ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803/tg-20210204183323

Original scientific paper

Bio-Inspired Workflow Scheduling on HPC Platforms

Mandeep Kaur*, Sanjay Kadam

Abstract: Efficient scheduling of tasks in workflows of cloud or grid applications is a key to achieving better utilization of resources as well as timely completion of the user jobs. Many scientific applications comprise several tasks that are dependent in nature and are specified by workflow graphs. The aim of the cloud meta-scheduler is to schedule the user application tasks (and the applications) so as to optimize the resource utilization and to execute the user applications in minimum amount of time. During the past decade, there have been several attempts to use bio-inspired scheduling algorithms to obtain an optimal or near optimal schedule in order to minimize the overall schedule length and to optimize the use of resources. However, as the number of tasks increases, the solution space comprising different tasks-resource mapping sequences increases exponentially. Hence, there is a need to devise mechanisms to improvise the search strategies of the bio-inspired scheduling algorithms for better scheduling solutions in lesser number of iterations/time. The objective of the research work in this paper is to use bio-inspired bacteria foraging optimization algorithm (BFOA) along with other heuristics algorithms for better search of the scheduling solution space for multiple workflows. The idea is to first find a schedule by the heuristic algorithms such as MaxMin, MinMin, and Myopic, and use these as initial solutions (along with other randomly generated solutions) in the search space to get better solutions using BFOA. The performance of our approach with the existing approaches is compared for quality of the scheduling solutions. The results demonstrate that our hybrid approach (MinMin/Myopic with BFOA) outperforms other approaches.

Keywords: BFOA; bio-inspired; cloud computing; HPC; makespan; scheduling; workflow

1 INTRODUCTION

High performance computing is about the use of high-productivity computing resources to solve challenging problems in scientific and engineering domains [1]. The HPC platform could comprise workstations, desktop machines, supercomputers, grid or cloud [2]. Grid computing is a kind of HPC loosely coupled collection of heterogeneous resources that are shared by the grid users for utilizing the ideal and under-utilized capacity of the resources [3, 4]. On the other hand cloud computing is a simplified form of grid computing that provides virtual server instance on shared resources based on user specifications [5].

Scientific computing is becoming more relevant in many research disciplines. A typical application may contain several dependent tasks specified as a workflow, which requires efficient scheduling of the tasks. The scheduling of workflows is a challenging task in HPC environment because of inter-dependency of the tasks that needs to be taken care of while scheduling the workflow tasks [6]. The grid/cloud meta-schedulers are responsible for fetching the matched resources that are capable to run the workflow application and schedule these workflows on the available resources [7]. The most important scheduling criterion for HPC environment is to produce a schedule with minimum schedule length so as to optimize the utilization of the resources [8].

During the past decade there have been several attempts to use bio-inspired scheduling algorithms to obtain an optimal or near optimal schedule of the tasks on a specified set of resources in order to minimize the overall schedule length and to optimize the use of resources. However, as the number of tasks increases, the solution space consisting of mapping of tasks to corresponding resources increases exponentially. Hence, there is a need to devise mechanisms to improvise the search strategies and/or mechanisms of the bio-inspired scheduling algorithms for better scheduling solutions in lesser number of iterations/time.

The objective of the research work in this paper is to use bio-inspired bacteria foraging optimization algorithm (BFOA) along with other heuristics algorithms for better search of the scheduling solution space. The idea is to first find a schedule by the deterministic or heuristic algorithms such as MaxMin, MinMin, and Myopic, and use these as initial solutions in the search space to get better solutions using BFOA. The advantage of using BFOA over other nature inspired evolutionary approaches is that it is computationally efficient and has good global convergence [9]. The performance of our approach with the existing approaches is compared for quality of the scheduling solutions. The results demonstrate that our hybrid approach (MinMin/Myopic with BFOA) outperforms other approaches.

The paper is organized into five sections. The first section provides background details and motivation of the research work. The second section provides overview of the workflow scheduling mechanism and also provides insights into the existing workflow scheduling approaches. The third section describes our proposed workflow scheduling approach and fourth section is about the experimental setup. The fifth section provides detailed discussion on the results and observations. The last section concludes the research work presented in this paper.

2 WORKFLOW SCHEDULING PROBLEM

The dependent-task or workflow application in grid and cloud environment is represented as a standard task graph (STG) or directed acyclic graph (DAG) as shown in Fig. 1. In DAG, graph vertices represent tasks and the edges represent task dependencies [8]. There could be n number of dependent tasks and m number of resources; it is very difficult to predict the best schedule with respect to large

mapping combinations between the tasks and the resources. Therefore, the workflow scheduling problem is a non-deterministic polynomial (NP)-complete problem [10]. The workflow scheduling problem can be solved by heuristic methods but the complexity of producing an appropriate schedule becomes high. Therefore, the metaheuristic methods are adopted to produce the near optimal schedule in heterogeneous distributed environment.

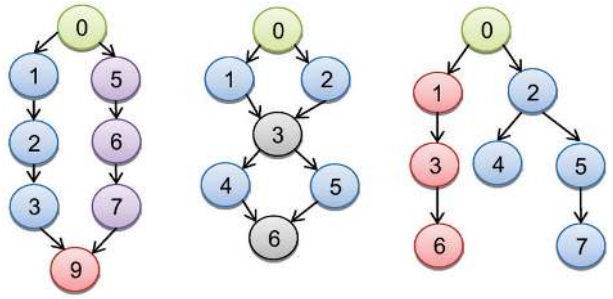

Figure 1 Workflow tasks

Let us assume workflow $W(T, E)$ consists of a set of tasks, $T = \{T_1, T_2, \dots, T_x, \dots, T_y, \dots, T_n\}$, and a set of dependencies among the tasks, $E = \{< T_a, T_b >, \dots, < T_x, T_y >\}$, where T_x is the parent task of T_y. The set $R = \{R_1, R_2, \dots, R_m\}$ represents the set of suitable resources in the Cloud. Therefore, the dependent task scheduling problem is the mapping of workflow tasks to Cloud resources ($T \rightarrow R$) so that the makespan M is minimized. The overall timespan of a complete schedule is known as total schedule length or makespan [11].

