Sādhanā (2019)44:215
https://doi.org/10.1007/s12046-019-1200-3

MCAMC: minimizing the cost and makespan of cloud service using non-dominated sorting genetic algorithm-II

A SATHYA SOFIA[1,*], P EMMANUEL NICHOLAS[2] and P GANESHKUMAR[3]

[1]Department of Computer Science and Engineering, PSNA College of Engineering and Technology, Dindigul, India
[2]Department of Mechanical Engineering, PSNA College of Engineering and Technology, Dindigul, India
[3]Department of Information Technology, PSNA College of Engineering and Technology, Dindigul, India
e-mail: sathyasofia@gmail.com

MS received 28 November 2016; revised 21 June 2019; accepted 9 September 2019

Abstract. The number of cloud users and their aspiration for completion of tasks at less energy consumption and operating cost are rapidly increasing. Hence, the authors of this paper aim to minimize the makespan and operating cost by optimally scheduling the tasks and allocating the resources of cloud service. The optimum task scheduling and resource allocation are obtained for each objective function using the simple genetic algorithm. Further, the non-dominated solutions of the dual objectives are obtained using the non-dominated sorting genetic algorithm-II, the most successful multi-objective optimization technique. A complex cloud service problem consisting of ten tasks, fifteen subtasks and fifteen heterogeneous resources is considered to investigate the proposed method. The numerical results obtained in the single objective and multi objective optimization problems show that the makespan and the operating cost are significantly reduced using the simple genetic algorithm and a wide range of non-dominated solutions are obtained in the multi-objective optimization problem, by which the cloud users shall be benefitted to choose the most appropriate solution based on the other design constraints they have.

Keywords. Cloud computing; task scheduling and resource allocation; optimization; NSGA-II.

1. Introduction

Cloud Computing is a dynamic provisioning of IT capabilities (hardware, software, or services) whereby virtualized applications, software, platforms, computation and storage are given in a pay-as-you-go manner. As it eliminates the initial investment cost for a private computing infrastructure as well as provides the ability to accommodate workloads with very large peak-to-average ratios, the cloud users are increasing tremendously day by day. The service providers and customers can concurrently receive the economic benefits, if the available resources are efficiently scheduled. Hence, workflow scheduling is indispensable to save the cost paid by the cloud users as well as to minimize the running time of the cloud facilities. For the past one-decade, various scheduling criteria such as time, cost, reliability, energy and security were considered in the cloud computing scheduling processes suggested by Smanchat and Viriyapant [1]. Li *et al* [2] proposed a new framework ECommer and suggests Cost Conscious Scheduling Heuristic (CCSH). It constructed a priority list of tasks and assigns a highest priority value task to the cost-efficient virtual machine. A cost-efficient task-scheduling algorithm was proposed by Su *et al* [3] to dynamically map the tasks to the most cost-efficient virtual machines based on the non-critical tasks. However, the resources such as network bandwidth and memory space were not taken into the account. A framework called VM Planner was constructed by Fang *et al* [4] to minimize the cost relevant to the power consumption of the network elements in data centers. Guo *et al* [5] proposed an electricity price-aware and cooling efficiency-aware load balancing to reduce the electricity cost for the geographically distributed data centers.

Malawski *et al* [6] maximized the amount of work completed subject to the time and cost constraints. Calheiros and Buyya [7] proposed a cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds. A cost efficient coordinated scheduling mechanism was suggested by Li *et al* [8] for the benefit of cloud service providers to lease VM instances in order to process hybrid workloads. A cloud service request model was developed by Lin and Lu [9] to minimize the rental cost of the cloud infrastructure by improving the resource utilization. They proposed the SHEFT algorithm (Scalable-Heterogeneous-Earliest-Finish-Time algorithm), which optimized the

*For correspondence
Published online: 03 October 2019

workflow execution time for a cloud computing environment. Mezmaz *et al* [10] minimized the energy consumption of cloud computing infrastructures using the dynamic voltage scaling (DVS) method. However, the makespan and other costs relevant to the utilization of the cloud facilities were not considered. An improved cost-based algorithm was proposed by Selvarani and Sadhasivam [11] for task scheduling in cloud computing which schedules tasks based on their cost to different resources. Cost of services varied for different tasks based on their complexity and this algorithm considers resource cost and processing capability of resources. They grouped tasks based on the processing capacity and selected some best resources to schedule them in such a way to reduce the cost.

Chaisiri *et al* [12] dealt with an optimal cloud resource provisioning (OCRP) algorithm to minimize the reservation, expending, and on-demand provisioning costs which can optimally adjust the tradeoff between reservation of resources and allocation of on-demand resources. Thomas *et al* [13] proposed an approach for scheduling in cloud computing by considering two parameters: task length and user priority and the algorithm was based on credit system which did not take into account deadline of tasks. A price prediction algorithm was suggested by Zhou *et al* [14] to predict the price of virtual resources and designed a novel rental decision-making algorithm to select the most profitable resource from a set of cloud resources. Research metrics such as energy efficiency, SLA-awareness, network load minimization, load balancing, profit maximization, hybrid cloud computing, and mobile cloud computing were considered in resource management techniques suggested by Mustafa *et al* [15]. An algorithm named honey bee behavior inspired load balancing was proposed by Krishna [16] to achieve well balanced load across virtual machines for maximizing the throughput. However, the cost factor was not considered as one of the objectives.

