Journal of Big Data

RESEARCH Open Access

Hybrid gradient descent spider monkey optimization (HGDSMO) algorithm for efficient resource scheduling for big data processing in heterogenous environment

V. Seethalakshmi[1*], V. Govindasamy[2] and V. Akila[1]

*Correspondence:
Seethaveera16@gmail.com
[1] Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India
Full list of author information is available at the end of the article

Abstract
Big Data constructed based on the advancement of distributed computing and virtualization is considered as the current emerging trends in Data Analytics. It is used for supporting potential utilization of computing resources focusing on, on-demand services and resource scalability. In particular, resource scheduling is considered as the process of resource distribution through an effective decision making process with the objective of facilitating required tasks over time. The incorporation of heterogeneous computing resources by the Big Data consumers also permits the option of reducing energy usage and enhanced resource efficiency. Further, optimal scheduling of resources is considered as an NP hard problem due to the dynamic characteristics of the resources and fluctuating users' demand. In this paper, a Hybrid Gradient Descent Spider Monkey Optimization (HGDSMO) algorithm is proposed to efficient resource scheduling by handling the issues and challenges in the Hadoop heterogenous environment. The proposed HGDSMO algorithm uses the Gradient Descentand foraging and social behavior of the spider monkey optimization algorithm involved in the objective of effective resource allocation. It is designed as the efficient task scheduling approach that balances the load of the cloud by allocating them to appropriate VMs depending on their requirements. It is also proposed as a dynamic resource management scheme for efficiently allocating the cloud resources for effective execution of clients' tasks. The simulation results of the proposed HGDSMO algorithm confirmed to be potent in throughput, load balancing and makespan compared to the baseline hybrid meta-heuristic resource allocation algorithms used for investigation.

Keywords: Heterogeneous Environment, Optimal Resource Allocation, Gradient Descent Algorithm, Spider Monkey Optimization (SMO), Load balancing

Introduction

In the recent days, Big Data has attracted a lot of attention from academia, industry as well as government as it offers substantial value to them. However, at the same time it poses a considerable number of challenges on existing infrastructure [1]. One of the most challenging issue is how to process the huge amount of data for analysis, since it is a time-consuming and labor-intensive task and hence, stretches existing infrastructure to its limits. Many studies are emerging now-a-days to explore the possibility of using cloud computing paradigm for Big Dataprocessing [2]. Those works are driven by a fact that the Big Data Processing requires scalable and parallel computing resources rather than using on-hand database management tools or traditional data processing applications. The core aim of scheduling in Big Data Processing completely focuses on the plan of processing and completing diversified tasks as much as possible based on a restricted number of data handling and alteration achieved in an effective manner. In general, different methods are highly preferable for resource allocation since they possess specialized architectural properties. In this context, identifying the best scheduling method for each and every specific data processing is considered as the important challenge [3]. This challenge is even more complex as the Big Data processing is considered as the largest batch tasks that run over a High Performance Computing (HPC) cluster by partitioning a job into smaller tasks for the purpose of distributing the work to the cluster nodes. However, the Big Data processing models need to be aware of the locality in which the data resides under the event of transferring the data to the nodes used for computation. Currently, the jobs are practically allocated to each computing node based on the two processes. The two processes are processes of investigating realistic contexts of utilizing resources and the static or dynamic scheduling enforced in Map Reduce clusters. Further, the job scheduling process can estimate the resource utilization associated with each allocated jobs, which may not be achieved by investigating the completed jobs.

The area of Big Data possesses a diversified number of research challenges that includes data volume handling, gig data analysis, security and data privacy, data visualization, storage, fault tolerance, job scheduling and energy optimization. The Big Data analysis is more difficult due to the heterogeneous and incomplete nature of data product. This challenge is also due to the availability of different structures, variations and formats of the collected data [4]. Furthermore, the process of dynamic scheduling of jobs with demands of distributed computing also necessitates resource scheduling across diversified geographical areas. In specific, Hadoop uses round method of scheduling when the number of smaller priority jobs is comparatively higher than the number of higher priority jobs. The scheduler also enforces weight and dynamically update rules based on the estimation of the situations. This Hadoop platform uses the aforementioned approach for job tracking and task allocation in the Big Data heterogeneous environment. However, Hadoop scheduler suffers from performance degradation under heterogeneous environments for the purpose of resolving the limitations that are more common in job tracking and task allocation. At this juncture, LATE (Longest Approximate Time to End) is utilized in the Hadoop environment for introducing high robustness under task scheduling in the heterogeneous environment [5]. It is identified that, the response time of the Hadoop increase with the incorporation of LATE scheduler. The method of delay scheduling is a method inherited in Big Data for enhancing the locality of data in an effective and efficient way. This delay scheduling process also concentrates on improving the throughput for the diversified type of tasks. It also ensures fairness in processing and completing tasks based on its policies and simplicity involved in sharing the resources in the Hadoop heterogeneous environment.

In this context, computing services are considered to possess virtual data centers that are highly optimized for facilitating software, hardware and information resources for utilization depending on the demand requested from the users [6]. However, Hadoop environment handles the fluctuations in workload and enables resources for computing and managing huge amounts of multimedia data and multimedia development environments in most of the circumstances [7]. Moreover, the increase in the number of more and larger datacenters introduces new challenges at the infrastructure management and monitoring level [8]. At this juncture, resource scheduling is determined to be a vital procedure for making conclusive decisions on the distribution of resources over time [9]. However, these resource scheduling problemsalso pose certain challenges due to the dynamic characteristics of the resources and fluctuating demands of the users [10]. Moreover, the fitness function computed for resource scheduling concentrates on the perspective of the objectives associated with users and providers [11]. From, the providers' dimension, they need to improve resource utilizations with resources available in the environment for focusing on the increase in profit and revenue growth [12]. On the other hand, users concentrate on the process of deriving maximum performance from their requisite services with reduced expenditure and cost [13]. In the literature, most of the resource scheduling algorithms were identified to consider the parameters of cost, load balancing, availability, throughput, reliability, makespan, energy and fault tolerance for confirming optimal scheduling and utilization process. In addition, most of the meta-heuristic resource scheduling algorithms proposed in the literature are identified to be potent in local searching or global searching process [14]. Thus, the hybridization of a potential local searching approach with the globally capable global optimization schemes is essential [15]. In this proposed scheme, a Hybrid Gradient Descent Spider Monkey Optimization (HGDSMO) algorithm is presented by integrating the local ability of Gradient Descent (GD) and global potential of the Spider Monkey Optimization (SMO) algorithm for effective and efficient resource scheduling process in the heterogeneous Hadoop environment.

The major contributions of the proposed HGDSMO resource allocation algorithm are as follows:

a) Hybridization of Gradient Descent and Spider Monkey Optimization Algorithm for the optimal resource allocation process in heterogeneous environment.
b) Formulation of objective functions for optimal resource scheduling through mathematical models that derive throughput, makespan and imbalance degree.
c) Design of HGDSMO resource allocation algorithms for the purpose of addressing the issue and challenges inherent with the proposed scheduling models.
d) Implementation of the proposed HGDSMO resource allocation algorithm using the Hadoop simulation tool.
e) Performance Investigation of the baseline meta-heuristic algorithms with the proposed HGDSMO algorithm by including the matrices of throughput, imbalance degree and makespan.

Related work

In this section, the complete categories of Meta-heuristic resource scheduling approaches [8] contributed over the decades are presented in Fig. 1. This section also presented the comprehensive review of the state-of-art Meta-heuristic resource scheduling approaches proposed in the recent years with an extract of literature depicting the shortcomings of the literature.

A Cuckoo Search meta-heuristic algorithm-based resource scheduling was proposed for heterogeneous environments [16]. This CS-based resource scheduling scheme used the factors of throughput, makespan and response time for estimating the performance. Then, an enhanced Multi-Objective Cuckoo Search Optimization (MOCSO) Scheme was proposed for handling the issues of resource scheduling in heterogeneous environments as portrayed in Fig. 1. The core objective of MOCSO scheme focuses on minimizing the cost of the user and improving the performance by reducing the time of makespan. The minimization of makespan in MOCSO in turn concentrates on enhancing the profit or the revenue for providers with maximized utilization of resources [17]. This MOCSO approach is also potent in solving resource scheduling issues through the formulation of multi-objectives in Big Data. It balances the multi-objectives through the determination of two factors that includes expected completion time and expected cost for computing completion matrices. A Global League Championship Algorithm (GLCA) was proposed for the effective task of resource allocation through the enforcement of scientific applications [18]. The simulation results of the GLCA scheme derived through Hadoop simulator confirmed its potential in enhancing the makespan to a maximum level of 14.44–46.41%. This GLCA scheme is also proved to be reducing the response time under effective resource scheduling processes. This GLCA scheme also confirmed a superior performance over the compared Genetic Algorithm, Min–Min, Ant Colony and Max–Min optimization algorithm-based scheduling approaches.

