Shahid Bahonar University of Kerman
Journal of Mahani Mathematical Research
Print ISSN: 2251-7952 Online ISSN: 2645-4505

EXTENDED TABU SEARCH-BASED SCHEDULING TO IMPROVE PROFITABILITY IN HETEROGENEOUS PARALLEL SYSTEMS

S. Bakhoda, M. Abdollahi Azgomi, and M.R. Ebrahimi Dishabi ✉

Article type: Research Article
(Received: 26 May 2023, Received in revised form 28 August 2023)
(Accepted: 13 October 2023, Published Online: 25 October 2023)

Abstract. Higher utilization of existing resources and facilities in order to increase efficiency and profitability is always one of the basic challenges for parallel processing systems and environments, and this challenge becomes more complicated when the system resources are heterogeneous. One way to achieve high efficiency and profitability of heterogeneous parallel systems is to schedule tasks optimally. In this paper, an extended tabu search-based scheduling algorithm (ESTS) is presented to improve the profitability of heterogeneous parallel systems, which can achieve suitable solutions in a short computational time. To evaluate the efficiency of the proposed solution, due to the lack of a suitable criterion to evaluate this problem, the obtained results are compared with both the results of an extended scheduling based on a genetic algorithm (ESGA) with a large number of chromosomes and a high number of generations, as well as an extended scheduling based on a simulated annealing algorithm (ESSA) with a linear temperature reduction. The benchmark files of different sizes were tested under the same conditions, and the comparison of results shows the superiority of the proposed solution in terms of profitability and computational time.

Keywords: Heterogeneous parallel systems, profitability, allocation and scheduling, tabu search, computational time.
2020 MSC: 91B32, 91B69.

1. Introduction

Parallel processing refers to the simultaneous execution of multiple tasks on multiple processing units, aiming to increase efficiency and speed up the task to achieve the desired result. One of the main goals for parallel systems is to increase the system's efficiency and profitability. Such a model has various practical applications. For instance, a set of tasks might represent a set of orders that might result in a certain profit for an organization. Due to the limitation of the resources, it should be decided whether to accept a specific order or reject it; how and when to perform tasks should be also determined. Delay

✉ mrebrahimy@m-iau.ac.ir, ORCID: 0000-0003-1963-9560
DOI: 10.22103/jmmr.2023.21570.1446
Publisher: Shahid Bahonar University of Kerman
How to cite: S. Bakhoda, M. Abdollahi Azgomi, M.R. Ebrahimi Dishabi, *Extended tabu search-based scheduling to improve profitability in heterogeneous parallel systems*, J. Mahani Math. Res. 2024; 13(1): 535 - 562.

in completing the orders results in penalties that reduce the total profit of the realized orders. Thus, maximizing the profit depends on minimizing the delay in completing the orders [17]. Due to limitations and scarcity of resources, the available resources should be managed well. One way to improve the efficiency and profitability of parallel systems is optimal task scheduling. Task scheduling is one of the most important challenges in achieving high efficiency in parallel processing environments. The purpose of task scheduling is to allocate tasks to free resources such that maximum parallelization while processing is realized. Task scheduling refers to determining the starting and ending times of a set of tasks regarding certain constraints. The constraints are either associated with time or resources [2]. There are various types of distributed scheduling, including static, dynamic, centralized, decentralized, preventive, and non-preventive scheduling [11].

In some real parallel systems, the processing resources are different from each other in some ways. This research has paid attention to this issue. Proper use of the system's heterogeneous resource capacity is necessary to achieve high efficiency and profitability, and an efficient allocation and scheduling process is very effective in achieving goals. The scheduling problem is associated with a wide range of optimization problems that have recently attracted attention [7]. Studies show that scheduling problems are very diverse and have their own conditions, characteristics, and limitations. Studies related to scheduling problems have mainly focused on homogeneous and identical resources, aiming at minimizing the total time or sum of time required to complete all tasks (makespan), reducing the delay in completing tasks, reducing job rejection, reducing the computational and execution time, etc. In this study, according to real-world systems conditions, the issue of processing resource heterogeneity in parallel systems has been addressed in terms of both processing speed heterogeneity and processing cost heterogeneity, which are focused simultaneously in quantitative studies. Also, in the literature, the issue of profitability in heterogeneous parallel systems has not been discussed much. Furthermore, in this study, some conditions and challenges that usually exist in real systems have been considered, such as the time limit, delay penalty, etc. A time limit has been set for completing tasks. In case of delay in completion, the tasks are not rejected, but they are penalized proportional to the amount of delay.

The investigated system can be generally considered for some parallel systems in which the processing resources are various in terms of processing speed and processing cost. This paper tries to propose an appropriate scheduling approach with low computational time for the optimal use of heterogeneous resources and, consequently, increase the efficiency and profitability of the system. Previous studies have presented various heuristic and metaheuristic approaches to solve scheduling problems and achieve the goals [24]. Considering the studies, one of the efficient approaches used by researchers to solve various problems is the tabu search (TS) [2,4,6–8,10,13,15,18,20,23,27–29,33]. A review of recent

studies shows that tabu search is a successful and fast technique for solving various scheduling problems. For example, the scheduling problems investigated in the new references [20] (2021), [28] (2020), [18] (2022), [10] (2022), [29] (2022), [15] (2021), etc. have been solved using different approaches based on tabu search. However, none of them are similar to the problem investigated in this study and have different conditions. In addition, studies show that genetic algorithm (GA) and simulated annealing (SA) are other appropriate approaches for solving various scheduling problems [3, 12, 31, 34].
This paper presents an extended tabu search-based scheduling algorithm (ESTS) in a heterogeneous parallel system with specific conditions. To this end, first, a vector approach for allocating and scheduling input tasks on heterogeneous resources is presented. Then, a tabu search-based strategy is used to improve the vector allocation approach and to achieve better results for the objective functions. In the proposed model, some efficient mechanisms used to mutate chromosomes in the genetic algorithm are used to generate mutated and better neighborhood solutions and, consequently, faster convergence to the good solutions. The proposed solution is tested on several suitable benchmark files of different sizes in terms of the number of tasks and resources. In order to evaluate the efficiency of the proposed algorithm, due to the existence of various conditions in this problem and the lack of seeing a problem with completely identical conditions in the literature, the proposed algorithm is compared with an extended scheduling based on a genetic algorithm (ESGA) with a large number of chromosomes and a high number of generations, so that an estimate of the closeness of this answer to the optimal answer can be obtained according to the nature of the genetic algorithm in searching the whole problem space. Furthermore, the proposed solution is compared with an extended scheduling based on a simulated annealing algorithm (ESSA) with a specified initial and final temperature and a linear temperature reduction. All algorithms were tested with the same benchmarks and under the same software and hardware conditions. In the end, the results of the algorithms are evaluated in terms of profitability and computational time.
In the rest of the paper, Section 2 presents previous studies in this context. Section 3 describes and models the problem conditions. Section 4 describes and formulates the proposed algorithm. Section 5 tests the proposed algorithm on the benchmark files. Section 6 compares and evaluates the experiment's results. Finally, the paper is concluded in section 7.

2. Literature review

One of the effective ways to increase the efficiency and utilization of parallel systems, both homogeneous and heterogeneous, is the optimal allocation and scheduling of tasks on available resources. The allocation and scheduling should be such that the maximum use of resources is made, and the time to perform tasks is minimized. Most of the scheduling problems are classified as

non-deterministic polynomial-time (NP-hard) problems. To solve such complex problems, most of the studies focus on the use of artificial intelligence techniques, heuristics, and metaheuristic techniques such as fuzzy logic, neural networks, genetic algorithms, particle swarm optimization (PSO), simulated annealing, etc [21]. To solve such problems, many efforts and research have been done by researchers, some of which are summarized in Table 1.
Researchers in the paper [19] have proposed a genetic algorithm (GA) for the task scheduling problem that considers both job and data parallelization. Evaluations showed that the proposed algorithm finds the optimal scheduling in a short execution time. In [22], researchers have divided the problem into two separate subsets. First, all tasks are allocated to the processors, and then the order of the assigned tasks on the processors is determined to form a complete schedule (AO: allocating and ordering). Evaluations showed that the proposed model increased the number of task graphs scheduled in a possible time frame and reduced the processor idle time. The scheduling of high-performance computing (HPC) applications has been examined in [26]. Two scheduling models, including list and pack, were presented. The simulations showed that pack scheduling provides a promising and workable solution. To solve the order acceptance and scheduling (OAS) problem on identical parallel machines with sequence-dependent setup times (SDST), a water-flow-like algorithm (WFA) was developed. The results showed that the proposed model is more efficient than other models, particularly for mid-to-large-sized problems [32]. Authors of [30] have investigated OAS on unrelated parallel machines to maximize the total net revenue of the accepted orders. A formulation-based branch-and-bound (B& B) is developed to handle complicated instances following the principle of divide and conquer. Article [16] proposes a reservation-based dynamic scheduling for deadline-constrained mouldable jobs. The goal is to maximize HPC as a service (HPCaaS) provider's profit. The results showed the efficiency of their approach. For efficient scheduling with optimum resource utilization and energy consumption, a multi-objective adaptive manta-ray foraging optimization (MAMFO) has been proposed in [25]. To address task scheduling and load balancing in Cloud-fog-edge collaboration among servers, an improved version of the min-min algorithm for workflow scheduling has been proposed, which considers cost, makespan, energy, and load balancing in heterogeneous environments [5].

