Combined Optimization in Solving the Task Scheduling Problem of Grid Systems based on The Proposed Combined Method

Maryam Fattahi,

Master of Computer-Software

Abstract

A computational grid is a set of large-scale heterogeneous systems. Job scheduling is one of the most important issues in a distributed grid system. Genetic Algorithm as a basic evolutionary algorithm, is a fair way to solve difficult problems that have failed with traditional conventional methods to achieve the desired results. This algorithm has been able to achieve acceptable results for the task scheduling problem in the grid system. In this research, the scheduling of tasks in the grid system is examined in a proposed way. Proposed method to create new evolutionary algorithms resulting from combining genetic algorithm with particle mass optimization algorithm (PSO-GA), genetic algorithm with colonial competition algorithm (GA-ICA), genetic algorithm with refrigeration simulation algorithm, SA-refrigeration algorithm (GA) Genetics deals with ant colony optimization (GA-ACO) to schedule distributed optimizations in the grid system. The main purpose of the hybrid algorithm is improving the local search process of the genetic algorithm (GA) by combining it with the evolutionary algorithms ICA, PSO, SA and ACO, preventing premature convergence and stopping at the local minimums and ensuring global optimization. Considering the time process of workflow and task execution time of works as comparison criteria, finally the proposed hybrid algorithm was able to achieve the desired results for the task scheduling problem in the distributed grid system.

Keywords: Hybrid algorithm, optimization, task scheduling, task execution time.

Introduction

The basis of grid computing was developed in 1980s and 1990s by Foster et al. The term grid was raised to provide a distributed computing infrastructure to meet the growing needs of science and engineering. The Significant improvements have been made in the structure of this system so far. Also, the term grid itself has been combined with other sciences- at least in general perception- over the years, and includes advanced networks to artificial intelligence.

Due to the increasing expansion of grid network users, the consequent increasing in the volume of user requests and to access various resources in the network, the implementation of a grid system to facilitate the task scheduling in the grid is of particular importance. In this respect, task scheduling, that is a difficult issue, is known as a challenging issue in the grid computing environment.

In general, it is not easy to find the optimal scheduling for such an environment using conventional traditional methods. While using metaheuristic methods, near-optimal solutions to this complex problem, will be easily created.

An overview of previous work

In a study [1], a Hybrid Particle Swarm Algorithm and a Hill Climbing Algorithm for Scheduling Task in Cloud Environments were investigated. The results showed that the convergence speed of the solutions in population-based algorithms is low, they are integrated with local search algorithms. Therefore, a Hybrid Particle Swarm Algorithm and a Hill Climbing Algorithm are proposed in this paper. Experimental results on the non-cyclical diagram of random and scientific directing (DAG) showed that the proposed algorithm works efficiently in terms of fabrication compared to the currently known algorithms of exploration and particle swarm optimization.

In a study [2], the task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with several QoS (service quality) limitations in cloud computing were investigated. In this study, the researcher combined the ant colony optimization algorithm (ACO) with the genetic algorithm (GA). GA was recalled to produce the initial pheromone for ACO efficiently. With the designed fitness function, the next QoS goals4 are evaluated. ACO is then used to search for the optimal source. This experiment shows that the proposed algorithm has a preferential performance in both resource balancing and service quality assurance.

In the study [3], planning in network systems was investigated using the ant colony algorithm. Task scheduling is an important factor that directly affects the performance and efficiency of the system. Network computing uses

heterogeneously distributed resources to support complex computational problems. Network can be classified into two types: computational network and data network. Job scheduling in the computing network is a very important issue. To use networks efficiently, we need an appropriate task scheduling algorithm to allocate jobs to resources on the networks. In this paper, a new algorithm based on cross-sectional ant colony optimization (ACO) is proposed to solve this problem. In this study, the proposed ACO algorithm for programming in Grid systems will be presented. The results of simulation show that our ACO algorithm optimizes the overall response time and also increases usage.

In a study [4], the PSO-based task scheduling algorithm was performed for network computing. Network computing is a high-performance computing solution to meet computational demand on a larger scale. Network computing includes resource management, task scheduling, security issues, information management, etc. Task scheduling is a key issue in achieving high performance in network computing systems. However, it is a big challenge to design and implement an efficient scheduling algorithm. In this paper, an innovative method based on particle swarm optimization algorithm is adopted to solve the work planning problem in a network environment. Each particle is a possible solution, and the position vector is converted from a continuous variable to a discrete variable. The purpose of this approach is to create an optimal program so that it can get the minimum task execution time in completing tasks. The results of the simulated experiments show that the particle swarm optimization algorithm is able to obtain a better program than the genetic algorithm.

In a study [5], network performance was performed based on particle swarm optimization. The task scheduling problem is a complete NP problem. This paper deals the problem of task scheduling in a network environment and proposes a task scheduling mechanism, which defines each possible scheduling scheme as a Resource Allocation Graph (T-RAG) and therefore turns the task scheduling problem into the problem of choosing the optimal graph. In addition, in order to find the optimal solution quickly and accurately, a task scheduling algorithm was proposed based on particle swarm optimization (PSO). This algorithm considers the longest path of the task-resource allocation graph as a fitness value and encodes each task-resource assignment as a particle. Finally, the experiment shows that the approach presented in this paper is effective for solving the problem of task scheduling. Each of the studied algorithms has led to the optimization of the completion time in different researches, so in the following, the integration of these methods in the optimization of the completion time in grid systems will be investigated.

