STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720 (Online)
Volume 5, Issue 2
www.stmjournals.com

A Greedy Stochastic Diffusion Search-based Fuzzy Scheduling in Cloud

Ranga Swamy Sirisati[1],*, Sridhar Mandapati[2]
[1]Research Scholar, Department of Computer Science, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India
[2]Associate Professor, Department of Computer Applications, RVR & JC College of Engineering, Guntur, Andhra Pradesh, India

Abstract

Users in cloud environment pay per use for the resources utilized. Security issues direly affect the cloud services due to which the performance of the cloud systems also gets affected. Hence, for robust security in data and systems in cloud, the onus is largely on service providers. The most important component of cloud computing is job scheduling. For distributing precedence among subtasks that achieve varied make span in heterogeneous computing system based on the different procedures adopted by different job scheduling algorithms. Additionally, every resource assigned to a task may consume variable amount of energy. The problem of solving the cloud scheduling problem is Non-deterministic Polynomial (NP) hard problem. The minimization of make span is the target of most investigations. This work considers both make span as well as energy consumption. For optimizing these tasks, the greedy and the Stochastic Diffusion Search (SDS) algorithms are combined in job scheduling in this work. Fuzzy reasoning is combined with greedy SDS mechanism by the proposed methodology for optimizing scheduling. The greedy SDS method is used for exploiting the fuzzy solution space and combining the benefits of both, the heuristics and evading the drawbacks is the basic tenet of the proposed approach. It has been shown that in terms of both the consumption of energy and makes span, the proposed algorithm performs better than the existing algorithms.

Keywords: *Cloud computing, scheduling, security, fuzzy logic, greedy search and stochastic diffusion search (SDS)*

****Author for Correspondence*** E-mail: sirisatiranga@gmail.com

INTRODUCTION

As cloud computing provides resources based on demand, it is referred to as 'on demand resource provisioning'. It is based on subscription. There is a central remote server that maintains all the data and the application. The service is pay per use in cloud computing, and the reliability boosts the network performance. Due to its reliability, speed, fault tolerance and effective communication among different networks, cloud computing is fast becoming an emerging technology [1].

The capability of running several operating systems on one physical system and sharing the underlying hardware resources is referred to as virtualization. Consolidation of resources and their management is one of the basic aspects of virtualization. The hardware and software system resources are abstracted from the operating system by means of virtualization. Hypervisor or Virtual Machine Monitor (VMM) lies between the hardware and the Operating System (OS). Para virtualization and full virtualization are the two types of virtualizations available.

While cloud computing presents several advantages, there are many drawbacks as well. Security is the main challenge in cloud computing. These issues with security are dynamic and are also of wide range. As different services like Network as a Service (NaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS) are provided, the security of corporate data in the cloud is a challenge. There are separate security issues associated with every service. The confidentiality, integrity and the availability of

© STM Journals 2018. All Rights Reserved

the data is referred to as Data Security. For cloud vendors, these are the main challenges. When the security of a client's data and its totality is managed by the service provider, customers become eventually accountable for the data security. Outsourced surveys and security certification are the vulnerabilities more often than of conventional service providers [2].

When users make use of the data, they are unaware about the location where the data is stored and hosted. They may also be unaware of the country were the data is stored. The resource data that has been externally processed has native risks. This is because they deploy services avoiding mortal, consistent and human resources managing the IT shops working on the house programs. In cloud computing, another major issue is the trust, which is centred on assurance and confidence, and which can exist between two humans or between machine and human. As users trust the cloud, they store data on it. The process of restoring lost data due to corruption or accident is defined as data recovery [3].

As the data in cloud is centralized and comprises a universal architecture, the cloud implementations comprise advanced security technologies. All the users can focus on securing the cloud architecture as the resource pool in cloud is, by and large, homogeneous. Also, advanced security capabilities result due to the automation within cloud, along with several security resources. As this field evolves exponentially, it is necessary to have a thorough insight. Many believe that this hype about cloud computing is transitory and hence, care should be exercised so as not to get carried away by its momentary popularity. Several drawbacks in traditional architecture can be overcome quintessentially by the unique characteristics of cloud computing. However, there may be several unclassified threats that may get introduced as a result of the innovative architecture [4].

The most important exercise in cloud computing is job scheduling. This increases the efficacy of the workload distribution. Thus, resources can be appropriately utilized. It is also a very challenging theoretical problem in cloud computing. There has been a lot of research performed in the field of job scheduling in cloud computing. It is the chief task that is set to begin at a designated time. Maximum effectiveness of workload balance can be achieved by means of scheduling. Making appropriate use of resources while maintaining a load among the resources is the objective of scheduling, and this greatly helps in reducing the execution time [5].