Table 1. Summary of research

Authors	Year	Problem type	Proposed model	Results
Juraszek et al.	2008	Selecting and scheduling on identical parallel machines	A SA algorithm	Maximizing the total profit compared with the B&B and list scheduling
Bozejko et al.	2017	Cycle job shop scheduling problems (JSSP)	A parallel TS	Minimizing cycle time
Liu Y et al.	2019	Scheduling of multiple data-parallel tasks on multicores	A GA algorithm	Finding the optimal scheduling in a short execution time

Alazzam et al.	2019	Job scheduling in a cloud computing environment	A hybrid algorithm based on tabu and harmony search algorithms	Minimizing the makespan and cost compared to TS, harmony search, and round-robin
Zhang et al.	2018	Flexible job shop scheduling problems (FJSSP)	Variable neighborhood search (VNS) based on a GA	Minimizing the makespan
Toshev et al.	2019	Flexible job shop scheduling problems	A hybrid PSO and TS algorithm	The good performance of the proposed algorithm compared to the reference sources and a GA (the mean error is 0.044% and the run time is very low)
Orr & Sinnen	2019	Task scheduling with communication delays	A new state-space model (AO)(AO: allocating and ordering)	Increasing the number of task graphs scheduled in a possible time frame and reducing the processor idle time
Sun et al.	2018	Scheduling high-performance computing (HPC) applications	List scheduling and pack scheduling models	Pack scheduling provides a promising and workable solution
Wu et al.	2017	Order acceptance and scheduling (OAS) problem on identical parallel machines	Development of a water-flow-like(WFA) algorithm (WFA.I and WFA.II)	WFA.II is more efficient than other models, particularly for mid-to-large-sized problems
Wang & Ye	2019	OAS on unrelated parallel machines	A formulation-based branch-and-bound (B&B)	Maximizing the total net revenue of the accepted orders
Chandran & Kumar	2020	Optimal energy-aware allocation of data center resources	Tabu job master (JM)	The makespan for tabu JM is better than TS, GA, and ABC
Bozejko et al.	2017	Optimal task allocation and scheduling for computing clusters	The two-level algorithm (TS is used to minimize the relatively low accuracy of the greedy packing strategy)	The proposed algorithm improves results compared to the greedy strategy

Ben Abdellafou et al.	2019	Scheduling problem on parallel machines with non-availability constraints	A metaheuristic TS	Minimizing the makespan and computing time compared to the proposed lower bound and the best heuristic
Dai et al.	2018	The multi-skill resource-constrained project scheduling problem	An improved TS (ITS algorithm)	ITS algorithm is a powerful solution methodology in terms of solution quality
Mathlouthi et al.	2021	Technician routing and scheduling problem (TRSP)	A TS augmented with an adaptive memory	The proposed algorithm can solve instances with up to 200 tasks
Romero et al.	2018	Flexible job shop scheduling problems	A mathematical programming solution, and a TS algorithm	TS algorithm obtained good quality solutions in lower computational times
Vela et al.	2020	JSSP with fuzzy sets modeling uncertain durations and flexible due dates	An evolutionary tabu search method (EATS)	The good behavior of the EATS. EATS performed favorably compared to other approaches
Alkhateeb & Abedalguni	2019	Optimizing problems (discrete and continuous)	A hybrid algorithm using the cuckoo search (CS) and SA	The potential of the proposal in terms of best solutions and computational time
Zorin & Kostenko	2014	Problems of multiprocessor scheduling	A developed SA algorithm (for determination of the minimal necessary number of processors, etc.)	The developed algorithm was substantiated both theoretically and experimentally
Cruz-Chávez et al.	2017	Flexible job shop scheduling problems	A SA algorithm accelerated by a partial scheduling mechanism and a cooling schedule mechanism	Facilitating a rapid approach to good solutions for FJSSP
Wei et al.	2018	The flow shop scheduling problem	A hybrid GA-SA algorithm (HGSA)	Minimizing the makespan compared with five state-of-the-art algorithms

Krim et al.	2022	Parallel-machine scheduling problem	Development of a mixed integer linear program model and two generalizable TS	Minimizing the job rejection costs and the weighted sum of completion times
Chen et al.	2022	Unrelated parallel machine scheduling	A mixed-integer programming (MIP) model and a hybrid TS	The TS outperforms the MIP model in terms of solution quality and makespan
Umam et al.	2022	The flow shop scheduling problem	A hybrid GA-TS algorithm	Minimizing the makespan
Hajibabaei et al.	2021	FJSSP with unrelated parallel machines	A linear MIP model and a TS algorithm for solving large-size instances	The TS obtained better solutions compared to the GA with less runtime
Momeni korbekandi et al.	2023	FJSSP for single-machine and multi-machine job shops	A novel meta-heuristic hybrid parthenogenetic algorithm (NMHPGA)	NMHPGA achieves better objective functions with faster convergence speed
Singh et al.	2022	Workflow scheduling with selected virtual machines	A multi-objective adaptive manta-ray foraging optimization (MAMFO)	MAMFO improved the work efficiency
Bisht & Vampugani	2022	Workflow scheduling for heterogeneous resources	An improved version of the min-min algorithm	Minimize the makespan, less energy consumption along with load balancing, and marginally less cost compared to min-min and ELBMM algorithms
Chawra & Gupta	2022	Optimization of the wake-up scheduling for 3D-wireless sensor networks	A hybrid of memetic and tabu search algorithms	Better coverage ratio and derivation of the optimal wake-up schedule over the existing schemes

To solve some similar problems, the tabu search (TS) algorithm has been used and has yielded effective and acceptable results. The paper [27] presents a hybrid metaheuristic algorithm, including PSO and TS. The novel algorithm is designed to solve flexible job shop scheduling problems (FJSSP). The results illustrate the good performance of the proposed algorithm. Article [7] presents a parallel TS for the cycle JSSP. The experiments confirmed the research results. In [8], energy-aware scheduling was incorporated and optimized using the proposed tabu job master (JM) and benchmarked by TS, GA, and artificial bee colony (ABC). The final results showed that the makespan for tabu JM is better. Article [2] proposes a hybrid job scheduling algorithm based on tabu and harmony search algorithms. The results showed that the hybrid algorithm has the best results in terms of makespan and cost. The paper [6] presents a solution to the problem of optimal task allocation and scheduling for computing clusters with multiple nodes. The two-level algorithm is presented where TS is used to minimize the relatively low accuracy of the greedy packing strategy. The proposed algorithm improved results compared to the greedy strategy. The scheduling problem on parallel machines subject to non-availability constraints with precedence constraints between the tasks is treated in [4]. A meta-heuristic TS was proposed, and the results were compared with a proposed lower bound and the best heuristic. The results showed that the last proposed version of tabu search is better in terms of makespan and computing time. The paper [9] presents a hybrid metaheuristic-based wake-up scheduling scheme (Memetic-Tabu-based-WS) where the best feature of the memetic algorithm and tabu search algorithm are combined. The results validate the superiority of the proposed scheme over the existing schemes with a better coverage ratio and derivation of the optimal wake-up schedule.