Introduction of proposed hybrid algorithms

Proposed GA-PSO hybrid algorithm

In this proposed algorithm, the particles are the answers obtained from the best processor mapping matrix on the workgroups generated by the genetic algorithm. And population forms a particle mass optimization algorithm. Each of them then receives a speed based on the processing speed that processors receive (code 8-2-4).

$$[v_{pt}] = [cpu - speed] * [job - group] \quad (1)$$

Each particle with the x-pt position vector in the mapping matrix has a velocity vector v-pt, and the velocity vector in them depends on the number of processors. Then a new speed for particle work based on the current particle speed and the best position ever seen and the best position found by adjacent work particles. Given the amount of speed obtained, the new position of the particle can be obtained in the next stage after updating the position. This algorithm stops when the maximum number of iterations (ie 100 times) is created to achieve an acceptable response, so that there is no significant improvement in the task execution process and finally, the best meeting place for all particles is presented as the response to the problem.

The best position for each pbest particle,is the workgroup on resources so that it has the shortest task execution time (cost function), and for all gbest particles, the best position of the whole work is on the resources which have the shortest task execution time. Equation (2) shows the new velocity of the i-th particle.

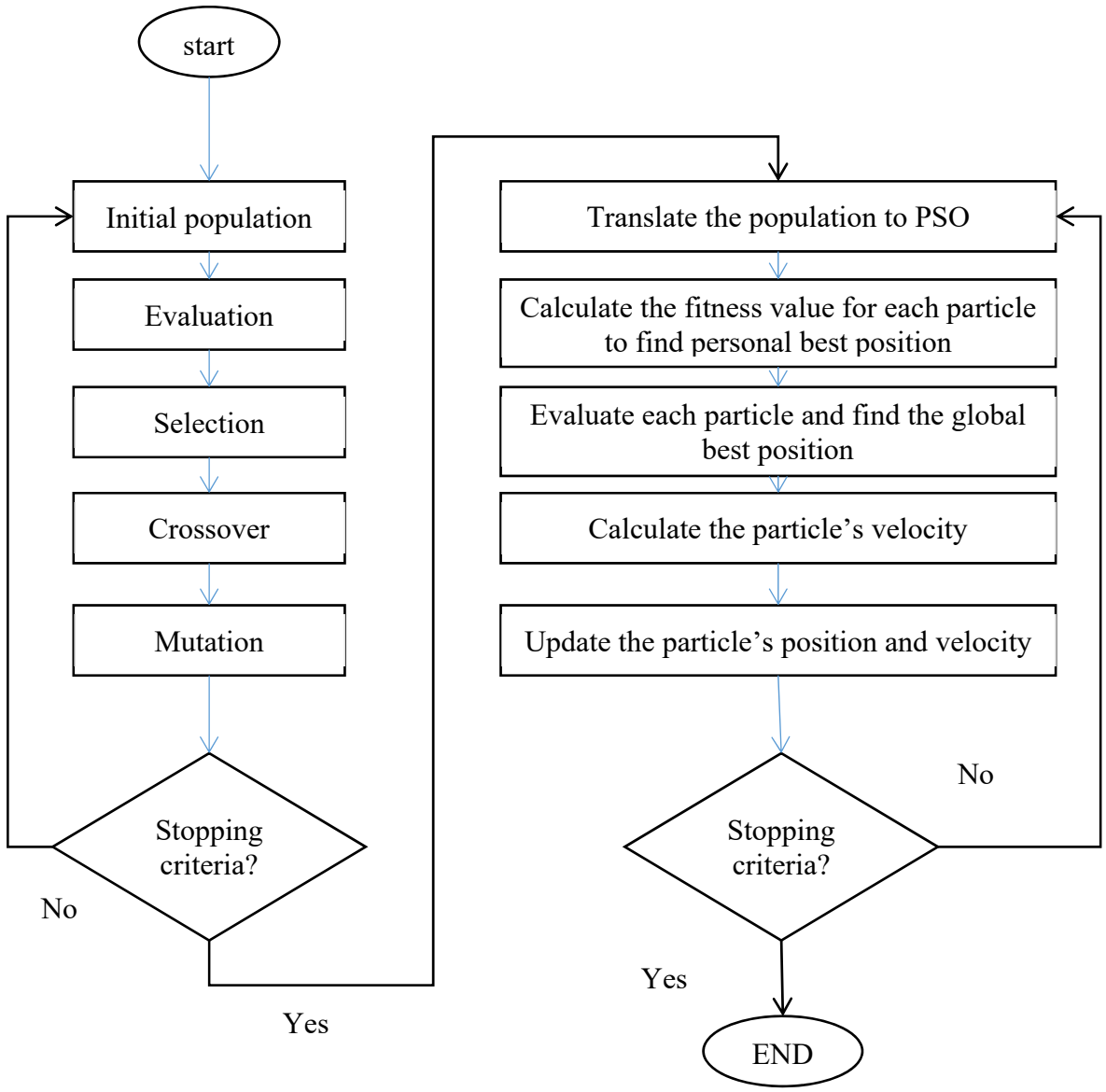
$$v_{pt+1} = w_t * v_{pt} + C_1 * rand(\ ) * (pbest - X_{pt}) + C_2 * rand(\ ) * (gbest - X_{pt}) \quad (2)$$