Scheduling in cloud computing is an NP-hard problem. There are several popular heuristics that provide solutions to this problem however, in a constrained manner. An algorithm that can ensure an almost optimal solution in duration less than the polynomial time is referred to as a heuristic. It finds an optimal path in the solution space that is available and overrides or eliminates other seemingly possible paths. Cluster scheduling and list scheduling are the algorithms that are based on heuristic. It is assumed by the clustering algorithm that there are several processors for doing the work for the sub tasks to be executed, thus it makes use of as many processors as possible for decreasing the make span of the job schedule spawned [6].

Several heuristics belong to the list scheduling class. Metaheuristic based algorithms additionally make use of a combinatorial process for finding a solution. In the given search space, these algorithms need sufficient sampling of solutions; they have performed superior to many other scheduling algorithms, such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Simulated Annealing (SA) and Genetic Algorithms (GA), and have been used successfully across several scheduling problems. The one problem that exists with the heuristic based algorithms is the high cost of computation involved along with the decreased efficiency [7].

SDS is a well-organized and robust optimization algorithm ensuring the best global solution. It is also defined mathematically. During the iterations, the greedy algorithm builds the globally best solutions using the locally best options available. The advantages of greedy

algorithms are simplicity, efficiency, and verification.

For cloud computing, the greedy search algorithm and greedy SDS fuzzy algorithm have been proposed. These are the sections into which the remainder of this investigation has been organized— the related works in literature have been discussed in the second section, the techniques used in this work have been discussed in the third section, the empirical outcomes have been discussed in the fourth section and the work has been concluded in the fifth section.

RELATED WORKS

A workflow Task Scheduling algorithm based on the Resources' Fuzzy Clustering (FCBWTS) was presented to minimize the make span of applications that were limited by precedence and which could be modelled as acyclic graph [8]. The FCBWTS considers the resource characteristics of cloud computing as a set of features that can explain the synthetic performance of the processing units in the resource system which are defined in this work. These features, along with the execution duration effect of the ready task in the crucial path, are used to pre-treat the processing unit network using the fuzzy clustering approach. This helps realize a logical division of the processor network. Hence, the cost to decide which of the processor can execute the current task is largely reduced.

A Two-stage Artificial Bee Colony (TABC) algorithm with several improvements used to solve Flexible Job Shop Scheduling Problem (FJSP) was introduced in [9]. This was with new job insertion and fuzzy processing time constraints. This has also proposed many novel generation techniques and strategies for improvement, and comparison has been made with each other. Minimizing the fuzzy completion time is the objective here. The proposed TBAC algorithm is used for solving 8 instances from re-manufacturing. The suggested strategies for improvement have been compared and elaborately discussed. 5 benchmark cases have been used to compare the two suggested Artificial Bee Colony (ABC) algorithms, with the best performance, with 7 algorithms that are present. The superior performance of the TBAC in solving the FJSP has been shown by the optimization outcomes and comparisons.

A feasible and flexible dynamic task scheduling scheme has been proposed. This is known as Dynamic Greedy Strategy (DGS) [10]. This can allocate virtual resources dynamically for executing computing jobs; the scheduling was duly and promptly completed, and so was the process of execution using greedy strategy. The suggested scheme can be realized using CloudSim simulator, and its outcomes have shown that the completion time of the tasks can be hastened as well as proper utilization of the cloud resources can be attained using the DGS.

For solving the task scheduling issue in the cloud environment, a metaheuristic approach of ACO algorithm was presented [11]. This focussed on chief objectives— decreasing the computation time or the make span and achieving better balancing of loads. It was proven that better results compared to the NSGA-II algorithm were obtained, in terms of lesser makes pan and better load balancing by Load Balancing ACO algorithm (LB-ACO). CloudSim was used for carrying out the simulations.

There are several tasks that vary in length, runtime, input and output data which comprise workflows. For scientific workflows, cloud computing is the most suited. The parameters like resource utilization make span and cost have been factored for workflow scheduling problem. The steps that can map the workflow tasks to cloud resources using the ACO have been suggested [12]. This tries to reduce the resource cost and the make span to minimum and maximize the resource utilization.

A heuristic that was based on PSO compared with Genetic Algorithm (GA) and First Come First Serve (FCFS). PSO, a population-based search algorithm, was suggested [13] which was influenced by the social behaviour of a flock of birds. Minimizing the make span and the waiting time for a given set of tasks has been the goal of this research. The CloudSim toolkit has been used for simulating this policy alongside two other algorithms. It has been

shown by the results that compared to the genetic and the FCFS algorithms, PSO performed much better.

An algorithm that could schedule a workflow application which was a data intensive application in the cloud computing environment, known as the Bat Algorithm (BA), has been suggested [14]. After successfully implementing the algorithm, the outcomes have been compared with PSO and Cat Swarm Optimization (CSO). Fair distribution of load along with optimal processing cost and better convergence has been achieved using the suggested BA algorithm.

