# Optimization of Task Scheduling in Cloud Computing Environments

## Introduction

Over the period in past few years, cloud computing has been changing in order to traditional cloud computing by providing benefits like on-demand services and broad access mobile services. Scheduling provides optimal allocation of resources among given tasks in a finite time to achieve a desired quality of service. The aim of the scheduling process to build a algorithm that specifies when and on which resource each task will be executed. In recent years, distributed computing has gained much attention due to high scalability, reliability, information sharing and low-cost than single processors standalone machines. Cloud computing has emerged the most popular distributed computing paradigm out of all other in the current environment. Cloud computing guarantees quality of service (QoS) to the users. To provide Quality of Service to the users, it is very important that jobs should be efficiently allocated to given resources. If the desired performance is not achieved, the users will hesitate to pay. Therefore the scheduling process is considered as the most important part in the cloud computing systems.

The problems of mapping tasks on unlimited computing resources in cloud computing it is known as a category of problems called as NP-hard problems. There are no algorithm is made to give optimal solution to a task allocation and scheduling problem within a polynomial time. Solution based on a very large search are not feasible because the cost of generating scheduling is very high. Metaheuristic techniques like MBO, ACO and PSO deals with these problems by providing near optimal solution within a reasonable time. These techniques has gained huge popularity due to its effectiveness to solve large complex problems in a short span of time.

In this paper, an extensive review is presented on two metaheuristic techniques namely Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Monarch Butterfly Optimization (MBO). Scheduling algorithms differs on dependency among the tasks to be scheduled. If there is some precedence in the existing tasks, the task only can be scheduled after the parent task is completed, but in this case tasks are independent to each other, they can be scheduled in any sequence.

# Chapter 1

## 1.1 Cloud Computing

Cloud computing is an approach that aims to increase capacity and capabilities in Information Technology networks by centralizing how the data is processed. It allows users to access various cloud based application without first installing them and increases access to personal information over the internet. Cloud computing has led to reduced costs of building IT infrastructure and acquiring new resources. Cloud services are defined by five essential characteristics which include:

- **On-Demand self-service:** A cloud computing service users get computing capabilities such as virtual machines time for processing and storage tasks whenever necessary to the users without need of the attention of each cloud provider.
- **Broad network access:** The cloud computing service users can access the cloud resources online by using various devices like workstation, laptops, smartphones that are connected to internet.
- **Resource pooling:** Multiple cloud users can share the cloud service provider's resources like cloud resources, that includes processing power, storage, network bandwidth and memory.
- **Rapid elasticity:** Resources are elastically allocated or released automatically or manually depending upon the consumer's needs. The cloud resources capabilities are often unlimited can be used anytime.
- **Measured service:** Cloud service providers can charge on consumers based on a pay as you go for pricing model. The cloud computing service usage is monitored and reported in real-time which ensured transparency between the cloud service providers and the users.

## 1.2 Cloud service models

- **Software as a Service (SaaS):** This refers to a service provided by a cloud application supported on a cloud infrastructure. Consumers can interact with the application through a user interface like a Web browser installed on different client devices. Consumers do not need to manage or manipulate any underlying cloud infrastructure such as servers or operating systems. However, they may be given limited control over particular application configuration settings.

- **Platform as a Service (PaaS):** PaaS is another type of service which is built on the cloud infrastructure and involves developers acquiring applications and platforms which include tools, libraries and programming languages supported by the cloud service provider. The developer is not required to manage the underlying infrastructure such as network, server, storage except managing the deployed applications and settings of the environment hosting those.

- **Infrastructure as a Service (IaaS):** This provides the cloud computing service user access to processing, network, storage, and other basic computing resources. IaaS also allows users to develop and execute software as well as operating systems and applications. The cloud computing service user is not required to manage the cloud infrastructure but still has control over the operating systems, applications and other resources like storage and networks applications, for example firewalls

## 1.3 Cloud services Deployment models

According to the NIST definition of cloud computing, cloud services can be deployed using four development models including: (i) Private cloud, (ii) Community cloud, (iii) Public cloud, and (iv) Hybrid cloud as shown in Fig. 1. Private cloud: In this deployment model, a particular organization owns a datacenter regardless if it is managed by a third party or by itself. This cloud is usually for exclusive use and may be located onsite or offsite.

