# Dynamic Load Balancing in Cloud Computing using Swarm Intelligence Algorithms

Abhijit Patil
Assistant professor
Department of Computer Engineering
D.J Sanghvi College of Engineering

Harshal Gala
Student
Department of Computer Engineering
D.J Sanghvi College of Engineering

Jai Kapoor
Student
Department of Computer Engineering
D.J Sanghvi College of Engineering

## ABSTRACT
In today's world, smart-phones and tablets have allowed cloud computing to realize it's true potential. It allows the users to use software, services and data on the go. This has resulted in increased study on cloud computing. With increased research in cloud computing, emphasis has been made in load balancing that allocates resources to multiple devices. Load balancing has played an important role in cloud computing by ensuring optimal use of resources with highest efficiency. The use of load balancing in the form of software and hardware has led many to discover new algorithms to achieve the same with better efficiency and minimum response time. This article discusses about the load balancing algorithms especially, swarm intelligence algorithms that can be used to balance load across devices. The algorithms taken into account are-PSO, Ant Colony Optimization, GSO and IWD.A study on advantages and limitations of the algorithm is made in order to realize the advantages of use of each algorithm in load balancing in it's own way.

## General Terms
Dynamic Load Balancing, Virtual Machines. Optimization.

## Keywords
Cloud computing, load balancing, swarm intelligence.

## 1. INTRODUCTION
**Cloud Computing**
Cloud Computing can be defined as disconnecting of software, hardware and applications from the actual computer itself. In layperson's terms, it can be stated as a philosophy of storage of data and programs over Internet rather than on the actual hardware device. Cloud Service Providers make use of this structured model to scale up IT Infrastructure by providing software and services [13] .They also ensure that the access of resources by organization is secure and is done efficiently. Cloud Computing can be deployed in three ways based on importance of certain criterions like security, efficiency etc. Private Cloud is a proprietary cloud service that provides services within the control of the responsible departments, it provides services only to a particular organization thereby conforming to high security standards [1]. Public Cloud on the other hand is basically used by Cloud Service Providers which efficiently provide services to multiple organizations/firms. Hybrid Cloud is a combination of Private and Public Cloud which provides special as well as public services depending on company to company requirements.

**Load Balancing in Cloud Computing**
In simple terms, load balancing allocates load(work) across multiple devices with an aim to use maximum resources with highest efficiency and minimum response time along with preventing single resource overload .It is essential for reliability and sharing through redundancy. Load Balancing is achieved mostly through a dedicated software or hardware to distribute workload. The most optimal load balancer ensures that no resource remains idle while other is being executed. Loads can be migrated depending on the amount of workload on a given resource. Load balancer also acts as a backup in case a resource fails to act by allocating its workload to other resources. Modifications can be achieved easily along with the necessary stability. When load balancing is applied during the process, it is dynamic load balancing whereas when load balancing is applied before the process, it is static load balancing.

