Epsilon-fuzzy dominance sort-based composite discrete artificial bee colony optimisation for multi-objective cloud task scheduling problem

B. Gomathi*
Department of Information Technology, Hindusthan College of Engineering and Technology, Coimbatore, 641 032, India
Email: gomathib@hindusthan.net
*Corresponding author

Karthikeyan Krishnasamy
Department of Information Technology, Karpagam College of Engineering, Coimbatore, 641 032, India
Email: karthiaish1966@gmail.com

B. Saravana Balaji
Department of Computer Science and Engineering, S.Veerasamy Chettiar College of Engineering and Technology, Puliyangudi, 627 855, India
Email: saravanabalaji.b@gmail.com

Abstract: Cloud computing environment provides on-demand virtualised resources for cloud application. The scheduling of tasks in cloud application is a well-known NP-hard problem. The task scheduling problem is more complicated while satisfying multiple objectives, which are conflict in nature. In this paper, Epsilon-fuzzy dominance based composite discrete artificial bee colony (EDCABC) approach is used to generate Pareto optimal solutions for multi-objective task scheduling problem in cloud. Three conflicting objectives, such as makespan, execution cost and resource utilisation, are considered for task scheduling problem. The Epsilon-fuzzy dominance sort approach is used to choose the best solutions from the Pareto optimal solution set in the multi-objective domain. EDCABC with composite mutation strategies and fast local search method are used to enrich the local searching behaviours which help to avoid the premature convergence. The performance and efficiency of the proposed algorithm is compared with NSGA-II and MOPSO algorithms. The simulation results express that proposed EDCABC algorithm substantially minimises the makespan, execution cost and ensures the proper resource utilisation when compare to specified existing algorithm.

Keywords: task scheduling; discrete artificial bee colony; cloud computing; makespan; execution cost; fuzzy dominance; load balancing.
1 Introduction

Cloud computing is an internet-based computing service which provides on-demand virtual resources to users (Boss et al., 2007; Dikaiakos et al., 2009). To dispatch the application tasks to cost effective virtual resources, efficient task scheduling algorithm is required to improve the resource utilisation in cloud environment. The simultaneous allocation of heterogeneous resources to tasks in cloud environment makes task scheduling problem as a NP-hard problem. Many heuristics algorithms are proposed to solve task scheduling problem in order to improve the service quality in cloud environment. Improved genetic algorithm (GA) (Kumar and Verma, 2012) is proposed to schedule the tasks in cloud environment and considered to minimise the makespan and improve the resource the utilisation. Priority-based self-adaptive learning particle swarm optimisation (PSO) (Zuo et al., 2014) is used to schedule the tasks in cloud environment by adaptively selecting velocity updating strategies. This algorithm guarantees the user level QoS and improve the credibility and economic benefit of cloud provider. In commercial cloud environment, tasks should complete its execution on cloud resources within budget before deadline. Hence, the proposed optimal EDCABC task scheduling algorithm is used to propose the trade-off approach for maximising the resource utilisation and minimising the makespan as well as execution cost.

In the previous multi-objective works (Ramezani et al., 2013), the classical optimisation methods have been used to combine multi-objectives into a single objective which produced only one solution at a time. When several solutions were required, classical method has to be run several times which takes larger execution time and led to different optimal solution at each execution. To overcome these problems, Epsilon-fuzzy dominance-based composite discrete artificial bee colony (EDCABC) algorithm is proposed to generate Pareto optimal solutions for multi-objective task scheduling problem in cloud environment. In proposed Epsilon-fuzzy dominance Pareto optimal solution concept, solutions in the population is assigned the perimeter value ['equation (9)'] which helps to measure how one solution dominates other solutions and
helps to produce best compromised solution which considers different objectives simultaneously and saves the execution time.

The proposed algorithm is used to consider three conflicting objectives such as makespan, execution cost and utilisation of cloud resources and able to provide faster convergence towards the Pareto front in short duration while preserving solution with good diversity to produce better efficiency. The cloudsim toolkit is used to simulate the task scheduling problem in cloud environment and used to compare the prominence of the proposed approach with well known task scheduling algorithms like NSGA-II, MOPSO in cloud environment.

In the rest of the paper, the related work of the task scheduling problem is mentioned in section 2. In Section 3, the problem formulation is specified. Section 4 presents multi-objective optimisation approach. Section 5 gives the information about EDCABC algorithm. Section 6 shows the simulation strategy and result analysis respectively. Finally, Section 7 gives the conclusion.