Casas *et al* [17] proposed an algorithm called Balanced and file Reuse–Replication Scheduling (BaRRS) for scheduling scientific application workflows optimally considering execution time and cost of executing the workflows in cloud. The main drawback in this was that they need to sort low quality solutions prior the scheduler analyzes it. Juarez *et al* [18] characterized a dynamic scheduling system for real time applications in distributed computing platforms to reduce the energy consumption. Their work mainly concentrated on energy alone and need to calculate monetary cost connected with the use of VMs and profits for service providers. Maheshwari *et al* [19] presented performance models used multi-site workflow scheduling technique to calculate execution time on resources and active probes to categorize the attainable network throughput connecting sites without considering optimization parameters. Arabnejad *et al* [20] presented a heuristic scheduling algorithm considering quadratic time complexity including the two constraints time and cost but they failed to give importance to the runtime environment. Moschakis and Karatza [21] studied the use of metaheuristic optimization algorithms in heterogeneous clouds intended for scheduling of bag-of-tasks applications. But they did not perform any optimization at the inter-cloud level. But optimization techniques are needed in cloud for performing effective and efficient task scheduling.

Many researchers performed task scheduling and resource allocation on cloud computing environment to reduce either the cost or the makespan or the energy consumption. However, reducing the operating costs as well as the total completion time (makespan) of all the tasks are the most crucial demands in the field of cloud computing. Hence, they are considered as the multiple objectives in this paper during the optimization of task scheduling and resource allocation. The genetic algorithm, which is one of the predominant optimization techniques for the complex multi-objective optimization problems and has the ability to find the optimum design even from the huge design space has been considered to find the optimum designs which is suggested by Nicholas *et al* [22].

The main objective of this work is to optimize the task scheduling and work allocation to minimize the makespan and operating cost of the cloud service. A case study problem, which consists of maximum of ten tasks, fifteen subtasks per task and fifteen heterogeneous resources, is considered to study. Initially, each objective function is optimized separately using simple genetic algorithm and finally both objective functions are minimized simultaneously using non-dominated sorting genetic algorithm-II.

2. Materials and methods

In this section, the mathematical models are formulated to compute the operating cost and the makespan. Further, the objective functions, constraints and the design variables of the single objective and multi-objective optimization problems are framed and presented.

2.1 *The mathematical model to obtain the operating cost*

The resource providers generally charge for the usage of resources based on the configuration of the resource, energy consumption and time taken for the completion of the tasks. In general, each task comprises of several subtasks, which are to be executed in a sequential manner. In this paper, the cost for execution (C_P) and cost for receiving (C_R) a subtask on a particular resource are considered to obtain the total cost. If 'm' is the number of tasks to be carried out using the cloud service and each task consists of 'n' number of subtasks, then the total cost required to complete all the tasks is given by:

$$C_T = \sum_{i=1}^{m}\sum_{j=1}^{n} C_P + C_R \tag{1}$$

The execution and receiving costs are calculated based on the time required for the execution and receiving the subtask as given in Eq. (2).

$$\begin{aligned} C_P &= T_P \times R_P \\ C_R &= T_R \times R_R \end{aligned} \tag{2}$$

where, T_P and T_R are the processing time and receiving time of a subtask on an allocated resource. Similarly, R_P and R_R are the rents charged by the resource providers to process and receive the subtask, respectively.

2.2 *The mathematical model to obtain the makespan*

The makespan is the total time required to complete all the subtasks and tasks. In this paper, the processing time, waiting time and receiving time are considered to obtain the makespan value. The processing time is the time required to complete a particular subtask on a specific resource and it is usually found based on the data base size (usually in GB) of a subtask and the processing capacity of a resource (usually in GB/h). The receiving time is the time required to transfer the task to a resource, which is to be found based on the data base size (usually in GB) of a subtask and the transfer rate of the network (usually in MBPS). Sometimes, a resource may be idle until the previous subtask is completely executed in another resource. This idle time of a resource is known as the waiting time.

The processing time and the receiving time are calculated using Eq. (3).

$$\begin{aligned} T_{P(\text{in sec})} &= \frac{DBS_{\text{subtask}}(\text{in GB})}{PC_{\text{resource}}\left(\text{in } \frac{GB}{h}\right)} \times 3600 \\ T_{R(\text{in sec})} &= \frac{DBS_{\text{subtask}}(\text{in GB})}{TR_{\text{network}}\left(\text{in } \frac{MB}{\text{sec}}\right)} \times 10^3 \end{aligned} \tag{3}$$

where, DBS is the database size, P_C is the processing capacity and T_R is the transfer rate. The total time required for the i^th resource can be found using Eq. (4).

$$T_i = \sum T_P^i + \sum T_R^i + \sum T_W^i \tag{4}$$

If the number of resources used by a cloud user is 'r', then the makespan is given by:

$$\text{Makespan} = \max(T_1, T_2, T_3, \ldots, T_r) \tag{5}$$

2.3 *Formulation of the optimization problem*

The main aim of the optimization technique is to identify the best design from the given set of feasible designs. Hence, a design function or design criterion is required to compare all the feasible designs and to choose the best one and this design function or design criterion, which is used to compare all the feasible designs, is known as the objective function. However, any real world engineering problem has more than one design function and the most crucial one of those design functions is considered as the objective function. The other secondary design functions are treated as the design constraints. Similarly, the variables, which are used to maximize or minimize the design functions, are called as the design variables.

