Computer & Robotics

Providing A Method Based On The Combination Of The Tree And Super-Peer Structures For Resource Discovery In The Grid Environment

Behnam Norouzi[a], Mahdi Mollamotalebi[b,*]

[a]*Department of Computer Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[b]*Department of Computer, Buinzahra Branch, Islamic Azad University, Buinzahra, Iran*

Received 4 December 2019; Accepted 7 September 2021

Abstract

The grid is a new generation of distributed networks that supply resources required for applications with high processing and memory loads. Given the specific features of the grid environment, including high dynamics and heterogeneity of resources/elements, some challenges are encountered on this computational platform. One of the main services of the grid platform is the discovery of resources. The purpose of the resource discovery service is to identify a list of resources available for assigning tasks. In this paper, using the assignment of prime numbers as the weight of the index tree nodes, and by combining the hierarchical method with a peer-to-peer (P2P) method, a new algorithm has been presented, in which several networks with the hierarchical internal structure are combined by means of the super-peer method. The results of the experiments and their comparison with previous methods indicate the improvement achieved by the proposed method in terms of the number of nodes visited in the searching process and by reducing the processing load induced by the calculation of the weight of the tree edges.

Keywords: edge weight, resource discovery, bitmap, prime number, local node resource

1. Introduction

The grid is a new generation of distributed networks and allows its users to share files. An example would be the Internet. These systems allow the resources of heterogeneous computers to be shared (Nabila Chergui, 2010). The main purpose of the grid is to use these shared resources, including processor power, bandwidth, and databases, and provide these for complex and heavy processing applications. Due to the specific features of the grid environment, like high dynamics and heterogeneity of resources and elements of the network, challenges are encountered in this computational framework (Mahdi MollaMotalebi, 2013).Resource discovery is one of the main services in the grid environment. The purpose of resource discovery is to identify a list of available resources for

* Corresponding Author. Email:motalebi@qiau.ac.ir

assigning tasks. When a user in a grid environment requests a resource to run a particular application, the resource discovery method should be able to find the resource requested by the user at a very low time and space costs the user so that s/he can exploit the message exchanged as soon as possible and with the least load (Khanli, 2011). In recent years, with the development of communication and Internet systems, as well as the users' increased need for high-precision and low-cost computations, the grid computing system is one of the solutions available to achieve the distributed systems. If an algorithm with low traffic and no reference to unnecessary nodes can provide a good source for users in a short time, it will significantly increase the system effectiveness. Given the importance of resource discovery in grid environments, this paper provides an improved method for discovering the resources in the grid environment to reduce the number of nodes visited while searching for resources and to reduce the number of nodes that need to be updated after resource allocation.

Among the following sections, the second one presents an overview of the recent works and resource discovery methods in the grid environment. In the third section, the proposed method is described in detail. The results of the implementation of the proposed method are presented and evaluated in the fourth section. The fifth part concludes the article.

2. Related Works

The grid computing network is a collection of several systems with different levels of computing power, resulting in a virtual supercomputer with the ability to perform complex mathematical, astronomical, and biomedical calculations in a short time. In grid computing networks, several users share the resources on the network. Each user, simultaneously with the others, makes requests independently, and the grid resource management system (GRMS) searches and allocates the resources necessary for the task received. In this process, the number of nodes visited, that of the resources discovered, and that of the updates made in the nodes are of great importance. In the following paragraphs, different grid discovery methods are reviewed and criticized.

The goal of the resource discovery systems is to find the resources available for assigning tasks considering the specific features of the grid environment (like high dynamics and resource heterogeneity) (Khanli, 2011). Over the past years, researchers have made several attempts to improve resource discovery techniques in the grid environment. In general, resource discovery methods in the grid are divided into five main categories: centralized mechanisms, decentralized mechanisms, peer-to-peer mechanisms, hierarchical mechanisms, and agent-based mechanisms.

In centralized mechanisms, one or more controller sets to discover the resources with the client/server architecture. In this method, the servers store information about the services they can provide. When an entity requests a particular service, its request is

sent to the server and a suitable resource is found and allocated to that entity. Therefore, all requests are processed by one or more controller sets, and in environments with thousands of nodes, this causes a bottleneck (Jaffari Navimipour, 2013). However, this method takes the least search time.

Another problem with these methods is the updating of information. In dynamic environments, in which each node can connect to the network at any moment, leave the network, or reconnect to it, updating methods that impose the least burden on the network are required. However, in environments controlled by centralized mechanisms, since the entire network information is stored in one or more controllers, network updates are difficult due to the network's dynamic characteristic.

In agent-based mechanisms, certain software components are used as agents (mobile agents) with autonomy, mobility, and reactivity features. Each agent can be mapped to a node to store its resource information. The agent can also scroll the network to find the requested resources. The autonomy of the agents enables them to act intelligently and change their behaviour according to the environment during the search. The use of these factors in potential grid environments increases the scalability and flexibility of the resource discovery process (Hamorline, 2010). Each agent can store the information of the nodes' resources. The agent also scrolls the network to find the requested resource. The independence of the agents makes them suitable for providing intelligent performance with the ability to change their behaviour based on different conditions during the search.

In decentralized (peer-to-peer) mechanisms, there is no central database, and all nodes cooperate to perform resource discovery in large-scale systems. These methods act better than the centralized methods in large-scale environments, though they create additional overheads to manage network architectures. Decentralized methods allow the users to dynamically leave the network or reconnect to it; so, they can be said to have high dynamics. Furthermore, these methods have high scalability, fault tolerance, and load balancing, and they prevent the problem of a single point of failure. However, many decentralized methods need to update their resource information periodically. Therefore, they cannot support requests with dynamic features (Jaffari Navimipour, 2013).

Centralized and decentralized methods cannot effectively function in distributed, dynamic, and large-scale environments. The high number of nodes in the environment causes these methods to be ineffective. Therefore, in large-scale grid environments, certain methods must be used to cover the requirements. Peer-to-peer technology appeared as a well-known method for sharing data in environments with high node populations and presented several advantages of the centralized structures (Hawa, 2013).In peer-to-peer systems, elements have the same role and, unlike traditional client/server systems, can exchange information and provide services (Tan, 2012).