2 Related work

Nowadays, the researchers are focused to define multi-objectives for task scheduling problem for generating effective schedule. When task scheduling problems are solved, scheduling criteria such as makespan, reliability, execution cost, etc. are considered. The matching and scheduling algorithms (Dogan and Ozguner, 2002) have been used to generate efficient schedule to reduce the effect of failure of machine as well as execution time of the application. In Qin and Jiang (2005), a machine’s failure during an idle period has not been considered in this reliability model since failure machine was fixed by spare unit in the idle period itself as well as the task is allocated to processor which guaranteed the deadline of task and reduces failure probability of application. An optimal scheduling algorithm (Dongarra et al., 2007) for independent unitary tasks is used to maximise the reliability and minimising the makespan with help of Pareto optimal solution set. The bi-objective hybrid GA (Mezmaz et al., 2011) is used to schedule the precedence constrained parallel applications in cloud environment in order to minimise energy consumption and makespan. Dynamic scaling voltage (DSV) technique has been used by processors to adjust the voltage dynamically to reduce energy consumption. This algorithm got benefit from exploration capability of GA and intensification power of energy consumption scheduling algorithm to improve the solution quality.

The weighted sum (Chitraa et al., 2001) based GA and evolutionary programming (EP) has been hybridised using simple neighbourhood search algorithm (SNS) to improve the convergence speed. Dependent tasks were considered for minimising the two conflicting objectives such as makespan and failure probability of application. The vectorised ordinal optimisation (Zhang et al., 2014) algorithm has been used to schedule the large number of tasks in virtual clusters in cloud computing platform. This algorithm deals with minimisation of resource cost, makespan and preserving fault tolerance at the same time. When cloud providers (Zuo et al., 2014) allocate infrastructure to user’s tasks, self-adaptive learning particle swarm optimisation (SLPSO)-based scheduling approach is used to maximise the profit of cloud providers while guaranteeing QoS. To improve the diversity and robustness in the search space, SLPSO uses different updating strategies to adaptively update the velocity of each particle.
Tsai et al. (2013) applied improved differential evolution algorithm (IDEA), which trade-offs between total cost and makespan in cloud computing environment using the non-dominated sorting technique. Awad et al. (2015) proposed multi-objective load balancing mutation particle swarm optimisation (MLBMPSO) to schedule tasks in cloud environment and considered to minimise round trip time and total cost. Reliability can be achieved in system by getting task failure to allocate and reschedule with available resource based on load of virtual machine (VM).

The above aforementioned multi-objective algorithms were used to generate the Pareto optimal solutions in the multi-objective problem. All the non-dominated solutions in Pareto set were considered as equal and not able to measure how one solution is dominated by other solution. Hence, the proposed Epsilon-fuzzy dominance-based EDCABC algorithm is used to measure the relative fitness of non-dominated solutions in Pareto optimal set which helps to minimise both makespan and execution cost as well as to provide the better and quicker convergence than other algorithms like NSGA-II and MOPSO especially when the number of objectives are larger.

3 Problem formulation

In order to conform to the parallel and distributed cloud environment, the dynamic batching mode is taken, where tasks are collected in a set that is analysed for mapping instead of mapping tasks into resources as they arrive. According to the collected information such as real status of resources and task details, more reasonable scheduling strategy can be designed in cloud environment. In this paper, multi-objective EDCABC scheduling algorithm using Epsilon-fuzzy-based sorting mechanism is proposed to find the optimal schedule by minimising execution time and cost of the incoming tasks as well as balance the load across resources.

To formulate the multi-objective task scheduling problem, the set of $n$ mutually independent tasks are represented as $T_j$, where $j = \{0, 1, ..., n-1\}$ and set of $m$ heterogeneous resources are represented as $R_i$, where $i = \{0, 1, ..., m-1\}$. Assume that the execution time $P_{ij}$ for task $j$ on resource $i$ is known. Furthermore, tasks are considered as non-preemptive. The following constraints give the assurance that only one task can be executed by processing resource at a time (constraint 2) and each task is allocated to exactly one processing resource(constraint 1). The permutation matrix entry $X_{i,j}$ (as shown in Table 1) is defined as:

$$X_{i,j} = \begin{cases} 
1, & \text{if task } j \text{ is assigned to resource } i \\
0, & \text{otherwise}
\end{cases}$$

(1)

$$\sum_{j=1}^{n} X_{i,j} = 1, \quad 1 \leq j \leq n$$

(2)

The cloud task scheduling problem can be formulated to minimise three conflicting objectives such as makespan MS (Gomathi and Karthikeyan, 2013), total execution cost CS and load balancing index $\beta$ as shown below:

Minimise $MS = \max_{1 \leq j \leq n} \sum_{i=1}^{m} R_{ij} * X_{i,j}$

(3)
Minimise $CS = \sum_{i=1}^{m} MS_i \ast \gamma_i$ \hspace{1cm} (4)

First objective is makespan which represents the largest finishing time of tasks among all cloud resources. Here, the total execution time of tasks allocated to each resource is calculated and finding the maximum execution time among cloud resources is considered as makespan. Let $\gamma_i$ be the cost per unit time of resource usage $R_i$ and second objective is the execution cost which represents the total execution cost of given task set in allocated resources.

