34.83%
IACO algorithm	59.83%

References

1. Asghari S, Navimipour NJ (2016) Service composition mechanisms in the multi-cloud environments: a survey. Int J New Comput Archit Appl (IJNCAA) 6:40–48
2. Asghari S, Navimipour NJ (2016) Review and comparison of meta-heuristic algorithms for service composition in cloud computing. Majlesi J Multimed Process 4
3. Ashouraie M, Jafari Navimipour N (2015) Priority-based task scheduling on heterogeneous resources in the expert cloud. Kybernetes 44:1455–1471
4. Navimipour NJ, Rahmani AM, Navin AH, Hosseinzadeh M (2015) Expert cloud: a cloud-based framework to share the knowledge and skills of human resources. Comput Hum Behav 46:57–74
5. Afrooz S, Navimipour NJ (2017) Memory designing using quantum-dot cellular automata: systematic literature review, classification and current trends. J Circ Syst Comput 26:1730004
6. Krynicki K, Jaen J, Mocholi JA (2013) On the performance of ACO-based methods in p2p resource discovery. Appl Soft Comput 13(12):4813–4831
7. Mirtaheri SL, Sharifi M (2014) An efficient resource discovery framework for pure unstructured peer-to-peer systems. Comput Netw 59:213–226
8. Navimipour NJ, Milani FS (2015) A comprehensive study of the resource discovery techniques in peer-to-peer networks. P2P Netw Appl 8:474–492
9. Akbari Torkestani J (2012) A distributed resource discovery algorithm for P2P grids. J Netw Comput Appl 35(11):2028–2036
10. Han X, Cuevas Á, Crespi N, Cuevas R, Huang X (2014) On exploiting social relationship and personal background for content discovery in P2P networks. Futur Gener Comput Syst 40(11):17–29
11. Deng Y, Wang F, Ciura A (2009) Ant colony optimization inspired resource discovery in P2P grid systems. J Supercomput 49:4–21
12. Wang L (2011) SoFA: an expert-driven, self-organization peer-to-peer semantic communities for network resource management. Expert Syst Appl 38(1):94–105
13. Beydoun G, Low G, Tran N, Bogg P (2011) Development of a peer-to-peer information sharing system using ontologies. Expert Syst Appl 38(8):9352–9364