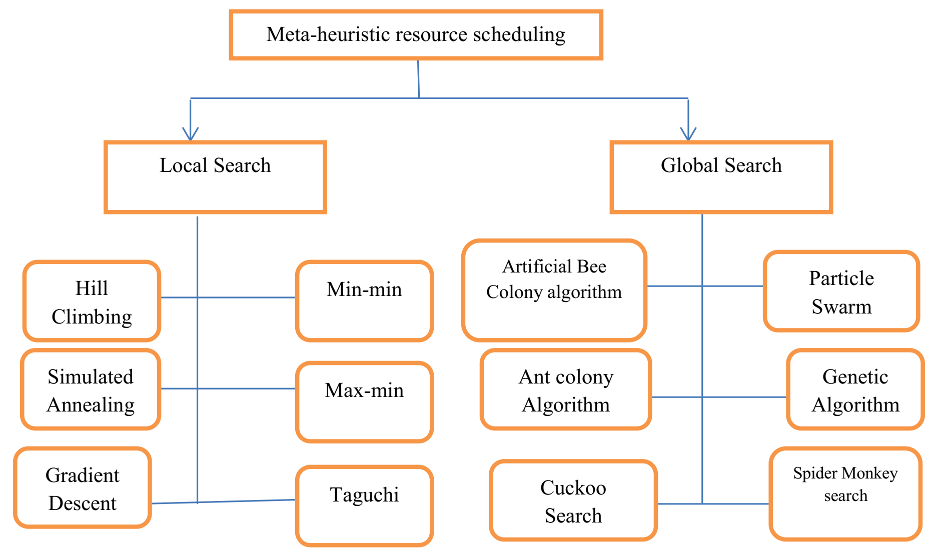

Fig. 1 The Existing schemes

A resource scheduling algorithm-based on Discrete Symbiotic Organism Search (SSOS) was proposed to improve the task of load balancing and resource sharing in a heterogeneous environment [19]. The characteristic properties of SSOS such as commensalism, mutualism and parasitism were contributed to achieve the objective of optimizing resource scheduling. This SSOS technique was identified to possess faster convergence rate in order to make it adaptable to large scale scheduling issues. An integrated PSO and ACO-based task scheduling optimization scheme was proposed in a Hadoop heterogeneous environment for handling the inadequacies realized under the deployment of Heterogeneous environment [20]. This integrated PSO and ACO-based resource scheduling scheme was determined to be potent in sustaining the particles under the fitness level with specific concentration. This PSO-ACO scheme was capable in guaranteeing the population diversity to the maximum degree for achieving global best solution. A Genetic Algorithm-based resource allocation technique was proposed for reducing the completion time of the task through the enforcement of potent allocation decisions [21]. The simulation experiments conducted for this GA-based resource allocation technique using Hadoop tool confirmed its predominance over the simple and greedy resource allocation schemes in terms of throughput and makespan.

A hybrid GA and PSO-based resource allocation approach was proposed, based on the principle of on-demand queues [22]. This GA-PSO algorithm incorporated the process of analyzing the new tasks by storing them into the on-demand queue. It also computed the priority that played the vital role in synchronized allocation of tasks into hosts or VMs. A Mean Grey Wolf Optimization Algorithm (MGWO) was proposed for improving the performance by eliminating the issues that are highly related to scheduling in the Hadoop system [23]. The objective function of MGWO algorithm concentrated on reducing the energy consumptions and makespan under resource scheduling processes. An Integrated Harmony Search and Cuckoo Search (IHS-CSO) algorithm-based resource scheduling was proposed to sustain balance between the exploitation rate and exploration related to availability of hosts or VMs [24]. This IHS-CSO scheme used energy consumptions, credit, penalty, memory usage and cost for devising objective function. It was confirmed to be superior than the standalone Gravity Search, Cuckoo Search (CS) and Harmony Search (HS) Schemes. In addition, a Hybrid Gradient Descent and Cuckoo Search Optimization (HGDCSO) algorithm was proposed for resource scheduling based on makespan, throughput and degree of imbalance in the heterogeneous environment [25]. This HGDCSO scheme was proposed for mutual integration of gradient descent and cuckoo search optimization algorithms which are capable in exploitation and exploration respectively. The parameters of makespan, throughput and degree of imbalance were used by the HGDCSO scheme for the formulation of the objective function. The number of migrated tasks during the deployment of the HGDCSO scheme was identified to be comparatively better than the existing approaches independent to the number of tasks and VMs available in the Hadoop.

The resource management scheme for Big Data processing is proposed for distributing the loads in the cloud environment [26]. It used the algorithm with the aspect of load balancer matching that derives the input based on the demands of the Big Data under processing. It was proposed a management algorithm that could work well in cloud computing scenario independent to the number of VMs and physical host machine.

It was proposed to address the issue of low availability associated with computational significances and energy. It was also identified to enhance the response time associated with the Big Data processing tasks. A Localization identity and dynamic priority-based hybrid scheduling algorithm was proposed for concentrating on the scalability issue of data locality rate with reduced completion time [27]. It was compared with default schedulers of Hadoop such as FIFO and fair that are capable of executing concurrent workloads that incorporates benchmarks of Terasoft and Word count. It was determined to rapid on par with the fair scheduling and the FIFO methodology for ensuring maximized data locality rate by preventing the resource wastages. A Binary Particle Swarm Optimization (BPSO)-algorithm based resource management scheme was proposed for handling the demand of higher computational power as it is the major challenge to energy requirement in a cloud scenario [28]. It was proposed for distributing the load of Big Data processing among the VMs in a fair manner by optimizing QoS parameters that satisfy the end user goals. It inherited the merits of modified transfer function that facilitated balanced exploitation and exploration during the process of optimizing QoS parameters. However, still there exists a room for improvement.

A Resource allocation framework was proposed for outsourcing the incoming Big Data tasks to the external clouds [29]. This architecture never used any inter-cloud agreements that are formulated for the federation of the clouds. It was proposed for benefitting the allocation of user tasks that marched towards the maximization of profits that in turn ensured a high degree of QoS. It was proposed as a reliable model of integer programming that is capable internally for updating itself towards the requirements of Big Data processing. It was also identified to minimize the computation time predominantly, but failed to handle it in the event of a large number of requests incoming to the cloud environment. In addition, An autonomic resource management approach was proposed for ensuring intelligent QoS in the event of Big Data processing [30]. This autonomic resource management approach supported resources that offered self configuration of applications, which are self healing in properties. It was proposed to provide self protection and sudden failure with resistance against security attacks and self optimizing characteristics. It also confirmed better performance at execution time, contention of resources, SLA violation and response time.

Extract of the literature

The aforementioned review conducted over the existing works of the literature confirmed the following limitations as listed as follows.

i. The majority of the existing approaches maximized data locality degree, but they are not able to reduce the time of job completion through the inclusion of higher data locality rate.
ii. Most of the reviewed approaches proposed using PSO, GA, ACO and ABC revealed that they have the limitation of trapping in the local point of optimality during the process of scheduling.
iii. The existing methodologies proposed with their own merits also possess some limitations with respect to the aspect of balancing the tradeoff between local search and global search during the process of resource management and scheduling.

Problem formulation and the mathematical model considered for resource scheduling in the proposed HGDSMO algorithm

Scheduling is the process of assigning the starting and completion times for the set of operations to be performed. Similar to other scheduling issues, resource scheduling is a method applied for effective distribution of potential resources that generally includes networks, processors, virtual machines and storages. The resource scheduling is responsible for satisfying the demands of the users under the interaction with the providers. It is highly suitable for load balancing for the purpose of facilitating the uniform distribution of resources based on the demand of the users and to provide some priority through the formulated set of rules. It also needs to ensure a heterogeneous environment, which is significant in serving the requests of the users with some specific quality of service. The problem of resource scheduling is better portrayed with the aid of Eq. (1).

$$RS_{HC} = \sum_{i,j=1}^{m,n} \left(AR_j + S_i \ldots N_i\right) \times \left(CT_i - CU_j^X\right) \tag{1}$$

where, 'm' is the number of tasks such as $CT_{(i)} = (CT_1, CT_2 \ldots\ldots, CT_m)$ that are assigned to n available physical in the data centers of the Hadoop $(AR_{(i)} = AR_1, AR_2, \ldots AR_n)$. Further, the available virtual resources in the Hadoop environment are considered to range between $S_{(i)}$ and $N_{(i)}$, respectively. In addition, the fitness value of each objective has the probability to be improved for the users.