The positive constant coefficients are C1 = C2 = 2 and the coefficient of inertia (w_t) in formula (3), shows the effect of the previous particle velocity vector on the new vector.

$$w_t = w_{hi} - ((w_{hi} - w_{low})t \,/\, T_{max}) \quad (3)$$

That $w_{low} = 0.1, w_{hi} = 0.9$ are the highest and lowest inertia coefficients, respectively and the maximum number of iterations allowed is $T_{\_max} = 100$, , and formula 4 shows the position of the i-th particle.

$$X_{pt+1} = X_{pt} + V_{pt+1} \quad (4)$$



Proposed hybrid genetic algorithm - colonial competition

The initial responses in this matrix algorithm, generated from the best matrix processor mapping on workgroups in the genetic algorithm, the population of the second algorithm- countries- produce colonial competition in the algorithm. The top 5 processors are selected as the imperial set. The imperialists have the shortest completion time in evaluating the objective function. The rest of the processors are also the colonial councils. The goal of optimization with the imperialist matrix is to find an optimal solution, ie an imperialist with a minimum point as a function of the cost target that has the shortest time to complete tasks. The imperialists, depending on their power, are pulling these colonies towards themselves in a certain way, which will be discussed later.The total power of any empire depends on both of its constituent parts (the imperialist state) as the core (and its colonies). Thus, the total cost of an empire, that determines the time of completion of work in this situation, is calculated based on the following equation.

$$T.C.n = Cost(imperialistn) + \zeta\ mean[Cost(colonies\ of\ empire\ n)] \quad (5)$$

In this regard $\{T.C.i\max_i\}$ is the maximum task execution time, which is actually the cost of the empire per replicate $T.C._n$ is the task execution time in the best position of the processors (selected imperialist), and $N.T.C._n$ is the total normalized cost of the empire. Now that the total cost has been normalized, the probability of the power of each empire (processor selection) relative to the total empires (processor positions) is calculated by relation 6:

$$Pp_n \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \quad (6)$$

The imperialist rivalry begins with the formation of the early empires. The imperialist countries attract the colonial countries by pursuing a policy of assimilation in various axes of optimization. Imperialist competition, along with the policy of assimilation, forms the core of this algorithm and causes countries to move towards the absolute minimum in order to complete the whole time. In fact, this central government tried to bring the colony closer to itself by pursuing a policy of absorption. As a result, in order to find the best processors for work, the power of optimal solutions is gradually increased and weak solutions are gradually eliminated during imperialist competitions. And any solution that has the shortest completion time in the cost function is the chosen imperialist.

The colony then moves towards its imperialist, with an angle $\theta = \pi/4$. In the division of colonies among empires, the vector P should be formed based on the probability of taking over each empire from the following equation;

$$\mathrm{P} = [\mathrm{pp1}, \mathrm{pp2}, \mathrm{pp3}, \ldots, \mathrm{ppNimp}] \quad (7)$$