In this paper, minimizing the operating cost and makespan are considered as the design objectives as well as the design constraint, whereas the allocation of tasks to various heterogeneous resources is used as the design variable. The single and multi-objective optimization problem to find the optimum allocation of the resources is stated in Eq. (6).

$$\begin{aligned} &\text{Let,} \\ &\quad \text{the resources, } R = \{R_1, R_2, R_3, \ldots, R_R\} \\ &\quad \text{the tasks, } M = \{M_1, M_2, M_3, \ldots, M_M\} \\ &\qquad \text{where, } M_i = \{N_1, N_2, N_3, \ldots, N_N\} \\ &\text{The objective functions are:} \\ &\quad \text{Minimize the cost, } f_1 = C_{RM} : R \times M, \\ &\qquad \text{the cost for resource } R_i \text{ to do task } M_j \\ &\quad \text{Minimize the makespan, } f_2 = T_{RM} : R \times M, \\ &\qquad \text{the time for resource } R_i \text{ to do task } M_j \end{aligned} \tag{6}$$

2.4 *Optimization technique*

Among the various optimization techniques suitable for the task scheduling problems, genetic algorithm (GA) has been used by many researchers [23–29]. The main reason behind this is GA has a very little chance to get stuck in any local optimum values [22]. Hence, the authors of this paper use the simple genetic algorithm and non-dominated sorting genetic algorithm-II to carry out the single objective and multi-objective optimization problems. The operating cost and the makespan are considered as the objective function and design constraint during the single objective optimization in order to investigate the influence of task scheduling and resource allocation on the operating cost and makespan of a most complicate cloud service. Subsequently, the bi-objectives, operating cost and makespan, are simultaneously minimized using the Non-dominated Sorted Genetic Algorithm-II (NSGA-II). Usually, the multi-objective optimization problems are treated like single objective optimization problems by assigning some suitable weighting function to each objective function and summing up all the objective functions together (through which single criterion can be obtained). However, those weighting functions cannot be accurately calculated and hence, a non-domination optimization technique is used in this paper to find the non-dominated solutions (Pareto optimal sets). The solutions exist in the Pareto optimal set are usually non-dominated by any other solution that are existing in the same set and so they are also called as non-dominated solutions.

2.5 *The simple genetic algorithm*

The simple genetic algorithm is used to find the optimum design of the single objective optimization problem. The step-by-step procedure of the simple genetic algorithm used in this paper is shown in figure 1. The resources are randomly allocated to carry out the subtasks of various tasks in a sequential manner and the initial population of size 'N' is generated in the similar fashion. The operating cost and the makespan of each chromosome (design) in the initial population are calculated using Eqs. (1) and (5). If the stopping criterion is met, the algorithm is stopped and the best design (sequence) is presented. Otherwise, the algorithm continues until the stopping criterion is met. Usually, the maximum number of generations is set as the stopping criterion.

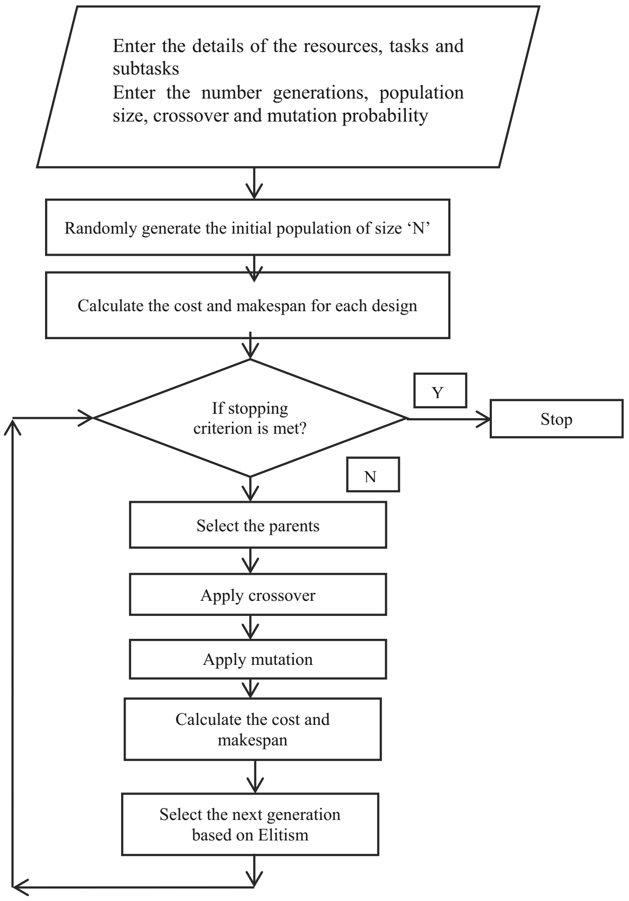
Enter the details of the resources, tasks and subtasks
Enter the number generations, population size, crossover and mutation probability
Randomly generate the initial population of size 'N'
Calculate the cost and makespan for each design
If stopping criterion is met?
Y
Stop
N
Select the parents
Apply crossover
Apply mutation
Calculate the cost and makespan
Select the next generation based on Elitism

Figure 1. Step-by-step procedure of simple genetic algorithm.