A metaheuristic-based algorithm that focussed on minimizing the cost and the makespan was proposed [15]. The suggested heuristic was used for scheduling workflow applications and was a combination of some well-known metaheuristic algorithms such as Gravitational Search Algorithm (GSA) and equally popular heuristic, Heterogeneous Earliest Finish Time (HEFT). For realizing the bi-optimization objective, a new factor known as the cost time equivalence has been introduced by this work. For comparing the performance of the suggested algorithm with the existing ones, it takes into consideration Monetary Cost Ratio (MCR) and also Schedule Length ratio (SLR) as performance metrics. The efficacy of the suggested algorithm has been demonstrated through exacting experiments over different scientific workflows, and its superiority has been proven over standard GSA, HEFT and the Hybrid Genetic Algorithm (HGA).

As seen from the literature survey, various kinds of techniques are utilized for scheduling in cloud. The main drawbacks of the methods are that they are computationally complex or local optima problem is there in heuristic methods. In our method, fuzzy greedy concept of combinatorial optimization problems evaluates the best part at a certain stage in the algorithm in the greedy approach. Regardless of any previous choices or future occurrences, the greedy algorithm always makes a choice that is momentarily best. Thus, whenever an algorithm gives an optimal solution, it is only aware of the absolutely best made choices, failing to do so will result in the best choice becoming the worst. This will result in another extreme case where the greedy algorithm produces the unique worst possible solutions. Vague concepts expressed in natural language are allowed by this fuzzy logic like the choice that is made in the greedy approach. This refines the process to achieve the best solution for scheduling.

METHODOLOGY

In this section, the fuzzy logic, fuzzy rule selection, rule selected SDS, greedy search aware fuzzy scheduling algorithm and greedy SDS fuzzy algorithm are discussed.

Fuzzy Logic and Fuzzy Rule Selection

Zadeh (1965) introduced the fuzzy logic that has been incorporated by the fuzzy controller. Unlike binary logic, the fuzzy logic does not follow strict allotment of elements to sets. Instead, a degree of membership to the set is possessed by every element in that set. A value between 0 and 1 is used for representing the degree. A fuzzy system has to be formulated for being able to apply fuzzy logic to a certain problem like the scheduling between the cloudlets or virtual machines. These systems are easy to comprehend, flexible, can bear irrelevant data and can also model non-linear functions of random complexity; the other benefits of this system include the ability to be used to approximate functions and to model any continuous function or system. The process of formulating the mapping from a given input to an output by utilizing the fuzzy logic is known as Fuzzy interference; the basis from which decisions can be formulated or the patterns can be recognized is based on that mapping [16].

For simulating the human decision making based on fuzzy control rules and associated language parameters, the Fuzzy Interference System (FIS) is employed. The outputs of the inferential rules can be associated using the low-high inference approach. A series of IF-THEN rules has been defined using the rule-based structure of the fuzzy logic; this is done for the outputs of some given input criteria wherein a set of linguistic control rules comprise the rule including the control goals in the system. The process of converting the

fuzzy output set into a single number is referred to as the process of defuzzification. The Smallest of Minimum (SOM) is the technique that is used in defuzzification. A range of output values has been included in the aggregate of the fuzzy set. For resolving one output value from the fuzzy set, it can be defuzzified. Taking up the collated linguistic values from the fuzzy control action that has been inferred, the defuzzifier generates a non-fuzzy control output that is used for representing the balanced load that is adjusted to load conditions. The membership function of the aggregated output can be computed using the defuzzification [17].

Rule Selected Fuzzy SDS Algorithm

A population-based and nature-motivated optimization algorithm that is based on direct communication (one-to-one) between the agents is the SDS algorithm, which is dual phased. The test phase involves every agent testing a potential solution to the problem (hypothesis), and the diffusion phase involves the agents sharing information regarding the hypothesis using one-to-one communication [18].

Another method is to use an SDS for membership function tuning in FLC. Every agent, in the SDS, is so shaped as to represent the membership function parameters of the inputs as well as the outputs of the FLC. Minimizing the control error of the FLC is the aim of the SDS. We can define the SDS objective function as follows: Certain assumptions exist in the formulation of model. As a basic integration of this hybrid algorithm, these assumptions have to be not only defined but also made available. These are the assumptions: (i) for input and output variables, Gaussian membership functions have been used. (ii) The complete rule base was considered. When all the possible combinations of input membership functions of all the input variables take part in fuzzy rule-base formation, a rule is considered complete [19].

The integration between optimization algorithm and fuzzy logic problem is as follows: (i) The mean value and standard deviation of each fuzzy membership function are the parameters. (ii) These parameters act as particles and looking for the global best fitness. (iii) It starts with an initial set of parameters. (iv) This parameter will be used to check the performance of the fuzzy logic, after the parameters had been adjusted using optimization method. (v) The process continues until the goal is achieved. The optimization method as shown in starts with the initial set of parameters. To obtain new values for the parameters of the membership function, it employs a fitness function.