- **Community cloud:** In this deployment model, the datacenter is shared among one or more organizations in the community regardless if it is managed by a third party or one of the organizations owning it. This cloud is also for exclusive use and may be located onsite or offsite.

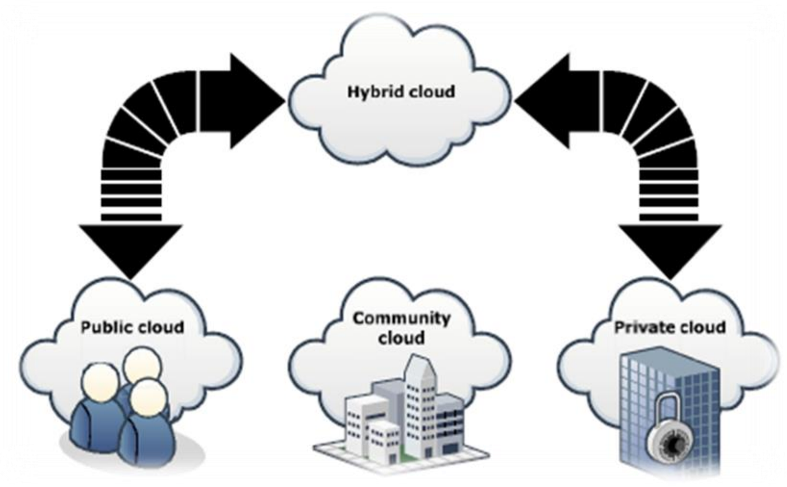


**Fig. 1.** Cloud services deployment models

- **Public cloud:** In this deployment model, the datacenter can be used by any cloud computing service user (e.g., a single user, research laboratory, company, or all of them together). This cloud is located at the same site as the cloud service provider.

- **Hybrid cloud:** In this deployment model, two or more of the previously mentioned deployment models are used in a particular datacenter. This deployment model requires standards or patented technology which allows the portability of data and applications.

# Chapter 2

## 1. Optimization metrics

There are mainly two types of entities involved in cloud: One is cloud service provider and another is cloud consumer. Cloud service providers provide their resources on rental basis to cloud consumers and cloud consumers submit their tasks for processing to these resources. They both have their own motivations when they become part of cloud environment. Consumers are concerned with the performance of their applications, whereas providers are more interested in efficient utilization of their resources. These rationales are articulated as objective functions/optimization criteria while scheduling consumer tasks on resources. Thus these optimization metrics can be classified into two types: Consumer-Desired and Provider-Desired. Following are some of the Consumer-Desired and Provider-Desired optimization criteria [5] while scheduling tasks in grid or cloud environment

## 2.1. Consumer-Desired

**Makespan:** Makespan indicates the finishing time of the last task. The most popular optimization factor while scheduling tasks is minimization of makespan as most of the users' desire fastest execution of their application.

$$Makespan = max_{i} \in tasks\{f_{i}\}$$

Where Fi denotes the finishing time of i.

Fig. 2 shows an example schedule having nine independent tasks scheduled on two resources.