The Hadoop environment is considered in the deployment of the proposed HGDSMO algorithm consists of different data centers. Each and every data centers in turn consist of inter-correlated VMs with diversified specification. If there is a collection of task $CT_{(i)} = (CT_1, CT_2, \ldots\ldots, CT_m)$ that are generated by the users depending on their required demands, then the Hadoop broker takes the responsibility of assigning the user tasks to the necessitated VMs $VM_{(i)} = (VM_1, VM_2, \ldots\ldots, VM_m)$ with reduced time of completion. In this context, Expected Time of Completion (ETC) is defined as the time expected for executing a complete set of task on a definite virtual resource. This ETC defined for each VM is depicted in Eq. (2).

$$ETC = \sum_{i=1}^{n} CT_{Task(i)} \tag{2}$$

The dimension of the ETC matrix $ETC(CT_i, VM_j)$ is determined by multiplying the number of tasks with the number of VMs available in a particular context of the Hadoop heterogeneous environment as presented by Eq. (3).

$$ETC(CT_i, VM_j) = \begin{pmatrix} CT_1VM_1 & CT_2VM_2 & CT_1VM_3 & \ldots\ldots\ \ldots\ldots & CT_1VM_m \\ CT_2VM_1 & CT_2VM_2 & CT_2VM_3 & \ldots\ldots\ \ldots\ldots & CTVM \\ CT_3VM_1 & CT_3VM_2 & CT_3VM_3 & \ldots\ldots\ \ldots\ldots & CTVM \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots\ \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots\ \ldots\ldots & \ldots\ldots \\ CT_nVM_1 & CT_nVM_2 & CT_nVM_3 & \ldots\ldots\ \ldots\ldots & CT_nVM_m \end{pmatrix} \tag{3}$$

Thus, the core objective of the proposed HGDSMO algorithm is to integrate the approaches of GD into the local searching process of the optimized SMO algorithm for effectively mapping the tasks on VMs with reduced Expected Time of Completion (ETC). Minimized ETC is essential for attaining reduced makespan, stable imbalance degree and enhanced throughput.This proposed HGDSMO algorithm considered the throughput, degree of imbalance and makespan as the objective function for enabling optimal process of resource scheduling in Hadoop. Hence, the fitness value of the proposed HGDSMO algorithm is computed based on throughput, degree of imbalance and makespan as represented in Eq. (4, 5, 6), respectively.

Throughput

$$f(x) = \sum_{i-1}^{n} CT_{TASK(i)}, \ \forall i \in N, \ 1 \le i \le n \tag{4}$$

Degree of imbalance

$$f(x) = \frac{\bigcup_{i=1}^{n} MaxCT_{Task(i)} - \bigcup_{i=1}^{n} MinCT_{Task(i)}}{MeanCT_i} \tag{5}$$

Makespan

$$f(x) = Max \bigcup_{i=1}^{n} CT_{Task(i)}, \forall i \in N, \ 1 \le i \le n \tag{6}$$

where $CT_{Task(i)}$ represents the time of completion associated with a specific task.

In this context, the stabilized and smooth imbalance degree with reduced makespan and increased throughput depicts the vitality of the proposed HGDSMO algorithm.Hence, this proposed scheme focuses on reducing the time to completion of the tasks on VMs for achieving reduced makespan, stable imbalance degree and enhanced throughput.

Methodology

In this section, the description of local and global search optimization algorithms, the primitive structure of Gradient Descent (GD) method, standard Spider Monkey Optimization (SMO) algorithm and the proposed HGDSMO algorithm is presented.

Comparison of local search and global search

A search method which always reaches a similar local optimal solution from the same point of starting is probably considered as a local search method. Likewise, the global search method's performance needs to be less dependent on its initial position. The local search method will focus on the nearby local optima and the global search technique need to be capable of identifying local optima at any particular point in the search space. However, the local search and global search concentrate on estimating a solution that plays a key role in optimizing the criteria of cost. In specific, local search methods initiates the process of exploring the state space at any arbitrary point and iteratively attempts in estimating a better solution evaluated in terms of the cost function. The arbitrary point used by the local search methods can be chosen through the utilization of the huge number of techniques that highly relies on the problem domain and the local search strategy.

Traditionally, the local search techniques are faster compared to the global search methods and they are potential in facilitating quite superior solutions when the step of initialization is adequate to the concerned problem domain. Moreover, these local and global search algorithms are iterative in nature and they always focus on identifying the superior estimated solutions possible at each current iteration. These algorithms provide us complete freedom for selecting the termination criteria. These algorithms aids in providing local optimal points which may incur a much higher cost compared to the global optimal points, which also relies on the initial solution from where the process of exploration is initiated. But ideally, the global search method completely focuses on estimating the best global solution which is attained mostly at the expense of long searching time. However, they are execution and termination when the criteria of termination come across in reality. Some specific instances of this global search includes GA, SA, PSO,ACO, ABC,etc. The local search algorithms do not completely concentrate on search, but it tries to transform itself from a current solution to a neighboring updated solution. This transformation of current solutions by the local search algorithms depends on the initial solution and initial search space. An instance of a local search method is the hill climbing algorithm, which initially starts with a random solution and iteratively attempts to identify the optimal solution by incrementally transforming a single element of a solution. If the transformation of single element from a solution yields a better solution, then an incremental transformation of the elements in the solutions is facilitated for constructing newer solutions. This process of incremental transformations is repeated until the solution could be no more improved. Thus, there exist NP problems in which identifying one definite and one optimal solution is not feasible. From the viewpoint of classification, these NP problems differ from the methods that are generally utilized for locating a solution. At this juncture, any method that explores towards the vicinity of a initial starting point and has the possibility to get trapped in the local optimal point is always considered as the local search method. One best example is the Gradient Descent method. The global method treats the complete feature space as a single unit under the process of estimating the best optimal solution. A suitable example of this category would be the exhaustive search.

The local search method of Gradient Descent

The local search method of Gradient Descent (GD) is the iterative optimizing approach of principal order. GD is primarily utilized for identifying a functions' local minimum, when one moves in step proportional to the negative of the function gradient at the current point of search. In contrast, if one moves in steps proportional to the positive of the function gradient for attaining its local maxima then it is termed as Sleepest Ascent Approach. If a multi-variable function $G(x)$ is unique and differentiable in the neighboring points of r with $G(r)$ identified to decrease rapidly from the point in the direction of the negative gradient of $G(r)$ at the point $(r, G(r))$, then it is determined as the method of Gradient Descent. It states that the next position S to the current position r is defined by Eq. (7).

$$s = r - \delta \nabla G(r) \tag{7}$$

where, $\nabla G(r)$ is the steepest ascent with δ as the weight factor. In this context, the condition $G(r) > G(s)$ needs to be satisfied for ensuring the minor sufficiency level of the weight factor δ. In other words, $\delta \nabla G(r)$ is subtracted from r, since the search process wants to move against the gradient concentrating towards the minimum local point of optimality. Keeping this in mind, a sequence $a_0, a_1, a_2, \ldots$ from a guessed arbitrary point a_o with the local minimum of the function G is generated based on Eq. (8, 9) respectively.

$$G(b_i) = \alpha\ (a_i - b_i)^2 + \beta\ (b_i - b_{i+1})^2 + \beta\ (b_i - b_{i-1})^2 \tag{8}$$

$$b_i^{'} = b_i + 2\alpha\ (a_i - b_i) + 2\beta\ (b_{i+1} - b_i - 2b_i)^2 \tag{9}$$

Such that a_0 satisfies the condition $G(a_2) \geq G(a_1) \geq G(a_2) \ldots\ldots$

Thus, the sequence starting from a_0 hopefully converges towards the expected local point of optimality. It is interesting to note that the step size S_C is permitted to dynamically change with each and every iteration. With specific assumptions related to the utilized function G and step function S_C the process of convergence facilitated by GD towards the minimum point of the locality can be guaranteed.

The Spider Monkey Optimization (SMO) algorithm

The Spider Monkey Optimization (SMO) algorithm is considered as one of the recently proposed nature inspired stochastic optimization method. SMO particularly is determined to be the superior in the category of swarm intelligent approaches. Spider moneys refer to the monkey species which pertains to the category of animals associated with the fission and fusion social structure. These spider monkeys always live in groups and portray intelligent foraging behavior during the food searching process. They facilitate the food searching in diversified directions by an appropriate information sharing process with other members of the group. This SMO algorithm was proposed based on the inspiration derived from the intelligent food foraging strategy followed by spider monkeys. The search space (the complete set of data centers, hosts and virtual machines (VMs)) of the optimization problem (resource scheduling problem) is considered as the food searching area of the spider monkeys. Each solution (the subset of hosts or VMs that has the possibility to be allocated to the tasks) is considered as the spider monkeys' position in the food searching area. The complete collection of solutions (subset of hosts or VMs) that has potential of facilitating resource scheduling is termed as the swarm. Fitness of a solution (the fitness value depicts the availability of hosts or VMs derived using throughput, degree of imbalance and makespan at a specific point of time depending on the number of tasks entering into the hadoop processing environment) refers to the degree to which the spider monkey is nearer to the food source.