Greedy Search Algorithm

An algorithm that employs a heuristic to make optimal choices locally at every stage with the objective of finding a global optimum is a greedy search algorithm. This algorithm neither involves backtracking nor revisiting the choices made by the algorithm. For several optimization problems, these are among the first to be considered. The general form of a greedy algorithm for a maximization problem is as given: (1) selection standard: Choose an item in a greedy way which is locally optimal as per a simple condition at the present stage. (2) feasibility condition: Accept an item being considered in case some feasible condition is met- like this item constitutes a feasible solution including the tasks that have been accepted so far under the problem constraints, else discard it. Thus, an item that has been taken into consideration and rejected will never be accounted for again. This criteria for selection is associated with the constraints and the objective function and is generally the ratio of the 'advantage' to 'cost' that can evaluate the efficacy of an item. The constraint in a problem of this type is derived from the ability to hold the selected tasks and the aim is minimization of social welfare; hence, we can say that, in this case, the selection criteria is the ratio of the task value to the task demand [20].

Greedy Search Aware Fuzzy Scheduling

Let us take into consideration combinatorial optimization problems [21]; every one of it is related to a discrete solution space X and a feasible space S wherein $S \subseteq X$ which is defined by the problem constraints, and an objective function $f : X \mapsto \Re$. The aim of minimization is finding a feasible solution

$x^* \in S$ such that $f(x^*) \le f(x),\ \forall x \in S,$ where, $x = (x_1, ..., x_n)$ is a vector of decision variables (solution). c(x) is the cost function. $\forall x \in \{x_1, ..., x_n\}$ is defined for each specific problem. Function c(x) is referred to as a greedy evaluation function. It represents the degree of prerogative to include x into the solution space that is being constructed sans bringing infeasibility.

An alternative approach is now described. X is treated as a fuzzy set having a well-defined membership function $\mu(x)$, the form of which is given by equation (1).

$$\mu(x) = \frac{1}{1 + \lambda^2 \rho \left(\left(\frac{1-\lambda}{\lambda} \right) x - \theta \right)^2} \qquad (1)$$

The variable $x \in X$ corresponds to one of the variables in the definition of the combinatorial optimisation problem; here it considers $x \in \Re$. The parameter θ is a basic measure for evaluating the priority to be assigned to variable x; here, it requires $\theta \in \Re$. The parameter λ is a tuning parameter selected experimentally such that $0 \le \lambda < 1$. This is a significant consideration in the developed algorithm at later stages. The parameter $\rho > 0$ is effectively a shape parameter, so that as the value of ρ increases, the graph of $\mu(x)$ becomes narrower. The proposed evaluation function $\mu : X$ a (0.1) has the following properties: $\mu(\lambda\theta / (1-\lambda)) = 1$ and $\mu(x) < 1$ for all $x \ne \lambda\theta / (1-\lambda)$.

The modification of the general formulae in the families if the described fuzzy members are given by equation (1). This function is referred to as a fuzzy greedy evaluation function and the role of the greedy evaluation function in deciding the degree of prerogative for an element x. The fuzzy greedy search algorithm is shown below:

BEGIN
$t = 0;$
$p(t) = \phi;$
WHILE $(P(t)$ *is not complete*$)$ *DO*
$P(t)$ = *construction* $(seed, \lambda, \theta);$
WHILE $($*not ter* min *ation condition*$)$ *DO*
BEGIN
$t = t + 1;$
$P(t)$ = *recombine* $(P(t$ - $1));$
evaluate $(P(t));$
update $(\lambda, \theta);$
WHILE $(P_C(t)$ *is not complete*$)$ *DO*
$P_C(t)$ = *construction* $(seed, \lambda, \theta);$
$P_R(t)$ = *select* $(P(t));$
$P(t) = P_C(t) \cup P_R(t);$
END
END

The variable P(t) represents a set of population members at iteration t. The variables $P_C(t)$ and $P_R(t)$ represent constructed and reproduced sub-populations respectively at iteration t.

Proposed Greedy SDS Fuzzy Scheduling

An algorithm that constructs an object X one step at a time is referred to as Greedy algorithm. The locally best option is selected at every step. There are many advantages of the Greedy algorithms: It is simple and easy to describe, and also to code compared to the other algorithms. It is efficient to implement compared to other algorithms. However, there are also several drawbacks. Designing the algorithms can be easy once the right approach has been found. Yet, this stage of finding the right approach is tough. Also, verifying a greedy algorithm needs a subtle argument.