Genetic algorithm uses a set of operators called as selection, crossover and mutation to reproduce the offspring. The selection is the first operator used in GA to select the healthy chromosomes (designs) for being the parents. As the healthy chromosomes have very high probability for being selected as parents, the genetic algorithm generally concerns with the maximization problem. However, it can also be used for the minimization problems by changing the sign or taking the reciprocal of the value of the objective function. The roulette wheel selection process is used in simple genetic algorithm due to its simplicity and effective performance. The crossover is performed based on the user defined crossover probability between the selected parents to produce the new and likely the better designs. The uniform crossover operator, in which the changes are obtained in gene level, is used and it is discussed using an example given in table 1. The problem that consists of three tasks and three resources is considered in table 1. As each task has three subtasks, each chromosome has a length of nine genes (nine resources) and the crossover is performed randomly among 60% of the genes, if the user defined mixing ratio is 0.6.

After the crossover, mutation is performed to avoid the genetic algorithm to get stuck in any local optima. During the mutation process, the randomly chosen gene of a chromosome is replaced by a new gene that is arbitrarily selected within the user specified upper and lower limits. The uniform mutation, which is used in this paper, is explained with an example given in table 2.

When the entire parent population is replaced by the child (new) population, possibilities are there to lose some better chromosomes. Hence, the parent and child chromosomes are combined together and the best chromosomes of size 'N' are selected as the new population at the end of the reproduction process. This method, which is used to retain the better chromosomes during the next generations, is called as elitism. This iterative process continues until either the best design is found or the stopping criterion is satisfied.

2.6 *The non-dominated sorting genetic algorithm-II (NSGA-II)*

In single objective optimization problem, the most crucial (primary) design function is set as the objective function, while the other secondary functions are considered as

Table 1. Uniform crossover operator.

Before crossover									
Parent 1	2	1	3	2	1	2	3	1	2
Parent 1	3	3	2	1	3	3	1	1	3
After crossover									
Child 1	2	3	3	1	1	3	1	1	3
Child 2	3	1	2	2	3	2	3	1	2

Table 2. Uniform mutation operator.

Before mutation	2	3	3	1	1	3	1	1	3
After mutation	2	3	3	1	1	3	1	2	3

design constraints. For example, the makespan and operating cost are the two crucial design functions in the field of cloud computing. If a cloud user gives much priority to the makespan, it is treated as the objective function and the operating cost is set as the design constraint. Likely, if another user desires for minimizing the operating cost, the operating cost is considered as design function while the makespan is treated as one of the design functions. However, both these design functions are extremely important to the cloud users and hence they should be viewed as the multiple design objectives. Hence, a multi-objective optimization technique is to be used to yield the benefits on account of makespan as well as operating cost. While the single objective optimization technique leads to a single optimum solution, the NSGA-II offers a set of solutions known as non-domination solutions or Parteo optimal solutions. The solutions existing in the Pareto-optimal set cannot be compared with other solutions existing in the same set. However, the users can choose the most appropriate one among the set of solutions based on the weights (importance) they give for each objective function. As these weights are varying time to time based on the priority of the tasks, available resources and other factors, it is desired to find as many solutions as possible in the Pareto-optimal set.

The Non-dominated Sorting Genetic Algorithm-II (NSGA-II), which is recommended as an advanced genetic algorithm by Deb *et al* [30] is considered in this paper to find the Pareto optimal solutions for the multi-objective optimization problem. The step-by-step procedure of NSGA-II is shown in figure 2. The initial population is randomly generated and the designs are sorted into each front based on the principle of non-domination. Both objectives are maintained using a fitness assignment system, which favors non-dominated solutions. The front, which is ranked as one in the present population, is absolutely a non-dominant one. Similarly, the solutions existing in the second front are only dominated by the solutions existing in the first front and the assigning the ranks to the fronts happen so on. The crowding distance is also used together with the objective functions to show the closeness of a design with its neighbors and also to rank it. If ‘S’ is the number of designs existing in a front and ‘m’ is the number of objective functions, then the designs are sorted in the worst order of its objective values (f_m) and the sorted indices vector (I_m) is found (generally the size of ‘I’ and ‘S’ are same). A very large crowding distance is assigned to the designs existing in the boundary of the front as given in Eq. (7).