This SMO algorithm includes a set of information rules that aids in sharing information and continuous learning (sharing information and continuously learning about the under-utilization and over-utilization of VMs) for potential updating of positions in the complete search space. This SMO algorithm updates its current position (current VM allocated with tasks) with a superior one (new VM to be allocated) based on the experience probability of VMs thresholds in processing tasks in a hadoop heterogeneous environment. It uses four used-defined factors such as maximum number of groups, perturbation rate, local leader limit and global leader limit. Similar to the other meta-heuristic approaches, SMO is also initiated with random initial positions of the spider monkeys (the location of hosts or VMs) generated in a uniform manner. The positions of the spider monkeys keeps on updating (the allocations of tasks to the hosts or VMs is updated) in each and every iteration. The complete set of hosts or VMs are partitioned into smaller groups, when the improvement in the global leader is not identified (global threshold limit). The potential VMs or hosts pertaining to each of the partitioned groups constitute the local leader (local threshold limit). However, the number of groups is only one with same local and global leader (local threshold and global threshold limits are the same) during initialization). Moreover, six iterative steps such as local leader phase, global leader phase, Learning phase of local leader, Learning phase of global leader, deciding phase of local leader and deciding phase of global leader in addition to the initialization phase is utilized for improving the allocation of hosts or VMs to the incoming tasks of the hadoop environment. Each of the aforementioned phases has their own objectives that concentrate on the significant execution of the complete SMO algorithm while focusing on the task of resource scheduling. The local and global leader phase of SMO is responsible for generating a new trial position of each spider monkey (position of hosts or VMs). If the currently generated position is superior to the existing position, the spider monkey replaces the old position with the new position and informs all the members of the groups (the allocation probability of newly identified VM is better than the allocation probability of currently used VM). The learning phase of local leadership and global leader is mainly for identifying the potential leader who can control the entire group as well as the divided local groups (identifying the VMs with high allocation probability from the complete set of VMs and determining the VMs with high allocation probability after partitioning them into subsets). In addition, the deciding phase of local leaders and deciding phase of global leader is included for verifying and resolving the issues of premature convergence and stagnation that are quite common in local and whole search space.

A short description about the various phases of SMO is explained as follows.

Local leader phase

The process of updating the current search space which is to be explored in subsequent iterations is achieved by this local leader phase. This phase constructs a new trail position for each of the spider monkey (the position of hosts or VMs) for effective updating of search space. This new trail position is determined with the aid of local leader position, current position and randomly selected entity of that group. Moreover, the possible dimensions of the solutions in the search space has the feasibility of being updated based on the value of perturbation rate. The generation of the newly constructed trail position is achieved based on Eq. (10).

$$SM_{POS(New,j)} = \left\{ \begin{array}{cc} SM_{POS(i,j)} + r(0,1) * (LLR_{POS(i,j)} - SM_{POS(i,j)}) & \\ +r(-1,1) * (GLR_{PSO(i,j)} - SM_{POS(i,j)}) & if \ \ r(0,1) \geq PER_{rate} \\ SM_{POS(i,j)} & if \ \ r(0,1) < PER_{rate} \end{array} \right\} \tag{10}$$

where, $r(0,1)$ is the randomly value generated for different dimensions of investigation in the search space with as the rate of perturbation included in the search space. In this context, if the fitness value of the spider monkey in the newly generated position is identified to be greater than the fitness value of the spider monkey in the currently existing position, then the newly generated location is utilized. Else, the current of the spider monkey is retained.

Global leader phase

Similar to the local leader phase, the global leader phase is also responsible for updating the current search space. But, the process of updating global leader phase is achieved in a different way. This phase concentrates purely on the updating process of search solution by considering anyone randomly chosen dimension. Thus, the spider monkey (hosts or VMs) that gets the probability of being updated by the randomly chosen single dimension depends on the probabilistic value derived based on Eq. (11).

$$PR_{Select} = 0.9 * \frac{Fit_{(i)}}{Max - Fit} + 0.1 \tag{11}$$

At this juncture, the trial position is again generated based on Eq. (12).

$$\begin{aligned} SM_{POS(i,j)} &= SM_{POS(i,j)} + r(0.1) * (GLR_{POS(i,j)} - SM_{POS(i,j)}) \\ &\quad + r(-1,1) * (SM_{Random(i,j)} - SM_{POS(i,j)}) \end{aligned} \tag{12}$$

Then, the fitness value of the existing hosts or VMs and the newly generated position of hosts or VMs are compared in order to identify the best one from the search space for the potential adoption process.

Learning phase of global leader

In this phase, if the position of a spider monkey (host or VM) has the superior fitness value compared to all other monkeys (hosts or VMs) in the population space, then that monkey (host or VM) will be selected as the global leader. Further, if the selected global leader position is not getting updated then the global limit trial is incremented by 1. This global limit trial is the significant factor for keeping track of the number of iterations in which the global leader is not updated.

Learning phase of local leader

The local leaders are selected in this phase for each and every individual group as similar to the global leader by the enforcement of greedy selection. Further, if the selected local leader position is not getting updated then the local limit counter is incremented by1. This local limit counter is utilized for tracking the number of iterations in which the local leader is not updated.

Deciding phase of local leader

In this phase, the local limit counter is utilized for checking whether there is any possibility of premature convergence or stagnation in the group, such that re-initialization of the local leader for possible updating of the search space could be initiated.

Deciding phase of global leader

This phase uses the global limit counter for checking whether there is any possibility of premature convergence or stagnation in the entire population, such that the partition of population into groups could be initiated.

Significance of SMO algorithm

This SMO algorithm essentially utilizes two searching potentialities that are the local limit counter and global limit counter in the local and global search process receptively. The local search of SMO is applied very insensitively with approximately $\frac{1}{5}th$ of the search time and the global search process for the remaining $\frac{4}{5}th$ of the search time. This clearly depicts that the search is explored more effectively on the global scale to the maximum degree of 80% compared to the 20% of the exploitation process facilitated in the local scale. This SMO algorithm is determined to a significant method due to the potential reduction in convergence rate and the number of iterations. It eliminates the issue of decreased output data by increasing the number of input data and the number of computations by conducting local search and global search of the SMO algorithm in the simultaneous point of time. The aforementioned reasons form the justification behind the motivation for the researchers to focus in this optimization process for the objective of deriving optimal results.

HGDSMO algorithm

The existing meta-heuristic approaches proposed in the literature for optimizing the problems of resource scheduling in heterogeneous Hadoop is considered to possess some shortcomings that lead to reduced speed and accuracy in task processing activities. Each of the proposed meta-heuristic approaches of the literature are potent in problem optimization of resource scheduling in some scenario, but not all the time due to its less adaptability in handling the behavioral change introduced by the incoming amount of heavy task load. The unbalanced handling of emerging load entering by the existing meta-heuristic resource handling approaches in the Hadoop heterogeneous environment indirectly impacts the objective function and performance. A considerable number of optimization algorithms were proposed in the literature from the recent past for effective resource scheduling in heterogeneous Hadoop, but not even a single approach is convenient in resolving the complete set of issues that are more common in optimizing the factors that influence the process of resource scheduling. Hence, HGDSMO algorithm is proposed for preventing the limitations that are possible during the process of effective and efficient resource scheduling in the heterogeneous Hadoop environment.

This proposed HGDSMO algorithm uses GD for achieving rapid optimization and foraging behavioral capabilities of the SMO algorithm for maintaining global optimum.

The proposed HGDSMO scheme utilizes the sensible organization of local and global search, and controls the searching process through the utilization of switching factor S_α. The local search of the proposed HGDSMO scheme is achieved with the computation of GD methodology based on Eq. (7, 8) or (9) for evaluating fitness value. However, the global search process of the proposed HGDSMO scheme is facilitated using the computation of Levy flights based on Eq. (13). The levy flights is considered a random walk in which the steps are formulated with respect to the step length, which are potentially distributed based on the distribution of heavy tailed probability. Moreover, the step direction of the levy flights is considered to be random and isotropic.

$$S_i^j = S_i^{j+1} + \alpha \oplus Levy(\delta) \tag{13}$$

where, S_i^j and S_i^{j+1} represents the newly generated solution and current solution in the search space with $\alpha \oplus Levy(\delta)$ as the probability of a transaction.

In the subsequent section, the Pseudocode of the proposed HGDSMO scheme is presented in Fig. 2. The flowchart of the proposed HGDSMO scheme is highlighted in Fig. 3.

The first step of the proposed HGDSMO scheme is the initialization phase in which the number of hosts or VMs are randomly initialized. Fitness quantifies the availability degree of the hosts or VMs for the objective of task processing. If the availability of the hosts or VMs changes depending on the number of tasks to be processed, then their availability degree is re-identified.

Further,categorize the number of hosts or VMs into groups based on their degree of availability that reactively changes with the rate of incoming tasks to the Hadoop heterogeneous environment. Re-initialize the groups based on updated fitness value until the maximum number of iterations is over or the termination condition is attained.