Article [13] considers the multi-skill resource-constrained project scheduling problem. Four neighborhood structures and two mutation operators based on problem characteristics are proposed to form an improved TS (ITS). A bicriterion scheduling problem on two different parallel machines with a periodic preventive maintenance policy is considered in [18]. A new problem relevant to practice, the development of a mixed integer linear program model, and two generalizable TS metaheuristics based on different neighborhood structures and solution spaces are presented. Article [10] proposes a mixed-integer programming (MIP) model and adopts a hybrid TS to achieve approximate practical solutions. The results showed that the TS outperforms the MIP model in terms of solution quality and makespan. Article [29] developed a hybrid GA-TS algorithm and successfully addressed makespan minimization in the flow shop. A linear MIP model is developed for the FJSSP in [15]. For solving large-size instances, a TS algorithm is developed. Article [20] presents a metaheuristic approach for a technician routing and scheduling problem (TRSP). This approach is based on a TS augmented with an adaptive memory, where the evaluation of each solution in the memory is driven by its cost and contribution to diversity. The FJSSP aimed to minimize the makespan is considered in [33]. Variable neighborhood search (VNS) based on a GA is proposed to increase the searchability and balance the intensity and diversity. A JSSP including machine operation flexibility and job splitting into sub-lots is considered in [23]. First,

an integer programming model for the resulting FJSSP with lot streaming is developed. Then, two solutions were tested. A mathematical programming solution that is not able to determine optimal solutions, and a TS algorithm that obtained good quality solutions in lower computational times. Article [28] deals with JSSP with fuzzy sets modeling uncertain durations and flexible due dates. To maximize due-date satisfaction under uncertainty first has been given a new measure of overall due-date satisfaction. Then, a neighborhood structure for local search is defined, and a neighbor-estimation procedure is provided. Furthermore, the TS procedure using the neighborhood is combined with a GA.
Simulated annealing (SA) has proven its success as a single-state optimization search algorithm for both discrete and continuous problems [3]. Researchers examined selecting and scheduling a set of jobs on a set of identical parallel machines simultaneously to maximize the total profit [17]. This problem was solved using the SA algorithm, and its efficiency was compared with the B& B and list scheduling. Article [34] proposes an SA algorithm to determine the minimum necessary number of processors and construct the static schedule for execution of the applied programs with allowance for the constraints on the time of schedule execution and reliability requirements. Article [12] presents an SA algorithm accelerated by a partial scheduling mechanism and a cooling schedule mechanism that is a function of the standard deviation. This facilitates a rapid approach to good solutions for the FJSSP. A hybrid algorithm using the cuckoo search (CS) and SA is provided in [3]. The main goal is to improve the solutions generated by CS using SA to explore the search space efficiently. The results showed the potential of the proposal in terms of best solutions and running time. Article [31] designs a hybrid GA-SA algorithm (HGSA) based on the hormone regulation mechanism for the flow shop scheduling problem. The results verified the effectiveness of the HGSA.
Studies show that scheduling problems are one of the basic challenges for many environments and systems; therefore, they are very diverse. Researchers have used a variety of heuristic and meta-heuristic approaches to solve various scheduling problems. Some of them such as GA and TS have been very successful and efficient. Therefore, according to the successful approach of TS, an extended tabu search-based scheduling algorithm is proposed to solve the studied problem.

3. Problem description and mathematical modeling

Parallel processing is one of the best ways to make optimal use of system resources and perform tasks quickly. There are many real environments and systems in which the processing resources are not the same and differ from each other in various aspects. Therefore, proper use and management of heterogeneous resources is essential to increase the operational capacity and productivity of the system. This study generally considers some parallel systems in which the processing resources are various in some respects. The problem focuses on the appropriate use of heterogeneous resources to improve the efficiency and profitability of the systems. Optimal allocating and scheduling of tasks on non-identical resources is one of the basic ways to achieve these goals. The problem examines a heterogeneous parallel system which has specific conditions and constraints. System resources vary in terms of processing speed and cost. Input tasks are independent and have specific profits and time limits. It is necessary to pay attention to the due date of the tasks, because the delay in completing the tasks reduces its profit and, as a result, reduces the total profit of the system. The aim is to provide an effective approach with a low computational time for assigning and scheduling tasks on heterogeneous resources so that it can ultimately improve the system's profitability. The conditions and characteristics of the problem and its modeling are described below.

Table 2. The symbols and parameters used to formulate the problem

Symbols	Description
T	A set contains input tasks
K	The number of input tasks
t_i	Each of the input tasks into the system
$t_{i \cdot et}$	The time when the task t_i enters the system
$t_{i \cdot pt}$	The expected processing time for the task t_i
$t_{i \cdot st}$	The expected preparation or setup time for the task t_i
$t_{i \cdot dt}$	The deadline or time limit for the task $t_i (t_{i \cdot dt} > t_{i \cdot et})$
$t_{i \cdot ft}$	The processing finish time for the task t_i
$t_{i \cdot r}$	The amount of revenue from processing the task t_i
$t_{i \cdot d}$	The amount of delay or tardiness in completing the task t_i
$t_{i \cdot dp}$	The amount of penalty for delay in completing the task t_i
$t_{i \cdot c}$	The amount of cost for processing and completing the task t_i
p_{dp}	Percentage of penalty for delaying the completion of tasks on their profit per unit of time
T_{nd}	A set of input tasks completed before the deadline $(t_{i \cdot ft} \leq t_{i \cdot dt})$ $(T_{nd} \subseteq T)$
T_d	A set of input tasks completed after the deadline $(t_{i \cdot ft} > t_{i \cdot dt}))$ $(T_d \subseteq T)$
N	A set contains available nodes
p	The number of available nodes
n_j	Each of the available nodes in the system to processes tasks
$n_{j \cdot s}$	The processing speed rate for the node n_j
$n_{j \cdot c}$	The processing cost rate for the node n_j
$PT_{t_i n_j}$	The processing time of task t_i on the node n_j
$PC_{t_i n_j}$	The task t_i processing cost of task t_i on the node n_j
$ST_{t_i n_j}$	The processing start time of task t_i on the node n_j
$FT_{t_i n_j}$	The processing finish time of task t_i on the node n_j
$prf_{t_i n_j}$	The amount of profit earned from processing the task t_i on the node n_j
prf	The amount of total profit earned from processing input tasks in the system

The set of input tasks and the set of system resources are represented by Eq.(1) and Eq.(2), respectively. The symbols used to formulate the problem are given in Table 2. The processing speed of resources is different, so the processing time ($PT_{t_i n_j}$) of a specific task t_i on the node n_j is calculated by multiplying the task t_i expected processing time ($t_{i.pt}$) and the node n_j processing speed rate ($n_{j.s}$) Eq.(3). It is obvious that the processing time of the task t_i on different nodes n_j and $n_{j'}$ with different speeds, would be different Eq.(4). Processing resources are also different in terms of processing cost rate. If the processing cost rate of the node n_j for processing at each time unit is considered to be $n_{j.c}$, the task t_i processing cost on the node n_j ($PC_{t_i n_j}$) is calculated by multiplying the task t_i processing time on the node n_j ($PT_{t_i n_j}$) and the node n_j processing cost rate ($n_{j.c}$) Eq.(5). As a result, if a specific task t_i is to be executed at different processing nodes n_j and n'_j with different processing costs rate, it will impose different costs to the system Eq.(6).

(1) $T = \{t_1, t_2, t_3, ..., t_i, ..., t_k\} \qquad 1 \le i \le k$

(2) $N = \{n_1, n_2, n_3, ..., n_j, ..., n_p\} \qquad 1 \le j \le p \qquad [(n_j, n_{j'}) \in N, n_j \neq n_{j'}]$

(3) $PT_{t_i n_j} = t_{i.pt} \times n_{j.s}$

(4) $\forall\ (n_j, n_{j'}) \in N,\ n_j \neq n_{j'}$ if $(n_{j.s} \neq n_{j'.s}) \rightarrow PT_{t_i n_j} \neq PT_{t_i n_{j'}}$

(5) $PC_{t_i n_j} = PT_{t_i n_j} \times n_{j.c}$

(6) $\forall\ (n_j, n_{j'}) \in N,\ n_j \neq n_{j'}$ if $(n_{j.c} \neq n_{j'.c}) \rightarrow PC_{t_i n_j} \neq PC_{t_i n_{j'}}$

Scheduling refers to determining the start and end times of a set of tasks according to certain constraints that are related to time or resources [2]. Input tasks enter the system dynamically at independent times ($t_{i.et}$). All tasks have a certain setup time ($t_{i.st}$) and processing time ($t_{i.pt}$). The start time of processing the task t_i on the node $n_j(ST_{t_i n_j})$ is equal to the maximum of two values, the task t_i entrance time ($t_{i.et}$) and the finish time of the previous task ($t_{i.pre}$) on the node $n_j(FT_{t_{i.pre} n_j})$, after the time specified for setup is passed ($t_{i.st}$) (Eq.(7)). The completion or finish time of the task t_i on the node $n_j(FT_{t_i n_j})$ is equal to the sum of the start time and processing time of the task (Eq.(8)).

