A New Approach to Optimize Time-Cost in Grid Resource Allocation

Narjes Khatoon. Naseri, Amin. Jula

Abstract—Increasing complexity and performance enhancement requirements are nowadays characteristics of scientific problems. Hence, GRID systems creation and improvement are in attention very much. What leads to enhancement the Quality of Service in grid systems is Time-Cost optimization in resource allocation to execute tasks entered into the system. In the most researchs which published their results in literature time optimization has just been considered or cost optimization. In this paper a new approach is proposed to optimization tasks processing time and cost simultaneously. Afterward, an improved gravitational attraction search algorithm has been applied to solve Grid Resource Allocation problem. At the last part, the results of proposed algorithm execution and basic gravitational attraction search algorithm has been compared and improvement in execution time of algorithm and obtaining more appropriate results in less time has been emphasized.

Keywords—Grid, Resource Allocation, Gravitational Attraction Search, Optimization.

I. INTRODUCTION

GRID Computing was suggested as substituted for usual super computers in the last decade 1990 to solve special problems which required mass numerical calculations and get access to a higher volume of the distributed data. The main purpose was that the fast enough Networks that are geographically dispersed, using suitable software, share their processing resources and data resources management into a single system to face with problems which each of these networks alone are not able to deal with [1]. Today increase in the efficiency of grid is a considerable problem which requires resource allocation efficiency and accurately. Jobs enter to grid are different and in proper workload. Deal with these different conditions needs different strategies, resource allocation and scheduling algorithms such as FPLTF, WQR, RR, Max-min, min-Max, FCFS [2]. Grid resource allocation problem focuses on response time. The purpose of resource allocation is optimal assigning jobs to resources.

Boyya introduced an economic framework for grid in 1995[3]. Due to this framework, users have to pay financial cost of using resources to their owners. It leads to motivate resource owners to share their resources. Since then, more resource allocation algorithms have considered cost and economic profit in the objective function [4]. But It is noteworthy that these algorithms applied different and separated policies and objective functions for optimization the finishing time and execution cost of tasks.

On the other hand, due to the increased dimensions of problems and the importance of speed to get to the proper responses, classical methods are not advanced enough to solve the problems and the random algorithms are mostly use to solve problems. Due to these circumstances heuristic search algorithms have gained noticeable growth. Heuristic algorithms are those which have come through inspiration of physical and biological processes in nature and most of them act as group population. Genetic Algorithm is an example of this type which has been inspired by the science of Inheritance and evolution 1975. Simulated annealing algorithm has been inspired by thermo dynamics 1983, immunity algorithm with defensive human system simulation 1986, algorithm for searching ant colony by making use of ant behavior to search for food 1991, and particle swarm optimization through imitating social behavior of birds 1995 are just a few examples to be mentioned [5-9].

In the same direction gravitational attraction search algorithm was offered by Webster through the inspiration of Mass and energy of gravitation force based on simulation of existing laws in nature [10]. The issue will be deal with in part three in details.

II. GRID RESOURCE ALLOCATION PROBLEM

Grid resource Allocation Problem consists in allocating distributed grid resources to heterogeneous tasks. Users introduce and send their tasks and their processing information to a grid broker. Generically, a task can be anything that needs a resource, a schedulable computation, a bandwidth request, data access or an access to any remote resource, such as remote instruments, databases, etc. A resource is anything that can be allocated to tasks, a processor, disc space, bandwidth, machine, device, person, etc.

In This paper grid resource allocation will be studied to optimize by improved gravitational attractions search algorithm. The structure of the paper will be continued like that in part two, grid and its resource allocation will be reviewed. Gravitational Attraction Search Algorithm has been introduced in part three, and in part four, the suggested algorithm will be offered, and finally in part five the results of proposed algorithm will be described and part six constitutes general conclusions. Grid resource Allocation Problem

Grid resource Allocation Problem consists in allocating distributed grid resources to heterogeneous tasks. Users introduce and send their tasks and their processing information...
to a grid broker. Generically, a task can be anything that needs a resource, a schedulable computation, a bandwidth request, data access or an access to any remote resource, such as remote instruments, databases, etc. A resource is anything that can be allocated to tasks, a processor, disc space, bandwidth, machine, device, person, etc. In this research, we consider grid resources as computational resources. Computational resources are all resources needed for computational tasks to be computed; they are not just processors, but also memory, disc space, various devices, etc [11].