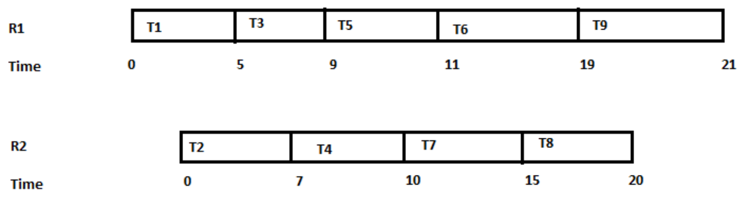


**Figure 2**

Makespan =21 time units (finishing time of the last task i.e. T9)

- **Economic cost:** It indicates the total amount the user needs to pay to service provider for resource utilization.

$$Economic\ cost = \sum_{i \in resources} \{C_i * T_i\}$$

where C*i* denotes the cost of resource *i* per unit time and *Ti* denotes the time for which resource *i* is utilized. In the example shown in Fig. 2, if the cost of using resource R1 is 1000 per time unit and R2 is 2500 per time unit, then economic cost = 1000 * 21 +2500 * 20= 470,000.

- **Fairness:** A desirable characteristic of scheduling process is fairness which requires that every task must get equal share of CPU time and no task should be starved.

## 2.2 Provider Desired

- **Resource utilization:** Another important criterion is maximization of resource utilization i.e. keeping resources as busy as possible. This criterion is gaining significance as service providers want to earn maximum profit by renting limited number of resources.

$$Average\ resource\ utilization = \frac{\sum_{i=1}^{n} \text{time taken by resource i to finish all the job}}{Makespan * n}$$

where *n* is no. of resources.

For the example in Fig. 2, Average Resource Utilization = (21+ 20)/21 * 2= 3.90

- **Throughput:** It is defined as the total number of jobs completing execution per unit time.

Generally, a task-resource mapping schedule is optimized on the basis of single or multiple criterions. Scheduling of tasks considering single optimization criteria like minimization of makespan is simpler to implement than multi-criteria scheduling, specifically when the conflicting criteria are considered like minimizing makespan and cost. If a user wants to execute his job faster, he has to spend more because faster resources are usually costlier. So, there is always a trade-off between cost and execution time optimization.

Also in multi-criteria optimizations [6], it is not feasible to find an optimal schedule with respect to all the defined criterions. Generally, in multi-criteria based scheduling, one criterion is identified for optimization and for other criterion, minimization constraints are established.

Optimization always has an objective and a constraint associated with it. The objective defines the best possible option whereas the constraint defines the restriction imposed. So both optimization objective and constraints are related to each other. Following are a few common constraints considered while scheduling:

- **Priority constraint:** It represents the urgency of a task to complete at earliest. Priority can be decided on the basis of deadline of a task, arrival time of a task or advance reservation. The tasks with shorter deadlines can be given higher priority and scheduled first. Similarly the tasks having advance reservation of resources can be provided with those resources prior to others.

- **Dependency constraint:** It represents the sequence of tasks based on their dependency. If there are precedence orders among tasks, then a task cannot be scheduled until all its parent tasks are finished, unlike independent tasks, where tasks are independent of each other and can be scheduled in any sequence.

- **Deadline constraint:** This represents the time till which the task or the batch of tasks should be finished.

- **Budget constraint:** This represents the restriction on the total cost of executing all tasks.

# Chapter 3

# The Scheduling Algorithms

## 3.1: Particle Swarm Optimization

### History of PSO

PSO has been proposed by Eberhart and Kennedy in 1995, subsequently developed in thousands of scientific papers, and applied to many diverse problems, for instance neural networks training, data mining, signal processing, and optimal design of experiments.

### Basic description of PSO

PSO is a swarm intelligence meta-heuristic inspired by the group behavior of animals, for example bird flocks or fish schools. Similarly to genetic algorithms (GAs), it is a population-based method, that is, it represents the state of the algorithm by a population, which is iteratively modified until a termination criterion is satisfied. In PSO algorithms, the population P={p1,…,pn} of the feasible solutions is often called a swarm. The feasible solutions p1,…,pn are called particles. The PSO method views the set Rd of feasible solutions as a “space” where the particles “move”. For solving practical problems, the number of particles is usually chosen between 10 and 50.

### The purpose of PSO

The usual aim of the particle swarm optimization (PSO) algorithm is to solve an unconstrained minimization problem: find x* such that f(x*) <=f(x) for all d-dimensional real vectors x. The objective function f: Rd -> R is called the fitness function.

