Improved GRASP Technique based Resource Allocation in the Cloud

Madhukar E[1]
Associate Professor, CSE
Sreenidhi Institute of Science and Technology
Hyderabad, India

Ragunathan T[2]
Professor & Dean, CSE
SRM University
Amaravathi, India

***Abstract*—In the era of cloud computing, everyone is somehow using cloud resources. However, the resources are limited in the Cloud. Cloud vendors look for enhanced returns on investments. Promising return on investment is possible only when the cloud resources are scheduled efficiently to execute jobs within the stipulated time. However, brute force methods require exponential time to produce a schedule. Heuristic and meta-heuristic algorithms have been proposed in the literature to allocate resources to the jobs. These algorithms still suffer from slow convergence. To overcome this problem, researchers clubbed various heuristics and meta-heuristic to form a new hybrid algorithm. With the same motive, this paper explores the limitations of greedy random adaptive search and shows that learning through a fixed set search enhances efficiency. Based on the results, it can be concluded that the proposed algorithm is on par with existing hybrid meta-heuristic algorithms**.

***Keywords*—Cloud computing; task scheduling; meta-heuristics; fixed set search; GRASP; resource allocation**

I. Introduction

Cloud computing is one of the emerging technologies in this era. It creates a new paradigm in information technology and computing. Conventional computing methods are ousted by Cloud computing by making "usage of computing as a utility" [1] which is charged on pay-as-you-use provision similar to "utilities like water, electricity, gas, and internet" [1].

Cloud provides a metered service that automatically delivers services when and where they are needed. It provides virtualized, well-managed, abstracted, and on-demand compute, storage, and network services with a deep internet backbone.

While cloud computing has origins in Cluster, Parallel, and Grid computing, it differs in terms of virtualization, resource pooling, elasticity, and heterogeneity from these technologies. The Cloud imposes several challenges to provide the features.

The challenges range from security, privacy, scalability, fault tolerance, energy consumption, interoperability, and scheduling. Cloud vendors have to be cautious with all these challenges to dispense the service to the users. It is an arduous task to stick to the service level agreement (SLA). Violations of SLA lead to many legal problems.

Resource scheduling is required to balance the service provider's challenges and fulfill the cloud user's requirements. However, resource scheduling is an NP-hard problem. With exhaustive search, it takes a longer time to give the schedule. Within a polynomial-time, it is not proven to give an optimal solution.

Cloud computing provides an infinite number of resources on demand. The users can focus on business innovations rather than focussing on the accumulation of physical resources. Its pervasiveness helps the users by providing the resources on-demand, convenient, and tailored to the requirements. Many start-ups can effectively utilize the services offered by cloud vendors. Special pricing schemes are offered to the corporate users, as the resources are needed at a large scale. The services can be divided into IaaS, PaaS, and SaaS. The phrase "everything as service" XaaS was termed in [2]. Many companies have moved to the Cloud to compute and store information. The characteristics of cloud computing benefit both cloud providers and users.

Three types of clouds exist based on their ownership. Public, private, and hybrid clouds fulfill the user's requirements. Public Cloud is made up of resources of third-party companies. Private clouds, in general, will be located on the premises of the organizations. Hybrid clouds consist of combined features of public and private clouds.

Resource scheduling in a Cloud computing environment is an important phenomenon. Researchers have been proposing new heuristic and metaheuristic algorithms. In this paper, an improved Greedy randomized adaptive search algorithm has been proposed. The learning mechanism is added to the existing algorithm. It is shown that the algorithm's simplicity is not lost even after the addition of the learning to the existing algorithm. Tasks will be allocated to the virtual machines with efficiency. The main objective is to minimize the makespan.

A. GRASP

The "Greedy Random Adaptive Search Procedure (GRASP)" [3] is one among many familiar meta-heuristic algorithms proposed by Feo and Resende. It helps to solve the NP hard problems, precisely like combinatorial problems. This algorithm has two phases in each iteration. The algorithm begins with a "randomized greedy allocation" [3]. The second phase will be added with a procedure of local search with existing solutions. If any improvement is added to the existing solution in the objective function, the new solution is considered a new incumbent solution. This procedure is continued till it reaches an optimal solution or to stopping criteria. However, it has a limitation of not learning.

The addition of the local search method improves the performance of the metaheuristic algorithm. Hybridization is such a process that can combine one or more metaheuristics, to enhance efficiency. However, complexity may be increased if such hybridization is done.