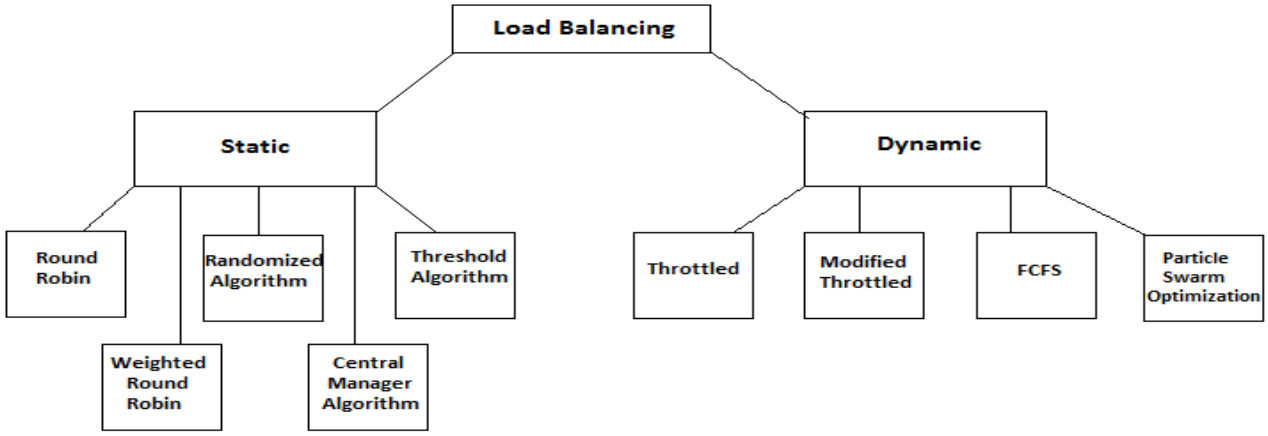


**Fig 1: Types of Load Balancing in Cloud Computing**

## 2. Types of Load Balancing in Cloud Computing

### Static Load Balancing

Load Balancing is achieved by pre-storing the necessary data about the system. The performance is evaluated after the execution process. Once the job is allocated to a node, the same job cannot be shifted to another node. The load here is distributed without considering the current load and state of the system which in turn affects the optimization of the system as it changes the dynamics of distribution due to load allocation.

### Dynamic Load Balancing

Unlike Static approach, this does not require prior knowledge about the system. It also takes the current state of the system into evaluation Dynamic Load Balancing is classified into two types based on processing and workload of the system- Distributed and Non Distributed .In Distributed, the node works side by side to achieve a common goal.

## 3. STATIC LOAD BALANCING

### 3.1 Round Robin Scheduling Algorithm (RSS)

RSS algorithm is the simplest of the static load balancing algorithms in which jobs are processed for equal amount of time which is time quantum. The data centre controller then requests to choose any randomly selected virtual machine. Servers and destination nodes are classified based on their processing times. Once the VMs are allocated a job then it moves to the end of the list. The disadvantage of this assignment is that the works are distributed unevenly (some nodes have high workload and others have not) and considering that the VM is not unoccupied then the unprocessed jobs have to stand in waiting (queue). This would make the process less efficient resulting in more Response time and inappropriate management of resources. No process is going to wait for its processing thereby removing starvation in the algorithm along with ensuring low throughput.

### 3.2 Weighted- Round Robin Scheduling Algorithm (WRRS)

WRRS algorithm was introduced to analyze the above problem found in Round Robin Scheduling such as starvation and priority scheduling. The algorithm is preemptive. In this algorithm, every node bears a weight depending on which requests are gathered, so requests are received as per the allotted loads. The load here is distributed evenly which would lead to high efficiency and appropriate management of resources.

### 3.3 Randomized Algorithm

In addition to the input, the algorithm uses a source of pseudo random numbers. Random decisions are taken depending on those random numbers. The output can vary if more test cases are performed on the same input.

It is simple and easy to implement. The algorithm is speedy with better probability and has better chances of producing optimum output.

The number of incorrect answer is finite. However, the number of wrong answers can be made small by the repeated continuous use of randomness. Randomized algorithm uses random numbers to choose slave processors. The slave processors are chosen at random as random numbers are generated based on the statistical distribution. Randomized algorithm will provide optimal performance among all load balancing algorithms for particular special purpose applications.

### 3.4 Central manager Algorithm

Central Manager Algorithm, is a step by step algorithm, in every step, central processor will choose a target processor to be given a job. The chosen target processor is the processor having the minimum load. The main processor is able to gather all target processors load information, thereby the choosing based on this algorithm are possible to be performed. The load manager makes load balancing decisions based on the system load data, allowing the best choice when of the process is been created. High degree of inter-process communication could make it a narrowing route state .This algorithm is expected to perform better than the parallel applications, especially when dynamic activities are created by different hosts.

### 3.5 Threshold Algorithm

Processes are allocated to the hosts as soon as it is created. To select hosts for new process locally, there is no need to send remote messages. Every processor stores a personal copy of the information about the load of the system. The processor's load is marked by three notable levels-underloaded, medium and overloaded. Two threshold parameters f_low and f_high can be used to describe these levels.

Below par loaded: load < f_low,

Medium; f_low ≤ load ≤ f_high,

Excessive Overload: load > f_high.

Initially, all the processors are initialized to underloaded state. When the current workload of a processor exceeds the given limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

## 4. DYNAMIC LOAD BALANCING

### 4.1 Throttled Load Balancing Algorithm (TLB)

TLB algorithm is a dynamic algorithm which completely establishes on virtual machines. In this assignment, the client appeals the TLB to find the suitable virtual machines to perform the operation. The virtual machines are assembled according to the invocations they can manage. Here the client first requests the load balancer to check the right virtual machine which access that load easily and perform the operations which is given by the client. The issue of this allotment is that the load balancer has to explore for the suitable virtual machine, which would cause delay in operation.