Due to the heterogeneous nature of resources in cloud, processing capability varies from resource to resource. When allocating processing resources to task set, most of the tasks are allocated to better processing capability resources which will get more workload than other idle resources. To balance the load across different processing resources, third objective such as load balancing index ($\beta$) is used. $\beta$ is computed to gauge the deviations of load on processing resources as follows (Kruekaew and Kimpan, 2014).

Minimise $\beta = \sqrt{\sum_{i=1}^{m} L_i - \bar{L}} \hspace{1cm} (5)$

where $L_i$ is the load of the resource $i$ and $\bar{L}$ is the average load of all resources. The smaller $\beta$ shows the better load balancing in cloud environment. To augment the performance of task scheduling in cloud, the load will be balanced across resources and maximise their utilisation while minimising makespan and execution cost in cloud environment.

### 4 Epsilon-fuzzy dominance in multi-objective optimisation

The concurrent optimisation of two or more objectives in optimisation problem is called as multi-objective optimisation problem (MOP) (Deb, 2001; Deb et al., 2002). A MOP can be defined as follows:

Minimise $S(x) = (S_1(x), S_2(x), ..., T(x)) \hspace{1cm} (6)$

where $T$ is the number of objectives in the problem, $S_i(x)$ is the $i^{th}$ objective function and $X$ is the solution. Many traditional methods like weighted method, distance method and min-max formulation method are available for optimisation of multiple objectives, but these methods combine multiple objectives into single objective which may provide single incorrect solution as well as need prior information about the optimum before optimisation. In this situation, the rank-based non-dominance sorting algorithm was used to generate Pareto dominance solution for multi-objective optimisation.

Let us take the Pareto dominance relation which shows that a small degradation in one or several objectives is usually permitted if a remarkable improvement in the other objectives is able to be achieved. Let us take two solution vectors $x_1$ and $x_2$, then solution $x_1$ is said to dominate $x_2$ (also written as $x_1 < x_2$) iff following two conditions hold:

\[ \forall i \in \{1, 2, ..., T\}: S_i(X_1) \leq S_i(X_2) \]

\[ \exists j \in \{1, 2, ..., T\}: S_j(X_1) < S_j(X_2) \]
A solution is called as non-dominated or Pareto optimal solution if there is no feasible solution which improves one objective without reducing another objective. The collection of Pareto optimal solutions is called Pareto set and the boundary of mapping the Pareto solutions in solution space is called Pareto front. Since all the non-dominated solutions in Pareto set must be considered as equal, rank-based method is not able to measure how one solution extends to dominate other solution. It is provided by the new dominance relation called fuzzy dominance (Koduru et al., 2007) method.

Let us assume to minimise $T$ number of objectives functions $S_i(x)$, $i = 1, \ldots, T$ in MOP. The solution set is denoted as $\Psi \subset \mathbb{R}^n$, where $n$ is the dimensionality.

### 4.1 Fuzzy dominance by a solution

Solution $u \in \Psi$ is said to fuzzy dominate solution $v \in \Psi$ if and only if $\forall i \in \{1, 2, \ldots, T\}$, $u \succ^i v$ holds. This relationship can be denoted as $u \succ^S v$. The degree of fuzzy dominance can be defined by invoking the concept of fuzzy intersection and using t-norm, and is computed as:

$$
\mu_{\text{dom}}^S(u \succ^S v) = \bigcap_{i=1}^T \mu_{\text{dom}}^i(u \succ^i v)
$$

(7)

In the prior fuzzy dominance work (Koduru et al., 2007), the membership functions $\mu_{\text{dom}}^i(.)$ used to compute the fuzzy dominance were defined to be zero for negative arguments. Hence, if $S_i(u) > S_i(v)$, the degree of fuzzy dominance $u \succ^i v$ was zero mandatorily. Hence, non-dominated solutions may not be assigned zero values mandatorily with help of $\varepsilon$. The membership functions used are trapezoidal, yielding non-zero values whenever their arguments are to the right of threshold $\varepsilon$. Mathematically, the membership function $(u \succ^i v)$ is defined as:

$$
\mu^i_{\text{dom}}(\Delta S_i) = \begin{cases} 
0 & \Delta S_i \leq -\varepsilon \\
\frac{\Delta S_i}{\Delta_i} & -\varepsilon < \Delta S_i < \Delta_i - \varepsilon \\
1 & \Delta S_i \geq \Delta_i - \varepsilon 
\end{cases}
$$

(8)

where $\Delta S_i = S_i(v) - S_i(u)$. If many solutions are having similar $\varepsilon$-fuzzy dominance value, then the diversity Fitness function which is equal to the perimeter of the largest $M$ dimensional hypercube in the objective space, can be used (Deb, 2001). The value of perimeter $I(y)$ for any solution $y$ is given by:

$$
I(y) = \sum_{i=1}^T (S_i(x) - S_i(z))/(\max(S_i) - \min(S_i))
$$

(9)

where $x$ and $z$ are the adjacent solutions to $y$ while merging the population in ascending order according to the $i^{th}$ objective, $S_i$. Boundary solutions are assigned $\infty$ values. When comparing multiple solutions with same $\varepsilon$-fuzzy dominance values, the priority is given to solutions with higher values of $I(y)$.