In 2012, Ali and Ahmed presented a scalable framework for discovering the source in peer-to-peer networks based on the hypercube computational grids. The service node is a normal node, which is more efficient than the other nodes in some respects. For

example, nodes with high availability, higher CPU speeds, and higher bandwidths in connections can be selected as service nodes. In addition, in the resource discovery stage, the best node is chosen from among the nodes that have high reliability as the resource discovery service node. This method provides load balancing among service nodes in the network environment with the proper distribution. This method acts in such a way that the closer (locally) resources can be found quickly. This method is appropriate for a large-scale distributed resource-sharing system with heterogeneous resources and different sharing policies. This method is also scalable in terms of response time. In addition, multi-feature requests are also supported. In the connection phase, the flooding method is used; therefore, it suffers from the burden of high traffic. Another weakness in this approach is the potential single point of failure (Jafari Navimipour, 2013).

Li (2010) proposed a method for resource discovery in the grid among a series of similar nodes. In this method, each of the nodes is in the control range of a cluster head responsible for a series of nodes similar to each other. The resource is discovered in the two intra-cluster and inter-cluster modes. In this method, to discover resources among clusters, resource-discovery requests are sent in random steps or flooding steps and a series of source routing tables are used to discover the resources within the clusters. This method is highly scalable due to the use of clustering, and if the request is intra-clustered, the traffic is low. Moreover, since it sends inter-cluster requests randomly or in a flooding way, there is no updating cost for routing tables. However, its high traffic due to the sending of flooding requests is seen as a disadvantage.

The Chang (2010) model proposed the use of a bitmap on a tree structure. The author connected the nodes in the environment with the tree structure and used two bitmaps called the “local resource bitmap” and the “index bitmap” and an additional bitmap called the “counter bitmap”. Local resource information is stored in the “local resource bitmap.” In addition, the nodes can notice whether they have the resource requested by the user by checking this bitmap. The other bitmap is the “index bitmap,” which is responsible for keeping resource information for children and generations of a node. This bitmap exists only in nodes, which are not leaf nodes and result from the logical OR between the bitmaps of children of a node. Finally, the “counter bitmap” keeps the number of each resource and is used in the update process.

This technique, which uses a flooding distribution mechanism to distribute resource requests, produces high traffic on the network, and the cost of its resource discovery is high. Also, it does not support the multi-feature resource discovery. That is, each request is only for the discovery of a resource and they cannot respond to requests that involve two or more resources. The advantage of this method is the discovery of all the resources requested by the user. This means there can be more than one resource and this method will discover all of them.

In the FRDT method, like the Chang (2010) method, a tree structure was used to discover the resources, with the difference that weight was considered for each tree edge. Due to the use of a tree with weighted edges,

this method prevents the propagation of requests to nodes due to the lack of a requested resource; therefore, packet traffic is low. The disadvantage of this method is that it discovers only one resource. In addition, it does not support the multi-feature resource discovery system.

Like the FRDT method, in the Prime method, Hadi Mo'tarez used a tree structure and bitmap to discover the resource in the grid environment and tried to find simple factors without the need for complicated computing and little space to store information. This method uses prime numbers so that if there is a resource, its value is multiplied by the value of the edge, which is a prime number, and the result is placed in the parent memory. Then, all the parents have their own information. This method compares the requesting node with its resources when the resource request is sent to a node. If it lacks the resources, it sends the request to the parent node, and if the result of the comparison is not zero, that is, the resource exists in itself or its children; thus the number in its memory is broken up into the corresponding prime numbers. In addition, the path of the node in which the source exists is obtained.

This method can find several resources. Its disadvantage is that by increasing the size of the network and, thus, increasing the depth of the tree, large numbers will be obtained in the data accumulation phase due to the multiplication of the first numbers by each other, which will increase the time complexity of the calculations for the breakdown of the generated numbers into prime numbers. This could potentially lead to the "flooding" of the packet in the upper levels of the tree (Mo'tarez, 2014). The below image shows how to work this algorithm.

```
T:=1;
M:=1;
For i:=1  to  (2*(x-1)) do   // x is the number of attributes
{
        For j:=1 to n do   //n is the number of children
        {
                if   child(j)_bm(i)  !=0  then
                        l:= edge-weight_child(j) * child(j)_bm(i);
                else if   child(j)_bm(i+1)==1 then
                        l:=edge-weight_child(j);
        m:=1 * m;
        }
Nod_bm(i):=l;
i+=2;
}
```

```
child(j)_bm(i)     ================ the child number j's  bitmap
edge-weight_child(j) ============== the weight of child(j)'s edge
Nod_bm(i)    ================== the number(i) bitmap of nod(offspring nod)
```

Fig 1: Prime Method Algorithm

Parisa Vaseghi provided a trust-based approach in 2014 to discover the resources. By adding trust to the binary resource discovery tree, they produced a trusted binary tree. Trust is defined as the competence of an entity and depends on the past behaviour and experiences of an entity. The extent of trust is considered to be an indication of the quality of services for an entity. The main purpose of the proposed mechanism is to discover trustable resources in the grid environment. In this method, by using the trust factor and managing it for each node, criteria like system integrity, reliability, and quality of service are considered for decision-making on resource discovery. The binary tree trust (BTT) algorithm is a reliable method of path selection instead of a short path. One of the problems of this method is the big response time

and the discovery of only one resource (Parisa Vaseghi, 2014).

In the method of Afsaneh Nasrollahi, fuzzy logic and taboo table have been used to discover sources with concepts to search for efficient sources. Resource discovery begins with using the network request to find suitable resources in the Grid. In the proposed architecture, each cluster contains other resources, and also a super factor monitors all the machines to ensure proper operation. Despite the monitoring, the practising physician must ensure the system achieves its maximum efficiency. The primary source acts as a local coordinator, and in the hierarchical model, these local coordinators communicate with a central coordinator (resource requester) for dedicated resources. If requested resources are not within their cluster will not be available, a request will be sent to a central coordinator and the central coordinator will respond to this request. (Nasrallah, 2016). One of the advantages of this method is that using fuzzy logic, a monitoring system is created that monitors the overall performance of the system and each cluster has a local coordinating factor and requests are answered locally. One of the drawbacks of this method is the centralization of the central node, which should be responsible in case of lack of resources in the local cluster.