At beginning of allocation process, the broker checks dynamic information of resources including number of processors, type of process, etc to be informed of last situations [12, 13].

Following this process the resources will be scheduled and allocated to tasks due to allocation policies and tasks' processing requirements.

After definition the Economic model for Grid, execution cost of applying resources added to the concern of finishing time of processing.

Since the cheapness and process speed are two opposite parameters, the broker has to apply user's decisions to allocate resources to tasks. For example broker should allocate cheaper resources to the tasks provided that the user is concerned about process cost and if the user is concerned about his finishing time, the broker has to assign fast resources for him.

III. GRAVITATIONAL ATTRACTION SEARCH ALGORITHM

A. Main Idea

The search space of a problem is assumed as a multidimensional system with different solutions to the problem. Each point in space is a unique solution to solve the problem, and each solution possesses a "mass" through which the objective function will be computed. Any solution is better, more objective function value is generated, and thereby its mass will be more. Besides, search agents are collection of particles. After constituting search space, its rules will be recognized to govern it, with the assumption that only gravitation and motion laws govern it.

Gravitation law: Each particle in search space will attract other particles toward itself. The amount of this force is proportional to the gravitational property of the particle and inversely to the distance between the two particles.

Laws of Motion: the real velocity of every particle is equal to the sum of a coefficient of the previous velocity of the particle and the change of its velocity. Change in velocity, acceleration, is equal to the imposed force on the particle divided to its gravitation mass.

B. Gravitational Attraction Force

The search space is considered as a set of m particles. Position of each particle in search space is a point in space and is taken in account as a solution to the problem. This position is obtained from equation 1 in which position of particle i in dimension d is shown by \( x_i^d \).

\[
X_i = (x_i^1, ..., x_i^d, ..., x_i^D)
\]

In this system in time t, a force is imposed from particle i to each particle j in the direction of dimension d as much as \( F_{ij}^d(t) \). The amount of this force is computed from equation 2 in which \( G(t) \) is gravitation constant in time t, \( R_{ij}^d(t) \) is distance between two particles i and j in that time and \( \varepsilon \) is a small number. To obtain distance between the particles Euclidean distance has been used as Equation 3.

\[
F_{ij}^d(t) = \frac{G(t) \times M_i(t) \times M_j(t)}{R_{ij}^d(t) + \varepsilon} (x_j^d(t) - x_i^d(t))
\]

\[
R_{ij}^d(t) = \|X_i(t), X_j(t)\|_2
\]

The imposed force on the particle i in direction of dimension d in time t, \( F_{ij}^d(t) \), is equal to the total amount of the forces of other particles of the system imposed to it, Equation 4.

\[
F_i^d(t) = \sum_{j=1, j \neq i}^{m} r_j F_{ij}^d(t)
\]

According to second Newton's law, each particle gains acceleration in direction of dimension d, and this acceleration is proportional to the imposed force on the particle in the same dimension divided by the particle gravitational mass. The particle acceleration in direction of dimension d in time t is shown by \( a_i^d(t) \) and it obtains from equation 5.

\[
a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}
\]

Velocity of each particle is equal to the sum of a coefficient of the present velocity of the particle and acceleration of the particle that is obtained from equation 6. The new position of the particle i in the dimension d is equal to the sum of its present position and its velocity that computes by equation 7.

\[
V_i^d(t + 1) = r_i \times V_i^d(t) + a_i^d(t)
\]

\[
x_i^d(t + 1) = x_i^d(t) + V_i^d(t + 1)
\]

\( r_i \) and \( r_j \) are the random numbers by uniform distribution in the interval of \((0, 1)\) which have been used to keep the random property of search. Equation 8 is used in order to adjust the gravitation constant. Gravitation constant tends to decline exponentially.

\[
G(t) = \beta^{-\frac{1}{\gamma}}
\]

IV. PROPOSED ALGORITHM

A. Assumptions considered to calculate Mass of solutions

- If a user defined conditions that the broker cannot satisfy them, tasks will be return to the user without any processing.

