J. Parallel Distrib. Comput. 74 (2014) 2333–2350

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Task scheduling using NSGA II with fuzzy adaptive operators for computational grids

Reza Salimi [a,*], Homayun Motameni [b], Hesam Omranpour [a]

[a] College of Computer Science, Tabari University of Babol, Iran

[b] Department of Computer Engineering, Islamic Azad University, Sari Branch, Sari, Iran

HIGHLIGHTS

- We enhanced the intelligence of genetic algorithms for adapting to the environment.
- Load balancing was improved indirectly using the fuzzy system without implementing the third objective function.
- Our proposed method creates the Pareto-optimal front faster and with higher quality and diversity.

ARTICLE INFO

Article history:
Received 19 March 2013
Received in revised form
1 January 2014
Accepted 17 January 2014
Available online 30 January 2014

Keywords:
Task scheduling
Load balancing
Grid computing
Non-dominated sorting genetic algorithm II
Variance-based fuzzy operators
Multi-objective optimization

ABSTRACT

Scheduling algorithms have an essential role in computational grids for managing jobs, and assigning them to appropriate resources. An efficient task scheduling algorithm can achieve minimum execution time and maximum resource utilization by providing the load balance between resources in the grid. The superiority of genetic algorithm in the scheduling of tasks has been proven in the literature. In this paper, we improve the famous multi-objective genetic algorithm known as NSGA-II using fuzzy operators to improve quality and performance of task scheduling in the market-based grid environment. Load balancing, Makespan and Price are three important objectives for multi-objective optimization in the task scheduling problem in the grid. Grid users do not attend load balancing in making decision, so it is desirable that all solutions have good load balancing. Thus to decrease computation and ease decision making through the users, we should consider and improve the load balancing problem in the task scheduling indirectly using the fuzzy system without implementing the third objective function. We have used fuzzy operators for this purpose and more quality and variety in Pareto-optimal solutions. Three functions are defined to generate inputs for fuzzy systems. Variance of costs, variance of frequency of involved resources in scheduling and variance of genes values are used to determine probabilities of crossover and mutation intelligently. Variance of frequency of involved resources with cooperation of Makespan objective satisfies load balancing objective indirectly. Variance of genes values and variance of costs are used in the mutation fuzzy system to improve diversity and quality of Pareto optimal front. Our method conducts the algorithm towards best and most appropriate solutions with load balancing in less iteration. The obtained results have proved that our innovative algorithm converges to Pareto-optimal solutions faster and with more quality.

1. Introduction

Grid computing has originated from a new computing environment that has emerged as a main-stream technology for scientific research and cooperation using large-scale computing resources sharing and distributed system integration. In fact, computational resources in grid are geographically distributed computers or clusters, which are aggregated to serve as a single computing resource logically [11,5]. On the other hand, the goal of load balancing algorithms is essential to fairly spread the load on computational resources for maximizing their utilization while minimizing the total task execution time [31,1,14]. In distributed computational systems, load balancing has an important role in reducing response time and avoiding overload. Load balancing is applied in the grid computing system, using some scheduling algorithms to ensure that the entire resource node computing the ratio of its own performance as an equal, therefore by improving

* Corresponding author.
E-mail addresses: rz.sa63@gmail.com (R. Salimi), motameni@iausari.ac.ir (H. Motameni), h.omranpour@tabari.ac.ir (H. Omranpour).

http://dx.doi.org/10.1016/j.jpdc.2014.01.006
0743-7315/

the utilization of resources based on nodes, the overall task completion time can be reduced [2]. Computational grids enabling resource sharing and coordination are now one of the common and acceptable technologies used for solving computational intensive applications rising in scientific and industrial problems. Nevertheless, due to the heterogeneity, dynamicity and autonomy of the grid resources, task scheduling within these systems has become a challenging research area [28]. Therefore, many research works have been done to overcome these challenges by proposing new algorithms and mechanisms. Applying the market model to the grids is a good approach which can easily take the dynamic characteristics of the grid resources into account and simplify the scheduling problem considering user-centric trends. Also performance and quality of scheduling algorithms are very important in grid computing because of variable conditions in resources and communications. The proposed algorithm enhances the intelligence of genetic algorithms in adapting to the environment using the intelligent rate for genetic operators and hence satisfies the better performance and quality. In this paper, we use a multi-objective heuristic genetic algorithm, NSGA II, for optimizing two objectives: Price and Makespan. Besides we considered load balancing and improved it in task scheduling indirectly using the fuzzy system instead of implementing the third objective function. We implemented a variance-based fuzzy crossover operator for this algorithm. Then we improved the Pareto optimal front using the fuzzy adaptive mutation rate. The rest of the paper is organized as follows. We begin with an overview of related works in Section 2. NSGA II and our approach are presented in Section 3. Experimental results and discussion are presented in Section 4. Finally the paper is concluded in Section 5.

2. Related works

Previously proposed approaches for the task scheduling problem in traditional grids are limited for users. These approaches do not provide different solutions with different qualities for users to select a solution based on their requirements and capabilities optionally. For example, approaches [11,5] mostly consider system and grid factors like maximum load balance and Makespan of the system as the main objectives of the task scheduling, ignoring the interests and requirements of users. The approach in [8] considers cost and Makespan as objectives in task scheduling without any consideration about load balancing using the genetic algorithm (GA) and variable neighborhood search. Buyya et al. in Ref. [3] have proposed only an economic model for grid resource management and scheduling, using marketing concepts such as commodity market, posted price modeling, bargaining modeling, contract net modeling, auction modeling and other. In [17], two types of GA have been presented for improving the performance of task scheduling algorithm. These only minimize the total execution time and satisfy the load balance. In [22] a job grouping method using particle swarm optimization (PSO) has been proposed to reduce the communication overhead and consequently reduce the completion time of the processes in the computational grid and improve resource utilization. In [21], a pure load balancing has been represented in the computational grid using the genetic algorithm without any consideration about Makespan or cost for grid resources. In [4], the various load balancing strategies have been studied based on a tree representation of a grid. This study enables transforming any grid architecture into a unique tree with at most four levels. The task scheduling algorithm in [4,20] only considers the load balancing without Makespan or cost for users. A hierarchical layered architecture has been offered for grid computing services in [27], so that an adaptive two levels algorithm has been applied to minimize the overall completion time or Makespan and maximize the system throughput. Using multi-objective optimization algorithm such as NSGA-II is observed in [26]. The NSGA-II is used in [26] to optimize the task scheduling problem in heterogeneous distributed computing systems with two objectives, Makespan and flow time without load balancing and or price. A fuzzy single objective algorithm was proposed in [29] to schedule tasks in multi-clusters and grids. The authors in [29] presented a layered task scheduling using a prioritization mechanism to consider communication requirements of tasks and network load to find a matching degree between available resources and jobs. They considered only the completion time of tasks and ignored the load balancing and the price of schedules. The use of multi-objective optimization was also observed in [19]. The authors used multi-objective particle swarm optimization (MOPSO) to meet user's QoS requirement for the implementation of grid workflow. They found the parallel relation between tasks using their algorithm through both positive layering and reverse layering to reduce only the task execution time. However other important objectives which have raised in grid computing such as price and even load balancing were ignored. The use of the nondominated sorting mechanism was also applied in the particle swarm optimization algorithm in [10] and Nondominated Sorting PSO (NSPSO) was introduced. Then in [25], the use of NSPSO was proposed for scheduling independent tasks on heterogeneous distributed environments. We studied the effect of the mutation rate on the diversity and the quality of Pareto front and proposed fuzzy adaptive mutation rate in [24] for NSGA-II to solve the tasks scheduling problem in the market-based grid computing. In that work three objectives: Price, Makespan and Load balancing were optimized using three-dimensional optimization. Also we applied the NSGA-II with fuzzy crossover operator in [23] to satisfy the load balancing indirectly to reduce the complexity of the algorithm.

3. NSGA-II and the proposed method

The NSGA-II algorithm [6] is the first and one of the commonly used evolutionary multi-objective optimization (EMO) algorithms which search solution space to find Pareto-optimal solutions in a multi-objective optimization problem. NSGA-II uses the elitist principle and an explicit diversity preserving mechanism. In addition, it emphasizes non-dominated solutions and forms the Pareto front as Pareto-optimal solutions [15]. NSGA-II uses an explicit diversity-preservation or niching strategy to assign a diversity rank to all the individuals that are in the same non-dominated front and thus have the same non-dominated rank in the population [23]. The members within each non-dominated front that are in the least crowded region in that front are assigned a higher rank. For calculating the density of solutions surrounding a particular solution in the population, a crowding distance metric is used that is achieved from the average distance of the two solutions on either side of the solution along each of the objectives. This particular niching strategy does not require any external parameters, so it was chosen for NSGA II. Details can be found elsewhere [15,7]. Because of the nature of the models of the multi-objective optimization problems, the non-dominated sorting genetic algorithm (NSGA) can be used to find the non-dominant optimal solutions. In the absence of any additional information about the multi-objective optimization problem, one of these Pareto-optimal solutions cannot be considered as better solution than the others [9]. The superiority of one solution to the others depends on several factors including user's choice and problem environment. Therefore, the NSGA-II determines a set of dominant solution and so Pareto front is obtained [18]. In this paper, the task scheduling problem is optimized with two objectives, Price and Makespan, with NSGA-II without fuzzy logic and with fuzzy logic; besides our method considers load balancing using fuzzy function indirectly. In NSGA-II with variance-based fuzzy crossover and fuzzy mutation, inputs for fuzzy functions are variance between costs of individuals, variance of frequency of resources and variance between values of genes in schedules. In the next subsection we explain the proposed approach with more details.

Fig. 1. A chromosome in the coding scheme.

3.1. Details of the proposed approach

We first use NSGA-II with two objectives, Makespan and Price for optimizing the independent task scheduling problem in market-based grid computing. So the fuzzy mutation operator which was proposed in [24] is applied for generating Pareto optimal solutions with more quality and in less iterations. In [24] two special functions are defined based on problem attributes for fuzzy control of mutation rate. Then load balancing is considered and then performance and quality of solutions are analyzed in three dimensional optimizing. Inputs of the fuzzy system in this stage are variation between individuals' fitness and also variation of individuals' gene that present the difference between the values of genes. These functions are used to increase diversity in the Pareto optimal front and form the Pareto front faster. The output of this fuzzy system determines the mutation rate in the population.

Because of optimizing three objectives, the complexity of the proposed algorithm in [24] is high; therefore we improved the performance of the proposed approach by reducing the number of objectives by satisfying the load balancing indirectly as in [23]. This is performed using the effective variance-based fuzzy crossover operator. In fact, the optimization process is realized with two objectives instead of three objectives. For this purpose we proposed a special function which is used in fuzzy control of the crossover rate for optimizing load balancing indirectly. Inputs of the fuzzy system in this stage are the variance of individuals' fitness along with the variance of frequency of resources in the schedules. The first input is used for increasing diversity in the Pareto optimal front and the second input along with Makespan objective is used to satisfy load balancing indirectly. The output of the fuzzy system determines the crossover rate in the population. Nevertheless improving the quality of Pareto front in [23] required the method which proposed in [24]. Therefore the proposed approach is the improved outcome of [24,23] so that their benefits are integrated to generate Pareto optimal solutions faster and with more quality.

3.2. Encoding mechanism

In the coding scheme of our problem, each solution is encoded as a vector of integers [17]. For a problem with n tasks and m resources, the length of the vector which can be considered as a chromosome is n. In addition, the content of each cell of vector which shows a gene value in chromosome can take a number between 1 and m that represents the resource allocated to that task. An example of a chromosome as a schedule with 10 tasks and 5 resources is shown in Fig. 1.

For creating an initial population with p individuals, a random number between 1 and m is assigned to each cell of the vector of size n for p times.

3.3. Objectives and fitness functions

Our main objective is to provide task assignments that will achieve minimum completion time and minimum Price for users. Our fuzzy NSGA-II algorithm is a two dimensional optimization. In this problem two objectives Price and Makespan are in conflict with each other naturally i.e. when Price is reduced then Makespan is increased and vice versa.

Table 1
Example of tasks and their sizes.

Tasks	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Size	36	18	28	23	31	24	16	29	20	12

Table 2
Example of resources and their speeds.

Processor number	1	2	3	4	5
Processor speed	2	3	2.4	1.6	2.7

3.3.1. Makespan

The first objective function of our approach is to decrease the Makespan of the task schedule. Makespan means the longest completion time among all the resources in the system [11,31]. Consider that T_i and C_j denote the size of the task i and processing speed of the resource j, respectively. Then, the execution time of the task i on the resource j can be formulated as follows:

$$t_{exe}(i, j) = T_i/C_j. \quad (1)$$

For each resource there will be a completion time for tasks which assigned to it. For example, Fig. 2 shows completion time for each resource. For example suppose 10 tasks with the sizes mentioned in Table 1 that are assigned to 5 resources.