### 4.2 Modified Throttled Load Balancing Algorithm (MTLB)

This algorithm is Throttled Load Balancing Algorithm with some modification. MTLB Algorithm maintains a set of virtual machines named as virtual machine's index table and state the condition of the virtual machines i.e (avail/not-avail)[1].Virtual machine at first index is initially chosen depending upon the state. If Virtual Machine is accessible, then the request is assigned and if it is not discovered then it returns (- 1) to the Data Centre Controller. When the next requirement arrives, the machine next to the already allocated virtual machine is selected. Until the index table size is reached the process is repeated continuously. This algorithm

focuses mainly on how incoming jobs are assigned to the available virtual machines intelligently.

## 4.3 FCFS Algorithm:

It is the most basic parallel task ordering dynamic load balancing algorithm. The working takes place by selecting the correct order of jobs. The job would only be allotted with the virtual machine for first execution, when the user requests which comes first to the Data Centre Controller. The operation of FCFS policy is simply done with FIFO queue. The Data Center Controller finds for virtual machine which is in empty (free) or full. Then the initial request from the queue is removed and is given to one of the virtual machine through the VM Load Balancer. The assignment of request is done in two ways: Either the requests can be sequenced in a queue manner or by assigning heavy load node less work and lower load node with more work. Many operation conditions can be taken into account in order to compute the present actual load weighing value and the load weighing value.

# 5. SWARM INTELLIGENCE

The science of calculations inspired by the idea of 'collective intelligence'. Collective Intelligence can be defined as cooperation of large numbers of similar agents in an environment for a better collective living of the system. Examples include schools of fish, flocks of birds, and colonies of ants. Such intelligence is collective rather than a centralized one and is distributed throughout the system. In nature such systems are commonly used to solve problems such as effective foraging for food, prey evading, or colony relocation. The information is stored in the agent itself and communication is done through the agent's transmitter. A transmitter is basically the source of communication between two agents. For example , a pheromone which is dropped when ant travels from one node to another. Here, we discuss the impact of different type of swarm intelligent algorithms in load balancing.

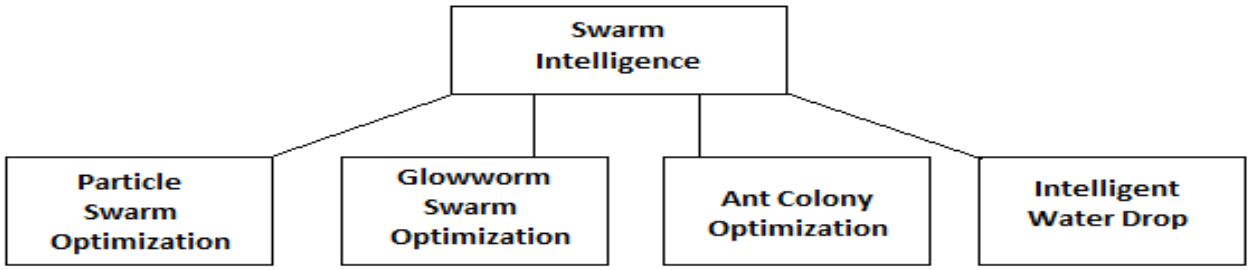


**Fig 2: Types of Swarm Intelligence Algorithms**

## 5.1 Particle Swarm Optimization:

It is an inspiration from group of flocking birds who are in search for some food. Every bird has its own knowledge regarding itself, location and obstacles. It also has knowledge through global coordinating system where each bird shares its own knowledge with others. The birds are benefited from the extra information they receive from other birds. A collective intelligence is available through information pool. This intelligence helps every bird to find food for itself. The same inspiration is used in load balancing where optimization is achieved through several iterations of improving a solution with regard to some measure of quality. There are a set of solutions where in particles are moved around in search space and evaluated by particle's position and velocity. Each movement of the particle is influenced by its local maxima and is also guided towards the best learnt positions and are constantly updated as found by other particles. However, this optimization does not guarantee optimal solution. When improved observations are obtained, these observations guide the movement of the swarm. This process is repeated for every generation with an aim to achieve optimal solution.