**Swarm topology**

Each particle i has its neighborhood Ni (a subset of P). The structure of the neighborhoods is called the swarm topology, which can be represented by a graph. Usual topologies are: fully connected topology and circle topology.

**Stopping rule**

The algorithm is terminated after a given number of iterations, or once the fitness values of the particles (or the particles themselves) are close enough in some sense.

**The Scheduling System**

Figure 1 illustrates an overview of the scheduling system. The system consists of three modules, the first module is the application which represents the set of the cloudlets (tasks). The second module is the Mapping Algorithms (MA) which estimates the expected time for each cloudlet to allocate on each virtual machine, and it is assumed that these values are available to the scheduler. A third module is a virtual machine (VMs) which used to execute the cloudlets. The cloudlets expected time have been stored in an m × n matrix, where m is the number of virtual machines, and n is the number of cloudlets. Obviously, n/m will generally be greater than 1, with more cloudlets than virtual machines, so that some machines will need to be assigned multiple cloudlets. The Estimated Running Time (ERT) is defined as the time of executing task j on resource r [21]. Each column of the expected running times (ERT) matrix contains the expected running time (ERT) of each cloudlet j on machine *i*.

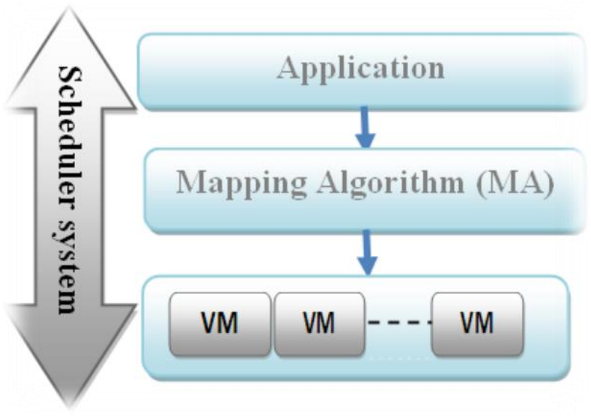


**Figure 3.Schedulingsystem**

The main objective of allocating tasks on virtual machines is to reduce the makespan. The makespan of a task is defined as the overall task completion time. We denote completion time of task T*i* on VM*j* asCT*ij*. Hence, the makespan is defined using the following equation

$$Makespan = \mathrm{CT}max\ [i,j] \mid i \in T,\ i = 1,2,\ \ldots\ n\ and\ j \in VM,\ j = 1,2,\ldots m \quad (1)$$

Where CT*max* [*i*,*j*] is the maximum completion time of task i on a VM *j*., and n, m are the number of tasks and virtual machines respectively.

Let VM=VM1,VM2,….VM*m* be the number of *m* virtual machines that must be processed *n* tasks represented by the group T=T1,T2,….T*n* . The virtual machines are parallel and independent, and the schedule allocates independent tasks to these VM*s*. Also, the Processing a task on a virtual machine cannot interrupt (i.e.) Non-preemption. We denote end time of a task T*i* by CT*j*. The aim of the proposed algorithms is to reduce the makespan which can be denoted as CT*max* . The run time of each task for each virtual machine must be calculated for the purpose of scheduling, If the processing speed of a virtual machine VM*j* is PS*j*, then the processing time for task P*i* can be calculated by equation.

$$\mathrm{T}ij = \mathrm{C}i/\mathrm{PS}j \quad (2)$$

Where P*ij* is the processing time of task Pi by virtual machine VM *j* and C*I* is the computational complexity of the task P*i*. The processing time P *ij* of each task P*i* on VM *j* are stored in the runtime matrix. The processing time of each task in the virtual machines can be calculated by equation (3):

$$\mathrm{Pij} = \sum_{i=1}^{n} Pij \quad (3)$$

According to (1), (2) and (3), the task scheduling algorithm should satisfy the following equation

$$\sum_{i=1}^{n} Pij \leq \mathrm{CT}max \quad (4)$$

By considering the load balancing, the tasks will be transferred from one VM to other to reduce CT*max*, as well as, response time. The processing time of a task varies from one VM to another based on the speed of the virtual machines. In case of transferring, the completion time of a task may vary because of load balancing, optimally.