Model-based heuristic algorithms are made up of identifying sets of parameters that define that model and help to find the target in the search space. These algorithms progressively modify their model after every iteration. In turn, the possibility of finding a quality solution is increased. The learning mechanism is part of Swarm intelligence, in which the information will be shared among the particles (components) so that the direction of the search will be changed. More emphasis is put on the learning mechanism in the recent past. In this phase, the focus is maintained to collect the information to enhance the quality of the solution.

GRASP algorithm falls into the class of population-based metaheuristics. The greedy function and stochastic model clubbed together to improve the efficiency of the algorithm. In addition to the GRASP, instances such as Semi-greedy heuristics[4] prove that rather than complex algorithms, simple algorithms also can give promising results.

In this paper, the boundaries of GRASP are extended by adding a learning mechanism [5]. A few examples of such methods are the "Dynamic Convexized method" [6] and "GRASP with Path relinking" [7]. These two algorithms show instances of intensified foraging behavior in the solution space. A significant level of improvement was shown with these methods.

The majority of the existing hybridized methods focus on the best quality solutions found so far to enhance the quality of the solution. However, algorithms like ACO use the elements of high-quality solutions with probability. The cross-entropy method (CE) [8] is akin to the concept in which the solutions are constructed based on the frequency of the elements from high-quality solutions. In the proposed algorithm, the focus is on the best elements of the finest solutions. The existing GRASP will be added with the theory of fixed set search to learn and select elements that direct towards the solution. Based on the prevalence of the elements which are part of high-quality solutions, a new solution is constructed.

II. Related Work

Scheduling started way back in Johnson's proposed work [9] to use the machinery efficiently as part of manufacturing. Scheduling had taken new directions with the invention of operating systems in computers. However, scheduling methods which have been part of the operating system are not suitable for the Cloud.

The Grid computing algorithms were tailor-made to suit Cloud computing. Batch and online mode heuristic [10] algorithms are two types of requirements for scheduling in the Cloud. Min-Min, Max-min, Round robin, and FCFS are part of batch mode. Most-fit task scheduling falls into online scheduling.

A. Deterministic and Exhaustive Algorithms

Online mode heuristic algorithms (OMHA) and batch mode heuristic algorithms (BMHA) [10] are two categories of scheduling algorithms in the Cloud. "First Come First Serve (FCFS), Round Robin(RR), Min- Min algorithms, and Max-Min algorithms come under BMHA. Most Fit Task Scheduling(MFTS) algorithms come under OHMA, in which schedule will be done when the job is received" [11]. First come, first serve, Shortest time remaining job, Priority scheduling, and Round-robin are not optimal for the Cloud.

The deterministic and exhaustive algorithms are two methods for scheduling algorithms. Both these methods are not suitable for large-scale environments like clouds. It is evident that finding the optimal solution within the polynomial-time for such NP-hard problems is not possible. The metaheuristic algorithms [12] could find the solutions within a short time by compromising the optimality. Simulated annealing(SA), Genetic algorithm (GA), Ant colony optimization, Particle swarm optimization (PSO) are a few such metaheuristic algorithms. Analytical Hierarchical processing was applied to prioritize the tasks in [13], which proved that there is some improvement in the makespan

B. Metaheuristic Scheduling Algorithms

The term "metaheuristic" was coined by Fred Glover in 1986 [14]. It indicates a heuristic with a non-problem-specific approach, and it is a combination of exploration and exploitation.

Applying metaheuristic algorithms for scheduling in the Cloud has become a common practice because of its efficiency. One among many such algorithms is ant colony optimization. It proved a significant improvement in the time complexity for optimization problems. It also proved that a near-optimal solution could be achieved by iteration after iteration. One such algorithm is proposed and applied by M. A Tawfeek et al. [15]. They compared with FCFS and round-robin algorithms. Since then, many researchers have shown some improvement by either hybridization or adding extra features.

Moon et al. [16] discuss ant colony optimization-based task scheduling. They claim that the global optimization problem was solved with slave ants by avoiding long paths where pheromone gets accumulated. Z. Chen *et al*. [17], used multiple populations in ACO to solve two objectives in the Cloud. They dealt with a new pheromone update by using non-dominated solutions from the global archive to guide a complementary heuristic to avoid the single-objective optimization.