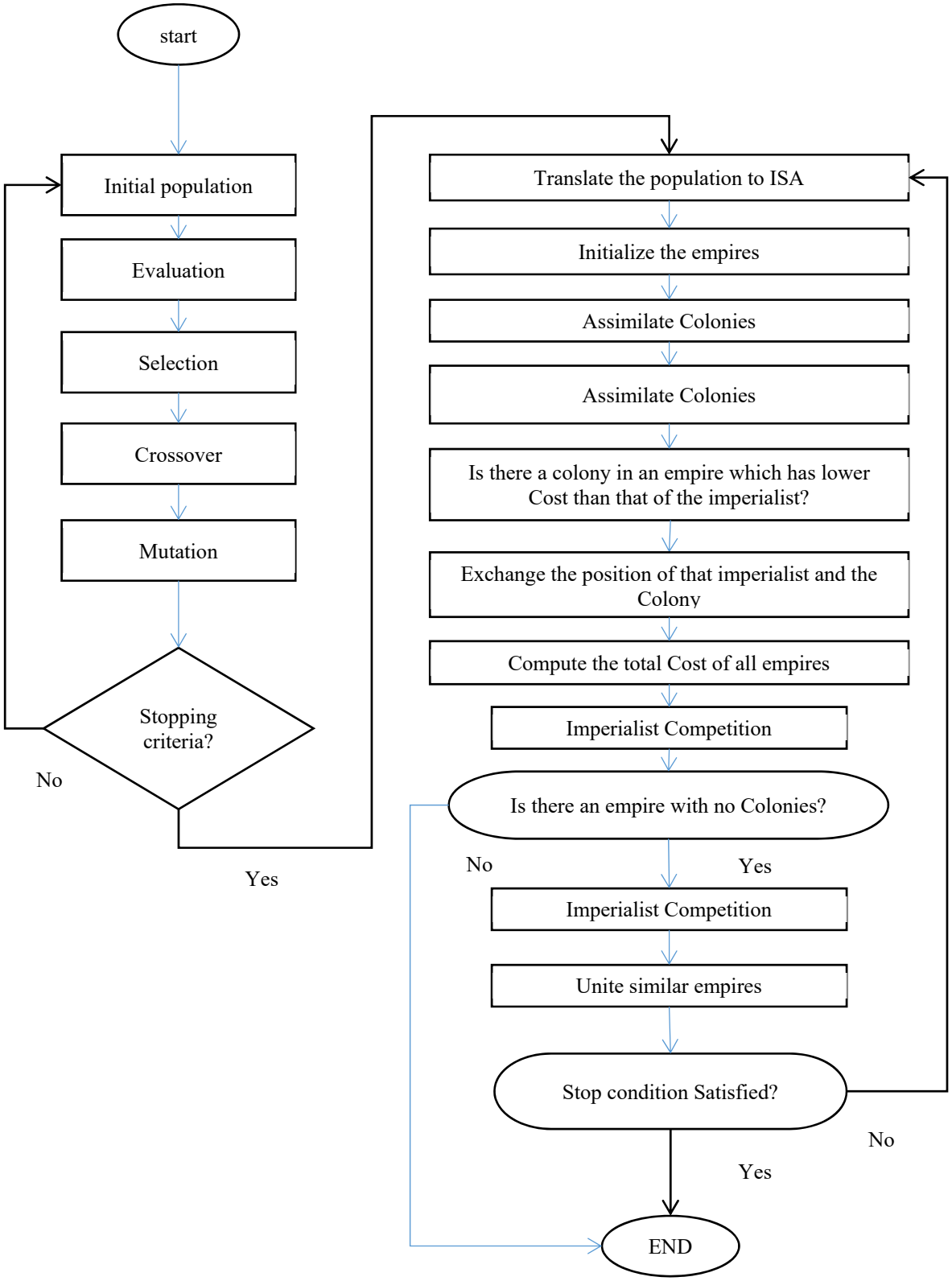
Then the vector R is created which is equal to P and with elements that are random numbers of uniform distribution.

$$\mathrm{R} = [\mathrm{r1}, \mathrm{r2}, \mathrm{r3}, \ldots\ldots, \mathrm{rNimp}]\ \mathrm{r1}, \mathrm{r2}, \mathrm{r3}, \ldots\ldots, \mathrm{rNimp} \sim \mathrm{U}(0,1) \quad (8)$$

Finally, the vector D is created by the difference between these two vectors. Each empire will have the maximum value in indexation with the highest probability of possession.

$$\mathrm{D} = \mathrm{P} - \mathrm{R} = [\mathrm{D1}, \mathrm{D2}, \mathrm{D3}, \ldots, \mathrm{D\ Nimp}] = [\mathrm{pp1} - \mathrm{r1}, \mathrm{pp2} - \mathrm{r2}, \mathrm{pp3} - \mathrm{r3}, \ldots\ldots, \mathrm{ppNimp} - \mathrm{rNimp}] \quad (9)$$

In the competition of empires, the weak imperialists gradually lose their colonies, and after the elimination of all their colonies, after the removal of all its colonies, that empire will disappear, and this process will continue until only the strongest empire remains. Finally, the position and cost of each colony is determined in relation to this single empire.