Bishop pioneered the SDS as a population-based matching algorithm which made use of direct patterns of communication like cooperative transport prevalent among social insects for evaluating the search as well as the optimization hypothesis. In cases where the component functions can be assessed individually using the swarm agents, SDS is the most viable algorithm. In terms of convergence to global optimum, linear complexity and minimal convergence criteria, SDS has a robust mathematical framework that can describe the algorithmic behaviour. Theoretically speaking, SDS can be used for

changing objective functions and can, in this manner, cater to the problems that are dynamic in nature. Thus, SDS finds its use across various fields like eye tracing in facial images using stochastic search and n-tuple network, mouth locating in human facial images and site decision for transmitting equipment for wireless networks [22].

The higher overheads of computation that result because of greedy algorithm being deployed into a problem that has an expensive fitness function can be mitigated using the hybrid of greedy and SDS in the partial function evaluation, deployed in the SDS. An initial set of experiments have been carried out to find if the greedy behaviour can be improved on its own by the information diffusion that has been deployed in SDS. This area has a lot of on-going research being conducted. The results of the initial experiments have been demonstrated in this work [23].

The performance of the hybrid algorithm has been evaluated in this novel architecture using a standard set of benchmarks. After partially evaluating the search space, the allocation or recruitment of resources and the partial function evaluation side of the SDS is utilized for assigning and deploying the resources (e.g. members of the greedy search population). There is one hypothesis and one status that is associated with every greedy and SDS agent. Each member of the greedy population is also an SDS agent, in the reported experiment here (hybrid algorithm), and is referred to as Greedy SD SAgents. Here, the greedy target vector is used to define the SDS hypothesis. The Greedy SDS Agents being active or inactive is determined by an additional Boolean variable (status).

Every agent has to partially evaluate its hypothesis in the test phase of the SDS. The hypotheses that are promising are retained and those that are not are discarded. There are several tests that are performed in the context of the hybrid greedy SDS algorithm for determining the activity for each of the Greedy SDS Agents. The fitness of every Greedy SDS Agents target vector is compared with an arbitrary Greedy SDS Agents in the test phase here. If the selected one has a superior fitness value, it becomes active else it is tagged as inactive. This mechanism ensures that at least half of the Greedy SDS Agents remain active across iterations. In the Diffusion phase, every Greedy SDS Agents that is inactive randomly selects another Greedy SDS Agents and if the selected one is found to be active, it communicates its hypothesis to the inactive Greedy SDS Agents; on the other hand, if the selected Greedy SDS Agents is also inactive then a new hypothesis is randomly generated by the selecting Greedy SDS Agents in the search space.

In order to permit scalable, distributed and noise tolerant optimization, there have been several versions of the conventional greedy algorithm in the recent past. The most recently proposed is the stochastic greedy algorithm, which being a linear time algorithm takes an element at each step, approximating in expectation the element that has the peak marginal contribution. The stochastic-greedy gives an approximation guarantee which is arbitrarily close to 1 – 1/e in expectation and does very well in practice [24].

Generally speaking, only the elements that are of the peak expected value approximately are iteratively added by a greedy algorithm. These may, however, not be necessarily because of the design decision but an artefact of its application in huge and noisy datasets. One can model this uncertainty with a probability distribution D: at every iteration, a value $\xi \sim D$ is being sampled, and the greedy algorithm adds an element whose marginal contribution is a ξ -approximation to the maximal marginal contribution.

In general, it refers to an algorithm that iteratively adds an element whose marginal contribution is approximately optimal in expectation as a stochastic greedy algorithm. It is easy to show that stochastic greedy algorithms give an approximation ratio of $1-1/e^{\mu}$ in expectation, where μ is the mean of the distribution modelling the uncertainty. This, however, is a weak guarantee as it leaves a non-negligible likelihood that the algorithm terminates with a solution with a poor

approximation guarantee. Indeed, as it later shows, there are cases where stochastic greedy algorithms have desirable guarantees in expectation, but with constant probability they have arbitrarily bad approximation guarantees.

Fast algorithm for sub-modular optimization analysis applies to the stochastic-greedy algorithm, thus showing its approximation guarantee holds with high probability; thereby implying it is the optimal algorithm in terms of running time and approximation guarantee for the problem of maximizing a sub-modular function under a cardinality constraint.

Combining the strengths of various techniques so as to improvise the efficacy of a single approach is the objective of the design of the hybrid technique. For heuristic scheduling here, a hybrid metaheuristic has been suggested. The use of the SDS data is the basis for the suggested approach. It also includes some of the ideas of Greedy Randomized Adaptive Search Procedure (GRASP). The solutions from the construction stage of GRASP are used in this approach as the initial SDS population. For this hybrid technique, a novel adaptive fuzzy greedy search operator has been described. A solution has been denoted by an integer string for applying in the hybrid method to fuzzy scheduling; every string is a sequence of n arranged agents that are numbered from 1 to n— they represent the order of agents in the circle. The suggested technique makes use of a set of candidate agents. Each of the agents can be included into the constructed partial solution that does not cause infeasibility [25].