Generally, a workflow is a set of dependent tasks. The entry task does not have any predecessing task and exit task does not have any successor task [7]. Each dependent task can be executed after the completion of its parent tasks. If a child task is dependent upon more than one parent than it has to wait until all the predecessing tasks complete their execution. The child task becomes a ready task when all the parent tasks complete their execution. If the child task executes upon the same resource where parent task has finished its execution then data transfer time is considered to be zero.

The multiple users can submit multiple workflows to the meta-schedulers. The two important aspects that have been taken care by our meta-scheduler for scheduling the workflow applications are: (1) the parallel handling of multiple workflow applications, and (2) the scheduling of workflow applications on heterogeneous and distributed resources.

3 CURRENT SOLUTIONS IN WORKFLOW SCHEDULING

The current HPC and cloud meta-schedulers use many heuristic algorithms to schedule the workflow applications. The most popular methods are described below.

Myopic-Myopic heuristic is based on the minimum time to compute strategy, where each ready task is assigned to the resource that is capable to complete the ready task at the earliest. Myopic heuristic is one of the simplest scheduling techniques for scheduling dependent tasks in grid environment because it considers a single task while allocating the resource for scheduling. The myopic heuristic is implemented in some real HPC environments such as Condor DAGMan [12]. The Myopic algorithm schedules the ready tasks one after other until all the tasks in ready queue get scheduled. It maps each task to the resource that can process the task at the earliest.

MinMin-This scheduling heuristic prioritizes dependent tasks according the task sizes and schedules the tasks based on the sizes of the tasks [13]. The Min–Min scheduling heuristic maps shortest task on the fastest. The task having minimum expected time for execution over all tasks is selected to be scheduled first on the fastest resource that takes minimum time for execution during each iteration and it keeps scheduling all the tasks until the ready queue is exhausted. MinMin is implemented in real HPC environment such as vGrADS[14] of Rice University USA.

MaxMin-The MaxMin [15] scheduling heuristic prioritizes the dependent tasks according to the expected time to compute, the task that requires the longest execution time is allocated to the fastest resource that is capable to process the task at the earliest. This heuristic arranges the workflow tasks into multiple independent task groups and schedules each group of tasks iteratively. In each iterative step, a task with maximum time to compute is selected to be scheduled on the fastest resource that can process the task at the earliest.

HEFT-Topcuoglu H. et al. proposed a list scheduling method known as Heterogeneous Earliest Finish Time (HEFT) [16], which sets higher priority to the dependent tasks having higher rank value. The rank value is based on the average execution time of each task and average data transfer time between the predecessor and successor task, where the tasks in the critical path have higher rank values. Afterwards, this heuristic sorts the tasks by the descending order of the rank values of the tasks and the task with a higher rank value are set to higher priority. During actual scheduling, tasks in a workflow are scheduled in the order of their priorities, and each task is assigned to the fastest resource that can process the task at the earliest.

PSO-Particle Swarm Optimization (PSO) is a random based scheduling approach that searches the search space to find the near optimal solution. The position of each particle in search space represents a potential solution. The swarm represents the total number of predefined particles. The particle swarm optimization explores the search space by using position vector and terminates when predefined stopping criterion is met. The best particle that provides minimum value for objective function, that is, makespan, is selected as a final solution. In [17], authors have implemented PSO based scheduling approach to minimize the makespan.

GA- Genetic algorithm is also a metaheuristic approach [18] that generates random solutions to achieve near optimal solution. In GA, each individual represents a potential solution. The search space is explored using crossover and mutation operators. The GA terminates after a predetermined stopping criterion is met. The best solution obtained during evaluations is printed as a final solution. The final solution

represents task-resource mapping and scheduling timing. In [19], authors have presented workflow scheduling using GA with the objective of minimization of overall schedule length.

GRASP-The Greedy methods randomized adaptive search procedure (GRASP) [20] is an iterative approach that searches the solution on random basis. In GRASP, predetermined iterations are conducted to search a near optimal solution for scheduling the tasks on available resources. A new solution is generated in each iteration and the best solution among all the iterations is taken as the final solution. This method determines the minimum and maximum time to compute for each task on the available resources. The average time is determined by applying GRASP equation for executing a ready task on the available resources. All the resources that take lesser or equal time to the average time (obtained from GRASP equation) are considered for scheduling decision and any one resource is allocated to the ready task on random basis.

3.1 Related Work

There exists many state-of-the-art works for scheduling workflow tasks in HPC environment but most of the approaches are capable to handle single workflow at a time level-wise.

Rahman et al. [21] have presented a dynamic workflow scheduling approach known as DCP-G that minimizes the workflow execution time dynamically along with reducing the scheduling overhead. Bogdan et al. [22] have introduced an improved critical path using descendant prediction method for workflow scheduling, which is known as ICPDP. This approach performs well for minimizing makespan and for balancing the load of HPC resources. It also minimizes idle time of processing elements to enhance the resource utilization. Wang et al. [23] have presented an extensive approach named look-ahead genetic algorithm (LAGA), which optimizes both makespan and reliability of workflow tasks. LAGA uses an evolution and evaluation method as a two phased methodology. In first phase, the evolution operators of GA decide the task-resource mapping and second phase allows the evaluation steps to govern the task order of solutions using max-min strategy.