Consider five resources (processors) that are used with the following speeds (see Table 2):

For example, the execution time of tasks on resource number 4 using Eq. (1) based on Fig. 1 is:

$t_{exe}(6, 4) = 24/1.6 = 15$

$t_{exe}(9, 4) = 20/1.6 = 12.5$

$t_{exe}(10, 4) = 12/1.6 = 7.5.$

In general, the completion time of tasks assigned to a resource j is calculated as follows:

$$t_{complete}(j) = \left(\sum_{(k \in A_j)} T_k\right) \Big/ C_j \quad 1 \le j \le m \quad (2)$$

where A_j is the set of task indexes which are assigned to resource j. For example, three tasks T_6, T_9 and T_{10} are assigned to resource P_4. Therefore, the completion time of tasks on P_4 will be:

$t_{complete}(4) = 15 + 7.5 + 12.5 = 35.$

Now, Makespan is:

$$Makespan = Max\{t_{complete}(j)\} \quad 1 \le j \le m. \quad (3)$$

Therefore, Makespan in Fig. 2 is 35. One of goals is to minimize Eq. (3), which means that the assigned tasks to resources will be completed in the shortest time.

3.3.2. Minimum price

As mentioned, the resource providers in market-based grids can ask about the cost from the users based on the amount of resources that are requested by them. Therefore, scheduling algorithms in a market-based grid should consider users' willingness to complete their applications in the most economical way possible [8]. So, the second objective function is the total price that must be minimized. Suppose w_j denotes unit price for resource j. Therefore, the execution cost of task i on resource j can be computed using the following equation:

$$Price(j) = t_{complete}(j) \times w_j. \quad (4)$$

Then, the total cost of the scheduling is calculated as follows:

$$Total\ Cost = \sum_{1 \le j \le m} Price(j) \quad (5)$$

where total cost denotes the overall cost resulting from a chromosome in population that represents a schedule.

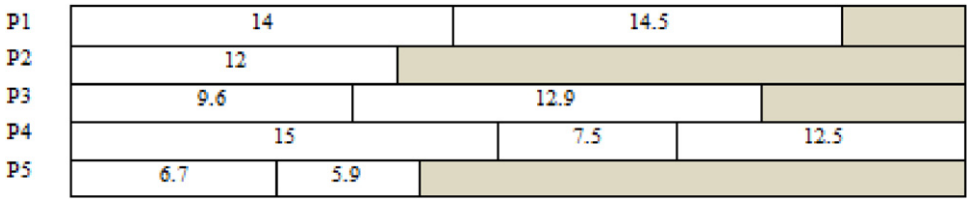

Fig. 2. Completion times of tasks on resources.

3.3.3. Maximum load balance

In the distributed systems, load balancing is a technique (usually performed by load balancers) to spread workload among resources, in order to achieve optimal resource utilization, throughput, and response time [31]. The load balancing mechanism attempts to distribute the load on each resource equitably, and maximize the resource utilization and minimize the total task execution time. In order to satisfy these goals, the load balancing mechanism should be 'fair' in distributing the load across the resources; it implies that the difference between the "heaviest-loaded" resource and the "lightest-loaded" resource should be minimized. So the other important objective of our approach is to obtain the maximum load balance. We first specify the average resource utilization.

The average resource utilization is obtained by dividing the sum of all the resources' utilization to the total number of resources [11]. So, we calculate the expected utilization of each resource based on the given tasks assignment. We calculate it by dividing the task completion time of each resource by the Makespan. Thus, the utilization of each resource and average resource utilization are:

$$P_u(j) = t_{complete}(j)/Makespan \quad 1 \le j \le m \tag{6}$$

$$P_e = \left(\sum_{1\le j\le m} P_u(j)\right) \Big/ m \tag{7}$$

where $P_u(j)$ is the utilization of each resource, $t_{complete}(j)$ is the completion time of tasks assigned to a resource j, P_e is the average resource utilization and m is the number of available resources.

We must note that a high value of the average resource utilization does not always mean a desirable load balance across all the resources in the system [11,20]. So, by minimizing the mean square deviation of $P_u(j)$ means the improvement of the load balance across all resources. The mean square deviation of $P_u(j)$ is achieved as follows:

$$P\_msd = [(\Sigma_j(P_u(j) - P_e)^2)/m]^{0.5} \quad 1 \le j \le m. \tag{8}$$

In fact, the mean square deviation shows how well the loads are balanced across all the resources. The lower the value of this performance criterion, the better the load balancing [11]. Hence, the third objective function for optimizing the scheduling is to improve the load balance across all resources that is satisfied by minimizing Eq. (8).

3.4. Fuzzy adaptive genetic operators

Various mutation rates have been tested and it was found that in general using high mutation rates optimizes objective functions with greater result values, whereas using low mutation rates these objective functions will be less optimized. For example, the price objective function had great result values so using high mutation rates the Pareto optimal fronts were obtained in which the price objective had been minimized more in comparison with using low mutation rates. However, these solutions had great Makespan objective values. Nevertheless using low mutation rates in the algorithm caused the Makespan objective function to be minimized more (optimized more) in comparison with using the higher mutation rate, but the obtained solutions had greater price values. This was shown in experimental results. The reason is the unique exploitation property of mutation operator. Therefore, the higher rate for mutation operator resulted in more exploitation in the range of solutions. But, when the sizes of values in various objectives are different, the objectives with greater size of values explored more. In contrast the lower mutation rate reduced the probability of exploration. Therefore its impact is not noticeable in the great range of values but tangible on the small range of values.

Since various objectives have different sizes of values, so by adjusting the probability of mutation, more optimization of a certain objective could be achieved. But, this could be difficult while the number of objectives is more than two. Moreover, it is necessary to optimize all objectives as much as possible so that the best possible solutions for any objective could satisfy other objectives. Thus, an adaptive method was proposed to apply a higher mutation rate for objectives with a greater range of values and a lower mutation rate for objectives with a smaller range of values. In addition, the mutation rate must be controlled to prevent reducing the quality of some solutions. As a result, the controlling mutation rate also causes fast convergence to Pareto optimal solutions. Therefore, the functions which have been defined have significant effects on both convergence of algorithm and generating qualified and diverse solutions.

Then the effect of the crossover rate for controlling Pareto optimal solutions was analyzed. It was found that the crossover rate did not have the same effect as the mutation rate did. However, an adaptive crossover rate based on the variance of frequency of resources with the help of Makespan optimality could impress load balancing efficiently. In the next subsection, we define fuzzy adaptive mutation and crossover rates and special functions for calculating inputs of their corresponding fuzzy systems.

3.4.1. Fuzzy mutation operator

In this paper, the specialized fuzzy mutation operator is developed and compared with the general mutation operator. Two functions are designed to provide the inputs of the fuzzy system. The first function was proposed in [24]; however we improved it and proposed a new function called binary variance. This new function calculates the variance of genes values in different chromosomes in order to increase the variety in the population. This function takes individuals existing in Pareto front as input, selects the best member, and then compares the genes of other individuals with the genes of the best member. This function will generate a number in the interval [0, 1]. The inputs of the second function are also the Pareto front members. It calculates the variance of average of three objectives functions for different members. This function was proposed in [24] and used to ensure diversity among the members according to their fitness. This function will generate a number in the interval [0, 1]. A fuzzy system is designed which consists of two inputs and one output. Then the values of the mentioned two functions will be the inputs of the fuzzy system. The output of the fuzzy system is used to determine the probability of mutation in the population and also the probability of mutation of genes in chromosomes which determines the number of genes for mutation in any chromosome. Special functions for calculating the inputs of mutation fuzzy system are explained in the next subsection.

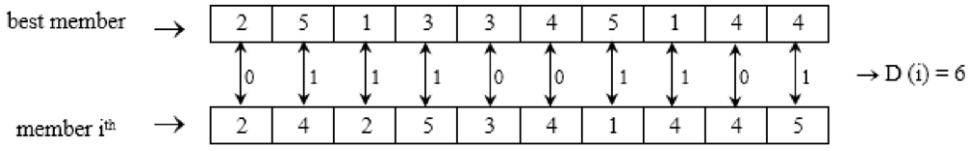

Fig. 3. An example of calculating binary variance of genes values.

3.4.1.1. Binary variance function of genes values. This function that we named it as binary variance takes individuals existing in Pareto front as input, selects the best member, and then compares genes of other individuals with genes of the best member. This function will generate a number in the interval [0, 1]. First it must select the best member in terms of average of fitness. Since various objectives have very different sizes of values, so we normalize them using Eqs. (9)–(11).

$$NormalMakespan(k) = (Makespan(k) - makespan^{min})/(makespan^{max} - makespan^{min}) \quad k \in PF \tag{9}$$

$$NormalTotalCost(k) = (TotalCost(k) - TotalCost^{min})/(TotalCost^{max} - TotalCost^{min}) \quad k \in PF \tag{10}$$

$$NormalP\_msd(k) = (P\_msd(k) - P\_msd^{min})/(P\_msd^{max} - P\_msd^{min}) \quad k \in PF \tag{11}$$

where $makespan^{min}$ and $makespan^{max}$ are minimum and maximum values of Makespan objective of the existing members in the Pareto front respectively. $TotalCost^{min}$ and $TotalCost^{max}$ are also minimum and maximum values of total price of the existing members in the Pareto front respectively. Finally P_msd^{min} and P_msd^{max} are minimum and maximum values of mean square deviation of the existing members in the Pareto front respectively. These values are obtained in any iteration and are used for mapping Makespan, Price and Load balance objectives' values to the interval [0, 1]; then we calculate the average of them using Eq. (12). The member with best average of objectives values (here with minimum average of objectives values) is selected using Eq. (13) and used for calculating variance of other samples.

$$AvgObjectives(k) = (NormalMakespan(k) + NormalTotalCost(k) + NormalP\_msd(k))/3 \quad \text{for all } k \in PF \tag{12}$$

$$BestMember = min\{AvgObjectives(PF)\} \quad PF \text{ is aset of members of Pareto front}. \tag{13}$$

Now it should calculate the number of genes whose values are different from the value of corresponding gene belonging to the best of every member of Pareto front. For this purpose, every gene of every member is compared to the corresponding gene belonging to the best; if they are unequal, one unit is added to a counter. When all of genes of chromosome i were checked, the value of the counter that is saved into the $D(i)$ shows a number of genes of chromosome i which are unequal to the corresponding genes belonging to the best. This process is repeated for all members of Pareto front. Fig. 3 denotes an example for this function.

When array D was completed for all of members, binary variance of genes values for member k is calculated using Eq. (14).

$$GeneBinVariance(k) = D(k)/n \tag{14}$$

where n is the number of genes and P is the number of members belonging to current Pareto front. $D(k)$ shows the number of genes belonging to member k which are different from corresponding genes in the best member. The value of this variance is in the interval [0, 1]. Then using Eq. (15), the average of these variances is calculated as the input of the fuzzy system.

$$MutFISinput1 = \Sigma_{k \in PF} GeneBinVariance(k)/P - 1. \tag{15}$$

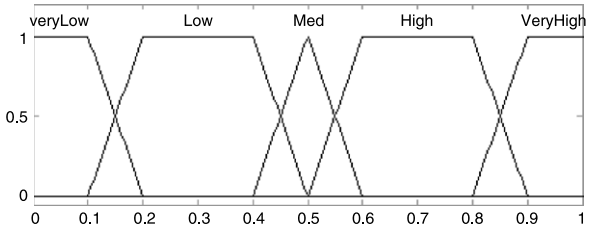

Fig. 4. Input membership function for variance of genes values (A1).

A high value from this variance signifies high diversity between genes of members belonging to Pareto front, and a low value from this variance signifies low diversity between genes and subsequently between members. So the effect of this factor on the mutation rate is a reverse effect. When the value of binary variance is low (i.e. low diversity), the mutation rate must be increased to increase diversity between members and vice versa.

3.4.1.2. Variance function of average of objectives values. This function also takes individuals existing in Pareto front as input and generates the variance of average of objectives values as output. For this purpose, first the objectives values are normalized using formulas (9)–(11). Then we calculate average of them using formula (12). There is now a vector that every of its element is the average of objectives values corresponding to a member belonging to Pareto front. Then for calculating the variance of them, the average of them is calculated as μ. So variance of average of objectives values is obtained from Eqs. (16) and (17).

$$\mu = \Sigma_{k \in PF} AvgObjectives(k)/P \tag{16}$$

$$\sigma^2 = \Sigma_{k \in PF} (AvgObjectives(k) - \mu)/P. \tag{17}$$

The obtained variance is typical of variation measure between averages of objectives values from various members of Pareto front and so this variance has a reverse effect on the mutation rate, because when this variance is low, diversity in objectives values is low and the mutation rate must be increased and vice versa. The output of this function should be used in fuzzy decision making in the fuzzy system; therefore it is normalized using formula (18).

$$MutFISinput2 = (\sigma^2 - \sigma^2_{min})/(\sigma^2_{max} - \sigma^2_{min}). \tag{18}$$

3.4.1.3. Designing fuzzy system for determining the fuzzy mutation rate. After the inputs of the fuzzy system were obtained, two membership functions must be designed. Figs. 4 and 5 show these membership functions for these two inputs. Moreover, the output membership function has been depicted in Fig. 6. Also the set of fuzzy rules for the fuzzy system is shown in Table 3.