(7) $$ST_{t_i n_j} = Max(t_{i.et}, FT_{t_{i.pre} n_j}) + t_{i.st}$$

(8) $$FT_{t_i n_j} = ST_{t_i n_j} + PT_{t_i n_j}$$

There is a deadline ($t_{i.dt}$) for completing tasks ($t_{i.dt} > t_{i.et}$). If the task t_i is completed after the deadline ($FT_{t_i n_j} > t_{i.dt}$), it is penalized and its profit is reduced, which decreases the total profit of the system. The amount of penalty depends on the delay. It is assumed that the delay penalty does not exceed the profit considered for the tasks ($t_{i.r}$). Considering such conditions, all completed tasks are categorized into two sets. The set T_{nd} ($T_{nd} \subseteq T$) includes the tasks that are completed before the deadline ($FT_{t_i n_j} \le t_{i.dt}$) and are without delay, so they will result in the predicted profit. The set T_d ($T_d \subset T$) includes the tasks that are completed after the deadline with delay ($FT_{t_i n_j} > t_{i.dt}$), therefore they are penalized.

It is assumed that for each unit time delay, a percentage (p_{dp}) of their profit

($t_{i.r}$) is reduced. If the finish time of the task t_i is assumed to be $t_{i.ft}$ ($t_{i.ft} = FT_{t_i n_j}$), the delay of the task t_i ($t_{i.d}$) considering the set T would be as in Eq.(9) and Eq.(10) considering the set T_d. Since it is assumed that the delay penalty ($t_{i.dp}$) never exceeds the predicted profit ($t_{i.dp} \leq t_{i.r}$), it is equal to the minimum of two values; The product of the task t_i delay ($t_{i.d}$) multiplied by p_{dp} percent of its profit, and its profit amount (Eq.(11)). Therefore, the profits resulting from the completion of the tasks of the set T_{nd} and the set T_d are given in Eq.(12) and Eq.(13), respectively. Considering the equations, the total profit of the system (prf) is equal to the subtraction of the processing cost and the penalty of the tasks from the sum of their profits, which is calculated using Eq.(14). If the task t_i processing cost is represented as $t_{i.c}$ ($t_{i.c} = PC_{t_i n_j}$), the total profit is shown using Eq.(15).

$$t_{i.d} = \max[(t_{i.ft} - t_{i.dt}), 0], (t_i \in T) \tag{9}$$

$$t_{i.d} = t_{i.ft} - t_{i.dt}, (t_i \in T_d) \tag{10}$$

$$t_{i.dp} = \min[(t_{i.d} \times (p_{dp} \times t_{i.r})), t_{i.r}] \tag{11}$$

$$prf_{t_i n_j} = t_{i.r} - PC_{t_i n_j}, (t_i \in T_{nd}) \tag{12}$$

$$prf_{t_i n_j} = t_{i.r} - PC_{t_i n_j} - t_{i.dp}, (t_i \in T_d) \tag{13}$$

$$prf = \sum_{t_i \in T} t_{i.r} - \sum_{t_i \in T} \sum_{n_j \in N} PC_{t_i n_j} - \sum_{t_i \in T_d} t_{i.dp} \tag{14}$$

$$prf = \sum_{t_i \in T_{nd}} (t_{i.r} - t_{i.c}) + \sum_{t_i \in T_d} (t_{i.r} - t_{i.c} - t_{i.dp}) \tag{15}$$

The investigated model tries to consider the real and significant conditions and limitations of heterogeneous parallel processing systems. However, the existence of some other special conditions and challenges in some real systems cannot be denied, which have not been addressed in this model. For example, the challenge of the possibility of failure in processing resources has not been considered in this model. This challenge and even other challenges can be addressed in future research.

4. The proposed algorithm (ESTS)

To solve the problem described in the previous section, a two-step solution is proposed (ESTS). First, a vector approach is presented for allocating tasks to heterogeneous resources and prioritizing their execution. Then, the vector allocation approach is improved and extended using a tabu search-based strategy to achieve better results for the problem objectives. Figure 1 shows the flowchart of the proposed ESTS solution.

4.1. Allocation approach.

The allocation approach is considered in the form of an allocation vector consisting of input tasks and heterogeneous resources. The vector structure determines how to allocate and schedule tasks on resources. The vector length (l_v) calculated using Eq.(16). The vector structure

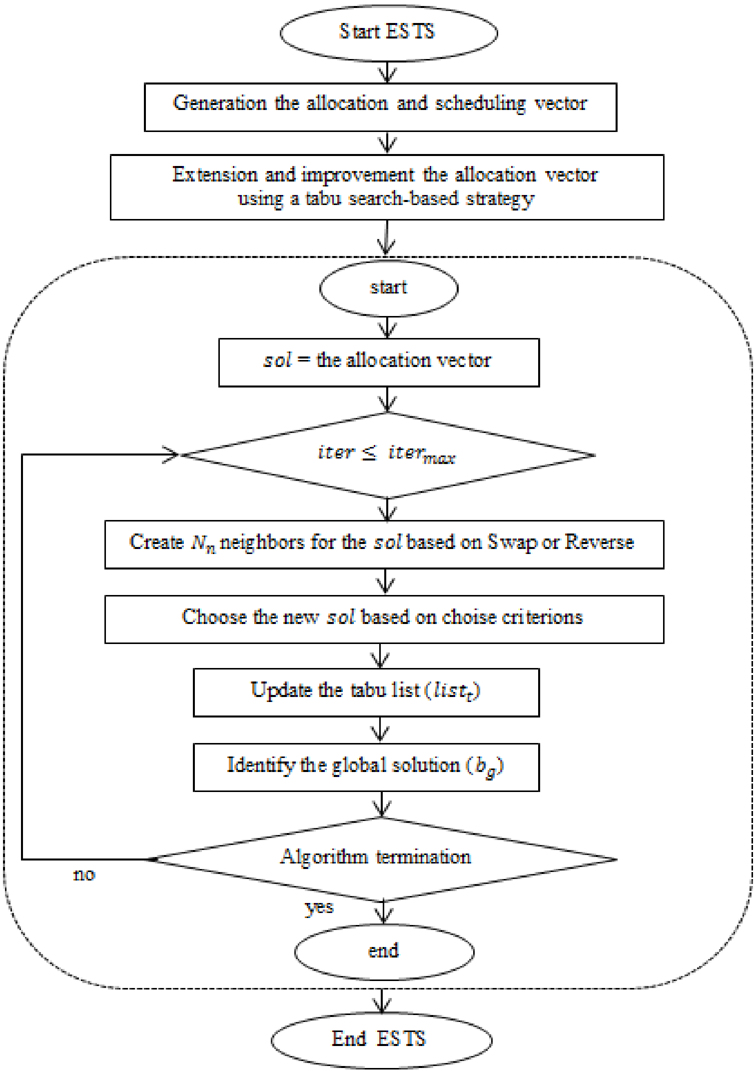


FIGURE 1. Flowchart of the proposed ESTS model

is formed by the juxtaposition of the numbers (x) in the range of 1 to l_v (Eq.(17)), which are distributed randomly. The distribution of the elements in the vector represents how the tasks are allocated to the nodes and their processing order. The numbers between 1 and k represent the task number (Eq.(18)). The numbers greater than k represent the processing nodes. For accurate numbering of nodes, the numbers greater than k can be subtracted from k to specify the nodes number (Eq.(19)). Thus, this specifies the nodes number from 1 to $p-1$ (except the last resource, n_p). For allocation, first, the position of the first node from the left side of the vector is specified; then, all elements (tasks) before that node are allocated to it, and they should be processed in the same order as they are in the vector. Then, the allocated node and tasks are eliminated from the vector, and the above process is repeated for the remained elements of the vector. This process is continued until the last node on the vector is eliminated. In the end, the remained tasks are allocated to the last node (n_p).

(16) $$l_v = k + p - 1$$

(17) $$x | 1 \leq x \leq l_v$$

(18) $$x | (1 \leq x \leq k); i = x \longrightarrow t_i | (1 \leq i \leq k)$$

(19) $$x | (k < x \leq l_v); j = x - k \longrightarrow n_j | (1 \leq j < p)$$

4.2. **Extension of the allocation approach.**

4.2.1. *Tabu search.* The tabu search (TS) is a metaheuristic optimization method originally suggested by Glover and Laguna (1997). TS is a modification of the local search method [7]. The TS algorithm first starts from a single random solution and updates it to one of its current neighbors. This process continues until the user-specified criterion is met and the best solution found during the search process is returned. The TS algorithm uses its memorized ability to avoid searching previously visited points using the Tabu list [1]. Tabu list is a tool in the TS that prevents being trapped in local optimal. Tabu search is known as the best local search method to get out of the local optimum and increase the global optimization capability [14].
To achieve better results for the problem objectives, a tabu search-based strategy is used to extend and improve the vector allocation approach. The main steps of the algorithm to improve the described vector approach for allocation and scheduling are explained below. The used symbols are shown in Table 3, and the proposed algorithm is given in Algorithm 1.

4.2.2. *Generating the initial solution.* The tabu search algorithm starts moving from an initial solution. The structure of the initial solution is formed according to the type of problem. In the proposed algorithm for solving the problem, the initial solution (sol) is the described allocation vector. Therefore, the length of the sol (l_s) is equal to the vector length ($l_s = l_v$). At first, this solution is considered the best solution ($b_g = sol$), which varies in the next iterations of the algorithm.