The proposed GA-ICA hybrid algorithm

Proposed Genetic Hybrid Algorithm - Refrigeration Simulation

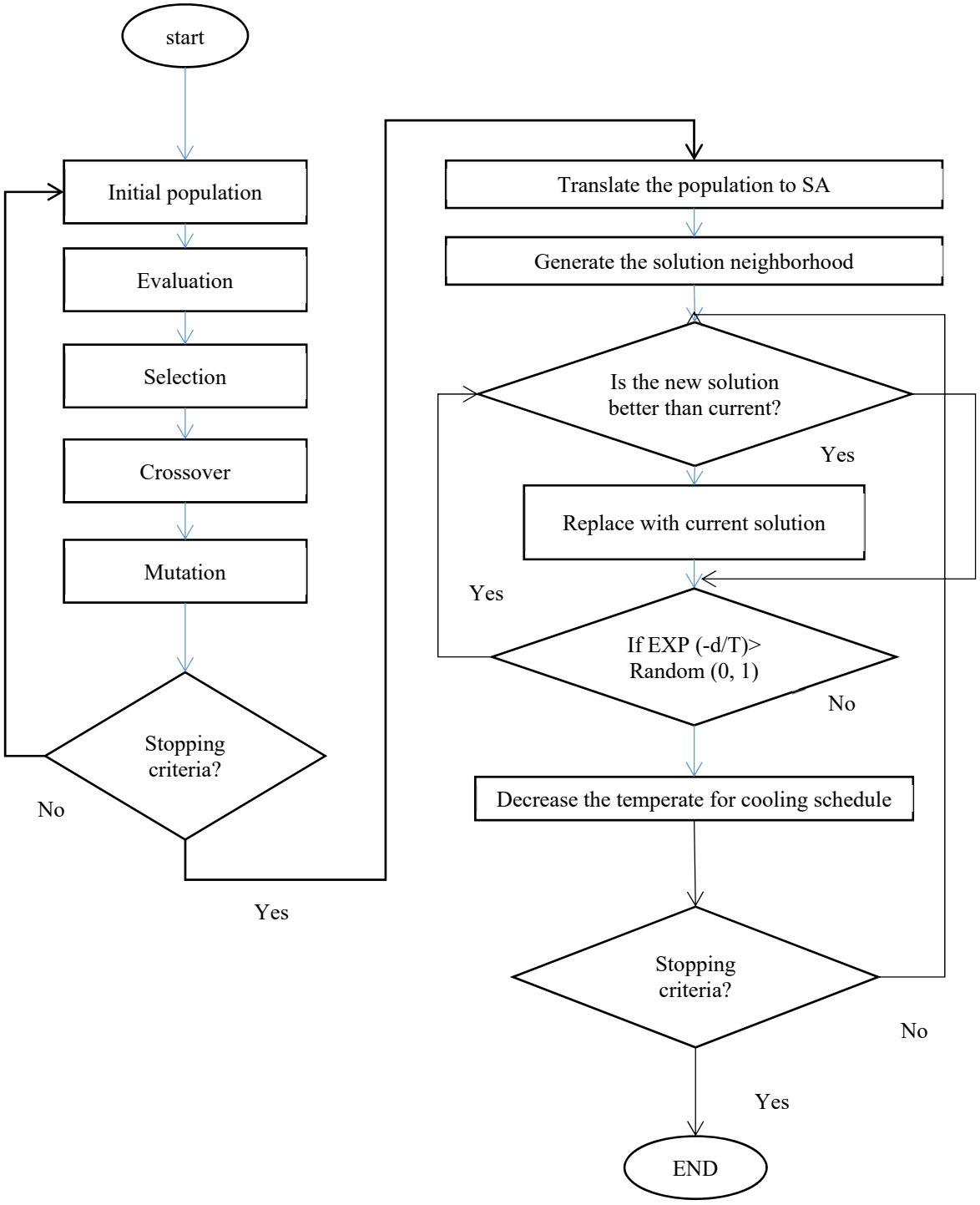
In this matrix algorithm, the initial responses that come from the best processor mapping matrix on workgroups in the genetic algorithm, and form the population of the second algorithm, refrigeration simulation. In fact, the initial population, which is the position of the work on the processors, is like materials that heats up to a temperature higher than their melting point and then gradually lower the temperature. The temperature decreases very slowly, so that the matter is in thermodynamic equilibrium. Refrigeration simulation algorithm is a meta-heuristic search algorithm that has a good ability to escape from local optimal points. In this method, a new neighborhood is generated by using the methods of exchanging works with each other and moving the works of different sources. In this algorithm, to evaluate the objective cost task (execution time), in each iteration, the difference between the task execution time in the new source and the current source is used, which is obtained from Formula 9:

$$d = Complete - time;(new - cpu) - Complete - time_i(current - cpu) \quad (10)$$

The initial answers to the problem are at the default temperature T_0 = 10 and are often inappropriate responses. This inadequacy can be analogized to fragility. The variable that plays the role of temperature then decreases with increasing number of iterations over time and ratio (T / e ^ i) to create better responses as well as the shortest task execution time at low temperatures. Then the temperature of the body remains constant until the best crystal structure formed with the least energy at that temperature.

$$T_{t+1} = \alpha T_t \quad (11)$$

α is the coefficient of temperature change, taken here as 0.95. The default temperature is T_0 = 10.



Proposed hybrid algorithm GA-SA

Proposed Genetic Hybrid Algorithm - Ant Colony Optimization

In this hybrid algorithm, the initial answers matrix that arises from the best processor mapping matrices created on workgroups in the genetic algorithm, the population of the second algorithm is the population of the ant colony optimization algorithm. And this population is then formulated, and the ants build a new solution (such as chromosome formation in a genetic algorithm) by storing information and following the pheromone of each path. In sending tasks to the processors in each path, the amount of pheromone is considered as t_ij, that is read or changed by ants. The amount and concentration of pheromones on a path is a criterion for assessing the desirability of that path and being selected by ants in order to create suitable paths for work to reach the appropriate resources. In each iteration, tasks are assigned to the best selected ants, and each ant from the ant colony creates a new

solution in each iteration. Ant colony optimization, which is based on the simple social behavior of ants, tries to find ways to find the optimal answer.