3.4.2. Fuzzy crossover operator

The proposed fuzzy crossover operator satisfies the load balance objective indirectly by using the fuzzy system and Makespan objective [23]. In fact, we optimize and satisfy three objectives through two objectives optimization. The fuzzy system corresponding to this operator uses two effective inputs for determining probability of crossover in population in order to increase the diversity and satisfy the load balance. In this paper, one point

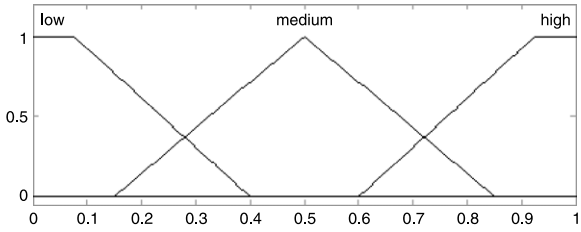

Fig. 5. Input membership function for variance of average of objectives values (B).

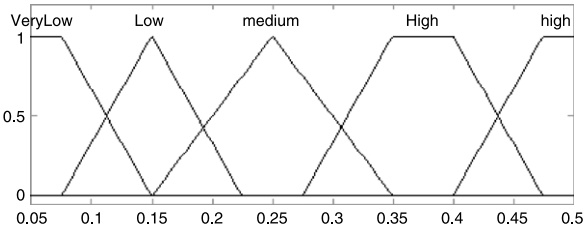

Fig. 6. Output membership function for the fuzzy mutation rate (PM).

Table 3
Fuzzy rule database for fuzzy mutation.

If	A1 is very low	and	B is low	then	PM is very high
If	A1 is very low	and	B is medium	then	PM is high
If	A1 is very low	and	B is high	then	PM is medium
If	A1 is low	and	B is low	then	PM is high
If	A1 is low	and	B is medium	then	PM is medium
If	A1 is low	and	B is high	then	PM is medium
If	A1 is medium	and	B is low	then	PM is high
If	A1 is medium	and	B is medium	then	PM is medium
If	A1 is medium	and	B is high	then	PM is low
If	A1 is high	and	B is low	then	PM is medium
If	A1 is high	and	B is medium	then	PM is low
If	A1 is high	and	B is high	then	PM is very low
If	A1 is very high	and	B is low	then	PM is medium
If	A1 is very high	and	B is medium	then	PM is medium
If	A1 is very high	and	B is high	then	PM is very low

crossover is used. Two functions are applied for calculating the variances as inputs for the fuzzy system. The first function is the variance of average of objectives values as mentioned in the previous section. The variance of average of objectives values for the fuzzy system causes the algorithm to be controlled toward generating more diverse solutions. In this method when diversity of solution is low, the crossover rate is increased to increase the explorative power, so Pareto front can be created faster. The second function calculates the variance of frequency of involved resources in schedules. This function was proposed in [23]; however we improved it and proposed the new version of it in this paper.

3.4.2.1. Variance function of frequency of involved resources. Inputs of this function in each generation are the members belonging to Pareto front. Each resource is used in a schedule for different times. This function is used for distributing the tasks to the resources equally. In fact this function is proposed and designed in order to cooperate with Makespan for balancing the loads. This function calculates the variance of frequency of involved resources in scheduling. For example in a scheduling ten tasks are assigned to resource 1 and five tasks to resource 2, so the frequency of resources 1 and 2 in this scheduling is ten and five, respectively. For defining this function, first the value of expected average of frequency of each resource in schedule must be calculated by dividing the total number of tasks on the total number of resources. Suppose this value is "α". Then the frequency of each resource is

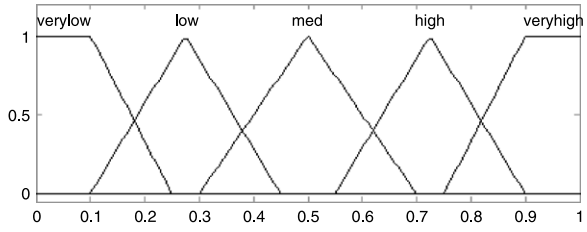

Fig. 7. Input membership function for variance of frequency of involved resources (A2).

calculated in the current schedule. Using this data, integer variance of frequency of involved resources is calculated. Since the collected data from the Pareto front are integer, we define the special variance as integer variance to meet our needs. In this way, the sum of the absolute value of the difference between the frequencies and average value is divided by the sum of these frequencies or number of total tasks. Suppose L_i is the frequency of the resource i in the current schedule and the total number of the resources is m. This variance is calculated as follows:

$$F(k) = (\Sigma_{i=1}^{m}|L_i - \alpha|)/n \quad \text{for all } k \in PF \tag{19}$$

$$XoverFISinput1 = \Sigma_{k\in PF}F(k)/P. \tag{20}$$

The value of this variance is used as the first input of the fuzzy system in each generation. The second input is also the variance of average of objectives values.

$$XoverFISinput2 = MutFISinput2. \tag{21}$$

The value of the integer variance of frequency of involved resources will be a positive number between zero and one, so that a low value near zero indicates nearly equal distribution of tasks to resources, and a high value near one means highly unequal distribution of tasks to resources. The value of this function has the direct effect on the crossover rate in the fuzzy system, so that when this value is high, some resources are idle and some are overloaded; therefore the crossover rate should also be increased to search the solution space more for discovering solutions better. And when this value is low, the number of assigned tasks to each resource is nearly equal and subsequently the crossover rate should also be low to maintain this state for keeping load balance with cooperation of Makespan objective.

The fuzzy crossover operator based on the integer variance of frequency of resources controls the algorithm to keep the schedules which assign the tasks to the resources equally. With this approach, the probability of having chromosomes in which any gene is equally repeated throughout the chromosome is increased. After several iterations, the algorithm increases the schedules in the population where the number of tasks assigned to each resource is nearly equal. Moreover the algorithm should minimize the Makespan objective. Therefore to minimize the Makespan, obviously the tasks should be displaced between resources. In such a situation, the Makespan becomes minimum only through assigning larger tasks to the resources with higher computational capacity and smaller tasks to the resources with less computational capacity which results in load balance indirectly. In fact the minimum Makespan and the equal distribution of tasks results in assigning tasks to the resources fairly. In general according to [13], this means the load balancing. This issue is illustrated in simulation results.

3.4.2.2. Designing the fuzzy system for determining the crossover fuzzy rate. After the inputs of the fuzzy system were obtained, two membership functions should be designed. Figs. 7 and 5 show these membership functions for these two inputs. Moreover, the output

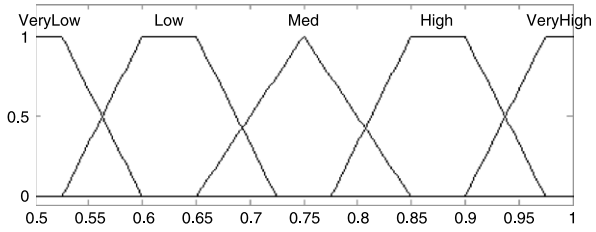

Fig. 8. Output membership function for the fuzzy crossover rate (pXover).

Table 4
Fuzzy rule database for fuzzy crossover.

If	A2 is very low	and	B is low	then	pXover is medium
If	A2 is very low	and	B is medium	then	pXover is low
If	A2 is very low	and	B is high	then	pXover is very low
If	A2 is low	and	B is low	then	pXover is medium
If	A2 is low	and	B is medium	then	pXover is low
If	A2 is low	and	B is high	then	pXover is very low
If	A2 is medium	and	B is low	then	pXover is high
If	A2 is medium	and	B is medium	then	pXover is medium
If	A2 is medium	and	B is high	then	pXover is low
If	A2 is high	and	B is low	then	pXover is very high
If	A2 is high	and	B is medium	then	pXover is high
If	A2 is high	and	B is high	then	pXover is medium
If	A2 is very high	and	B is low	then	pXover is very high
If	A2 is very high	and	B is medium	then	pXover is high
If	A2 is very high	and	B is high	then	pXover is medium

membership function has depicted in Fig. 8. Also the set of fuzzy rules for the fuzzy system is shown in Table 4.

4. Experiments

In this section, the performance of proposed approaches which have been discussed in the previous section is simulated and evaluated. These approaches attempt to improve the performance of NSGA-II and quality and diversity of solutions. The performance criteria such as Load balancing level and Makespan are used to evaluate the performance of the scheduling algorithms in grid computing [13], so we compare our approach with other approaches using these criteria along with the price of schedules. Simulations are performed in five stages. The first stage is two objectives optimization without load balancing using the fuzzy mutation rate for improving algorithm performance and diversity and quality of solutions. This stage of experiment shows that the Makespan optimization is independent of the load balancing and has no effect on the load balancing. The second stage is optimization with three objectives: Makespan, Price and Load balancing. This optimization is not suitable because only two objectives Makespan and Price are important for the user. So making decision is difficult and sophisticated with three objectives. Therefore we use the fuzzy crossover rate to optimize load balancing indirectly. So we optimize task scheduling with two objectives, Makespan and Price directly and Load balancing indirectly using fuzzy crossover and Makespan optimization in the third stage. In this stage, the quality of the Pareto front is not suitable in terms of Makespan, so in the fourth stage we use the fuzzy crossover and mutation rate for aggregation all observed benefits in previous stages. And in the final stage of experiment, the performance and quality of our approach are compared with another multi-objective heuristic algorithm. For this purpose our approach is compared with the nondominated particle swarm optimization (NSPSO) algorithm.

The simulated model for grid includes users' applications and several independent resources. Applications are fragmented to many independent tasks and submitted to scheduler for dispatching to resources to execute. Moreover existing resources in grid

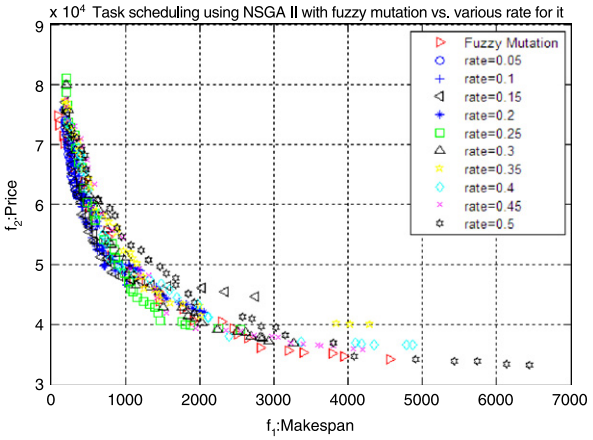

Fig. 9. Obtained Pareto-optimal fronts, population size: 100.

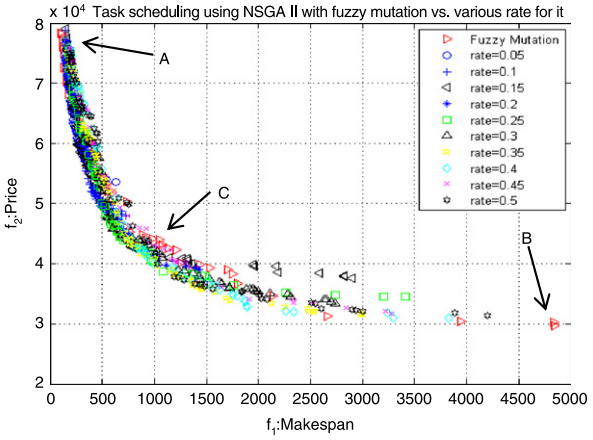

Fig. 10. Obtained Pareto-optimal fronts, population size: 200.