4.2.3. *Choose and Move to the neighborhood.* Like all metaheuristic algorithms, the initial solution should change such that it tends to get better. Therefore, it is necessary to generate a number (N_n) of neighborhood solutions. In the

Table 3. Symbols used to formulate the problem by the proposed tabu search-based algorithm

Symbols	Description
sol	The solution selected by the algorithm (to move towards it)
l_s	The solution length
N_n	The number of neighbors generated for the solution (sol)
b_{fn}	The best free neighbor (based on the fitness function)
b_{tn}	The best tabu neighbor
b_g	The best solution found by the algorithm
$iter$	The current iteration number of the algorithm execution
$iter_{\max}$	The maximum number of iterations (to stop the algorithm)
$iter_{\min}$	The minimum number of iterations (to execution the algorithm)
$iter_{ni}$	The number of iterations considered in the algorithm based on the no-improvement condition
$iter_{stop}$	The iteration in which the execution of the algorithm ends
p_l	The effective factor on the tabu movement limit
l_t	The tabu movement limit in the TS algorithm
$list_t$	$list_t[l_s, l_s]$ is the tabu list to place tabu movements

Algorithm 1 Proposed algorithm based on the tabu search

sol $\leftarrow$ initial solution ;
$b_g \leftarrow sol$;
$list_t[l_s, l_s] = 0$;
for $iter = 1 : iter_{\max}$ **do**
    Generate N_n neighbors for the *sol* (based on swap or reverse) ;
    Identify the best free neighbor (b_{fn}) and the best tabu neighbor (b_{tn}) ;
    Choose the new solution (sol) ;
    Update the tabu list ;
    Identify the global solution (b_g) ;
    Check the algorithm termination ;
end for

proposed model, some efficient mechanisms used to mutate chromosomes in the genetic algorithm, such as the swap and reverse, are used to generate mutated and better neighborhood solutions. Creating better neighborhood solutions leads to faster convergence to good solutions. Consequently, the computational time of the algorithm to achieve the appropriate solutions is reduced. In swap and reverse, two elements (m_1, m_2) are randomly chosen on the current solution. In swap, their positions are exchanged, and in reverse, all elements between them are substituted inversely. The fitness of generated neighbors is calculated considering the objective function ($fit = prf$). The best free neighbor (b_{fn}) and the best tabu neighbor (b_{tn}) are detected, and their fitness is compared. If the fitness of the best free neighbor ($b_{fn}.fit$) is better, it is accepted as the new solution (Eq.(20)). But, if the fitness of the best tabu neighbor ($b_{tn}.fit$) is better than the fitness of the best free neighbor and also better than the best solution found by the algorithm (b_g), it is accepted as the new solution (Eq.(21)). Also, if there is no free neighbor and all are in the tabu list, the best tabu neighbor is accepted as the new solution (Eq.(22)). Then, the best solution found by the algorithm is updated. To this end, the fitness of the new solution ($sol.fit$) is compared with the fitness of the best solution found by the algorithm ($b_g.fit$), and if it is better, it is substituted (Eq.(23)).

$$\text{(20)} \quad if\ b_{fn}.fit \geq b_{tn}.fit \longrightarrow sol = b_{fn}$$

$$\text{(21)} \quad if\ b_{tn}.fit > b_{fn}.fit \quad \& \quad b_{tn}.fit > b_g.fit \longrightarrow sol = b_{tn}$$

$$\text{(22)} \quad if\ b_{fn} = \emptyset \longrightarrow sol = b_{tn}$$

$$\text{(23)} \quad if\ sol.fit > b_g.fit \longrightarrow b_g = sol$$

4.2.4. *Updating the tabu list.* The tabu list is a tool in the tabu search algorithm that prevents the algorithm from being trapped in local optimal. The initial value of the tabu list is zero ($list_t[l_s, l_s] = 0$). The tabu list is updated whenever the algorithm moves towards the new solution; that is, the movement towards the neighbor solution is inserted in the tabu list to prevent the algorithm from returning to that solution and creating a cycle. After inserting the new movement, several moves that were previously inserted in the list, are eliminated. The duration that a movements remain in the tabu list is determined by the tabu limit (l_t), which is given by Eq.(24). p_l is the effective factor on the tabu limit. To update the tabu list and insert a new movement in the list, Eq.(25) to Eq.(27) are applied. m_1 and m_2 are two positions selected on the solution (sol) using the swap or reverse.

$$\text{(24)} \quad l_t = \lfloor (iter_{\max} - iter_{\min}) \times p_l \rfloor$$

$$\text{(25)} \quad list_t = list_t - 1$$

$$\text{(26)} \quad list_t = \max(list_t, 0)$$

$$\text{(27)} \quad list_t(m_1, m_2) = l_t;\ list_t(m_2, m_1) = l_t\ \ (m_1 \neq m_2; m_1, m_2 \in \{1, 2, ..., l_s\})$$

4.2.5. *Algorithm termination.* Several conditions are considered to terminate the algorithm. According to the performance of the algorithm and evaluation of the results, two values are considered as the minimum ($iter_{\min}$) and the maximum number of algorithm iterations ($iter_{\max}$). The maximum value is determined considering the experiment results such that in more iterations, the results are not improved significantly; therefore, to prevent time loss, it is better to terminate the algorithm. Furthermore, if no improvement is achieved for a certain number of iterations ($iter_{ni}$), the algorithm is terminated.

5. Computational experiments

5.1. **Experimental data.** The implementation and experiments have been done in MATLAB and the Intel Core i5 system. Several suitable benchmark files of different sizes (in terms of the number of tasks and heterogeneous resources) are used to test the proposed algorithm and to evaluate its effectiveness for solving small and large-size problems. The benchmark files include a file with 50 tasks, which are processed on 6 and 10 heterogeneous resources, respectively, and a larger file containing 500 tasks, which are processed on 30, 50, and 100 resources, respectively. Considering the problem conditions, in the benchmarks, the input tasks have separate numbers, entry time, setting time, processing time, deadline, and profit. Also, processing resources have different numbers, processing speed rates, and processing cost rates. The data for parameters related to tasks and heterogeneous resources have been randomly generated in certain ranges that correspond to real-world conditions. For example, the heterogeneous resource speed rate varies between the values of 0.5 and 1.5. The due date of the tasks is determined according to their entry time, setting time, and processing time.

5.2. **Testing the proposed model.**

5.2.1. *Parameters setting.* The most important parameters of the proposed ESTS algorithm include the number of neighborhood solutions (N_n), the tabu limit (l_t), the minimum ($iter_{min}$) and maximum ($iter_{max}$) repetition of the algorithm, and the number considered to repeat the algorithm for the condition of not improving the results ($iter_{ni}$). The parameters are adjusted based on the experiment by assigning different values to them, repeating the experiments, benchmark file size, and attention to the obtained results, so that the best value for the objective functions is obtained. For example, in the experiments, the number of neighbors (N_n) gradually increases from 20 to 500 to see their effect on the results. Each experiment is repeated 12 times ($exe\ i,\ 1 \leq i \leq 12$) to avoid sufficing the random results and obtain a more accurate result. The result of each execution is the ratio of profit to computational time ($exe\ i \rightarrow \frac{prf}{computational\ time}$). The effect of various parameters on the results will be determined in the experiments.

5.2.2. *Test results.* The proposed solution is tested on the specified benchmark files. First, the benchmark file containing 50 tasks is respectively applied on 6 and 10 heterogeneous resources. The results are given in Table 4. Figure 2 shows the graph of increasing the amount of profit in an experiment using the proposed ESTS for 50 tasks and 10 resources. The number of neighbors (N_n), the iteration number at which the algorithm is terminated ($iter_{stop}$), the obtained profit (prf), and the computational time is represented below the figure.