In SFLA, the population consists of a group of frogs searching for food in a pool. Food searching consists of two processes: the inter-group relation of frogs for location discovery, and the intra-group of frogs that belong to different memeplexes of general evolution. In the proposed method, at each stage, when the node receives a request, it initially looks up for it in the local resources. If the resource is found, a message is sent to the origin node and the task is completed. Otherwise, each node deduces one unit of the TTL, and in case the request is still valid, it sends the request to its neighbours. Thus, the user can use the resources of other machines without getting involved in the detailed aspects of addressing. (Ahmadian, Zavvar, Saeedi, & Ramezani, 2018)

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. 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. The pheromone scent in this method will create a repulsion effect instead of an attraction effect for the other ants. The proposed resource discovery process uses this algorithm through four steps. 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). The best peer is selected to move by the first ant and the pheromone is updated according to Eq. Causing the algorithm to establish load balancing among the peers. Therefore, the travelled route by the previous ant might not be followed by the second ant because the available pheromone in the edge travelled by the previous ant has caused aversion. (Asghari, 2019)

(Wang Tun, 2020) Presented a discovery technique based on the super-peer network. First, create some clusters and organize the grid nodes. In each cluster, a super-peer maintains the summary of resource information that is owned by client peers. They use cobweb to cluster and summarize the resource information. Each super-peer uses the RIs to structure and keep the summary information of peers within the cluster. When a peer joins or leaves the cluster, the RIs entry related to it is created or deleted. Each cluster super-peer sends the maximum probability of each attribute to the neighbour super-peers. When the resource information of a peer changes, that peer sends the updates to its super-peer. Then the super-peer creates the summary again and updates the probabilities of RIs which has higher than the threshold. Next, the maximum probability of attributes is resubmitted to the neighbour super-peers. The neighbour super-peers are selected based on the HRI information and not by random or flooding methods. Consequently, the queries are forwarded to super-peers that are likely to match the query requirements and have nodes in their cluster that own the requested resources.

3. Proposed Method

Since there are different resources in the grid environment and each resource has different attributes, these resources and their attributes must be taken into account in the indexing structure. The proposed method uses two matrices: a local resource matrix and a path length matrix, which is called the surface matrix. For example, figure 1 shows the sample resource matrix in which each room is related to an attribute displayed with a value of 0 or 1.

Unix	Mac	Linx	Windows
1	0	0	1

Fig 1: Local resource matrix in the BTS tree

The “local resource” room can only have a value of 0 or 1, with the value of 1 indicating the presence of the requested resource in the node itself, and a value of 0 indicating the absence of the resource locally. For example, figure 1 shows a bitmap for a node in an environment in which there are four types of operating systems.The following matrix shows the path matrix, in which non-zero numbers indicate the existence of the resource. If the number is 1, it means the resource exists locally in the node, but if it is greater than one, the resource is present in the children of that node.

Unix	Mac	Linux	Windows
3	6	21	0

Fig 2: The surface matrix in the BTS method tree

The numbers greater than 0 and 1, as stated, indicate the existence of a resource in the child nodes. This number indicates the length of the path to that assumed node. Either these numbers are the products of two or more prime numbers, or the number of the node containing the resource is directly placed in it. The parameters of the number of nodes and the number of levels are required for the formation of the

index tree. A single formula is used to create a tree, which is also used in the other methods.

$$\frac{K^{H}-1}{K-1} > N \qquad \text{Relation 1}$$

Relation 1 specifies the number of nodes at each level. In this relation, H and N are the tree height and the number of nodes in the environment respectively. We can consider the smallest number that applies to K in the relation as the number of children of the index services (ISs). For example, in a grid with N = 2,000 nodes, and in a tree with a height of H = 4, the value of 13 is obtained for "k". So, the root node has 13 children. Each of the children of these 13 nodes also has 13 children (a total of 169 nodes at this level), and each of the children of these 169 nodes will have 11 children on an average. Once the tree's topology is created for each number of nodes in the environment, by assigning resources and sending requests to each tree node randomly (these numbers can be obtained from random number generation functions), the number of visited nodes can be simply calculated for any number of requests (Montresor, 2004).In figure 3, an example of a tree after the allocation of resources is observed with four levels. In the leaf nodes, only the local resource matrix is present. However, in addition to the local matrix, there is also a path matrix in the parent nodes, indicating the existence of a resource in the children. The amount of its element implies that a node contains one or more units of a specified resource, and this is due to the features of the prime numbers.

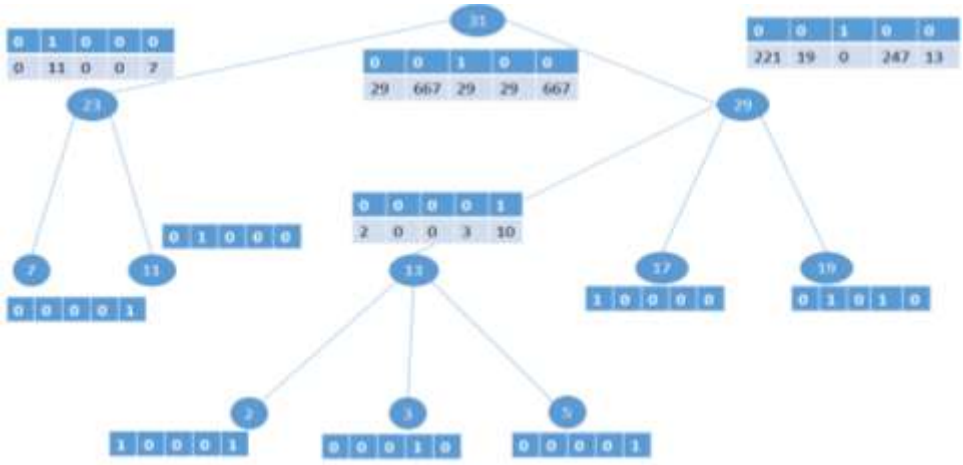

Fig 3: Resource index binary tree with four levels

One of the disadvantages of the hierarchical method is its average scalability compared to the decentralized methods. In this paper, a hybrid method was proposed to solve this problem by taking inspiration from the method presented by Carlo Maestroiani (2005). In the proposed structure, several trees are interconnected with the P2P method and these form the grid structure, as there are some physical organizations, like large

corporation or organization with many branches throughout the country or the world. Its internal structure is hierarchical and each organization is linked to another one. The organizations' relationship is through the root node of each tree. So, if the requests reach the root node, the latter recognizes the presence or absence of the resource using its matrices. If the requested resource does not exist in that organization, the resource request is sent to other organizations available through P2P communication.