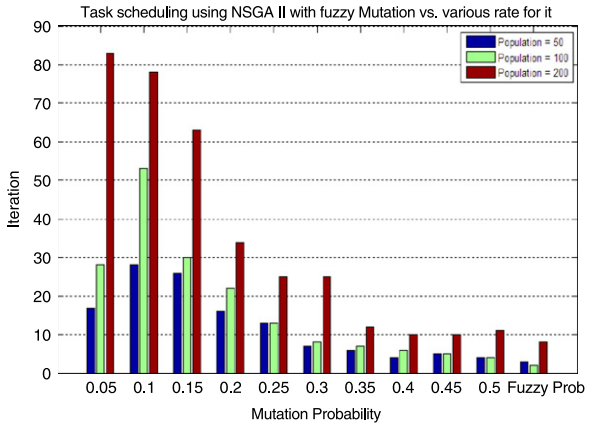

Fig. 11. Number of iterations for creating the complete Pareto front in various sizes of population.

have running incomplete tasks that are considered as dedicated tasks on each resource. The number of tasks and of resources are not predictable in the real environment and are considered 500 and 50 respectively in all experiments [11]. Each task has a size which is generated randomly from the interval [20, 100]. Each resource has a Price and a processing speed or computation capacity in time unit that are chosen randomly in the interval [1, 5] and [2, 10] respectively. Naturally, the resources with higher processing speeds will have higher prices.

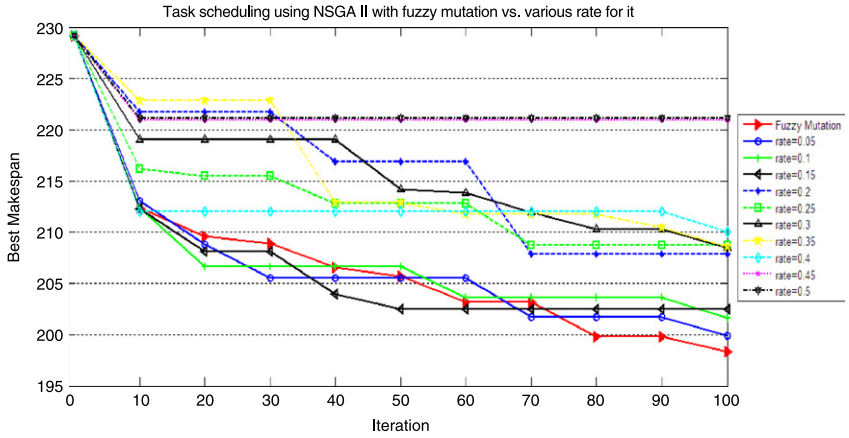

Fig. 12. Diagram of the best values of makespan in the mutation rate experiment.

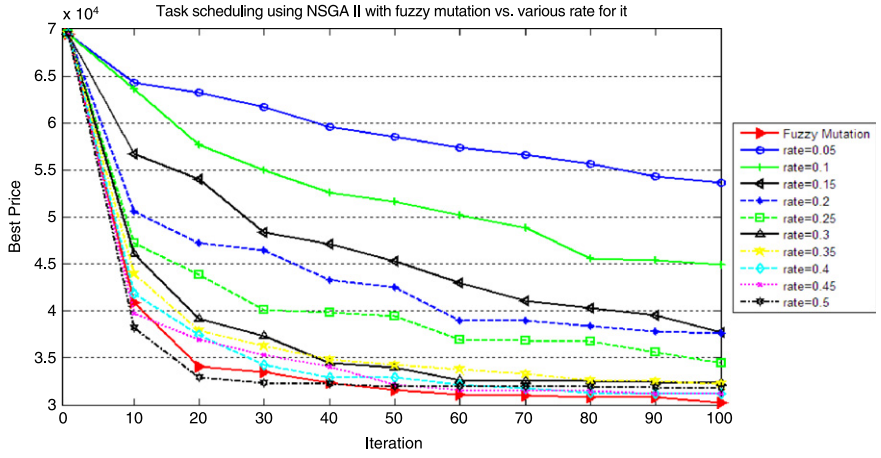

Fig. 13. Diagram of the best values of price in the mutation rate experiment.

In this section, NSGA-II is simulated with variable rates for genetic operators that are determined through the fuzzy system. The individuals of initial population are created randomly (see Section 3.2). In order to eliminate the effect of the randomly generated values to the obtained results, each experiment is repeated 10 times and the average value of the results is reported.

4.1. Simulation of NSGA-II with fuzzy mutation operator for optimizing Makespan and Price

Since, in the market-based grid environment, two factors Price and Makespan are very important, we first optimize the task scheduling problem in terms of these two factors. Various fixed rates for mutation are compared to the adaptive fuzzy rate for it. For simulating the algorithm with adaptive fuzzy mutation rate, two functions binary variance of genes values (Eq. (15)) and variance of average of objectives values (Eq. (18)) are used to generate inputs of the fuzzy system. The output of this system is the mutation rate. In this stage, standard one point crossover is used in the algorithm. Tables 5 and 6 are used as the specific parameters. In Table 6 mutation probability is the likelihood of mutating a particular solution and Bit mutation probability is the likelihood of mutating each bit of a solution in mutation.

Obtained Pareto optimal fronts with population's different sizes are shown in Figs. 9 and 10. The quality of the Pareto optimal solutions obtained from NSGA-II with various rates for mutation can be seen in these figures. The quality of the Pareto optimal solutions obtained from the fuzzy method is better than all other rates.

Table 5
Problem parameters.

Parameter	Value
Population size	100, 200
Number of generations	100
Number of tasks	500
Size of tasks	20–100
Number of resources	50
Price for resources	1–5
Processing speed of resources	2–10

Table 6
Genetic operators.

Algorithm	Parameters		
	Crossover probability	Mutation probability	Bit mutation probability
NSGA II	0.9	0.05–0.5	0.05–0.5
Fuzzy NSGA II	0.9	PM (Fig. 6)	PM (Fig. 6)

In the analysis for instance, the points A and B in Fig. 10 are the best solutions in terms of only one objective. For example point A is the best member of Pareto-optimal front in terms of Makespan while the point B is the best solution in terms of Price. But the point C solutions have compelling quality in terms of both objectives Makespan and Price. In other words, these points have moderate

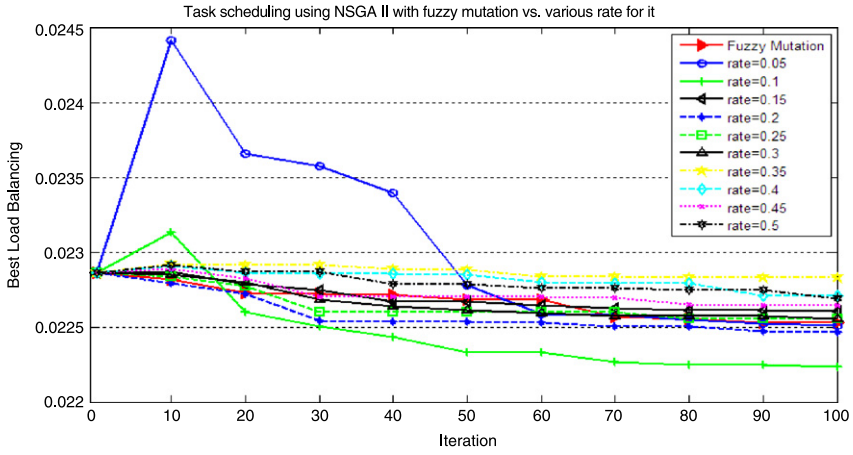

Fig. 14. Diagram of best members in terms of the mean square deviation of resources utilization.

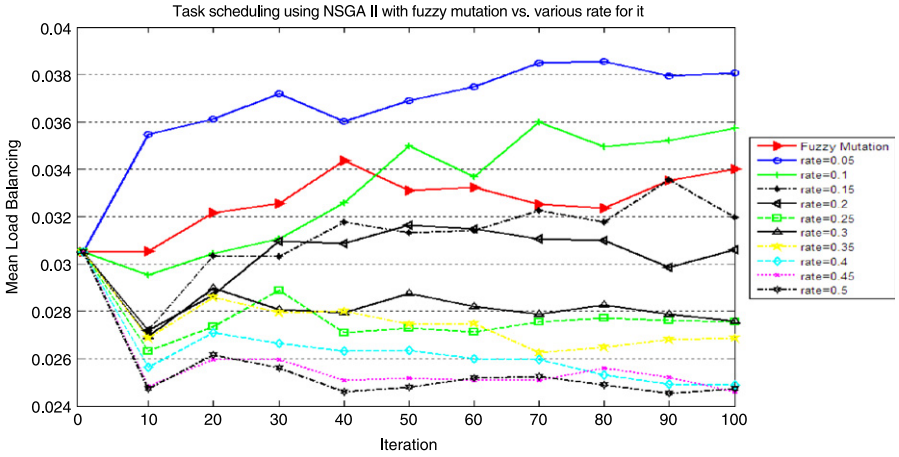

Fig. 15. Diagram of mean of all members in terms of the mean square deviation of resources utilization.

Makespan and Price. Furthermore Pareto-optimal front is created by NSGA-II with fuzzy mutation rate much faster than others. The average number of iterations in 10 times experiments for creating complete Pareto front with all members of the different populations by every mutation probability is reported in Fig. 11.

Figs. 9 and 10 demonstrate the superiority of the proposed method. In addition, we analyzed the proposed method and compared it with fixed rates for any objective separately in the optimization process. We followed Makespan and Price diagrams during the optimization process and reported in Figs. 12 and 13. The results indicate the superiority of the fuzzy adaptive rate for the mutation operator.

Various rates of mutation operator are compared in Figs. 12 and 13. According to the figures, when the mutation rate is low (0.05), the quality of solutions is better in terms of Makespan and is worse in terms of the Price and vice versa. In this comparison, it can be seen that an efficient adaptive rate can use this feature bilaterally and has the best performance and quality in terms of both Makespan and Price. The mean square deviation of resources utilization diagrams are depicted in Figs. 14 and 15. Clearly, these are not suitable solutions in terms of load balancing for the grid system. Figs. 12 and 14–16 show that the Makespan optimization is independent of the load balancing and has no effect on the load balancing. So according to the experiments these two objectives have no relation to each other.

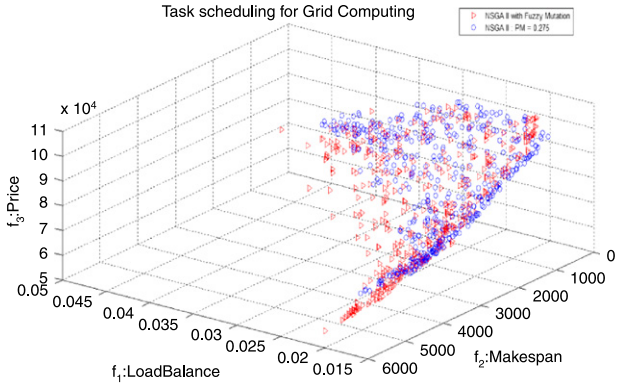

Fig. 16. Obtained Pareto-optimal fronts, population size: 500.

4.2. Simulation of NSGA-II with fuzzy mutation operator for optimizing Makespan, Price and load balancing

The load balancing objective has advantages including reduction of response time and increment efficiency of resources and system. Hence, optimizing these three objectives as multi-objective optimization is useful to the extent. Because of the three-dimensional Pareto front, to avoid figure overcrowding, rather than a three-dimensional Pareto front for each mutation rate, a three-dimensional Pareto front for the proposed method, and a

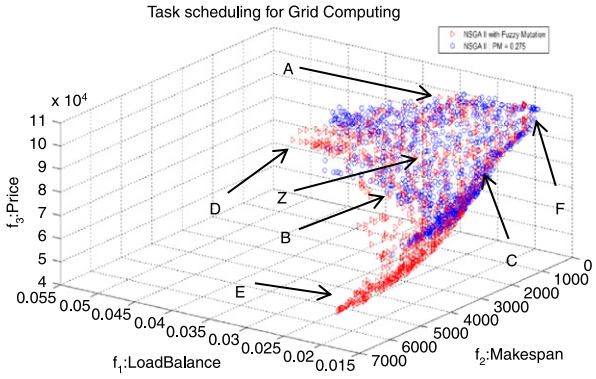

Fig. 17. Obtained Pareto-optimal fronts, population size: 1000.

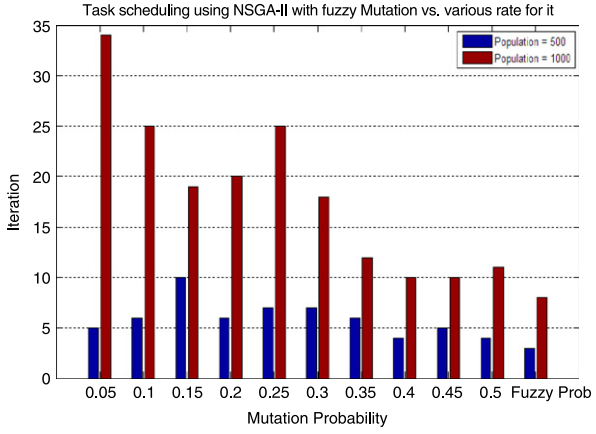

Fig. 18. Number of iterations for creating the complete Pareto front in various sizes of population.