$$d_1^m = d_S^m = \infty \tag{7}$$

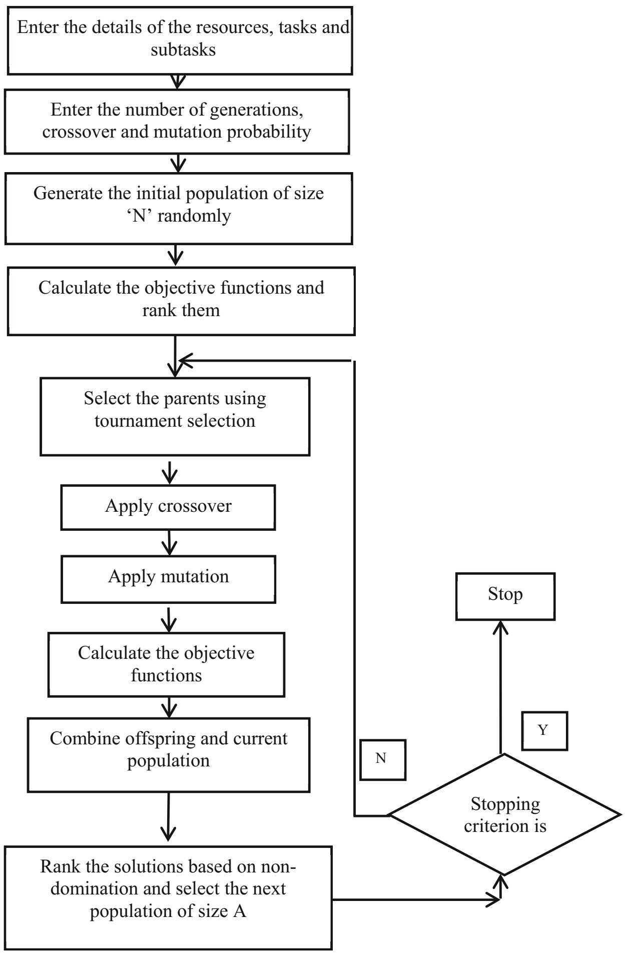
Enter the details of the resources, tasks and subtasks
Enter the number of generations, crossover and mutation probability
Generate the initial population of size ‘N’ randomly
Calculate the objective functions and rank them
Select the parents using tournament selection
Apply crossover
Apply mutation
Calculate the objective functions
Combine offspring and current population
Rank the solutions based on non-domination and select the next population of size A
Stopping criterion is
N
Y
Stop

Figure 2. Step-by-step procedure of NSGA-II.

Where, $d_1^m$and d_s^m are the crowding distances of initial and final designs for the objective function ‘*m*’. The crowding distance for the remaining designs is calculated using Eq. (8).

$$d_j^m = I_j^m + \frac{I_{j+1}^m - I_{j-1}^m}{f_{\max}^m - f_{\min}^m} \tag{8}$$

Where, I is the sorted indices vector and $f_{\min}^m$ and $f_{\max}^m$ are the minimum and maximum objective values for the objective function ‘m’. Based on the rank and crowding distance, the parents from the current population are selected using the tournament selection method. The design, which has a lower rank than another design, is selected. If both are having an equal rank, then the factor called as crowding distance is used to select the best one. The design having the higher crowding distance considered as the better one. The uniform crossover, uniform mutation and the elitism method that are used in the simple genetic algorithm are also used in NSGA-II also. The maximum number of generations specified by the user is considered as the stopping criterion for the Non-dominated Sorting Genetic Algorithm-II.

3. Numerical results and discussion

In cloud computing environment, the various heterogeneous resources located at different places are used by the cloud users based on pay per use method. The subtasks of various tasks are simultaneously accomplished by different resources under the constraint of executing the subtasks in a sequential manner. For example, if a task M has n subtasks, the subtask N_k can only be executed after the completion of the subtask N_{k-1}. The real world engineering usually has very large number of tasks and subtasks and hence, it is indispensable to the cloud users to optimize the task scheduling and allocation of resources to minimize the operating cost and time. Hence, ten tasks and fifteen resources cloud computing problem has been considered in this paper to minimize the operating cost and makespan through the single objective and multi-objective optimization methods. The details of tasks, subtasks and resources are provided in tables 3 and 4.

3.1 *Single objective optimization to minimize the operating cost and the makespan*

Initially, the resources are optimally allocated using the simple genetic algorithm to minimize either the operating cost or the makespan value in order to investigate the efficiency of the optimization procedure and to explore the influence of task scheduling and resource allocation on the operating cost and makespan. In addition to the operating cost or makespan, execution of subtasks in a sequential manner is also incorporated as one of the design constraints. The optimum control parameters of GA were found in trial and error method. In order to find the optimum control parameters, the GA has been run several times for the different configurations of its control parameters.

Table 4. Details of resources.

Resource number	Processing speed in GB/h	Cost for using the resource (in Rs.)
R_1	1.4	149.036468
R_2	0.85	90.486427
R_3	0.6	63.872772
R_4	0.74	78.7764188
R_5	0.64	68.1309568
R_6	0.2	21.290924
R_7	0.56	59.6145872
R_8	0.12	12.7745544
R_9	0.22	23.4200164
R_{10}	1.8	191.618316
R_{11}	0.64	68.1309568
R_{12}	0.2	21.290924
R_{13}	0.56	59.6145872
R_{14}	0.12	12.7745544
R_{15}	0.22	23.4200164

The crossover probability is varied from 0.65 to 0.9 with an increment of 0.5, whereas the mutation probability is varied from 0.02 to 0.09 with an increment of 0.01. Similarly, the population size is varied from 40 to 80 with an increment of 10. The number of generations was set as 500 for all those configurations and the maximum number of generations taken for the convergence of solutions is chosen as the ideal one. Based on the performance of GA with those different configurations of control parameters, the ideal values are identified and listed in table 5.

At first, the various heterogeneous resources listed in the table 4 are optimally allocated with an aim of reducing the operating cost. The maximum permissible makespan that is set as 750 hours is considered as the additional design constraint. The convergence of the solutions obtained for the cost minimization problem is shown in figure 3 and the optimum designs are listed in table 6 along with their corresponding cost and makespan.

The results show that the cost has been minimized to Rs. 13709 for which the makespan is 686.78 hours.

Table 3. Details of tasks and subtasks.