The advantage of this method, compared to other hierarchical methods, is that there is no need to increase the size of the tree to increase the scale of the grid, as it reduces the tree's efficiency and the involvement of the nodes. Furthermore, a huge traffic burden is imposed on the network, and the whole system is damaged by the failure of a node. However, in the proposed hybrid method, which will be described in detail below, there is no longer a very large tree. Instead, there are several trees connected with each other and these respond to each other's requests. In each section, the requests first look within their own organization for a resource search and optimization after the resource discovery. If there is no resource in that organization, the requests are sent to the other trees. As a result, a small number of nodes are expected to be involved in this process and to reduce costs, such as traffic, node upgrades after resource allocation, and overall system overhead. To implement a tree and its readiness to enter the resource discovery mechanism, the following steps must be followed:

1) Collecting the information of children by each node

2) Discovery operation

3) Updating information after each assignment.

The data-collection phase begins with the leaf nodes. Figure 4 shows the tree's prototype with local resource information.

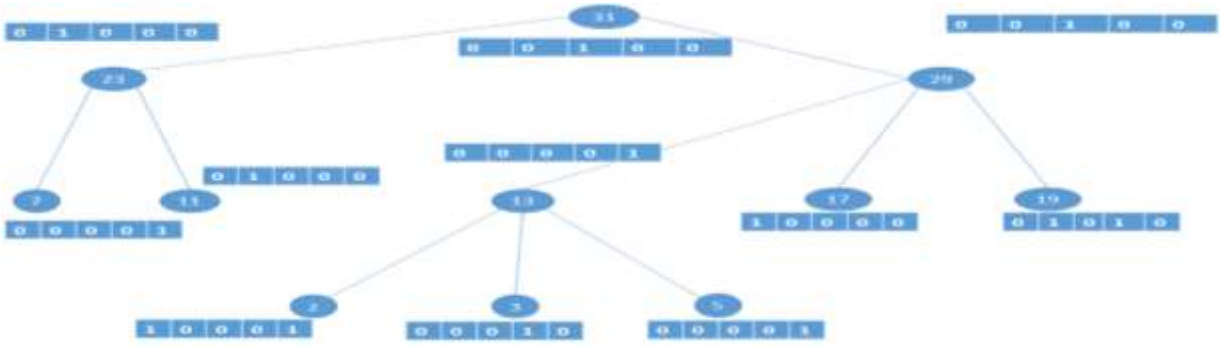

Fig 4: A tree with local resources

In Figure 4, resources are specified in each node and based on these resources, the path matrix is determined by the weight of the nodes. The path matrix in the leaf nodes is equal to the value of the resource matrix. To create the path matrix, one must

move from the leaf nodes upwards. Several states may appear for the computation of the path matrix:

1. The node is a parent and its children are leaves, and the value of the element in the resource parent node is 0 or 1 in the local resource matrix. In this case, the amount of that resource will be examined in the matrix of the children.

a. If the resource exists only in one child, the node number of the resource is exactly inserted in the element of the path matrix.

b. If the resource exists in several children, the values of their nodes, which are all prime numbers, are multiplied, and the result is inserted in the path matrix.

2. The node is a parent and its children are not leaf nodes: it acts as stated in cases a and b shown above, except that if the resource does not exist in the local matrix, but there is a node in its path matrix, it is the same as when the node contains that resource. Figure 5 shows how the matrix is completed on the tree.

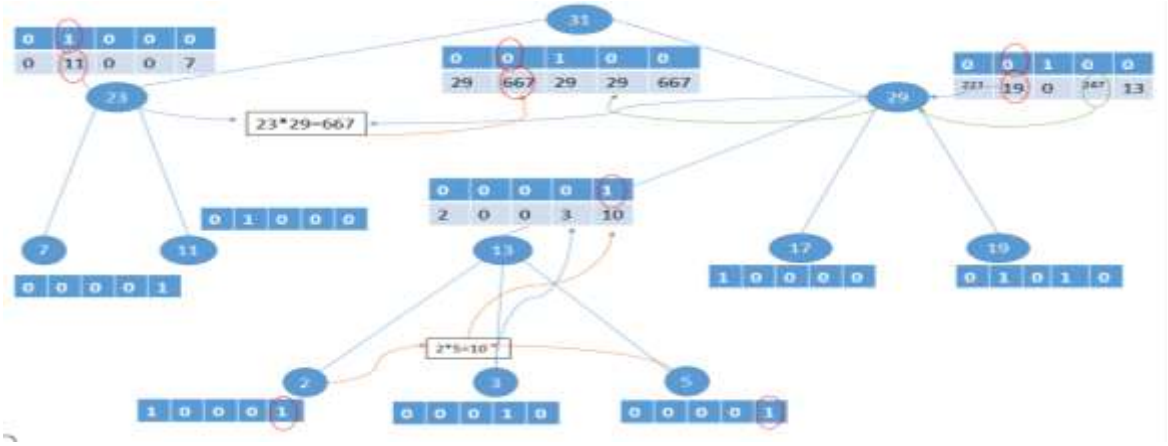

Fig 5: The tree after the information collection phase

To clarify the method of data collection, the process of completion of the first element from the right is described with an example. As mentioned above, the work method proceeds from the bottom to the top, starting with nodes 2, 3, and 5, and assessing the first element from the right, which indicates the existence of the resource. Nodes 2 and 5 have this resource, but these nodes do not have a path matrix. So, the control goes one level higher and reaches node 13, which has this resource. However, for the complete presentation of the nodes at the time of resource discovery, by which the maximum nodes containing the resource can be discovered, the number of children is also examined. Nodes 2 and 5 have this resource, and since there is more than one node, the values of the nodes which are prime numbers must be multiplied, and the product, which is 10, is included in the element of the path matrix. This process reaches node 29, which itself does not have this resource and, by checking the children, finds that this resource exists only in node 13. Since only node 13 contains this resource, it adds 13 to the path matrix. On the left, there are only seven nodes for this matrix, which, in node 23, is referred directly to as node 7. Node 31 does not have this

resource locally, and nodes 29 and 23 indirectly have this resource. Given that the nodes containing the resource are more than one, their values must be multiplied, and 667 = 29 * 23 lies in the element of the path matrix of node 31.