PSO can therefore also be used on optimization problems that are partially irregular, noisy, change over time, etc.

Algorithm

Algorithm Parameters[13] : A:Population of Agents, pi: Position of agent ai in solution space, f: objective function ,vi: velocity of agent's ai ,v(ai):Neighborhood of agent ai(fixed) [x*] = PSO()

P = Particle_Initialization();

For i=1 to it_max

For each particle p in P do

fp = f(p);

If fp is better than f(pBest)

pBest = p;

end

end

gBest = best p in P;

For each particle p in P do

v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);

p = p + v;

end

end

Particle Update Rule

Particle update rule

p = p + v with

v = v + c1 * rand * (pBest – p) + c2 * rand * (gBest – p)

where

- *p*: particle's position
- *v*: path direction
- $c_1$: weight of local information
- $c_2$: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

General Algorithm

1. Create a 'population' of agents (particles) uniformly distributed over X
2. Evaluate each particle's position according to the objective function
3. If a particle's current position is better than its previous best position, update it
4. Determine the best particle (according to the particle's previous best positions)
5. Update particles' velocities:
6. Move particles to their new positions:
7. Go to step 2 until stopping criteria are satisfied

The particles generated depend upon the amount of processing devices used, number of tasks and predefined population size. Initially there is a random generation of particles and the fitness value decides goodness of schedule and allocation of tasks. The pBest and gBest values are calculated. Then the

velocity and positions are updated. The process is repeated for maximum number of iterations possible. The optimal solution is obtained. Each candidate solution represents each particle due to which each particle corresponds to a decision for task assignment using vector of r elements and each element holding a value from 1 to n. As soon as maximum iterations are processed, the algorithm terminates. Due to this, the near optimal solution is obtained.

## 5.2 Ant Colony Optimization

Graph Searching Technique to find the shortest path from source to the destination nodes. An analogy here is drawn from ants who collectively come together and carry out tasks as a team. An ant explores some kind of food and as it explores, it drops in some kind of chemicals named pheromones in its path. Due to this, the other ants can know about the kind of path using the already dropped pheromones. When an ant reaches a node, it leaves in chemicals in the entire route it has followed. Every ant follows a similar process. Once the ant reaches the node, the magnitude with which the ant drops in chemicals is directly proportional to the goodness of the path. Basically, the ant will leave more pheromones if the path is the shortest one and conversely, it will leave few pheromones for the largest path. The use of pheromones is to direct the ants of new generation by giving them knowledge about the path. The path optimization keeps on improving along with every generation. Any ant considers two factors while it moves-attractiveness and traits. The attractiveness basically is based on how far the node is from the destination. It is a kind of heuristics which allows to move ant as close to destination as possible. Traits measure the content of pheromones. It infers the fitness value by measure of pheromones dropped by ants of every generation. It determines how good a path is, as decided by earlier ants. The probability p that any ant i from location j to move to edge k is equal to summation of traits and attractiveness [7].

Algorithm

Initialize the base attractiveness, τ, and traits, η, for each edge;

for i < IterationMax do:

for each ant do:

choose probabilistically (based on previous equation) the next state to move into; add that move to the tabu list for each ant;

repeat until each ant completed a solution;

end;

for each ant that completed a solution do:

update attractiveness τ for each edge that the ant traversed;

end;

if (local best solution better than global solution)

save local best solution as global solution;

end;

end;

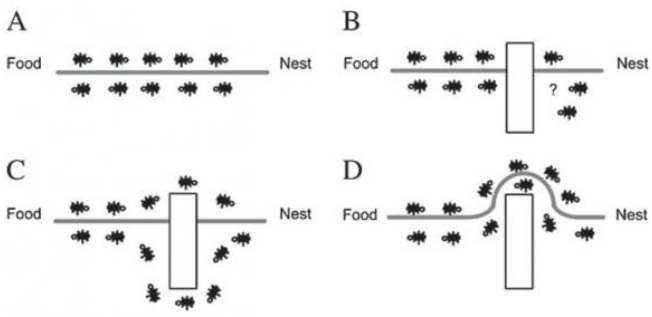


**Fig 3: Ant Colony Optimization**

Load balancing can be achieved effectively especially when it comes to balancing cloud based dynamic applications. A proposed optimization on ant colony algorithm has been performed that provides highly efficient load balancing for web services. By maximizing or minimizing different parameters of performance like processing load, memory available, delay or network load for the clouds of different sizes, one can develop an effective load balancer using ant colony optimization algorithm.