Pareto front with a mean value for the mutation rates (here 0.275) are compared. Like before, for simulating algorithm with adaptive fuzzy mutation rate, two functions binary variance of genes values (Eq. (15)) and variance of average of objectives values (Eq. (18)) are used to generate inputs of the fuzzy system. The output of this fuzzy system is the mutation rate. Problem parameters and genetic operators for three objectives optimization are shown in Tables 6 and 7. Optimization results are shown in Figs. 16 and 17.

In the analysis of these Pareto fronts, the points A, B and C in Fig. 17 are the best solutions in terms of only one objective. For example point A shows the best members on the edge of Pareto-optimal front in terms of Makespan while the edges B and C are the best solutions in terms of Price and Load balancing respectively. The points F are the best solutions only in terms of both two objectives Makespan and Load balancing. The points E are the best solutions in terms of Price and Load balancing and the points D are the best solutions in terms of Makespan and Price. Also the points Z show the best solutions in terms of all three objectives. For example, a user may due to financial and time constraints choose a good solution in terms of Price and Makespan which is not suitable in terms of Load balancing (Points D in Fig. 17). Load balancing as Makespan and Price is not important for the user; therefore it is desirable to all solutions to have a good Load balancing. In this stage also Pareto-optimal front is created by NSGA-II with fuzzy mutation rate much faster than others. The average number of iterations in 10 times experiments for creating complete Pareto front with all members of the different populations by every mutation probability is reported in Fig. 18.

We set once the population size to 500 and 1000 and the fixed mutation rate to 0.275 to show the Pareto optimal front as a layer in Figs. 16 and 17 for a clear comparison so that superiority of the proposed method is observed in these two figures. Then the experiment is performed again with the various rates of mutation as the first stage. We compared the proposed method with fixed rates in every objective in the optimization process separately. Figs. 19–21 show Makespan, Price and Load balancing diagrams during the optimization process.

Table 7
Problem parameters.

Parameter	Value
Population size	500, 1000
Number of generations	100
Number of tasks	500
Size of tasks	20–100
Number of resources	50
Price for resources	1–5
Processing speed of resources	2–10

Table 8
Genetic operators.

Algorithm	Parameters		
	Crossover probability	Mutation probability	Bit mutation probability
NSGA II	0.5–1	0.2	0.2
Fuzzy NSGA II	pXover (Fig. 8)	0.2	0.2

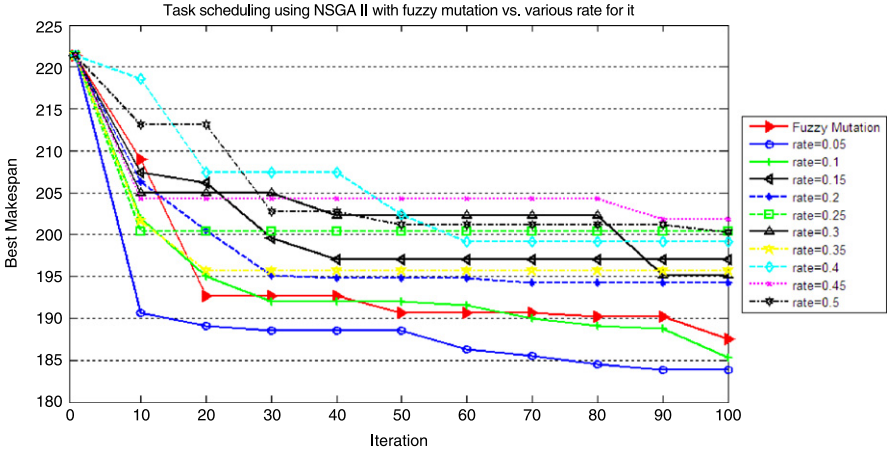

Fig. 19. Diagram of the best values of makespan in the mutation rate experiment.

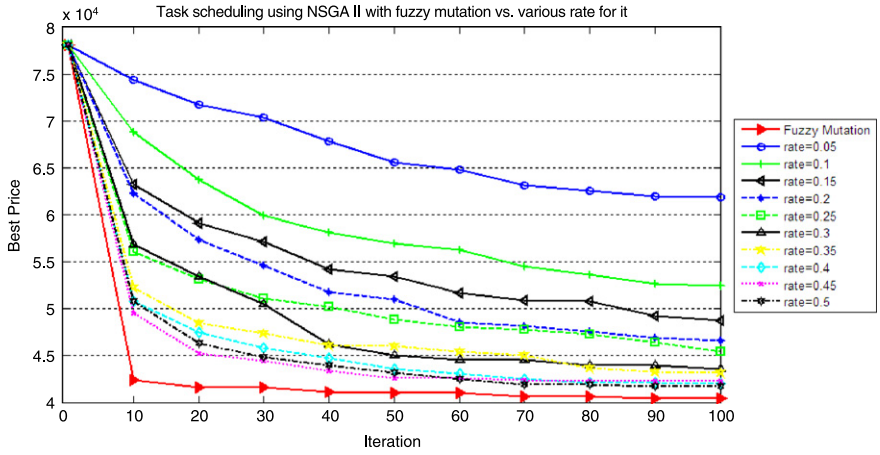

Fig. 20. Diagram of the best values of price in the mutation rate experiment.

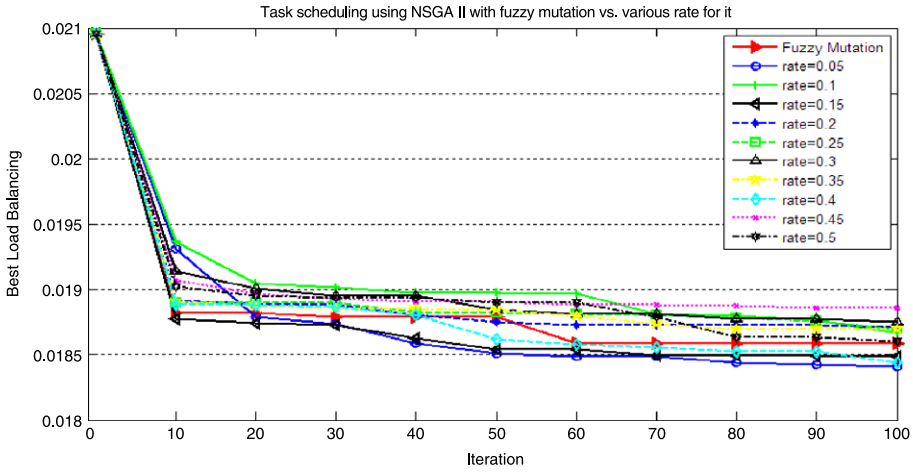

Fig. 21. Diagram of best values of the mean square deviation of resources utilization.

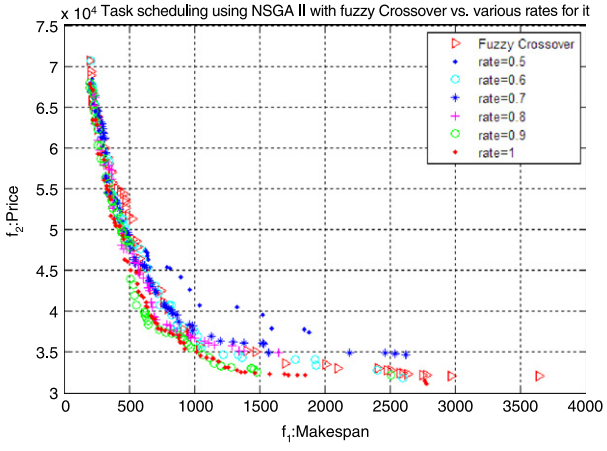

Fig. 22. Obtained Pareto-optimal fronts, population size: 100.

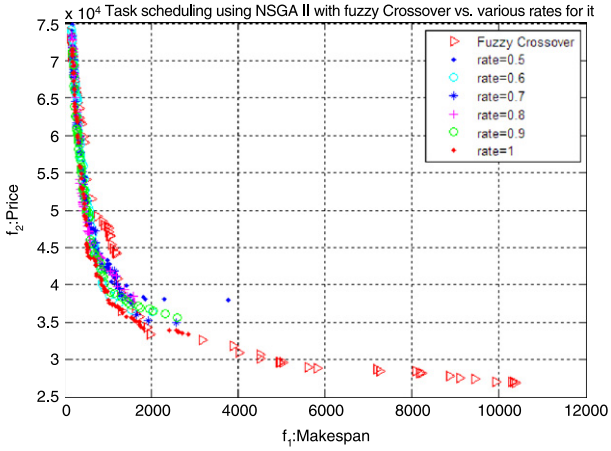

Fig. 23. Obtained Pareto-optimal fronts, population size: 200.

4.3. Simulation of NSGA-II with fuzzy crossover operator for optimizing the load balancing objective indirectly

In this study, we satisfied one special objective indirectly by the adaptive crossover rate along with the optimality of other objective. In [16] the special property of crossover and mutation operators has been introduced explorative and exploitive respectively. Explorative power of crossover and exploitive power of mutation change by changing crossover and mutation rates respectively. With this information we are able to obtain our goals by controlling the genetic operators. However all three objectives are essential in market-based grid computing, but two objectives Makespan and Price are very important for users and decision makers, so decision making for them will be difficult when there are three ob-

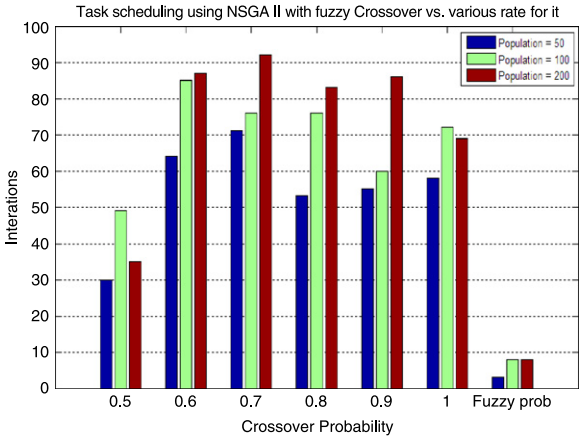

Fig. 24. Number of iterations for creating the complete Pareto front in various sizes of population.

jectives. Because there are solutions which are good in terms of two objectives Makespan and Price but are not suitable in terms of load balance (points D in Fig. 17), it is possible to select the solution which is not suitable in terms of load balance. Therefore we intend to apply the load balancing on all solutions which are optimizing in terms of two objectives Makespan and Price. So there will be solutions which have been optimized in terms of Makespan and Price and also have been satisfied in terms of the load balancing simultaneously. In fact, two-objective optimization in the task scheduling problem along with proposed fuzzy crossover balances the load across all resources for all schedules existing in the two dimensional Pareto front. While three-objective optimization will not be able to create such a two dimensional Pareto front because a part of three-dimensional Pareto front which is optimal in terms of Price and Makespan has not the suitable load balance (points D in Fig. 17). For simulating algorithm with variance-based fuzzy crossover rate, proposed functions for calculating the inputs of fuzzy system are the integer variance of resources frequency in schedules (Eq. (20)) and variance of average of objectives values (Eq. (21)). The output of the fuzzy system is the probability of crossover in population. Problem parameters and genetic operators for two-objective optimization are shown in Tables 5 and 8.

Obtained Pareto optimal fronts from the different sizes of population are shown in Figs. 22 and 23. The quality of the Pareto optimal solutions obtained from NSGA-II with various rates for crossover can be seen in these figures.

In this stage also Pareto-optimal front is created by NSGA-II with fuzzy crossover rate much faster than others. The average number of iterations in 10 times experiments for creating complete Pareto front with all the members of the different populations with different probabilities of crossover is reported in Fig. 24.

The superiority of the proposed method is observed in Figs. 22 and 23. In addition, we analyzed the proposed method and compared it with fixed rates of crossover for every objective separately

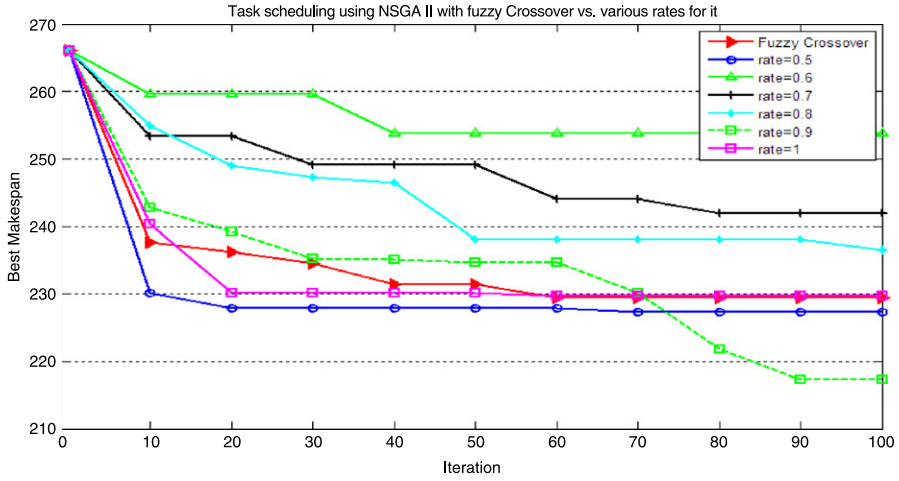

Fig. 25. Diagram of the best values of makespan in the crossover rate experiment.