An extension of the fuzzy greedy evaluation concept in the form of metaheuristic is the greedy SDS fuzzy scheduling. The technique that has been suggested is an iterative procedure that is based on the population. The initial population is the beginning of the process. The construction phase of the SDS can be used for its generation. The algorithm works on the set of individuals (called population) which is divided into two different sub-sets generated in different ways. One set is spawned by means of a selection scheme as well as a recombination operator, and this itself is the standard evolutionary approach. The other set employs the construction process and is akin to the SDS construction phase; the exception is that a general version of the fuzzy greedy evaluation function has been adopted in place of the conventional greedy evaluation function.

One of the sets is generated through a recombination operator and also with a selection scheme. This is a standard evolutionary approach. The other set is built employing a construction procedure. It is similar to the SDS construction phase except that the procedure adopts a generalised version of the fuzzy greedy evaluation function instead of the classical greedy evaluation function.

RESULTS AND DISCUSSION

In this section, the fuzzy, rule selected fuzzy SDS, greedy fuzzy and greedy SDS fuzzy methods are used. The make span, degree of imbalance and resource utilization are shown in Tables 1–3 and Figures 1–3.

Table 1: *Makespan for Greedy SDS Fuzzy.*

Number of Jobs	Fuzzy	Rule Selected Fuzzy SDS	Greedy Fuzzy	Greedy SDS fuzzy
200	44	40	42	38
400	92	86	88	82
600	147	138	141	132
800	193	182	187	176
1000	247	226	236	216

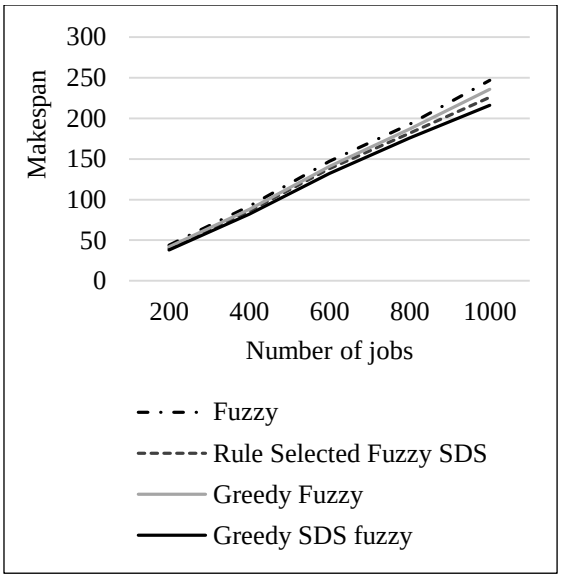

Fig. 1: *Makespan for Greedy SDS Fuzzy.*

From the Figure 1, it can be observed that the greedy SDS fuzzy has lower makespan by

14.63%, 5.12% and 10% for 200 number of jobs, by 11.49%, 4.76% and 7.05% for 400 number of jobs, by 10.75%, 4.44% and 6.59% for 600 number of jobs, by 9.21%, 3.35% and 6.06% for 800 number of jobs and by 13.39%, 4.52% and 8.84% for 1000 number of jobs when compared with fuzzy, rule selected fuzzy SDS and greedy fuzzy. The proposed Greed SDS fuzzy is able to achieve better scheduling as the allocation of resources and the partial function evaluation side of the SDS is utilized for assigning and deploying the resources.

From the Figure 2, it can be observed that the greedy SDS fuzzy has lower degree of imbalance by 12.76%, 4.44% and 8.69% for 200 number of jobs, by 8.69%, 4.44% and 4.44% for 400 number of jobs, by 10%, 5.12% and 5.12% for 600 number of jobs, by 10.52%, 5.4% and 5.4% for 800 number of jobs and by 17.14%, 6.06% and 11.76% for 1000 number of jobs when compared with fuzzy, rule selected fuzzy SDS and greedy fuzzy. As the globally best solutions are obtained by the proposed Greedy SDS fuzzy, the degree of imbalance is significantly lower.

From the Figure 3, it can be observed that the greedy SDS fuzzy has higher resource utilization by 6.13%, 2.4% and 4% for 200 number of jobs, by 5.84%, 2.07% and 3.31% for 400 number of jobs, by 6.33%, 2.7% and 3.41% for 600 number of jobs, by 6.26%, 2.66% and 4.09% for 800 number of jobs and by 4.69%, 2.19% and 2.07% for 1000 number of jobs when compared with fuzzy, rule selected fuzzy SDS and greedy fuzzy. The proposed Greedy SDS fuzzy due to its selection of global best solutions are able to efficiently allocate resources.