To discover the resource of the request, users must present their requests in the form of a bitmap. To search the resource in a tree, the method used to generate the path matrix is inversely used. For example, if the resource contained in the fifth element is requested and the request is issued from node 7, first, the value of the local resource matrix, and then, the path matrix are checked. Several situations will exist:

1. The value of this element of the local resource and path matrices is equal to zero (in the leaf node, the path matrix is not checked). In this case, the search does not take place in the children, which means that there is no resource in that node. So, the request is sent to its parent node. In the absence of the resource, the request is directed to the upper root node, and if the resource is not found in the root node, the request is sent to other organizations.

2. The value of this element of the local matrix is zero and the path matrix is greater than zero. If its value is equal to a prime number, it represents a node and must refer to that node, and if it is not a prime node, it must be decomposed to the prime numbers and refer to the several nodes to reach the resource.

3. The value of this element of the local matrix is equal to zero and the path matrix is greater than zero. This means that this node itself has the requested resource. The children of this node also have this resource. In this case, it's the same as in situation 2, with the difference that the parent node is prioritized for resource allocation because it has a shorter path.

4. The value of this element of the local matrix is equal to 1 and the path matrix is zero. That is, the node alone has this resource and this resource does not exist in children.

If several nodes have one resource, the node assignment priority is that the least number of nodes are scrolled to reach that node. In the method presented in this study, trees are interconnected with the P2P method and they respond to their neighbouring requests. If the number of requests and resource allocations are increased in a tree, and if a resource that is absent in the tree or is not answered for some reason is requested, the request is sent to other trees to be discovered (each request is sent to adjacent trees and if the adjacent tree does not have the resource, it sends it to its adjacent trees, except for the tree from which the request was received). Once it has received the request, the receiver tree compares it with the values in the local matrix and the root node path matrix and determines whether there is a resource in the tree. The tree that has the least number of visits gets acceptance priority.

After discovering the resource of the request in the system, the original node, which now has the address of the discovered resources, chooses the most appropriate of these resources (the closest resource). In this case, after allocating the resource, the node information must be updated because the allocated

resource has been removed from the list of free resources. The updating process in the proposed method can be done simply and in the shortest time and at the least cost. To do this, we need to consider two cases:

1) The node has local resources and is a root node. In this case, the task can be completed only by assigning the value of zero to the element of the matrix of local resources and the surface matrix.

2) The node has local resources and the nodes of the children also have this resource. In this case, the amount of the element in the local matrix is zero and there is no need for any other change in the tree. That is because in the data collection phase, if the resource is locally available, the children are also checked, and there is no need to change the matrix in the other nodes.

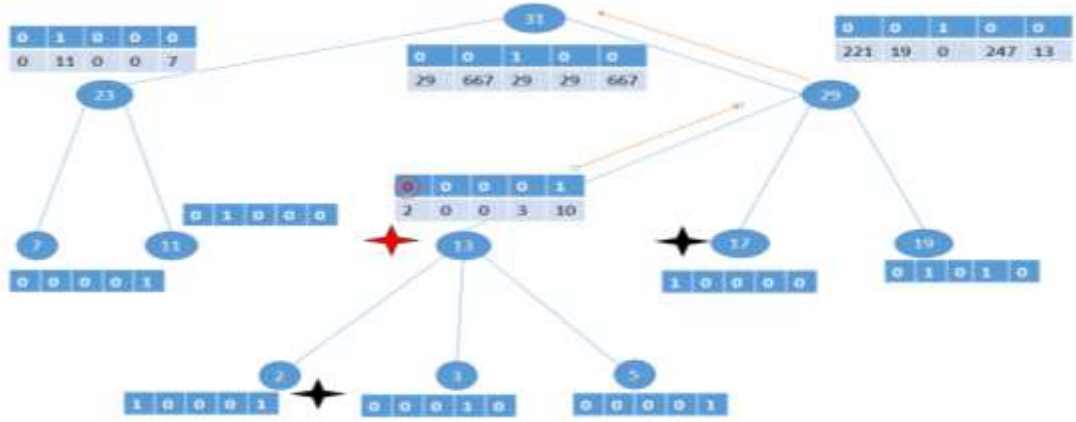

Fig 6: Demonstration of the tree after the resource allocation in the non-root node

In Figure 6, only one node will be updated. The local matrix changes, and since the two nodes still have resources, at node 29, 221 = 17 * 13 is true and does not need to change.

3) It is the leaf node or the parent node, but it does not contain local resources and its children contain the resource. In this case, the matrices should be changed.

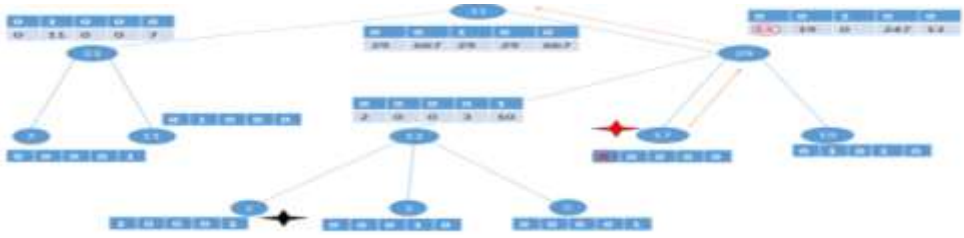

Fig 7: Demonstration of the tree after the resource allocation in the leaf node

In Figure 7, node 17 is assigned. So, its local resource is 0, and its parent node, which was formerly 221 and was obtained by multiplying 17 by 13, becomes 13. Since only a node contains the resource, the link list is used and the number of node 13 is placed in the element. (If two or more nodes still contain the resource, they should be multiplied by each other.) Node 31 does not need to change, and only two nodes will be updated.

4. Results and Evaluation

In the research, a method has been proposed for improving the resource indexing in the tree structure for the grid environment. Considering the features of the grid environment, the large number of resources and nodes, the high dynamics, and resource heterogeneity, it is important to reduce the number of nodes visited in resource discovery and make fewer computations and better use of storage space. In this section, to evaluate the proposed method, the results of implementation and testing were reported in different conditions and compared with the results of the similar and recent methods in terms of the number of nodes visited, the degree of reduction of node updates and the scalability. The evaluation parameters of the proposed method are the number of nodes visited in the resource search process, the update rates of the nodes after the allocation of the resource, and the scalability in terms of the computation rate and local and global search capabilities. The proposed method was compared with other methods and implemented in the Eclipse Neon release (4.6.0) environment in the Windows 10 Pro 64bit operating system. Furthermore, the hardware conditions of the test environment are as follows:

CPU: Core I5 4250 U

RAM: 4 GB DRR3 bus 1600

HDD: 250 GB SSD

The depth of the tree in this simulation is 4, like in the previous methods, and the resource diversity per node is 5. In each test iteration, between 100 and 1,000 resource requests are sent. The resources are randomly distributed among the nodes in the environment, and requests are sent randomly to any of the nodes in the environment. It should be noted that after searching for a resource in the structure of the resource index and finding the first resource consistent with the applicant's request, the search process is stopped and the address of the resource is reported.

