Mathematics in Engineering, 6(4): 559–606.
DOI: 10.3934/mine.2024023
Received: 03 February 2024
Revised: 25 June 2024
Accepted: 31 July 2024
Published: 08 August 2024

http://www.aimspress.com/journal/mine

Research article

A novel hybrid model for task scheduling based on particle swarm optimization and genetic algorithms

Karishma and Harendra Kumar*

Department of Mathematics and Statistics, Gurukula Kangri (Deemed to be University) Haridwar, Uttarakhand 249404, India

* **Correspondence:** Email: balyan.kumar@gmail.com.

Abstract: Distributed real time system has developed into an outstanding computing platform for parallel, high-efficiency applications. A real time system is a kind of planning where tasks must be completed with accurate results within a predetermined amount of time. It is well known that obtaining an optimal assignment of tasks for more than three processors is an NP-hard problem. This article examines the issue of assigning tasks to processors in heterogeneous distributed systems with a view to reduce cost and response time of the system while maximizing system reliability. The proposed method is carried out in two phases, Phase I provides a hybrid HPSOGAK, that is an integration of particle swarm optimization (PSO), genetic algorithm (GA), and k-means technique while Phase II is based on GA. By updating cluster centroids with PSO and GA and then using them like initial centroids for the k-means algorithm to generate the task-clusters, HPSOGAK produces 'm' clusters of 'r' tasks, and then their assignment onto the appropriate processor is done by using GA. The performance of GA has been improved in this article by introducing new crossover and mutation operators, and the functionality of traditional PSO has been enhanced by combining it with GA. Numerous examples from various research articles are employed to evaluate the efficiency of the proposed technique, and the numerical results are contrasted with well-known existing models. The proposed method enhances PIR values by 22.64%, efficiency by 6.93%, and response times by 23.8 on average. The experimental results demonstrate that the suggested method outperforms all comparable approaches, leading to the achievement of superior results. The developed mechanism is acceptable for an erratic number of tasks and processors with both types of fuzzy and crisp time.

Keywords: genetic algorithm; task scheduling; *k*-means; response time; particle swarm optimization; system reliability; system cost

Abbreviations:

Indices	i, j	Task index, $i, j = 1, 2, ..., r$
	k, l	Processor index/Clustering index, $k, l = 1, 2, ..., m$
Parameters	$T = \{t_i : i = 1, 2, ..., r\}$	Set of r number of tasks
	$P = \{p_k : k = 1, 2, ..., m\}$	Set of m number of processors
	$\widehat{e}_{i,k}$	Fuzzy E_T of i^{th} task on k^{th} processor
	$FE_TM = \left[\widehat{e}_{i,k}\right]$	Fuzzy E_T matrix
	$\widehat{c}_{i,j}$	Fuzzy ITC_T between tasks t_i and t_j
	$FITC_TM = \left[\widehat{c}_{i,j}\right]$	Fuzzy ITC_T matrix
	$e_{i,k}$	Crisp E_T of task t_i on p_k processor
	$E_TM = \left[e_{i,k}\right]$	E_T matrix
	$c_{i,j}$	Crisp ITC_T between tasks t_i and t_j
	$ITC_TM = \left[c_{i,j}\right]$	ITC_T matrix
	$NE_TM = \left[e^{\hat{n}}_{i,k}\right]$	New crisp E_T matrix
	$NITC_TM = \left[c^{\hat{n}}_{i,j}\right]$	New crisp ITC_T matrix
	R_k	Reliability of processor p_k
	R_{kl}	Reliability of kl communication link
	λ_k	Failure rate of processor p_k
	μ_{kl}	Failure rate of kl connecting path
	cl_k	k^{th} task cluster
	Cd_k	k^{th} cluster centroid
	$V_k(t)$	Velocity of k^{th} particle
	$P_k(t)$	Local best fitness of k^{th} particle
	$G_k(t)$	Globally best fitness of k^{th} particle
	g_{Cd}	Fitness value of a particle in PSO
Decision variables	$x_{i,k}$	$\begin{cases} 1, & \mathit{if}\ t_i\ \mathit{task\ is\ allocated\ to}\ p_k\ \mathit{processor} \\ 0, & \mathit{otherwise} \end{cases}$
	$d_{k,l}$	Inter-processor distance with unit data transfer where $d_{k,l}$ =0 if $k=l$.
	$u_{i,k}$	$\begin{cases} 1, & \mathit{if}\ t_i\ \mathit{task\ does\ belong\ to\ cluster}\ cl_k \\ 0, & \mathit{otherwise} \end{cases}$

1. Introduction

Distributed real time system (DRTS) comprises a set of geographically distributed varied-pace heterogeneous processors that are interconnected to each other via fast communication networks. It has become a spectacular stage for computing high efficiency parallel applications. Parallel applications can be split into multiple tasks and executed simultaneously on different processors in the system. To make optimal use of this system, the key challenge lies in generating a task assignment model that assigns each task to the most appropriate processor for parallel execution. Authors Mall [1] and Jin and Tan [2] did explain that tasks in distributed system are performed in two ways, a hard real time system, and a soft real time system. Jobs are created and outcomes are generated without any time delay in hard real time system. For example, missile system, satellite system etc. (there should not be time lag). Whereas the soft real time system does not have any time limit to deliver the result or within a fixed pre-defined time. For example, web searching. Singh et al. [3] mentioned that according to the complexity of the distributed environment we need of system where multiple systems are connected and work together to optimize the goal. Typically, throughput processor usage, tasks waiting time etc., are the execution scales for the task assignment problem. In the system, if scheduling is not executed carefully, processors may take their maximum time to run the calculations. DNS (domain name system) is a simple example of DRTS that is used in a network to translate the domain name to IP address and internet [3]. The utilization of distributed system and multiprocessors is becoming very successful in real time applications. And the reason behind this is to provide the fault tolerance feature and lightning response time to the system and prices of these systems. Real time tasks' assignment in distributed environment subsists of two sub-problems: primarily partition of a set of tasks and secondary assignment of tasks to the worthy processor. Based on the time when scheduling decisions are made, there are two classes of assignment policies in DRTS, namely static and dynamic. By the static assignment of the task the permanent allocation of the task can be achieved whereas, in the dynamic assignment, the task is allotted at the time of node arrival or while the task is running. In order to acquire a simple and quick method, it is required to use approximations that yield the nearest optimum performance in a feasible amount of time. In the present work, the analysis focuses on static task assignment within a heterogeneous environment that provides diverse designing capabilities. This allocation technique can be utilized for a large set of real time applications that are able to plan a method which allows deterministic execution. Since static method does not have run time overhead and can be created applying very complex algorithmic process, it is far better than dynamic method.

As compared to centralized systems, DRTS is way more intricate. Excessive complexity can increase the likelihood of system failures. Task allocation technique and reliability play a key role in the efficient utilization of such multiprocessing system. Reliability of the system can be defined as the possibility of execution of a task having each component in working condition. The intricacy of the assignment issue in DRTS is heightened by various factors, such as the variability in task execution time, the discriminative nature of tasks, inter-dependency among tasks, and the challenge of load balancing. Various articles are assigned with the fundamental objective of reducing the total amount of communication and execution time being one of the performance parameters. In order to decrease operational costs, increase capacity, and optimize resource utilization in data centres, Jeyakrishnan and Sengottuvelan [4] developed the BSO algorithm for resource allocation and load balancing. The CSS task scheduling technique was developed by Xavier and Annadurai [5] to avoid local convergence and find the most optimal VM for tasks. This technique optimized the overall computing cost while increasing the throughput of the cloud system and there has been presented a task scheduling paradigm by Huang et al. [6] for task-VM mapping using several discrete PSO algorithm variations in cloud environment. By taking inspiration from the way crocodiles hunt, Abualigah et al. [7] developed the Reptile Search Algorithm (RSA), a brand-new meta-heuristic optimizer. The primary goal of RSA was to discover strong search techniques that can deliver higher-quality resolutions to challenging real-world issues.

In the presented article, a novel technique based on PSO-GA has been set up to solve the tasks allotment issue in a heterogeneous static environment. The GA framework can effectively deliver promising results in a broad and complex solution space, making it well suited for the scheduling issue. On the other hand, the PSO algorithm boasts easy implementation and impressive global search capabilities. Despite these advantages, the original PSO encounters limitations due to its slower convergence rate, making it unsuitable for directly tackling assignment issues. Therefore, this article has been hybridized with genetic operators to prevent the shortcomings of PSO. To reach an ideal solution and prevent early convergence, the GA employs continuous iteration. The process of convergence is iterated until it is accomplished. In current technique, the convergence of GA has been improved by presenting new genetic operations and population initialization method. Two phases are addressed in the generated approach. In the first phase a hybrid algorithm HPSOGAK, union of particle swarm optimization (PSO), genetic algorithm (GA) and k-means clustering approach has been induced. The HPSOGAK algorithm is used for the formation of clusters of tasks such that exceedingly communicated tasks assembled together in order to decrease inter-task communication time (ITC_T). In the second phase, GA is employed to assign task-clusters to processors efficiently, aiming to minimize execution time (E_T). The vital assists of the proposed technique in a distributed heterogeneous system are outlined below:

- Proposing a new task allocation model in distributed environment that takes into consideration the execution costs of tasks assigned to processors as well as the communication cost between tasks.
- In the context of task allocation onto worthy processors, PSO and GA based algorithms are presented.
- To improve the performance of GA, new crossover and mutation approaches are introduced.
- Proposed algorithm's performance is evaluated through studies based on assessment criteria like as response time, cost of the system, system reliability, performance improvement ratio (PIR %), efficiency, and resource utilization. It consistently delivers superior outcomes in these assessments.
- Analyze the run-time complexity of the proposed method.

The remainder of this article is organized as follows: The work related to this area is presented in Section 2. The definitions and terminology that will be used throughout the work are discussed in Section 3. The model of task allocation problem is explained in Section 4 and Section 5 provides more information on the proposed model's methodology. Four examples, two crisp and two fuzzy are solved in Section 6 by the proposed technique and Section 7 shows the compare of functioning of the suggested model to other existing techniques. Conclusion and recommendations for further research on this task allocation problem are provided in Section 8 and notations used throughout the paper are defined in the nomenclature section.

2. Related work

DRTS provides accurate results in both logical and temporal aspects, distinguishing itself from other types of systems. It can be roughly divided into three areas: environment, controller, and controlled object. In this process, input is received by the controller from the environment, and as an output, it provides information to the controlled object. In DRTS, task allotment is an essential phase. The primary objective of real time system is to establish an allocation model that ensures meeting all deadlines while considering execution time. Over the years, a comprehensive study of task assignment in a multiprocessing environment has led to the development of many efficient scheduling mechanisms for distributed system. Authors Davis and Burns [8] and Zhang et al. [9] came up with their work on a multiprocessing system. Casanova et al. [10] did mention that task allotment issue belongs to the category of NP- complete problem which includes the number of tasks and each task executes on the single processor and communicates with other tasks. In the proposed article, to resolve the task allotment complication, three algorithms namely PSO, GA and *k*-means clustering technique are integrated together. PSO and GA both are driven by procedures happening in nature. Inspired by the movements of bird flocks and schooling fish, Kennedy and Eberhart [11] developed a PSO technique. The particles of this computational technique have two characteristics, velocity, and position. On the basis of these two characteristics personal best and global best positions of particle of PSO are determined. Like PSO, GA [12] is one more familiar optimization technique. It is a metaheuristic approach originated by Charles Darwin's theory of "survival of the fittest". This technique refers to the natural election process, in which the most eligible individuals are elected for breeding in order to generate the succeeding generation of offspring. Clustering is a process where a group of data is assigned into smaller groups while considering that the objects in the same groups are more familiar than those in other groups. It is a heuristic approach, used to identify groups of similar objects in datasets with two or more variable quantities. It can be classified into two types viz. soft clustering and hard clustering. *K*-means clustering technique is a popular hard clustering technique where k determines how many predetermined clusters must be formed during the procedure. The task is an event which dictates the course of action and when task occurs, processing and responding done by the system accordingly. Periodic tasks, aperiodic tasks and sporadic tasks are the three categories of tasks. In a distributed environment, task assignment system is one of the elemental and exacting problems. This system plays a key role in using the resources efficiently in an economic way. Actually, allotment of tasks onto proper processors is the arrangement of tasks in such manner that several efficient constraints like system cost (CS), system reliability (RS), response time (RT) etc. are optimized. There are essentially several algorithms to achieve optimization; these algorithms are categorized into two types: dynamic and static priority algorithms. In dynamic priority model, the preference changes dynamically while in static priority model preference assigned to static nature. This article focuses on static prioritization. The solution for the task allotment issue is given by various researchers in their articles by using different techniques. Some of them evolved well organized task scheduling algorithms using heuristic approaches and meta heuristic approaches. The detail study of various techniques that are used to solve the assignment issues have shown in the Table 1.

Table 1. Study of research articles generated by various authors.

S. No.	Researchers	Core technique	Highlights	Evaluation criteria	Nature of scheduling
1	Wu et al. [13]	Clustering approach	Market- situated hierarchical allocation policy in cloud workflow systems	Makespan, Cost, CPU time, cloud computing, workflow scheduling,	Static
2	Naderam et al. [14]	Heuristic approach	Heuristic approach on the basis of matching	Clustering, task assignment, Lagrangian relaxation,	Dynamic
3	Wang et al. [15]	Clustering approach	Developed a model to abate energy utilization of tasks execution	Green computing, cluster computing, schedule	Static
4	Tripathy et al. [16]	Directed search optimizatio n	Three novel algorithms, for scheduling, were presented	Assignment, Cost, Service, neural network, makespan	Dynamic
5	Xiao et al. [17]	Hybrid fog computing/ SDN VANET	Task allocation method to improve reliability within delay constraint	Reliability, node processing capacity, communication bandwidth	Static
6	Neamatollahi et al. [18]	HCSP clustering	To reduce the clustering energy value proposed HCSP algorithm	Network lifetime, cluster, energy, wireless sensor network, cost	Static
7	Kumar et al. [19]	*k*-means clustering method	Allocation method to optimize the cost and reliability	Reliability, execution cost, time, cluster	Static
8	Dao et al. [20]	Bat algorithm	BA is subjected to parallel processing, and a PBA communication approach is suggested to address JSSP	Makespan, flowtime, tardiness, weighted earliness	Dynamic
9	Kumar and Tyagi [21]	Clustering approach	*k*-means and fuzzy *c*-means algorithms were taken	Communication cost, execution cost, time	Static
10	Heidari et al. [22]	HHO	Developed the Harris Hawks Optimizer, a unique optimization method inspired by nature	Optimization, scalability	-
11	Alkhateeb and Abed-alguni [23]	Hybrid CS-SA	Developed a hybrid model combining CS and SA to enhance the results produced by CS.	Standard deviation, benchmark functions	Dynamic
12	Tirkolaee et al. [24]	Hybrid SAAFSA	Authors presented a high-performance decision-making for FSS issues in line with total cost and energy consumption reduction to enhance the productivity	Completion time, cost, energy consumption, sensitivity analysis	Static
13	Kanemitsu et al. [25]	Clustering approach	An algorithm was proposed to minimize the schedule length of heterogeneous processors	Efficiency, SLR, CCR	Static
14	Mishra and Majhi [26]	Bird swarm optimizatio n	Introduced a load balancing technique that distributes loads among virtual machines fairly	Load balancing, makespan, task scheduling	Dynamic
15	Alawad and Abed-alguni [27]	DJaya	To solve the PFSSP, suggested a novel approach called Discrete Jaya with Refraction Learning	Reliability, makespan, ARD	Dynamic
16	Haris and Zubair [28]	MMHHO	A new, effective hybrid optimization technique called MMHHO was designed to enhance load balancing in the cloud network	Cost, response time, load balancing	Dynamic

Table 2. Study of research publications authored by various researchers using PSO and GA.