Amalarethinam and Selvi [24] have proposed minimum makespan grid workflow scheduling (MMGWS) that minimizes makespan of the workflows in HPC. This approach makes advance reservation of the desired resources and schedules the tasks on the basis of their respective priorities. The results of proposed approach are compared with Min-Min and HEFT scheduling algorithms. Garg et al. [25] have presented an adaptive workflow scheduling (AWS) to optimize makespan considering dynamic availability of the resources. This algorithm also takes care of load balancing by rescheduling the tasks to new resources form overloaded resources.

The existing approaches have not explained that the performance of bio-inspired algorithms deteriorates if the search space is huge. The performance depends upon the quality of scheduling solutions and the computational time to obtain the near optimal schedule. Hence, we are proposing bio-inspired hybrid BFOA approach for better search of the scheduling solution that provides scheduling solutions of better quality within less computational time. The idea is to first find a schedule by the heuristic algorithms such as MaxMin, MinMin, and Myopic, and use these as initial solutions (along with other randomly generated solutions) in the search space to get better solutions using BFOA. Most of the existing workflow scheduling approaches are based on single workflows whereas we have addressed scheduling problem of multiple workflows which are to be scheduled in parallel. The performance of our approach with the existing approaches is compared for quality of the scheduling solutions. The results demonstrate that our hybrid approach (MinMin/Myopic with BFOA) outperforms other approaches.

4 PROPOSED APPROACH

In this paper, a Bacterial foraging optimization algorithm based workflow scheduling mechanism is presented with two different aspects a) Starting with some random solution and searching for the optimal or sub-optimal scheduling solution, 2) Starting with a solution generated by MaxMin, MinMin and Myopic scheduling strategies and then searching for the optimal or sub-optimal schedule from this starting solution.

4.1 Bacterial Foraging Optimization Algorithm (BFOA)

Kevin Passino proposed the Bacterial Foraging Optimization Algorithm (BFOA) in 2002. BFOA provides vigorous search techniques that let a high quality solution to be achieved within a large search space [26].The bacterial foraging algorithm explores the new regions of the search space by chemotaxis and elimination-dispersal process and BFOA exploits the best solutions from the past searches through reproduction process. A bacterium is any solution in the search space, which is represented by a set of parameters. A bacteria foraging optimization algorithm maintains a bacteria population consisting of a set of bacterium that evolves over generations [26].The quality of a bacterium in a bacteria population is determined by an objective function. A typical Bacterial Foraging Optimization Algorithm consists of the following steps:

Algorithm

1. Initialize B, p, N_c, N_{re}, N_s, N_{ed}, P_{ed} and $S(i)$, (i = 1, 2, …, B).
   Choose the initial values randomly for θi where $i=1,2....B$ in the search space. B represents population of bacteria;N_{ch} represents number of chemotactix steps; N_s represents swim length; N_{ed} represents the probability of removal of bacteria; $S(i)$ specifies the size of the step taken in diverse direction during tumbling. The position p of each bacterium in bacteria population B is updated automatically and the iterations stop after meeting the stopping criteria.
2. Elimination loop: $el= el+1$.
3. Reproduction loop: $r=r+1$
4. Chemotactic loop: $c =c +1$

a) For i = 1, 2, …, B take a chemotactic step for i^{th} bacterium.
b) Compute fitness $C(i, c, r, el)$.
c) Let $C(i, c, r, el) = C(i, c, r, el) + C_{cc}(\theta^i(c, r, el), \theta(c, r, el))$
d) Let $C_{previous} = C(i, c, r, el)$ to retain this value until a better cost/fitness is found.
e) Tumble: Create a random vector $\Delta(i) \in \Re p$ with each element $\Delta k(i) \in [-1, 1]$ (k = 1, 2, …, p).
f) Make a movement with a step of size $S(i)$ for i^{th} bacterium in the direction of the tumble.

$$\theta^i(c+1, r, el) = \theta^i(c, r, el) + S(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

g) Compute $C(i, c + 1, r, el)$
h) Swim.
Let m = 0 (Initialize the swin length counter)
While $m < N_s$ Let $m = m + 1$
If $C(i, c + 1, r, el) < C_{previous}$ (if there exists improvement), let $C_{previous} = C(i, c + 1, r, el)$ and let

$$\theta^i(c+1, r, el) = \theta^i(c+1, r, el) + S(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

Use this $\theta^i(c + 1, r, el)$ to compute the new $C(i, c + 1, r, el)$.
Else, let $m = N_s$. End of while Loop.
i) Move to next bacterium (i + 1), if $i \neq B$
5. If $c < N_c$ go to step 3. In this case, repeat chemotaxis steps, till the end of bacteria life.
6. Go for reproduction.
a. For the given reproduction r and elimination dispersal el, and for each i = 1, 2, 3, …, B, let

$$C^i_{health} = \sum_{c=1}^{Nc+1} C(i, c, r, el)$$ be the health of bacterium i.

Sort the bacteria in ascending order of health of bacteria as C_{health}.
b. The B_r bacterium with poor C_{health} values die and the other B_r bacteria with the best values split into two bacteria to keep the population size same.
7. If $r < N_{re}$, move to step 2.
8. Go for Elimination-Dispersal with the pre-determined probability P_{ed}. If $el < N_{ed}$, then go to step 1, otherwise end.

4.2 Problem Definition

Using BFOA algorithm to solve the workflow scheduling problem needs suitable representation of bacterium in the given bacteria population. The problem definition consists of the following:

- The problem consists of a set of resources and the tasks of one or more workflows that need to be scheduled
- The task sequence is fixed, while the resource allocation to the individual tasks varies
- A child task in any workflow can be scheduled only after the completion of its parent tasks.
- The expected execution time (EET) of each task is calculated on the basis of task size (specified in Millon Instructions) and processor's capacity (specified in MIPS (Million instructions per second).