In this way, each ant starts moving from a random position point towards the optimal source. At any given moment, the amount of pheromone changes is fixed as ΔΓ_ij (t). If the path is used by the Kth ant, it has a direct ratio to the amount of pheromone stored in the path, ie Q = 1, and the inverse ratio with the length of the path of the Kth ant, ie $L^k(t)$, which in the formula (11) is displayed:

$$\Delta\Gamma_{ij}^{K} = \frac{Q^k}{L}(t) if\ arc\ (i,j) is\ used\ by\ ant\ k \quad (12)$$

The tasks entered into the system in each iteration are assigned to the number of the best selected ants (5. = m) that have the shortest task execution time. The initial pheromone value is 5 that is applied to pheromone in a single matrix whose rows are processors and its columns are tasks. The evaporation rate of pheromone is ρ_0 =4. The pheromone value of each path is given in the instant (t + 1) by formula (12), and each ant creates a new path in each iteration, and this process continues until all tasks are scheduled and a general solution is formed. Then the pheromone update is done according to the following formula:

$$\Gamma_{ij}(t+1) = (1-\rho).\Gamma_{ij}(t) + \sum_{k-1}^{m} \Delta\Gamma_{ij}^{k}(t)\ \ for\ each\ (i,j) (13)$$

In a simple simulation system, the ants find their next destination with a definite probabilistic law (p_{ij}^k) in which at each stage, the probable paths are calculated and the best path is selected. Any path to a source with a higher pheromone content is more likely to be chosen as the preferred path than in Formula 13.

$$p_{ij}^{k}(t) = \left[\Gamma_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta} / \sum L \left[\Gamma_{ij}(t)\right]^{\alpha}.\left[\eta_{ij}\right]^{\beta} \quad if\ jL\epsilon\ N_k^i (14)$$

The quantification of exploratory information in the simple ant colony system will be $\eta_{ij} = 1$and the pheromone coefficient (α) and the exploratory coefficient (β) will be α = β = 1, respectively.

Implement and evaluate results

The proposed hybrid algorithm generates an approximate and near-optimal response with a random process of resources. The following tables show the completion time and task execution time of each algorithm for several different executions. These performances were performed in MATLAB software and on a processor (CORE i5 & CORE i5) with 28 cycles per second.

The following tables measure the execution times of 20,000, 30,000, and 40,000 jobs for 20 sources in a source at 28 cycles per second, respectively.

Investigation of hybrid algorithms for 20 processors and 2000 tasks

ALGORITHM	GA-ACO	GA-SA	GA-ICA	GA-PSO	ACO	SA	ICA	PSO	GA
task execution time	119.93	108.93	118.57	104.250	113.50	147.8	149.8	139.78	132.94
algorithm execution time	19.77	4.16	6.27	5.14	17.62	2.48	2.82	2.35	2.51

Investigation of hybrid algorithms for 30 processors and 3000 tasks

ALGORITHM	GA-ACO	GA-SA	GA-ICA	GA-PSO	ACO	SA	ICA	PSO	GA
task execution time	280.20	179.12	185.93	170.75	181.60	235.30	244.93	211.56	201.14
algorithm execution time	28.80	5.37	7.76	7.64	27.06	2.98	3.63	2.87	3.08

Investigation of hybrid algorithms for 40 processors and 40,000 tasks

ALGORITHM	GA-ACO	GA-SA	GA-ICA	GA-PSO	ACO	SA	ICA	PSO	GA
task execution time	134.46	123.40	131.61	115	125.35	180.40	169.40	164.41	142.3
algorithm execution time	42.69	6.7	9.76	10.76	38.48	3.5	4.73	3.37	3.85

According to the obtained information, the proposed GA-PSO algorithm had the best task execution time. And the larger the system, the better the algorithm for implementing the scheduler in massive distributed systems. Below is a diagram of the time process of performing tasks in the grid computing system.