To evaluate the efficiency of the proposed solution for solving large samples, a larger benchmark file with 500 tasks is tested on 30, 50, and 100 heterogeneous resources, respectively. The results are given in Table 5. According to the Table, using the file with 500 tasks and 30 nodes, the maximum profit obtained is 2142, and the computational time to achieve this profit is 421. In the experiment with 50 nodes, the maximum profit is 3032 with an execution time of 437. Similarly, in the experiment with 100 nodes, the maximum profit

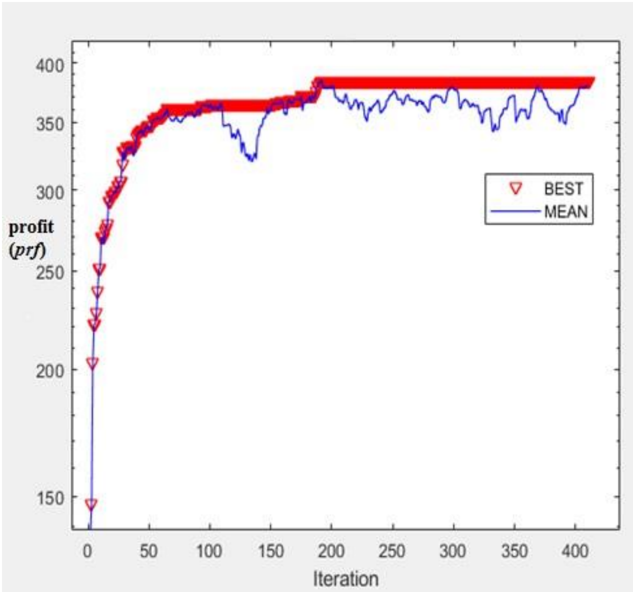
Table 4. Experiment results using the proposed ESTS algorithm

$exe\ i \rightarrow \frac{prf}{computational\ time}\ 1 \leq i \leq 12$														
$iter_{\min} = 300,\ iter_{\max} = 1200,\ iter_{ni} = 220,\ p_l = 0.06 \rightarrow l_t = 54$														
	$k = 50,\ p = 6$							$k = 50,\ p = 10$						
$N_n \rightarrow$	20	50	80	100	200	300	500	20	50	80	100	200	300	500
exe 1	$\frac{353.0}{2.9}$	$\frac{375}{7.1}$	$\frac{395.2}{16.8}$	$\frac{403.2}{9.8}$	$\frac{405.7}{49.1}$	$\frac{406.7}{38.8}$	$\frac{411.3}{68.3}$	$\frac{393.0}{3.1}$	$\frac{420.1}{13.9}$	$\frac{424.6}{27.5}$	$\frac{428.3}{20.0}$	$\frac{428.1}{67.0}$	$\frac{422.9}{57.4}$	$\frac{428.0}{52.0}$
exe 2	$\frac{351.8}{2.7}$	$\frac{402.3}{10.9}$	$\frac{394.9}{16.9}$	$\frac{402.5}{12.3}$	$\frac{413.1}{32.4}$	$\frac{405.3}{42.8}$	$\frac{411.8}{77.0}$	$\frac{395.3}{4.8}$	$\frac{412.6}{5.8}$	$\frac{422.6}{15.1}$	$\frac{427.4}{31.1}$	$\frac{423.2}{28.6}$	$\frac{432.5}{86.3}$	$\frac{429.1}{80.3}$
exe 3	$\frac{344.4}{3.6}$	$\frac{390.8}{5.7}$	$\frac{401.4}{20.0}$	$\frac{399.5}{15.5}$	$\frac{408.9}{43.4}$	$\frac{407.0}{37.4}$	$\frac{415.5}{73.7}$	$\frac{382.8}{2.6}$	$\frac{419.1}{10.8}$	$\frac{428.8}{20.5}$	$\frac{420.2}{20.7}$	$\frac{426.1}{23.7}$	$\frac{424.2}{61.1}$	$\frac{430.0}{112.7}$
exe 4	$\frac{350.8}{5.0}$	$\frac{391.3}{5.0}$	$\frac{400.2}{9.2}$	$\frac{374.4}{14.0}$	$\frac{410.1}{35.7}$	$\frac{408.3}{26.7}$	$\frac{387.0}{43.5}$	$\frac{390.3}{2.1}$	$\frac{419.4}{7.4}$	$\frac{421.6}{17.5}$	$\frac{425.5}{34.3}$	$\frac{426.0}{23.9}$	$\frac{430.8}{43.4}$	$\frac{430.8}{68.7}$
exe 5	$\frac{351.8}{2.0}$	$\frac{388.5}{4.4}$	$\frac{399.4}{23.6}$	$\frac{402.4}{8.9}$	$\frac{398.9}{27.4}$	$\frac{411.4}{28.5}$	$\frac{413.5}{62.8}$	$\frac{392.7}{5.9}$	$\frac{421.5}{10.7}$	$\frac{425.5}{18.2}$	$\frac{427.3}{32.9}$	$\frac{429.3}{40.5}$	$\frac{431.5}{65.6}$	$\frac{437.3}{84.7}$
exe 6	$\frac{354.1}{3.4}$	$\frac{394.3}{9.1}$	$\frac{398.3}{18.0}$	$\frac{396.1}{17.3}$	$\frac{411.3}{33.3}$	$\frac{409.3}{44.6}$	$\frac{344.7}{66.7}$	$\frac{396.5}{3.8}$	$\frac{416.7}{10.9}$	$\frac{420.0}{11.2}$	$\frac{428.3}{15.1}$	$\frac{429.7}{66.5}$	$\frac{427.0}{83.3}$	$\frac{429.7}{77.0}$
exe 7	$\frac{339.2}{2.6}$	$\frac{398.3}{9.8}$	$\frac{396.5}{13.2}$	$\frac{408.2}{12.6}$	$\frac{407.1}{24.9}$	$\frac{409.4}{37.4}$	$\frac{384.4}{90.0}$	$\frac{396.0}{3.8}$	$\frac{411.7}{5.0}$	$\frac{420.7}{14.8}$	$\frac{424.6}{23.3}$	$\frac{425.9}{21.7}$	$\frac{429.7}{41.4}$	$\frac{433.0}{113.6}$
exe 8	$\frac{365.2}{5.9}$	$\frac{372.9}{6.1}$	$\frac{393.6}{9.9}$	$\frac{403.8}{13.2}$	$\frac{407.6}{47.9}$	$\frac{401.3}{54.8}$	$\frac{407.7}{53.9}$	$\frac{389.1}{2.9}$	$\frac{415.5}{9.2}$	$\frac{421.8}{12.1}$	$\frac{428.9}{23.6}$	$\frac{430.8}{41.7}$	$\frac{429.7}{60.7}$	$\frac{433.6}{143.2}$
exe 9	$\frac{366.5}{4.7}$	$\frac{393.2}{7.9}$	$\frac{397.4}{15.0}$	$\frac{376.9}{12.2}$	$\frac{407.7}{12.3}$	$\frac{410.4}{24.3}$	$\frac{357.8}{51.9}$	$\frac{391.3}{2.5}$	$\frac{416.7}{5.0}$	$\frac{421.6}{16.2}$	$\frac{421.0}{18.7}$	$\frac{426.9}{23.5}$	$\frac{429.4}{82.4}$	$\frac{429.3}{150.6}$
exe 10	$\frac{364.6}{6.5}$	$\frac{383.0}{6.5}$	$\frac{400.1}{11.0}$	$\frac{396.8}{13.8}$	$\frac{411.5}{36.4}$	$\frac{396.2}{65.4}$	$\frac{403.9}{74.6}$	$\frac{391.2}{2.4}$	$\frac{417.3}{7.8}$	$\frac{422.1}{15.5}$	$\frac{429.1}{24.5}$	$\frac{432.0}{40.8}$	$\frac{428.5}{60.1}$	$\frac{433.9}{124.1}$
exe 11	$\frac{361.5}{4.3}$	$\frac{383.0}{4.5}$	$\frac{396.2}{11.6}$	$\frac{401.0}{11.0}$	$\frac{410.2}{43.9}$	$\frac{401.1}{40.9}$	$\frac{407.4}{44.3}$	$\frac{393.5}{2.3}$	$\frac{407.5}{4.8}$	$\frac{426.3}{17.1}$	$\frac{422.2}{13.1}$	$\frac{427.6}{31.1}$	$\frac{419.3}{28.7}$	$\frac{421.5}{67.4}$
exe 12	$\frac{363.9}{4.0}$	$\frac{392.3}{7.5}$	$\frac{403.3}{16.7}$	$\frac{399.7}{14.6}$	$\frac{407.3}{21.6}$	$\frac{409.8}{42.5}$	$\frac{403.4}{76.2}$	$\frac{382.0}{1.9}$	$\frac{422.9}{7.7}$	$\frac{424.5}{21.7}$	$\frac{421.9}{12.3}$	$\frac{429.6}{56.0}$	$\frac{426.8}{53.4}$	$\frac{425.4}{72.5}$

is 3881 with an execution time of 358.
Considering the experiment results, it was observed that the parameters of the number of neighbors, the maximum repetition, and the number of repetitions for the no-improvement condition are the most important parameters affecting the results, and by increasing their value, better results are obtained, but the execution and computational time of the algorithm increases. Also, results showed that the benchmark size has a great effect on setting the parameters of the maximum repetition and repetitions for the no-improvement.