Simulation results and discussion

The simulation experiments of the proposed HGDSMO scheme are conducted using Hadoop simulator. The simulation parameters used under the implementation of the proposed HGDSMO and the benchmarked HGDCSO, IHS-CSOand MGWO schemes are portrayed in Table 1. The Hadoop task scheduler used for processing Big Data is suitable in considering the data transmission overhead that exploits the principle of data locality. The time of task completion is influenced by the storage device's speed that are related to Hard Disk Drives (HDDs) and Solid State Drives (SDDs) in which the data is stored on heterogeneous clusters. The poor utilization of speed devices due to the ignorance of the different storage devices' speed is also an important issue that needs to be addressed for classifying storage categories and scheduling strategy. The key idea of scheduler in a heterogeneous Hadoop environment concentrates on the priorities of different classes for the purpose of minimizing the time of execution. The scheduling process generally enhances the rate of locality by considerably reducing the processing time of tasks mapped through the reduction of network traffic, since it includes minimizing tasks mapped for fetching data in a remote way. The process of managing clusters of Hadoop with multiple numbers of Map Reduce tasks

```
Input: Initial Population $a_i (1 \le i \le n)$, Switching probability $S_\alpha$ and probability of the transaction $\delta$.
Output: Best Solution $Best_{S(i)}$.
    1. Formulate the objective function $f(a)$ with $a = (a_i, .........a_d)^T$. //the objective function formulated using makespan, throughput and degree of imbalance//
    2. Consider the initial population of n spider monkeys $a_i$ with $1 \le i \le n$, Switching probability $S_\alpha \in [0.1]$ and the maximum number of iterations. // the number of hosts or VMs that could be possibly utilized for resource scheduling process//
    3. While (($t < Iteration_{Max}$) or (Termination Condition)) do // the objective function formulated using makespan, throughput and degree of imbalance//
        3.1 Randomly select each spider monkey (host or VM) say $i$ using GD based on Equation (7), (8) or (9) for evaluating fitness value.
        3.2 Verify $G_i = f(a_i^{t+1})$ for evaluating its fitness or quality $G_i$.
    4. While Fitness of $a_i$ is not improved at (($t < Iteration_{Max}$) or (Termination Condition)) do
        4.1 Divide the complete set of population $a_i$ with $1 \le i \le n$ into '$k$' groups
        4.2 Initiate the learning phase of local group leader for updating the initial population based on the complete set of dimensions used for fitness evaluation.
        4.3 Initiate learning phase of the group leader for updating the initial population based on randomly selected single dimension.
        4.4 Initialize global trial counter and local trial counter
5   If ($G_i > G_j$) then
    5.1 $G_j \leftarrow G_i$ //The old solution is replaced by the new solution
    5.2 End If
6   If ($r[0.1] < S_\alpha$) then
    6.1 Re-initialize the group and the entire population
    6.2 Apply Levy flights for facilitating superior global solution.
    6.3 Increment the global and local trial counter by 1.
    6.4 End If.
7   If ($G_i < G_{MIN}$) then //The old solution is replaced by the new solution
    7.1 $Best_{S(i)} = a_i$.
    7.2 $G_i = G_{MIN}$. //Rank the feasible solutions or groups and determine the current best solution
    7.3 Increment $t$ by 1.
    7.4 End While
8   Return $Best_{S(i)_i}$.
9   End.
```
Fig. 2 The Pseudocode of HGDSMOScheme

to be computed over multiple nodes also needs efficacy in attaining significant utilization and performance. This process of resource scheduling is also influenced by some specific issues that accounts to energy and synchronization. The Hadoop also necessitates large amounts of energy in processing the data that exists within the data center under which energy becomes the complex issue to be resolved. In addition, the complete cost of energy in the data center increases with parallel reduction in the energy consumption. Synchronization that achieves the process of transferring intermediate outputs of the mapping process to the input of Hadoop is also essential.

The performance metrics used for investigating the proposed HGDSMO schemeand the benchmarked HGDCSO, IHS-CSOand MGWO schemes are throughput, degree of imbalance and makespan as defined in [25]. The performance of the HGDSMO scheme is evaluated in five dimensions such as, (i) mean response time and execution time under varying number of tasks and executable instruction length, (ii) the migrated task count identified with different number of VMs under a constant number of tasks, (iii) the migrated task count identified with different number of tasks under a constant number of VMs and (iv) throughput, degree of imbalance and makespan under a different number of tasks.

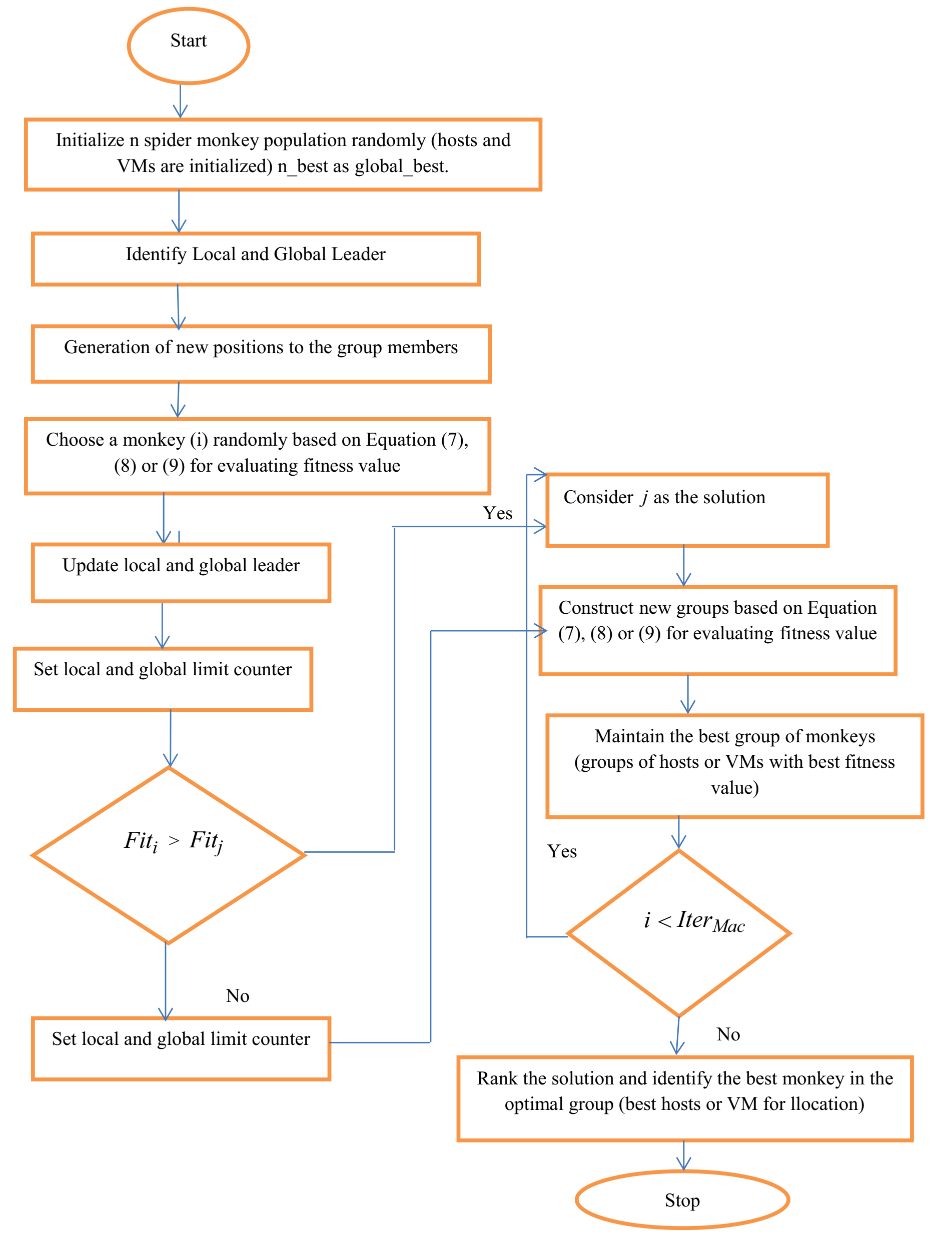
Start
Initialize n spider monkey population randomly (hosts and VMs are initialized) n_best as global_best.
Identify Local and Global Leader
Generation of new positions to the group members
Choose a monkey (i) randomly based on Equation (7), (8) or (9) for evaluating fitness value
Update local and global leader
Set local and global limit counter
$Fit_i > Fit_j$
Yes
No
Set local and global limit counter
Consider j as the solution
Construct new groups based on Equation (7), (8) or (9) for evaluating fitness value
Maintain the best group of monkeys (groups of hosts or VMs with best fitness value)
Yes
$i < Iter_{Mac}$
No
Rank the solution and identify the best monkey in the optimal group (best hosts or VM for llocation)
Stop

Fig. 3 Flowchart of HGDSMO

In the first fold of investigation, the proposed HGDSMO scheme is evaluated using mean response time under varying number of tasks and executable instruction length handled during processing. Figure 4 depicts the mean response time of the proposed HGDSMO scheme estimated under a different number of tasks. The response time of the proposed HGDSMO scheme analyzed with executable instruction length of 2000 confirmed an increase from 8.12 s to 28.12 s under respective increase in the number of tasks from 100 to 1000. Similarly, the response time of the proposed HGDSMO scheme with executable instruction length of 4000 demonstrated an improvement from 8.36 s to 34.52 s under an increasing number of tasks. Furthermore, the response time of the

Table 1 Simulation parameters considered in implementing the HGDSMO Scheme

Type	Parameter	Value
Tasks	Number of tasks	100–1000
	Length of the task	90,000
	Size of the file	600
Users	Number of users	100
	Number of brokers	5
Data center	Number of data centers	2
VM	Policy type	Time-shared
	Number of VMs	10
	RAM	512 MB
	Size	10,000
	Operating system	Linux
	VMM	Xen
	Number of CPUs	1 on each VM
	Bandwidth	10,000
	MIPS	1000
Hosts	Storage	10,00,000
	RAM	2048 MB
	Bandwidth	10,000