Table 2: *Degree of Imbalance for Greedy SDS Fuzzy.*

Number of Jobs	Fuzzy	Rule Selected Fuzzy SDS	Greedy Fuzzy	Greedy SDS fuzzy
200	2.5	2.3	2.4	2.2
400	2.4	2.3	2.3	2.2
600	2.1	2	2	1.9
800	2	1.9	1.9	1.8
1000	1.9	1.7	1.8	1.6

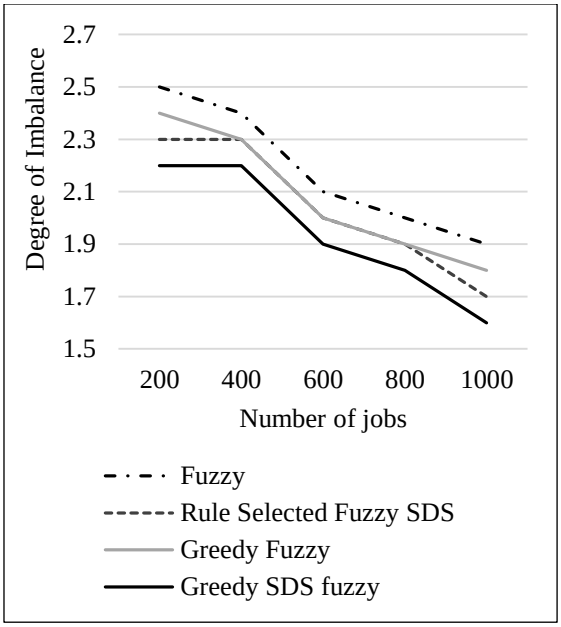

Fig. 2: *Degree of Imbalance for Greedy SDS Fuzzy.*

Table 3: *Resource Utilization for Greedy SDS Fuzzy.*

Number of Jobs	Fuzzy	Rule Selected Fuzzy SDS	Greedy Fuzzy	Greedy SDS fuzzy
200	79	82	80.7	84
400	78	81	80	82.7
600	81	84	83.4	86.3
800	82	85	83.8	87.3
1000	79	81	81.1	82.8

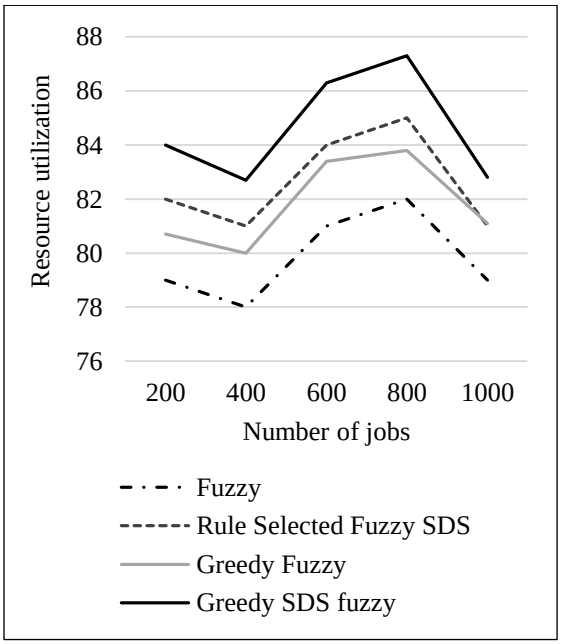

Fig. 3: *Resource Utilization for Greedy SDS Fuzzy.*

CONCLUSION

Cloud computing has enormous future prospects, but the security threats embedded in cloud computing approach are directly proportional to its offered advantages. In this work, the greedy SDS fuzzy scheduling method is proposed. Greedy approach is one of the approaches used to solve the job scheduling problem. A greedy algorithm always makes the choice that looks best at that moment. SDS is a multi-agent population-based global search and optimization algorithm; it is a distributed mode of computation utilizing interaction between simple agents. The greedy SDS fuzzy evaluation concept which can be integrated into approaches for hard combinatorial optimization problems. The proposed method evaluates objects in a way that combines fuzzy reasoning with a greedy-SDS mechanism, thereby exploiting a fuzzy solution space using greedy methods. Results show that the greedy SDS fuzzy has higher resource utilization by 6.13%, 2.4% and 4% for 200 number of jobs, by 5.84%, 2.07% and 3.31% for 400 number of jobs, by 6.33%, 2.7% and 3.41% for 600 number of jobs, by 6.26%, 2.66% and 4.09% for 800 number of jobs and by 4.69%, 2.19% and 2.07% for 1000 number of jobs when compared with fuzzy, rule selected fuzzy SDS and greedy fuzzy. In future, the existing results can be enhanced, and more number of resources can be utilized with less number of jobs by proposing the various hybrid techniques.

