Efficient Job Scheduling in Grid Computing with Modified Artificial Fish Swarm Algorithm

Saeed Farzi

***Abstract*—one of the open issues in grid computing is efficient job scheduling. Job scheduling is known to be NP-complete, therefore the use of non-heuristics is the de facto approach in order to cope in practice with its difficulty. In this paper, we propose a modified artificial fish swarm algorithm (MAFSA) for job scheduling. The basic idea of AFSA is to imitate the fish behaviors such as preying, swarming, and following with local search of fish individual for reaching the global optimum. The results show that our method is insensitive to initial values, has a strong robustness and has the faster convergence speed and better estimation precision than the estimation method by Genetic Algorithm (GA) and simulated annealing (SA).**

***Keywords*—AFSA, GA, Grid computing , Job scheduling, SA.**

I. Introduction

Grid computing is a high performance computing environment to solve larger scale computational demands. Grid computing contains resource management, job scheduling, security problems, information management and so on. Job scheduling is a fundamental issue in achieving high performance in grid computing systems. However, it is a big challenge for efficient scheduling algorithm design and implementation. Unlike scheduling problems in conventional distributed systems, this problem is much more complex as new features of Grid systems such as its dynamic nature. And the high degree of heterogeneity of jobs and resources must be tackled. The problem is multi-objective in its general formulation, the two most important objectives being the minimization of makespan and flowtime of the system. Job scheduling is known to be NP-complete [1], therefore the use of non-heuristics is the de facto approach in order to cope in practice with its difficulty. Single heuristic approaches for the problem include Local Search (Ritchie and Levine [2]), Simulated Annealing (Yarkhan and Dongarra [3], Abraham et al. [4]) and Tabu Search (Abraham et al. [4]). GAs for scheduling are addressed in several works (Braun et al. [5], Zomaya and Teh [6], Martino and Mililotti [7], Abraham et al. [4], Page and Naughton [8])[10].

In recent years, with the rise of artificial intelligence and artificial life, the research of swarm intelligence aroused great concern of numerous scholars, and some new-type bionic algorithms with the swarm intelligence are become hot research topics such as the Genetic Algorithm (GA) [11], the Particle Swarm Optimization (PSO) algorithm [12], the Ant Colony Optimization (ACO) algorithm [13], the Bees Algorithm (BA) [14], and Artificial Fish Swarm Algorithm (AFSA) [15]. Their applications in many kinds of scientific research fields show their good properties and practical value. They have some common characters and also have their unique characters. AFSA as a new optimization algorithm becomes a very hot topic, it offers new ideas to solve the optimization problem in signal processing [16], complex function optimization [17], neural network classifiers [18], network combinatorial optimization [19], multi-user detection in communication [20][21], sequence code estimation [22], and some applications [23]. In these applications, the algorithm reflects good performances and becomes a prospective method in solving optimization problems. Its basic idea is to imitate the fish behaviors such as preying, swarming, following with local search of fish individual for reaching the global optimum, it is random and parallel search algorithm, it has the good ability to overcome local extrema, obtain the global extrema and has fast convergence speed. To improve the global convergence of AFSA, we use new behavior- leaping behavior and adaptive step in preying, swarming and following behaviors.

In this paper, we introduce modified AFSA and then use it to build job scheduling method with efficient performance. We compare our method with GA and SA. The comparison result demonstrates that our method has better performance.

The paper is organized as follows. Artificial swarm algorithm is reviewed in Section II. In Section III, The Modified Artificial Fish-Swarm algorithm (MAFSA). In Section IV, Grid Resource Management and Scheduling issues. In Section V, The MAFSA Approach is discussed in details. Some simulation experiments are presented in Section VI. Finally, we draw some conclusions in Section

II. ARTIFICIAL FISH SWARM ALGORITHM

The basic idea of AFSA is to imitate the fish behaviors such as preying, swarming, following with local search of fish individual for reaching the global optimum; it is random and parallel search algorithm. The AFSA is generated from long observation of fish swarm in nature, using the swarm intelligence in the solution of the optimization problem, and combining with the artificial intelligence [29].

A. *SOME DEFINITIONS AND CONCEPRS*

Artificial Fish (AF) is a fictitious entity of true fish, which is used to carry on the analysis and explanation of problem, and can be realized by using animal ecology concept. With the aid of the object-oriented analytical method, we can regard the artificial fish as an entity encapsulated with one’s own data and a series of behaviors, which can accept amazing information of environment by sense organs, and do stimulant reaction by the control of tail and fin. The environment in which the artificial fish lives is mainly the solution space and the states of other artificial fish. Its next behavior depends on its current state and its environmental state (including the quality of the question solutions at present and the states of other companions), and it influences the environment via its own activities and other companions' activities [29].

