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:

<table>
<thead>
<tr>
<th>Indices</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i, j$</td>
<td>Task index, $i, j = 1, 2, \ldots, r$</td>
</tr>
<tr>
<td>$k, l$</td>
<td>Processor index/Clustering index, $k, l = 1, 2, \ldots, m$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T = {t_i : i = 1, 2, \ldots, r}$</td>
<td>Set of $r$ number of tasks</td>
</tr>
<tr>
<td>$P = {p_k : k = 1, 2, \ldots, m}$</td>
<td>Set of $m$ number of processors</td>
</tr>
<tr>
<td>$\hat{e}_{i,k}$</td>
<td>Fuzzy $E_T$ of $i^{th}$ task on $k^{th}$ processor</td>
</tr>
<tr>
<td>$FE_T = \left[ \hat{e}_{i,k} \right]$</td>
<td>Fuzzy $E_T$ matrix</td>
</tr>
<tr>
<td>$\hat{c}_{i,j}$</td>
<td>Fuzzy $ITC_T$ between tasks $t_i$ and $t_j$</td>
</tr>
<tr>
<td>$FITC_T = \left[ \hat{c}_{i,j} \right]$</td>
<td>Fuzzy $ITC_T$ matrix</td>
</tr>
<tr>
<td>$e_{i,k}$</td>
<td>Crisp $E_T$ of task $t_i$ on $p_k$ processor</td>
</tr>
<tr>
<td>$E_T = \left[ e_{i,k} \right]$</td>
<td>$E_T$ matrix</td>
</tr>
<tr>
<td>$c_{i,j}$</td>
<td>Crisp $ITC_T$ between tasks $t_i$ and $t_j$</td>
</tr>
<tr>
<td>$ITC_T = \left[ c_{i,j} \right]$</td>
<td>$ITC_T$ matrix</td>
</tr>
<tr>
<td>$NE_T = \left[ e^a_{i,k} \right]$</td>
<td>New crisp $E_T$ matrix</td>
</tr>
<tr>
<td>$NITC_T = \left[ c^a_{i,j} \right]$</td>
<td>New crisp $ITC_T$ matrix</td>
</tr>
<tr>
<td>$R_p$</td>
<td>Reliability of processor $p_k$</td>
</tr>
<tr>
<td>$R_{kl}$</td>
<td>Reliability of $kl$ communication link</td>
</tr>
<tr>
<td>$\lambda_p$</td>
<td>Failure rate of processor $p_k$</td>
</tr>
<tr>
<td>$\mu_{kl}$</td>
<td>Failure rate of $kl$ connecting path</td>
</tr>
<tr>
<td>$c l_k$</td>
<td>$k^{th}$ task cluster</td>
</tr>
<tr>
<td>$C d_k$</td>
<td>$k^{th}$ cluster centroid</td>
</tr>
<tr>
<td>$V p_k (t)$</td>
<td>Velocity of $k^{th}$ particle</td>
</tr>
<tr>
<td>$P p_k (t)$</td>
<td>Local best fitness of $k^{th}$ particle</td>
</tr>
<tr>
<td>$G p_k (t)$</td>
<td>Globally best fitness of $k^{th}$ particle</td>
</tr>
<tr>
<td>$g_{c d}$</td>
<td>Fitness value of a particle in PSO</td>
</tr>
</tbody>
</table>

Decision variables:

- $x_{i,k}$: \begin{cases} 1, & \text{if $t_i$ task is allocated to $p_k$ processor} \\ 0, & \text{otherwise} \end{cases}
- $d_{k,l}$: Inter-processor distance with unit data transfer where $d_{k,k}=0$ if $k=l$.
- $u_{i,k}$: \begin{cases} 1, & \text{if $t_i$ task does belong to cluster $c l_k$} \\ 0, & \text{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). 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.