S. No.	Researchers	Core technique	Highlights	Evaluation criteria	Nature of scheduling
1	Agarwal and Srivastava [29]	PSOGA	Makespan time was considered	Makespan, PIR	Static
2	Kang et al. [30]	Hybrid PSO	Approached an assignment algorithm on heterogeneous distributed environment	Task schedules, execution time, CER, DWR	Static
3	Samal et al. [31]	GA	GA based assignment result in multiprocessor environment	Fault-tolerance, scheduling, primary- backup	Static
4	Raju et al. [32]	PSO	Optimized tasks allocation strategy to reduce execution time	Deadline violation, resource allocation, waiting time, turnaround time	Static
5	Mutingi and Mbohwa [33]	GA	An approach to tackle the complications of supplier selection from a vaguely multi-criteria perspective	Price, lead time, quality	Static
6	Zhou et al. [34]	PSO	Modified PSO to control the local optimum	Inertia weight, cloud computing, CPU time, memory, allocation	Dynamic
7	Luan et al. [35]	Hybrid GA and ACO	Demonstrated a hybrid model to solve supplier selection issue	Supplier election, fitness value	Static
8	Tang et al. [36]	GA-ACO	The GAPACO algorithm was designed to save time and cost	Makespan, execution time, cost	Static
9	Mapetu et al. [37]	PSO	PSO based technique for load balancing	Makespan, time	Static
10	Kumar et al. [38]	Hybrid GA	Scheduling of tasks based on hybrid model	Scheduling, execution cost, time	Static
11	Kumar and Tyagi [39]	Hybrid GA, B&B	Hybrid model to reduce the system cost and time	Assignment, cost, communication time, reliability	Static
12	Devi and Garg [40]	Hybrid PSO	A hybrid model was developed to address the problem of redundant allocation	CPU time, reliability,	Static
13	Agarwal and Srivastava [41]	OPSO	To evade premature convergence, combined opposition-based learning technique with PSO	Makespan, scheduling, opposite number, PIR	Dynamic
14	Zhang et al. [42]	Hybrid GA	To resolve the QAP, the hybrid algorithm EGATS was devised.	CPU time, reliability,	Static
15	Chauhan et al. [43]	Hybrid GA	An approach to maximize the reliability of the system	Reliability, system time, execution cost	Static
16	Amirteimoori et al. [44]	PSO-GA	To simultaneously schedule tasks and transporters in a flow shop system, a MILP model was proposed.	Efficiency, time	Dynamic
17	Karishma and Kumar [45]	PSO	Authors developed a model to optimize flowtime, cost and time	Cost, Execution time, Flowtime	Static
18	Proposed technique	HPSOGAK	Algorithms based on PSO-GA are discussed to improve the system's reliability and reduce response time and cost.	Response time, cost, reliability, efficiency, resource utilization	Static

The meta heuristic techniques GA and PSO are based on the principles of biological evolution and swarms, respectively. These algorithms have been applied to address optimization problems across various domains, such as remote monitoring systems, energy-storage optimization, industrial engineering, and more. There have been numerous attempts to use these metaheuristic algorithms to solve a tasks allocation problem under various assumptions and limitations. Such as, in [29], the authors proposed a hybrid PSOGA model to enhance task scheduling in cloud computing, utilizing GA to refine solutions within PSO through genetic operations. However, a significant drawback of this approach was that GA tends to be slow and computationally intensive, due to the evaluation of many functions and the slow convergence of its operators. Unlike PSOGA, which relies on single-point crossover, the proposed method employs novel crossover and mutation techniques, to ensure a more diverse set of offspring. Additionally, a dynamic inertia weight balances local and global search components of PSO. This approach synergizes PSO and GA strengths, achieving faster convergence and superior optimization results. Table 2 includes an overview of some task scheduling approaches that comprise PSO, GA, and hybridization of these with other meta-heuristic or heuristic techniques, along with a list of all the salient characteristics.

Here, we are going to discuss some of the works done by the authors aimed at maximizing reliability without repetition. Reliability, a crucial factor to raise the system's performance, is used for examining fault-tolerant designs under significant unpredictability. If the program is carefully allocated to suitable processors, while considering the probability of failure for both communication lines and processors, a distributed system can achieve enhanced reliability in executing a program. Processor outages and communication errors have an impact on system reliability and user service quality. To raise the distributed system's reliability, Attiya and Hamam [46] approached a simulated annealing model in a heterogeneous environment and compared it to the B&B technique. Minghua et al. [47] came up with a model which gives the approximation for the upper and lower boundary of RAND 5 with a single run of the model. Authors Kang et al. [48] suggested HBMO algorithm, the power combination of simulated annealing and GA, to increase the distributed system's reliability. Other than these researchers Donight et al. [49] determined a novel method for fuzzy reliability that uses the beta type distribution as the membership function. In intelligent transportation systems, Noori and Jenab [50] established a model for rail vehicles with speed sensors on the basis of fuzzy reliability. Authors addressed a clustering based mathematical approach to get system's optimal reliability and cost by assigning tasks to the processors [19].

Although the use of a multiprocessor was expected to offer suitable solutions for the problem of task scheduling in DRTS, doing so caused several challenges. When a failure occurs in DRTS, we expect it to be noticed right away, and the distributed operation reverts to the previous checkpoint it had reached. Due to some such failures in system, uncertain results can be obtained. To rescue from this some authors had been dealing with the algorithms for the vague computation in which some of them got their best result and some got failed. Table 3 describes work of such authors with their drawbacks.

Table 3. The summary of existing literature regarding to their drawbacks.

S. No.	Literature	Existing environment	Drawback of existing environment
1	Cederholm and Petterson [51]	The goal of this study was to develop an algorithm for scheduling a collection of tasks in a multiprocessor environment	This research did not provide an outlook as per increasing the number of processors and the size of system
2	Jie et al. [52]	The authors focused on fixed and hard real-time scheduling in uniprocessor and multiprocessor systems	This work did not provide a distributed, soft real-time, dynamic, or heterogeneous assignment algorithm
3	Singh et al. [3]	Authors work presented a basic viewpoint on model PDS, CDS and RMA. This study had provided a tool that is created using the C programming language to accomplish job scheduling and energy consumption	This study did not include any suggestions for dealing with the genuine deadline or other assignment techniques
4	Li and Cheng [53]	This study had proposed a strategy for dealing with the few classes of real-time allotment problem and many optional ways in light of the RRP model, as well as achieving task scheduling transparency on the basis of conventional partitions	The transformation methods for the following aspects were not provided in this work. Non-preemptive policies, soft real-time policies, and dynamic job priorities
5	Narale and Butey [54]	Through the use of the Throttled load balancing technique, this study suggested a method to decrease data centre processing time and transfer cost	If the number of VMs rises, the response time and cost would as well
6	Adhikari et al. [55]	To achieve the best resource clustering, a load balancing technique that relies on the Bat-algorithm had been presented. Reduced degree of unbalance, better load balancing, and minimize makespan and cost	Homogeneous environment, lower scalability, unique task
7	Li and Wu [56]	To minimize task execution time and enhance system resource utilization, an ACO-based algorithm was developed. Maximize load balancing, maximal scalability	The presented technique was less reliable, having a longer response time and lower quality of service
8	Shao et al. [57]	The flow-shop scheduling problem (DBFSP) was examined in this article in order to reduce the maximum completion time	RPD and makespan performance could be improved
9	Hosseinioun et al. [58]	In this work, a method for conserving energy was introduced using the DVFS approach	The static energy was not taken into account in this model
10	Negi et al. [59]	The suggested PLB model takes into account makespan time, task starvation, multi-queuing model, resource mapping, and optimum performance	It did not take into account reallocating resources in a dynamic environment
11	Princess and Radhamani [60]	With an efficiency of 97%, hybrid Harris Hawk optimizing technique was developed	It did not offer a better degree of imbalance

3. Definitions and terminology

Appropriate task arranging onto relevant processors can enhance reliability of the system. Various performance specifications impact scheduling strategies. The $\widehat{e}_{i,k}$, where $1 \le i \le r,\ 1 \le k \le m$, is determined by the task's efficiency and the processor's attributes. If tasks t_i and t_j are on separate processors, $\widehat{c}_{i,j}$ symbolizes the fuzzy communication time between them. Under this report, it is assumed that data exist about $\widehat{e}_{i,k}$ and $\widehat{c}_{i,j}$ times, and these times are demonstrated as FE_TM and $FITC_TM$ in the form of matrix respectively and by the process of defuzzification FE_TM and $FITC_TM$ are renewed into crisp matrix.

3.1. Defuzzification

Robust's ranking approach has been used in this present work to defuzzify the FE_T and $FITC_T$. If $\left(a_\alpha^L, a_\alpha^U\right)$ represents the α -cut for fuzzy (triangular/trapezoidal) communication/execution time, then the defuzzification is performed as:

$$e_{i,k} = R\left(\widehat{e}_{i,k}\right) = \frac{1}{2}\int_0^1 \left(a_\alpha^L, a_\alpha^U\right) d\alpha$$

$$c_{i,j} = R\left(\widehat{c}_{i,j}\right) = \frac{1}{2}\int_0^1 \left(a_\alpha^L, a_\alpha^U\right) d\alpha . \tag{1}$$

After defuzzification, crisp values of communication time are stored in $ITC_TM = \left[c_{i,j}\right]$ matrix and values of execution time are stored in $E_TM = \left[e_{i,k}\right]$ matrix.

Moreover, the following are some key terms that will be used all across the article:

3.2. Communication time (C_T)

If the tasks are on multiple processors, then the total amount of time which needed to transmit data among tasks t_i and t_j is known as communication time. $TITC_T$ may be prescribed as:

$$TITC_T = \sum_{i,j=1}^{r} \sum_{\substack{k=1 \\ k \ne l}}^{m} \left(d_{k,l} \cdot \widehat{c}_{i,j}\right) \cdot x_{j,l} \cdot x_{i,k} , \tag{2}$$

where, the inter-processor distance $d_{k,l}$ represents the communication time per unit of data transferred between processor p_k and processor p_l . In scenarios where tasks t_i and t_j are assigned to different processors, the communication time is calculated as $\left(d_{k,l} \cdot \widehat{c}_{i,j}\right)$, where $d_{k,l} = 0$ if $k=l$ [39]. Therefore, under this approximation, the communication time for tasks on different processors corresponds directly to the amount of data exchanged between them.

3.3. Execution time (E_T)

The execution time is the time of executing so every task t_i on processor p_k. Total execution time (TE_T) is calculated as:

$$TE_T = \sum_{i=1}^{r} x_{i,k} \cdot \widehat{e}_{i,k} . \qquad (3)$$

3.4. System cost (CS)

In terms of time units, cost of the system is the total sum of execution and communication times. It can be determined as:

$$CS = \sum_{i,j=1}^{r} \sum_{\substack{k=1 \\ k\neq l}}^{m} \left(d_{k,l} \cdot \widehat{c}_{i,j}\right) \cdot x_{j,l} \cdot x_{i,k} + \sum_{i=1}^{r} \sum_{k=1}^{m} x_{i,k} \cdot \widehat{e}_{i,k} . \qquad (4)$$

3.5. Response time (RT)

The amount of computation to be performed by each processor determines the RT. Response time is inversely proportional to stability of the system. It can be calculated by using the following:

$$RT = \max_{1\leq k\leq m} \left\{ \sum_{i,j=1}^{r} \sum_{\substack{k=1 \\ k\neq l}}^{m} \left(d_{k,l} \cdot \widehat{c}_{i,j}\right) \cdot x_{j,l} \cdot x_{i,k} + \sum_{i=1}^{r} x_{i,k} \cdot \widehat{e}_{i,k} \right\} . \qquad (5)$$

3.6. System reliability (RS)

The probability that each engaged component will be operational during the execution process is referred to as RS. The stability of a system is improved by having a reliable system. The following equation is used to determine the RS:

$$RS = \left[\prod_{k=1}^{m} R_k\right] \cdot \left[\prod_{\substack{k=1 \\ k\neq l}}^{m} R_{k,l}\right], \qquad (6)$$

where R_k and $R_{k,l}$ are the processor p_k reliability and the *kl* link reliability, respectively. In a given time interval, let's call it t, processor p_k reliability pursues a Poisson distribution,

$$R_k = e^{-\int_0^t \lambda_k(t)dt}$$

where λ_k remains constant all across the procedure. In terms of E_T, in which task t_i is assigned to p_k processor, reliability of p_k can be computed as

$$R_k = e^{-\lambda_k \cdot \sum_{i=1}^{r} x_{i,k} \cdot \hat{e}_{i,k}} .$$

Correspondingly, in a specified interval of time t, the reliability of the *kl* path is $e^{-\mu_{k,l} t}$ and in terms of communication time the reliability between *kl* paths is given by

$$R_{k,l} = e^{-\mu_{k,l} \cdot \sum_{i,j=1}^{r} \sum_{\substack{k=1 \\ k \neq l}}^{m} \left(d_{k,l} \cdot \hat{c}_{i,j} \right) \cdot x_{j,l} \cdot x_{i,k}} .$$

3.7. Performance improvement ratio (PIR%)

Term PIR refers to the ability of a suggested algorithm, determined by the decrease in execution time. It is one of the essential parameters that ascertain the effectiveness of the proposed algorithm. It can be computed as per Eq (7), as follows:

$$PIR\ \% = \frac{RT_i - RT_{proposed}}{RT_i} \times 100 , \tag{7}$$

where $RT_{proposed}$ and RT_i represent the obtained RT for the proposed and i^{th} algorithm, respectively.

4. Model of tasks allocation problem

DRTS comprises a diverse set of heterogeneous processing units interconnected via advanced networks. These processing units and communication networks may possess different resource capabilities and varying levels of bandwidth utilization. This model can perform all functions simultaneously and communicate at any point during the program's execution. The following assumptions are employed to generate a model for solving the task allocation problem.

- In a real time structure, for every processor has a varying processing capability.
- On various processors, a task's execution time may vary.
- The allocated task persists on the processor while the program is executing.
- RS, RT, and CS are determined by the times of communication and execution.
- Heterogeneous processors are used in this DRTS paradigm. Consequently, the processors may be restricted by different memory and compute units and they have varying failure rates and processing times. Additionally, the communication links' failure rates may vary.
- While executing, a module consumes a certain amount of its designated processor's compute resources. It is possible for two modules that are running on different processors to communicate and incur a certain amount of intermodule communication time in terms of data quantity.

In this article, task allocation issue is preferred and timing-constrained tasks are placed on processors in a way that optimises RS, CS and RT under the following conditions:

- In DRTS all the tasks are non-preemptive.
- In a certain amount of time each and every processor execute single task.

The object of the task assignment problem in DRTS, where $E_T M$ and $ITC_T M$ are assumed to be given, is to minimize the RT and maximize the RS. The task allocation problem addressed with system resource constraints is as follows:

$$\min RT = \max_{1 \le k \le m} \left\{ \sum_{i,j=1}^{r} \sum_{\substack{k=1 \\ k \ne l}}^{m} \left(d_{k,l} \cdot \widehat{c}_{i,j} \right) \cdot x_{j,l} \cdot x_{i,k} + \sum_{i=1}^{r} x_{i,k} \cdot \widehat{e}_{i,k} \right\}.$$

Such that,

$$\sum_{l=1}^{m} x_{j,l} = 1 \ \forall j, \tag{8}$$

$$x_{j,l} \in \{0,1\} \ \forall j, l. \tag{9}$$

Each and every module must be allotted to precisely single processor, according to constraint (8) and $x_{j,l}$ must be binary variables, according to constraint (9). With a quadratic objective function, the aforementioned formulation is an NP-hard 0–1 programming problem.

5. The proposed method

This section elucidates the developed method for addressing the task scheduling problem, aiming to minimize CS and RT while maximizing RS. This paper introduces a novel scheduling method developed through the integration of a PSO-based swarm technique with GA and *k*-means techniques. In recent decades, GA and PSO have been widely utilized in a number of scientific domains. The study of computing systems motivated by collective intelligence is known as swarm intelligence. Large numbers of homogeneous agents in the environment work together to form collective intelligence. That includes the examples of flocks of birds, fish schooling, and ant colonies. Swarm intelligence can be used to find the best solutions for the issues like task scheduling. PSO is used in this paper because it is the best strategy for all size problems. An approximate solution to an optimization or search problem can be found using the GA search approach. Since GA considers all potential solutions, it takes longer to find a solution to any given problem. The procedure of selecting which task should be carried out at each moment in time is called task scheduling. The assignment algorithm prioritizes each active task at runtime and allots the highest priority task to the processor. In the present study task-clusters are formed using hybrid particle swarm optimization genetic algorithm *k*-means (HPSOGAK) and their assignment onto appropriate processor is done by using genetic algorithm. The main drawback of the traditional PSO algorithm is to establish a balance between local and global search, which leads to local best or optima in large number of optimization issues. Additionally, the standard PSO algorithm suffers from slow convergence rates, rendering it unsuitable for directly addressing assignment problems. Numerous attempts have been made to mitigate these issues, as detailed in Section 2. As an illustration, the approach in [30], devised a hybrid PSO-based method for task scheduling in heterogeneous systems, faced restrictions on operators' actions, potentially made parts of the search space unreachable. The motivation behind developing the proposed HPSOGAK approach is to overcome the drawbacks and limitations associated with the standard PSO algorithm. This work combines PSO and GA techniques to enhance the capabilities of conventional PSO, with the goal of effectively tackling the crucial task scheduling problem in heterogeneous computing systems. By combining the exploration capabilities of PSO with the refinement strengths of GA, the proposed hybrid method effectively balances the search process, leading to improved optimization results. Even though, using only genetic operators to determine the optimal solution is difficult. Therefore, the convergence rate of GA is improved by providing new population initialization, encoding and genetic operations. This article introduces new crossover and mutation techniques that help GA function better. In addition to initializing particles with HPSOGAK, a global-local best inertia weight is used to balance the local and global search components of the standard PSO algorithm. In the hybrid HPSOGAK algorithm, *k*-means clustering technique is implemented to produce a finite number of task-clusters as the conception of *k*-means is quite straightforward and comfortable to accomplish. The fundamental disadvantage of the *k*-means clustering method is that it may produce vacant clusters. Hence, PSO-GA approach is employed to address this shortcoming. After using the PSO-GA approach, non-empty clusters of tasks are obtained within a low number of iterations of *k*-means. Now, the fundamental characteristics of PSO and GA are as follows.