	ST_1	ST_2	ST_3	ST_4	ST_5	ST_6	ST_7	ST_8	ST_9	ST_{10}	ST_{11}	ST_{12}	ST_{13}	ST_{14}	ST_{15}
	Size of the subtasks in GB														
T_1	0.5	0.8	1.2	1.5	0.4	1.8	0.9	2.1	0.6	0.4	0.9	0.5	1.3	0.3	0.7
T_2	0.4	1.2	0.4	0.6	1.3	1.2	0.9	1.5	0.6	1.4	1.9	2.3	0.8	0.6	0.5
T_3	0.8	0.5	0.2	0.3	0.8	0.4	0.9	1.2	1.3	1.5	0.8	0.75	0.35	0.25	0.7
T_4	1.3	0.2	0.6	0.85	0.94	0.86	1.3	1.9	2.1	0.3	1.1	0.9	1.1	0.6	0.5
T_5	1.3	0.3	0.8	0.56	0.42	0.25	0.26	1.3	0.8	0.65	0.53	0.26	0.35	0.15	0.6
T_6	1.9	1.85	0.36	0.68	0.52	0.19	0.5	0.6	1.3	0.25	0.69	1.3	1.9	0.6	1.6
T_7	0.6	0.8	0.95	0.85	0.4	0.6	1.6	0.3	0.55	0.6	1.2	1.6	0.3	0.8	2.1
T_8	1.2	1.3	0.25	0.39	0.57	0.26	1.2	1.6	0.35	0.61	2.0	1.6	0.8	0.5	0.6
T_9	0.85	0.62	0.53	0.47	0.16	0.25	0.96	0.53	0.57	0.52	0.68	0.64	0.9	0.2	1.6
T_{10}	0.63	1.2	1.9	2.3	1.5	0.5	0.4	0.16	0.52	1.2	0.35	0.84	1.3	0.36	0.98

Table 5. GA control parameters.

Population size	60
Crossover probability	0.85
Mutation probability	0.06
Number of generations	350

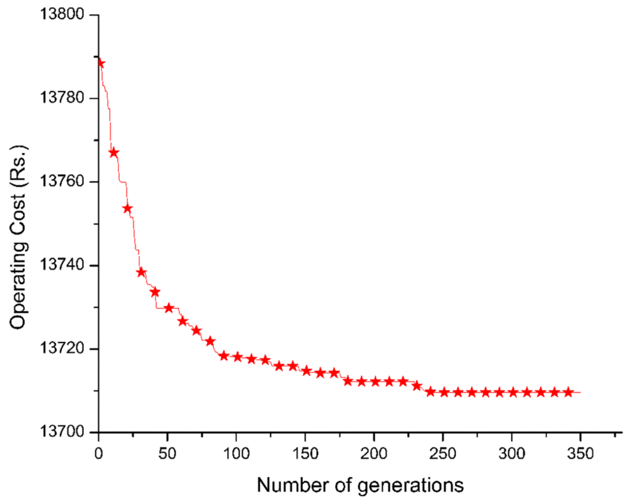

Figure 3. Convergence graph for the cost minimization problem.

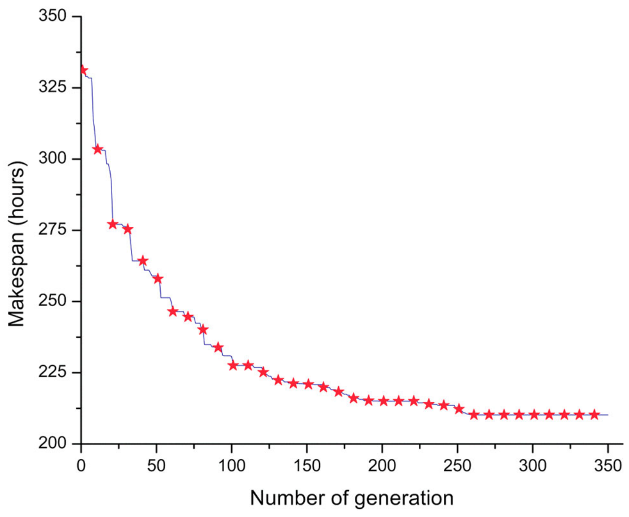

Figure 4. Convergence graph of makespan minimization problem.

Consequently, the makespan is minimized with an additional constraint of maximum limitation to the operating cost as Rs. 14,000. The convergence graph obtained for the makespan minimization problem is shown in figure 4 and the optimum allocation of the resources is listed in table 7 along with the corresponding makespan and cost.

3.2 *Multi-objective optimization to minimize the operating cost and makespan*

As the cloud users extremely prefer completing the tasks at minimum time as well as cost, the various heterogeneous resources are optimally allocated to minimize the operating cost and makespan simultaneously. Hence, a bi-objective optimization is carried out using the non-dominated sorting genetic algorithm-II to find the Pareto optimal designs. The completion of subtasks in a sequential manner is considered as the design constraint. The Pareto optimal designs, which are obtained using NSGA-II, are shown in figure 5.

Figure 5 shows that a wide range of solutions are obtained in the Pareto optimal set and hence, the cloud user is given a wide range of solutions. These Pareto optimal designs cannot be compared with one another unless some additional information is provided. The cloud users can choose the most appropriate design (allocation of resources) from this Pareto optimal set based on the other influencing factors such as priority of tasks and availability of resources.

If G is the number of generations, P is the size of population, M is the number of objective functions, c(m) and c(c) are the complexities of mutation and crossover then the computational complexity of the single objective optimization (SOO) and multi-objective optimization (MOO)

Table 6. Optimum allocation of resources for the cost minimization problem.