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

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

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

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

3.1. Defuzzification

Robust’s ranking approach has been used in this present work to defuzzify the $FE_i$ and $FITC_j$. If $\left( a^L_\alpha, a^U_\alpha \right)$ represents the $\alpha$-cut for fuzzy (triangular/trapezoidal) communication/execution time, then the defuzzification is performed as:

$$ e_{i,k} = R\left( \tilde{e}_{i,k} \right) = \frac{1}{2} \int_0^1 \left( a^L_\alpha, a^U_\alpha \right) d\alpha $$

$$ c_{i,j} = R\left( \tilde{c}_{i,j} \right) = \frac{1}{2} \int_0^1 \left( a^L_\alpha, a^U_\alpha \right) d\alpha . \tag{1} $$

After defuzzification, crisp values of communication time are stored in $ITC_j M = \begin{bmatrix} c_{i,j} \end{bmatrix}$ matrix and values of execution time are stored in $E_i M = \begin{bmatrix} e_{i,k} \end{bmatrix}$ 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. $TITC_j$ may be prescribed as:

$$ TITC_j = \sum_{i,j=1}^{r} \sum_{k=1}^{m} \left( d_{k,j} \cdot \tilde{c}_{i,j} \right) \cdot x_{i,j} \cdot x_{i,k} , \tag{2} $$

where, the inter-processor distance $d_{k,j}$ represents the communication time per unit of data transferred between processor $p_k$ and processor $p_j$. In scenarios where tasks $t_i$ and $t_j$ are assigned to different processors, the communication time is calculated as $(d_{k,j} \cdot \tilde{c}_{i,j})$, where $d_{k,j} = 0$ 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 (ET)

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

\[
TE_T = \sum_{i=1}^{r} x_{i,k} \cdot \hat{e}_{i,k}. \tag{3}
\]

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:

\[
CS = \sum_{i,j=1}^{r,m} (d_{k,j} \cdot \hat{c}_{i,j}) \cdot x_{j,i} \cdot x_{i,k} + \sum_{i=1}^{r} \sum_{k=1}^{m} x_{i,k} \cdot \hat{e}_{i,k}. \tag{4}
\]

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:

\[
RT = \max_{1 \leq k \leq m} \left\{ \sum_{i,j=1}^{r,m} (d_{k,j} \cdot \hat{c}_{i,j}) \cdot x_{j,i} \cdot x_{i,k} + \sum_{i=1}^{r} \sum_{k=1}^{m} x_{i,k} \cdot \hat{e}_{i,k} \right\}, \tag{5}
\]

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:

\[
RS = \left[ \prod_{k=1}^{m} R_k \right] \cdot \left[ \prod_{k=1}^{m} R_{k,l} \right], \tag{6}
\]

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

$$R_{k,l} = e^{-\mu_{k,l} \sum_{i,j=1}^{m} \sum_{k=1}^{n} (d_{k,j} \cdot c_{i,j}) \cdot x_{j,l} \cdot x_{i,k}}.$$

### 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:

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

where $RT_{\text{proposed}}$ 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 \( E_j M \) and \( IT C_j M \) 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:

\[
\min RT = \max \sum_{i=1}^{r} \sum_{k=1}^{m} (d_{i,k} \cdot \bar{e}_{i,j}) \cdot x_{j,i} \cdot x_{j,k} + \sum_{i=1}^{r} x_{i,k} \cdot \bar{e}_{i,k}.
\]

Such that,

\[
\sum_{j=1}^{m} x_{j,i} = 1 \forall j,
\]

\[
x_{j,i} \in \{0,1\} \forall j, i.
\]

Each and every module must be allotted to precisely single processor, according to constraint (8) and \( x_{j,i} \) 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.

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

(10)

where, $\left(c_{i,j}^\hat{\ }\right)$ represents the element of the matrix $NITC_f M$ 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.

\[
V_k(t+1) = \omega \times V_k(t) + \eta_1 \times \phi_1 \times (P_k(t) - C_d(t)) + \eta_2 \times \phi_2 \times (G(t) - C_d(t))
\]  

(11)

where, \( \omega \) stands for the inertia weight and the values of cognitive constant \( \eta_1 \) and social constant \( \eta_2 \) typically fall between [0, 2]. \( \phi_1, \phi_2 \) 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 \( t+1 \).

\[
C_d(t+1) = C_d(t) + V_k(t+1).
\]