14. Navimipour NJ, Rahmani AM, Navin AH, Hosseinzadeh M (2014) Resource discovery mechanisms in grid systems: a survey. J Netw Comput Appl 41:389–410
15. Navimipour NJ, Asghari S (2017) Energy-aware service composition mechanism in grid computing using an ant colony optimization algorithm. 대한전자공학회 학술대회, pp 282–286
16. Aznoli F, Navimipour NJ (2016) Cloud services recommendation: reviewing the recent advances and suggesting the future research directions. J Netw Comput Appl 77:73–86
17. Navimipour NJ (2015) A formal approach for the specification and verification of a trustworthy human resource discovery mechanism in the expert cloud. Expert Syst Appl 42:6112–6131
18. Keshanchi B, Navimipour NJ (2016) Priority-based task scheduling in the cloud systems using a memetic algorithm. J Circ Syst Comput 25:1650119
19. Vakili A, Navimipour NJ (2017) Comprehensive and systematic review of the service composition mechanisms in the cloud environments. J Netw Comput Appl 81:24–36
20. Azad P, Navimipour JN (2017) An energy-aware task scheduling in cloud computing using a hybrid cultural and ant colony optimization algorithm. Int J Cloud Appl Comput 7:20–40
21. Milani BA, Navimipour NJ (2017) A systematic literature review of the data replication techniques in the cloud environments. Big Data Res 10:1–7. https://doi.org/10.1016/j.bdr.2017.06.003.
22. Sharif SH, Mahmazi S, Navimipour NJ, Aghdam BF (2013) A review on search and discovery mechanisms in social networks. Int J Inf Eng Electron Bus 5:64–73
23. Asghari S, Azadi K (2017) A reliable path between target users and clients in social networks using an inverted ant colony optimization algorithm. Karbala Int J Mod Sci 3(3):143–152. https://doi.org/10.1016/j.kijoms.2017.05.004.
24. Bakratsas M, Basaras P, Katsaros D, Tassiulas L (2017) Hadoop mapreduce performance on SSDs for analyzing social networks. Big Data Res. https://doi.org/10.1016/j.bdr.2017.06.001
25. Merz P, Gorunova K (2007) Fault-tolerant resource discovery in peer-to-peer grids. J Grid Comput 5:319–335
26. A. Arunachalam and O. Sornil (2015) "Issues of Implementing Random Walk and Gossip Based Resource Discovery Protocols in P2P MANETs & Suggestions for Improvement," Procedia Comput Sci, 57:509–518
27. Meshkova E, Riihijärvi J, Petrova M, Mähönen P (2008) A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Comput Netw 52:2097–2128
28. Trunfio P, Talia D, Papadakis H, Fragopoulou P, Mordacchini M, Pennanen M, Popov K, Vlassov V, Haridi S (2007) Peer-to-peer resource discovery in grids: models and systems. Futur Gener Comput Syst 23:864–878
29. Gaeta R, Sereno M (2011) Generalized probabilistic flooding in unstructured peer-to-peer networks. IEEE Trans Parallel Distrib Syst 22:2055–2062
30. Lua EK, Crowcroft J, Pias M, Sharma R, Lim S (2005) A survey and comparison of peer-to-peer overlay network schemes. IEEE Commun Surv Tutorials 7:72–93
31. Kirk P (2003) Gnutella protocol development. Retrieved June, vol. 27, pp 2011
32. Ghamri-Doudane S, Agoulmine N (2007) Enhanced DHT-based P2P architecture for effective resource discovery and management. J Netw Syst Manag 15:335–354
33. Maymounkov P, Mazieres D (2002) Kademlia: a peer-to-peer information system based on the xor metric. In: Peer-to-peer systems Springer, pp 53–65
34. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput Commun Rev 31:149–160
35. Yang B, Garcia-Molina H (2003) Designing a super-peer network. In: Data engineering, 2003. Proceedings. 19th international conference on, pp 49–60
36. Tan Y-H, Lü K, Lin Y-P (2012) Organisation and management of shared documents in super-peer networks based semantic hierarchical cluster trees. P2P Netw Appl 5:292–308
37. Stokes M (2002) Gnutella2 specifications part one. Rapport technique
38. Stokes M (2003) Gnutella2 specification document–first draft. Gnutella2 website http://www.gnutella2.com/gnutella2_draft. htm
39. Haasn MI (2011) Semantic technology and super-peer architecture for internet based distributed system resource discovery. Int J New Comput Archit Appl (IJNCAA) 1:848–865
40. Ali HA, Ahmed MA (2012) HPRDG: a scalable framework hypercube-P2P-based for resource discovery in computational grid. In: Computer Theory and Applications (ICCTA), 2012 22nd International Conference on, pp 2–8
41. Yang M, Yang Y (2010) An efficient hybrid peer-to-peer system for distributed data sharing. IEEE Trans Comput 59:1158–1171
42. Liu M, Harjula E, Ylianttila M (2013) An efficient selection algorithm for building a super-peer overlay. J Internet Serv Applic 4:1–12
43. Loo BT, Huebsch R, Stoica I, Hellerstein JM (2004) The case for a hybrid P2P search infrastructure. In: Peer-to-peer systems III. Springer, pp 141–150
44. Papadakis H, Trunfio P, Talia D, Fragopoulou P (2008) Design and implementation of a hybrid P2P-based grid resource discovery system. In: Making grids work. Springer, pp 89–101
45. Napster L (2001) Napster. URL: http://www.napster.com
46. Jin X, Chan S-HG (2010) Unstructured peer-to-peer network architectures. In: Handbook of peer-to-peer networking. Springer, pp 117–142
47. Mashayekhi H, Habibi J (2010) Combining search and trust models in unstructured peer-to-peer networks. J Supercomput 53:66–85
48. Zaharia M, Keshav S (2008) Gossip-based search selection in hybrid peer-to-peer networks. Concurr Comput Pract Exper 20:139–153
49. Barjini H, Othman M, Ibrahim H (2010) An efficient hybridflood searching algorithm for unstructured peer-to-peer networks. Inf Comput Appl:173–180
50. Kumar A, Xu J, Zegura EW (2005) Efficient and scalable query routing for unstructured peer-to-peer networks. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, pp 1162–1173
51. Chawathe Y, Ratnasamy S, Breslau L, Lanham N, Shenker S (2003) Making gnutella-like p2p systems scalable. In: proceedings of the 2003 conference on applications, technologies, architectures, and protocols for. Comput Commun:407–418
52. Dorigo M, Maniezzo V, Colorni A (1991) The ant system: an autocatalytic optimizing process
53. Jaén J, Mocholí JA, Catalá A, Navarro E (2011) Digital ants as the best cicerones for museum visitors. Appl Soft Comput 11:111–119
54. Mocholi JA, Martinez V, Jaen J, Catala A (2012) A multicriteria ant colony algorithm for generating music playlists. Expert Syst Appl 39:2270–2278
55. Krauter K, Buyya R, Maheswaran M (2002) A taxonomy and survey of grid resource management systems for distributed computing. Softw Pract Exper 32:135–164
56. Souri A, Navimipour NJ (2014) Behavioral modeling and formal verification of a resource discovery approach in grid computing. Expert Syst Appl 41:3831–3849
57. Asghari S, Navimipour J (2017) Cloud services composition using an inverted ant colony optimization algorithm. Int. J. Bio-Inspired Comput in press. Google Scholar
58. Dias JC, Machado P, Silva DC, Abreu PH (2014) An inverted ant colony optimization approach to traffic. Eng Appl Artif Intell 36:122–133

59. Yu Q, Chen L, Li B (2015) Ant colony optimization applied to web service compositions in cloud computing. Comput Electr Eng 41: 18–27
60. Wang D, Yang Y, Mi Z (2015) A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng 43:129–141

Saied Asghari received his B.S. in computer engineering, software engineering, from Khoy Branch, Islamic Azad University, Khoy, Iran, in 2014; the M.S. in computer engineering, software engineering, from Tabriz Branch, Islamic Azad University, Tabriz, Iran, in 2016. His research interests include Cloud Computing, Peer-to-Peer Networks, Social Networks and Grid Computing.

Nima Jafari Navimipour received his B.S. in computer engineering, software engineering, from Tabriz Branch, Islamic Azad University, Tabriz, Iran, in 2007; the M.S. in computer engineering, computer architecture, from Tabriz Branch, Islamic Azad University, Tabriz, Iran, in 2009; the Ph.D. in computer engineering, computer architecture, from Science and Research Branch, Islamic Azad University, Tehran, Iran in 2014. He is an assistance professor in the Department of Computer Engineering at Tabriz Branch, Islamic Azad University, Tabriz, Iran. He has published more than 100 papers in various journals and conference proceedings. His research interests include Cloud Computing, Social Networks, Fault-Tolerance Software, Computational Intelligence, Evolutionary Computing, and Network on Chip.