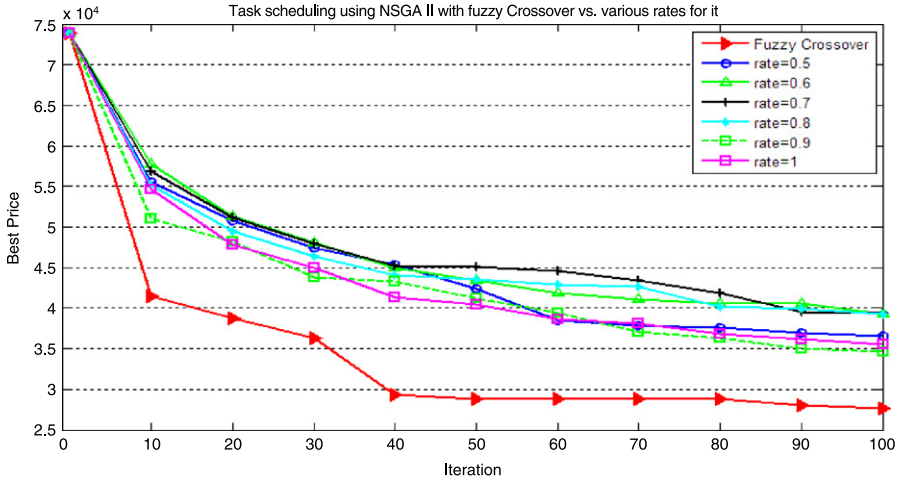

Fig. 26. Diagram of the best values of price in the crossover rate experiment.

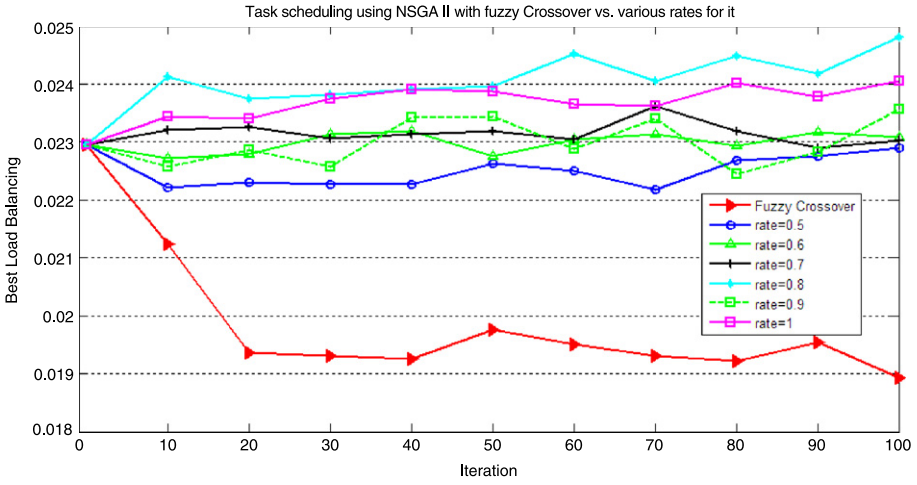

Fig. 27. Diagram of best values of the mean square deviation of resources utilization.

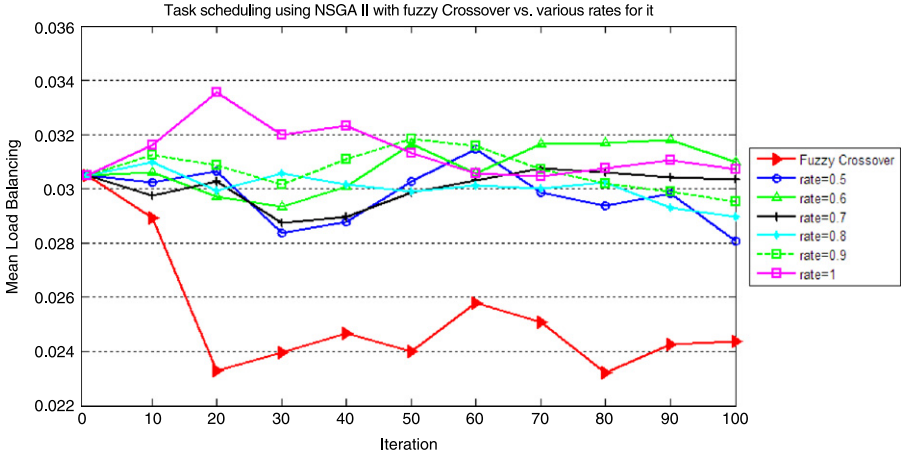

Fig. 28. Diagram of mean of all members in terms of the mean square deviation of resources utilization.

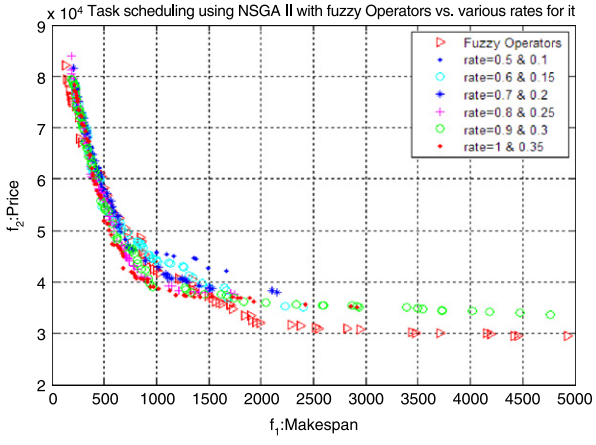

Fig. 29. Obtained Pareto-optimal fronts, population size: 100.

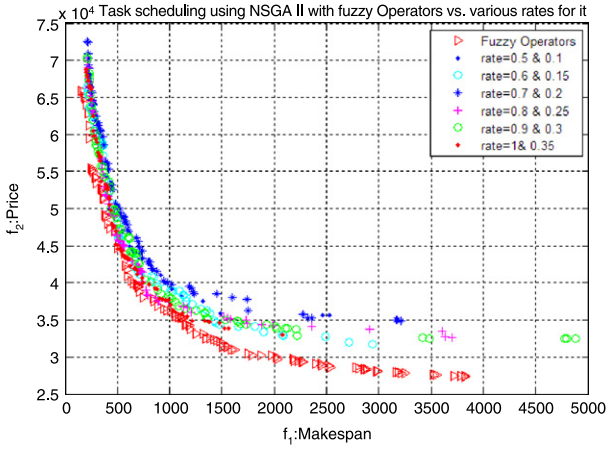

Fig. 30. Obtained Pareto-optimal fronts, population size: 200.

in the optimization process. We followed Makespan, Price and Load balance diagrams during the optimization process in Figs. 25–28. The results indicate the superiority of the fuzzy adaptive rate in terms of load balancing especially.

In this stage of the experiments the load balance was improved without the direct use of optimization mechanism, but the quality of all solutions should also be improved. Therefore we use the fuzzy variance-based crossover rate along with fuzzy mutation rate in the next stage to improve both quality and performance simultaneously.

4.4. Simulation of NSGA-II with proposed fuzzy variance-based operators

In the fourth stage of the experiments, fuzzy crossover and mutation rates are used for optimizing two objectives Makespan and

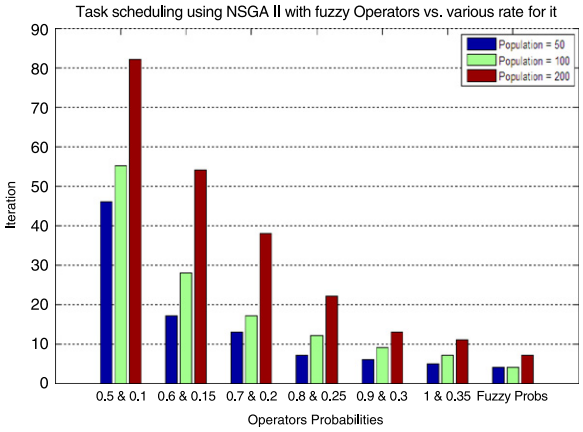

Fig. 31. Number of iterations for creating the complete Pareto front in various sizes of population.

Price and satisfying load balancing indirectly. Due to the low quality of obtained solutions in the previous stage in terms of Makespan (Fig. 25), in this stage we use both fuzzy crossover and fuzzy mutation rates to improve the performance of algorithm, diversity and quality of solutions and satisfy load balancing indirectly in order to reduce the complexity and ease in making decision for users. In the simulation, the algorithm with fuzzy adaptive crossover and mutation rates uses two functions for calculating the inputs of the crossover fuzzy system including integer variance of frequency of involved resources in scheduling (Eq. (20)) and variance of average of objectives values (Eq. (21)). The output of this fuzzy system is the probability of crossover in population. Also two functions binary variance of genes values (Eq. (15)) and variance of average of objectives values (Eq. (18)) are used to generate inputs of the mutation fuzzy system. The output of this system is the mutation rate. Tables 5 and 9 show problem parameters and genetic operators for two-objective optimization in this stage.

Table 9
Genetic operators.

Algorithm	Parameters		
	Crossover probability	Mutation probability	Bit mutation probability
NSGA II	0.5–1	0.1–0.35	0.1–0.35
Fuzzy NSGA II	pXover (Fig. 8)	PM (Fig. 6)	PM (Fig. 6)

The superiority of the proposed method is observed in Figs. 29 and 30. In this stage, the NSGA-II with fuzzy crossover and mutation rates creates the Pareto-optimal front much faster than three previous stages. The average number of iterations during 10 times experiment for creating complete Pareto front with all members of the different population with different probabilities for two genetic operators is reported in Fig. 31.

In addition, we analyzed the proposed method and compared it with fixed rates in every objective separately. We followed

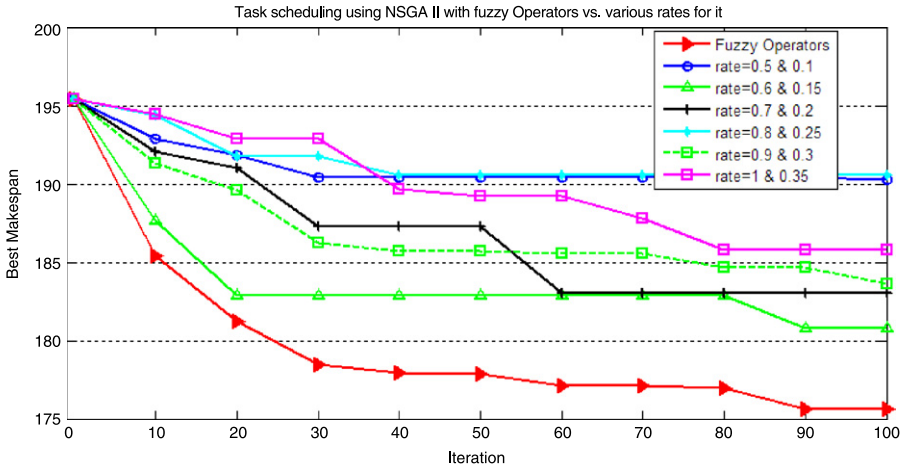

Fig. 32. Diagram of the best values of makespan in different rates experiment.

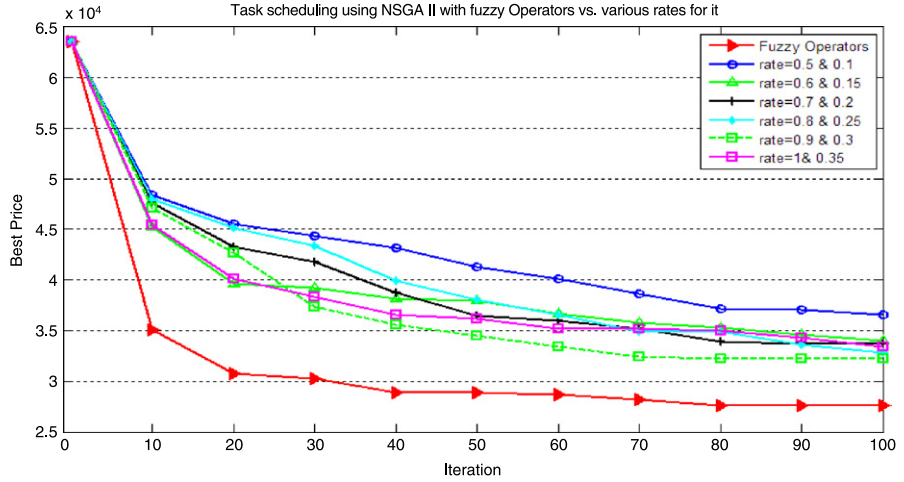

Fig. 33. Diagram of the best values of price in different rates experiment.