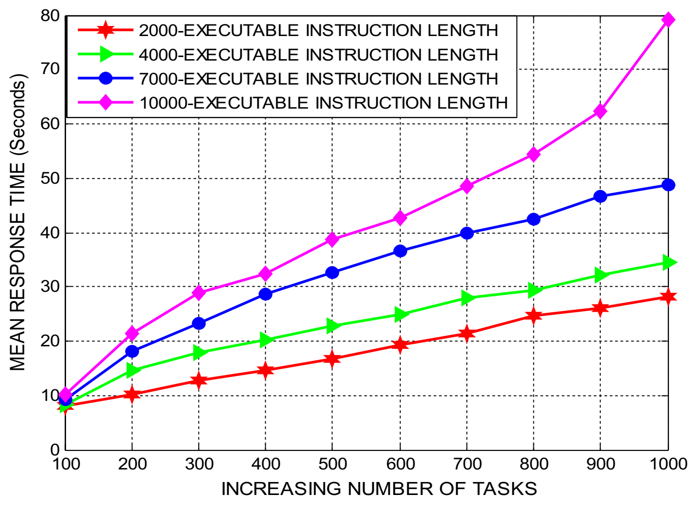

Fig. 4 HGDSMO Scheme Mean Response Time with different Number of Tasks

proposed HGDSMO scheme analyzed with executable instruction length of 7000 also confirmed a significant increase from 9.15 s to 48.94 s with an increase in the number tasks. In addition, the response time of proposed HGDSMO scheme with executable instruction length of 10,000 also confirmed a mean response time increasing from 10.26 s to 79.12 s under an increasing number of tasks. This vital improvement in mean response time facilitated by the proposed HGDSMO scheme is mainly due to the appropriate estimation of under and over utilization factor used for allocating the VMs during the process of resource scheduling. Figure 5 portrays the mean response time of the proposed HGDSMO scheme estimated under a different executable instruction ranging

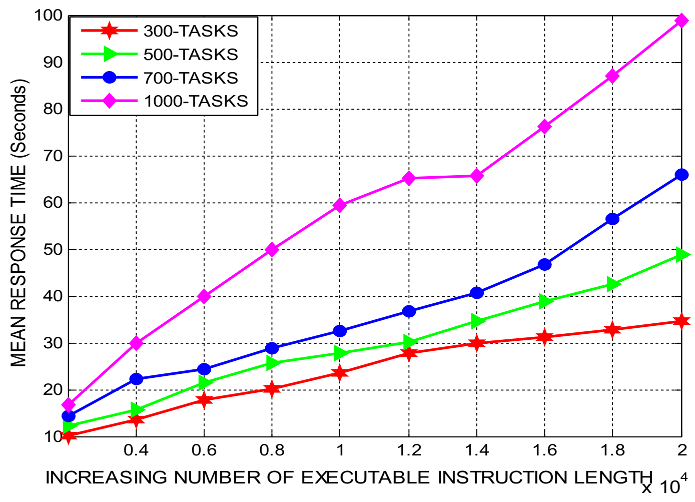

Fig. 5 HGDSMO Scheme-Mean Response Time with different Number of Executable Instruction Length

from 2000 to 20000 bytes.The response time of the proposed HGDSMO scheme analyzed with a number of tasks as 300 confirmed an increase from 10.16 to 34.68 s under respective increase in the executable instruction length. Similarly, the response time of the proposed HGDSMO scheme analyzed with a number of tasks as 500 confirmed an increase from 12.36 to 48.94 s under respective increase in the executable instruction length. The response time of the proposed HGDSMO scheme analyzed with a number of tasks as 700 confirmed an increase from 14.35 to 65.84 s under respective increase in the executable instruction length. In addition, response time of the proposed HGDSMO scheme analyzed with a number of tasks as 1000 confirmed an increase from 16.84 to 98.78 s under respective increase in the executable instruction length. This realizable enhancement in the mean response time facilitated by the proposed HGDSMO scheme is mainly due to the maximum global capability introduced by the SMO optimization algorithm for deciding about the allocation of tasks on VMs during the process of resource scheduling.

Figure 6 illustrates the execution time of the proposed HGDSMO scheme estimated under a different number of tasks. The execution time of the proposed HGDSMO scheme analyzed with executable instruction length of 2000 confirmed an improvement from 5.16 to 21.65 s under respective increase in the number of tasks from 100 to 1000. Similarly, the execution time of the proposed HGDSMO scheme with executable instruction length of 4000 confirmed an enhancement from 6.18 to 36.87 s under an increasing number of tasks. Furthermore, the execution time of the proposed HGDSMO scheme analyzed with executable instruction length of 7000 also confirmed a significant increase from 8.14 to 52.32 s with an increase in the number of tasks. In addition, the execution time of proposed HGDSMO scheme with executable instruction length of 10,000 also confirmed a increase from 18.26 to 89.74 s under an increasing number of tasks. This realizable improvement in execution time enabled by the proposed HGDSMO scheme is mainly due to the utilization of integrated GD and SMO that sustains thebalance between intensification and extensification process

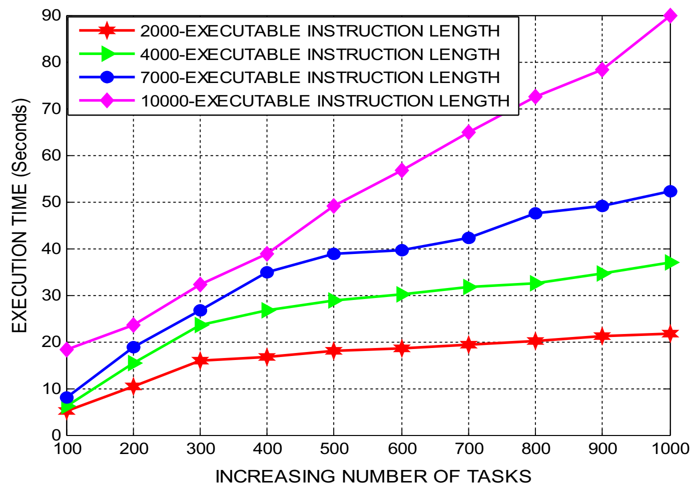

Fig. 6 HGDSMO Scheme-Execution Time with different Number of Tasks

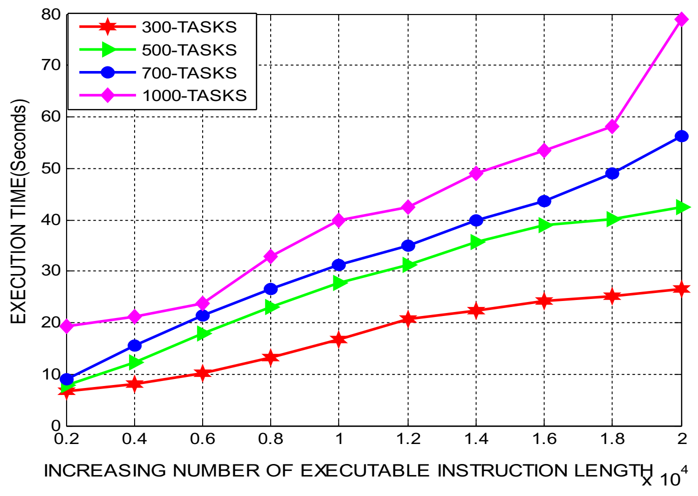

Fig. 7 HGDSMO Scheme-Execution Time with different Number of Executable Instruction Length

during resource scheduling process. Figure 7 highlights the execution time of the proposed HGDSMO scheme estimated under a different executable instruction ranging from 2000 to 20000 bytes. The execution time of the proposed HGDSMO scheme analyzed with a number of tasks as 300 confirmed an increase from 6.78 to 26.45 s under respective increase in the executable instruction length. Similarly, the execution time of the proposed HGDSMO scheme analyzed with a number of tasks as 500 confirmed an increase from 7.84 to 42.38 s under respective increase in the executable instruction length. The execution time of the proposed HGDSMO scheme analyzed with a number of tasks as 700 confirmed an increase from 9.12 to 56.34 s under respective increase in the executable instruction length. In addition, execution time of the proposed HGDSMO scheme analyzed with a number of tasks as 1000 confirmed an increase from 19.21 to 78.94 s under respective increase in the executable

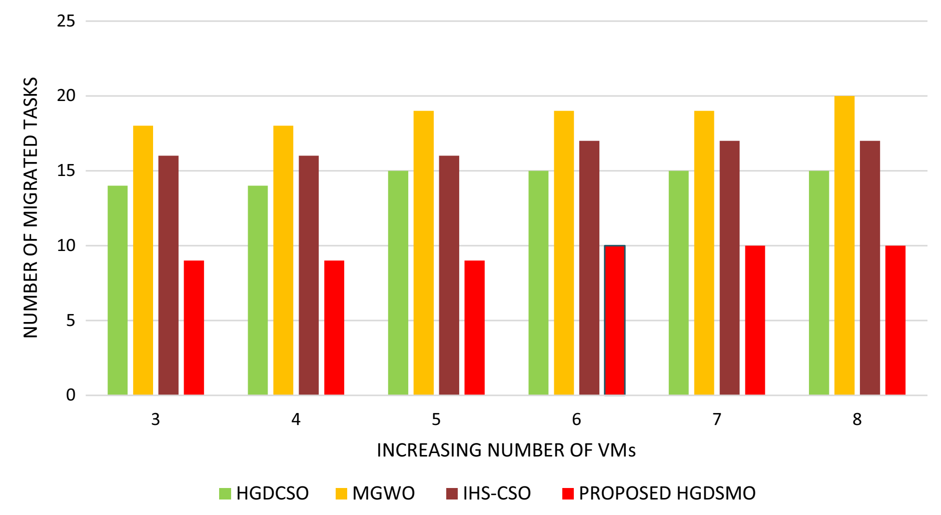

Fig. 8 HGDSMO Scheme -Migrated Tasks count with different Number of VMs (Tasks-500)