The main objective of the proposed task scheduling PSO algorithm is that the tasks should be allocated on the virtual machine in order to minimize the makespan and maximize the resource utilization.

## Procedure PSO

1. **Initialization:** Initialize position vector and velocity vector of each particle.

2. **Conversion to discrete vector:** Convert the continuous position vector to discrete vector.

3. **Fitness:** Calculate the fitness value of each particle using fitness function.

4. **Calculating pbest:** Each particle's pbest is assigned its best position value till now. If particle's current fitness value is better than particle's pbest, then replace pbest with current position value.

5. **Calculating gbest:** Select the particle with best fitness value from all particles as gbest.

6. **Updation:** Update each particle's position vector and velocity vector using following equations:

Vi+1=ωVi+c1rand1*(pbest-xi) + c2rand2*(gbest-xi)

Xi+1=Xi+Vi+1

Where

ω=inertia

c1, c2=acceleration coefficients

rand1, rand2=uniformly distributed random numbers and ε [0, 1]

pbest=best position of each particle

gbest= best position of entire particles in a population

i=iteration

7. Repeat steps 2 to 6 until stopping condition is met. Stopping condition may be the maximum number of Iterations or no change in fitness value of particles for consecutive iterations.

8. **Output:** Print best particle as the final solution.

## 3.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a global optimization algorithm, inspired by the social behavior of ants. ACO can be efficiently used to solve graph based optimization problems. Explore ACO and implement an ACO-based solution to the Traveling Salesman Problem. The problem is as follows: Given a list of cities and the distances between each pair of cities, find the shortest possible tour (a route that visits each city exactly once and returns to the origin city).

ACO is a method that has been suggested since the early nineties but was first formally proposed and put forward in a thesis by Belgian researcher Marco Dorigo and Luca Maria Gambardella in 1992, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem and followed up by Dorigo, Birattari, and Stutzle's thesis in 2006, Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique.

When humans reached a standstill in devising mathematical algorithms for computation of various problems, they began looking at the most complex system they know; nature itself. There are a number of different phenomena in nature that are incredibly complex but occur as if automated, and these can be integrated into algorithms to solve problems which were much tougher or even impossible to solve otherwise. This movement spawned the birth of Natural or Bio-Inspired Computing, Genetic Algorithms, Swarm Intelligence and many, many more.

Ants are very small organisms, but are among the most industrious and organized creatures in the whole world. Their communication techniques, planning, and behavior in a number of situations can be studied and modeled to devise new algorithms to solve a variety of problems, and this is the base of Ant Colony Optimization.

**Swarm Intelligence:** is a problem solving approach inspired by social behaviors of animals and interests. It is defined as "the collective behavior of decentralized, self-organized systems, natural or artificial," and finds application in a number of areas, most recently artificial intelligence.

**Stigmergy:** as defined by entomologist P. Grasse is the communication system employed by ants or termites that triggers actions as responses to significant chemically-induced stimuli. He went on to say that "workers are stimulated by performance they have achieved." It is a local and indirect form of communication that depends on the modification of the environment rather than direct exchange of messages. This is done by leaving a trail of chemicals known as pheromones which are bio receptors that can be detected by other ants.

**The Ant colony Procedure:**

The bulk of the ant colony optimization algorithm is made up of only a few steps. First, each ant in the colony constructs a solution based on previously deposited pheromone trails. Next ants will lay pheromone trails on the components of their chosen solution, depending on the solution's quality. In the example of the traveling salesman problem this would be the edges (or the paths between the cities). Finally, after all ants have finished constructing a solution and laying their pheromone trails, pheromone is evaporated from each component depending on the pheromone evaporation rate**.**

**Pheromone Update:** Similar to ants, the algorithm too will have to deposit pheromones over the traversed trails to ensure that they are preferred in the future as well. There are two parts to this, first being Q- the amount of pheromone deposited along each trail, a constant. The second is Ck or the cost of a solution.