6. **Evaluation and comparison**

Scheduling problems are very diverse and have their conditions, characteristics, and limitations. Considering the studies conducted, no problem was observed that is exactly similar to the conditions considered in the problem under study. Therefore, it would not be fair to compare the proposed solution with previous works because of the differences in the type of problem. Consequently, it was decided to implement other powerful algorithms and test them under the same conditions and benchmarks. Considering the studies, it was observed that the genetic and the simulated annealing algorithms have shown good and successful performance in solving various scheduling problems. One of the most common techniques to deal with scheduling problems is the genetic algorithm. GA has been the most popular technique in evolutionary computation research [21]. The simulated annealing algorithm is a simple and effective meta-heuristic optimization algorithm for solving optimization problems in large search spaces. Also, simulated annealing has proven its success as a single-state optimization search algorithm for both discrete and continuous problems [3]. Consequently, in order to evaluate the efficiency of the proposed



BEST: the best profit obtained in each iteration
MEAN: the average profits obtained in each iteration

parameters value: $k = 50$, $p = 10$, $N_n = 50$, $p_l = 0.06 \rightarrow l_t = 54$
$iter_{min} = 300$, $iter_{max} = 1200$, $iter_{ni} = 220$

results: $iter_{stop} = 411$, $prf = 384.5$, $computational\ time = 7.2$

Figure 2. Increasing the amount of profit in a experiment using the proposed ESTS

solution, due to the existence of various and new conditions in the problem under study and the lack of seeing a problem with completely identical conditions in the literature, the proposed model is compared with both an extended scheduling based on a genetic algorithm (ESGA) and an extended scheduling based on a simulated annealing algorithm (ESSA). In both, the same vector allocation approach used in the proposed ESTS has been improved and extended.
The proposed ESTS results are compared with the results of ESGA with a large number of chromosomes and a high number of generations, so that an estimate of the closeness of this answer to the optimal answer can be obtained according to the nature of the genetic algorithm in searching the whole problem space.

TABLE 5. Experiment results using the proposed ESTS algorithm

$iter_{\min} = 1000,\ iter_{\max} = 3000,\ iter_{ni} = 250,\ exe\ i \rightarrow \frac{prf}{computational\ time},\ 1 \leq i \leq 10$												
$\frac{k}{p}$	l_t	N_n	exe 1	exe 2	exe 3	exe 4	exe 5	exe 6	exe 7	exe 8	exe 9	exe 10
$\frac{500}{30}$	60	20	$\frac{1605}{76.1}$	$\frac{1792}{81.8}$	$\frac{1738}{71.5}$	$\frac{1810}{82.6}$	$\frac{1728}{74.6}$	$\frac{1762}{60.0}$	$\frac{1732}{74.1}$	$\frac{1785}{65.1}$	$\frac{1741}{78.1}$	$\frac{1763}{74.5}$
	60	40	$\frac{1873}{134.3}$	$\frac{1899}{132.1}$	$\frac{1897}{131.7}$	$\frac{1963}{132.3}$	$\frac{1858}{132.9}$	$\frac{1934}{111.9}$	$\frac{1991}{151.3}$	$\frac{1897}{125.8}$	$\frac{1879}{134.2}$	$\frac{1945}{130.5}$
	90	70	$\frac{2100}{277.2}$	$\frac{1980}{225.3}$	$\frac{1943}{303.2}$	$\frac{2092}{299.7}$	$\frac{2028}{296.9}$	$\frac{2100}{320.8}$	$\frac{2058}{305.2}$	$\frac{1986}{297.4}$	$\frac{2081}{299.9}$	$\frac{2044}{296.5}$
	90	100	$\frac{2141}{426.3}$	$\frac{2142}{421.8}$	$\frac{2064}{425.2}$	$\frac{2049}{380.0}$	$\frac{2135}{359.4}$	$\frac{2089}{373.2}$	$\frac{2087}{374.6}$	$\frac{2125}{360.4}$	$\frac{2112}{384.3}$	$\frac{2095}{405.8}$
$\frac{500}{50}$	60	20	$\frac{2298}{52.9}$	$\frac{2427}{49.7}$	$\frac{2516}{52.7}$	$\frac{2270}{41.2}$	$\frac{2328}{44.1}$	$\frac{2378}{43.3}$	$\frac{2451}{44.1}$	$\frac{2335}{46.8}$	$\frac{2482}{51.6}$	$\frac{2504}{44.3}$
	75	40	$\frac{2736}{135.7}$	$\frac{2759}{136.3}$	$\frac{2830}{164.5}$	$\frac{2684}{96.8}$	$\frac{2793}{185.3}$	$\frac{2798}{136.2}$	$\frac{2841}{143.0}$	$\frac{2766}{118.4}$	$\frac{2794}{133.6}$	$\frac{2818}{128.4}$
	75	70	$\frac{2975}{309.2}$	$\frac{2939}{311.5}$	$\frac{2971}{371.5}$	$\frac{2914}{369.3}$	$\frac{2925}{367.0}$	$\frac{2938}{361.1}$	$\frac{2958}{360.4}$	$\frac{2933}{341.5}$	$\frac{2972}{328.9}$	$\frac{2948}{363.8}$
	75	100	$\frac{2956}{437.1}$	$\frac{2995}{436.2}$	$\frac{264}{436.6}$	$\frac{2993}{439.2}$	$\frac{2904}{439.0}$	$\frac{2949}{440.8}$	$\frac{2982}{437.2}$	$\frac{2991}{439.5}$	$\frac{3032}{437.1}$	$\frac{2975}{439.7}$
$\frac{500}{100}$	60	20	$\frac{3160}{59.1}$	$\frac{3224}{51.0}$	$\frac{3243}{76.4}$	$\frac{3204}{55.6}$	$\frac{3163}{59.1}$	$\frac{3167}{49.2}$	$\frac{3209}{52.6}$	$\frac{3241}{57.3}$	$\frac{3184}{54.9}$	$\frac{3227}{62.7}$
	60	45	$\frac{3552}{87.1}$	$\frac{3673}{221.7}$	$\frac{3691}{207.1}$	$\frac{3588}{129.3}$	$\frac{3677}{262.1}$	$\frac{3697}{269.1}$	$\frac{3675}{185.3}$	$\frac{3649}{144.8}$	$\frac{3691}{252.7}$	$\frac{3676}{196.5}$
	90	75	$\frac{3754}{235.1}$	$\frac{3767}{333.8}$	$\frac{3766}{340.4}$	$\frac{3760}{316.4}$	$\frac{3753}{298.8}$	$\frac{3763}{238.9}$	$\frac{3759}{284.5}$	$\frac{3765}{322.7}$	$\frac{3766}{305.7}$	$\frac{3758}{244.4}$
	90	100	$\frac{3829}{493.6}$	$\frac{3861}{454.1}$	$\frac{3869}{447.9}$	$\frac{3881}{358.0}$	$\frac{3821}{417.5}$	$\frac{3846}{401.7}$	$\frac{3854}{370.4}$	$\frac{3860}{418.7}$	$\frac{3858}{426.9}$	$\frac{3872}{350.8}$

In ESGA, the three main genetic operators including selection, crossover, and mutation, have been adopted. The proposed ESTS is also compared with the ESSA with a specified initial and final temperature and a linear temperature reduction method. Several (Nn) neighbors are generated and the fitness of the best neighbor $(b_n.fit)$ is compared with the fitness of the current answer $(a.fit)$; if it is higher it is selected as the new answer (Eq.(28)); otherwise, the Boltzmann probability function (PR) is calculated (Eq.(29) and (Eq.(30)). A random number (between zero and one) is generated; if it is less than or equal to the Boltzmann probability, the best neighbor is chosen as the new answer; otherwise, the same current answer will be considered the new answer. Then, the temperature is reduced. The temperature reduction mechanism (TR) is considered linearly (Eq.(31) and Eq.(32)). The initial temperature (T_b) and final temperature (T_f) are considered equal to 100 and 1, respectively (T_i: the temperature of the i'th iteration).

$$if\ (b_n \cdot fit > a \cdot fit) \rightarrow a = b_n \tag{28}$$

$$else\ \rightarrow E = \frac{(b_n \cdot fit - a \cdot fit)}{a \cdot fit} \tag{29}$$

$$PR = e^{-E/T_i} \tag{30}$$

$$TR = (T_b - T_f)/iter_{\max} \tag{31}$$

$$T_i = T_i - TR \tag{32}$$

The stopping conditions of ESTS, ESGA, and ESSA algorithms are considered the same. The benchmark files are tested under the same software and

hardware conditions. To obtain better results in the ESGA, we have gradually increased the number of initial populations and the probability of crossover and mutation. Also, the number of initial answers and the number of neighbors have gradually increased in the ESSA. Considering the obtained results, the efficiency of the various algorithms can be compared.
The most important goal of analyzing the complexity of algorithms is to estimate their execution time, which is especially considered in this study. In the experiments, the execution time of the algorithms is carefully recorded so that they can be accurately compared and evaluated. In order to evaluate the profitability, the maximum profit obtained using the algorithms is compared. The maximum profit and the algorithm computational time to obtain it for all benchmarks are shown in Table 6. It is observed that in all benchmarks, the proposed ESTS obtains more profit in lower computational times, and as the benchmark size increases, the difference increases. For example, in the experiment of the benchmark with 500 tasks on 50 nodes, the maximum profit obtained using the ESTS, ESGA, and ESSA is about 3032, 2934, and 2864, with a computational time of 437, 1260, and 3760, respectively. The difference in the maximum profit obtained for all benchmarks is shown in Figure 3. Figure 4 shows the difference in computational time of the algorithms to obtain the maximum profit. Furthermore, The difference in computational time of the algorithms for obtaining similar profits is shown in Table 7 and Figure 5.