164
There is just one task assigned to a resource at a time and other tasks can be assigned the resource after finishing current running task.

- Grid has \( N \) nodes which each node has \( j \) resource.
- Each resource has its cost that can be calculated by Equation 9.

\[
Cc = \sum_{i=1}^{j}(nR_i \times C_i) + AccessCost
\]

Where \( Cc \) is the cost for using a Node, \( n \) is number of resources in the node and \( AccessCost \) is accessing cost of the node.

- The market is a competitive market. For this reason it is important to apply best resource assigning to satisfy the user.

\section*{B. Mass Calculation}

According to presented definitions and suppositions in previous parts equation 14 is proposed to calculate the cost of processing tasks which means mass in the algorithm. This equation composed of two parts that each of them computes a part of the mass of a solution. Eventually the sum of these two parts determines the mass of solutions. Mass of Processing-Cost is calculated by first part which is shown in Equation 10. In this equation price proposed by the customer (Cc) is divided into the price calculated for a resource (Cs) and the result is multiplied by the price priority (Pc).

\[
PROCESS \_COST = \frac{Cc}{Cs} \times Pc
\]

\[
TIME \_COST = \frac{Tc}{Ts} \times Pt
\]

Dividing \( Cc \) into \( Cs \) leads to generating bigger number for calculating the mass in the case of the \( Cc \) being smaller than \( Cs \).

Calculating Time-Cost being done using Equation 11. In this equation time proposed by the customer \( (Tc) \) is divided into the time estimated for a resource to execute the tasks \( (Ts) \) and the result is multiplied by the time priority \( (Pt) \).

\[
\text{Mass}(i) = \text{PROCESS \_COST}(i) + \text{TIME \_COST}(i)
\]

Due to what mentioned above, if a customer declares one of priorities equal to \( \%100 \), another one will be \( \%0 \) and this situation is not appropriate in the case of equality the mass of two different solutions because it is not guaranteed that two mentioned solutions be equal in terms of their processing performance. Hence, in these cases it is important to participate the time of execution contrary to what the customer wanted but this effectiveness should not be high.

Accordingly, Equation 11 and Equation 12 will be applied to calculate the mass of solutions with a slight change. To avoid each of two priorities from being zero, it is important to add a small number like "1" to two priorities and then participate them to calculate the mass. Hereupon, the equation will be as shown in Equation 13.

\[
\text{Mass}(i) = \left( \frac{Cc}{Cs} \right) \times (Pc + 1) + \left( \frac{Tc}{Ts} \right) \times (Pt + 1)
\]

\section*{C. Applied Algorithm}

Initially, each job is allocated to a processor at random in such a way that each processor will have one or more jobs for execution. At this time one solution produced to solve the problem, and in the same way \( SolNo \) different solutions will be generated at random to solve the problem. Then mass of each particle will be computed and recognized mass of the related particle. If we call \( T_i \) the required time to execute all the appropriate jobs to processor \( i \), mass of solution will be the largest \( T_i \). The procedure of computing mass of particle is reflected in Fig. 1.

\begin{figure}[h]
\centering
\includegraphics[width=\linewidth]{Fig1.png}
\caption{Calculating mass of solutions Pseudo code}
\end{figure}

Finally, the solutions sort descending based on mass and the solution with the largest mass as in the Equation 14. will be recognized as the best. Since equation \( "a" \) has been known as the best, with the minimum amount of time for the biggest jobs in gravitational method, in must be related to the biggest particle.

\[
best(t) = \max_{j \in \{1, \ldots, m\}} fit_j(t)
\]

As the best solution, a solution with minimal time to execute jobs and in gravitational attraction search the best solution is solution with largest mass, we can subtract the calculated number for mass from a fixed number and thus it will be a solution with minimum time for execution has maximum amount of mass.