**Evaporation:** An important part of updation is also the evaporation of pheromones. Since they are only chemicals, they are not permanent and tend to fade with time. Pheromone evaporation makes the simulation more realistic and also allows for selection of newer paths, much like induced randomness. For this, an evaporation factor is taken as a parameter, and the new Δvalue is simply the old value multiplied by (1- the evaporation factor).

**Procedure ACO**

1. **Initialization:**
    - *i.* Initialize the pheromone value to a positive constant for each path between tasks and resources
    - *ii.* Optimal solution=null
    - *iii.* Place the m ants on random resources
2. **Solution Construction of each ant:**

Repeat for each ant

- *i.* Put the starting resource in tabu list of this ant (for the first task).
- *ii.* For all the remaining tasks
    - *a.* Choose the next resource rj for the next task ti by applying following transition rule

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \epsilon allowed}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}$$

if j ε allowed, allowed means not in tabu list

else 0

b. Put the selected resource in previous step into tabu list of this ant End For

Until each ant builds its solution

3 **Fitness:** Compute the fitness value of the solution of each ant

4 **Replacement:** Replace the Optimal solution with the ant's solution having best fitness value if its fitness value is better than optimal solution.

5 **Pheromone Updation:**

   a. Update local pheromone for each edge

   b. Update global pheromone

6 Empty tabu lists of all ants

7 Repeat steps 2 to 6 until stopping condition is met. Stopping condition may be the maximum number of iterations or no change in fitness value of ants' solutions in consecutive iterations

8. **Output:** Print optimal solution

## 3.3 Monarch butterfly optimization

In nature, the eastern North American monarch population is known for its southward migration during the late summer/autumn from the northern USA and southern Canada to Mexico, covering thousands of miles. By simplifying and idealizing the migration of monarch butterflies, a new kind of nature-inspired metaheuristic algorithm, called monarch butterfly optimization (MBO).

As one of the most familiar North American butterflies, the monarch butterfly has an orange and black pattern that can be easily recognized. It is a milkweed butterfly in the family Nymphalidae. Female and male monarchs have different wings that can be used to identify them. The eastern North American monarch is known for its ability of migrating by flying thousands of miles from the USA and southern Canada to Mexico every summer. It involves which flying over west of the Rocky Mountains to California. In order to overwinter, they move thousands of miles to Mexico. Southward movements commence in August and end at the first frost. However, during the spring, opposite things happen. The female ones lay eggs for generating offspring during these movements. Recent research shows some butterflies perform Levy flight when they migrate or move

## 3.31 MBO

In order to make the migration behavior of monarch butterflies address various optimization problems, the migration behavior of monarch butterflies can be idealized into the following rules.

**1.** All the monarch butterflies(jobs) are only located in Cloud 1 or Cloud 2. That is to say, monarch butterflies in Land 1 and Land 2 make up the whole monarch butterfly population.

**2.** Each child monarch butterfly individual is generated by migration operator from monarch butterfly(job) in Cloud 1 or in Cloud 2.

**3.** In order to keep the population unchanged, an old monarch butterfly will pass away once a child is generated. In the MBO method, this can be performed by replacing its parent with newly generated one if it has better fitness as compared to its parent. On the other hand, the newly generated one is liable to be discarded if it does not exhibit better fitness with respect to its parent. Under this scenario, the parent is kept intact and undestroyed.

**4.** The monarch butterfly individuals with the best fitness moves automatically to the next generation, and they cannot be changed by any operators. This can guarantee that the quality or the effectiveness of the monarch butterfly population will never deteriorate with the increment of generations. The next subsections will present a snapshot of the migration operator and butterfly adjusting operator.