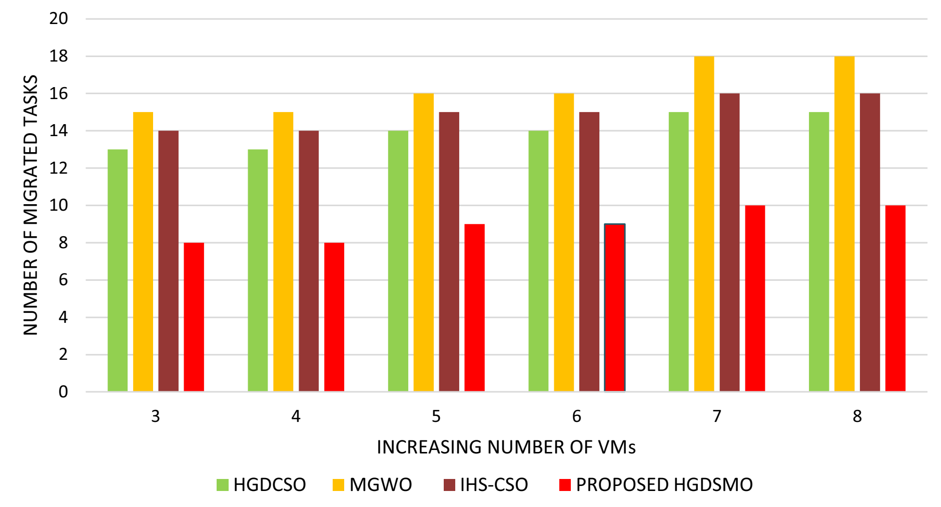

Fig. 9 HGDSMO Scheme -Migrated Tasks count with different Number of VMs (Tasks-1000)

instruction length. This improvement in the execution time enabled by the proposed HGDSMO scheme is mainly due to the switching factor and levy flights that alternates between local and global optimization process introduced by the SMO optimization algorithm in resource scheduling.

In the second dimension of investigation, the proposed HGDSMO schemeand the benchmarked HGDCSO, IHS-CSO and MGWO schemes are evaluated based on migrated task count identified with different number of VMs under a constant number of tasks. Figure 8 and 9 demonstrates the significance of the proposed HGDSMO scheme evaluated using a number of migrated tasks visualized under a different number of VMs with tasks count assigned to 500 and 1000, respectively. The number of migrated tasks after the implementation of the proposed HGDSMO scheme was identified to be predominantly reduced due to the utilization of the local and global searching process inherent with GD and SMO approaches. It also aids in marching towards significant allocation of tasks on the VMs independent to the tasks entering into the Hadoop. The number of migrated tasks after the implementation of the proposed HGDSMO scheme evaluated with different number of VMs and tasks assigned to 500, exhibits a phenomenal reduction of 6.52%, 7.18% and 8.65%, compared to the baseline HGDCSO, IHS-CSO and MGWO schemes. The number of migrated tasks after the implementation of the proposed HGDSMO scheme evaluated with different number of VMs and tasks assigned to 1000, also demonstrates a considerable reduction of 6.12%, 7.03% and 8.21%, compared to the baseline HGDCSO, IHS-CSO and MGWO schemes.

In the third dimension of investigation, the proposed HGDSMO schemeand the benchmarked HGDCSO, IHS-CSO and MGWO schemes are evaluated based on migrated tasks' count identified with different number of tasks under a constant number of VMs. Figures 10, 11 depict the predominance of the proposed HGDSMO scheme evaluated using a number of migrated tasks under different number of tasks with number of VMs assigned to 5 and 10, respectively. The number of migrated tasks with the proposed HGDSMO scheme is confirmed to be potentially minimized due to levy flights included in the global searching capability of SMO, since the objective functions are locally and globally exploited and explored independent to the number of tasks to be allocated to a specific number of VMs. The number of migrated tasks determined through the proposed HGDSMO scheme evaluated with different number of tasks and VMs assigned to 5, portrays a phenomenal reduction of 5.76%, 6.94% and 7.62%, compared to the baseline HGDCSO, IHS-CSO and MGWO schemes. The number of migrated tasks of the proposed HGDSMO scheme evaluated with different number of tasks and VMs assigned to 10, portrays a phenomenal reduction of 5.68%, 7.16% and 8.92%, compared to the baseline HGDCSO, IHS-CSO and MGWO schemes.

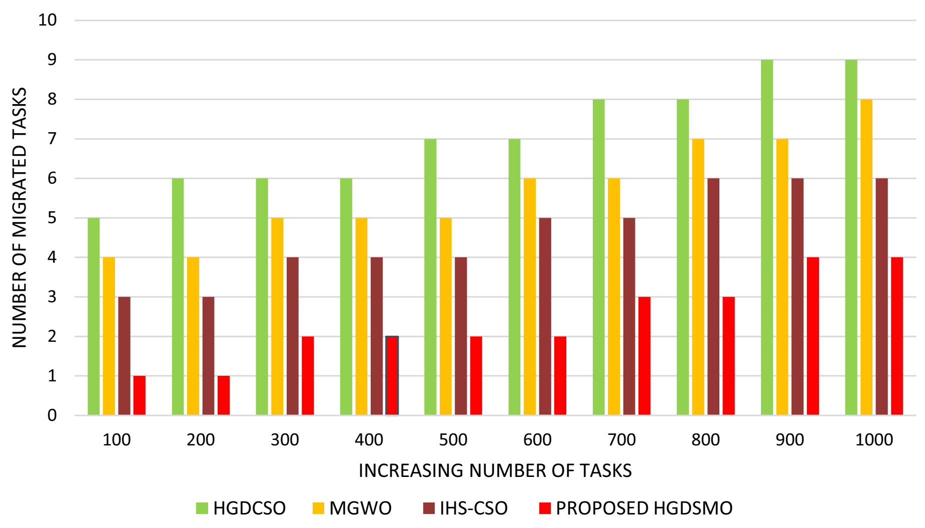

Fig. 10 HGDSMO Scheme-Number of Migrated Tasks with increasing Tasks (VMs = 5)

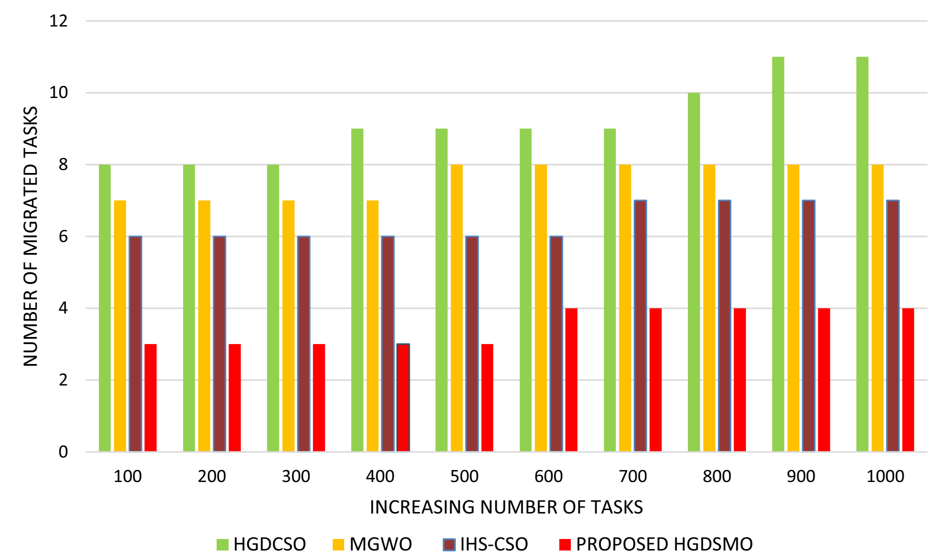

Fig. 11 HGDSMO Scheme-Number of Migrated Tasks with increasing Tasks (VMs = 10)

Table 2 The Makespan of the proposed HGDSMO Scheme with different Tasks

Number of tasks	Proposed HGDSMO	HGDCSO	IHS-CSO	MGWO
100	612.78	637.58	678.21	689.84
200	657.56	686.68	695.87	703.42
300	725.13	786.14	804.68	826.85
400	765.12	786.94	821.46	845.68
500	812.12	824.68	846.84	867.32
600	896.28	905.68	934.94	945.64
700	1032.24	1068.92	1089.12	1123.44
800	1456.78	1568.56	1678.42	1289.42
900	1678.18	1719.32	1724.62	1768.94
1000	1858.14	1952.14	1968.94	1979.42