(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:

![Chromosomes encoding](image)

**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 $\epsilon \in (0,1]$, the following is the fitness function for the allocation algorithm GA:
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:

\[ P_{rob}^i = 1 - \left\{ \frac{f_{ret}(\text{chromosome } _i)}{\sum_{i=1}^{m} \left( f_{ret}(\text{chromosome } _i) - f_{ret}(\text{chromosome } _i) + 1 \right) } \right\}. \]  

(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 $C^{(t+1)}$, $C^{(t+1)} = \text{Reverse} \ (C^{(t+1)})$

1: While ($k \leq m$) 
2: { 
3: if ($k=1$) 
4: { 
5: $C^{(t+1)}_1 = B'_1$
6: } 
7: if $(1 < k \leq m)$ 
8: { 
9: $x'_k = \text{image} \ (C^{(t+1)}_{k-1}) \ \text{IN} \ B^{(t)}$
10: $y'_k = \text{image} \ (x'_k) \ \text{IN} \ A^{(t)}$
11: if ($(y'_k \ \text{IN} \ C^{(t+1)})=\text{True}$) 
12: { 
13: $C^{(t+1)}_k = \min (B^{(t)} - C^{(t+1)})$
14: } 
15: else 
16: { 
17: $C^{(t+1)}_k = y'_k$
18: } 
19: } 
20: } 
21: $C^{(t+1)} = \text{Reverse} \ (C^{(t+1)})$

# For offspring $D^{(t+1)}$
22: While ($k \leq m$) do 
23: { 
24: if ($k=1$) 
25: { 
26: $D^{(t+1)}_1 = A'_1$
27: } 
28: if $(1 < k \leq m)$ 
29: { 
30: $w'_k = \text{image} \ (D^{(t+1)}_{k-1}) \ \text{IN} \ A^{(t)}$
31: $v'_k = \text{image} \ (w'_k) \ \text{IN} \ B^{(t)}$
32: if ($(v'_k \ \text{IN} \ D^{(t+1)})=\text{True}$)
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:

**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 \( C^{(r+1)} \) of dimension \( m \)

**Output:** A mutated offspring \( E^{(r+1)} \)

# Create mutated offspring \( E^{(r+1)} \) as follows:

**1:** Select two random cut-points over \( C^{(r+1)} \) say, cut-point1 and cut-point2 respectively
2: $x' = C^{(r+1)}$ (cut-point1 to cut-point2)
3: $x' = \text{Shuffle } (x)$
4: Place $x'$ in $C^{(r+1)}$ between cut-point1 and cut-point2
5: While $(k \leq m)$
6: 
7: if ($k == 1$)
8: 
9: $E^{(r+1)}_1 = \min(x')$
10: 
11: if (1 < $k \leq m$)
12: 
13: $y^{(r+1)} = \text{image } (E^{(r+1)}_{k-1})$ IN $C^{(r+1)}$
14: if ($(y^{(r+1)} \text{ IN } E^{(r+1)}) = \text{True}$)
15: 
16: $E^{(r+1)}_k = \text{image } (y^{(r+1)})$
17: else
18: 
19: $E^{(r+1)}_k = y^{(r+1)}$
20: 
21: 
22: Reshuffle the bits of $E^{(r+1)}_k$ to their original position from Step 2 to get mutated offspring $E^{(r+1)}$

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:

![Figure 4](image)

**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 $ITC_T M = \left[ c_{i,j} \right]$ is renewed from $FITC_T M = \left[ \bar{c}_{i,j} \right]$. 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

$$Cd_k = (c_{d_{k,1}}, c_{d_{k,2}}, \ldots, c_{d_{k,s}}).$$

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, $\omega$ is determined based on the calculation in Eq (15).
\[ \omega = \omega_{\text{max}} + \left( \frac{\omega_{\text{min}} - \text{current iter.}}{\text{max iter.}} \right) \times 0.5 \]  

(15)

where,

\[ \omega_{\text{max}} = \text{Maximum weight} , \]

\[ \omega_{\text{min}} = \text{Minimum weight} , \]

Max iter. = Maximum iteration number.