The aim of the proposed work is to minimize the overall schedule length of the task-resource mapping explained in subsequent sections.

4.3 Objective Function

The scheduling of dependent tasks in a workflow focuses on some of the scheduling criteria such as minimizing the total schedule length, minimizing flowtime, minimizing the overall execution cost, executing the tasks within the user specified deadline. The objective function is used to evaluate the current bacteria (population) to produce quality solutions. In our research work, we are taking total schedule length as objective function, which is to be minimized to produce a potential schedule.

Let $T = \{t_1, t_2, ..., t_n\}$ be the n tasks in a given set of workflows that need to be scheduled on a set of m resources $R = \{r_1, r_2, …, r_m\}$. Let B be the start time of the first task, while F be the finish time of the last task in a schedule. The schedule length is defined as the total time span TS_i between B and F for i^{th} schedule. The objective function is to minimize the total schedule length TS_i or makespan over all possible schedules, that is, $\arg_i(\min\{TS_i\}, i \in \text{Schedules})$.

4.4 Initial Bacteria Population in BFOA

The initial population of the bacteria represents random scheduling solutions. That is, the position of each bacterium represents a possible schedule (possible solution in the problem space). The position of each bacteria is an n–dimensional vector, where the i^{th} element of the vector represents the resources ID on which the i^{th} task is executed. The task sequence remains constant, while the resource allocation changes across different schedules. The total number of bacterium is determined by the pre-decided bacteria population size.

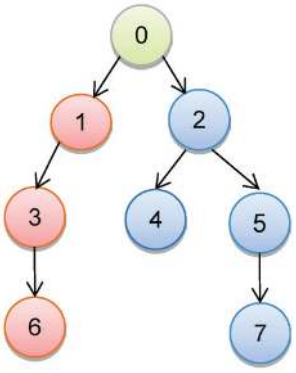

Figure 2 An example workflow

An example of Bacteria position is shown in Tab. 1, where the tasks T_0 to T_7 have fixed order, while the resource string varies across the tasks and also over different schedule solutions. Fig. 2 displays an example of workflow.

Table 1 Bacteria population sample

Position vector (R_1, R_2, …, R_0) of a bacterium							
T_0	T_1	T_3	T_2	T_4	T_5	T_6	T_7
R_1	R_2	R_3	R_1	R_4	R_5	R_6	R_0

The initial bacteria population in the bacterial foraging optimization algorithm could comprise a) Random solutions (positions), or b) Random solutions with a few bacteria positions initialized with solutions obtained by MaxMin, MinMin and Myopic scheduling strategies. The BFOA then searches for the optimal or sub-optimal schedule from this initial population.

The idea behind incorporating these heuristic strategies in BFOA is to enhance the exploration capability of the proposed approach. If there are n jobs and m resources, each job will have n^m combinations. For example, for 100 jobs and 10 resources, there would be 100^{10} combinations. This search space is huge and it may not be computationally feasible to find the near optimal solutions. With initial solutions obtained from some heuristic techniques such as MaxMin, MinMin, and Myopic, the exploration by the evolutionary algorithm would start from these positions or solutions and the algorithms would attempt to improve upon these solutions (find optimal/sub-optimal solutions), if applicable, in relatively lesser amount of time. The following sections describe the steps involved in BFOA in the context of scheduling problem.

4.4.1 Chemotactic Process

Chemotaxis process allows the bacterium to move towards the sources of food. The bacterium swims to change directions during this process and follows the same direction if it finds good fitness over the previously swimming steps. In this process, the bacterium explores search space for better solutions. The tasks are allocated to different combination of resources to achieve better fitness values. The movement of the i^{th} bacterium at the c^{th} chemotactic, r^{th} reproductive, and el^{th} elimination dispersal step can be mathematically expressed as follows:

Where Δ indicates a vector in the random direction whose elements lie in in [−1, 1].

The chemotaxis process allows the bacteria to explore the search space to find better solutions for the given problem. For example, task T1 is allocated to resource R1. The new allocation is 2 (1 (previous resource id) + 0.99987 (value achieved by tumbling process)). Table 1 shows the resource allocation to workflow tasks before and after the chemotaxis process. The task-resource mapping of each bacterium is changed during chemotaxis process if each bacterium finds better fitness; otherwise the previous solution is preserved.

4.4.2 Swarming

Swarming allows the population of bacteria gathers together and moves as concentric patterns of swarms. The cost or fitness of a bacterium position is affected by swarming. The health of bacteria is needed during reproduction step where bacteria with poor health die and bacteria with good health go for reproduction.

The swarming equation of bacteria is affected by the cell-to-cell signaling is given by Eq. (1), where $\|\ \|$ is the Euclidean norm, ωa and ωr are measures of the width of the attractant and repellent signals respectively, M measures the magnitude of the cell-cell signaling effect in the given swarming equation. The swarming equation drives certain weight that is added to cost (fitness value) of each bacterium to determine the health of bacterium.

$$C_{cc}(\varphi^i, \varphi) = \left\{ -M \left(\sum_{k=1}^{B} \mathrm{e}^{-\omega a \left\| \varphi^i - \varphi^k \right\|^2} - \sum_{k=1}^{B} \mathrm{e}^{-\omega r \left\| \varphi^i - \varphi^k \right\|^2} \right) \right\}. \quad (1)$$

For example, if the fitness value (makespan) of a bacterium is 100 seconds, the swarming weight is 44.999, and the health of the bacterium is 100 + 44.999 = 145. The better the health, better would be the bacterium position. The health of bacterium allows it to make decision for the swimming step, that is, whether to swim in the same direction if the health is improving or stay back at the previous position. The health of the bacterium plays a crucial role in reproduction step, where the bacterium with better health survives and bacterium with poor health dies.