The AF realizes external perception by its vision shown in Fig.1. X is the current state of an AF, Visual is the visual distance, and Xv is the visual position at some moment. If the state at the visual position is better than the current state, it goes forward a step in this direction, and arrives the Xnext state; otherwise, continues an inspecting tour in the vision. The greater number of inspecting tour the AF does, the more knowledge about overall states of the vision the AF obtains. Certainly, it does not need to travel throughout complex or infinite states, which is helpful to find the global optimum by allowing certain local optimum with some uncertainty [29].

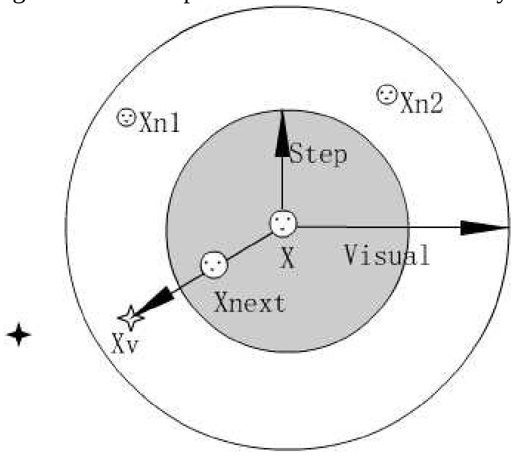

Fig.1 Vision concept of the Artificial Fish

Let X=(x1, x2, …, xn) and Xv=(x1v, x2v, …xnv), then process can be expressed as follows:

$$x_i^v = x_i + Visual.Rand(), i \in (0, n] \quad (1)$$

$$X_{next} = X + \frac{X_v - X}{\| X_v - X \|} Step.Rand(). \quad (2)$$

Where Rand () produces random numbers between 0 and 1, Step is the step length, and xi is the optimizing variable, n is the number of variables.

The AF model includes two parts (variables and functions). The variables include: X is the current position of the AF, Step is the moving step length, Visual represents the visual distance, try_number is the try number and $d_.$ is the crowd factor (0 < $d_.$ < 1). The functions include the behaviors of the AF: AF_Prey, AF_Swarm, AF_Follow, AF_Move.

B. THE BASIC BEHAVIORS OF AFSA

Fish usually stay in the place with a lot of food, so we simulate the behaviors of fish based on this characteristic to find the global optimum, which is the basic idea of the AFSA. The basic behaviors of AF are defined [24, 25] as follows for maximum:

(1). AF_Prey: This is a basic biological behavior that tends to the food; generally the fish perceives the concentration of food in water to determine the movement by vision or sense and then chooses the tendency.

Behavior description: Let Xi be the AF current state and select a state Xj randomly in its visual distance, Y is the food concentration (objective function value), the greater Visual is, the more easily the AF finds the global extreme value and converges.

$$X_j = X_i + Visual.Rand(). \quad (3)$$

If Yi<Yj in the maximum problem, it goes forward a step in this direction;

Otherwise, select a state Xj randomly again and judge whether it satisfies the forward condition. If it cannot satisfy after try_number times, it moves a step randomly. When the try_number is small in AF_Prey, the AF can swim randomly, which makes it flee from the local extreme value field.

$$X_i^{(t+1)} = X_i^{(t)} + Visual.Rand(). \quad (5)$$

(2). AF_Swarm: The fish will assemble in groups naturally in the moving process, which is a kind of living habits in order to guarantee the existence of the colony and avoid dangers. Behavior description: Let Xi be the AF current state, Xc be the center position and nf be the number of its companions in the current neighborhood (dij <Visual), n is total fish number. If Yc>Yi and $\frac{n_f}{n} < d_.$, which means that the companion center has more food (higher fitness function value) and is not very crowded, it goes forward a step to the companion center;

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_c - X_i^{(t)}}{\| X_c - X_i^{(t)} \|} Step.Rand(). \quad (6)$$

Otherwise, executes the preying behavior. The crowd factor limits the scale of swarms, and more AF only cluster at the optimal area, which ensures that AF move to optimum in a wide field.