5.1. Particle swarm optimization

In an effort to address issues related to optimization, Kennedy and Eberhart [11] introduced the basic version of PSO. It is a meta-heuristic optimizing technique that draws inspiration from animal social behavior and information-sharing strategies, such as that of soaring birds and fish schooling. A particle swarm is a gathering of particles that can all move around in the problem space and be drawn to advantageous locations. Each and every particle of PSO in a population has two characteristics, velocity and position. On the basis of these two factors all the particles look for their food in that available search space. Influenced by the natural phenomena of schooling and flocking, PSO particles are distinguished not only by their position but also by a velocity that allows them to move within the search space. Each and every particle in PSO represents a location in the specified search space and a potential answer to the problem. Its goal is to optimize the problem-dependent fitness function, a function that provides each particle of the population a specific value demonstrating the superiority of the g best and p best in the solution. Centroids of task-clusters are represented as swarm particles in the present article, and the following Eq (10) can be used to get the fitness value for each particle.

$$g_{Cd} = \max\left(\frac{1}{1+\sum_{i=1}^{r}\left(\sum_{j=1}^{r}\left(\left\|c_{i,j}^{\hat{n}} - cd_{k,j}\right\|\right)^{2}\right)}\right), \qquad k = 1, 2, \ldots m \tag{10}$$

where, $\left(c_{i,j}^{\hat{n}}\right)$ represents the element of the matrix $NITC_T M$ and for the centroid of the k^{th} cluster Cd_k; $Cd_k = \left(cd_{k,1}, cd_{k,2}, \ldots, cd_{k,r}\right)$.

In the search space each particle's position is influenced by both its best position (*p* best) and the position of the next best particle (*g* best). All the particles migrate to the ideal solution, updating their p best and g best results. All of these particles have now reached their destination in the best possible manner. Each bird in this process is viewed as a distinct particle. Therefore, each and every particle has its own position and velocity. PSO is an iterative procedure in which each particle modifies its position and velocity in accordance with its prior experience as well as that of its neighbors. Determine the p best and g best for upgrading every particle's velocity and position i.e., cluster centroids based on its fitness values by the Eqs (11) and (12) respectively.

$$V_k(t+1)=\omega\times V_k(t)+\eta_1\times\phi_1\times\left(P_k(t)-Cd_k(t)\right)+\eta_2\times\phi_2\times\left(G_k(t)-Cd_k(t)\right) \tag{11}$$

where, ω stands for the inertia weight and the values of cognitive constant η_1 and social constant η_2 typically fall between [0, 2]. ϕ_1, ϕ_2 are two random numbers, have values in the range from [0,1]. The inertia component is the first, cognitive component is the second, and social component is the third component in the velocity upgrade calculation in Eq (11). Here, an algorithm's two successive iterations are represented by (t) and ($t+1$).

$$Cd_k(t+1)=Cd_k(t)+V_k(t+1). \tag{12}$$

The three components that make up the velocity vector, which controls how a particle moves through a given search space, are as follows: *momentum*, also known as *inertia*, keeps a particle from abruptly changing direction; *cognitive component*, that is accountable for the trend to return the particles to their previous foremost position; and *social component*, which assists a particle in moving through the swarm's best position. These factors influence how the velocity of the k^{th} particle updates according to Eq (11). The iterative procedure described in Eqs (11) and (12) will continue until the halting condition is satisfied.

The graphic depiction of the PSO is shown in Figure 1. The particle changes direction with each iteration, and often, the new path is optimal. This decision is based on both the individual's personal best position and the global best position.

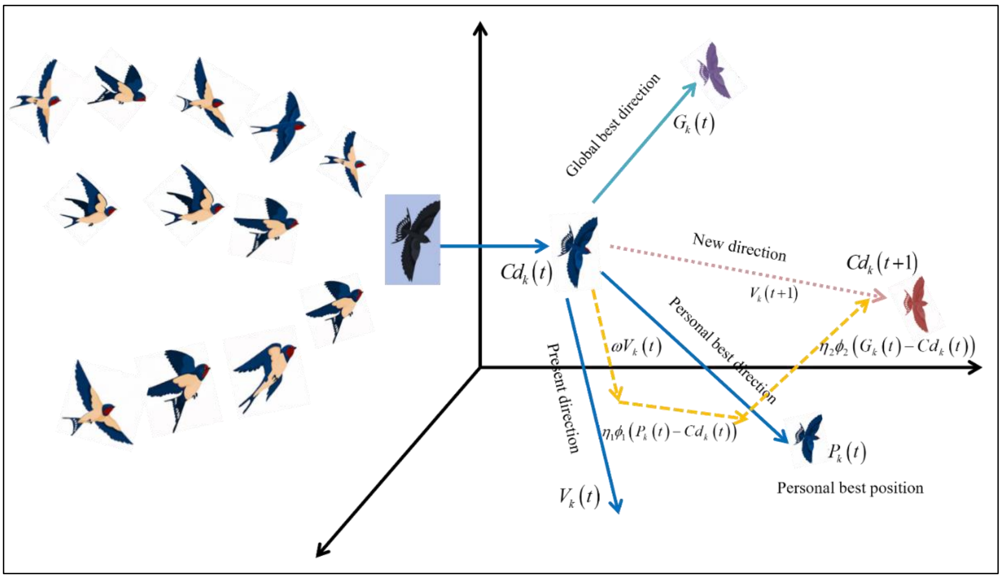


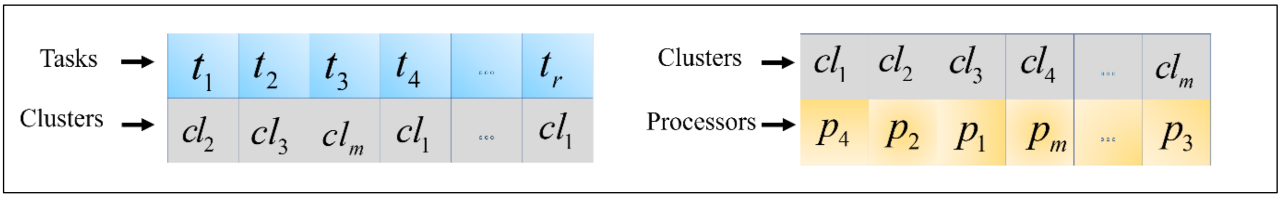
Figure 1. Graphical presentation of PSO.

5.2. *Genetic algorithm*

Like PSO, GA is also a sort of meta-heuristic optimization approach. Inspired from Charles Darwin's theory of "Survival of the fittest", the GA is capable of producing excellent solutions for a variety of issues, including optimization and search, by imitating the processes of natural selection, breeding, and mutation. As robust stochastic search in algorithms, GA has lately been used to solve the job-shop assignment problem and task allocation issue. Based on a concept of natural genetics and selection, this category of techniques combines the concept of the fittest surviving, random yet structured search, and parallel assessment of locations in the search space. A population of specific solutions is continuously modified by the GA. The population develops towards the best option through subsequent generations. GA can be employed to address a variety of optimization problems, such as those involving stochastic, discontinuous, highly nonlinear, or non-differentiable objective functions. The following subsections of GA cover the novel encoding, population initiation approach and genetic processes.

5.2.1. Evolution of an initial population

Presenting chromosome like a string of integers is particularly beneficial for task clustering and allocating task-clusters onto processors. Each sole chromosome demonstrates the allocation of one of the possible modes of each task into a cluster or to a task-cluster on a processor. The number of tasks 'r' (or processors 'm') involved in a program, determines each chromosome's length used to cluster of tasks (or schedule task-clusters onto processors). Any of the program's total participating processors could be a chromosome's digit and each and every gene connected with a chromosome provides information on scheduling and clustering. Here, an example of an encoded chromosome is presented, where the 'r' tasks are randomly clustered to form 'm' clusters, and their allocation is assigned across 'm' processors. Chromosomes are encoded as follows in Figure 2 for task clustering and scheduling:



Tasks →	t_1	t_2	t_3	t_4	…	t_r
Clusters →	cl_2	cl_3	cl_m	cl_1	…	cl_1

Clusters →	cl_1	cl_2	cl_3	cl_4	…	cl_m
Processors →	p_4	p_2	p_1	p_m	…	p_3

Figure 2. Chromosomes encoding. (a) Encoding of clustering of tasks. (b) Encoding of task-clusters scheduling.

5.2.2. Fitness function

The optimization-related objective function is the fitness function in GA. The accuracy of tasks allocation on processors is expressed by the fitness function, which assigns a value to every chromosome in the population. Optimizing RT, RS and CS are the main goals of the task allocation issue. By clustering the tasks that are highly communicated via HPSOGAK technique and then allocating them to appropriate processors via GA technique, the assignment problem has been solved.

For any random number $\varepsilon \in [0,1]$, the following is the fitness function for the allocation algorithm GA:

$$f_{fet} = 1/\left\{\varepsilon + \sum_{i=1}^{r}\sum_{k=1}^{m} x_{i,k}\,.\,\widehat{e}_{i,k}\right\}. \tag{13}$$

The purpose of this fitness function is to effectively evaluate the quality of solutions, guide the optimization process towards optimal results, and helps to improve the convergence speed.

5.2.3. Selection

Pick the best, dismiss the rest, is the guiding idea of the selection operator. The process of selection determines which chromosomes should be maintained and allowed to regenerate and which ones should be eliminated. A selection operator's primary goal is to maintain the population dimension while decreasing the proportion of low-quality chromosomes and increasing the proportion of high-quality chromosomes in a population. The roulette-wheel selection approach is used in this research article, where each and every time a single chromosome is chosen for a new population. This procedure takes place on a spinning wheel, and the spinning numbers are commensurate with the size of the population. The following is the given selection probability for task scheduling onto processor:

$$P_{rob}^{i} = 1 - \left\{\frac{f_{fet}\left(chromosome\_i\right)}{\sum_{i=1}^{m}\left(f_{fet}\left(chromosome\_i\right)\right) - f_{fet}\left(chromosome\_i\right) + 1}\right\}. \tag{14}$$

From the foremost population, eliminate the chromosomes with the lowest selection probability. Furthermore, include a chromosome by replicating one chromosome with higher selection probability to new population.

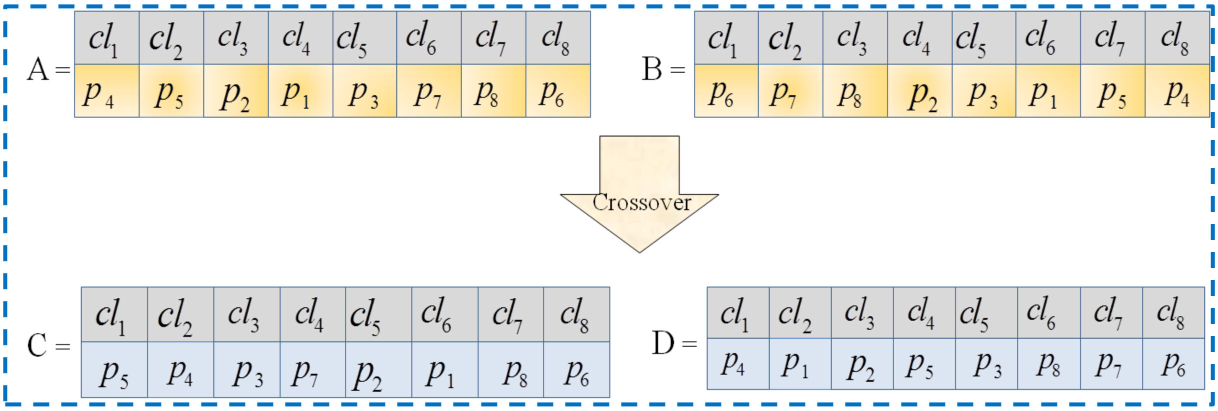
5.2.4. Crossover

The most essential part of a genetic algorithm is crossover. By combining information from the two chromosomes of parents, it produces the two chromosomes of the offspring. In GA, the crossover operator is employed on the chromosomes with the lowest probability rather than the chromosomes with the highest probability in the population. The crossover probability, which ranges between 0 and 1, determines how frequently crossover occurs between chromosomes in each generation, meaning the likelihood of two chromosomes exchanging parts. A 100% crossover rate implies that all offspring are produced through crossover, while a 0% rate means no crossover occurs, resulting in a new generation nearly identical to the old one. The present article proposes a novel crossover strategy with a probability of 0.8, based on several experimental studies such as [38,39]. Tests show that a 0.8 crossover probability offers a balance between rapid convergence and high-quality results. Lower probabilities slow convergence, whereas higher one risk losing genetic diversity or causing premature convergence. This impacts GA and PSO convergence rates and aids in achieving optimal clustering outcomes. The presented crossover operator selects two parents with equal probability for the next generation. The goal of this crossover operator is to produce offspring who inherit groups with a significant level of variety from two chosen parents. In Algorithm 1, the suggested crossover operator is described as follows:

Algorithm 1: Proposed crossover operator.

```
Input:   Select two distinct parents $A^{(t)}$ and $B^{(t)}$ of dimension m from parent pool
Output:  Two distinct offspring $C^{(t+1)}$ and $D^{(t+1)}$
         # Create offspring $C^{(t+1)}$ and $D^{(t+1)}$ respectively as follows:
         # For offspring $C^{(t+1)}$, $C^{(t+1)}$= Reverse $\left(C^{'(t+1)}\right)$
         While $\left(k \le m\right)$
         {
          if ($k$==1)
          {
             $C_1^{'(t+1)} == B_1^t$
          }
          if $\left(1<k\le m\right)$
          {
             $x_k^t$= (image ($C_{k-1}^{'(t+1)}$)) IN $B^{(t)}$
             $y_k^t$= (image ($x_k^t$)) IN $A^{(t)}$
             if (($y_k^t$ IN $C^{'(t+1)}$)=True)
             {
                $C_k^{'(t+1)}$=min$\left(B^{(t)} - C^{'(t+1)}\right)$
             }
             else
             {
                $C_k^{'(t+1)}$= $y_k^t$
             }
          }
         }
         $C^{(t+1)}$=Reverse $\left(C^{'(t+1)}\right)$
         # For offspring $D^{(t+1)}$
         While $\left(k \le m\right)$ do
         {
             if (k=1)
             {
                $D_1^{(t+1)} == A_1^t$
             }
             if$\left(1 < k \le m\right)$
             {
                  $w_k^t$= (image ($D_{k-1}^{(t+1)}$)) IN $A^{(t)}$
                  $v_k^t$= (image ($w_k^t$)) IN $B^{(t)}$
                  if (($v_k^t$ IN $D^{(t+1)}$)=True)
                {
                    $D_k^{(t+1)} = \min\left(A^{(t)} - D^{(t+1)}\right)$
                }
                else
                {
                    $D_k^{(t+1)} = v_k^t$
                }
            }
        }
```

In this Algorithm 1, (t) represents the generation count and the image of any bit in a chromosome is the bit after it. An example of the developed crossover technique is shown below. Through use of the procedures in Algorithm 1, the two offsprings C and D that were produced from the parents' chromosomes A and B are as follows in Figure 3:



A =

cl_1	cl_2	cl_3	cl_4	cl_5	cl_6	cl_7	cl_8
p_4	p_5	p_2	p_1	p_3	p_7	p_8	p_6

B =

cl_1	cl_2	cl_3	cl_4	cl_5	cl_6	cl_7	cl_8
p_6	p_7	p_8	p_2	p_3	p_1	p_5	p_4

C =

cl_1	cl_2	cl_3	cl_4	cl_5	cl_6	cl_7	cl_8
p_5	p_4	p_3	p_7	p_2	p_1	p_8	p_6

D =

cl_1	cl_2	cl_3	cl_4	cl_5	cl_6	cl_7	cl_8
p_4	p_1	p_2	p_5	p_3	p_8	p_7	p_6

Figure 3. An example of the proposed crossover technique.

5.2.5. Mutation

Following crossover, the chromosomes undergo a mutation operation. GA uses mutations in chromosome populations as a genetic operator to preserve genetic diversity from one generation to the succeeding. A mutation operator's major purpose is to prevent chromosomes from becoming too identical to each other as well after a certain number of iterations. A novel mutation technique with a probability of 0.1 is used in this article. This technique can be comprehended with the help of the subsequent Algorithm 2.