## 5.3 Intelligent Water Drops

One surprising observation noted while looking at the changing paths of the rivers makes us wonder the science behind it. If we manage to comprehend the science of this, we can use the same concept in developing algorithms for artificial intelligence and load balancing. The IWD algorithm is an attempt to model the dynamics that occur in a flowing river and then shape it in the form of an algorithm. The IWD algorithm is based on two properties-velocity and soil. Both the properties are dynamic and change during the span of an IWD. An IWD's path is from a source to a destination. The IWD starts with a predefined initial velocity and soil. During the process, it travels in the surrounding from which it removes some soil and in turn, gains some velocity. An IWD is organized (flown) in distinct steps. While moving from present location to its next location, the IWD velocity is escalated and is non-linearly proportional to the inverse of the amount of soil present in the two areas. Hence we can infer that the route with less soil will result in relative increment in velocity of the IWD as compared to the route with more soil [11]. The IWD accumulates soil during its trip in the environment. This soil is removed from the path joining the two locations. The amount of soil that is accumulated by IWD to pass from its present location to the next location, is non-linearly proportional to the inverse of the time needed for the

IWD. This time interval can be computed by the physics of linear motion. Thus, observed time is reliant on the velocity of the IWD and inversely proportional to the separation between the two points. It can be claimed that soil is the origin for information providing such that the environment and water drops both have information of soil. An IWD needs a system to select the path to its next destination. This system makes the IWD prefer the paths having low soil content to the paths having high soil content. This path selection is achieved by imposing an evenly randomized distribution on the soils of the untraversed paths. Then, the odds of the next path to select is reversely proportional to the soils of the already traversed paths. Therefore, paths with lower soils have better probability to be selected by the IWD.

The IWD algorithm has two types of parameters: Static and Dynamic parameters. Static parameters remain unchanged during the process of the IWD algorithm. On the other hand, dynamic parameters are updated after each iteration of the IWD algorithm. The pseudo-code of an IWD-based algorithm may be specified in eight steps[1]:

1. Initialization of Static Parameters

    - Graphical Representation of a Problem G

    - Updating values of Static Parameters

2. Initialization of Dynamic Parameters: amount of soil and velocity of IWDs

3. Even Random Distribution of IWDs on G(Graphical representation of the Problem)

4. Construction of Solution by IWD along with updation of amount of soil and velocity

    - Local soil updating on the graph

    - Soil and velocity updating on the IWDs

5. Local searching for each IWD's solution. This is an optional solution for better results

6. Updation of global soil values

7. Updating the best optimal solution

8. Go back to step b if ending condition is falsified

Intelligent Water Drops Algorithm provides an effective way for achieving load balancing especially when it comes to web service composition .The possibilities of implementing this in load balancers is much easier than PSO [6].It is also much more effective when it comes to composition based services.

## 5.4 Glowworm Swarm Optimization

It is one of a swarm intelligence algorithm. The behavior pattern of glowworm is the apparent capability of glowworm to change the intensity of the luciferin emission and thus arrive to light at different intensities. Luciferin is a term for the light-emitting compound found in organisms that generate bioluminescence. They typically undergo an enzyme-catalyzed oxidation resulting to a state that emits light upon decaying to its ground state .The GSO algorithm was developed and introduced in the Indian Institute of science, Bangalore in 2005 [2].

**Algorithm**

GSO algorithm makes the agents radiance at intensities approximately proportional to the operation value being optimized. It is assumed that glowworms of brighter intensities allure glowworms that have lower intensity. It assimilate an influential decision ranging by the effective distant glowworm is lessened when a glowworm has satisfactory number of nearby glowworms or which ranges beyond the approach of the glowworms

Working Of Algorithm:

1. The glowworms are spread all around the region

2. The swarm slowly gathers based on their distance to the optimum peaks and sufficient no. of nearby glowworms beyond the range of perception of the glowworm.