### 4.2 Fuzzy dominance in a population

Given a population of solutions $P \subset \Psi$, a solution $v \in P$ is said to be fuzzy dominated in $P$ iff it is fuzzy dominated by any other solution $u \in P$. In this case, the degree of fuzzy
dominance can be computed by performing a union operation over every possible \( \mu^{\text{dom}}(u \succ S v) \), carried out using t-co norms as:

\[
\mu^{\text{dom}}(P \succ S v) = \bigcup_{u \in P} \mu^{\text{dom}}(u \succ S v)
\]

In this manner, each solution is assigned a single measure to reflect the amount it dominates the others in a population. Better solutions within the set are assigned lower fuzzy dominance values. The global best solution can be selected by this sorting method after completion of each iteration.

5 Multi-objective EDCABC algorithm

An artificial bee colony (ABC) algorithm is proposed by Karaboga and Basturk (2008) to influence the behaviour of honey bee swarm intelligence and optimise multi-variable and multi-modal continuous functions. Since it has lesser control parameters and ease of implementation, researchers attempt to extend the ABC algorithm to solve MOP and showed that the performance of the ABC algorithm is competitive to other swarm intelligent algorithms. In this paper, EDCABC algorithm is proposed to solve the multi-objective task scheduling problem in cloud environment. The main features of the proposed CDABC are as follows:

a. Composite discrete ABC is proposed to solve task scheduling problem in cloud.
b. To balance exploitation and exploration ability, a self-adaptive neighbourhood structure strategy is applied.
c. Composite mutation strategy and fast local search algorithm are reused to magnify the exploitation performance.
d. Non-dominated solutions are maintained with help of Pareto archive set. The proposed EDCABC algorithm is explained in the following section.

5.1 Artificial bee colony

The ABC algorithm is an evolutionary algorithm that models the foraging behaviour of honey bee colony. The bee colony consists of three categories of bees: employed bees, onlooker bees and scout bees. Half of the colony comprises of employed bees, and the other half comprises onlooker bees. In the ABC algorithm, each cycle of the search consists of three steps: employed bees move towards their the food sources and then sharing the excellence of food sources with onlookers to choose food sources and decides scout bees (abandoned employee bee) and then randomly move them towards the possible food sources.

In ABC algorithm, number of food sources in population is determined by number of employee bees or onlooker bees. Initially, ABC produces SN number of n dimensions of population randomly. Let \( P_i = \{P_{i1}, P_{i2}, \ldots, P_{in}\} \) represents the \( i^{th} \) food source in the colony. The number of food source is equal to number of employee bee or onlooker bee. Each food source is generated as follows:
where the random number \( r \) is generated in the range \([0, 1]\). \( P_{\text{min}} \) and \( P_{\text{max}} \) are lower and upper bounds for the dimension \( j \) respectively. To find new neighbourhood food source by employee bee using its present position, the following equation is used:

\[
\theta_{i,j} = \rho_{i,j} + \phi_{i,j} (\rho_{i,j} - \rho_{k,j})
\]

(12)

where the random number \( \phi_{i,j} \) is generated in the range \([-1, 1]\). The employee bee compares the fitness of new food solution with old one and memories the better solution.

An onlooker bee collects the information of the food sources from all employed bees and selects a food source depending on the probability value associated with that food source, \( P_i \) using the following equation:

\[
P_i = \frac{\text{Fitness}_i}{\sum_{i=1}^{SN} \text{Fitness}_i}
\]

(13)

where \( \text{Fitness}_i \) is the fitness value of the solution \( i \) evaluated by its employed bee, which is proportion to nectar quantity of the food source in the position \( i \). With help of chosen food source, onlooker bee produces a new source by ‘equation (12)’.

**Figure 1** Flowchart for ABC algorithm

Generate initial population of solution and evaluate them

Find new solution for employed bee and evaluate according to its strategies

According to greedy search, generate new population for onlooker bee and evaluate them

Find abandoned solution for scout bee and replace them with new randomly generated solution

Memorise best food source found so far

Is termination criteria met?