Task no.	Allocation of resources	Cost (Rs.)	Makespan (h)
T_1	2-3-14-14-9-14-12-8-6-7-14-4-14-11-15	13709.6	686.78
T_2	3-9-11-7-14-14-4-12-1-14-8-6-14-15-2		
T_3	13-4-10-2-14-12-3-12-12-6-15-8-11-5-9		
T_4	3-4-7-15-8-12-8-8-8-5-9-6-8-2-11		
T_5	8-3-7-6-11-2-15-5-8-13-14-1-7-10-9		
T_6	9-14-7-4-11-1-5-13-6-15-14-12-8-3-14		
T_7	1-2-7-9-5-11-12-4-3-7-6-15-8-13-14		
T_8	8-15-6-13-2-3-8-14-5-8-8-8-11-12-7		
T_9	14-11-6-7-10-4-14-1-2-15-13-14-9-3-14		
T_{10}	6-14-15-14-8-11-14-7-2-12-3-9-14-1-4		

Table 7. Optimum allocation of resources for makespan minimization problem.

Task no.	Allocation of resources	Cost (Rs.)	Makespan (h)
T_1	6-1-3-4-14-5-7-11-15-12-1-9-10-2-13	13968.9	210.23
T_2	8-13-12-15-2-10-11-3-9-4-1-5-10-14-7		
T_3	1-2-6-12-1-9-3-1-10-1-5-7-15-14-11		
T_4	5-14-1-7-1-6-3-10-1-12-13-9-4-2-11		
T_5	1-10-4-3-7-8-9-1-11-1-2-15-13-6-5		
T_6	10-10-15-2-9-14-7-5-10-3-10-10-10-13-11		
T_7	4-10-2-13-9-11-10-15-5-7-1-10-12-3-1		
T_8	10-10-8-14-3-15-10-5-6-11-10-7-1-13-2		
T_9	11-2-15-6-4-12-1-13-10-3-5-7-5-9-1		
T_{10}	1-10-10-4-10-7-15-6-9-13-12-3-2-11-5		

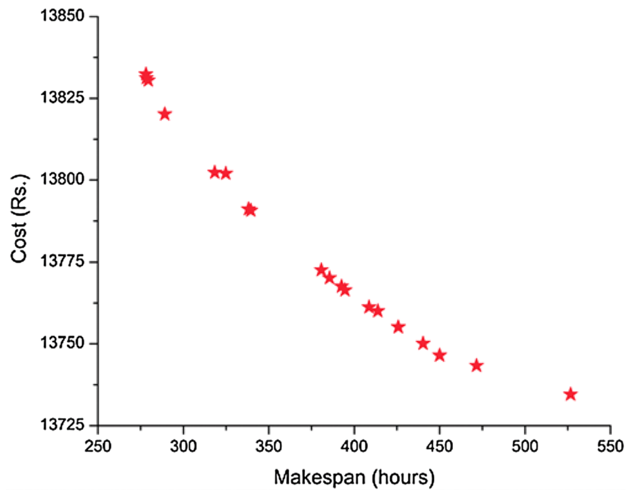

Figure 5. Pareto optimal designs.

techniques were analysed using Big-O method [31] as given in Eq. (9).

$$\begin{aligned}&\text{Computational Complexity of SOO} = \\ &O(G \times P \times [c(m) + c(c)]) \\ &\text{Computational Complexity of MOO} = \\ &\quad O\left(G \times P^2 \times M \times [c(m) + c(c)]\right)\end{aligned} \tag{9}$$

The crossover and mutation probabilities were kept constant in both single objective and multi objective optimization techniques and therefore these factors are ignored in the calculation of computational complexity of the optimization techniques. The optimum values of G and P were identified based on trial and error method as discussed in section 3.2 for both SOO and MOO. The identified optimum values of G and P are 350 and 60 in the case of SOO and 400 and 60 in the case of MOO, respectivley. As per Eq. (9), the computational complexity of MOO is much higher than that of SOO. However, the cloud users are benefited with a set of solutions rather than the single solution by which they can choose the best one based on the other influencing factors.

4. Conclusions

The main objective of this paper is to minimize the cost as well as the makespan of the cloud service by optimally varying the task scheduling and resource allocation. A cloud service consisting of ten tasks and fifteen resources has been considered as a case study. Initially, the task scheduling and resource allocation were optimized separately using the simple genetic algorithm to minimize the cost and makespan of the cloud service. Subsequently, the multi-objective optimization has been performed to obtain the non-dominated solutions for the multiple objectives makespan and cost. The numerical results show that the simple genetic algorithm proposed in this paper effectively minimizes the makespan or the operating cost. However, aiming to minimize the makespan critically increases the operating cost and vice versa. So, the work is extended as a multi-objective optimization problem by considering the makespan and operating cost are the dual objectives. Unlike the other multi-objective optimization techniques, NSGA - II used in this paper offers a wide range of solutions. Hence, the users can select the most appropriate one based on the time varying considerations like the priority of tasks and availability of the cloud resources. However, the users should have a high level of skill and require additional information to identify the most suitable (optimum) one from the set of Pareto-Optimal solutions.