(3). AF_Follow: In the moving process of the fish swarm, when a single fish or several ones find food, the neighborhood partners will trail and reach the food quickly. Behavior description: Let Xi be the AF current state, and it explores the companion Xj in the neighborhood (dij <Visual), which has the greatest Yj . If Yj>Yiand $\frac{n_f}{n} < d_.$, which means that the companion Xj state has higher food concentration (higher fitness function value) and the surroundings is not very crowded, it goes forward a step to the companion Xj ,

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\| X_j - X_i^{(t)} \|} Step.Rand(). \quad (7)$$

Otherwise, executes the preying behavior.

(4). AF_Move: Fish swim randomly in water; in fact, they are seeking food or companions in larger ranges. Behavior description: Chooses a state at random in the vision, then it moves towards this state, in fact, it is a default behavior of AF_Prey.

$$X_i^{(t+1)} = X_i^{(t)} + Visual.Rand(). \quad (8)$$

III. THE MODIFIED ARTIFICIAL FISH-SWARM ALGORITHM (MAFSA)

In the AFSA, there are many parameters that have impacts on the final optimization result. In order to elevate the global convergence of AFSA, we modified the AFSA from the two aspects as follows.

A. THE LEAPING BEHAVIOR

The preying behavior, swarming behavior and following behavior are all local behaviors in some degree. If the objective functions value is not changed after several iterations, it manifests that the function might fall into local minimum. If the program continues iteration, every AF's result will gradually be same and the probability of leaping out local optimum will be smaller [30]. To increase the probability to leap out local optimum and attain global optimum, we attempt to add leaping behavior to AF. The AF's leaping behavior is defined as follow.

AF_Leap: If the objective function is almost the same or difference of the objective functions is smaller than a proportion during the given (m-n) iterations, Chooses some fish randomly in the whole fish swarm, and set parameters randomly to the selected AF. b is a parameter or a function that can make some fish have other abnormal actions (values), eps is a smaller constant [30].

IF (BESTFC(m)-BESTFC(n))<eps

$$X_{some}^{(t+1)} = X_{some}^{(t)} + b.Visual.Rand(). \qquad (9)$$

B. ADAPTIVE STEP LENGTH IN MAFSA

In [28], M.Jian et al, consider the parameter Step, with the increase of the Step, the speed of convergence is accelerated. However, when the increase of the Step is out of a range, the speed of convergence is decelerated, and sometimes the emergence of vibration can influence the speed of convergence greatly. Using the adaptive step may prevent the emergence of vibration, increase the convergence speed and enhance the optimization precision. In the behaviors of AF_Prey, AF_Swarm and AF_Follow, which use the Step parameter in every iteration, the optimized variables (vector) have the various quantity of Step*Rand ( ) , Step is a fixed parameter, Rand( ) is a uniformly distributed function. M.Jian et al, give an adaptive Step method as follow (t means iteration time).

$$Step_{t+1} = \frac{a.N - t}{N} Step_t \qquad (10)$$

Where $a = (1.1 \sim 1.5)$, N is the all iteration times.

This method has a relation with the iteration time, and gradually decreases the *Step*, so decreases the various quantities of optimized variables in each iteration time. Using the adaptive step, we can select the *Step* more randomly which can guarantee the fast convergence, the result's precision and stability [28].

IV. Grid Resource Management and Scheduling Issues

The grid resource broker is responsible for resource discovery, deciding allocation of a job to a particular resource,

$$X_i^{(t+1)} = X_i^{(t)} + \frac{X_j - X_i^{(t)}}{\| X_j - X_i^{(t)} \|} Step.Rand(). \qquad (4)$$

binding of user applications (files), hardware resources, initiate computations, adapt to the changes in grid resources and present the grid to the user as a single, unified resource. Job scheduling in computational grids is a multi-objective optimization problem. In this work, we are concerned with two-objective cases (makespan, flowtime).

To formulate the problem, we consider J_n independent user jobs n={1, 2, ….N} on R_m heterogeneous resources m={1, 2, …., M} with an objective of minimizing the completion time and utilizing the resources effectively. The speed of each resource is expressed in number of cycles per unit time, and the length of each job in number of cycles. Each job J_n has processing requirement P_j cycles and resource R_m has speed of S_i cycles/second. Any job J_n has to be processed in resource R_m, until completion. To formulate our objective, define C_j as the completion time the last job j finishes processing. Define $C_{max} = max\ \{C_j , j=1, …, N\}$, the makespan and $\sum C_j$, as the flowtime. An optimal schedule will be the one that optimizes the flowtime and makespan [21]. The conceptually obvious rule to minimize $\sum C_j$ is to schedule Shortest Job on the Fastest Resource (*SJFR*). The simplest rule to minimize C_{max} is to schedule the Longest Job on the Fastest Resource (*LJFR*). Minimizing $\sum C_j$ asks the average job finishes quickly, at the expense of the larges job taking a long time, whereas minimizing C_{max}, asks that no job takes too long, at the expense of most jobs taking a long time. In summary, minimization of C_{max} will result in maximization of $\sum C_j$.