Output the best food source
The onlooker bee will evaluate new food solution and compare it with primary solution. Food source with better nectar amount will be accepted. When any food source cannot be improved for determined time, then employee bee will abandon that food source and it becomes scout. The scout begins to generate new food source by using ‘equation (11)’, then the next iteration will be started. The entire process will be repeated until it reaches the termination condition. The total flowchart of ABC algorithm is shown in Figure 1.

Like other stochastic optimisation algorithms, such as GA, PSO and SA, as the dimensionality of the search space increases, ABC algorithm provides poor convergence behaviour. Hence, discrete artificial bee colony with composite mutation strategies and fast local search techniques are proposed for solving task scheduling problem in cloud computing. The composite mutation strategies are used to enhance the searching ability in order to prevent sticking in local optimum. Fast local search is proposed to magnify the local search performance of EDCABC algorithm.

5.2 Epsilon-fuzzy dominance-based composite discrete artificial bee colony

To solve the task scheduling problem in cloud environment, continuous ABC algorithm is modified as composite discrete ABC with Epsilon-fuzzy dominance-based sort method to generate Pareto optimal set in multi objective optimisation problem as shown below:

1. Initialise the population size SN, trail limit L. Assume that the number of employed bees or onlooker bees is equal to the number of individuals in the population.
2. Randomly initialise individuals in the population  \( P = \{ X_1, X_2, \dots, X_{SN} \} \).
3. Evaluate individuals in the population based on the objective functions. Sort the individuals in the population based on Epsilon-fuzzy dominance method and perimeter. Initialise external Pareto archive set with it.
4. Employed bee phase
   a. Produce a new food source for \( i^{th} \) employed bee with help of composite mutation strategies and find best among them.
   b. If the selected food source is better than current food source, then it will become current food source.
5. Onlooker bee phase:
   Update the winning probability value for each individual. For each onlooker bee phase:
   a. Choose a food source in the population based on probability value of it. Generate new food source for each onlooker bee by using composite mutation strategies. If best food source among them is better than current food source, then it will become current food source.
   b. Update external archive set as given in the employed bee phase.
6 Scout phase:
   a If a solution in the population does not improve for the predetermined number of
      trail limit $L$, then solution is abandoned by employed bee and employee bee
      becomes scout.
   b The scout produces new food source by performing several insertion operations
      on the best solution in the population and initialise limit to zero.

7 Conduct fast local search on the best solution in the population to improve
   the solution.

8 If termination condition is not reached, then go to Step 4. Otherwise, produce
   solution in the external archive set as output.

The flowchart of the proposed EDCABC algorithm is shown in Figure 2. Apart from
basic components of ABC, the proposed EDCABC algorithm contains the following
additional components:

**Figure 2** Flowchart of the proposed EDCABC algorithm
Table 1 Individual representation with resource-task mapping

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

5.2.1 Solution representation

To optimise the task scheduling problem using EDCABC in cloud, each solution needs to map with one of the individual in the population. Each individual is used to represent task-resource mapping, which maps each task into suitable resource in cloud. Task-resource mapping produces \( m \times n \) matrix, where \( m \) is the number of available resources and \( n \) is the number of tasks as shown in Table 1. Here, each task is allocated to only one resource. The resource allocation is done with help of the following equation, \( R_i = T_j \mod m \), where \( R_i \) is the resource number to which the task \( T_j \) (task number) is assigned and \( m \) is the number of resources. This equation helps to balance load across the cloud resource and improves the resource utilisation. If the task \( T_j \) is assigned to resource \( R_i \), then \( (R_i, T_j) = 1 \) in the table, otherwise, \( (R_i, T_j) = 0 \).

5.2.2 Composite mutation strategies

Single mutation strategy in various evolutionary algorithms can quickly get stuck into the local optimal solution in case of diverse instance of task scheduling problem. To solve the multi-objective task scheduling problem using EDCABC, multiple mutation strategies are needed to avoid getting stuck into local optimal solution and to search the solution space efficiently. There are four neighbourhoods, i.e., inverse, adjacent interchange, insert and swap (Nowicki and Smutnicki, 1996), are used as the various mutation operations to escape from different local optimal solution and enable the proposed algorithm to solve the task scheduling. The following four neighbourhood operations are used in the proposed algorithm:

- Swamp mutation: Randomly choose two different positions from a job permutation and swap them. In Figure 3, tasks 5 and 8 are selected for swapping.
- Inverse mutation: Inverse the subsequence between the two different random positions of a job permutation. In Figure 3, task sequences 5, 6, 2, 8 are reversed.
- Insert mutation: Randomly choose two different positions from a job permutation and insert the back one before the front. In Figure 3, tasks 4 and 8 are selected and task 8 is inserted before task 4.
- Adjacent exchange mutation: Randomly choose two adjacent positions from a job permutation and swap them. Adjacent task positions 6 and 7 are selected and exchanged tasks in those positions.