To evaluate the proposed method, the results obtained from its simulation were compared with the simulation results of the two similar and recent methods (Khanli, Kargar, 2011; Mo'tarez, 2014) in terms of the evaluation parameters, which are referred to as the FRDT (Footprint Resource Discovery Tree) method and the Prime method in the following sections. The method proposed in the present research is referred to as the BTS (Binary Tree and Super-peer hybrid indexed) method. Table 1 shows the results measured for the product values obtained by calculating the nodal weights in the structure of the resource index tree in different conditions of the number of nodes and the tree index investigation levels for the proposed and the Prime methods. It should be noted that all tree characteristics in terms of the number of children and

the maximum levels, and the rest of the conditions are considered the same in both methods.

Table 1
A comparison of the values of the node-weight products in the index structure of the BTS and the Prime

Method	Tree level	Number of nodes	Node/edge weight
BTS method	Leaf	3	Multiplication is not performed
Prime method	Leaf	3	Multiplication is not performed
BTS method	Parent at level 1	4	10
Prime method	Parent at level 1	4	10
BTS method	Parent at level 2	6	247
Prime method	Parent at level 2	6	273
BTS method	Parent at level 3	10	667
Prime method	Parent at level 3	10	7,917
BTS method	Parent at level 4	11	20,677
Prime method	Parent at level 4	11	254,422

The weights calculated in the nodes/edges of the trees formed indicate that, unlike the BTS method, the weights were expanded exponentially in the Prime method. Figure 8 shows the results obtained for the number of nodes visited by the resource discovery requests in the grid environment, assuming the discovery of the first resource. This simulation was performed for a variable number of nodes and a total of 100 resource requests.

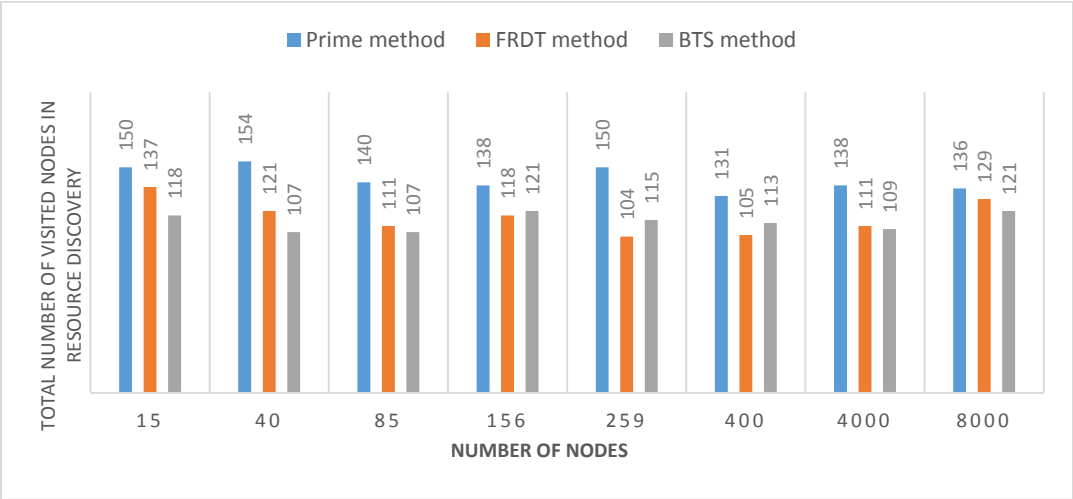

Fig 8: The number of nodes visited by discovery requests for 100 resources

As seen in figure 8, in all the three methods, the results are nearly close to each other and act in the same way. This is because in all the three methods, the exact address of the requested resource exists in the parent nodes, and the request is sent directly to the destination node without the need for visiting additional nodes. This value is slightly lower in the BTS method than in the other methods since it is an improved method, and because of the use of direct addresses as well as the reduction of the tree size.

The result of the implementation of the above simulation with 300 requests of the resource sent by the users is shown in figure 9. It is worth noting that the major difference between the FRDT method and the Prime and BTS methods is in the number of resources discovered, as the Prime method discovers several resources for a request, while the FRDT method can only discover one resource.

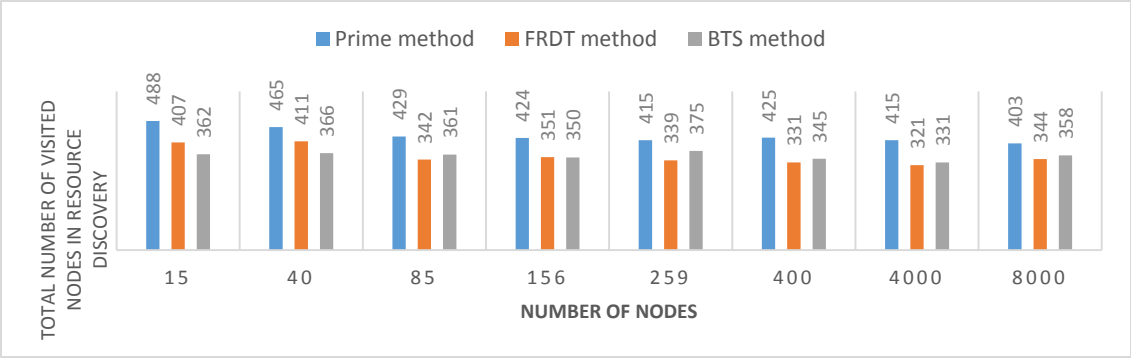

Fig 9: The number of nodes visited for 300 resource requests

As figure 9 shows, by increasing the number of resource requests, the BTS method visits fewer nodes in an environment with fewer nodes. In subsequent implementations, the number of nodes—the figure needs to be updated after resource allocation—has been investigated along with the visited nodes. Figure 10 shows the results of the total number of visited and updated nodes during the request and the resource search process for 300 requests for the proposed and FRDT methods.