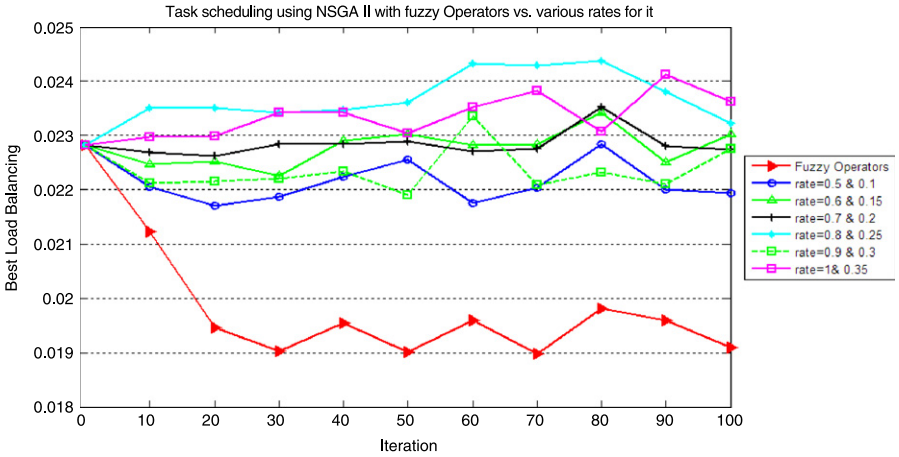

Fig. 34. Diagram of best values of the mean square deviation of resources utilization.

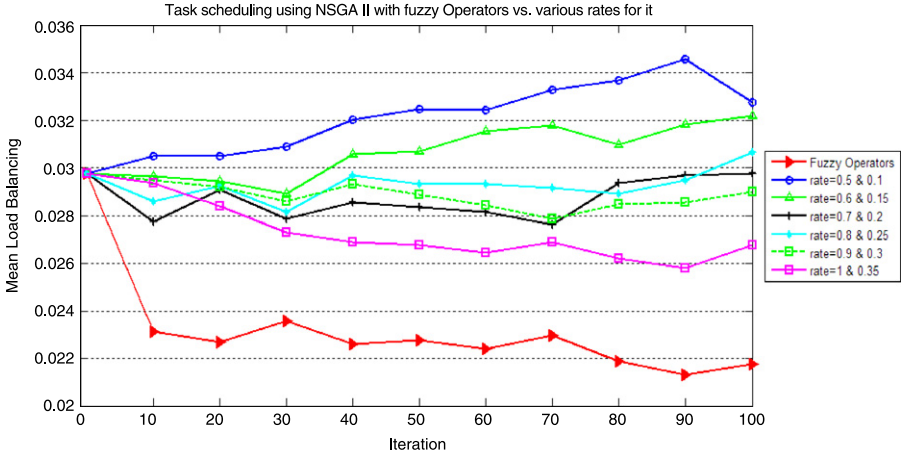

Fig. 35. Diagram of mean of all members in terms of the mean square deviation of resources utilization.

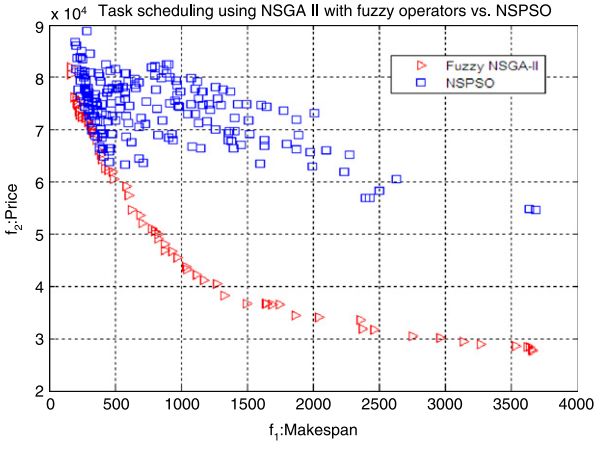

Fig. 36. Three dimensional image of NSPSO Pareto front and two-dimensional Pareto optimal front of Fuzzy NSGA-II on the two-dimensional plane.

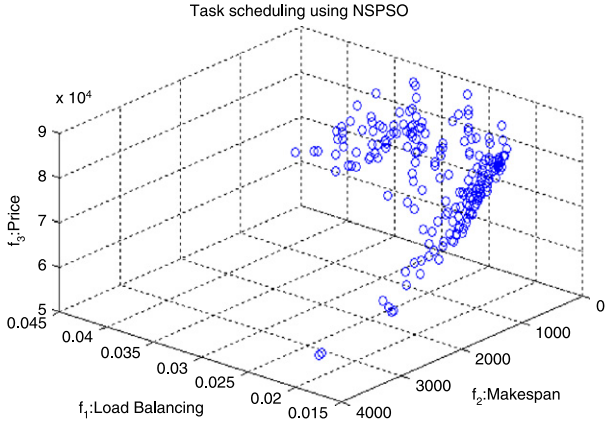

Fig. 37. NSPSO Pareto optimal front.

Makespan and Price and Load balancing diagrams during the optimization in Figs. 32–35.

4.5. Comparing the proposed variance-based fuzzy NSGA-II with NSPSO

In this experiment, we compare our algorithm with NSPSO that proposed in [10]. The NSPSO method which proposed for task scheduling in [25] is used in this stage of the experiments. The same initial population for both algorithms is set to 200 and created randomly. Both algorithms are run for 100 iterations. Other problem parameters and genetic operators are also according to Tables 5 and 9. $C1$ and $C2$ for NSPSO are set to 2.0 and other parameters of NSPSO are set according to [30]. In this experiment, our algorithm is used to optimize two objectives Makespan and Price and satisfy load balancing objective indirectly and NSPSO is used to optimize all three mentioned objectives. In general, the execution time of NSPSO is less than NSGA-II, but since we have reduced the number of optimization objectives to two objectives in our proposed fuzzy NSGA-II rather than three objectives, and in addition, the fuzzy

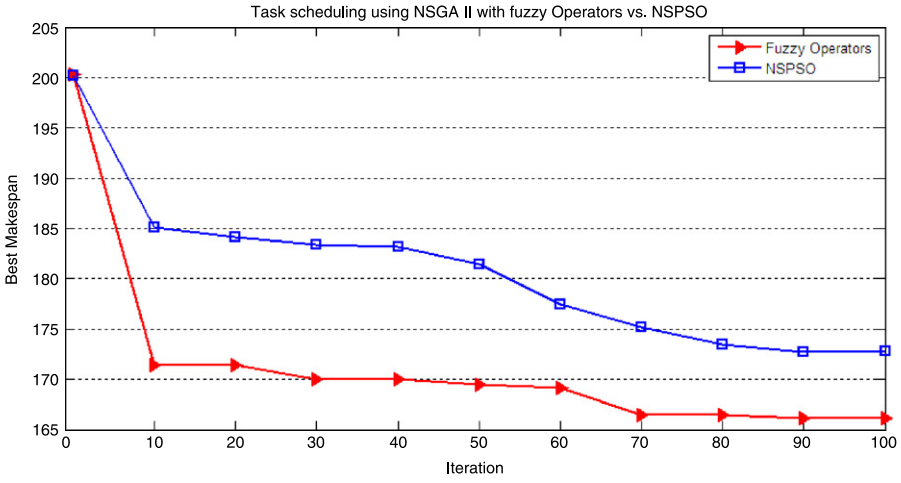

Fig. 38. Diagram of the best values of makespan.

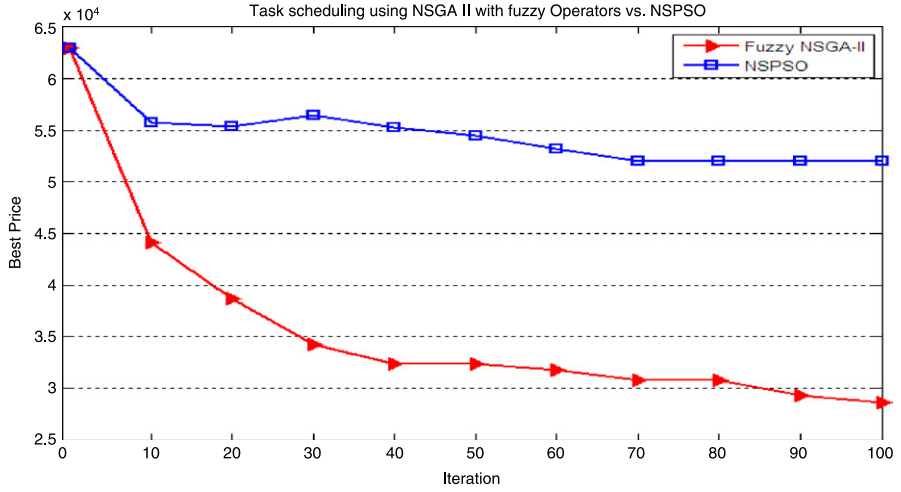

Fig. 39. Diagram of the best values of price.

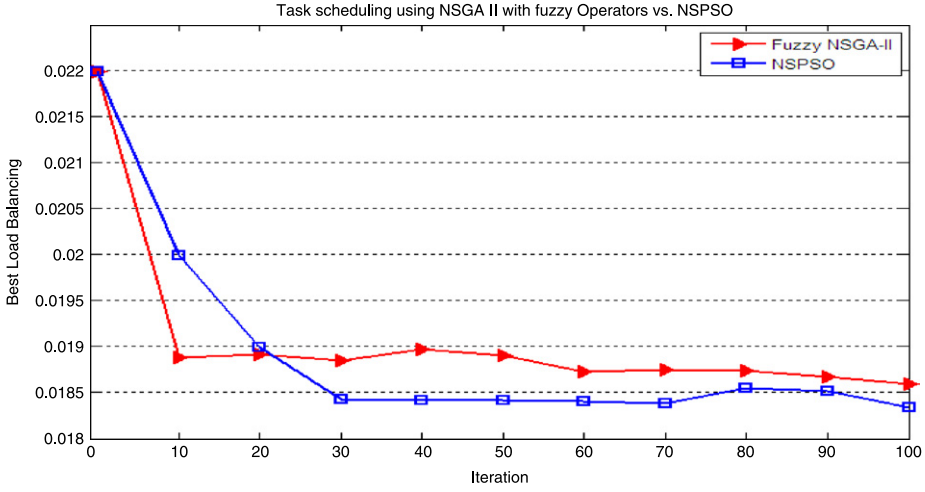

Fig. 40. Diagram of best values of the mean square deviation of resources utilization.

adaptive operators in our approach reduce the number of iterations required for creating the complete Pareto optimal front, the complexity and the execution time of our fuzzy NSGA-II will be less than NSPSO. The simulation results show that our algorithm converges to Pareto optimal front with more quality and faster than NSPSO.

Fig. 36 shows the image of three-dimensional Pareto optimal front of NSPSO along with two-dimensional Pareto optimal front of our algorithm. We depicted two Pareto optimal fronts with different dimensions together in Fig. 36 for a clear comparison. the Pareto fronts of NSPSO and our algorithm are three-dimensional and two-dimensional respec-

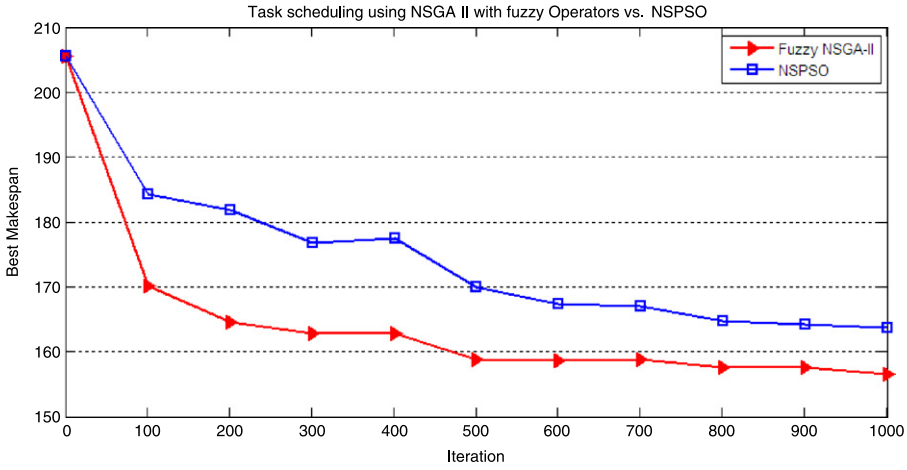

Fig. 41. Diagram of the best values of makespan, 1000 iterations.