In [18], the authors proposed an algorithm in which the greedy strategy is combined with the GA algorithm. They show that their method shows better results in task scheduling. The Differential algorithm in [19] was considered as one of the simple algorithms to search for the optimal solutions in the search space. To derive potential off-springs, better individuals were applied with the Taguchi method. In [20], the moth search algorithm and differential algorithms were hybridized. In the presence of Levy flights, they used a differential evolutionary algorithm to enhance the exploitation potential and used phototaxis for explorations. In [21], the authors integrate project scheduling along with the workflow scheduling problem. Two artificial bee colony algorithms proposed by them help to solve the workflow scheduling. They claim that their method is practically applicable for complicated workflow scheduling problems. In [22], the researchers discuss the provision of resources with QoS such as makespan, cost, and task migration reduction. They show that their method achieves better results with their objectives with improved efficient artificial bee colony. In [23], by using whale optimization, they proposed a W-scheduler. Multi-objectives were proposed and compared with PBACO, SLPSOSA, and SPSO-SA. Agarwal et.al. [24], discusses the application of genetic algorithms. They discuss mainly the distribution of the load among the virtual machines. They compare it with FCFS and prove that their method outperforms in terms of QoS.

C. Maintaining the Integrity of the Specifications

The initial stages of the metaheuristic algorithm exhibit divergence, which covers a large search space, and decreases as the solution is near-optimal. Premature and slow convergence[12] are the problems with existing metaheuristic algorithms. The probability of achieving an optimal solution with high diversity is maximum. This high diversity suffers from slow convergence. Contrary to this, the convergence might be fast with a less accurate solution if divergence is less. To enhance the efficiency of the metaheuristic algorithm, it has become a general practice to add two or more metaheuristic algorithms to form a new hybrid algorithm.

Generally, three kinds of combinations [12] are used to hybridize the algorithms. The first type is a mix of population-driven and single solution-based algorithms. Combining two population-based algorithms is the second type, and the combination of metaheuristic and heuristic algorithms is the third type.

In [25] and [26], the authors fused the Genetic algorithm with the Particle swarm optimization algorithm (HGPSO) and the Genetic algorithm with Ant colony optimization (HGA-ACO), respectively. In the former algorithm, the initial population is generated by GA, and the individuals with good fitness are selected as candidates for PSO. In the latter, the efficient pheromone for ant colony optimization is initialized using a genetic algorithm. The ACO is used to improve GA solutions for crossover GA action. The findings of the experiments demonstrate that the suggested system performs well in terms of mission allocation and maintaining service efficiency parameters.

Two-hybrid metaheuristic algorithms have been introduced [27] by Ben Alla, H. et al. PSO, which is hybridized with fuzzy logic, is the first proposed algorithm. Simulated annealing is combined with PSO in the second algorithm. They use Dynamic dispatch queues for these algorithms. Discrete PSO has been combined [28] with a local search in which the authors use hill climbing for the avoidance of local optima. They claim that their algorithm has shown better performance in the minimization of makespan. In [29-32], PSO and fruit fly algorithms (FOA) were merged. The essential parameters, position, and velocity of PSO have been redefined. With the help of a fruit fly smell operator, the issue of prematurity has been resolved.

III. Proposed Work: Improved GRASP Algorithm

Fixed set search and GRASP are combined to make an improvement in the performance of the algorithm to allocate jobs to VMs.

A. General Procedure for GRASP

As GRASP is an iterative process [3], each iteration consists of the construction phase and a local search phase. A feasible solution is built iteratively, one element at a time in the development process. The choice of the next element to be added is decided at each construction iteration by ordering all the elements in a candidate list with respect to a greedy function. The pseudocode for GRASP is presented in Fig. 1 with algorithm 1.

The advantage of choosing each element is calculated. The heuristic is adaptive because, during each iteration of the construction process, the benefit associated with each element is modified to accommodate the improvements made by the previous element's selection.

Algorithm 1. Pseudocode – GRASP

```
while GRASP Stop Criteria not Satisfied do
        create solution Sol using greedy random method
        local search (Sol)
        update if Sol is the new best
end while
```

Fig. 1. Pseudo Code for GRASP.

The "Restricted candidate list (RCL)" [3] is labeled by considering the list of best candidates. This technique makes it possible to obtain new solutions in every iteration of GRASP without compromising the power of adaptive greedy processes.

The procedure for creating the initial population is presented in algorithm 2, Fig. 2, **J** is a set of n jobs represented with J_1, J_2..J_n. In this discussion, tasks and jobs are considered the same for simplicity. VM is a set of virtual machines denoted with VM_1, VM_2, VM_3…VM_n. The greedy adaptive random search procedure is applied to generate an initial population Pop. This procedure is presented in Fig. 2 in algorithm 2.