Figure 3 illustrates the above four neighbours operation.
In the composite DABC algorithm, several mutation strategies are composited at each generation to create new solution in search space. To enhance neighbourhood structure and diversity of the population, above approaches are used to generate the following neighbourhood food sources for individuals in the population:

1. Use one-insert operator in a solution.
2. Use one-swap operator in a solution.
3. Use one-inverse operator in a solution.
4. Use one-adjacent exchange operator in a solution.
5. Use two-swap operator in a solution.
6. Use two-insert operator in a solution.

A new food source is created with help of each mutation strategy at every iteration and best food source enters into next generation when it is better than current food source.

5.2.3 Enhancement of elite archive

Elite archive method (Coello et al., 2004) in MOP is used to store best non-dominated solutions in elite archive set. Initially, SN numbers of solutions are stored in archive set. In each iteration, solutions from current generation and solutions from archive of previous generations are combined and these 2*SN solutions are sorted with help of Epsilon-fuzzy dominance values. If multiple solutions are having same Epsilon-fuzzy dominance values, then perimeter \( I(.) \) value is calculated using equation (9) and solution with largest perimeter value is preferred. Hence, the best SN solutions are taken to update archive set from 2*SN solutions which are sorted based on Epsilon-fuzzy dominance and perimeter.

5.2.4 Fast local search

To enrich exploitation ability of the proposed algorithm, fast local search algorithm (Li and Yin, 2013) is incorporated. Fast local search is used to enhance the local search ability to produce better solutions from neighbourhood of a solution. This algorithm uses new individual enhancement scheme which uses insert, swap and inverse operations alternatively at each generation to avoid being trapped into local optimal points. This algorithm selects one operation from above mentioned three operations and applied on best solution \( x \) found by EDCABC in each iteration. It attempts to move from the current solution \( x \) to its neighbourhood \( x' \). If the objective fitness of \( x' \) is smaller than the fitness of the current solution, \( x' \) is accepted as a new basic solution. After finishing one operation, the search process keeps generating the individual’s neighbourhood randomly and the solution is accepted until the stopping criterion is reached. The fast local search algorithm is listed in below.
In the below algorithm, Let \( pr_s \) be the probability of executing swapping operation, \( pr_i \) be the probability of executing inserting operation, \( pr_{inv} \) be the probability of executing inversing operation.

1. Let \( X_p \) be the best individual to be improved and \( n \) be the dimension of the solution.
2. Initialise probability value for swapping, insertion and inverse operation \( pr_s, pr_i \) and \( pr_{inv} \) respectively.
3. Evaluate fitness \( S(X_p) \).
4. for \( l = 1 \) to \( n*(n-1) \)
   a. \( q = \text{rand}(0) \)
   b. if \( (0 \leq q \leq pr_s) \)
      Execute swapping operation on individual \( X_p \) and obtain individual.
   c. Else if \( (pr_s \leq q \leq pr_s + pr_i) \)
      Execute insertion operation on individual \( X_p \) and obtain individual \( X_p \).
   d. Else if \( (pr_s + pr_i \leq q \leq pr_s + pr_i + pr_{inv}) \)
      Execute inversing operation on individual \( X_p \) and obtain individual \( X_p \).
   e. End if
   f. Evaluate fitness \( S(X_p) \)
   g. if \( (S(X_p) - S(X_p) \geq 0) \)
      \( X_p = X_p \)
      \( S(X_p) = S(X_p) \)
   h. End if
5. Endfor

6 Experimental evaluation

Cloudsim toolkit (Calheiros et al., 2009) is used to simulate the task scheduling problem and evaluate the proposed approach in the cloud environment. Cloudsim is a Java-based simulation toolkit that enables modelling and simulation of cloud computing resources and task scheduling environment. The cloudsim toolkit models the cloud system components such as data centres, VMs and resource provisioning policies. The simulation environment demonstrated on an Intel dual core machine with 4 GB RAM and 500GB hard disk. The simulation design for the proposed algorithm consists of three data centre with five hosts. Let us consider that each host consists of five VMs. The speed of the VMs can be measured by million instructions per second (MIPS). With the varying number of cloudlets, simulation is conducted in cloudsim to test the proposed algorithm.

This section demonstrates the performance of the proposed EDCABC algorithm for multi-objective task scheduling problem by comparing with highly competitive techniques: GA-based non-dominated sort genetic algorithm (NSGA-II) (Deb et al., 2002), weight-based multi-objective particle swarm optimisation (MOPSO) (Ramezani et al., 2013) algorithm. The efficiency of the proposed algorithm is evaluated with help of
test suits which were drawn from workload of high performance computing centre called HPC2N (U. University, 2006).

6.1 Performance metrics

The proposed multi-objective task scheduling algorithm is evaluated by the following metrics (Li and Yin, 2013; Koduru et al., 2007):

1. **Generation distance (GD)** is the metric which measures the distance between true Pareto front (obtained by combining the non-dominated solutions over 10 runs) and Pareto front produced at termination stage of algorithm.