REFERENCES

1. Kumar VV, Dinesh K. Job scheduling using fuzzy neural network algorithm in cloud environment. *Bonfring International Journal of Man Machine Interface*, Vol.2, No.1, 1, pp.1-6, 2012.
2. Ahmed M, Hossain MA. Cloud computing and security issues in the cloud. *International Journal of Network Security & Its Applications.* 2014; 6 (1): 25–36p.
3. Kaur M, Singh H. A review of cloud computing security issues. *International Journal of Advances in Engineering & Technology*. 2015; 8 (3): 397–403p.
4. Zissis D, Lekkas D. Addressing cloud computing security issues. *Future Generation Computer Systems.* 2012; 28 (3): 583–592p.
5. Thakur P, Mahajan M. Different scheduling algorithm in cloud computing: a survey. *International Journal of Modern Computer Science.* 2017; 5 (1): 44–50p.
6. Singh RM, Paul S, Kumar A. Task scheduling in cloud computing. *International Journal of Computer Science and Information Technologies*. 2014; 5 (16): 7940–7944p.
7. Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal.* 2015; 16 (3): 275–295p.
8. Guo F, Yu L, Tian S, Yu J. A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment. *International Journal of Communication Systems*. 2015; 28 (6): pp. 1053–1067.
9. Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF, Sadollah A. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-based systems*, 2016; 109, pp. 1–16.
10. Ma L, Lu Y, Zhang F, Sun S. Dynamic task scheduling in cloud computing based on greedy strategy. In: *Proc. of International Conf. On Trustworthy computing and Services, Springer*, Berlin, Heidelberg. 2012; pp. 156–162.
11. Gupta A, Garg R. Load balancing based task scheduling with ACO in cloud computing. In: *Proc. of International Conf. On Computer and Applications (ICCA)*, IEEE, 2017, pp. 174–179.
12. Vinothina V, Sridaran R. An approach for workflow scheduling in cloud using ACO. In: *Big Data Analytics Springer*, Singapore, 2018, pp. 525–531.
13. Jamali S, Alizadeh F, Sadeqi S. Task scheduling in cloud computing using particle swarm optimization. *The Book of Extended Abstracts.* 2016; 192: 192–198p.
14. Sagnika S, Bilgaiyan SS, Mishra BSP. Workflow scheduling in cloud computing environment using bat algorithm. In: *Proc. of International Conf. On Smart System, Innovations and Computing, Springer*, Singapore, 2018, pp. 149–163.
15. Choudhary A, Gupta I, Singh V, Jana PK, A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud

computing. *Future Generation Computer Systems.* pp. 1–27, 2018.
16. Mehranzadeh A, Hashemi SM. A novel-scheduling algorithm for cloud computing based on fuzzy logic. *International Journal of Applied Information Systems* (IJAIS). 2013; 5 (7): 28–31p.
17. Sethi S, Sahu A, Jena SK. Efficient load balancing in cloud computing using fuzzy logic. *IOSR Journal of Engineering*. 2012; 2 (7): 65–71p.
18. Omran M, Moukadem I, Al-Sharhan SS, Kinawi M. Stochastic diffusion search for continuous global optimization", In: *Proc. of International Conf. On Swarm Intelligence* (ICSI), Cergy, France, 2011.
19. Vaneshani S, Jazayeri-Rad H. Optimized fuzzy control by particle swarm optimization technique for control of CSTR. World Academy of Science. *Engineering and Technology*. 2011; 59, pp. 686–691.
20. Sheibani K. Fuzzy Greedy search and job-shop problem. *In 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG). Nottingham*, UK. 2006; pp. 147–150.
21. Alhakbani H, Al-Rifaie MM. Feature selection using stochastic diffusion search. In: *Proc. of Conf. On the Genetic and Evolutionary Computation.* ACM. 2017; pp. 385–392.
22. Al-Rifaie MM, Bishop JM, Blackwell T. Information sharing impact of stochastic diffusion search on differential evolution algorithm. *Memetic Computing.* 2012; 4 (4): 327–338p.
23. Loiseau P, Wu X. Greedy and dynamic programming algorithms for scheduling deadline-sensitive parallel tasks. Technical report, available as, 2015.
24. Hassidim A, Singer Y. Robust guarantees of stochastic greedy algorithms. *In International Conference on Machine Learning*, 2017, pp. 1424–1432.
25. Sheibani K. A hybrid metaheuristic using fuzzy greedy search operator for combinatorial optimization with specific reference to the travelling salesman problem. *Iranian Journal of Operations Research.* 2011; 2 (2): 63–71p.
26. Li Y, Chen M, Dai WW, Qiu M. Energy optimization with dynamic task scheduling mobile cloud computing. *IEEE Systems Journal.* 2017; 11 (1): 96–105p.
27. Dong Z, Liu NN, Rojas-Cessa R, Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centres. *Journal of Cloud Computing.* 2015; 4 (1) 5: 1–14p.

Cite this Article
Ranga Swamy Sirisati, Sridhar Mandapati. A Greedy Stochastic Diffusion Search-based Fuzzy Scheduling in Cloud. *Journal of Artificial Intelligence Research & Advances*. 2018; 5(2): 58–68p.