Algorithm 2: Generate the initial population with GRASP

```
1. J={$J_1$, $J_2$, $J_3$…...$J_n$} is the set of jobs
2. VM= {$VM_1$, $VM_2$,$VM_3$….$VM_n$}
3. Pop ={ }// null
4. while not completed, do
5.          Pop= Pop U Apply GRASP and allocate jobs to VMs
6.          calculate the overall completion time
7. end while
8. $R_{mbest}$= AGRASP( Pop,n)
9. end.
```

Fig. 2. Algorithm to Generate Initial Population and RCL.

The set of solutions generated by GRASP will be sorted according to the overall completion time. RCL is helpful in reducing the search space. Top 'm' best solutions considered, and in the present case, RCL is stored in the $R_{\mathbf{mbest}}$. This procedure is presented in Fig. 3 in algorithm 3.

Algorithm 3: AGRASP

```
1. Algorithm(Pop,n)
2. Sort the jobs in increasing order of execution time
3. temp=Select top 'm' elements from the sorted list of jobs
4. return(temp)
5. end
```

Fig. 3. AGRASP for Best Solutions.

The solution created by GRASP may not be locally optimal. It adds benefits by applying local search. Iteratively, a local search algorithm operates by successively substituting the incumbent solution with a more robust solution in the neighborhood.

The right choice of the neighborhood structure with good neighborhood search techniques and a better initial solution leads to a thriving local search. Exponential time may be required for such a local optimization procedure as it starts arbitrarily. However, efficiency improves significantly with the best initial solution. As it is known that the initial population is generated with greedy random selection in the GRASP algorithm, the algorithm may not be optimal. But with the help of local search like, 2-opt, or 3-opt there can be improvements. The procedure for the local search is shown in the following algorithm 4 in Fig. 4.

Algorithm 4:Procedure for the Local search

```
1. Local search(LS(RCL))
2. Swap two randomly selected allocations.
3. Calculate the overall completion time.
4. If the newly calculated completion time is less than the best
5. best= new best
6. end
```

Fig. 4. Algorithm for Local- Search.

B. Fixed Set Search (F-GRASP)

GRASP algorithm does not incorporate any learning in its iterations [5]. The idea of the addition of "learning" called fixed set search(FSS) was proposed in [5]. This added feature will not affect the simplicity of the GRASP algorithm in both calculations and complexity. This learning is used in this paper to address the scheduling in the Cloud.

To make fixed set search more efficient, two rules are used. First, the solution space can be minimized by fixing certain sections of the solution. Second if a large number of good solutions are considered, there might be some similarities among them. A fixed set is defined as the set created by these standard components. It is possible to discover a near-optimal solution by "filling the gap."

FS represents a fixed set. The set consists of the elements which help to generate the best solutions. The following requirements should be satisfied by the proposed method. First, the engendered fixed set FS should consist of elements from the best solutions. Second, it should be able to generate random fixed sets. In turn, these sets should help to generate high-quality solutions. Third, feasible solutions should be generated from fixed set FS. Fourth, the capability to monitor the number of elements in the fixed set generated should be possible.

The random selection of high-quality solutions can achieve the first and second requirements. Select k random solutions from the set Pop and store in a set $R_{mbest} = \{R_1, R_2, R_3...R_k\}$. The set of edges $Ed=\{ed_{11}, ed_{12},..ed_{1j}, ed_{21}, ed_{2j}...ed_{i1}, ed_{i2}.. ed_{ij}\}, i\epsilon| J |, j \epsilon | VM |$, denotes the solution. The representation ed_{ij} is used to indicate that job 'i' is delegated to VM ' j.' A cost function $C(ed_{i,j},R_{mbest})$ equal to '1' if $ed_{i,j} \epsilon R_{mbest}$ and '0' otherwise. If job 3 is allocated to VM 4, for example, and is present in R_1, R_2, and R_4. The cost function gets calculated as follows.

$$T(ed_{3,4},\{R_1,R_2,R_3,R_4\}) = C(ed_{3,4},R_1) + C(ed_{3,4},R_2) + C(ed_{3,4},R_3) + C(ed_{3,4},R_4).$$

The count is 3.

$$T(ed_{i,j}, R_{mbest}) = \sum_{R_k \epsilon R_{mbest}} C(ed_{i,j}, R_k) \quad (1)$$

The size of the FS has to be adaptable. It will be fixed to a value, and changes made as required. To simplify, Eq. (2) is used.

$$maxsize[\, i \,] = |\, J \,| - \left\lfloor \frac{|J|}{2^i} \right\rfloor \text{ 'i' is the iteration number} \quad (2)$$

The fixed set size is initialized to *maxsize* and changes after each iteration. If the number of jobs is 5, then the size of the fixed set can be considered as 3. This indicates that three assignments from the fixed set with the highest count for edges will be considered.