3. The swarm slowly goes to their optimum peaks

4. Swarm clusters in their optimum solution

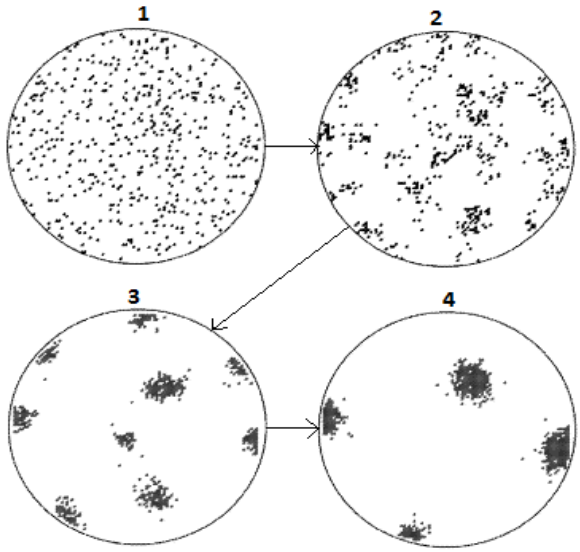


**Fig 4: Glowworm Swarm Optimization**

**Load Balancing with GSO**

In GSO, each glowworm scatters in the objective function definition space. These glowworms carry own luciferin values and have the respective field of idea scope called local-decision range. As the glow seeks for the neighbor set in the local-decision range [10], a brighter glow has a higher affinity to attract this glow toward this traverse, and the flight direction each time different will change along with the choice of nearest glowworm. Also, the local-decision range size will be influenced by the neighbor quality, when the neighbor density is less, glowworm's policy making radius will enlarge favors and seek for more neighbor.

Each glowworm g encodes the object function value $W(x_g(t))$ at its present location $x_g(t)$ into a luciferin value $l_g$ and transmits the same within the neighborhood. The set of neighbors $(M_g(t))$of glowworm g consists of those glowworms with high luciferin value and that are present within a dynamic domain and the formula at each iteration. Local-decision range update is given by equation (1)[12]:

$$k_d^g(t+1)=min\{k_s, max\{0,k_d^g(t)+\alpha(y_t-|R_g(t)|)\}\} \quad (1)$$

and $k_d^g(t+1)$ is the glowworm g's location-decision range at the t+1 iteration, $k_s$ is the sensor range ,$y_t$ is the neighborhood range, α which affects the rate of change of neighborhood range. The quantity of glow in local-decision range is given by equation (2):

$$R_g(t)=\{j:||x_j(t)-x_g(t)\}||< k_d^g; l_g(t)<l_j(t)\} \quad (2)$$

here, $x_j(t)$ is the glowworm g's luiferin at the t iteration; the set of neighbors of glowworm g consists of those glowworms that have a relatively higher luciferin value and that located within a dynamic decision domain whose range $k_d^g$ is bounded about a range $k_d(0<k_d^g< k_d)$.Every glowworm in equation (3) g selects a neighbor j with a probability $P_{gj}(t)$ and process as. $kER_g(t)$

$$P_{gj}(t)=\frac{lj(t)-lg(t)}{\sum k\in Rg(t)lu(t)-lg(t)} \quad (3)$$

Movement update is shown by (4):

$$X_g(t+1)=x_g(t)+s(\frac{xj(t)-xg(t)}{||xj(t)-xg(t)||}) \quad (4)$$

Luciferin-update is in equation (5):

$$l_g(t)=(1-P)l_g(t-1)+¥J(x_g(t)) \quad (5)$$

here, $l_g(t)$ is a luciferin value of glowworm g at the t iteration, P(0,1) lead to reflection of the cumulative kindness of the path followed by the glowworms in their present values, the parameter ¥ only scales the function values, $J(x_g(t)$ is the value of test function.

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-points of white space above the section head. Sections and subsequent sub- sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head.

| Algorithms | Particle Swarm Optimization | Ant Colony Optimization | Intelligent Water Drops | Glowworm Swarm Optimization |
|---|---|---|---|---|
| Author | Millie Pant, Dr.R.Umarani et al. | V.Selvi , Dr.R.Umarani | S. Rao Rayapudi, Shah Hosseini. | K.N. Krishnanand and D. Ghose |
| Issues | Makespan,cost, Deadline. | Execution time, Maximum execution time, idle time. | Soil, velocity. | Accuracy and convergence rate |
| Advantage | It is very simple,have no overlapping and mutation calculation, it adopts the real no. code and it is decided directly by the solution. | Inherent parallelism; Positive feedback accounts for rapid discovery of good solution; Efficient for TSP and similar problems; Can be used in dynamic application | Best choice for finding optimial solution;In order to solve servicescomposition IWD algo are used; Higher coorectness value ,feasibility and effectiveness than PSO. | Solving the Knapsack problem ; Dispatching System of Public Transport; multi-model Function with collective robotics and also chasing multiple mobile signal sources |
| Limitation | It suffers from the problem of premature convergence, particularly in case of multimodal optimization problems. | Theoretical analysis is difficult; Sequences of random decision decisions; Probability on distribution changes by iteration; Time to convergence uncertain | Only for web service composition. | Situations, in which the glowworm does not change its position. |