2. **Spacing metric (SM)** is used to measure the spacing between solutions in Pareto front in order to measure diversity among the solutions. GD and Spacing can be represented as follows:

   For GD,
   \[
   d_i = \min_{j=1}^{\mid P \mid} \sqrt{\sum_{i=1}^{F} (S_i(k) - S_i(j))^2} \quad (14)
   \]
   \[
   GD = \frac{\sum_{i=1}^{\mid P \mid} d_i}{\mid P \mid} \quad (15)
   \]
   where \( F \) is the number objective functions, \( S_i \) is the fitness value for objective \( i \) and \( d_i \) is the Euclidean distance between true Pareto front solutions \( Q \) and nearer Pareto front solutions. Average of \( d_i \) is used to calculate GD.

   For SM,
   \[
   d_i = \min_{j=1}^{\mid P \mid} \left[ \sum_{i=1}^{F} |S_i(k) - S_i(j)| \right] \quad (16)
   \]
   \[
   SM = \frac{\sum_{i=1}^{\mid P \mid} (\overline{d} - d_i)^2}{F - 1} \quad (17)
   \]
   where \( d_i \) is the distance between the solution and its nearest solution in the Pareto front solutions and it is different from Euclidean distance. \( \overline{d} \) is the mean value of the \( d_i \). The lowest value of GD and SM are advisable for heuristics algorithms.

6.2 Best compromise solution

To select the best compromised solution among the non-dominated solutions in Pareto optimal set, the following simple linear membership function (Abido, 2003) acts as decision maker.
Epsilon-fuzzy dominance sort-based composite discrete artificial bee colony

\[
    \mu = \begin{cases} 
    1, & F_i \leq F_i^{\min} \\
    \frac{F_i^{\max} - F_i}{F_i^{\max} - F_i^{\min}}, & F_i^{\min} \leq F_i \leq F_i^{\max} \\
    0, & F_i \geq F_i^{\max} 
    \end{cases}
\]

where \( F_i^{\max} \) and \( F_i^{\min} \) are the maximum and minimum value of the \( i^{th} \) objective function in the Pareto set. The normalised membership function \( \mu_k \) for \( k^{th} \) non-dominated solution is defined as:

\[
    \mu_k = \frac{\sum_{i=1}^{F} \mu_{i}^{k}}{\sum_{k=1}^{Q} \sum_{i=1}^{F} \mu_{i}^{k}}
\]

where \( Q \) is the number of non-dominated solutions in the Pareto set and \( F \) is the number of objective functions in the MOP. The best compromised solution is solution which is having highest membership function value.

6.3 Evaluation of proposed approach

To evaluate the proposed approach, test suits have been taken from the above mentioned High performance computing centre workload. In the proposed multi-objective approach, makespan, execution cost and load balancing index are examined to analyse the characteristics of EDCABC algorithm. The simulation results are presented and analysed by using tables and graphs as shown below. Initially, individual objectives are examined and then multiple objectives are considered.

Figure 4 shows the makespan of three algorithms based on ten VMs with different task sets. The proposed EDCABC algorithm provides better results when compare to other two methods NSGA-II and MOPSO. For small datasets, makespan of EDCABC algorithm is quite closer to other two algorithms. However, as the number of tasks increases, the proposed approach has a higher chance to reduce the makespan by 9%–11% compared to NSGA-II and 17%–18% compared to MOPSO because EDCABC algorithm is able to avoid the premature convergence by using the composite mutation strategies and fast local search algorithm.

Figure 4  Measure of makespan using ten resources (see online version for colours)
The targeting results of the execution cost using EDCABC with other two methods are shown in Figure 5. Simulations are conducted by executing the different task sets in ten VMs. When the number of tasks increases, the proposed approach reduces the execution cost by 8%–10% compared to NSGA-II and 20–22% compared to MOPSO.

Table 2 Performance metrics of EDCABC with various task sets

<table>
<thead>
<tr>
<th>Objective</th>
<th>20 tasks</th>
<th>50 tasks</th>
<th>100 tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan(s)</td>
<td>397.3604</td>
<td>473.1025</td>
<td>589.7031802</td>
</tr>
<tr>
<td>Execution cost (USD)</td>
<td>92.92261</td>
<td>102.4792</td>
<td>241.4455158</td>
</tr>
<tr>
<td>Load balancing index</td>
<td>0.104925</td>
<td>0.133639</td>
<td>0.165623055</td>
</tr>
</tbody>
</table>