The notation F-GRASP is considered for fixed set search GRASP. Fig. 5 explains the procedure for finding the best allocation with F-GRASP. The notation Pop_n, R_{mbest} represent the initial population and RCL, respectively.

Algorithm 5. Pseudo-code for the fixed set search

```
1. Popn represents initial population using GRASP with n elements
2. Rmbest ={R1,R2,R3…Rk} where Ri ϵ Popn, i ϵ N,1 ≤ i ≤ k
3. Count= $T(ed_{i,j}, R_{mbest})$ //find the frequency of each edge with Eq. (1)
4. Set FS={ed1(jobi,vmk),ed2(jobi,vmk,…edmaxsize(jobi,vmk }
5. Allocate the jobs to VMs according to FS.
6. Allocate the remaining jobs according to GRASP
7. while stopping criteria not reached do
8.  Apply local search to S
9. end while
```

Fig. 5. Algorithm F-GRASP.

The set FS is used to store the edges with the highest allocation. By considering the fixed set with the highest count, an initial allocation in the solution space is done. The remaining allocation is done with the GRASP. By this, it reduces the number of iterations. After fixing the allocation, the total completion time will be calculated. The swap in the allocation of the jobs is done till there is no improvement in the makespan. The same is explained with an example in section 4.

Table I is considered for the execution times of each job on every VM. J_1, J_2, J_3, J_4, and J_5 are the given jobs. VM_1, VM_2, VM_3, VM_4, and VM_5 are the VMs available for allocation. The challenge is to allocate the jobs to VMs with minimum makespan by the scheduler.

Table II consists of the initial population represented by Pop_n. For example, the representation $J_1 \rightarrow VM_1$, $J_5 \rightarrow VM_2$, $J_1 \rightarrow VM_3$, $J_3 \rightarrow VM_4$, $J_4 \rightarrow VM_5$ considered as one of the allocations.

For each allocation, fitness (total execution time) is calculated and sorted in ascending order of fitness function. Table III holds these values. Top 'm' best allocations considering fitness function are selected.

C. Worked Out Example

Table I is considered for the execution times of each job on every VM. J_1, J_2, J_3, J_4, and J_5 are the given jobs. VM_1, VM_2, VM_3, VM_4, and VM_5 are the VMs available for allocation. The challenge is to allocate the jobs to VMs with minimum makespan by the scheduler.

Table II consists of the initial population represented by Pop_n. For example, the representation $J_1 \rightarrow VM_1$, $J_5 \rightarrow VM_2$, $J_1 \rightarrow VM_3$, $J_3 \rightarrow VM_4$, $J_4 \rightarrow VM_5$ considered as one of the allocations.

For each allocation, fitness (total execution time) is calculated and sorted in ascending order of fitness function. Table III holds these values. Top 'm' best allocations considering fitness function are selected for allocation and presented in Table IV. This list is considered as RCL(Restricted Candidate List). R_{mbest} is the notation used for RCL. The allocation will be done randomly. As an example, an allocation of J_3-J_5-J_4-J_1-J_2 is considered. The execution time of J_3 on VM_1 is 11, J_5 on VM_2 is 10, J_4 on VM_3 is 14, J_1 on VM_4 is 9, J_2 on VM_5 is 9. The overall completion time (11+10+14+9+9) is 53.

By applying a local search, there can be an improvement. However, in the proposed method, to reduce the number of swaps as part of 2-opt, a fixed set is introduced.

Equ. (2) calculates the size of the fixed set—the number of VMs=5. Hence the maxsize=3. From Table IV, allocation with minimum completion time is $J_3 \rightarrow VM_1$, $J_5 \rightarrow VM_2$, $J_4 \rightarrow VM_3$, $J_1 \rightarrow VM_4$, $J_2 \rightarrow VM_5$.