V. THE MAFSA APPROACH

When we want to optimize job scheduling with the Modified AFSA, the food concentration (FC)[1] and the structure of AF are two key issues that must be exactly determined.

A. FOOD CONCENTRATION (FC)

Several optimization criteria can be considered for this problem, certainly the problem is multi-objective in its general formulation. The fundamental criterion is that of minimizing the *makespan*, that is, the time when finishes the latest job. A secondary criterion is to minimize the *flowtime*, which is, minimizing the sum of finalization times of all the jobs. It should also be noted that *makespan* and *flowtime* are contradictory objectives; trying to minimize one of them could not suit to the other, especially for planning close to optimal ones. In our method, the optimization criteria are sorted by their importance. The criterion with more priority is *makespan* and the second criterion is *flowtime*. So FC is calculated as:

$$FC = \frac{1}{((h)makespan + (1-h)\overline{flowtime}) + 1} \qquad (11)$$

Where h is priority factor ($h \in [0-1]$) and [2] $\overline{flowtime} = flowtime / M$. M is number of resources.

[2] The *makespan* and *flowtime* values are in incomparable ranges, due to the fact that *flowtime* has a higher magnitude order over *makespan*, and its difference increases as more jobs and machines are considered.

B. THE STRUCTURE OF AF

The most important consideration is the representation strategy, that is, how to encode the solutions of the job scheduling into AFs. In this paper, feasible solutions are encoded in a vector, called *fish*, of size *N (*number of jobs), where *fish*[*i*] indicates the resource where job *i* is assigned by the schedule. Thus, the values of this vector are natural numbers included in the range [1*; M (number of resources)*].

C. DISTANCE BETWEEN TWO FISHES

we define the distance of two AFs as follow.

$$d_{ij} = \sqrt{\sum_{h=0}^{N} (fish_i[h] - fish_j[h])^2} \tag{12}$$

Where *N* is the number of jobs.

VI. Experimental studies

In our experiments, Genetic Algorithm (GA), Simulated Annealing (SA) were used to compare the performance with MAFSA Specific parameter settings of all the considered algorithms are described in Table.2 Each experiment (for each algorithm) was repeated 10 times with different random seeds. The makespan values of the best solutions throughout the optimization run were recorded. And the averages were calculated from the 10 different trials. In a grid environment, the main emphasis was to generate the schedules as fast as possible. So the completion times for 10 trials were used as one of the criteria to improve their performance. First we tested a small scale job scheduling problem involving 3 resources and 13 jobs represented as (3, 13). The resource speeds of the 3 resources are 4, 3, 2 CPU TIME, and the job length of 13 jobs are 6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60 cycles, respectively.

Table.1 shows the results (makespan) for 10 GA runs, 10 SA runs and 10 MAFSA runs. The optimal result is supposed to be 46. While GA provided the best results twice, SA, and MAFSA provided the best result three, five times respectively.

We tested the GA, SA and MAFSA for large scale job scheduling. Table.3 shows the results. We can see that the performance of MAFSA is better than GA and SA under the same condition. Also MAFSA usually

spent the least time to allocate all the jobs on the grid node, GA was the second, and SA had to spend more time to complete the scheduling.

One of the most important things about MAFSA is designing in parallel computing structure. The MAFSA is object-oriented and it can easily implement. Another important thing is that the AFSA has no special requirements to initial values and has the ability of the global convergence, so the initial value can be set with stochastic values or fixed values, and other parameters can be set in a wide range. In brief, the algorithm has strong adaptability.

VII. Conclusion

In this paper, we develop a novel method by introducing the modified AFSA for job scheduling in gird computing. AFSA is novel method to search global optimal value by AF`s prey behavior, swarming behavior and following behavior. The step constrains in the three behaviors affects the global search capacity of the AF. Therefore, we modified the AFSA with adding leaping behavior and using adaptive Step. Then we have used MAFSA to introduce a novel method for job scheduling. We evaluate the performance of our method and compared it with two algorithms, which have been introduced before two (GA, SA), under the same condition. From the simulated experiment the result of MAFSA is better than others.