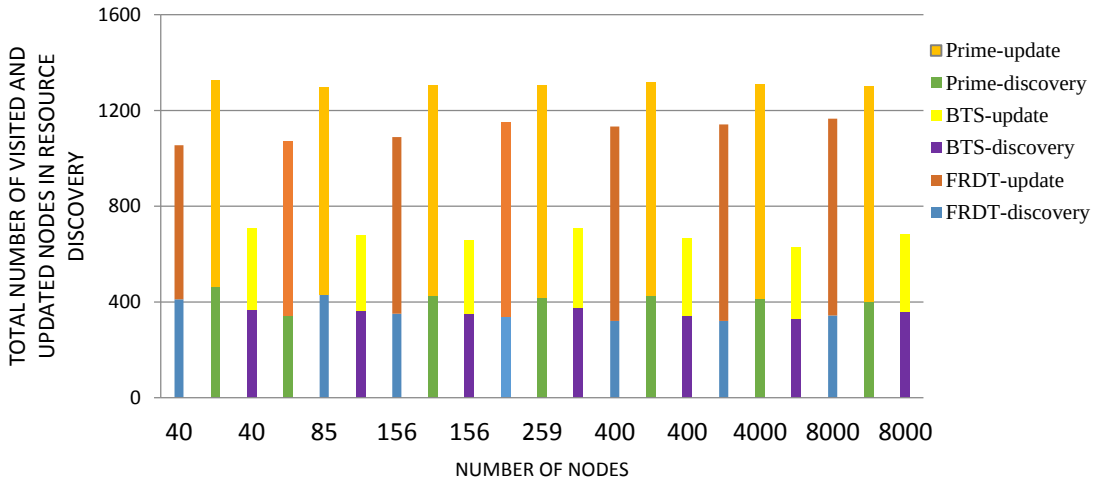

Fig 10: The number of visited and updated nodes for 300 requests for the proposed and FRDT methods

As figure 10 shows, the proposed method works better on node upgrading and involves much fewer nodes in the update process. This is due to the hybrid use of prime numbers and direct addressing that point directly to the node. In the following section, the implementation process for the 1,000 requests is repeated, and as in the previous step, the visited nodes and the nodes for which an update should be performed are reported in figure 11.

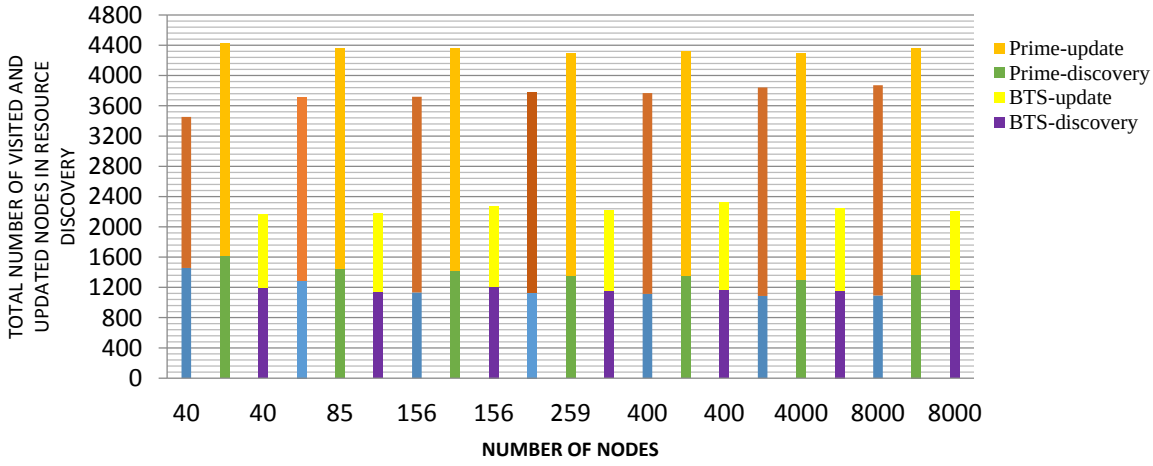

Fig 11: The number of visited and updated nodes for 1,000 requests

As figure 11 shows, with an increase in the number of requests in the grid environment to 1,000, the behavior of the BTS method does not significantly change and, with a great difference, is better in terms of updating the nodes after allocating the resource. Hence, fewer nodes change their information. In the following section, the speed of computations is investigated. In this process, the CPU usage time of each method, until the completion of the resource discovery process and updating of the assigned nodes, is calculated to indicate which method is faster and has a lower processing load.

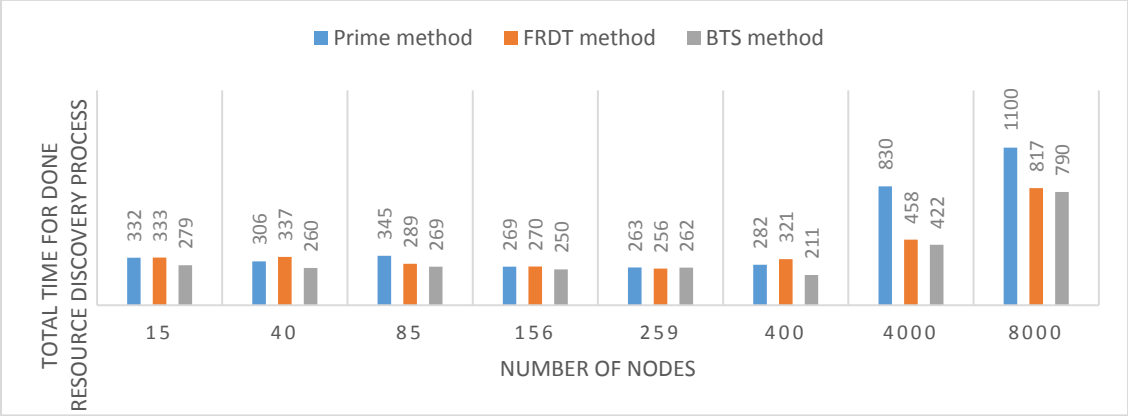

Fig 12: The CPU usage time of each method for 1,000 requests in milliseconds

As figure 12 shows, in the BTS method, the time allocated to the CPU is less than that in the other two methods, due to the combination of the two structures, P2P and trees, in which, unlike the other methods, there are no big trees. Instead, there are several smaller trees with fewer values. This reduces the processing time.