References

[1] Smanchat S and Viriyapant K 2015 Taxonomies of workflow scheduling problem and techniques in the cloud. *Future Gener. Comput. Syst.* 52: 1–12

[2] Li J, Su S, Cheng X, Huang Q and Zhang Z 2011 Cost-conscious scheduling for large graph processing in the cloud. In: *Proceedings of IEEE 13th International Conference on High Performance Computing and Communications HPCC,* pp. 808–813

[3] Su S, Li J, Huang Q, Huang X, Shuang K and Wang J 2013 Cost-efficient task scheduling for executing large programs in the cloud. *Parallel Comput.* 39(4): 177–188

[4] Fang W, Liang X, Li S, Chiaraviglio L and Xiong N 2013 VMPlanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Comput. Netw.* 57(1): 179–196

[5] Guo Z, Duan Z, Xu Y and Chao H J 2014 JET: electricity cost-aware dynamic workload management in geographically distributed data centers. *Comput. Commun.* 50: 162–174

[6] Malawski M, Gideon J, Deelman E and Jarek N 2015 Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. *Future Gener. Comput. Syst.* 48: 1–18

[7] Calheiros R N and Buyya R 2012 Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds. In: *Proceedings of IEEE International Conference on Web Information Systems Engineering*. Springer, Berlin, pp. 171–184

[8] Li J, Su S, Cheng X, Song M, Ma L and Wang J 2015 Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads. *Parallel Comput.* 44: 1–17

[9] Lin C and Lu S 2011 Scheduling scientific workflows elastically for cloud computing. In: *Proceedings of IEEE International Conference on Cloud Computing,* pp. 746–747

[10] Mezmaz M, Melab N, Kessaci Y, Lee Y C, Talbi E G, Zomaya A Y and Tuyttens D 2011 A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *J. Parallel Distrib. Comput.* 71(11): 1497–1508

[11] Selvarani S and Sadhasivam G S 2010 Improved cost-based algorithm for task scheduling in cloud computing. In: *Proceedings of IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–5

[12] Chaisiri S, Lee B S and Niyato D 2012 Optimization of resource provisioning cost in cloud computing. *IEEE Trans. Serv. Comput.* 5(2): 164–177

[13] Thomas A, Krishnalal G and Raj V J 2015 Credit based scheduling algorithm in cloud computing environment. *Procedia Comput. Sci.* 46: 913–920

[14] Zhou A, Sun Q, Sun L, Li J and Yang F 2015 Maximizing the profits of cloud service providers via dynamic virtual resource renting approach. *EURASIP J. Wirel. Commun. Netw.* 1: 1–12

[15] Mustafa S, Nazir B, Hayat A and Madani S A 2015 Resource management in cloud computing: Taxonomy prospects and challenges. *Comput. Electr. Eng.* 47: 186–203

[16] Krishna P V 2013 Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Appl. Soft Comput.* 13(5): 2292–2303

[17] Casas I, Taheri J, Ranjan R, Wang L and Zomaya A Y 2017 A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Future Gener. Comput. Syst.* 74:168–178

[18] Juarez F, Ejarque J and Badia R M 2018 Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Gener. Comput. Syst.* 78: 257–271

[19] Maheshwari K, Jung E S, Meng J, Morozov V, Vishwanath V and Kettimuthu R 2016 Workflow performance improvement using model-based scheduling over multiple clusters and clouds. *Future Gener. Comput. Syst.* 54: 206–218

[20] Arabnejad H, Barbosa J G and Prodan R 2016 Low-time complexity budget deadline constrained workflow scheduling on heterogeneous resources. *Future Gener. Comput. Syst.* 55: 29–40

[21] Moschakis I A and Karatza H D 2015 A meta-heuristic optimization approach to the scheduling of Bag-of-Tasks applications on heterogeneous clouds with multi-level arrivals and critical jobs. *Simul. Model. Pract. Theory* 57: 1–25

[22] Nicholas P E, Padmanaban K P and Vasudevan D 2014 Buckling optimization of laminated composite plate with elliptical cutout using ANN and GA. *Struct. Eng. Mech.* 52(4): 815–827

[23] Xu Y, Li K, Hu J and Li K. 2014 A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Inf. Sci.* 270: 255–287

[24] Tao F, Feng Y, Zhang L and Liao T W. 2014 CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Appl. Soft Comput.* 19: 264–279

[25] Zheng L, Lu Y, Guo M. Guo S and Xu C Z 2014 Architecture-based design and optimization of genetic algorithms on multi-and many-core systems. *Future Gener. Comput. Syst.* 38: 75–91

[26] Kolodziej J and Khan S U 2012 Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. *Inf. Sci.* 214: 1–19

[27] Dasgupta K, Mandal B, Dutta P, Mandal J K and Dam S 2013 A genetic algorithm GA based load balancing strategy for cloud computing. *Procedia Technol.* 10:340-347

[28] Yu J, Buyya R and Tham C K 2005 Cost-based scheduling of scientific workflow applications on utility grids. In: *Proceedings of IEEE First International Conference on In e-Science and Grid Computing*, pp. 8–147

[29] Kara N, Soualhia M, Belqasmi F, Azar C and Glitho R 2014 Genetic-based algorithms for resource management in virtualized IVR applications. *J. Cloud Comput.* 3(1): 1–18

[30] Deb K, Pratap A, Agarwal S and Meyarivan T A M T 2002 A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6(2): 182–197

[31] Jensen, M T 2003 Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. *IEEE Trans. Evol. Comput.* 7(5): 503–515