References

[1] M.R. Garey and D.S. Johnson. "Computers and Intractability – A Guide to the Theory of NPCompleteness." W.H. Freeman and Co., 1979.

[2] G. Ritchie and J. Levine, "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments." Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, 2003.

[3] A. Yarkhan and J. Dongarra." Experiments with scheduling using simulated annealing in a grid Environment, " In 3rd International Workshop on Grid Computing (GRID2002), 2002, pp.232–242.

[4] M A. Abraham, R. Buyya, and B. Nath." Nature's heuristics for scheduling jobs on computational grids, " In The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India, 2000.

[5] H.J. Braun, T. D Siegel, N. Beck, L.L. Blni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, and B. Yao. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, " Journal of Parallel and Distributed Computing, vol.61(6), 2001, pp.810–837.

[6] A.Y. Zomaya and Y.H. Teh. "Observations on using genetic algorithms for dynamic load-balancing, " IEEE Transactions On Parallel and Distributed Systems, vol.12(9), 2001, pp.899–911.

[7] V. Di Martino and M. Mililotti. "Sub optimal scheduling in a grid using genetic algorithms, " Parallel Computing, vol.30, 2004, pp.553–565.

[8] J. Page and J. Naughton. "Framework for task scheduling in heterogeneous distributed computing using genetic algorithms, " Artificial Intelligence Review, vol.24, 2005, pp.415–429.

[9] L. Zhang, Y. Chen, R. sun, S. Jin and B. Yang , " A Task Scheduling Algorithm Based on PSO for Grid Computing, " International Journal of Computational Intelligence Research. Vol.4(1), 2008, pp. 37–43.

[10] J. Carretero, F. Xhafa and A. Abraham, "GENETIC ALGORITHM BASED SCHEDULERS FOR GRID COMPUTING SYSTEMS, " International Journal of Innovative Computing, Information and Control ICIC International Vol.3(6), 2007, pp.1-19, .

[11] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning.Addison-Wesley Longman , 1989.

[12] Eberhart, R., Shi, Y., Kennedy, J.:Swarm Intelligence. Morgan Kaufmann, San Francisco , 2001.

[13] Dorigo, M., Stzle, T.: Ant Colony Optimization. MIT Press, Cambridge , 2004.

[14] Pham, D.T., Ghanbarzadeh, A., Koc, E.: The Bees Algorithm.Technical Note, Manufacturing Egnineering Centre, Cardiff University, UK , 2005.

[15] Li, X.L.: A New Intelligent Optimization-Artificial Fish Swarm Algorithm. Doctor thesis, Zhejiang University of Zhejiang, China , 2003.

[16] Jiang, M.Y., Yuan, D.F.: Wavelet Threshold Optimization with Artificial Fish Swarm Algorithm. Proc. of IEEE International Conference on Neural Networks and Brain, Beijing China, 2005, pp.569 -572.

[17] Xiao, J.M., Zheng , X.M., Wang , X.H.: A Modified Artificial Fish-Swarm Algorithm. Proc. of IEEE the 6th World Congress on Intelligent Control and Automation, Dalian China, 2006, pp.3456-3460.

[18] Zhang, M.F., Cheng, S., Li, F.C.: Evolving Neural Network Classifiers and Feature Subset Using Artificial Fish Swarm. Proc. of IEEE International Conference on Mechatronics and Automation, Luoyang China, 2006, pp.1598-1602.

[19] Shan, X.J., Jiang, M.Y.: The Routing Optimization Based on Improved Artificial Fish Swarm Algorithm. Proc. of IEEE the 6th World Congress on Intelligent Control and Automation, Dalian China, 2006, pp.3658-3662.

[20] Jiang, .Y., Wang, Y., Pfletschinger, S., Lagunas, M.A.: Optimal Multiuser Detection with Artificial Fish Swarm Algorithm. Proc. of International Conference on Intelligent Computing(ICIC 2007). CCIS 2, Springer-Verlag Berlin Heidelberg , pp.1084-1093, 2007.

[21] Yu , Y., Tian , Y.F., Yin, Z.F.: Multiuser Detector Based on Adaptive Artificial Fish School Algorithm. Proc. of IEEE International symposium on communications and information technology, 2005, pp.1480-1484.