Frequency of the allocation is counted with variable Count. Count(J_3,VM_1) = 1, Count(J_5,VM_2)= 2, Count(J_4,VM_3)= 1, Count (J_1,VM_4) = 1, Count(J_2,VM_5) = 4.From the values, it is evident that allocation of J_5 toVM_2 has a count as 2, and J_2 to VM_5 as 4. As the remaining counts are not considerable, the fixed set holds the two allocations. The fixed set is FS={(J_5,VM_2), (J_2,M_5)}, therefore the new allocation is

{ $J_5 \rightarrow VM_2$, $J_2 \rightarrow M_5$}

TABLE I. EXECUTION TIME OF JOBS ON EACH VM

Execution times of a job on a Virtual machine					
	VM_1	**VM_2**	**VM_3**	**VM_4**	**VM_5**
J_1	13	10	18	9	13
J_2	19	18	15	11	9
J_3	11	15	12	10	18
J_4	11	15	14	11	19
J_5	10	10	13	11	14

TABLE II. INITIAL POPULATION

Initial Population Pop_n				
VM_1	**VM_2**	**VM_3**	**VM_4**	**VM_5**
J_2	J_5	J_1	J_3	J_4
J_3	J_4	J_5	J_2	J_1
J_4	J_5	J_1	J_3	J_2
J_1	J_2	J_3	J_5	J_4
J_4	J_3	J_1	J_2	J_5
J_1	J_4	J_3	J_5	J_2
J_3	J_5	J_4	J_1	J_2
J_5	J_3	J_2	J_4	J_1
J_1	J_3	J_5	J_4	J_2
J_4	J_2	J_1	J_5	J_3
J_1	J_2	J_5	J_3	J_4
J_4	J_5	J_3	J_1	J_2
J_4	J_2	J_5	J_3	J_1
J_2	J_5	J_4	J_3	J_1
J_2	J_4	J_3	J_5	J_1

TABLE III. SORTED LIST OF VMS

Sorted list of allocation of jobs to VMs Pop_n					**Total execution time**
J_3	J_5	J_4	J_1	J_2	53
J_4	J_5	J_1	J_3	J_2	58
J_1	J_4	J_3	J_5	J_2	60
J_1	J_3	J_5	J_4	J_2	61
J_3	J_4	J_5	J_2	J_1	63
J_5	J_3	J_2	J_4	J_1	64
J_4	J_2	J_5	J_3	J_1	65
J_2	J_5	J_4	J_3	J_1	66
J_4	J_3	J_1	J_2	J_5	69
J_2	J_4	J_3	J_5	J_1	70
J_1	J_2	J_3	J_5	J_4	73
J_1	J_2	J_5	J_3	J_4	73
J_2	J_5	J_1	J_3	J_4	76
J_4	J_2	J_1	J_5	J_3	76

TABLE IV. SELECTION OF BEST CANDIDATES

R_{mbest}= Best Candidates selected from Pop_n					
J_3	J_5	J_4	J_1	J_2	53
J_4	J_5	J_1	J_3	J_2	58
J_1	J_4	J_3	J_5	J_2	60
J_1	J_3	J_5	J_4	J_2	61
J_3	J_4	J_5	J_2	J_1	63

The greedy random method can be applied to the remaining. For VM_1 the jobs J_1, J_3, and J_4 are the choices. As J_3 and J_4 are the same, VM1 decisions cannot be taken. Move on to the next VM, i.e., on to VM_3. J3's execution time is minimum on VM_3. Based on this, J_3 is allocated to VM_3. VM_4 is left with J_1 and J_4. Here, J_1 having less execution time, hence assigned to VM_4. VM_1 will be allocated with J_4. VM_1 can be allocated either with J_3 or J_4 as they both have equal values. Here, J_3 is allocated to VM_3. And VM_1 is left with J_4 and is allocated. The overall completion time is 51, which is the newly updated value. The best solution is shown in Table V.

TABLE V. FINAL ALLOCATION

VM_1	VM_2	VM_3	VM_4	VM_5	Completion time
J_4	J_5	J_3	J_1	J_2	51

IV. RESULT

The proposed algorithm is implemented in MATLAB R2020a. Computations are performed on a PC with Intel core™ i7 CPU@1.80-GHz with 8 GB of RAM.The comparison is done among three algorithms. The Genetic algorithm(GA), Fixed set search-GRASP from now considered as (F-GRASP), GRASP are chosen for comparison. The overall completion (makespan) time is calculated. The allocation with minimum overall completion time is considered as the best allocation. However, as the scheduling is NP-complete, the near-optimal allocation changes in each run. With 10 jobs, and in 100 iterations, the best makespan with the algorithms GA=140, F-GRASP=148, GRASP=160, with 200 iterations GA=134, F-GRASP=132, GRASP=133, 300 iterations GA=133, F-GRASP=130, GRASP=135, 400 iterations GA=133, F-GRASP=130, GRASP=132, and after 500 iterations GA=132 F-GRASP=129 GRASP=131, GA=132, F-GRASP= 130 GRASP=130. The results show that the proposed algorithm is equally competing with existing metaheuristic algorithms like the Genetic algorithm and GRASP. In some instances, it is showing better results than the algorithms with which it has been compared.