The inertia weight is set to decrease linearly from a high initial value \( \omega_{\text{max}} = 0.9 \) to a lower final value \( \omega_{\text{min}} = 0.4 \) 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.

\textbf{Algorithm 3:} Formation of task-clusters by employing HPSOGAK.

1: Initialize \( r, m, \omega, \eta_1, \eta_2, \phi_1, \phi_2, FITC_T M \)

2: Defuzzification of \( FITC_T M = \left[ \tilde{c}_{i,j} \right] \) into crisp \( ITC_r M = \left[ c_{i,j} \right] \) through using Robust’s ranking approach

3: \textbf{for} \( 0 \leq i \leq j \leq r \) \textbf{do}

4: \{ 

5: \hspace{1em} \text{NITC}_T M = \left[ \hat{c}_{i,j} \right] = \begin{cases} 
\max_j c_{i,j} - c_{i,j}, & \text{if } i \neq j \\
0, & \text{otherwise}
\end{cases}

\}

6: \textbf{end for}

7: Generate ‘m’ number of task-clusters of ‘r’ tasks randomly named as \( \{cl_1, cl_2, \ldots cl_m \} \)

8: \textbf{for} \( k^{th} \text{ cluster } (1 \leq k \leq m) \) \textbf{do}

9: \{ 

10: \hspace{1em} \text{Calculate the initial } Cd_k \text{ for PSO-GA as follows:}

11: \hspace{2em} Cd_k = \frac{1}{n_k} \sum_{i,j=1}^r c_{i,j} \cdot u_{i,k}

12: \}

13: \textbf{end for}

14: \text{t=0}

15: \textbf{for} \( 1 \leq k \leq m \) \textbf{do}

16: \{ 

17: \hspace{1em} \text{Initialize } V_k \left( t \right) \text{ and } Cd_k \left( t \right)

18: \hspace{1em} P_k \left( t \right) = Cd_k \left( t \right)

19: \}

\textit{Mathematics in Engineering}  
Volume 6, Issue 4, 559–606.
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 \(FE_pM\) must be added and represented as a new row in the \(NFE_pM\) 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, \( E_T M = \tilde{e}_{i,k} \) is updated into \( NE_T M = e^\hat{b}_{i,k} \). 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 \( FE_T M = \tilde{e}_{i,k} \)

2: Defuzzification of \( FE_T M = \tilde{e}_{i,k} \) into crisp \( E_T M = e_{i,k} \) through using Robust’s ranking approach

3: Modify \( E_T M = e_{i,k} \) into \( NE_T M = e^\hat{b}_{i,k} \) 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:

\[
f_{fet} = 1/ \left( e + \sum_{i=1}^{m} \sum_{k=1}^{m} x_{i,k} \cdot \tilde{e}_{i,k} \right)
\]

11: Assess the probability of each chromosome in the population through using following formula:

\[
P^i_{rob} = 1 - \left( \frac{f_{fet}(chromosome \_i)}{\sum_{i=1}^{m}(f_{fet}(chromosome \_i)) - f_{fet}(chromosome \_i) + 1} \right)
\]

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.
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 ($\omega$) 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 $\omega$-equation (Eq (15)) for PSO. Where the value of $\omega$ 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.

<table>
<thead>
<tr>
<th>System parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inertia weight ( \omega )</td>
<td>0.4–0.9</td>
</tr>
<tr>
<td>( \phi_1 ) and ( \phi_2 )</td>
<td>[0,1]</td>
</tr>
<tr>
<td>( \eta_1 ) and ( \eta_2 )</td>
<td>1.49</td>
</tr>
<tr>
<td>Failure rate of processor, ( \lambda_k )</td>
<td>[0.0002–0.00012]</td>
</tr>
<tr>
<td>Failure rate of ( k ) communicating path, ( \mu_{k,j} )</td>
<td>[0.0005–0.00015]</td>
</tr>
<tr>
<td>Maximum iteration of GA</td>
<td>50</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Maximum iteration of PSO</td>
<td>100</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Example</th>
<th>Reference</th>
<th>Used Technique</th>
<th>Tasks Assignment</th>
<th>RS</th>
<th>CS</th>
<th>RT</th>
<th>Resource Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Djigal et al. [63]</td>
<td>List based scheduling algorithm</td>
<td>{t_1, t_2, t_3, t_7} → p_1 {t_4, t_6, t_8, t_{10}} → p_2 {t_5, t_6} → p_3</td>
<td>-</td>
<td>304</td>
<td>181</td>
<td>75.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed model</td>
<td>Hybrid PSO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Kumar et al. [64]</td>
<td>Heuristic approach</td>
<td>{t_1, t_4} → p_1 {t_5} → p_2 {t_2, t_3} → p_3</td>
<td>0.97096</td>
<td>(145, 240, 360)</td>
<td>(110, 175, 250)</td>
<td>68.06</td>
</tr>
</tbody>
</table>