Algorithm 2: Proposed mutation operator.

```
Input:   Select an offspring chromosome $C^{(t+1)}$ of dimension $m$
Output:  A mutated offspring $E^{(t+1)}$
         # Create mutated offspring $E^{(t+1)}$ as follows:
         Select two random cut-points over $C^{(t+1)}$ say, cut-point1 and cut-point2 respectively
```

```
2:   $x^{'}=C^{(t+1)}$ (cut-point1 to cut-point2)
3:   $x^{'}$=Shuffle $\left(x^{'}\right)$
4:   Place $x^{'}$ in $C^{(t+1)}$ between cut-point1 and cut-point2
5:   While $(k \le m)$
6:   {
7:   if (k==1)
8:   {
9:     $E_1^{'(t+1)} =$ min$\left(x^{'}\right)$
10:  }
11:  if$\left(1 < k \le m\right)$
12:  {
13:    $y^{'(t+1)}$=(image ($E_{k-1}^{'(t+1)}$)) IN $C^{(t+1)}$
14:    if (( $y^{'(t+1)}$ IN $E^{'(t+1)}$ )=True)
15:    {
16:      $E_k^{'(t+1)}$=image ( $y^{'(t+1)}$ )
17:    else
18:    {
19:      $E_k^{'(t+1)} = y^{'(t+1)}$
20:  }
21:  }
22:  Reshuffle the bits of $E_k^{'(t+1)}$ to their original position from Step 2 to get mutated
     offspring $E^{(t+1)}$
```

In this Algorithm 2, (t+1) represents the generation count and the image of any bit in a chromosome is the bit after it. Below is an example that demonstrates this operator's process. Through use of the procedures in Algorithm 2, the mutated offspring E that was produced is as follows in Figure 4:

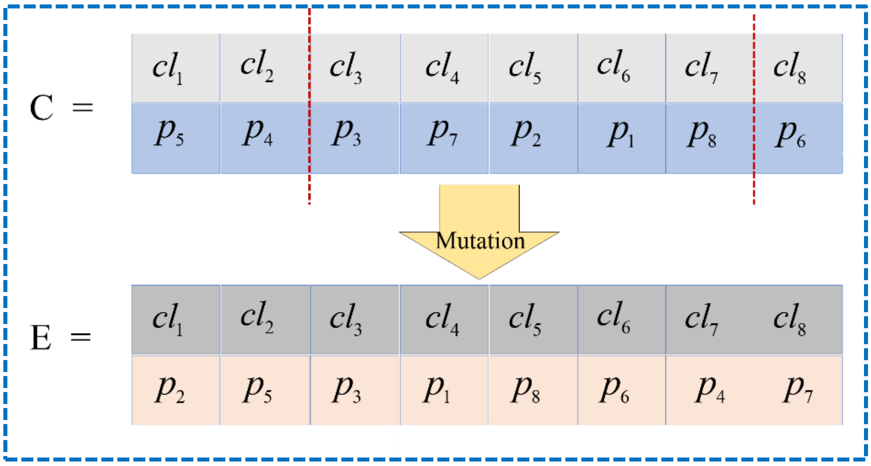

C =
cl_1 cl_2 cl_3 cl_4 cl_5 cl_6 cl_7 cl_8
p_5 p_4 p_3 p_7 p_2 p_1 p_8 p_6
Mutation
E =
cl_1 cl_2 cl_3 cl_4 cl_5 cl_6 cl_7 cl_8
p_2 p_5 p_3 p_1 p_8 p_6 p_4 p_7

Figure 4. An example of the proposed mutation technique.

5.3. *Stopping criteria*

The stopping condition is used to prevent PSO and GA from running indefinitely. Both the algorithms run until they converge in order to obtain a better quality of solution, and the result is the best solution as far obtained. A terminating criterion is needed to determine this convergence behavior. Convergence happens when the most ideal reported value does not change during the maximum number of generations. In this work, two distinct categories of stopping criteria are used.

- An upper bound on the maximum number of iterations (generations).
- The algorithm keeps running for a specified number of generations until the best result obtained throughout the evolution process does not improve.

5.4. *Developed algorithms*

To address the task allocation issue, the suggested model uses a PSO-based hybrid model HPSOGAK to generate task-clusters and GA to find the task-clusters' assignment onto processors. The proposed technique is appropriate for both fuzzy and crisp time. The proposed task scheduling method is divided into two phases. The HPSOGAK and GA techniques are discussed in the first and second phases, respectively.

5.4.1. Phase I

The HPSOGAK algorithm is developed in this phase by integrating PSO, GA and *k*-means. In the suggested approach, centroids are taken as particles that are improved using the PSO-GA technique and then used like initial centroids in the *k*-means algorithm. The method elevates the 'p best and g best' position of particles determined from PSO using genetic operators and then *k*-means clustering approach is employed to acquire a finite number of task-clusters. To create the task-clusters using *k*-means clustering approach $ITC_TM = \left[c_{i,j}\right]$ is renewed from $FITC_TM = \left[\widehat{c}_{i,j}\right]$. To provide the best outcomes for all data points, the *k*-means algorithm repeatedly reduces the distances between each data point and its centroid. To make a one-to-one correlation, the number of processors 'm' and task-clusters 'k' must be the same. Here n_k and Cd_k indicates the tasks' number in k^{th} cluster and the centroid of the k^{th} cluster respectively, where

$$Cd_k = \left(cd_{k,1}, cd_{k,2}, \ldots, cd_{k,r}\right).$$

By implementing the HPSOGAK, ITC_T is minimized by clustering together the extremely communicative tasks.

To address various allocation issues based on PSO, several authors have proposed their mechanisms, wherein the inertia weight remains either constant or decreases linearly during the process. Based on existing knowledge, the assignment of tasks in DRTS has been categorized as NP-hard problems, and the complexity of these assignments further increases with the increase in the number of deployed processors and the tasks that have been submitted. So, balancing the local search and the global search is important, and for that, in the present work, the inertia weight, ω is determined based on the calculation in Eq (15).

$$\omega = \omega_{\max} + \left(\frac{\omega_{\min}\text{ - current iter.}}{\text{max iter.}} \right) \times 0.5 \tag{15}$$

where,

$$\omega_{\max} = \text{ Maximum weight },$$

$$\omega_{\min} = \text{ Minimum weight },$$

$$\text{Max iter.} = \text{Maximum iteration number.}$$

The inertia weight is set to decrease linearly from a high initial value $(\omega_{\max} = 0.9)$ to a lower final value $(\omega_{\min} = 0.4)$ as the algorithm progresses through its iterations. This approach enhances global exploration at the start of the iterative process and promotes a more localized, fine-tuned search towards the end of the iterations.

Phase I is described in detail as in Algorithm 3.

Algorithm 3: Formation of task-clusters by employing HPSOGAK.

```
Initialize $r, m, \omega, \eta_1, \eta_2, \phi_1, \phi_2, FITC_T M$
Defuzzification of $FITC_T M = \left[\hat{c}_{i,j}\right]$ into crisp $ITC_T M = \left[c_{i,j}\right]$ through using Robust's ranking approach
  for $0 \le i \le j \le r$ do
    {
      $NITC_T M = \left[c^{\hat{n}}_{i,j}\right] = \begin{cases} \max\limits_{i} \max\limits_{j} c_{i,j} - c_{i,j} \text{ , if } i \ne j \\ 0 \text{ , otherwise} \end{cases}$
    }
  end for
Generate 'm' number of task-clusters of 'r' tasks randomly named as $\{cl_1, cl_2, ...cl_m\}$
  for $k^{th}$ cluster $(1 \le k \le m)$ do
   {
     Calculate the initial $Cd_k$ for PSO-GA as follows:
     $Cd_k = \frac{1}{n_k} \sum_{i,j=1}^{r} c^{\hat{n}}_{i,j} . u_{i,k}$
    }
  end for
t=0
for $(1 \le k \le m)$ do
    {
    Initialize $V_k(t)$ and $Cd_k(t)$
    $P_k(t) = Cd_k(t)$
    }
```

```
20:         end for
21:         Formation of new centroids using PSO-GA technique
22:           if (PSO-GA stopping criteria = false) do
23:           for $k^{th}$ particle $(1 \le k \le m)$ do
24:             {
25:               Evaluate the fitness value $g_{Cd}$ of each particle by using Eq (10)
26:               Determine the p best for each particle based on its fitness value
27:               Implement the crossover operator to update p best
28:               Implement the mutation operator to update p best
29:               Determine the g best based on its fitness value
30:               Implement the mutation operator to update g best
31:               Update $V_k\left(t+1\right)$ by using Eq (11)
32:               Update $Cd_k\left(t+1\right)$ by using Eq (12)
33:             }
34:           end for
35:         end if
36:         Assign the centroids obtained in Step 32 as the initial centroids for the k-means algorithm
37:         Repeat
     (a)      for $(1 \le i \le r \ and \ 1 \le k \le m)$ do
               {
                 Distance $\hat{d}_{ik}$ between task $t_i$ and the cluster centre $Cd_k$
                 $\hat{d}_{ik} = \sqrt{\sum_{j=1}^{r}\left(c_{i,j}^{\hat{n}} - cd_{k,j}\right)^2}$
               }
              end for
     (b)    Create new task-clusters by allocating each task to its closest cluster centroid
            # Update novel clusters as follows
     (c)      for $\left(1 \le k \le m\right)$ do
               {
                 $Cd_k = \frac{1}{n_k}\sum_{i,j=1}^{r} c_{i,j}^{\hat{n}} . \ u_{i,k}$
               }
              end for
     (d)    Apply 37 (a), (b) steps and encore the process until the stopping criteria are met
38:         end
```

5.4.2. Phase II

Throughout this phase, GA has been implemented to schedule task-clusters into processors. During the program's execution, the clusters obtained from the preceding Algorithm 3 will remain unchanged. Identical clustered tasks perform similarly to a single task and might be assigned to the same processor. The i^{th} and j^{th} rows of the FE_TM must be added and represented as a new row in the NFE_TM if tasks t_i and t_j do belong to the same cluster. Only one processor may be assigned to all of the tasks that are present in one cluster at once. According to the task-clusters that are acquired from algorithm HPSOGAK, $E_TM = \left[e_{i,k}\right]$ is updated into $NE_TM = \left[e^{\hat{n}}_{i,k}\right]$. Phase II is described in detail as an algorithm. Algorithm 4 provides a summary of this stage's numerous steps:

Algorithm 4: To determine the assignment of task-clusters onto processors.

```
1:  Input $FE_TM = \left[\hat{e}_{i,k}\right]$
2:  Defuzzification of $FE_TM = \left[\hat{e}_{i,k}\right]$ into crisp $E_TM = \left[e_{i,k}\right]$ through using Robust's ranking approach
3:  Modify $E_TM = \left[e_{i,k}\right]$ into $NE_TM = \left[e^{\hat{n}}_{i,k}\right]$ by fusing the rows based on the cluster acquired from Algorithm 3
4:  To produce the initial population, encode '$m$' chromosomes
5:  Start
6:    Size of $population=m$
7:    for $k$=1 to size of population '$m$'
8:    chromosome $ch_k$=select in random order
9:  end
10: Through using the following formula, compute the fitness value of each chromosome in the population:
      $$f_{fet} = 1/\left\{\varepsilon + \sum_{i=1}^{r}\sum_{k=1}^{m} x_{i,k} \,.\, \hat{e}_{i,k}\right\}$$
11: Assess the probability of each chromosome in the population through using following formula:
      $$P^{i}_{rob} = 1 - \left\{\frac{f_{fet}\left(chromosome\_i\right)}{\sum_{i=1}^{m}\left(f_{fet}\left(chromosome\_i\right)\right) - f_{fet}\left(chromosome\_i\right) + 1}\right\}$$
12: New population to be produced-
13: For the new population, select the chromosome having high probability
14: Select the chromosomes with the least probability and apply the proposed crossover operator to them
15: Implement the proposed mutation operator
16: If the prerequisites for stopping are met, stop; otherwise go to Step 10
17: Applying Eqs (4) – (6) to determine CS, RT, and RS respectively.
18: End
```

In Figure 5, the whole processes of Algorithms 3 and 4 are shown, providing a succinct understanding of the algorithms' processes.

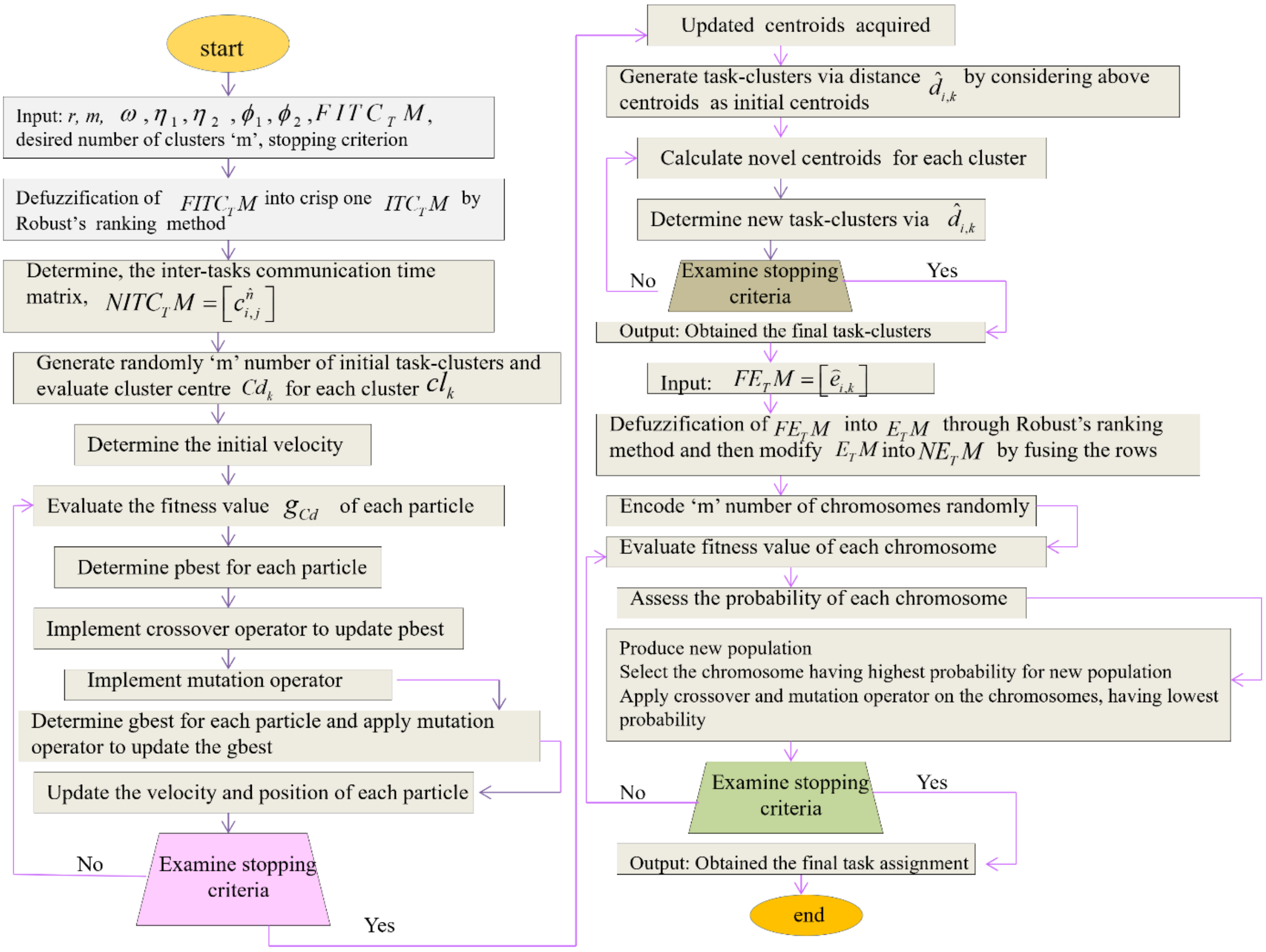

start
Input: r, m, $\omega, \eta_1, \eta_2, \phi_1, \phi_2, FITC_T M$, desired number of clusters 'm', stopping criterion
Defuzzification of $FITC_T M$ into crisp one $ITC_T M$ by Robust's ranking method
Determine, the inter-tasks communication time matrix, $NITC_T M = \left[c_{i,j}^{\hat{n}}\right]$
Generate randomly 'm' number of initial task-clusters and evaluate cluster centre Cd_k for each cluster cl_k
Determine the initial velocity
Evaluate the fitness value g_{Cd} of each particle
Determine pbest for each particle
Implement crossover operator to update pbest
Implement mutation operator
Determine gbest for each particle and apply mutation operator to update the gbest
Update the velocity and position of each particle
No
Examine stopping criteria
Yes
Updated centroids acquired
Generate task-clusters via distance $\hat{d}_{i,k}$ by considering above centroids as initial centroids
Calculate novel centroids for each cluster
Determine new task-clusters via $\hat{d}_{i,k}$
No
Examine stopping criteria
Yes
Output: Obtained the final task-clusters
Input: $FE_T M = \left[\hat{e}_{i,k}\right]$
Defuzzification of $FE_T M$ into $E_T M$ through Robust's ranking method and then modify $E_T M$ into $NE_T M$ by fusing the rows
Encode 'm' number of chromosomes randomly
Evaluate fitness value of each chromosome
Assess the probability of each chromosome
Produce new population
Select the chromosome having highest probability for new population
Apply crossover and mutation operator on the chromosomes, having lowest probability
No
Examine stopping criteria
Yes
Output: Obtained the final task assignment
end

Figure 5. Flowchart depicting the procedure of Algorithms 3 and 4.

5.5. *Performance parameter*

If the setting of PSO and GA algorithm parameter is improper, it will slow down the solution speed and have an impact on the outcomes' quality. Each population-based technique may converge to the global best location in a realistic period of time by maintaining a proper balance between local search and global search. Inertia weight (ω) and crossover-mutation, respectively, are the coefficients that keep this balance in the PSO and GA algorithms. Therefore, the developed technique yields new crossover and mutation strategies for GA and an updated ω-equation (Eq (15)) for PSO. Where the value of ω steadily decreased throughout the iteration and balanced the rate of PSO convergence. This section provides details on a few significant parameters that the proposed task assignment algorithm considers for analytical evaluation. The articles by Kumar and Tyagi [39], Agarwal and Srivastava [41], and Shatz et al. [61] are taken into consideration for determining the parameters. Following Table 4 presents the parameters:

Table 4. Input parameters and the corresponding values.