## 3.32 Schematic presentation of MBO algorithm

By idealizing the migration behavior of the monarch butterfly individuals(Tasks), MBO method can be formed, and its schematic description can be given as shown in Algorithm 3. A brief presentation of the MBO algorithm is shown in Fig. 1. According to Algorithm 3, firstly, all the parameters are initialized followed by the generation of initial population and evaluation of the same by means of its fitness function. Subsequently, the positions of all monarch butterflies are updated step by step until certain conditions are satisfied. It should be mentioned that, in order to make the population fixed and reduce fitness evaluations, the number of monarch butterflies, generated by migration operator and butterfly adjusting operator, are NP1 and NP2, respectively.

## Monarch Butterfly Optimization algorithm

```
Begin

    Step 1: Initialization. Set the generation counter $t = 1$; initialize the population P of $NP$
            monarch butterfly individuals randomly; set the maximum generation $MaxGen$,
            monarch butterfly number $NP1$ in Land 1 and monarch butterfly number $NP2$
            in Land 2,max step $SMax$, butterfly adjusting rate $BAR$, migration period $peri$,
            and the migration ratio $p$.

    Step 2: Fitness evaluation. Evaluate each monarch butterfly according to its position.

    Step 3: While the best solution is not found or $t < MaxGen$ do
              Sort all the monarch butterfly individuals according to their fitness. Divide
              monarch butterfly individuals into two subpopulations (Land 1 and Land 2);
              for $i$= 1 to $NP1$ (for all monarch butterflies in Subpopulation 1) do
              Generate new Subpopulation 1 according to Algorithm 1.
            end for $i$

            for $j$= 1 to $NP2$ (for all monarch butterflies in Subpopulation 2) do
                Generate new Subpopulation 2 according to Algorithm 2.

            end for $j$

            Combine the two newly-generated subpopulations into one whole population;
            Evaluate the population according to the newly updated positions;
            $t = t+1$.

            Step 4: end while

            Step 5: Output the best solution.

            End.
```

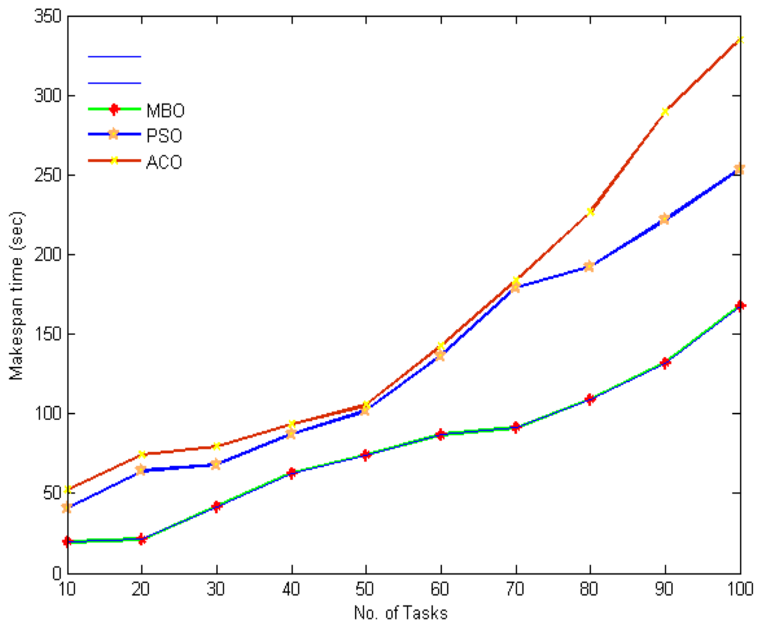
# Chapter 4

## Results and Output

Having coded project, we must now observe and verify the output. Here are few situations.