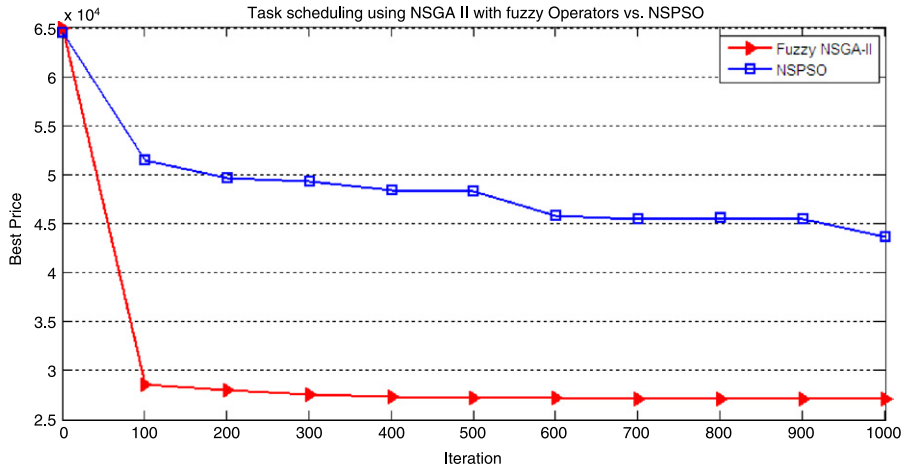

Fig. 42. Diagram of the best values of price, 1000 iterations.

tively. Fig. 37 shows three-dimensional Pareto optimal front of NSPSO. Makespan and Price and load balancing diagrams during the optimization process can be observed in Figs. 38–40 separately. The superiority of our fuzzy NSGA-II is observed in Figs. 36, 38 and 39 in terms of two objectives Makespan and Price. Fig. 40 shows the diagram of the best values of the mean square deviation of resources utilization for both algorithms. The NSPSO optimizes directly this objective through three-objective optimizing, but in our fuzzy NSGA-II, this objective is satisfied indirectly only through two-objective optimizing, so this result is acceptable.

Although the NSPSO requires less execution time for each iteration, it requires more iterations to create the complete Pareto optimal front compared to our approach. Experimental results in [25,30,12] show that the particle swarm optimization (PSO) algorithms perform better than genetic algorithms in very high iterations, for example 2000 iterations in [30] or 18 000 iterations in [12], while in low iterations, genetic algorithms (GA) perform better than PSO (refer to [25,30,12]). In fact, GA converges to optimal solutions faster than PSO in the task scheduling problem. In this experiment, the execution time of NSPSO was about 6 times lesser than our algorithm. But the quality of solutions and the convergence rate of our algorithm were better than NSPSO. We also compared two algorithms in terms of Price and Makespan objectives for 1000 iterations to ensure the quality of our solutions in high iterations. Figs. 41 and 42 prove the superiority of our algorithm in terms of the quality of solutions and the convergence rate even in high iterations. So as mentioned, although NSPSO has less execution time, it requires several times more iterations to generate Pareto optimal front near obtained Pareto optimal front from the variance-based fuzzy NSGA-II.

5. Conclusions

In this paper NSGA-II with fuzzy adaptive operators was implemented for task scheduling in the market-based grid environment and compared with general NSGA-II with fixed rates for operators and NSPSO. It is clear that the quality of schedules achieved by the proposed method is better than the quality of schedules achieved by the standard NSGA-II. By comparing the performance of the algorithms it is seen that NSGA-II with fuzzy adaptive operators maintains a uniform spread of solutions in the obtained Pareto-optimal front. Spread and span of solutions in the proposed method is more than others. Our method enhances the intelligence of Genetic Algorithms in adapting to environment using the intelligent rate for genetic operators and so causes the better performance and quality. In the experiments, first the effect of the mutation rate on the Pareto front was observed. We applied the fuzzy method to propose the adaptive mutation rate in order to get a better Pareto front. Then we studied the load balancing problem. We use the fuzzy adaptive rate for crossover and Makespan objective optimization to achieve this goal. For this purpose, we defined two functions that used the information of the individuals existing in the Pareto front and generated the inputs of the fuzzy system. Then this fuzzy system was able to direct and control the rate of crossover operator. In addition, the Makespan

objective also indirectly would play role. In this case, it was observed that the load balancing was satisfied well by using the fuzzy adaptive rate for crossover and fixed rate for mutation, in fact, each solution which is selected by the user, it will satisfy the load balancing, but the two-dimensional Pareto optimal front will not have a good quality. So we took advantages of the two fuzzy adaptive rates for crossover and mutation. As a result, the Pareto-optimal front is created by our method much faster and with higher quality and diversity. Briefly we could optimize three objective functions using two objective functions. This method reduces computation, so that if numbers of iterations and population are p and q, respectively, we could reduce complexity of computation $p \times q$ times by removing the third objective function i.e. load balancing while it was satisfied by the fuzzy system and Makespan optimization indirectly. Also our algorithm was compared with the NSPSO and proved that variance-based fuzzy NSGA-II converges to Pareto-optimal solutions faster and with more quality.

References

[1] P. Berenbrink, T. Friedetzky, Z. Hu, A new analytical method for parallel, diffusion-type load balancing, J. Parallel Distrib. Comput. 69 (1) (2009) 54–61 (ELS).

[2] A.G. Bronevich, W. Meyer, Load balancing algorithms based on gradient methods and their analysis through algebraic graph theory, J. Parallel Distrib. Comput. 68 (2) (2008) 209–220 (ELS).

[3] R. Buyya, J. Giddy, D. Abramson, An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications, in: Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 2000), August 1, 2000, Kluwer Academic Press, Pittsburgh, USA, 2000.

[4] J. Chandra Patni, M.S. Aswal, O. Prakash Pal, A. Gupta, Load balancing strategies for grid computing, IEEE (2011) 239–243.

[5] M.I. Daoud, N. Kharma, A high performance algorithm for static task scheduling in heterogeneous distributed computing systems, J. Parallel Distrib. Comput. 68 (4) (2008) 399–409 (ELS).

[6] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.

[7] A. Jaszkiewicz, J. Branke, Interactive multiobjective evolutionary algorithms, in: Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 179–193.

[8] S. Kardani-Moghaddam, F. Khodadadi, R. Entezari-Maleki, A. Movaghar, A hybrid genetic algorithm and variable neighborhood search for task scheduling problem in grid environment, in: International Workshop on Information and Electronics Engineering, IWIEE, Procedia Engineering 29, 2012, pp. 3808–3814.

[9] B.T.B. Khoo, B. Veeravalli, T. Hung, C.W.S. See, A multi-dimensional scheduling scheme in a grid computing environment, J. Parallel Distrib. Comput. 67 (6) (2007) 659–673 (ELS).

[10] X. Li, A non-dominated sorting particle swarm optimizer for multi-objective optimization, in: Proceeding of Genetic and Evolutionary Computation Conference 2003, GECCO'03, USA, 2003.

[11] Y. Li, Y. Yang, R. Zhu, A hybrid load balancing strategy of sequential tasks for computational grids, in: International Conference on Networking and Digital Society, IEEE, 2009.

[12] H. Liu, A. Abraham, A.E. Hassanien, Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, Future Gener. Comput. Syst. (2010) 1336–1343 (ELS).

[13] S.A. Ludwig, A. Moallem, Swarm intelligence approaches for distributed load balancing on the grid, J. Grid Comput. 9 (3) (2011) 279–301 Springer Science.

[14] J. Ma, Lanzhou, A novel heuristic genetic load balancing algorithm in grid computing, in: Second International Conference on Intelligent Human–Machine Systems and Cybernetics, 2010.

[15] N.K. Madavan, Multiobjective optimization using a Pareto differential evolution approach, IEEE (2002).

[16] M. Mitchell, An Introduction to Genetic Algorithms, A Bradford Book the MIT Press, Cambridge, Massachusetts, London, England 1999, Fifth printing.

[17] F.A. Omara, M.M. Arafa, Genetic algorithms for task scheduling problem, J. Parallel Distrib. Comput. 70 (1) (2010) 13–22 (ELS).

[18] S. Padhee, N. Nayak, S.K. Panda, S.S. Mahapatra, Multi-objective parametric optimization of powder mixed electro-discharge machining using response surface methodology and non-dominated sorting genetic algorithm, Indian Acad. Sci., Sadhana 37 (Part 2) (2012) 223–240.

[19] Y. Pei, Z. Univ, A MOPSO approach to grid workflow scheduling, in: Asia-Pacific Conference on Wearable Computing Systems (APWCS), IEEE, 2010, pp. 403–406.

[20] H. Peng, Q. Li, One kind of improved load balancing algorithm in grid computing, in: International Conference on Network Computing and Information Security, 2011.

[21] S. Prakash, D.P. Vidyarthi, Load balancing in computational grid using genetic algorithm, Adv. Comput. 1 (1) (2011) 8–17. http://dx.doi.org/10.5923/j.ac.02.

[22] G.S. Sadasivam, V.V. Rajendran, An efficient approach to task scheduling in computational grids, Int. J. Comput. Sci. Appl. 6 (1) (2009) 53–69 Techno mathematics Research Foundation.

[23] R. Salimi, N. Bazrkar, M. Nemati, Task scheduling for computational grids using NSGA II with fuzzy variance based crossover, Adv. Comput. (2013) 22–29. http://dx.doi.org/10.5923/j.ac.20130302.02.

[24] R. Salimi, H. Motameni, H. Omranpour, Task scheduling with load balancing for computational grid using NSGA II with fuzzy mutation, in: 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012, pp. 79–84.

[25] G. Subashini, M.C. Bhuvaneswari, Non-dominated particle swarm optimization for scheduling independent tasks on heterogeneous distributed environments, Int. J. Adv. Soft Comput. Appl. 3 (1) (2011).

[26] G. Subashini, M.C. Bhuvaneswari, NSGA-II with controlled elitism for scheduling tasks in heterogeneous computing systems, Int. J. Open Probl. Compt. Math. (ISSN: 1998-6262) 4 (1) (2011).

[27] A. Touzene, S. Al-Yahai, H. AlMuqbali, A. Bouabdallah, Y. Challal, Performance evaluation of load balancing in hierarchical architecture for grid computing service middleware, IJCSI Int. J. Comput. Sci. Issues (ISSN: 1694-0814) 8 (2) (2011).

[28] B. Ucar, C. Aykanat, K. Kaya, M. Ikinci, Task assignment in heterogeneous computing systems, J. Parallel Distrib. Comput. 66 (1) (2006) 32–46 (ELS).

[29] H. Vahdat-Nejad, R. Monsefi, M. Naghibzadeh, A new fuzzy algorithm for global job scheduling in multiclusters and grids, in: CIMSA, International Conference on Computational, Intelligence for Measurement Systems and Applications, IEEE, 2007.

[30] L. Zhang, Y. Chen, R. Sun, S. Jing, B. Yang, A task scheduling algorithm based on PSO for grid computing, Int. J. Comput. Intell. Res. (ISSN: 0973-1873) 4 (1) (2008) 37–43.

[31] A.Y. Zomaya, Yee-Hwei Teh, Observations on using genetic algorithms for dynamic load-balancing, IEEE Trans. Parallel Distrib. Syst. 12 (9) (2001).

Reza Salimi received B.S. degree in Computer Science from Taft University of Payam-Noor, Yazd, Iran, and M.S. degree in Computer Science in Tabari University of Babol, Iran, in 2008 and 2012, respectively. His current research interests include Evolution Algorithms, parallel processing, Economic Grid Computing, and Task scheduling in Dynamic Environments.

Homayun Motameni received B.S. degree in Computer Engineering-Software Engineering in Shahid Beheshti of Tehran University and M.S. degree in Computer Engineering-Machine Intelligence from Islamic Azad University-Science and Research Branch in 1995 and 1998, respectively. He received Ph.D. degree in Computer Engineering-Software Engineering from Islamic Azad University-Science and Research Branch in 2007. His current research interests include Evolution Algorithms, Petri Net, software systems modeling and evaluation using Petri Net, and machine learning.

Hesam Omranpour received B.S. degree in Computer Engineering-Software Engineering from Iran, University of Science and Technology and M.S. degree in Artificial Intelligence from Amirkabir University of Technology in 2006 and 2009, respectively. He is now working towards his Ph.D. degree in Artificial Intelligence in Amirkabir University of Technology, Tehran, Iran. His current research interests include Evolution Algorithms, Robotic, and Computational Models for brain.