The usage of fixed set search reduces the search space. Thus it converges with the near-optimal solution faster than the other two algorithms. Fig. 6. represents the number of iterations on the X coordinate and best makespan on the Y coordinate. F-GRASP shows promising results with the Genetic algorithm and GRASP.

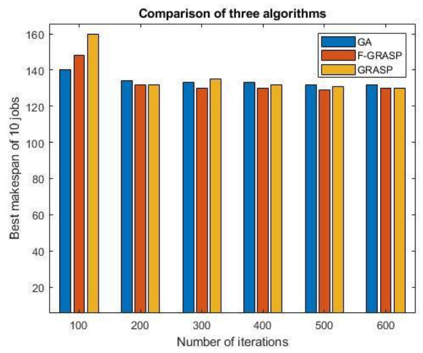

Fig. 6. Comparison of Makespan of F-GRASP with GA and GRASP.

V. CONCLUSION AND FUTURE WORK

This paper discusses the limitations of the GRASP algorithm. Learning is added to improve the efficiency of the algorithm. With the inclusion of a fixed set search, the learning is accomplished. The algorithm's search space reduces by accumulating the elements of high-quality solutions. The algorithm starts with a greedy random approach, and each iteration shows some improvement and finally reaches an optimal solution. The algorithm shows remarkable improvement in performance. While the addition of fixed set search and the 2-Opt algorithm strengthens the algorithm significantly, there is still space to test with 3-Opt or 4-Opt algorithms. The proposed algorithm is evaluated using MATLAB. The time complexity is $O(2^n\ n^2)$ and space complexity is $O(n^2)$. Alternative methods can be explored to reduce the time complexity. The open-source cloud platforms such as Open stack or Cloud stack by interested researchers with the proposed algorithm.

REFERENCES

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

[2] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra and B. Hu, "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends," 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, 2015, pp. 621-628, doi: 10.1109/CLOUD. 2015.88.

[3] Feo, T.A., Resende, MGC Greedy Randomized Adaptive Search Procedures. JGlobOptim 6, 109–133(1995). https://doi.org/10.1007/ BF01096763.

[4] J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. Operations Research Letters, 6:107–114, 1987.

[5] Jovanovic R., Tuba M., Voß S. (2019) Fixed Set Search Applied to the Traveling Salesman Problem. In: Blesa Aguilera M., Blum C., Gambini Santos H., Pinacho-Davidson P., Godoy del Campo J. (eds) Hybrid Metaheuristics. HM 2019. Lecture Notes in Computer Science, vol 11299. Springer, Cham. https://doi.org/10.1007/978-3-030-05983-5_5.

[6] Zhu, M., Chen, J.: Computational comparison of GRASP and DCTSP methods for the Traveling Salesman Problem, pp. 1044–1048 (2017).

[7] Festa, P., Resende, MGC: Hybridizations of GRASP with path-relinking.Talbi, E.G. (ed.) Hybrid Metaheuristics. Studies in Computational Intelligence,vol. 434, pp. 135–155. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30671-6 5.

[8] De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross entropy method. Ann. Oper. Res. 134(1), 19–67 (2005).

[9] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," Naval Res. Logistics Quart., vol. 1, no. 1, pp. 61–68, 1954.

[10] S. Dubey, V. Jain and S. Shrivastava, "An innovative approach for scheduling of tasks in cloud environment," 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, 2013, pp. 1-8, doi: 10.1109/ICCCNT.2013.6726727.

[11] E. Madhukar and T. Ragunathan, "Dynamic and Static Characteristics Based Algorithm to Allocate VMs to Jobs in the Cloud," in 2016 International Conference on Information Technology (ICIT), Bhubaneswar, 2016 pp. 81-86.doi: 10.1109/ICIT.2016.028.

[12] Metaheuristic Scheduling for Cloud: A Survey Chun-Wei Tsai and Joel J. P. C. Rodrigues Senior Member, IEEE.

[13] A. Makwe and P. Kanungo, "Scheduling in cloud computing environment using analytic hierarchy process model," 2015 International Conference on Computer, Communication and Control (IC4), Indore, 2015, pp. 1-4, doi: 10.1109/IC4.2015.7375723.