System parameters	Values
Inertia weight ω	0.4–0.9
ϕ_1 *and* ϕ_2	[0,1]
η_1 *and* η_2	1.49
Failure rate of processor, λ_k	[0.0002–0.00012]
Failure rate of *kl* communicating path, $\mu_{k,l}$	[0.0005–0.00015]
Maximum iteration of GA	50
Crossover probability	0.8
Mutation probability	0.1
Maximum iteration of PSO	100

6. Performance assessment

Let's assess the technique which is presented by considering the four real-applications-based examples. All four examples are drawn from proven existing models. In two examples, time is regarded in the crisp form, however in the other two examples; time is viewed as a fuzzy number. Fuzzy numbers can be of any form, including Gaussian, bell-shaped, triangular, trapezoidal, and so on. Table 5 illustrates the test setup, employing the presented HPSOGAK and GA algorithms for solving task scheduling problems. It represents the solutions of RS, RT, and CS of the system, along with the allocation onto processors for all four analyzed examples. Additionally, the effectiveness of the presented PSO-based task scheduling technique has been assessed in terms of resource utilization (Abdelkader and Omara [62]). The resource utilization is carried out in terms of RT and is contrasted with other models. According to Table 5, it is vivid that in order to reduce the RT and CS and maximize the RS and utilization, the proposed model produces a better quality of results than any existing models for allocation.

Table 5. Comparison of the aforementioned examples' outcomes with those of existing methods.

Example	Reference	Used Technique	Tasks Assignment	RS	CS	RT	Resource Utilization (%)
1	Djigal et al. [63]	List based scheduling algorithm	$\{t_1,t_2,t_3,t_7\}\rightarrow p_1$ $\{t_4,t_6,t_8,t_{10}\}\rightarrow p_2$ $\{t_5,t_9\}\rightarrow p_3$	-	304	181	75.32
	Proposed model	Hybrid PSO	$\{t_8,t_{10}\}\rightarrow p_1$ $\{t_2,t_5,t_9\}\rightarrow p_3$ $\{t_1,t_3,t_4,t_6,t_7\}\rightarrow p_2$	0.98074	275	162	79.39
2	Kumar et al. [64]	Heuristic approach	$\{t_1,t_4\}\rightarrow p_1$ $\{t_5\}\rightarrow p_2$ $\{t_2,t_3\}\rightarrow p_3$	0.97096	(145, 240, 360)	(110,175,250)	68.06

Continued on next page

Example	Reference	Used Technique	Tasks Assignment	RS	CS	RT	Resource Utilization (%)
2	Proposed model	Hybrid PSO	$\{t_1\}\to p_2$ $\{t_2,t_3,t_5\}\to p_1$ $\{t_4\}\to p_3$	0.98780	(145,230,320)	(95,165,240)	74.35
3	Sharma et al. [65]	Clustering approach	$\{t_4,t_5\}\to p_1$ $\{t_2,t_3\}\to p_2$ $\{t_1\}\to p_3$	0.9956	123	97	77.89
	Proposed model	Hybrid PSO	$\{t_3,t_4,t_5\}\to p_2$ $\{t_1\}\to p_1$ $\{t_2\}\to p_3$	0.9979	81	68	81.34
4	Chauhan et al. [43]	Hybrid GA	$\{t_1,t_8\}\to p_1$ $\{t_4,t_9\}\to p_2$ $\{t_2,t_5\}\to p_3$ $\{t_6,t_7\}\to p_4$ $\{t_3\}\to p_5$	(0.9349,0.9506,0.9693)	(403.8,634.8,965.8)	(32,53,78)	51.29
	Proposed model	Hybrid PSO	$\{t_3\}\to p_5$ $\{t_4,t_9\}\to p_2$ $\{t_1,t_8\}\to p_1$ $\{t_2,t_5\}\to p_3$ $\{t_6,t_7\}\to p_4$	(0.9349,0.9506,0.9693)	(403.8,634.8,965.8)	(32,53,78)	51.29

Example 1. Here, the problem is grabbed from Djigal et al. [63] model. In which DRTS consist ten tasks $\{t_1, t_2, \ldots\ldots, t_{10}\}$ that have to be allocated on three processors $\{p_1, p_2, p_3\}$. In this problem $e_{i,k}$ and $c_{i,j}$ have been assessed as crisp number shown in Figures 6 and 7 respectively.

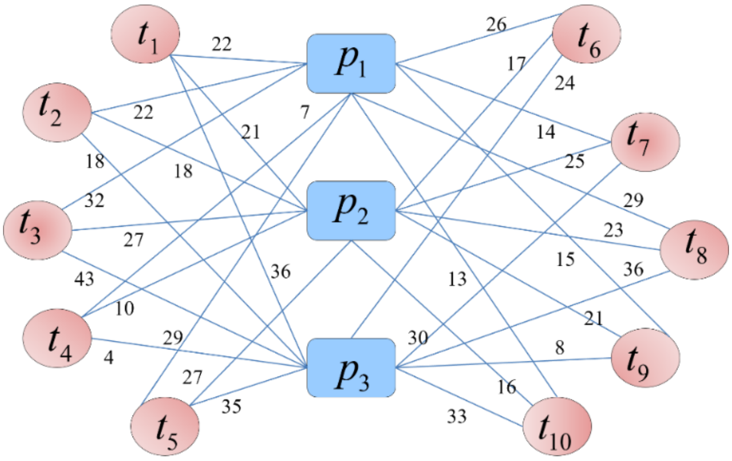


Figure 6. Tasks crisp execution time ($e_{i,k}$) on processors.

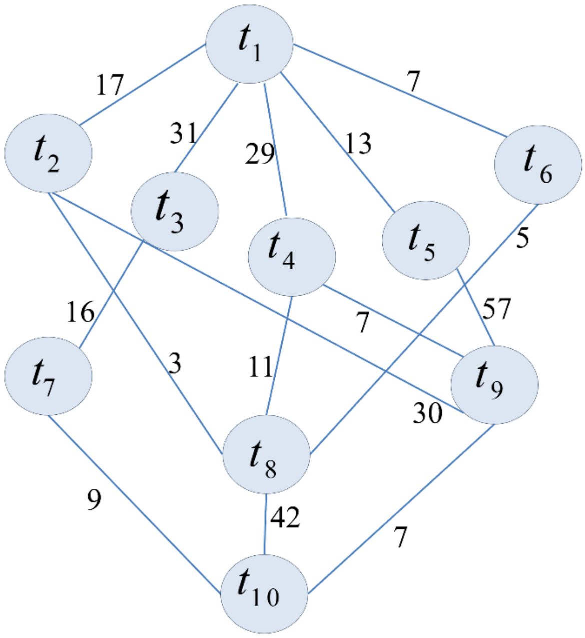


Figure 7. Crisp inter-task communication time ($c_{i,j}$) between tasks.

The following chromosome in Figure 8 gives the final task-clusters as a result of carrying out the steps of Algorithm 3:

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
cl_3	cl_2	cl_3	cl_3	cl_2	cl_3	cl_3	cl_1	cl_2	cl_1

Figure 8. Final task-clusters of Example 1.

The optimal allocation of task-clusters onto worthy processors has been achieved employing the steps in Algorithm 4, that is displayed by the following chromosome in Figure 9:

cl_1	cl_2	cl_3
p_1	p_3	p_2

Figure 9. Allocation of task-clusters onto processors of Example 1.

The task-clusters and their allotment on processors can be written as follows using the two chromosomes mentioned above:

$$cl_1 = \{t_8, t_{10}\} \rightarrow p_1,\ \ cl_2 = \{t_2, t_5, t_9\} \rightarrow p_3 \ \text{ and } \ cl_3 = \{t_1, t_3, t_4, t_6, t_7\} \rightarrow p_2 .$$

Task-clusters assignments are shown in Figure 10.

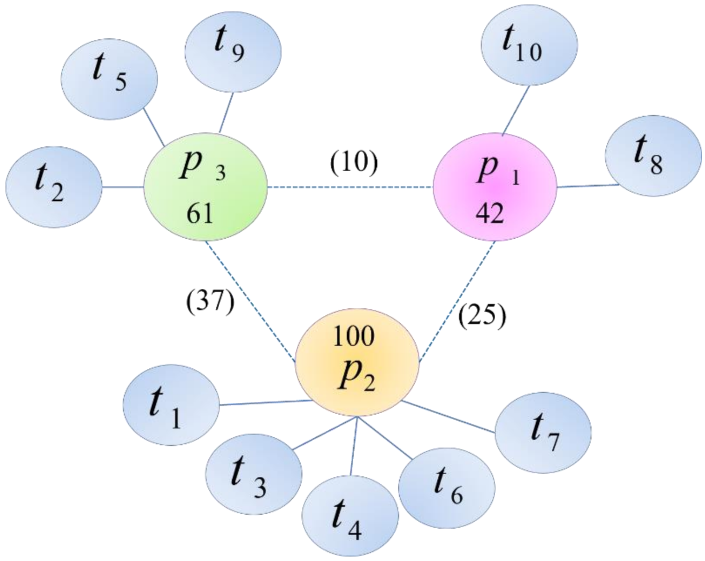


Figure 10. Solution graph of Example 1.

In Figure 10, the processors are enclosed by solid circles and depict the corresponding $e_{i,k}$ on them, while the figures with dotted lines in parenthesis depict $c_{i,j}$. Now, based on the information gleaned from the optimal allocation; the RT is 162 and the CS is 275 of the system. Equation (6) determines the system's reliability (RS), that is 0.98074.

Example 2. Here, the problem is acquired from Kumar et al. [64] model. In which DRTS consist five tasks $\{t_1, t_2, \ldots, t_5\}$ that have to be allocated on three processors $\{p_1, p_2, p_3\}$. In this problem $\widehat{e}_{i,k}$ and $\widehat{c}_{i,j}$ have been assessed as fuzzy triangular number shown in Figures 11 and 12 respectively.

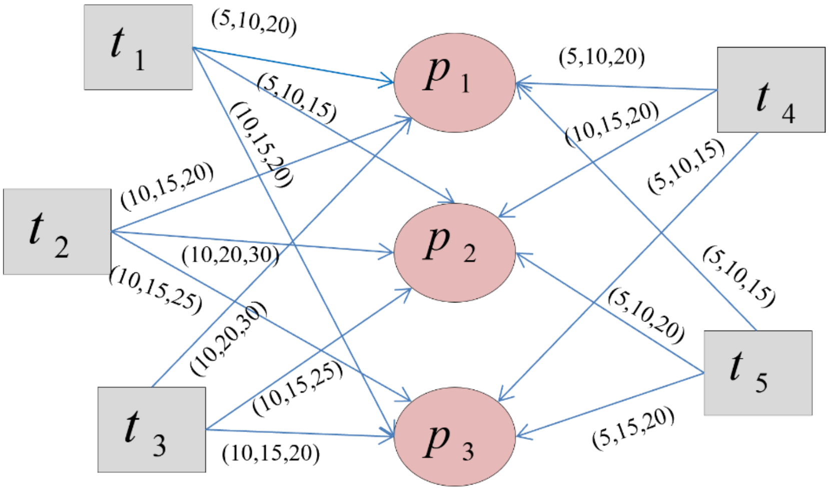


Figure 11. Fuzzy execution time ($\widehat{e}_{i,k}$) on processors.

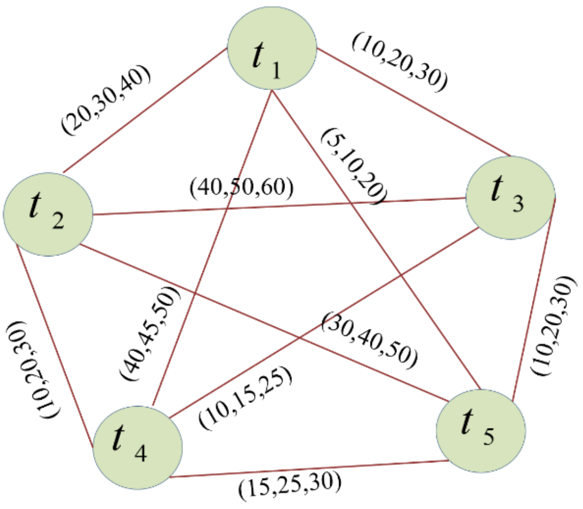
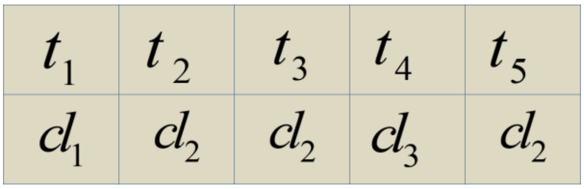


Figure 12. Fuzzy inter-task communication time ($\widehat{c}_{i,j}$) between tasks.

To resolve the aforementioned example, first use Robust's ranking technique to defuzzify the $\widehat{e}_{i,k}$ and $\widehat{c}_{i,j}$ times into crisp times $e_{i,k}$ and $c_{i,j}$ respectively, and then generate E_TM and ITC_TM by inserting $e_{i,k}$ and $c_{i,j}$, respectively in them.

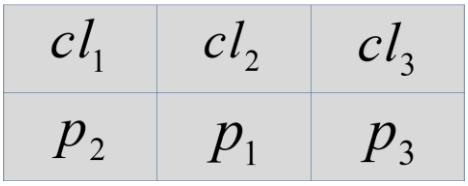
The following chromosome in Figure 13 gives the final task-clusters as a result of carrying out the steps of Algorithm 3:



t_1	t_2	t_3	t_4	t_5
cl_1	cl_2	cl_2	cl_3	cl_2

Figure 13. Final task-clusters of Example 2.

The optimal allocation of task-clusters onto worthy processors has been achieved employing the steps in Algorithm 4, that is displayed by the following chromosome in Figure 14:



cl_1	cl_2	cl_3
p_2	p_1	p_3

Figure 14. Allocation of task-clusters onto processors of Example 2.

The task-clusters and their allotment on processors can be written as follows using the two chromosomes mentioned above:

$$cl_1=\{t_1\}\rightarrow p_2,\ cl_2=\{t_2,t_3,t_5\}\rightarrow p_1 \text{ and } cl_3=\{t_4\}\rightarrow p_3.$$

Task-clusters assignments are shown in Figure 15.

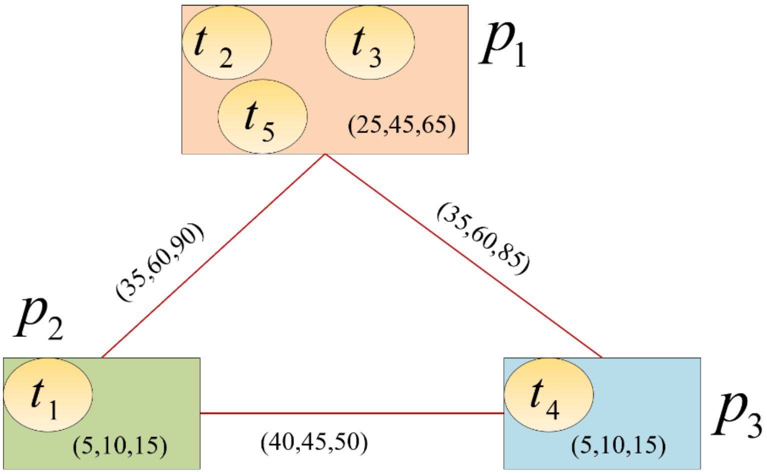


Figure 15. Solution graph of Example 2.

Now, based on the information gleaned from the optimal allocation in Figure 15; the RT is (95,165,240) and the CS is (145,230,320) of the system. Eq (6) determines the system's reliability (RS) that is 0.98780.

Example 3. Here, the problem is grabbed from Sharma et al. [65] model. In which DRTS consist five tasks $\{t_1,t_2,\ldots,t_5\}$ that have to be allocated on three processors $\{p_1,p_2,p_3\}$. In this problem $e_{i,k}$ and $c_{i,j}$ have been assessed as crisp number shown in Figures 16 and 17 respectively.

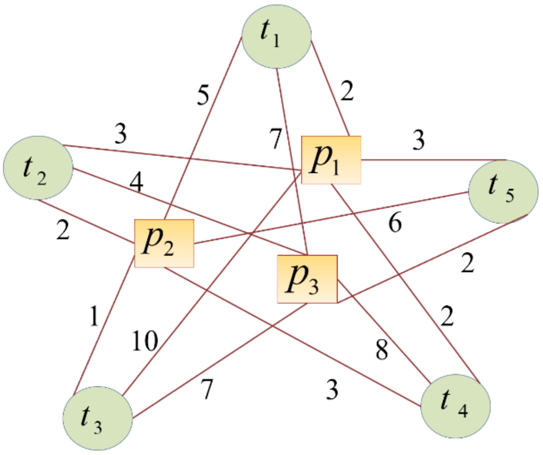


Figure 16. Tasks crisp execution time ($e_{i,k}$) on processors.