Table 3 Performance metrics of EDCABC, NSGA-II and MOPSO

<table>
<thead>
<tr>
<th>Objective</th>
<th>EDCABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan(s)</td>
<td>473.1025</td>
<td>478.7824</td>
<td>502.749283</td>
</tr>
<tr>
<td>Execution cost (USD)</td>
<td>102.4792</td>
<td>132.4624</td>
<td>141.398723</td>
</tr>
<tr>
<td>Load balancing index</td>
<td>0.133639</td>
<td>0.283456</td>
<td>0.49528425</td>
</tr>
</tbody>
</table>

Table 4 Best compromised solution generated by EDCABC algorithm for ten resources with different task set

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Metrics</th>
<th>EDCABC</th>
<th>NSGA-II</th>
<th>MOPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>GD</td>
<td>0.015111</td>
<td>0.01607</td>
<td>0.02653</td>
</tr>
<tr>
<td></td>
<td>Spacing (SM)</td>
<td>0.033596</td>
<td>0.0452</td>
<td>0.19501</td>
</tr>
<tr>
<td>50</td>
<td>GD</td>
<td>0.013113</td>
<td>0.01735</td>
<td>0.02492</td>
</tr>
<tr>
<td></td>
<td>Spacing (SM)</td>
<td>0.041923</td>
<td>0.06967</td>
<td>0.19027</td>
</tr>
<tr>
<td>100</td>
<td>GD</td>
<td>0.012713</td>
<td>0.01689</td>
<td>0.02010</td>
</tr>
<tr>
<td></td>
<td>Spacing (SM)</td>
<td>0.088159</td>
<td>0.093173</td>
<td>0.18355</td>
</tr>
</tbody>
</table>

In the multi-objective task scheduling problem, execution cost and makespan are considered together as shown below. The Pareto optimal solutions are analysed and generated by EDCABC, NSGA-II and MOPSO as well as best compromise solution [using ‘equations (18) and (19)’] among Pareto solutions are obtained with help of fuzzy approach as shown in Figure 6.
Even though, NSGA-II and MOPSO were used to generate non-dominated solutions at each iteration, level of non-domination between solutions were not measured, so the solutions are not closer to Pareto front when compare to EDCABC algorithm. Since the usage of perimeter operator and Epsilon-fuzzy dominance in the proposed approach, it selects the solutions near to Pareto front and shown that the diversity of the Pareto optimal solutions of the EDCABC algorithm is better than other two algorithms. The best compromise solution among 25 Pareto front solutions for different task set were shown in Table 2 and these different set of tasks were run using ten resources. When number of tasks was increased, then execution time, cost and load metrics values were increased. The performance metrics of three different algorithms were shown in Table 3. The best compromise solution of the proposed algorithm is compared with remaining two algorithms. It shows that the proposed algorithm performs better than other two algorithms. The results generated by (taking average of 20 runs) the three algorithms based on GD and Spacing metrics were compared in Table 4. Since the Epsilon-fuzzy dominance sort technique in the proposed approach is selected the better solutions for the next iteration, Pareto solution set is very nearer to Pareto front which causes the smaller value of GD for the proposed algorithm. The lower value of spacing metric of the proposed approach represented that there is a uniform spacing between solutions in the Pareto optimal set.

Figures 7 and 8 show that makespan and execution cost reduced when the number of iterations is increased. Since epsilon-fuzzy dominance sorting selects the solutions which are near to Pareto front, the improvement in the quality algorithm at each iteration is shown. However, NSGA-II and MOPSO were getting struck into local optimum, but composite mutation strategies and fast local search algorithm in EDCABC give the better and quicker convergence as well as reduces the makespan and execution cost as compare to other two algorithms. This shows that EDCABC is a better approach for multi-objective task scheduling problem in cloud environment.
Figure 7  Convergence analysis for makespan (see online version for colours)

![Convergence analysis for makespan](image)

Figure 8  Convergence analysis for execution cost (see online version for colours)

![Convergence analysis for execution cost](image)

7 Conclusions

The multi-objective task scheduling problem in cloud environment is solved using Epsilon-fuzzy dominance sort-based CABC algorithm. The proposed approach considers three conflicting objectives such as makespan, execution cost and average load balancing index while solving the task scheduling problem. The composite mutation strategies and fast local search methods in EDCABC algorithm provide a good balance between global exploration and local exploitation abilities which enhances convergence speed and quality of the solution.

The simulation results show that EDCABC approach produces better results when compare to NSGA-II and MOPSO in terms of convergence towards Pareto front set. Uniform space between solutions in Pareto set reduces the computation overhead with help of Epsilon-fuzzy dominance sorting method and perimeter operator. The best compromised solution among the Pareto optimal solutions are generated using linear membership function. The load among heterogeneous cloud computing systems is balanced better than NSGA-II and MOPSO. In future research, multi-objective task scheduling problem will be implemented in a real cloud environment. Hybrid heuristics approach will be used to schedule tasks which will consider task priorities and extend problem to minimise the energy consumption.
References