[14] Mohamed Abdel-Basset, Laila Abdel-Fatah, Arun Kumar Sangaiah, Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review, Editor(s): Arun Kumar Sangaiah, Michael Sheng, Zhiyong Zhang, In Intelligent Data-Centric Systems, Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Academic Press, 2018.

[15] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," 2013 8th International Conference on Computer Engineering & Systems (ICCES), Cairo, 2013, pp. 64-69, doi: 10.1109/ICCES.2013.6707172.

[16] YoungJu Moon, HeonChang Yu, Joon_Min Gil & JongBeom Lim. A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Hum. Cent. Comput. Inf. Sci. 7, 28 (2017). https://doi.org/10.1186/s13673-017-0109-2.

[17] Z. Chen et al., "Multi-objective Cloud Workflow Scheduling: A Multiple Populations Ant Colony System Approach," in IEEE Transactions on Cybernetics, vol. 49, no. 8, pp. 2912-2926, Aug. 2019, doi: 10.1109/TCYB.2018.2832640.

[18] Zhou, Z., Li, F., Zhu, H. et al. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput & Applic 32, 1531–1541 (2020). https://doi.org/10.1007/s00521-019-04119-7.

[19] Jinn-Tsong Tsai, Jia-Cen Fang, Jyh-Horng Chou " Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm " https://doi.org/10.1016/j.cor.2013.06.012.

[20] M.A. Elaziz, S. Xiong, K.P.N. Jayasena, et al., Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.01.023.

[21] Y. Liang, A. H. Chen and Y. Nien, "Artificial Bee Colony for workflow scheduling," 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 558-564, doi: 10.1109/CEC.2014.6900537.

[22] Thanka, MR, Uma Maheswari, P. & Edwin, E.B. An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment. Cluster Comput 22, 10905–10913 (2019). https://doi.org/10.1007/s10586-017-1223-7.

[23] Sreenu, K., Sreelatha, M. W-Scheduler: whale optimization for task scheduling in cloud computing. Cluster Comput 22, 1087–1098 (2019). https://doi.org/10.1007/s10586-017-1055-5.

[24] Singh, P., Dutta, M. & Aggarwal, N. A review of task scheduling based on meta-heuristics approach in cloud computing. Knowl Inf Syst 52, 1–51 (2017). https://doi.org/10.1007/s10115-017-1044-2.

[25] Senthil Kumar, A.M., Venkatesan, M. Task scheduling in a cloud computing environment using HGPSO algorithm. Cluster Comput 22, 2179–2185 (2019). https://doi.org/10.1007/s10586-018-2515-2.

[26] Senthil Kumar, A.M., Venkatesan, M. Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment. Wireless Pers Commun 107, 1835–1848 (2019). https://doi.org/10.1007/s11277-019-06360-8.

[27] Ben Alla, H., Ben Alla, S., Touhafi, A. et al. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. Cluster Comput 21, 1797–1820 (2018). https://doi.org/10.1007/s10586-018-2811-x.

[28] Mirsaeid Hosseini Shirvani, A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems, Engineering Applications of Artificial Intelligence, Volume 90, 2020, 103501, ISSN 0952-1976, https://doi.org/10.1016/j.engappai. 2020.103501.

[29] Bhushan, S. B., & Reddy, P. C. (2018). A Hybrid Meta-Heuristic Approach for QoS-Aware Cloud Service Composition. International Journal of Web Services Research (IJWSR), 15(2), 1-20. doi:10.4018/IJWSR.2018040101.

[30] Ayaluri MR, K. SR, Konda SR, Chidirala SR. 2021. Efficient steganalysis using convolutional auto encoder network to ensure original image quality. PeerJ Computer Science 7:e356 https://doi.org/10.7717/peerj-cs.356.

[31] A. M. Reddy, V. V. Krishna, L. Sumalatha and S. K. Niranjan, "Facial recognition based on straight angle fuzzy texture unit matrix," 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), Chirala, 2017, pp. 366-372, doi: 10.1109/ICBDACI.2017.8070865.

[32] Ilaiah Kavati, A. Mallikarjuna Reddy, E. Suresh Babu, K. Sudheer Reddy, Ramalinga Swamy Cheruku, Design of a fingerprint template protection scheme using elliptical structures, ICT Express, 2021, ISSN 2405-9595,https://doi.org/10.1016/j.icte.2021.04.001.