4.4.3 Reproduction

The reproduction step allows the bacteria to exploit the search space. The objective here is to search a limited region of the search space with the possibility of improving the local solution. The existing solutions are refined here to improve the fitness value. After chemotactic steps, a reproduction step is followed. In reproduction, the bacteria with bad health die and the bacteria with good health split into two bacteria to keep the population size same [27]. The bacteria are sorted according to their health. The scheduling solutions with higher or poor makespan will be removed from the current bacteria population.

However, this step generates duplicate bacterium (scheduling solutions) in the bacteria population, but it is mandatory to remove the worse population (with poor fitness) from the current solutions.

4.4.4 Elimination-dispersal

This step allows the bacteria to search for the search space to get newer and refined solutions (if any), that is, searching a much larger portion of the search space with the possibility of finding better solutions. In elimination process, a bacterium is stochastically selected for elimination from the population and is replaced by a new bacterium located at a random new location within the search space, according to a predefined probability P_{ed}. For example if the elimination-dispersal probability is 0.25 then 25% of the scheduling solutions or bacteria will selected on random basis and will be replaced by newly generated scheduling solutions or bacteria.

However, this process introduces new population and removes redundant bacteria, which are generated in reproduction step. But it may also kill the best scheduling

solutions found so far. In order to preserve the best population elitism is applied during elimination-dispersal step. If the elitism rate is 10% then the top 10% of bacterium with minimum fitness or make span are preserved, while the rest 90% of bacteria undergo elimination dispersal process.

Tab. 8 is depicting the operational patameter values for BFOA.

4.5 Genetic Algorithm

Each chromose represents a potential solution in the problem space. The individual in population is generated in a similar manner as each bacterium is generated in BFOA. The total chromosomes are decided by the pre-determined population size. The genetic operators are explained below.

4.5.1 Selection

For selecting chromosomes for reproduction from the prevailing population, the tournament selection is applied. Two tournaments are held to select potential parents for mating. Parent 1 with better fitness value has been selected and this method is repeated in the second tournament to obtain second parent 2. The two parent chromosomes are selected for the crossover. Fig. 3 depicts the binary tournament selection procedure.

Table 2 Task resource assignment before chemotaxis

HPC task-resource assignment string							
T_0	T_1	T_3	T_2	T_4	T_5	T_6	T_7
R_2	R_3	R_1	R_2	R_3	R_3	R_0	R_6

Table 3 Task resource assignment after chemotaxis

HPC task-resource assignment string							
T_0	T_1	T_3	T_2	T_4	T_5	T_6	T_7
R_1	R_2	R_3	R_1	R_4	R_5	R_6	R_0

Table 4 Control operational parameters for BFOA

Sr. No	Parameters	Type/Values
1	Bacteria Population	50
2	Maximum number of steps, N_s	3
3	Number of chemo tactic steps, N_c	20
4	Number of reproduction steps, N_{re}	2
5	Number of elimination-dispersal steps, N_{ed}	z2
6	Probability, P_{ed}	0.25
7	Size of the chemotaxis step, $C(i)$	0.8

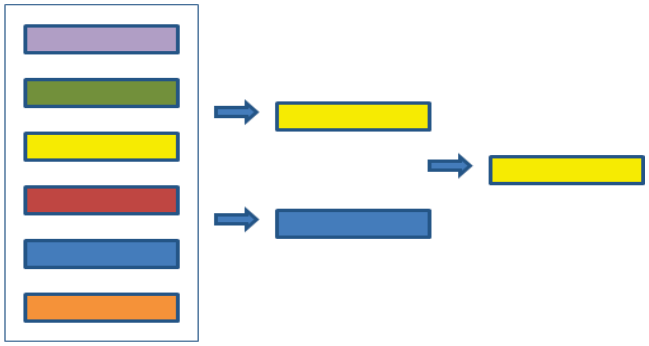

Figure 3 The binary tournament selection

4.5.2 Crossover

The crossover operator allows the individuals to search for new solutions. The parents selected in tournament selection go for reproduction of Offsprings/children during crossover [29]. In The two-point Crossover is used in our proposed approach. Tab. 5 and Tab. 6 are showing before and after crossover points.

Table 5 Before two-point crossover

Parent 1	R_1	R_2	R_3	R_4	R_3	R_4	R_5	R_6
Parent 2	R_0	R_7	R_5	R_9	R_1	R_3	R_2	R_8

Table 6 After two-point crossover

Offspring 1:	R_1	R_2	R_3	R_9	R_1	R_3	R_5	R_6
Offspring 2:	R_0	R_7	R_5	R_4	R_3	R_4	R_2	R_8

4.5.3 Mutation

Next to crossover operation, the mutation operation is applied to one chromosome based on the mutation probability. The mutation operation helps to diversify the population and to obtain new solutions. Replace Mutation is used in the research work presented in this paper. When the mutation probability of is decided by a chromosome with the selection of two random points and the resources of those tasks are interchanged.

Tab. 7 is depicting the controlled parameter values for GA. The fine tuning of controlled parameters is based on the empirical study of the GA based research outcome.