Table 3 The degree of imbalance of the proposed HGDSMO Scheme with different Tasks

Number of tasks	Proposed HGDSMO	HGDCSO	IHS-CSO	MGWO
100	0.1374	0.1676	0.1872	0.1978
200	0.1367	0.1656	0.1845	0.1972
300	0.1345	0.1624	0.1832	0.1967
400	0.1329	0.1612	0.1824	0.1954
500	0.1307	0.1604	0.1812	0.1943
600	0.1278	0.1589	0.1802	0.1934
700	0.1267	0.1576	0.1789	0.1924
800	0.1256	0.1562	0.1778	0.1912
900	0.1204	0.1554	0.1765	0.1906
1000	0.1175	0.1532	0.1713	0.1902

Table 4 The Throughput of the proposed HGDSMO Scheme with different Tasks

Number of tasks	Proposed HGDSMO	HGDCSO	IHS-CSO	MGWO
100	2945.24	2892.24	2784.56	2656.22
200	2956.35	2893.67	2786.56	2658.92
300	2963.12	2894.78	2788.64	2712.35
400	2974.86	2898.45	2789.64	2716.89
500	2798.62	2912.68	2812.35	2718.24
600	2832.56	2926.84	2843.58	2719.48
700	2868.42	2956.84	2849.58	2846.42
800	2879.12	2976.52	2856.82	2858.96
900	2983.42	2988.14	2878.42	2869.42
1000	3078.42	2998.85	2889.54	2876.82

In addition, Tables 2, 3, 4 demonstrates the makespan, imbalance degree and throughput of the proposed HGDSMO scheme evaluated using under a different number of tasks in the Hadoop heterogeneous senvironment. The results clearly proposed that the makespan of the HGDSMO resource allocation algorithm is estimated to be excellent by 4.82%, 5.78% and 6.94%, remarkable to the compared HGDCSO, IHS-CSO and MGWO schemes. Further, the degree of imbalance of the HGDSMO resource allocation algorithm is estimated to be excellent by 4.98%, 5.89% and 7.14%, remarkable to the compared HGDCSO, IHS-CSO and MGWO schemes. In addition, the throughput of the HGDSMO resource allocation algorithm is also proved to be remarkable by 5.16%, 6.84% and 7.78%, remarkable to the compared HGDCSO, IHS-CSO and MGWO schemes.

Conclusions

The proposed HGDSMO resource allocation algorithm was contributed as a reliable attempt for achieving the objective of resource allocation with reduced energy consumptions and execution time in the hadoop heterogeneous environment. This proposed HGDSMO resource allocation algorithm used makespan, imbalance degree and throughput for the objective function that quantifies the availability of the hosts or VMs to the tasks in the hadoop heterogeneous environment. It is capable in facilitating a suitable balance between the process of exploitation and exploration through the utilization of GD and SMO with levy flights to ensure switching and enforce maximum global optimization between the process. The simulation results and statistical investigation confirmed that the proposed HGDSMO algorithm is excellent than the baseline meta-heuristic optimal resource scheduling techniques proposed for the Hadoop heterogeneous environment. The number of migrated tasks of the proposed HGDSMO algorithm is identified to be superior by 5.32%, 6.78% and 7.98%, excellent to the benchmarked HGDCSO, HCHS and MGWO algorithms with different number of tasks under a constant number of initialized VMs. The number of migrated tasks of the proposed HGDSMO algorithm is identified to be superior by 4.87%, 5.98% and 6.74%, excellent to the benchmarked HGDCSO, HCHS and MGWO algorithms with different number of VMs under a constant number of initialized tasks. It is also planned to formulate a Hybrid Gradient Descent Emperor Penguin Optimization algorithm for investigating and comparing the process of the proposed HGDSMO under resource scheduling of hosts or VMs to task processing in Hadoop.

Authors' contributions
All authors read and approved the final manuscript.

Funding
No Funding.

Availability of data and materials
Not applicable.

Competing interests
I' being the corresponding author declare that we have no competing interests.

Author details
[1] Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India. [2] Department of Information Technology, Pondicherry Engineering College, Puducherry, India.

Received: 31 March 2020 Accepted: 21 June 2020
Published online: 10 July 2020

References
1. Bose S, Sarkar D, Mukherjee NA. Framework for heterogeneous resource allocation in sensor-cloud environment. Wirel Pers Commun. 2019;1(1):43–54.
2. Rekha PM, Dakshayini M. Efficient task allocation approach using genetic algorithm for cloud environment. Clust Comput. 2019;1(1):43–56.
3. Rana N, Abd Latiff MS, Muhammad Abdulhamid S. A cloud-based conceptual framework for multi-objective virtual machine scheduling using whale optimization algorithm. Int J Innovat Comput. 2018;8(3):67–76.
4. Madni SH, Latiff MS, Ali J, Abdulhamid SM. Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds. Arabian J Sci Eng. 2018;44(4):3585–602.
5. Senthil Kumar AM, Venkatesan M. Task scheduling in a cloud computing environment using HGPSO algorithm. Clust Comput. 2018;1(2):45–56.
6. Natesan G, Chokkalingam A. Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. ICT Express. 2018;1(1):34–45.
7. Pradeep K, Prem Jacob T. A hybrid approach for task scheduling using the cuckoo and harmony search in cloud computing environment. Wirel Pers Commun. 2018;101(4):2287–311.
8. Madni SH, Abd Latiff MS, Abdulhamid SM, Ali J. Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in iaas cloud computing environment. Clust Comput. 2018;22(1):301–34.
9. Madni SH, Abd Latiff MS, Abdulhamid SM. Optimal resource scheduling for IaaS cloud computing using cuckoo search algorithm. Sains Humanika. 2017;9(1–3):56–67.
10. Chen X, Long D. Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm. Clust Comput. 2017;1(1):78–88.
11. Shojafar M, Canali C, Lancellotti R, Abawajy J. Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems. IEEE Transact Cloud Comput. 2016;1(1):1.
12. Kimpan W, Kruekaew B. Heuristic task scheduling with artificial bee colony algorithm for virtual machines. *2016* Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS), 2016: 1(1); 32–46.
13. Thaman J, Singh M. Current perspective in task scheduling techniques in cloud computing: a review. Int J Foundat Comput Sci Technol. 2016;6(1):65–85.
14. Abdulhamid SM, Abd Latiff MS, Abdul-Salaam G, Hussain Madni SH. Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. PLoS ONE. 2016;11(7):e0158102.
15. Abdullahi M, Ngadi MA, Abdulhamid SM. Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Generat Comput Syst. 2016;56(1):640–50.
16. Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. Egypt Inform J. 2015;16(3):275–95.
17. Beegom AS, Rajasree MS. A particle swarm optimization based pareto optimal task scheduling in cloud computing. Lect Notes Comput Sci. 2014;1(1):79–86.
18. Netjinda N, Sirinaovakul B, Achalakul T. Cost optimal scheduling in Iaas for dependent workload with particle swarm optimization. J Supercomput. 2014;68(3):1579–603.
19. Cho K, Tsai P, Tsai C, Yang C. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput Appl. 2014;26(6):1297–309.
20. Huang MG, Ou ZQ. Review of task scheduling algorithm research in cloud computing. Adv Mater Res. 2014;926–930(1):3236–9.
21. Tsai C, Rodrigues JJ. Metaheuristic scheduling for cloud: a survey. IEEE Syst J. 2014;8(1):279–91.

22. Mehrotra P, Djomehri J, Heistand S, Hood R, Jin H, Lazanoff A, Biswas R. Performance evaluation of amazon elastic compute cloud for NASA high-performance computing applications. Concurr Comput Pract Exp. 2013;28(4):1041–55.
23. Tawfeek MA, El-Sisi A, Keshk AE, Torkey FA. Cloud task scheduling based on ant colony optimization. 2013 8th International Conference on Computer Engineering & Systems (ICCES). 2013: 1(2); 67–75.
24. Anousha S, Ahmadi M. An improved min-min task scheduling algorithm in grid computing. Grid Pervasive Comput. 2013;1(1):103–13.
25. Li J, Peng J. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. J Comput Appl. 2011;31(1):184–6.
26. Balakrishna G, Moparthi NR. ESBL: design and implement a cloud integrated framework for IoT load balancing. Int J Comput Commun Control. 2019;14(4):459–74.
27. Gandomi A, Reshadi M, Movaghar A, Khademzadeh A. HybSMRP: a hybrid scheduling algorithm in Hadoop MapReduce framework. J Big Data. 2019;6(1):106.
28. Kumar M, Sharma SC, Goel S, Mishra SK, Husain A. Autonomic cloud resource provisioning and scheduling using meta-heuristic algorithm. Neural Comput Appl. 2020: 1(1).
29. Zuo X, Zhang G, Tan W. Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Trans Autom Sci Eng. 2014;11(2):564–73.
30. Gill SS, Chana I, Singh M, Buyya R. Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing. Clust Comput. 2017;21(2):1203–41.

Publisher's Note