[22] Jiang, M.Y., Wang, Y., Rubio, F.: Spread Spectrum Code Estimation by Artificial Fish Swarm Algorithm. Proc. of IEEE International Symposium on Intelligent Signal Processing (WISP'2007). Alcalá de Henares , Spain , 2007.

[23] Jiang, M.Y., Yuan, D.F.: Artificial Fish Swarm Algorithm and Its Applications. Proc. Of International Conference on Sensing, Computing and Automation, Chongqing China, 2006, pp.1782-1787.

[24] X.L.Li, "A New Intelligent Optimization- Artificial Fish Swarm Algorithm, " PhD thesis, Zhejiang University, China, June, 2003,

[25] M.Y.Jiang, D.F.Yuan, "Wavelet Threshold Optimization with Artificial Fish Swarm Algorithm, " in Proc. of the IEEE International Conference on Neural Networks and Brain, (ICNN&B'2005), Beijing, China, 13-15, Oct. 2005, pp.569-572.

[26] A. Abraham, H. Liu, W. zhang and T. chang, "Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, " Springer-Verlag Berlin Heidelberg, 2006, pp. 500–507.

[27] M.Jian, Y.Wang, S.Pfletschinger, "optimal Multiluser Detection with Artificial Fish Swarm Algorithm", ICIC CCIS2, springer, 2007, pp.1084-1093.

[28] M.Jian, N.Mastorakis, D.Yuan, M.A.Languanas, "Multi-thershold Image Segmentation with Improved Artificial Fish Swarm Algorithm Block-Coding and Antenna selection, " European computing Conference(ECC), September 2007.

[29] M.Jian , Y.wang, F.Rubio, D.Yuan, " Spread Spectrum code estimation by artificial fish swam algorithm block-coding and antenna selection, " IEEE international symposium on intelligent signal processing (WISP), October 2007.

[30]] C.Wang, C.Zhou, J.Wma , "an improved artificial fish-swarm algorithm and it `s application in feed-forward neural networks, " proceeding of fourth international conference on machine learning and cybernetics, August 2005, pp.18-21.

Saeed Farzi was born in Kermanshah in 1983. He is an faculty member at the Department of Computer Engineering, Islamic Azad University-beranch of Kermanshah, Iran. He received his B.S. in Computer Engineering from Razi university in Iran, in 2004 and M.S. in Artificial Intelligence from Isfehan university in Iran, in 2006. His current research interests include artificial intelligence, Neural Network, soft computing and Grid computing

Table 1: The results for 10 GA runs, 10 SA runs, and 10 MAFSA runs.

Method	1	2	3	4	5	6	7	8	9	10	Average makespan	Best makespan
GA[26]	47	46	47	47.33	46	47	47	47	47.33	49	47.11	46
SA[26]	46.5	46.5	46	46	46	46.66	47	47	47.33	47	46.6	46
MAFSA	47	46.66	46.4	46	46	46	47	46.4	46	46	46.34	46

Table 2: Design Parameters of GA,SA and LEM3

Algorithm	Parameter name	Value
GA[26]	Size of the population	20
	Probability of crossover	0.8
	Probability of mutation	0.02
	Scale for mutations	0.1
SA[26]	Number operations before temperature adjustment	20
	Number of cycles	10
	Temperature reduction factor	0.85
	Vector for control step of length adjustment	2
	Initial temperature	50
MAFSA	Visual	3
	Step	2.6
	Number of fishes	200
	d (Crowded factor)	0.8
	h (priority factor)	0.9

Table 3: Run time and performance comparison for large dimension problems

	(Resource, Job)						
Method	*(100,10)*	*(200,15)*	*(200,30)*	*(300,15)*	*(300,30)*	*(500,30)*	*(500,50)*
GA	Makespan:123.3	Makespan:151.1	Makespan:198	Makespan:243.3	Makespan:276	Makespan:401.98	Makespan:325
	Time:2986	Time:3210	Time:3615.6	Time:3991	Time:4526.3	Time:9650.1	Time:15142
SA	Makespan:125	Makespan:151.28	Makespan:198	Makespan:2407.1	Makespan:274	Makespan:409.1	Makespan:326
	Time:3568.3	Time:3950.1	Time:4758.6	Time:7125	Time:8123.0	Time:18986	Time:35256
MAFSA	Makespan:120.3	Makespan:150.08	Makespan:193.7	Makespan:240	Makespan:274	Makespan:401.98	Makespan:324.99
	Time:1205.8	Time:2008	Time:2534	Time:3110	Time:3682	Time:7821	Time:9320.2