Table 7 Control operational parameters for GA

Sr. No	Parameters	Type/Values
1	Crossover	Two-Point
2	Crossover Probability	0.8
3	Mutation Type	Swap
4	Crossover Probability	0.2
5	Population Size	100

4.6 Particle Swarm Optimization (PSO)

Particle Swarm Optimization is an intelligence technique which is based on a swarm of particles moving in search space and communicating with each other for determining a near optimal solution [30]. Population of particles is known as swarm in PSO. Each particle is represented in a similar manner as a bacterium is represented in bacteria population. The parameters used in the PSO algorithms are:

Velocity (vector): This vector determines the direction in which a particle needs to fly in order to improve its current position in the flock.

pbest (personal best): It is the personal best position (solution) of a given particle found so far.

gbest (global best): Position of the best particle in the entire swarm.

Inertia weight: Denoted by ω, the inertia weight is used to control the impact of the previous history of velocities on the current velocity of a given particle. It can be taken as random value or constant value.

Learning factors: There are two learning factors used in PSO that is, $C1$ and $C2$. The parameter $C1$ represents the attraction of a particle towards its own success, while parameter $C2$ represents the attraction of a particle towards the success of global best position.

Every particle gets updated during each iteration by its personal best value, that is, *pbest* and the global best value, that is, *gbest*. The particle modifies its position with the help of velocity and position vector to explore the search space for better solutions (Doctor Kennedy and Eberhart in 1995).
When the position of a particle is changed, the task-resource mapping is changed. The tasks are assigned to different set of resources as long as the position of each particle is updated. Tab. 8 depicts the values of control parameters used for workflow scheduling in our research work.

Table 8 Control parameters for PSO

Sr. No	Parameters	Values
1	$C1$	1.25
2	$C2$	1
3	ω	0.9
4	$r1$	0.1
5	$r2$	1
6	*Swarm size*	100

4.7 Multiple Workflows Scheduling

In multiple workflows scheduling, a group of tasks are scheduled level-wise in the workflow trees using the heuristic algorithms (MinMin, MaxMin, and Myopic). We explain the multiple workflow scheduling mechanism that we have implemented in our work with one such heuristic algorithm namely, MinMin. We first take all the tasks on level one of all the workflow trees and schedule them using the MinMin algorithm. The task with minimum time to compute is picked first and is scheduled on the fastest resource (the processor which takes shortest time to execute the task). We then schedule the tasks at second level of the workflows. The shortest task is allocated to the fastest resource and this process continues until all the tasks at all the levels are scheduled. MinMin applies its scheduling strategy of shortest job on fastest resource at each workflow level and schedule the all the tasks in the workflows. The other heuristics such as MaxMin and Myopic also perform scheduling according to the levels of workflow. The only difference is their scheduling strategies. MaxMin schedules the longest task on the fastest resource at each workflow level, whereas, myopic chooses the tasks in an arbitrary fashion and schedules them on the fastest resources. The reverse is also true when meta-heuristic algorithms (BFOA, PSO, and GA) allocate a set of tasks to the corresponding resources.

4.8 Experimental Setup

The heterogeneous resources with different processing capacities are simulated and defined in terms of MI/sec (Million Instructions/sec). The size of each task is generated between 8000 MI and 20,000 MI from a uniform distribution. The total number of tasks in the workflow are N. Our workflow generator can generate single workflows with many dependent tasks and multiple workflows (with specific number of tasks in each workflow). In our case study, 10 tasks have been generated in each workflow. If there are 10 workflows then the total number of tasks would be 100. The workflows for evaluation are created using the following parameters:

- Type = Random workflows
- One workflow= 10 tasks each
- $N = \{50, 100, 200, 300, 400\}$
- $M = \{5\}$

Note that for GRASP, 500 iterations are considered and the value of α is set to 0.01.

5 RESULTS AND OBSERVATIONS

The scheduling heuristics are evaluated on the basis of total schedule length. The total schedule length for a set of workflows or dependent tasks is defined by the time span between the start of the first task and the end of the last task. Two sets of scheduling simulations have been carried out. In the first simulation, the search for scheduling solutions was initiated with a random generation of solutions in the search space. The exploitation and exploration of the solution space has been performed in order to refine the existing solutions and to search newer solutions using the metaheuristics algorithms such as BFOA, PSO and GA.

In the second simulation, a hybrid approach is used where heuristic algorithms such as MaxMin, MinMin, and Myopic, have been used to get initial solutions in the search space. These along with randomly generated solutions serve as the starting solutions in the search space to the metaheuristics algorithms for possible improvements in the scheduling solutions. The performance of our hybrid approach is compared with the existing approaches for quality of the scheduling solutions. The results demonstrate that our hybrid approach (MinMin/Myopic with BFOA) outperforms other approaches.

The space complexity increases exponentially with increase in task-resource combinations. For 200 tasks and 5 resources, there exist 5^{200} scheduling solutions. The evolutionary techniques perform well when the search space is small (less number of tasks and lesser number of task-resource combinations). As the search space complexity increases, it is difficult to determine the near optimal solution.

Tab. 9 and Fig. 4 depicts that heuristic algorithm perform well even if the number of workflows with number of tasks go on increasing while the quality of solutions obtained by BFOA and other evolutionary techniques such as GA and PSO starts deteriorating as the search space increases.