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

**Example 1.** Here, the problem is grabbed from Djigal et al. [63] model. In which DRTS consist ten tasks \(\{t_1, t_2, \ldots, t_{10}\}\) 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.](image-url)
The following chromosome in Figure 8 gives the final task-clusters as a result of carrying out the steps of Algorithm 3:

\[
\begin{align*}
& t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \\
& cl_1, cl_2, cl_3, cl_1, cl_3, cl_3, cl_3, cl_3, cl_1, cl_1
\end{align*}
\]

\textbf{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:

\[
\begin{align*}
& cl_1, cl_2, cl_3 \\
& p_1, p_3, p_2
\end{align*}
\]

\textbf{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:

\[
\begin{align*}
& cl_1 = \{t_8, t_{10}\} \rightarrow p_1, \quad cl_2 = \{t_2, t_5, t_9\} \rightarrow p_3 \quad \text{and} \quad cl_3 = \{t_1, t_3, t_4, t_6, t_7\} \rightarrow p_2.
\end{align*}
\]
Task-clusters assignments are shown in Figure 10.

![Figure 10. Solution graph of Example 1.](image)

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 $\tilde{e}_{i,k}$ and $\tilde{c}_{i,j}$ have been assessed as fuzzy triangular number shown in Figures 11 and 12 respectively.

![Figure 11. Fuzzy execution time ($\tilde{e}_{i,k}$) on processors.](image)
To resolve the aforementioned example, first use Robust’s ranking technique to defuzzify the $\tilde{e}_{i,k}$ and $\tilde{c}_{i,j}$ times into crisp times $e_{i,k}$ and $c_{i,j}$ respectively, and then generate $E_T M$ and $ITC_T M$ 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:

\[
\begin{array}{ccccc}
  t_1 & t_2 & t_3 & t_4 & t_5 \\
  cl_1 & cl_2 & cl_2 & cl_3 & cl_2 \\
\end{array}
\]

**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:

\[
\begin{array}{ccc}
  cl_1 & cl_2 & cl_3 \\
  p_2 & p_1 & p_3 \\
\end{array}
\]

**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:

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

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

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

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

Task-clusters assignments are shown in Figure 20.
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 $\{t_1, t_2, \ldots, t_9\}$ that have to be allocated on five processors $\{p_1, p_2, \ldots, p_5\}$. In this problem $\tilde{e}_{i,k}$ and $\tilde{c}_{i,j}$ have been assessed as fuzzy triangular number shown in the matrices $FE_T M$ and $FITC_T M$ respectively.