Focused on gravitation search algorithm we realize that, due to being too improper solutions than suitable solutions, sum of gravitational masses of improper solutions exceeds the sum of
gravitational masses of proper solutions, and consequently improper solutions attract proper solutions and proper solutions will be lost. We suggest that for a limited number of ideal solutions explain a field named "Virtual Mass". The amount of virtual mass of each solution will be defined as the basis of its quality compared with other solutions and its rank in the ordered set of solutions. The calculating method for virtual mass is reflected in Fig. 2.

Fig. 2 Virtual mass pseudo code

Where, \( \alpha \) is a real random number in range (0, 1), \( k \) is rank of solution in the sorted set, \( \text{NumK} \) is number of solutions that virtual mass should be calculated for them. This increases the gravitational mass of solutions and as a result, improper solutions will be absorbed by them. However, determining the number of solutions involved in this is very important. To dynamic determining the number of solutions that should have virtual mass Roulette Wheel algorithm has been used.

This algorithm generates a random number between 1 and sum of masses of all solutions. Then, to achieve this random number, algorithm attempts to calculate sum of masses of solutions starting from the best solution. Achieving the desired number in calculating the total condition will finish this process.

V. ALGORITHM EXECUTION RESULTS

In this part we will deal with the execution results of proposed algorithm which has been suggested and the classic GAS for some sample experimental problems which are executed on personal computer under identical conditions. The procedure was that the proposed algorithm executed five times to solve the same problem and their average calculated. Then the same problem solved through classic GAS and their average calculated. Afterwards From the average of results prepared charts for comparing performances.

The first problem is a resource allocation with twenty processors and forty jobs. The problem can be assumed as a simple resource allocation problem, but the results obtained in their comparison of the two algorithms are very effective and noticeable.

All used runtimes in this problem are created at random by computer and in each execution more than 200 iterations have been operated and the results have been recorded. The obtained results reflected in fig. 4 show that proposed algorithm has been quickly obtained the proper solutions after 38 iterations and solutions are converged on it, while the classic GAS tries to reach more suitable solutions through gradual slope in the curve.

The other problem which is used for the comparison of proposed algorithm and classic GAS has been solved is a grid resource allocation problem with fifteen processors and sixty jobs. All used runtimes in this problem are created at random by computer like the previous problem. Each execution is stopped after 200 iterations and the results have been recorded. The difference between the former and later is increase number of jobs and reduction of processors which brings about a more difficult problem.

The results are reflected in fig. 5 as a chart. Studying the chart shows that the classic GAS after a few iterations gets involved in the trap of local optima and obviously local optimization prevents us from finding real solutions, while the proposed algorithm continues finding better solutions steadily after execution 200 iterations with an acceptable slope and doesn’t fall in the trap of local optima.

VI. CONCLUSION

In this paper, a new approach is proposed to calculate and evaluate presented solutions for grid resource allocation based on the time and cost of using resources simultaneously. Thereafter, by applying some different improvement to the gravitational attraction search algorithm an improved algorithm proposed to solve the problem which is away from impediment of trapping in the local minima of problem search space.

Suggested approach in calculating and evaluating solutions is applied in objective function of proposed algorithm. Improvements imposed in gravitational attraction search algorithm led to attaining best solutions in less time in solving the grid resource allocation problem.

Fig. 3 Comparison of two algorithms running on the problem with 20 processors and 40 jobs

Fig. 4 Comparison of two algorithms running on the problem with 15 processors and 60 jobs
Fig. 5 Comparison of two algorithms running on the problem with 30 processors and 70 jobs

REFERENCES