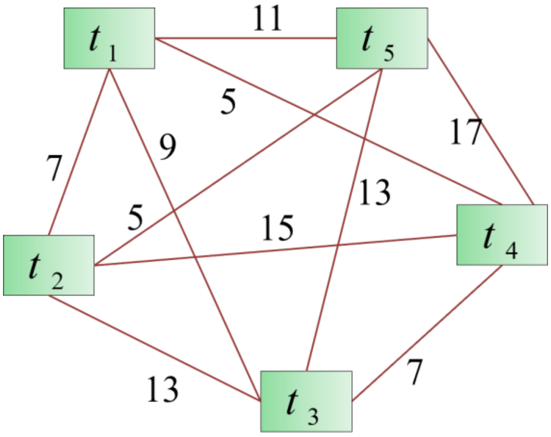


Figure 17. Crisp inter-task communication time ($c_{i,j}$) between tasks.

The following chromosome in Figure 18 gives the final task-clusters as a result of carrying out the steps of Algorithm 3:

t_1	t_2	t_3	t_4	t_5
cl_2	cl_3	cl_1	cl_1	cl_1

Figure 18. Final task-clusters of Example 3.

The optimal allocation of task-clusters onto worthy processors has been achieved employing the steps in Algorithm 4, that is displayed by the following chromosome in Figure 19:

cl_1	cl_2	cl_3
p_2	p_1	p_3

Figure 19. Task-clusters allocation on processors of Example 3.

The task-clusters and their allotment on processors can be written as follows using the two chromosomes mentioned above:

$$cl_1 = \{t_3, t_4, t_5\} \rightarrow p_2,\ \ cl_2 = \{t_1\} \rightarrow p_1 \text{ and } cl_3 = \{t_2\} \rightarrow p_3 .$$

Task-clusters assignments are shown in Figure 20.

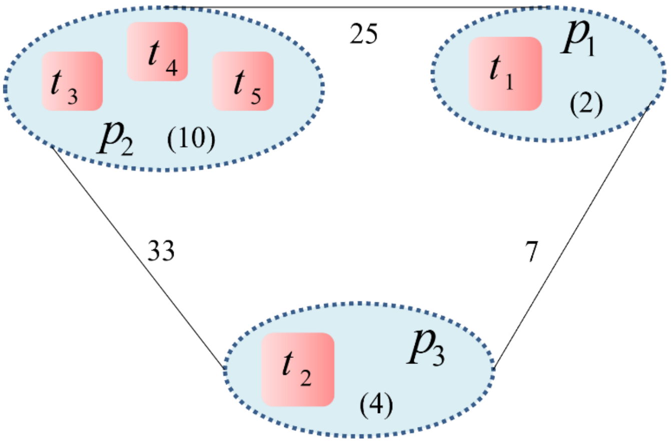


Figure 20. Solution graph of Example 3.

In Figure 20, the processors are enclosed by dotted circles and depict the corresponding $e_{i,k}$ on them, while the figures with solid lines depict $c_{i,j}$. Now, based on the information gleaned from the optimal allocation; the RT is 68 and the CS is 81 of the system. Eq (6) determines the system's reliability (RS), that is 0.9979.

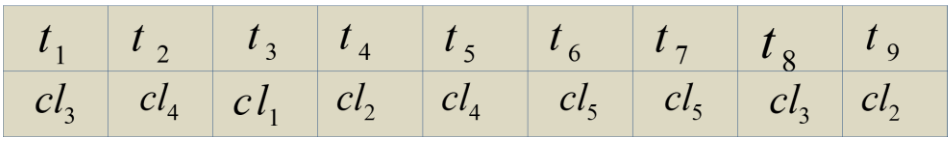
Example 4. Here, the problem is acquired from Chauhan et al. [43] model. In which DRTS consist nine tasks $\{t_1, t_2, \dots, t_9\}$ that have to be allocated on five processors $\{p_1, p_2, \dots, p_5\}$. In this problem $\widehat{e}_{i,k}$ and $\widehat{c}_{i,j}$ have been assessed as fuzzy triangular number shown in the matrices FE_TM and $FITC_TM$ respectively.

$$FE_TM = \left[\widehat{e}_{i,k}\right]_{9\times 5} = \begin{array}{c|ccccc|} & p_1 & p_2 & p_3 & p_4 & p_5 \\ t_1 & (7,9,11) & (8,13,15) & (12,14,15) & (5,8,10) & (11,12,14) \\ t_2 & (1,3,4) & (2,7,10) & (9,11,12) & (6,8,9) & (12,14,16) \\ t_3 & (10,12,14) & (6,8,10) & (3,7,10) & (3,5,7) & (4,6,10) \\ t_4 & (3,7,10) & (4,5,7) & (11,14,17) & (8,10,12) & (7,9,10) \\ t_5 & (10,12,13) & (10,13,15) & (6,8,9) & (6,7,9) & (4,6,8) \\ t_6 & (1,3,5) & (4,5,6) & (10,11,12) & (6,8,10) & (7,9,10) \\ t_7 & (10,14,15) & (10,12,13) & (8,10,11) & (6,9,10) & (8,13,15) \\ t_8 & (5,7,9) & (6,8,10) & (8,10,12) & (9,11,13) & (12,14,16) \\ t_9 & (9,11,13) & (6,10,14) & (5,8,11) & (5,7,9) & (10,13,16) \end{array}.$$

$$FITC_TM=\left[\widehat{c}_{i,j}\right]_{9\times 9}=\begin{array}{c|ccccccccc|} & \mathrm{t}_1 & \mathrm{t}_2 & \mathrm{t}_3 & \mathrm{t}_4 & \mathrm{t}_5 & \mathrm{t}_6 & \mathrm{t}_7 & \mathrm{t}_8 & \mathrm{t}_9 \\ \mathrm{t}_1 & (0,0,0) & (1,2,4) & (2,3,5) & (6,7,8) & (4,5,7) & (4,6,7) & (0,0,0) & (5,6,9) & (3,4,7) \\ \mathrm{t}_2 & (1,2,4) & (0,0,0) & (3,5,6) & (0,0,0) & (6,8,9) & (1,3,4) & (1,2,3) & (3,4,5) & (2,3,4) \\ \mathrm{t}_3 & (2,3,5) & (3,5,6) & (0,0,0) & (1,3,5) & (2,4,6) & (1,3,5) & (0,2,4) & (3,5,7) & (6,7,8) \\ \mathrm{t}_4 & (6,7,8) & (0,0,0) & (1,3,5) & (0,0,0) & (1,3,4) & (1,2,3) & (4,6,7) & (1,3,7) & (6,9,10) \\ \mathrm{t}_5 & (4,5,7) & (6,8,9) & (2,4,6) & (1,3,4) & (0,0,0) & (1,5,6) & (4,7,8) & (0,0,0) & (2,3,4) \\ \mathrm{t}_6 & (4,6,7) & (1,3,4) & (1,3,5) & (1,2,3) & (1,5,6) & (0,0,0) & (8,9,10) & (5,7,9) & (3,4,5) \\ \mathrm{t}_7 & (0,0,0) & (1,2,3) & (0,2,4) & (4,6,7) & (4,7,8) & (8,9,10) & (0,0,0) & (6,7,8) & (4,5,7) \\ \mathrm{t}_8 & (5,6,9) & (3,4,5) & (3,5,7) & (1,3,7) & (0,0,0) & (5,7,9) & (6,7,8) & (0,0,0) & (2,3,4) \\ \mathrm{t}_9 & (3,4,7) & (2,3,4) & (6,7,8) & (6,9,10) & (2,3,4) & (3,4,5) & (4,5,7) & (2,3,4) & (0,0,0) \end{array}.$$

To resolve the aforementioned example, first use Robust's ranking technique to defuzzify the $\widehat{e}_{i,k}$ and $\widehat{c}_{i,j}$ times into crisp times $e_{i,k}$ and $c_{i,j}$ respectively, and then generate E_TM and ITC_TM by inserting $e_{i,k}$ and $c_{i,j}$, respectively in them.

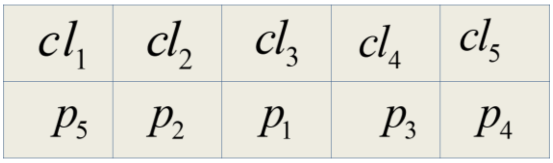
The following chromosome in Figure 21 gives the final task-clusters as a result of carrying out the steps of Algorithm 3:



t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
cl_3	cl_4	cl_1	cl_2	cl_4	cl_5	cl_5	cl_3	cl_2

Figure 21. Final task-clusters of Example 4.

The optimal allocation of task clusters onto worthy processors has been achieved employing the steps in Algorithm 4, that is displayed by the following chromosome in Figure 22:



cl_1	cl_2	cl_3	cl_4	cl_5
p_5	p_2	p_1	p_3	p_4

Figure 22. Task-clusters allocation on processors of Example 4.

The task-clusters and their allotment on processors can be written as follows using the two chromosomes mentioned above:

$$cl_1=\{t_3\}\to p_5,\ cl_2=\{t_4,t_9\}\to p_2,\ cl_3=\{t_1,t_8\}\to p_1,$$

$$cl_4=\{t_2,t_5\}\to p_3 \text{ and } cl_5=\{t_6,t_7\}\to p_4.$$

Task-clusters assignments are shown in Figure 23.

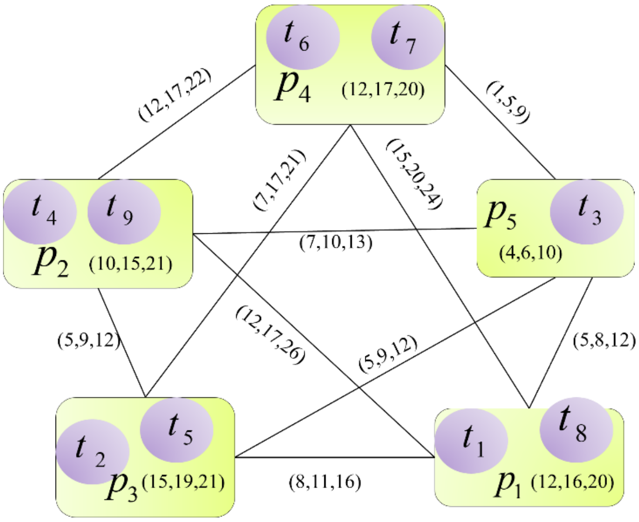


Figure 23. Solution graph of Example 4.

In Figure 23, the processors are enclosed by rounded rectangles and depict the corresponding $e_{i,k}$ on them, while the figures with solid lines in parenthesis depict $c_{i,j}$. Now, based on the information gleaned from the optimal allocation; the RT is (32, 53, 78) and the CS is (403.8, 634.8, 965.8) of the system. Eq (6) determines the system's reliability (RS), that is (0.9349, 0.9506, 0.9693).

7. Comparison

In several situations with variable numbers of tasks and processors, we compared the proposed meta-heuristic based method to well-established approaches in order to assess its effectiveness. CS, RT, RS, PIR, (discussed in Subsections 3.4.–3.7.) efficiency [62] and, resource utilization is being taken into account as performance measures for analytical assessment to analyze the performance of the proposed task assignment algorithm in DRTS. Efficiency is defined as the ratio of sequential computation time to scheduling time, taking the number of processors into consideration. Efficiency and PIR are incredibly beneficial in assessing the accuracy of the outcomes produced by the given approach. In terms of these metrics, the given approach performs better than existing techniques. Also, to examine the quality of outcomes from a statistical perspective, Friedman's test is carried out. Data of different task assignment problems have been accumulated from the articles of Kumar et al. [38], Kumar and Tyagi [39], Chauhan et al. [43], Ilavarasan et al. [66], Kumar and Tyagi [67], Topcuoglu et al. [68]. The following five scenarios have been taken into consideration in this study.

7.1. Scenario 1

In this scenario, 30 real-world issues from the existing methods have been taken into consideration in order to evaluate the proposed technique. Table 6 shows a comparison of the outcomes of these issues based on CS and RT which is also depicted in Figure 24. As per shown in Figure 24, the developed method provides superior quality outcomes than other available task scheduling methods used to reduce CS and RT.

Table 6. Output comparison from different exploratory articles.

S. No.	References	Size of the model (*r, m*)	Technique used	By reference technique		By proposed technique	
				CS	RT	CS	RT
1	Dai and Zhang [69]	(10,3)	Heuristic approach	218	147	199	126
2	Yadav et al. [70]	(4,4)	Heuristic approach	29	27	23	21
3	Kumar et al. [64]	(5,3)	Heuristic approach	246.25	177.5	231.25	166.25
4	Attiya and Hamam [46]	(6,4)	Simulated annealing approach	145	51	60	21
5	Sharma et al. [65]	(5,3)	Clustering approach	123	97	81	68
6	Daoud and Kharma [71]	(5,2)	List based scheduling algorithm	36	27	33	18
7	Topcuoglu et al. [68]	(10,3)	List based scheduling algorithm	230	172	203	128
8	Ilavarasan et al. [66]	(10,3)	HPS approach	236	131	203	128
9	Khandelwal [72]	(8,3)	Clustering approach	122	58	76	46
10	Govil and Kumar [73]	(10,4)	Heuristic approach	163	89	154	77
11	Yadav et al. [74]	(9,3)	Heuristic approach	2563.9	1272.6	2475.70	1181.20
12	Kafil and Ahmad [75]	(5,3)	A* technique	65	28	62	28
13	Kumar and Yadav [76]	(8,3)	Heuristic approach	83	54	76	46
14	Lo [77]	(4,3)	Graph theoretic approach	38	30	35	30
15	Kaushal and Kumar [78]	(10,4)	Heuristic approach	400	200	154	77
16	Kumar et al. [19]	(9,3)	Clustering approach	1094	422	963	415
17	Koppiddakis et al. [79]	(3,2)	Heuristic approach	30	24	14	13
18	Shatz et al. [61]	(4,4)	Heuristic approach	33	21	23	21
19	Akbari and Rashidi [80]	(8,3)	Genetic algorithm	170	111	154	88
20	Bittencourt et al. [81]	(9,3)	DAG scheduling	474	347	474	336
21	Daoud and Kharma [71]	(11,2)	List based scheduling algorithm	188.5	142	157	115
22	Kumar et al. [38]	(5,3)	Genetic algorithm	246.25	177.5	231.25	166.25
23	Yadav et al. [82]	(9,3)	Artificial neural network approach	1372	479	963	415
24	Djigal et al. [63]	(10,3)	List based scheduling algorithm	304	181	275	162
25	Kumar and Tyagi [21]	(6,4)	Clustering approach	84	36	60	21
26	Yadav et al. [83]	(8,4)	Clustering approach	83.8	37	80.2	37
27	Ucar et al. [84]	(7,3)	Heuristic approach	270	195	270	185
28	Kumar and Tyagi [39]	(5,3)	Hybrid Genetic algorithm	246.25	177.5	231.25	166.25
29	Gupta and Yadav [85]	(9,3)	Heuristic approach	2525	1221.9	2475.70	1181.20
30	Elsadek and Wells [86]	(9,3)	Heuristic approach	1372	479	963	415

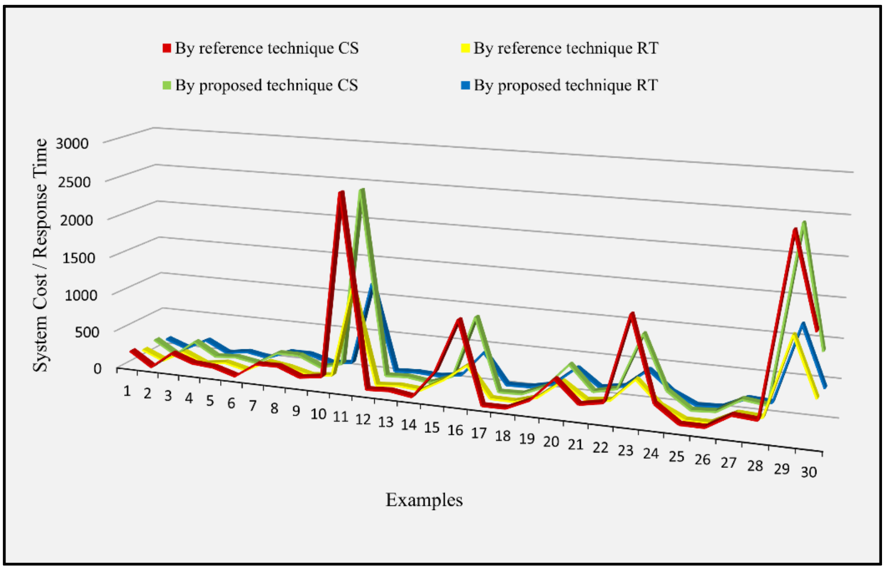


Figure 24. Graphical presentation of Table 6.

7.2. Scenario 2

In this subsection, we will assess the performance of the proposed algorithm by comparing it to that of other well-known algorithms, focusing on PIR %. Table 7 displays the results acquired for PIR %, and it is clear that the developed PSO-based algorithm is superior to these other existing approaches. The hybrid PSO's capability to attain the minimum value for RT in this study, coupled with the close correlation between PIR% and RT, constitutes the primary factors contributing to its performance. In most instances, the proposed algorithm significantly outperforms its well-known competitors, as demonstrated by the PIR values presented in Table 7 and illustrated in Figure 25. The overall average PIR % for the proposed approach, GA-B&B, Clustering approach, GA, and B&B are 22.64%, 19.90%, 17.03%, 20.21%, and 16.19%, respectively.