$$FE_T M = \left[ \tilde{e}_{i,k} \right]_{9 \times 5} = t_5$$

$$\begin{bmatrix}
   t_1 & (7,9,11) & (8,13,15) & (12,14,15) & (5,8,10) & (11,12,14) \\
   t_2 & (1,3,4) & (2,7,10) & (9,11,12) & (6,8,9) & (12,14,16) \\
   t_3 & (10,12,14) & (6,8,10) & (3,7,10) & (3,5,7) & (4,6,10) \\
   t_4 & (3,7,10) & (4,5,7) & (11,14,17) & (8,10,12) & (7,9,10) \\
   t_5 & (10,12,13) & (10,13,15) & (6,8,9) & (6,7,9) & (4,6,8) \\
   t_6 & (1,3,5) & (4,5,6) & (10,11,12) & (6,8,10) & (7,9,10) \\
   t_7 & (10,14,15) & (10,12,13) & (8,10,11) & (6,9,10) & (8,13,15) \\
   t_8 & (5,7,9) & (6,8,10) & (8,10,12) & (9,11,13) & (12,14,16) \\
   t_9 & (9,11,13) & (6,10,14) & (5,8,11) & (5,7,9) & (10,13,16)
\end{bmatrix}$$
To resolve the aforementioned example, first use Robust’s ranking technique to defuzzify the $\tilde{e}_{i,k}$ and $\tilde{c}_{i,j}$ 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:

$$FITC_tM = \begin{bmatrix} c_{i,j} \end{bmatrix}_{i,j=1} = \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \\ t_1 & (0,0,0) & (1,2,4) & (2,3,5) & (6,7,8) & (4,5,7) & (4,6,7) & (0,0,0) & (5,6,9) & (3,4,7) \\ t_2 & (1,2,4) & (0,0,0) & (3,5,6) & (0,0,0) & (6,8,9) & (1,3,4) & (1,2,3) & (3,4,5) & (2,3,4) \\ t_3 & (2,3,5) & (3,5,6) & (0,0,0) & (1,3,5) & (2,4,6) & (1,3,5) & (0,2,4) & (3,5,7) & (6,7,8) \\ t_4 & (6,7,8) & (0,0,0) & (1,3,5) & (0,0,0) & (1,3,4) & (1,2,3) & (4,6,7) & (1,3,7) & (6,9,10) \\ t_5 & (4,5,7) & (6,8,9) & (2,4,6) & (1,3,4) & (0,0,0) & (1,5,6) & (4,7,8) & (0,0,0) & (2,3,4) \\ t_6 & (4,6,7) & (1,3,4) & (1,3,5) & (1,2,3) & (1,5,6) & (0,0,0) & (8,9,10) & (5,7,9) & (3,4,5) \\ t_7 & (0,0,0) & (1,2,3) & (0,2,4) & (4,6,7) & (4,7,8) & (8,9,10) & (0,0,0) & (6,7,8) & (4,5,7) \\ t_8 & (5,6,9) & (3,4,5) & (3,5,7) & (1,3,7) & (0,0,0) & (5,7,9) & (6,7,8) & (0,0,0) & (2,3,4) \\ t_9 & (3,4,7) & (2,3,4) & (6,7,8) & (6,9,10) & (2,3,4) & (3,4,5) & (4,5,7) & (2,3,4) & (0,0,0) \\ \end{bmatrix}$$

**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:

$$\{cl_1 \rightarrow p_5, \ cl_2 = \{t_4, t_9\} \rightarrow p_2, \ cl_3 = \{t_1, t_8\} \rightarrow p_1, \ cl_4 \rightarrow p_3, \ cl_5 \rightarrow p_4 \}$$

**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:

$$cl_1 = \{t_3 \rightarrow p_5, \ cl_2 = \{t_4, t_9\} \rightarrow p_2, \ cl_3 = \{t_1, t_8\} \rightarrow p_1, \ cl_4 \rightarrow p_3 \ and \ cl_5 = \{t_6, t_7\} \rightarrow p_4 \}$$

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.

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

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.