Table 9 Metaheuristic Hybrid Approach Vs Heuristic Approaches

Tasks	MaxMin	MinMin	Myopic	GRASP	GA	PSO	BFOA
50	165	145	139	138	134	132	126
100	290	244	226	233	243	242	228
200	602	518	504	501	519	516	509
300	765	633	643	607	677	681	665
400	996	822	811	840	957	925	913

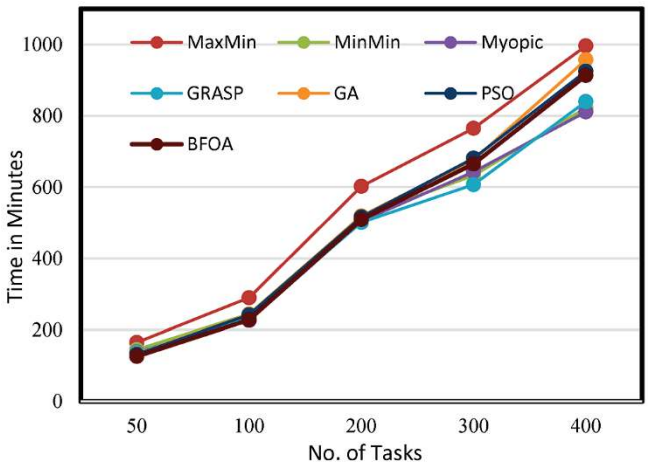

Figure 4 Heuristic approaches vs Metaheuristic approaches

Table 10 Metaheuristic vs Heuristic Approaches

Tasks	MaxMin	MinMin	Myopic	GRASP	GA	PSO	BFOA
50	165	145	139	138	121	120	115
100	290	244	226	233	218	225	202
200	602	518	504	501	476	496	446
300	765	633	643	607	587	581	575
400	996	822	811	840	793	796	774

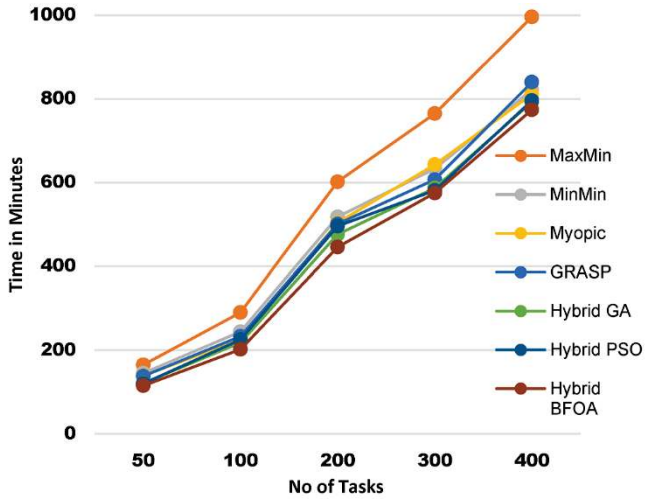

Figure 5 Heuristic approaches Vs Metaheuristic Hybrid Approach

Tab. 10 and Fig. 5 depicts that the hybrid approach improves the quality of scheduling solutions and minimizes the computational time to obtain the near optimal solution. If the metaheuristic algorithms start with solutions obtained by heuristic solution along with some randomly generated solutions, then the quality of solution improves significantly, because the exploration of the search space is guided in a better direction. The solutions generated by hybrid BFOA are improved by 8% for 50 tasks, 11% for 100 tasks, 12% for 200 tasks, 13% for 300 tasks and 15% for 400 tasks, respectively.

6 CONCLUSION

In this paper, we have used the bio-inspired bacteria foraging optimization algorithm (BFOA) together with other heuristic algorithms to find scheduling solutions for multiple workflows. Starting with a random schedule for the multiple workflows, the BFOA attempts to find a scheduling solution. However, this may not always result in optimal or sub-optimal solution and sometimes may take large amount of computation time to get to a desired scheduling solution. Hence, we have used a hybrid approach where heuristic algorithms such as MaxMin, MinMin, and Myopic have been used to get initial solutions in the search space, which then serve as starting points (along with other randomly generated solutions) for getting better solutions using bacteria foraging optimization algorithm. The performance of our hybrid approach presented in this paper is compared with the existing approaches. The results demonstrate that our hybrid approach (MinMin/Myopic with BFOA) outperforms other approaches.

Acknowledgement

The work has been carried out at Centre for Development of Advanced Computing (C-DAC), SPPU Campus using PARAM-HPC setup and we would also like to thank KPIT R & D, Pune to help us in handling of multiple workflows in parallel environment.

Notice

This paper was presented at IC2ST-2021 – International Conference on Convergence of Smart Technologies. This conference was organized in Pune, India by Aspire Research Foundation, January 9-10, 2021. The paper will not be published anywhere else.

7 REFERENCES

[1] Kaur, M. & Kadam, S. S. (2017). Discovery of resources using MADM approaches for parallel and distributed computing, *Engineering Science and Technology, 20*(3), 1013-1024. https://doi.org/10.1016/j.jestch.2017.04.006

[2] Rankin, S. (2016). An Introduction to HPC. http://www.hpc.cam.ac.uk/getting-help/introtohpc-course

[3] http://www.ianfoster.org/wordpress/wp-content/uploads/2014/01/History-of-the-Grid-numbered.pdf

[4] Kaur, M. (2012)- Semantic Resource Discovery with Resource Usage Policies in Grid Environment, *Int. J. Comput. Sci. Issues, 9*(5), 301-307.

[5] Shawish, A. & Salama, M. (2013). Cloud Computing : Paradigms and Technologies. *Inter-cooperative Collective Intelligence: Techniques and Applications*, 39-68. https://doi.org/10.1007/978-3-642-35016-0_2

[6] http://escience2015.mnm-team.org/wp-content/uploads/2015/09/deelman_escience_2015_keynote.pptx.pdf

[7] Rahman, M., Hassan, R., Ranjan, R., & Buyya, R. (2013). Adaptive workflow scheduling for dynamic grid and cloud computing environment. *Concurrency and Computation: Practice and Experience, 25*, 1816-1842. https://doi.org/10.1002/cpe.3003

[8] Yu, J., Buyya, R., & Ramamohanarao, K. (2008). Workflow Scheduling Algorithms for Grid Computing. *Metaheuristics for Scheduling in Distributed Computing Environments*, 173-214. https://doi.org/10.1007/978-3-540-69277-5_7