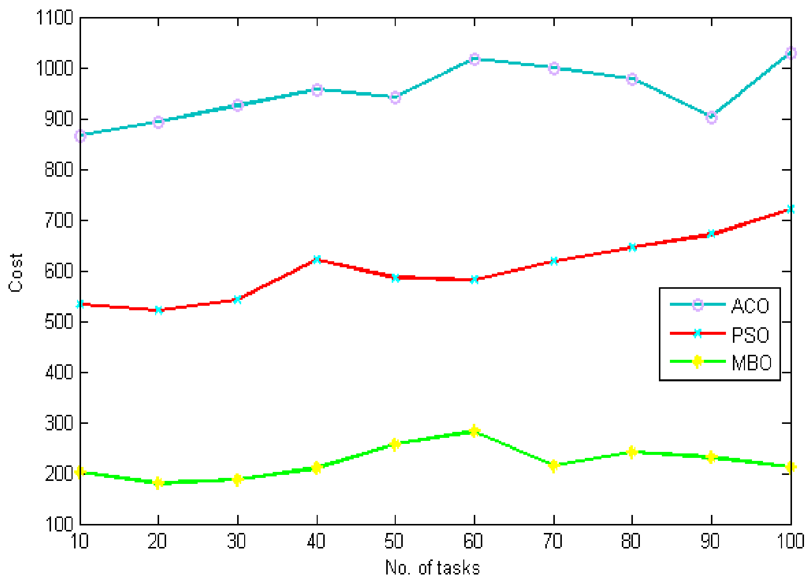
### Comparision of Makespan time

| No.of Tasks | Makespan Time (sec) | | |
|---|---|---|---|
| | **MBO[Proposed]** | **PSO** | **ACO** |
| 10 | 19.6 | 40.7 | 52.2 |
| 20 | 21.2 | 64.1 | 74.1 |
| 30 | 41.6 | 68.2 | 79.6 |
| 40 | 62.9 | 87.4 | 93.3 |
| 50 | 74.2 | 101.9 | 105.4 |
| 60 | 86.7 | 136.2 | 142.3 |
| 70 | 91.4 | 179.1 | 183.3 |
| 80 | 109.2 | 192.2 | 226.2 |
| 90 | 131.7 | 221.7 | 289.1 |
| 100 | 167.5 | 252.5 | 334.6 |



**Comparision of Cost**

| No.of Tasks | Cost | | |
|---|---|---|---|
| | **MBO[Proposed]** | **PSO** | **ACO** |
| **10** | 203 | 532 | 865 |
| **20** | 180 | 520 | 893 |
| **30** | 187 | 542 | 925 |
| **40** | 210 | 620 | 957 |
| **50** | 257 | 585 | 942 |
| **60** | 282 | 581 | 1017 |
| **70** | 215 | 618 | 1000 |
| **80** | 242 | 645 | 978 |
| **90** | 231 | 671 | 902 |
| **100** | 212 | 720 | 1130 |



**Conclusion:**

From the values of output from the above table, I have obtained that the makespan time is minimum for the Monarch Butterfly Optimization than the makespan time and cost of Ant Colony Optimization and Particle Swarm Optimization.

**Future Work:**

- In future work Reduce the Makespan time and Cost with the algorithms to get more optimized results.
- Implement the algorithm for other optimization factors like, Flow time, Tardiness etc.
- Implementing Hybrid optimization algorithm in order to get more optimized results.

**References**

1. Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments by - A.I.Awad, N.A.El-Hefnawy, H.M.Abdel kader (2015)
2. Mobile cloud computing: Challenges and future research directions. by-Talal H. Noor, Sherali Zeadally, Abdullah Alfazi, Quan Z. Sheng (2018)
3. A review Of metahuristic Sheduling Techniques in Cloud computing. By-Mala Kalra , Sarabjit Singh .(2015)
4. Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. By- Sobhanayak Srichandan , Turuk Ashok Kumar , Sahoo Bibhudatta
5. Task Scheduling Using PSO Algorithm in Cloud Computing Environments, By- Ali Al-maamari and Fatma A. Omara
6. Monarch butterfly optimization by Gai-Ge Wang, Suash Deb, Zhihua Cui5