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

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.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Algorithms</th>
<th>Tasks</th>
<th>Processors</th>
<th>RS</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Topcuoglu et al. [68]</td>
<td>({t_3, t_7})</td>
<td>(p_1)</td>
<td>0.9614</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_4, t_6, t_8})</td>
<td>(p_3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_1, t_2, t_5, t_9, t_{10}})</td>
<td>(p_2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Kumar et al. [19]</td>
<td>({t_3, t_7, t_{10}})</td>
<td>(p_1)</td>
<td>0.9629</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_4, t_8, t_9})</td>
<td>(p_2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_1, t_2, t_5, t_6})</td>
<td>(p_3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Chauhan et al. [43]</td>
<td>({t_3, t_7, t_{10}})</td>
<td>(p_1)</td>
<td>0.9629</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_4, t_8, t_9})</td>
<td>(p_2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_1, t_2, t_5, t_6})</td>
<td>(p_3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ilavarasan et al. [66]</td>
<td>({t_3, t_7, t_8})</td>
<td>(p_1)</td>
<td>-</td>
<td>236</td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_1, t_2, t_6})</td>
<td>(p_3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>({t_4, t_5, t_9, t_{10}})</td>
<td>(p_2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
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.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Tasks &amp; Processors</th>
<th>GA RT</th>
<th>GA Efficiency</th>
<th>GA Utilization (%)</th>
<th>PSO-GA RT</th>
<th>PSO-GA Efficiency</th>
<th>PSO-GA Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(9,3)</td>
<td>422</td>
<td>0.7867</td>
<td>81.37</td>
<td>415</td>
<td>0.8001</td>
<td>83.69</td>
</tr>
<tr>
<td>2</td>
<td>(6,4)</td>
<td>21</td>
<td>0.7976</td>
<td>78.66</td>
<td>21</td>
<td>0.7976</td>
<td>78.66</td>
</tr>
<tr>
<td>3</td>
<td>(4,3)</td>
<td>65</td>
<td>0.5246</td>
<td>-</td>
<td>30</td>
<td>0.6105</td>
<td>67.31</td>
</tr>
<tr>
<td>4</td>
<td>(6,4)</td>
<td>34</td>
<td>0.6126</td>
<td>52.94</td>
<td>21</td>
<td>0.7976</td>
<td>78.66</td>
</tr>
<tr>
<td>5</td>
<td>(9,3)</td>
<td>120</td>
<td>0.5629</td>
<td>-</td>
<td>118</td>
<td>0.5801</td>
<td>61.13</td>
</tr>
<tr>
<td>6</td>
<td>(8,4)</td>
<td>37</td>
<td>0.7901</td>
<td>-</td>
<td>37</td>
<td>0.7901</td>
<td>78.38</td>
</tr>
<tr>
<td>7</td>
<td>(9,3)</td>
<td>118.2</td>
<td>0.6183</td>
<td>-</td>
<td>118.2</td>
<td>0.6183</td>
<td>69.23</td>
</tr>
<tr>
<td>8</td>
<td>(4,3)</td>
<td>19</td>
<td>0.6140</td>
<td>63.25</td>
<td>19</td>
<td>0.6140</td>
<td>63.25</td>
</tr>
<tr>
<td>9</td>
<td>(5,2)</td>
<td>86</td>
<td>0.3139</td>
<td>-</td>
<td>18</td>
<td>0.7502</td>
<td>79.33</td>
</tr>
<tr>
<td>10</td>
<td>(10,4)</td>
<td>85</td>
<td>0.5973</td>
<td>-</td>
<td>77</td>
<td>0.6421</td>
<td>71.77</td>
</tr>
<tr>
<td>11</td>
<td>(7,3)</td>
<td>195</td>
<td>0.4708</td>
<td>-</td>
<td>185</td>
<td>0.5954</td>
<td>63.13</td>
</tr>
<tr>
<td>12</td>
<td>(10,4)</td>
<td>65</td>
<td>0.7803</td>
<td>-</td>
<td>58</td>
<td>0.7992</td>
<td>81.02</td>
</tr>
<tr>
<td>13</td>
<td>(10,3)</td>
<td>130</td>
<td>0.5385</td>
<td>55.38</td>
<td>128</td>
<td>0.5891</td>
<td>78.41</td>
</tr>
<tr>
<td>14</td>
<td>(8,3)</td>
<td>78</td>
<td>0.7629</td>
<td>77.38</td>
<td>78</td>
<td>0.7629</td>
<td>77.38</td>
</tr>
<tr>
<td>15</td>
<td>(5,3)</td>
<td>177.5</td>
<td>0.7102</td>
<td>-</td>
<td>166.25</td>
<td>0.7732</td>
<td>81.79</td>
</tr>
</tbody>
</table>
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.

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

Figure 26. Graphical presentation of Table 9 in terms of efficiency.
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 $\chi^2$ 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