[9] Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial Foraging Optimization Algorithm : Theoretical

Foundations, Analysis, and Applications. *Foundations of Computational Intelligence, 3*(1), 23-55. https://doi.org/10.1007/978-3-642-01085-9_2

[10] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P. et al. (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing, 61*(6), 810-837. https://doi.org/10.1006/jpdc.2000.1714

[11] Kaur, M. (2016). FastPGA based scheduling of dependent tasks in grid computing to provide QoS to grid users. *IEEE International Conference on Internet of Things and Applications (IOTA)*, Pune, 418-423. https://doi.org/10.1109/IOTA.2016.7562764

[12] Couvares, P., Kosar, T., Roy, A., Weber, J., & Wenger, K. (2007). Workflow Management in Condor. research.cs.wisc.edu/htcondor/doc/workflow_condor_2007.pdf

[13] Kaur, M. & Kadam, S. (2018). A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling. *Applied Soft Computing, 66*, 183-195. https://doi.org/10.1016/j.asoc.2018.02.011

[14] Kennedy, K., Cooper, K., Koelbel, C., Tapia, R., & Torczon, L. (2009). The VGrADS Project, http://vgrads.rice.edu/

[15] Kaur, M. & Kadam, S. (2019). Discovery of resources over Cloud using MADM approaches. *International Journal for Engineering Modelling, 32*, 83-92. https://doi.org/10.31534/engmod.2019.2-4.ri.02m

[16] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst., 13*(3), 260-274. https://doi.org/10.1109/71.993206

[17] Lei Zhang, B. Y. & Chen, Y. (2006) Task Scheduling Based on PSO Algorithm in Computational Grid, *Sixth Int. Conf. Intell. Syst. Des. Appl., 2*, 696-704. https://doi.org/10.1109/ISDA.2006.253921

[18] Carretero, J., Xhafa, F., & Abraham, A. (2007). Genetic Algorithm Based Schedulers for Grid Computing Systems. *Int. J. Innov. Comput. Inf. Control*, 3(6), 1-19.

[19] Wang, X., Shin, C., Buyya, R., & Su, J. (2011). Optimizing Makespan and Reliability for Workflow Applications with Reputation and Look-ahead Genetic Algorithm. *J. Futur. Gener. Comput. Syst., 27*(8), 1124-1134. https://doi.org/10.1016/j.future.2011.03.008

[20] Resende, M. G. C. & Ribeiro, C. C. (2002). Greedy randomized adaptive search procedures. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 50*, 1-29.

[21] Rahman, M., Hassan, R., Ranjan, R., & Buyya, R. (2013). Adaptive workflow scheduling for dynamic grid and cloud computing environment. *Concurr. Comput. Pract. Exp., 25*(3), 1816-1842. https://doi.org/10.1002/cpe.3003

[22] Bogdan, S., Catalin, L., Florin, P., & Valentin, C. (2007). A Hybrid Algorithm for Scheduling Workflow Applications in Grid Environments. *Lect. Notes Comput. Sci., 4804*, 1331-1348. https://doi.org/10.1007/978-3-540-76843-2_15

[23] Wang, X., Shin, C., Buyya, R., & Su, J. (2011). Optimizing Makespan and Reliability for Workflow Applications with Reputation and Look-ahead Genetic Algorithm. *Futur. Gener. Comput. Syst., 27*(8), 1124-1134. https://doi.org/10.1016/j.future.2011.03.008

[24] Amalarethinam, D. & Selvi, F. (2012). A Minimum Makespan Grid Workflow Scheduling algorithm. *2012 International Conference on Computer Communication and Informatics*, Coimbatore, India, 2012, 1-6. https://doi.org/10.1109/ICCCI.2012.6158777

[25] Garg, R. & Singh, A. K. (2015). Adaptive work flow scheduling in grid computing based on dynamic resource availability. *Eng. Sci. Technol. an Int. J., 18*(2), 256-269. https://doi.org/10.1016/j.jestch.2015.01.001

[26] Kaur, M. (2016). Elitist Multi-Objective Bacterial Foraging Evolutionary Algorithm for Multi-Criteria based Grid Scheduling Problem. *IEEE International Conference on Internet of Things and Applications(IOTA' 2016)*, 431-436. https://doi.org/10.1109/IOTA.2016.7562767

[27] Dang, J., Brabazon, A., Neill, M. O., & Edelman, D. (2008). Option Model Calibration Using a Bacterial Foraging Optimization Algorithm. *Springer-Verlag Berlin Heidelb.*, *vol. LNCS 4974*, 113-122. https://doi.org/10.1007/978-3-540-78761-7_12

[28] Bhandari, D., Murthy, C. A., & Pal, S. K. (2012). Variance as a Stopping Criterion for Genetic Algorithms with Elitist Model. *Fundam. Informaticae, 120*, 145-164. https://doi.org/10.3233/FI-2012-754

[29] Maheswaran, M., Ali, S., Siegel, H. J., & Hensgen, D. (1999). Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. J*ournal of Parallel and Distributed Computing, 59*, 107-131. https://doi.org/10.1006/jpdc.1999.1581

[30] Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. *2010 IEEE International Conference on Advanced Information Networking and Applications*, 400-407. https://doi.org/10.1109/AINA.2010.31

Authors' contacts:

Mandeep Kaur
(Corresponding author)
Dept. of Comp. Sc., Savitribai Phule Pune University,
SPPU Campus, Ganeshkhind Road, Pune, Maharashtra 411007, India
mandeep.gondara@gmail.com

Sanjay Kadam
Centre for Development of Advanced Computing,
SPPU Campus, Ganeshkhind Road, Pune, Maharashtra 411007, India
sskadam@cdac.in