5. Conclusion

In this paper, an improved tree structure was proposed to discover resources in the grid environment. To improve scalability, several trees were used so that there was no tree with a large number of nodes. Rather, these nodes were divided into smaller trees to minimize the cost of discovering and updating. Moreover, an indexing method was used which, in addition to reducing the number of node visits, also shortened the process of updating the nodes after allocating the resource, and provided higher scalability than the Prime method. The weight of the nodes in the proposed method was smaller than the Prime method. Furthermore, none of the previous methods supported the relationship among the trees. The results indicated that the proposed method has better scalability, and the number of nodes visited and the updates of the nodes after the allocation of the resource in it decreased compared with other methods.

Refrences

[1] Adriana Iamnitchi, I. F. (2001). *On fully decentralized resource discovery in grid environments.* Springer Berlin Heidelberg.

[2] Alireza Souri, N. J. (2013). Behavioral modeling and formal verification of a resource discovery approach in Grid computing. *Expert Systems with Applications-41*, 3831-3849.

[3] C. Mastroianni, D. T. (2007). Evaluating resource discovery protocols for hierarchical and super-peer grid information systems. *Parallel, Distributed and Network-Based Processing, 2007. PDP'07. 15th EUROMICRO International Conference on* (pp. 147-154). IEEE.

[4] Caminero, A. C.-G. (2013). P2P based resource discovery in dynamic grids allowing multi-attribute and range queries. *Parallel Computing, 39(10)*, 615-637.

[5] Carlo Mastroianni, D. T. (2005). A super-peer model for resource discovery services.

[6] Chang, R.-S. H.-S. (2010). A resource discovery tree using bitmap for grids. *Future Generation Computer Systems. (۲۶)*, 29-37.

[7] Foster, I. (2004). Brain Meets Brawn: Why Grid and Agents Need Each Other. *AAMAS '04 Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1* (pp. 8-15). Washington, DC: IEEE Computer Society.

[8] Hawa, M. A.-S.-A. (2013). On enhancing reputation management using peer-to-peer interaction history. *Peer-to-Peer Networking and Applications*.

[9] Iamnitchi A, F. I. (2002). A peer-to-peer approach to resource location in grid environments.

[10] Jafari Navimipour, n. A. (2013). Resource discovery mechanisms in grid systems: A survey. *J. Network and Computer Applications Vol. 41*, 389-410.

[11] Jennings Nicholas. (2001). Anagent-based approach for building complex software systems. *Communications on ACM*.

[12] Kakarontzas G., S. I. (2006). Agent-based resource discovery and selection for dynamic grids.

[13] Khanli, A. K. (2011). FRDT: Footprint Resource Discovery Tree for grids. *Future Generation Comp. Syst. Vol. 27 No. 2*, 148-156.

[14] Kovvur RMR, K. V. (2010). Adaptive resource discovery models and resource selection in grids.

[15] Li, J. (2010). Grid resource discovery based on semantically linked virtual organizations. *Future Gener. Comput. Syst. (۲۶)*, 361-373.

[16] Mahdi MollaMotalebi, A. S. (2013). C-HLRD: a category-based hierarchical localisation technique for resource discovery in grid environments. *International Journal of Web and Grid Services, 9*(3), 268–286.

[17] M.MollaMotalebi, R. (2014). THE EFFICIENCY CHALLENGES OF RESOURCE DISCOVERY IN GRID ENVIRONMENTS. *Cybernetics and Systems* , 671-692.

[18] Marín Péreza, J. B. (2012). utureGenerationComputerSystems. *Future Generation Computer Systems*, 40-55.

[19] Min Cai, M. F. (2003). MAAN: A Multi-Attribute Addressable Network for Grid Information Services. *GRID '03 Proceedings of the 4th International Workshop on Grid Computing* (p. 184). Washington, DC, USA: IEEE Computer Society .

[20] Montresor, A. (2004). A robust protocol for building superpeer overlay topologies. *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on* (pp. 202-209). Bologna Univ., Italy: IEEE.

[21] motarez, h. (2014). Tree resource discovery using prime numbers. *Second information technology conferences* (pp. 40-48). tabriz: islamic azad university of tabriz.

[22] Nabila Chergui, S. C. (2010). Semantic Grid resource discovery based on SKOS ontology. *International Journal of Grid and Utility Computing, 8*(4), 269–281.

[23] Parisa Vaseghi, A. H.-R. (2014). BTT: Representing the Binary Tree Trust Mechanism For Resource Discovery in Grid. *IEEE.* mashhad: IEEE.

[24] Qiao Yi, B. F. (2006). Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use.

[25] Talia, D. .. (2005). Peer-to-peer protocols and grid services for resource discovery on grids. *Advances in Parallel Computing*, 83-103.

[26] Tan, Y.-H. L.-P. (2012). Organisation and management of shared documents in super-peer networks based semantic hierarchical cluster trees. *Peer-to-Peer Networking and Applications*, 292-308.

[27] Tangpongprasit, S. .. (2005). A time-to-live based reservation algorithm on fully decentralized resource discovery in Grid computing. *Parallel Computing 31(6)*, 529-543.

[28] torkestani, a. (2012). A distributed resource discovery algorithm for P2P grids. *Journal of Network and Computer Applications*, 2028-36.

[29] Y Deng, F. W. (2009). Ant colony optimization inspired resource discovery in P2P Grid systems. *The Journal of Supercomputing*, 4-21.

[30] Zhong Liu, C. Z. (2005). Decentralized Grid Resource Discovery Based on Resource Information Community. *Journal of Grid Computing.springer*.

[31] Afsaneh Zargar Nasrollahi, Ali Asghar Pourhaji Kazem Resource discovery in Grid computing using Fuzzy Logic and Tabu Table [Journal] // IJCSNS International Journal of Computer Science and Network Security. - 2016. - p. VOL.16 No.9.

[32] Ahmadian, A., Zavvar, M., Saeedi, A., & Ramezani, R. (2018). Resource Discovery in Non-Structured Peer to Peer Grid Systems Using the Shuffled Frog Leaping Algorithm. Journal of Telecommunication, Electronic and Computer Engineering , 10, 4.

[33] Wang Tun, J. P. (2020). Optimizing Resource Discovery Technique in the P2P Grid Systems. Wireless Communications and Mobile Computing , 1069824, 9.

[34] Asghari, S., Navimipour, N.J. Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm. Peer-to-Peer Netw. Appl. 12, 129–142 (2019). https://doi.org/10.1007/s12083-018-0644-2