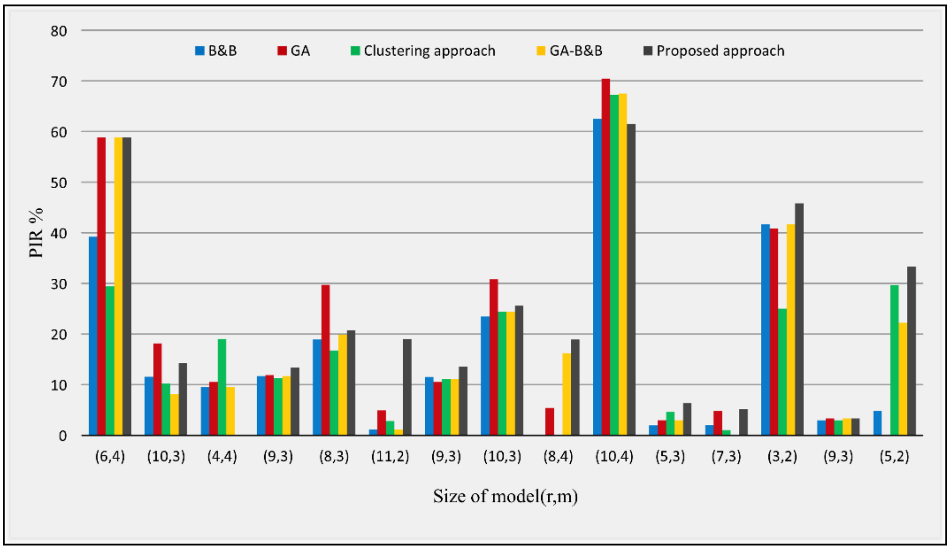


Figure 25. Graphical presentation of Table 7.

Table 7. PIR of the proposed approach in terms of RT for various problem sizes compared to other algorithms.

Size of the model (*r*, *m*)	B&B	GA	Clustering approach	GA-B&B	Proposed approach
(6,4)	39.21	58.82	29.41	58.82	58.82
(10,3)	11.56	18.16	10.20	8.16	14.28
(4,4)	9.52	10.52	19.04	9.52	0
(9,3)	11.69	11.89	11.27	11.69	13.36
(8,3)	18.91	29.72	16.76	19.81	20.72
(11,2)	1.16	4.92	2.81	1.16	19.01
(9,3)	11.52	10.52	11.10	11.11	13.53
(10,3)	23.46	30.81	24.41	24.41	25.58
(8,4)	0	5.40	0	16.21	18.91
(10,4)	62.5	70.45	67.23	67.5	61.50
(5,3)	1.97	2.98	4.61	2.98	6.33
(7,3)	2.05	4.87	1.02	0	5.12
(3,2)	41.66	40.83	25	41.66	45.83
(9,3)	2.97	3.33	2.97	3.33	3.33
(5,2)	4.81	0	29.62	22.22	33.33

7.3. Scenario 3

The performance study of the suggested method based on RS and CS is presented in this subsection. By resolving the running problem from Topcuoglu et al. [68] article, the effectiveness of the developed approach has been demonstrated. In Table 8, which is provided below, are the outcomes for this problem using the proposed algorithm, the Chauhan et al. [43] algorithm, the Kumar et al. [19] algorithm, the Kumar and Tyagi [39] algorithm, and the algorithm of Ilavarasan et al. [66]. In light of this, it can be said that the proposed algorithm produces optimal allocation when compared to the other algorithms listed in Table 8.

Table 8. Comparative results of Topcuoglu et al. [68] algorithm with proposed and other existing algorithms.

S. No.	Algorithms	Tasks	Processors	RS	CS
1	Topcuoglu et al. [68]	$\{t_3,t_7\}$	p_1	0.9614	230
		$\{t_4,t_6,t_8\}$	p_3		
		$\{t_1,t_2,t_5,t_9,t_{10}\}$	p_2		
2	Kumar et al. [19]	$\{t_3,t_7,t_{10}\}$	p_1	0.9629	224
		$\{t_4,t_8,t_9\}$	p_2		
		$\{t_1,t_2,t_5,t_6\}$	p_3		
3	Chauhan et al. [43]	$\{t_3,t_7,t_{10}\}$	p_1	0.9629	224
		$\{t_4,t_8,t_9\}$	p_2		
		$\{t_1,t_2,t_5,t_6\}$	p_3		
4	Ilavarasan et al. [66]	$\{t_3,t_7,t_8\}$	p_1	-	236
		$\{t_1,t_2,t_6\}$	p_3		
		$\{t_4,t_5,t_9,t_{10}\}$	p_2		

Continued on next page

S. No.	Algorithms	Tasks	Processors	RS	CS
5	Kumar and Tyagi [39]	$\{t_4,t_8,t_9\}$	p_2	0.9629	224
		$\{t_3,t_7,t_{10}\}$	p_1		
		$\{t_1,t_2,t_5,t_6\}$	p_3		
6	Proposed algorithm	$\{t_3,t_7,t_{10}\}$	p_2	0.9840	203
		$\{t_2,t_4,t_8,t_9\}$	p_1		
		$\{t_1,t_5,t_6\}$	p_3		

7.4. Scenario 4

This subsection presents the performance analysis of the specified approach based on RT, efficiency, and resource utilization. The article by Chauhan et al. [43] has been used to consider various task assignment issues where GA-based mechanisms were generated. These problems are also implemented on the proposed model and the obtained results are tabulated in Table 9. From the study of Table 9, it can be concluded that the developed method produces finer results in terms of considered parameters. Table 9 concludes that proposed PSO-GA algorithm generates results of higher quality than to GA algorithm having RT with an average 170.16 and 181.04 respectively. Results are also depicted in Figure 26 with respect to efficiency. The outcomes of Figure 26 shows that the suggested PSO-GA based model performs better than the GA based model in terms of efficiency for the majority of the processor quantities, while for the remaining processor quantities, both models deliver similar results. The average efficiency of the GA and PSO-GA algorithms is 0.6387 and 0.7013, respectively.

Table 9. Comparison study of GA with proposed PSO-GA in terms of RT, efficiency & utilization under different size of problems.

S. No.	Tasks & Processors	GA			PSO-GA		
		RT	Efficiency	Utilization (%)	RT	Efficiency	Utilization (%)
1	(9,3)	422	0.7867	81.37	415	0.8001	83.69
2	(6,4)	21	0.7976	78.66	21	0.7976	78.66
3	(4,3)	65	0.5246	-	30	0.6105	67.31
4	(6,4)	34	0.6126	52.94	21	0.7976	78.66
5	(9,3)	120	0.5629	-	118	0.5801	61.13
6	(8,4)	37	0.7901	-	37	0.7901	78.38
7	(9,3)	1181.2	0.6183	-	1181.20	0.6183	69.23
8	(4,3)	19	0.6140	63.25	19	0.6140	63.25
9	(5,2)	86	0.3139	-	18	0.7502	79.33
10	(10,4)	85	0.5973	-	77	0.6421	71.77
11	(7,3)	195	0.4708	-	185	0.5954	63.13
12	(10,4)	65	0.7803	-	58	0.7992	81.02
13	(10,3)	130	0.5385	55.38	128	0.5891	78.41
14	(8,3)	78	0.7629	77.38	78	0.7629	77.38
15	(5,3)	177.5	0.7102	-	166.25	0.7732	81.79

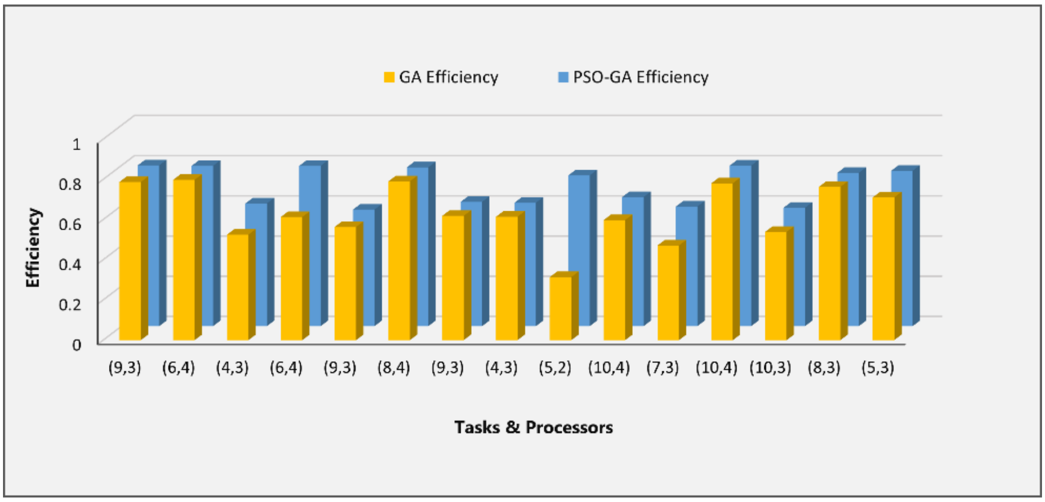


Figure 26. Graphical presentation of Table 9 in terms of efficiency.

7.5. Scenario 5

In this subsection, to examine the quality of outcomes from a statistical perspective, Friedman test is carried out. This test analysis whether there are statistically significant differences between the dependent groups. The Friedman's test is a non-parametric statistical test which is used to assess the significance of the data given in Tables 7 and 10 displays the results of this test.

Table 10. Outcomes of Friedman's test in terms of PIR.

S. No.	**Problem's size**	**Ranks of the algorithms**				
		B&B	**GA**	**Clustering approach**	**GA-B&B**	**Proposed technique**
1	6×4	4	**2**	5	**2**	**2**
2	10×3	3	**1**	4	5	2
3	4×4	3.5	2	**1**	3.5	5
4	9×3	3.5	2	5	3.5	**1**
5	8×3	4	**1**	5	3	2
6	11×2	4.5	2	3	4.5	**1**
7	9×3	2	5	4	3	**1**
8	10×3	5	**1**	3.5	3.5	2
9	8×4	4.5	3	4.5	2	**1**
10	10×4	4	**1**	3	2	5
11	5×3	5	3.5	2	3.5	**1**
12	7×3	3	2	4	5	**1**
13	3×2	2.5	4	5	2.5	**1**
14	9×3	3.5	**2**	3.5	**2**	**2**
15	5×2	4	5	2	3	**1**
Sum of ranks		56	36.5	54.5	48	28
Sum of ranks squared		3136	1332.25	2970.25	2304	784
Average of ranks		3.73	2.43	3.63	3.2	1.86

It is noticeable that the developed method yields the best average of ranks as compared to B&B, GA, Clustering method, and GA-B&B. It is clear proof that the proposed PSO-GA based algorithm is a promising technique for tasks scheduling. In the outcomes of Table 7, the χ_r^2 statistic is 10.686 (4, N=15) and the p-value is 0.00226. Hence the results of the proposed method are significant at $p< 0.05$.

8. Conclusions

The problem of task scheduling is resolved in this article using a PSO-based approach. The purpose of this technique is to use evolutionary algorithms for static task scheduling onto heterogeneous processors in DRTS. The present research demonstrates an efficient method for handling the scheduling issue with different objectives simultaneously, such as response time, total cost, and system reliability optimization. So far, hybrid PSO-GA model has not been applied in this manner to solve this type of problem to handle all these objectives simultaneously. Throughout this article, two algorithms have been developed: HPSOGAK, a combination of PSO, GA, and *k*-means for task-cluster formation to minimize ITC_T, and GA for allocation of tasks to processors to minimize total CS, RT and maximize RS of the system. The following are the essential contributions of this work:

(i) In this study, a scheduling mechanism based on the evolutionary algorithms PSO and GA is designed to deal with the real-life applications.

(ii) PSO is employed in this study as it is the best approach for all size problems, and PSO and GA are integrated to enhance traditional PSO's functionality and address its primary shortcomings.

(iii) By providing new encoding, population initialization, new crossover and mutation approaches, the convergence rate of GA is improved.

(iv) Both crisp and fuzzy times can be used with the model presented in heterogeneous system.

(v) Numerous real-world problems have been solved in order to compare the proposed model's performance to that of existing methods.

(vi) In terms of PIR %, RT, CS, RS, efficiency, and resource utilization, the accuracy of the presented model is examined. In all of these, as discussed in the comparison section, the proposed technique delivers superior results when compared to different meta-heuristics and traditional algorithms.

(vii) Friedman's test is applied to statistically evaluate the quality of results.

(viii) The run time complexity of various existing techniques, including those by Elsadek and Wells [86], Kumar et al. [19], Kumar and Tyagi [39], are O (r^2+m^2+r^2m), O (r^2+rm) and O (rm+m^2+m) respectively. Whereas the proposed technique's run time complexity is O (m^2 +rm), which is lower than that of existing approaches, when ($r>m$).

(ix) In DRTS, the described model is applicable any number of jobs/tasks and processors.

Implementation results demonstrate that the developed model is more effective than existing methods. From the obtained results, it can be concluded that proposed approach is suitable to deal with the issues of tasks scheduling. According to these results, the proposed technique is a worthwhile substitute for resolving the task allocation issue but the given model also has some limitations. The PSO can easily fall into the local optimum because of its slow convergence rate during the iterative procedure. Consequently, the HPSOGAK algorithm also has certain limitations. In future study we shall address the comparison of the algorithm's performance using both fuzzy and crisp times. Specifically, case studies will be conducted to compare the same tasks and processors with and without defuzzification applied. This approach will help to further evaluate the robustness and versatility of the algorithm, enhancing the comprehensiveness of the findings. Additionally, the advantages of various algorithms, such as DE algorithm, WOA, and GWO, will be utilized for further algorithm enhancement. By employing these techniques, the algorithm's overall scheduling performance can be improved. To continually enhance the task assignment strategy, the task scheduling policy in the dynamic environment of the DRTS will be considered, and the implications of PSO parameters and local search techniques on the system's overall performance will be examined.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors are grateful to the Editor-in-Chief, associate editor, and learned reviewers for their critical comments and valuable suggestions, which led to a fine version of the manuscript.

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. R. Mall, *Real-time systems: theory and practice*, Pearson Education India, 3 Eds., 2009.
2. H. Jin, P. Tan, A novel dynamic allocation and scheduling scheme with CPNA and FCF algorithms in distributed real-time systems, *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, **1** (2005), 550–556. https://doi.org/10.1109/ICPADS.2005.38
3. Y. Singh, M. Popli, S. S. P. Shukla, Energy reduction in weakly hard real time systems, *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, 2012, 909–915. https://doi.org/10.1109/RAIT.2012.6194555
4. V. Jeyakrishnan, P. Sengottuvelan, A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms, *Wireless Pers. Commun.*, **94** (2017), 2363–2375. https://doi.org/10.1007/s11277-016-3481-8
5. V. M. A. Xavier, S. Annadurai, Chaotic social spider algorithm for load balance aware task scheduling in cloud computing, *Cluster Comput.*, **22** (2019), 287–297. https://doi.org/10.1007/s10586-018-1823-x
6. X. Huang, C. Li, H. Chen, D. An, Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies, *Cluster Comput.*, **23** (2020), 1137–1147. https://doi.org/10.1007/s10586-019-02983-5
7. L. Abualigah, M. A. Elaziz, P. Sumari, Z. W. Geem, A. H. Gandomi, Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer, *Expert Syst. Appl.*, **191** (2022), 116158. https://doi.org/10.1016/j.eswa.2021.116158
8. R. I. Davis, A. Burns, A survey of hard real-time scheduling for multiprocessor systems, *ACM Comput. Surv. (CSUR)*, **43** (2011), 1–44. https://doi.org/10.1145/1978802.1978814