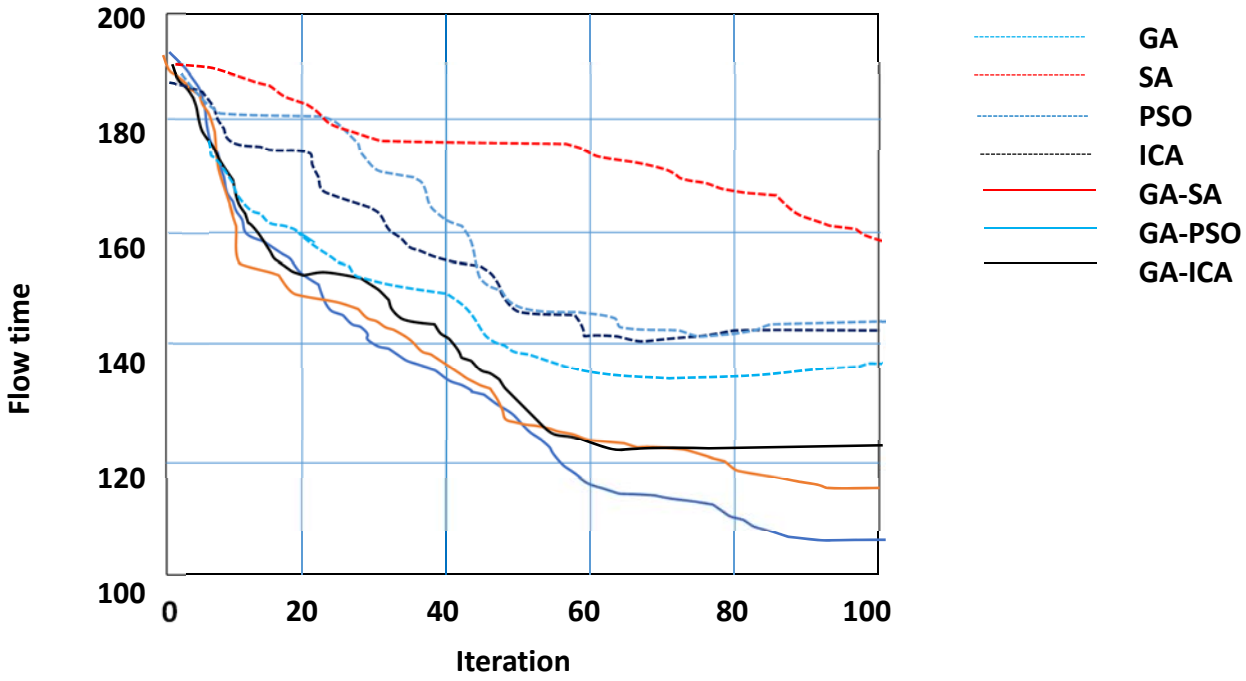


Figure 1 Time flow chart of tasks of first type algorithms in terms of number of iterations for 20 processors and 20 tasks

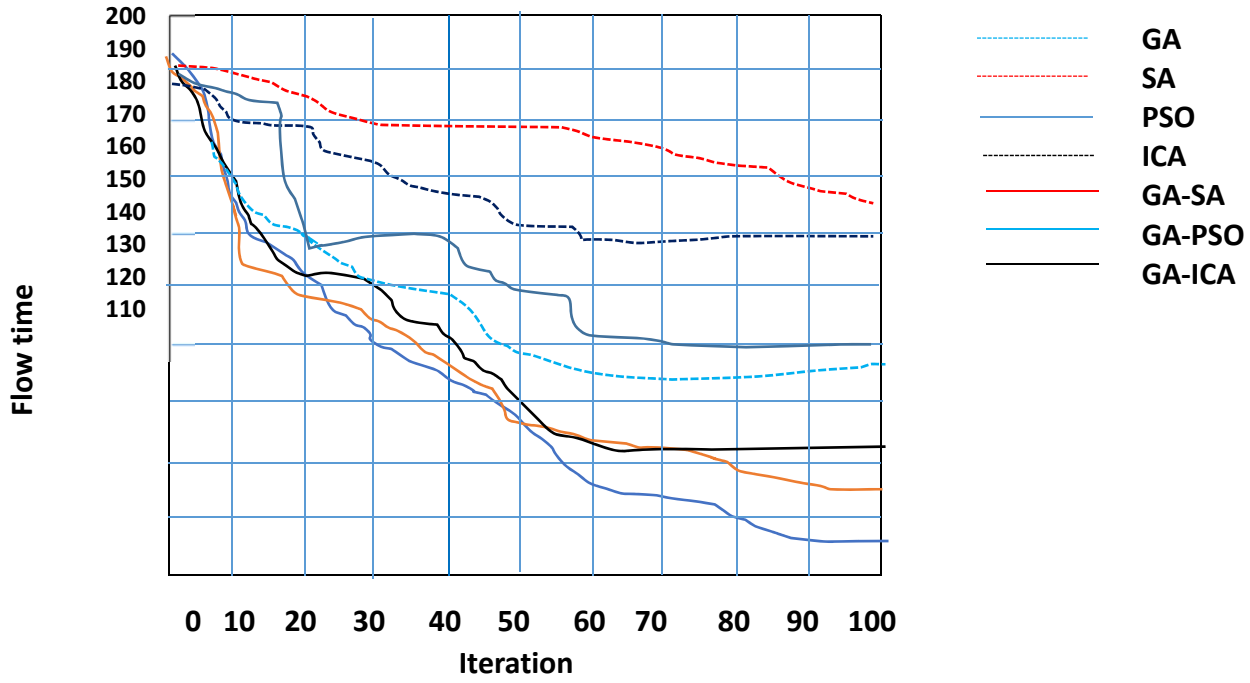


Figure 2 Time flow chart of tasks of first type algorithms in terms of number of iterations for 30 processors and 3000 tasks