**Fig 5: Comparison Of Swarm Intelligence Algorithms**

## 6. FUTURE WORK

Ant Colony Optimization: Better efficiency can be achieved if clusters can be developed to implement the total solution of load balancing. By altering and experimenting with different parameters like processor load, memory capacity, delay for clouds of different requirements[7][8]. A heuristic algorithm can be developed to change the pheromone measuring mechanism to minimize make span of cloud services and achieve portability of request servicing. Fault tolerant issues can be considered in future experiments and research.

Particle Swarm Optimization: A mathematical analysis has been made by Clerc and Kennedy (Clerc M, Kennedy J , 2002) on convergence from Math's point of view. By studying the stability of the condition transmitting matrix, limited conditions can be observed where particles can move stably[8]. Later studies explored the effect of casualty on the locus of the particle, and convergence was studied from point of measuring space. Though the theory remains unproved, it still provides substantial information for performing future work on this. Research can be carried out to study the topology of the particle .The mimicking of different societies can redefine neighboring topology .By studying different topologies, we can achieve better spread of algorithm and best property for PSO. By implementing PSO along with other algorithm can be also a focus for research. Advantages of PSO along with advantages of other algorithms that can overcome the disadvantages of PSO can do wonders in achieving optimal solution. For example, the particle swarm optimization algorithm can achieve better optimization and efficiency with the simulated annealing (SA) approach [9]. It can be connected with the hereditary agents, the algorithm of a colony of ants, vague method and etc.

Intelligent Water Drops: The ever increasing demand of web services has made it necessary to solve the difficulty of service selection to satisfy user demands in web service composition. Web services composition is a new software development paradigm, and it is a key point to achieve service oriented computing currently. IWD allows better than PSO when it comes to correctness of the value, feasibility and efficiency [6]. More focus can be made on studying the computational overhead with a myriad number of services. Focusing on achieving the same efficiency with large number of services can help in achieving the most optimal solution for load balancing.

GSO:A study can be made on discontinuities in chosen objective function. Several intriguing questions about the relation of parameter values and algorithm refining, based on different applications can be raised[2]. An analysis can be made on effects of different aspects of algorithm on its performance along with its comparison with other swarm intelligence algorithms. Blending with other algorithms for more effective load balancing can also be studied.

## 7. ACKNOWLEDGMENTS

We take this opportunity to express our profound gratitude and deep regards to our professor for his exemplary guidance, monitoring and constant encouragement throughout the process. We would also like to thank various laboratories for their invaluable guidance.

## 8. REFERENCES

[1] https://en.wikipedia.org/wiki/Cloud_computing

[2] Multiagent and Grid Systems – An International Journal 2 (2006) 209–222 209 IOS Press Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications.

[3] Analysis of Particle Swarm Optimization Algorithm.

[4] Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications

[5] 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN) Performance Study of Some Dynamic Load Balancing Algorithms in Cloud Computing Environment.

[6] An intelligent water drops algorithm based service selection and composition in service oriented architecture.

[7] Load balancing in a network using Ant colony optimization technique.

[8] International Journal of Computer Applications (0975 – 8887) Volume 63– No.15, February 2013 8 Comparative Analysis of Various Evolutionary Techniques of Load Balancing: A Review

[9] International Journal of Computer Applications (0975 – 8887) Volume 5– No.4, August 2010 1 Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques

[10] 2014 International Conference of Intelligent Computing Application-A load balancing model in public cloud using ANFIS and GSO

[11] Intelligent water drops algorithm A new optimization method for solving the multiple knapsack problem

[12] Journal of Convergence Information Technology volume 6 number 2-Feb 2011-Using Load Swarm Optimization Algorithm for Clustering Analysis