9. Y. Zhang, A. Sivasubramaniam, J. Moreira, H. Franke, Impact of workload and system parameters on next generation cluster scheduling mechanisms, *IEEE Trans. Parall. Distr. Syst.*, **12** (2001), 967–985. https://doi.org/10.1109/71.954632
10. H. Casanova, A. Legrand, D. Zagorodnov, F. Berman, Heuristics for scheduling parameter sweep applications in grid environments, *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000)*, 2000, 349–363. https://doi.org/10.1109/HCW.2000.843757
11. J. Kennedy, R. C. Eberhart, Particle swarm optimization, *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000)*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
12. D. E. Goldberg, *Genetic algorithm in search, optimization and machine learning*, Boston: Addison-Wesley Longman Publishing Co., Inc., 1989.
13. Z. Wu, X. Liu, Z. Ni, D. Yuan, Y. Yang, A market-oriented hierarchical scheduling strategy in cloud workflow systems, *J. Supercomput.*, **63** (2011), 256–293. https://doi.org/10.1007/s11227-011-0578-4
14. M. Naderam, M. Dehgham, H. Pedram, Upper and lower bounds for dynamic cluster assignment for multi-agent tracking in heterogeneous WSNs, *J. Parall. Distr. Com.*, **73** (2012), 1389–1399. https://doi.org/10.1016/j.jpdc.2013.04.007
15. L. Wang, S. U. Khan, D. Chen, J. Kolodziej, R. Ranian, C. Z. Xu, et al., Energy-aware parallel task scheduling in a cluster, *Future Gener. Comp. Sy.*, **29** (2013), 1661–1670. https://doi.org/10.1016/j.future.2013.02.010
16. B. Tripathy, S. Dash, S. K. Padhy, Dynamic task scheduling using a directed neural network, *J. Parall. Distr. Com.*, **75** (2015), 101–106. https://doi.org/10.1016/j.jpdc.2014.09.015
17. Y. Xiao, Z. Ren, H. Zhang, C. Chen, C. Shi, A novel task allocation for maximizing reliability considering fault-tolerant in VANET real time systems, *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, 1–7. https://doi.org/10.1109/PIMRC.2017.8292511
18. P. Neamatollahi, S. Abrishami, M. Naghibzadeh, M. H. Y. Moghaddam, O. Younis, Hierarchical clustering-task scheduling policy in cluster-based wireless sensor networks, *IEEE Trans. Ind. Inform.*, **14** (2018), 1876–1886. https://doi.org/10.1109/TII.2017.2757606
19. H. Kumar, N. K. Chauhan, P. K. Yadav, A high performance model for task allocation in distributed computing system using k-means clustering technique, In: *Research anthology on architectures, frameworks, and integration strategies for distributed and cloud computing*, **9** (2021), 1244–1268. https://doi.org/10.4018/978-1-7998-5339-8.ch060
20. T. K. Dao, T. S. Pan, T. T. Nguyen, J. S. Pan, Parallel bat algorithm for optimizing makespan in job shop scheduling problems, *J. Intell. Manuf.*, **29** (2018), 451–462. https://doi.org/10.1007/s10845-015-1121-x
21. H. Kumar, I. Tyagi, Implementation and comparative analysis of k-means and fuzzy c-means clustering algorithms for tasks allocation distributed real time system, *Int. J. Embedded Real-Time Commun. Syst.*, **10** (2019), 66–86. https://doi.org/10.4018/IJERTCS.2019040105
22. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris Hawks optimization: algorithm and applications, *Future Gene. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028

23. F. Alkhateeb, B. H. Abed-alguni, A hybrid cuckoo search and simulated annealing algorithm, *J. Intell. Syst.*, **28** (2019), 683–898. https://doi.org/10.1515/jisys-2017-0268
24. E. B. Tirkolaee, A. Goli, G. W. Weber, Fuzzy mathematical programming and self-adaptive artificial fish swarm algorithm for just-in-time energy-aware flow shop scheduling problem with outsourcing option, *IEEE Trans. Fuzzy Syst.*, **28** (2020), 2772–2783. https://doi.org/10.1109/TFUZZ.2020.2998174
25. H. Kanemitsu, M. Hanada, H. Nakazato, Clustering-based task scheduling in a large number of heterogeneous processors, *IEEE Trans. Parall. Distr. Syst.*, **27** (2016), 3144–3157. https://doi.org/10.1109/TPDS.2016.2526682
26. K. Mishra, S. K. Majhi, A binary bird swarm optimization based load balancing algorithm for cloud computing environment, *Open Comput. Sci.*, **11** (2021), 146–160. https://doi.org/10.1515/comp-2020-0215
27. N. A. Alawad, B. H. Abed-alguni, Discrete Jaya with refraction learning and three mutation methods for the permutation flow shop scheduling problem, *J. Supercomput.*, **78** (2022), 3517–3538. https://doi.org/10.1007/s11227-021-03998-9
28. M. Haris, S. Zubair, Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing, *J. King Sau. Univ.-Comput. Inf. Sci.*, **34** (2022), 9696–9709. https://doi.org/10.1016/j.jksuci.2021.12.003
29. M. Agarwal, G. M. S. Srivastava, Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment, *Int. J. Inf. Tech. Dec. Making*, **17** (2018), 1237–1267. https://doi.org/10.1142/S0219622018500244
30. Y. Kang, H. Lu, J. He, A PSO-based genetic algorithm for scheduling of tasks in a heterogeneous distributed system, *J. Soft.*, **8** (2013), 1443–1450. https://doi.org/10.4304/jsw.8.6.1443-1450
31. A. K. Samal, R. Mall, C. Tripathy, Fault tolerant scheduling of hard real-time tasks on multiprocessor system using a hybrid genetic algorithm, *Swarm Evol. Comput.*, **14** (2014), 92–105. https://doi.org/10.1016/j.swevo.2013.10.002
32. I. R. K. Raju, P. S. Varma, M. V. R. Sundari, G. J. Moses, Deadline aware two stage scheduling algorithm in cloud computing, *Indian J. Sci. Tech.*, **9** (2016), 1–10. https://doi.org/10.17485/ijst/2016/v9i4/80553
33. M. Mutingi, C. Mbohwa, Modeling supplier selection using multi-criterion fuzzy grouping genetic algorithm, In: *Grouping genetic algorithms: studies in computational intelligence*, Cham: Springer, **666** (2017), 213–228. https://doi.org/10.1007/978-3-319-44394-2_12
34. Z. Zhou, J. Chang, Z. Hu, J. Yu, F. Li, A modified PSO algorithm for task scheduling optimization in cloud computing, *Concur. Comput.: Pract. Exper.*, **30** (2018), e4970. https://doi.org/10.1002/cpe.4970
35. J. Luan, Z. Yao, F. Zhao, X. Song, A novel method to solve supplier selection problem: Hybrid algorithm of genetic algorithm and ant colony optimization, *Math. Comput. Simul.*, **156** (2018), 294–309. https://doi.org/10.1016/j.matcom.2018.08.011
36. L. Tang, X. Zhang, Z. Li, Y. Zhang, A new hybrid task scheduling algorithm designed based on ACO and GA, *J. Inf. Hiding Multim. Signal. Process.*, **9** (2018), 1585–1594.
37. J. P. B. Mapetu, Z. Chen, L. Kong, Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing, *Appl. Intell.*, **49** (2019), 3308–3330. https://doi.org/10.1007/s10489-019-01448-x

38. H. Kumar, N. K. Chauhan, P. K. Yadav, Hybrid genetic algorithm for task scheduling in distributed real-time system, *Inter. J. Syst., Cont., Commun.*, **10** (2019), 32–52. https://doi.org/10.1504/IJSCC.2019.097417
39. H. Kumar, I. Tyagi, Hybrid model for tasks scheduling in distributed real time system, *J. Ambient Intell. Human Comput.*, **12** (2021), 2881–2903. https://doi.org/10.1007/s12652-020-02445-6
40. S. Devi, D. Garg, Hybrid genetic and particle swarm algorithm: redundancy allocation problem, *Int. J. Syst. Assur. Eng. Manag.*, **11** (2020), 313–319. https://doi.org/10.1007/s13198-019-00858-x
41. M. Agarwal, G. M. S. Srivastava, Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing, *J. Ambient Intell. Human. Comput.*, **12** (2020), 9855–9875. https://doi.org/10.1007/s12652-020-02730-4
42. H. Zhang, F. Liu, Y. Zhou, Z. Zhang, A hybrid method integrating an elite genetic algorithm with tabu search for the quadratic assignment problem, *Inf. Sci.*, **539** (2020), 347–374. https://doi.org/10.1016/j.ins.2020.06.036
43. N. K. Chauhan, I. Tyagi, H. Kumar, D. Sharma, Tasks scheduling through hybrid genetic algorithm in real-time system on heterogeneous environment, *SN Comput. Sci.*, **3** (2022), 75. https://doi.org/10.1007/s42979-021-00959-0
44. A. Amirteimoori, I. Mahdavi, M. Solimanpur, S. S. Ali, E. B. Tirkolaee, A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation, *Comput. Ind. Eng.*, **173** (2022), 108672. https://doi.org/10.1016/j.cie.2022.108672
45. Karishma, H. Kumar, A new hybrid particle swarm optimization algorithm for optimal tasks scheduling in distributed computing system, *Intell. Syst. Appl.*, **18** (2023), 200219. https://doi.org/10.1016/j.iswa.2023.200219
46. G. Attiya, Y. Hamam, Task allocation for maximizing reliability of distributed systems: a simulating annealing approach, *J. Parallel Distr. Com.*, **66** (2006), 1259–1266. https://doi.org/10.1016/j.jpdc.2006.06.006
47. M. Jiang, J. Zhou, M. Hu, Fuzzy reliability analysis of disk array systems, *2007 Chinese Control Conference*, 2007, 314–317. https://doi.org/10.1109/CHICC.2006.4347012
48. Q. M. Kang, H. He, H. M. Song, R. Deng, Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization, *J. Syst. Soft.*, **83** (2010), 2165–2174. https://doi.org/10.1016/j.jss.2010.06.024
49. S. S. Donight, S. Khanmohammadi, A fuzzy reliability model for series-parallel system, *J. Ind. Eng. Int.*, **7** (2011), 10–18.
50. K. Noori, K. Jenab, Fuzzy reliability-based traction control model for intelligent transportation systems, *IEEE Trans. Syst. Man Cybern.: syst.*, **43** (2012), 229–234. https://doi.org/10.1109/TSMCA.2012.2204047
51. L. Cederholm, N. Petterson, *Distributed real time system survey*, Sweden: Mälardalen University, 2009.
52. J. Li, R. Guo, Z. Shao, The research of scheduling algorithm in real-time system, *2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, 2010, 333–336. https://doi.org/10.1109/CCTAE.2010.5544771
53. Y. Li, A. M. K. Cheng, Transparent real-time task scheduling on temporal resource partitions, *IEEE Trans. Comput.*, **65** (2015), 1646–1655. https://doi.org/10.1109/TC.2015.2449857

54. S. A. Narale, P. K. Butey, Throttled load balancing scheduling policy assist to reduce grand total cost and data center processing time in cloud environment using cloud analyst, *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, 1464–1467. https://doi.org/10.1109/ICICCT.2018.8473062
55. M. Adhikari, S. Nandy, T. Amgoth, Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud, *J. Netw. Comput. Appl.*, **128** (2019), 64–77. https://doi.org/10.1016/j.jnca.2018.12.010
56. G. Li, Z. Wu, Ant colony optimization task scheduling algorithm for SWIM based on load balancing, *Future Internet*, **11** (2019), 90. https://doi.org/10.3390/fi11040090
57. Z. Shao, D. Pi, W. Shao, Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment, *Expert Syst. Appl.*, **145** (2020). https://doi.org/10.1016/j.eswa.2019.113147
58. P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, R. Ghaemi, A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm, *J. Parallel. Distr. Com.*, **143** (2020), 88–96. https://doi.org/10.1016/j.jpdc.2020.04.008
59. S. Negi, M. M. S. Rauthan, K. S. Vaisla, N. Panwar, CMODLB: an efficient load balancing approach in cloud computing environment. *J. Supercomp.*, **77** (2021), 8787–8839. https://doi.org/10.1007/s11227-020-03601-7
60. G. A. P. Princess, A. S. Radhamani, A hybrid meta-heuristic for optimal load balancing in cloud computing, *J. Grid Comput.*, **19** (2021), 21. https://doi.org/10.1007/s10723-021-09560-4
61. S. M. Shatz, J. P. Wang, M. Goto, Task allocation for maximizing reliability of distributed computing system, *IEEE Trans. Comput.*, **41** (1992), 1156–1168. https://doi.org/10.1109/12.165396
62. D. M. Abdelkader, F. Omara, Dynamic task scheduling algorithm with load balancing for heterogeneous computing system, *Egypt. Inform. J.*, **13** (2012), 135–145. https://doi.org/10.1016/j.eij.2012.04.001
63. H. Djigal, J. Feng, J. Lu, Task scheduling for heterogeneous computing using a predict cost matrix, *ICPP Workshops '19: Workshop Proceedings of the 48th International Conference on Parallel Processing*, 2019, 1–10. https://doi.org/10.1145/3339186.3339206
64. H. Kumar, M. P. Singh, P. K. Yadav, A tasks allocation model with fuzzy execution and fuzzy inter-tasks communication times in a distributed computing system, *Int. J. Comput. Appl.*, **72** (2013), 24–31.
65. M. Sharma, H. Kumar, D. Garg, An optimal task allocation model through clustering with inter-processor distances in heterogeneous distributed computing systems, *Int. J. Soft Comput. Eng. (IJSCE)*, **2** (2012), 50–55.
66. E. Ilavarasan, P. Thambidurai, R. Mahilmannan, High performance task scheduling algorithm for heterogeneous computing system, In: M. Hobbs, A. M. Goscinski, W. Zhou, *Distributed and parallel computing. ICA3PP 2005*, Lecture Notes in Computer Science, Springe, **3719** (2005), 193–203. https://doi.org/10.1007/11564621_22
67. H. Kumar, I. Tyagi, A new hybrid optimization technique for scheduling of periodic and non-periodic tasks, *Augment Hum. Res.*, **6** (2021), 11. https://doi.org/10.1007/s41133-021-00049-z

68. H. Topcuoglu, S. Hariri, M. Y. Wu, Performance-effective and low complexity task scheduling for heterogeneous computing, *IEEE Trans. Parallel Distr. Syst.*, **13** (2002), 260–274. https://doi.org/10.1109/71.993206
69. Y. Dai, X. Zhang, A synthesized heuristic task scheduling algorithm, *Sci. World J.*, **2014** (2014), 465702. https://doi.org/10.1155/2014/465702
70. P. K. Yadav, M. P. Singh, K. Sharma, Task allocation model for reliability and cost optimization in distributed computing system, *Int. J. Model. Simul. Sci. Comput.*, **2** (2011), 131–149. https://doi.org/10.1142/S179396231100044X
71. M. I. Daoud, N. Kharma, A high performance algorithm for static task scheduling in heterogeneous distributed computing systems, *J. Parallel Distr. Comput.*, **68** (2008), 399–409. https://doi.org/10.1016/j.jpdc.2007.05.015
72. A. Khandelwal, Optimal execution cost of distributed system: through clustering, *Int. J. Eng. Sci. Tech.*, **3** (2011), 2320–2328.
73. K. Govil, A. Kumar, A modified and efficient algorithm for static task assignment in distributed processing environment, *Int. J. Comput. Appl.*, **23** (2011), 1–5.
74. P. K. Yadav, P. P. Singh, P. Pradhan, A task allocation algorithm for optimum utilization of processor in heterogeneous distributed system, *Int. J. Res. Rev. Eng. Sci. Tech.*, **2** (2013), 153–160.
75. M. Kafil, I. Ahmad, Optimal task assignment in heterogeneous distributed computing systems, *IEEE Concurrency*, **6** (1998), 42–50. https://doi.org/10.1109/4434.708255
76. A. Kumar, P. K. Yadav, Task management algorithm for distributed system, *The 15th International Conference of International Academy of Physical Sciences*, 2014.
77. V. M. Lo, Heuristic algorithms for task assignment in distributed system, *IEEE Trans. Comput.*, **37** (1988), 1384–1397. https://doi.org/10.1109/12.8704
78. U. Kaushal, A. Kumar, Improving the performance of DRTS by optimal allocation of multiple tasks under dynamic load sharing scheme, *Int. J. Sci. Eng. Res.*, **4** (2013), 1316–1321.
79. Y. Kopiddakis, M. Lamari, V. Zissimopoulos, On the task assignment problem: two new heuristic algorithms, *J. Parallel Dist. Comput.*, **42** (1997), 21–29. https://doi.org/10.1006/jpdc.1997.1311
80. M. Akbari, H. Rashidi, A multi objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems, *Expert Syst. Appl.*, **60** (2016), 234–248. https://doi.org/10.1016/j.eswa.2016.05.014
81. L. F. Bittencourt, R. Sakellariou, E. R. M. Madeira, DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm, *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, 27–34. https://doi.org/10.1109/PDP.2010.56
82. P. K. Yadav, M. P. Singh, A. Kumar, B. Agarwal, An efficient tasks scheduling model in distributed processing systems using ANN, *Int. J. Circuits Syst.*, **1** (2011), 53–66.
83. P. K. Yadav, P. Pradhan, P. P. Singh, A fuzzy clustering method to minimize the inter task communication effect for optimal utilization of processor's capacity in distributed real time systems, In: K. Deep, A. Nagar, M. Pant, J. Bansal, *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20–22, 2011,* Advances in Intelligent and Soft Computing, Springer, **130** (2012), 159–168. https://doi.org/10.1007/978-81-322-0487-9_16

84. B. Ucar, C. Aykanat, K. Kaya, M. Ikinci, Task assignment in heterogeneous computing systems, *J. Parallel Dist. Comput.*, **66** (2006), 32–46. https://doi.org/10.1016/j.jpdc.2005.06.014
85. R. Gupta, P. K. Yadav, Task allocation model for balance utilization of available resource in multiprocessor environment, *J. Comput. Eng.*, **17** (2015), 94–99. https://doi.org/10.9790/0661-17419499
86. A. A. Elsadek, B. E. Wells, A heuristic model for task allocation in heterogeneous distributed computing system, *Int. J. Comput. Appl.*, **6** (1999), 1–36.