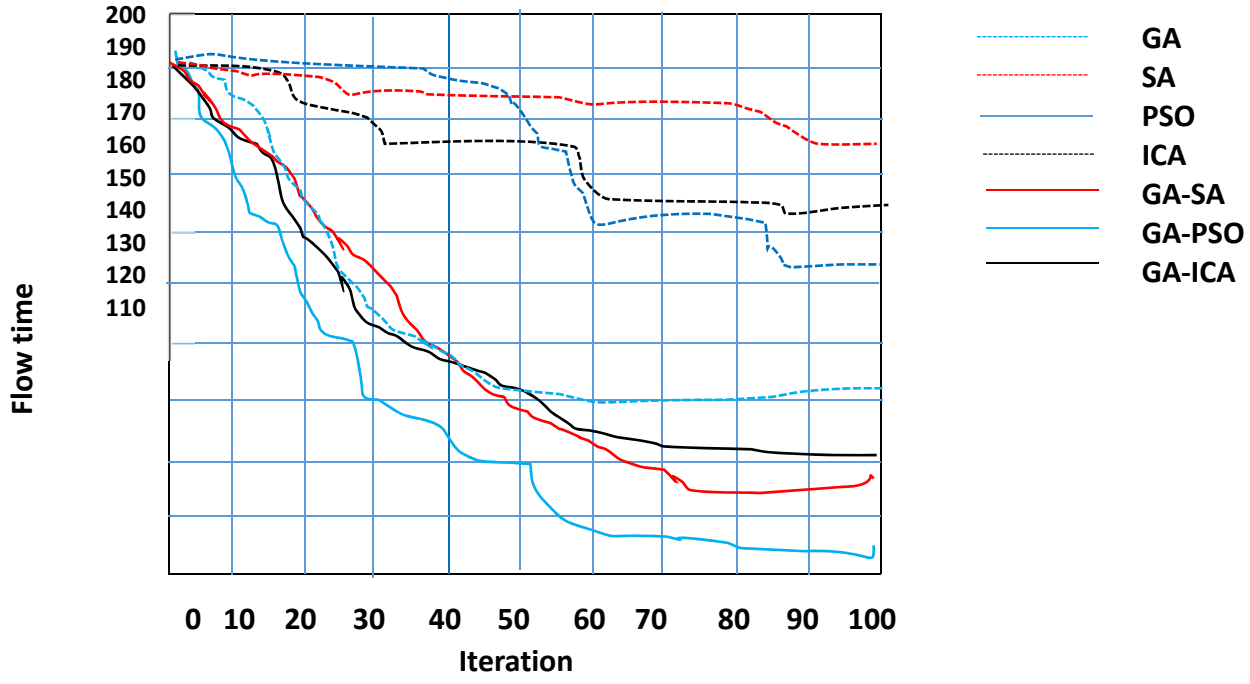


Figure 3. Time flow chart of tasks of first type algorithms in terms of number of iterations for 30 processor and 3000 tasks

Examining time flow chart of tasks of first type algorithms in terms of number of iterations for 30 processors and 3000 tasks and 40 processor and 4000 tasks, it is observed that the simple implementation algorithm of this algorithm has the best time flow chart of tasks based on the number of iterations of the objective function.

Conclusion

Since in the proposed evolutionary algorithms, the distribution of tasks to resources is done randomly and the results are close to the optimal and the time course of the workflow is dynamic in each execution, so the possibility of a slight change in finding the answer to the best algorithm in the system should be considered.

Upcoming work

Evolutionary algorithms are difficult to find approximate solutions to optimization problems. The key aspect that distinguishes these algorithms from conventional algorithms This is why evolutionary algorithms are population-driven, and the original population is completely randomly generated, covering the entire search space. The probable method in these proposed algorithms is evolutionary algorithms and the definitive optimal search method is proposed. Given that the scheduling systems that have been proposed so far for grid computing have not been very successful due to the low accuracy of the proposed algorithms, therefore, we designed and implemented a new scheduling system with hybrid evolutionary algorithms and hybrid evolutionary algorithms using genetic algorithms, in this study. And we tried to create the best task execution time and also the process of executing the workflow in the shortest possible time with these proposed search methods, that finally, among these proposed algorithms, GA-PSO hybrid algorithms showed more effective performance.

This algorithm, in addition to being successful in improving the execution time of tasks in the grid system, has also appeared better in the process of execution of tasks than other evolutionary algorithms. This algorithm has a high convergence speed and it is observed that before the completion of the total number of iterations, this algorithm achieves the optimal global answer, which is not guaranteed by the genetic algorithm alone. In future works, we will develop multiple hybrid and hybrid algorithms for the task scheduling system in grid computing. And in more advanced stages, it has combined these multiple algorithms with neural networks and fuzzy relationships to determine the optimal task scheduling process in massive distributed systems.

References:

[1] Dordaie, N. and N.J. Navimipour, *A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments.* ICT Express, 2018. **4**(4): p. 199-202.
[2] Dai, Y., Y. Lou, and X. Lu. *A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization Algorithm with Multi-QoS Constraints in Cloud Computing*. in *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*. 2015.
[3] Molaiy, S. and M. Effatparvar, *Scheduling in Grid Systems using Ant Colony Algorithm.* International Journal of Computer Network and Information Security, 2014. **6**: p. 16-22.
[4] Zhang, L., et al., *A Task Scheduling Algorithm Based on PSO for Grid Computing.* International Journal of Computational Intelligence Research, 2008. **4**: p. 37-43.
[5] Chen, T., et al., *Task Scheduling in Grid Based on Particle Swarm Optimization*. 2006. 238-245.