Table 6. Comparison of the maximum profit obtained

$\frac{k}{p} \rightarrow$		$\frac{50}{6}$	$\frac{50}{10}$	$\frac{500}{30}$	$\frac{500}{50}$	$\frac{500}{100}$
$\frac{prf}{computational\ time}$	$ESTS \rightarrow$	$\frac{415.8}{73.7}$	$\frac{437.3}{74.7}$	$\frac{2142.1}{421.8}$	$\frac{3032.8}{437.1}$	$\frac{3881.6}{358.0}$
	$ESGA \rightarrow$	$\frac{412.8}{96.5}$	$\frac{431.6}{131.0}$	$\frac{2036.1}{1077.6}$	$\frac{2934.1}{1260.6}$	$\frac{3765.3}{1946.0}$
	$ESSA \rightarrow$	$\frac{413.5}{665.1}$	$\frac{431.7}{1043.5}$	$\frac{2063.4}{3052.3}$	$\frac{2864.9}{3760.8}$	$\frac{3685.9}{4602.0}$

TABLE 7. Algorithms computational time for obtaining similar profit

	$\frac{k}{p} = \frac{50}{6}$			$\frac{k}{p} = \frac{50}{10}$			$\frac{k}{p} = \frac{500}{30}$			$\frac{k}{p} = \frac{500}{50}$			$\frac{k}{p} = \frac{500}{100}$		
	ESTS	ESGA	ESSA	ESTS	ESGA	ESSA	ESTS	ESGA	ESSA	ESTS	ESGA	ESSA	ESTS	ESGA	ESSA
	$\frac{401.4}{20.0}$	$\frac{401.7}{30.8}$	$\frac{401.8}{150}$	$\frac{407.5}{4.8}$	$\frac{407.2}{8.2}$	$\frac{407.8}{63.0}$	$\frac{1858}{132}$	$\frac{1853}{488}$	$\frac{1852}{1025}$	$\frac{2427}{79.7}$	$\frac{2251}{60.2}$	$\frac{2451}{866}$	$\frac{3160}{59.1}$	$\frac{3153}{61.4}$	$\frac{2623}{255}$
	$\frac{403.8}{13.2}$	$\frac{403.2}{27.5}$	$\frac{403.3}{165}$	$\frac{412.6}{5.8}$	$\frac{411.4}{11.3}$	$\frac{410.0}{70.6}$	$\frac{1873}{134}$	$\frac{1871}{511}$	$\frac{1868}{1026}$	$\frac{2684}{96.8}$	$\frac{2678}{173}$	$\frac{2534}{911}$	$\frac{3552}{87.1}$	$\frac{3598}{292}$	$\frac{3245}{899}$
$\frac{Prf}{Comp.\ Time}$	$\frac{405.7}{49.1}$	$\frac{405.1}{63.5}$	$\frac{405.9}{311}$	$\frac{421.5}{10.7}$	$\frac{421.8}{21.7}$	$\frac{421.2}{101}$	$\frac{1897}{131}$	$\frac{1880}{509}$	$\frac{1878}{1021}$	$\frac{2830}{164}$	$\frac{2755}{402}$	$\frac{2823}{3756}$	$\frac{3588}{129.3}$	$\frac{3599}{284}$	$\frac{3325}{1429}$
	$\frac{407.1}{24.9}$	$\frac{407.2}{67.2}$	$\frac{407.4}{278}$	$\frac{424.5}{21.7}$	$\frac{424.6}{24.8}$	$\frac{423.1}{142}$	$\frac{2028}{296}$	$\frac{2014}{1097}$	$\frac{2027}{3070}$	$\frac{2938}{361}$	$\frac{2915}{1229}$	$\frac{2839}{3762}$	$\frac{3691}{207}$	$\frac{3677}{276}$	$\frac{3635}{3647}$
	$\frac{411.4}{28.5}$	$\frac{411.2}{73.1}$	$\frac{411.4}{670}$	$\frac{429.1}{24.5}$	$\frac{429.3}{48.1}$	$\frac{429.7}{430}$	$\frac{2049}{380}$	$\frac{2036}{1077}$	$\frac{2044}{3056}$	$\frac{2975}{309}$	$\frac{2929}{1220}$	$\frac{2849}{3767}$	$\frac{3767}{333}$	$\frac{3767}{681}$	$\frac{3648}{4025}$
	$\frac{413.5}{62.8}$	$\frac{412.8}{96.5}$	$\frac{413.5}{665}$	$\frac{430.8}{41.7}$	$\frac{429.2}{57.6}$	$\frac{430.7}{737}$	$\frac{2064}{425}$	$\frac{2061}{1148}$	$\frac{2063}{3052}$	$\frac{3032}{437}$	$\frac{2934}{1260}$	$\frac{2864}{3760}$	$\frac{3829}{493}$	$\frac{3828}{1928}$	$\frac{3681}{4602}$

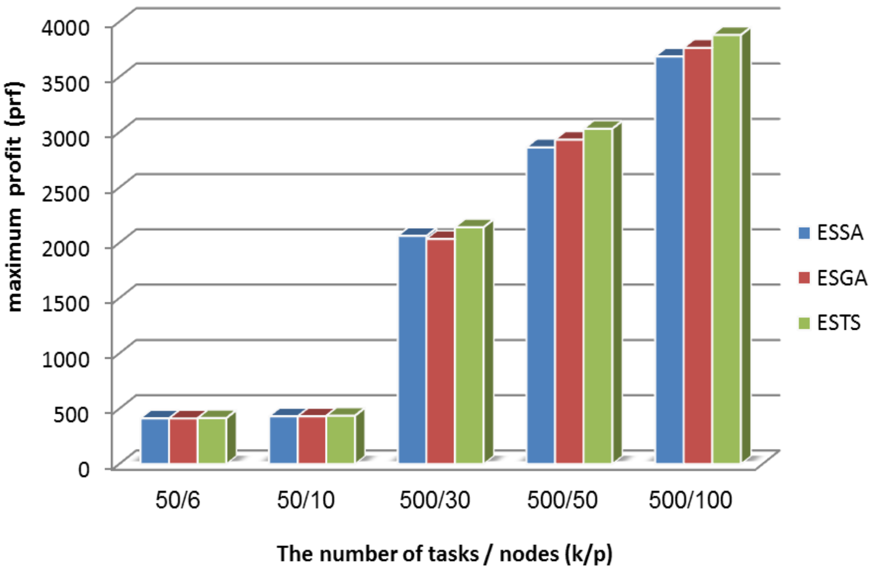


Figure 3. Comparison of the maximum profit obtained

In order to better understand the superiority of the proposed algorithm in terms of profitability and execution time, their improvement percentage has been calculated. Table 8 shows the percentage of profitability improvement of the proposed ESTS compared to the ESGA and ESSA in the various benchmarks. It is observed that, in terms of profitability, the proposed ESTS algorithm is about 0.7% to 5.2% better than ESGA and 0.5% to 5.3% better than ESSA for the various benchmarks. The percentage of computational time improvement of the ESTS for obtaining the most profit is also shown in Table 9. According to the table, it can be claimed that the proposed ESTS is about 1.3 to 5.4 times better than ESGA and 7 to 13 times better than ESSA in terms of computational time.

Table 8. The percentage of profitability improvement (prf) of the proposed ESTS compared to the ESGA and ESSA

$\frac{k}{p} \rightarrow$	$\frac{50}{6}$	$\frac{50}{10}$	$\frac{500}{30}$	$\frac{500}{50}$	$\frac{500}{100}$
$ESGA \rightarrow$	0.726	1.32	5.206	3.36	3.088
$ESSA \rightarrow$	0.556	1.29	3.814	5.86	5.309

Eventually, evaluation and comparison of results show that the proposed ESTS achieves more profit for all benchmark files. ESTS is also more efficient

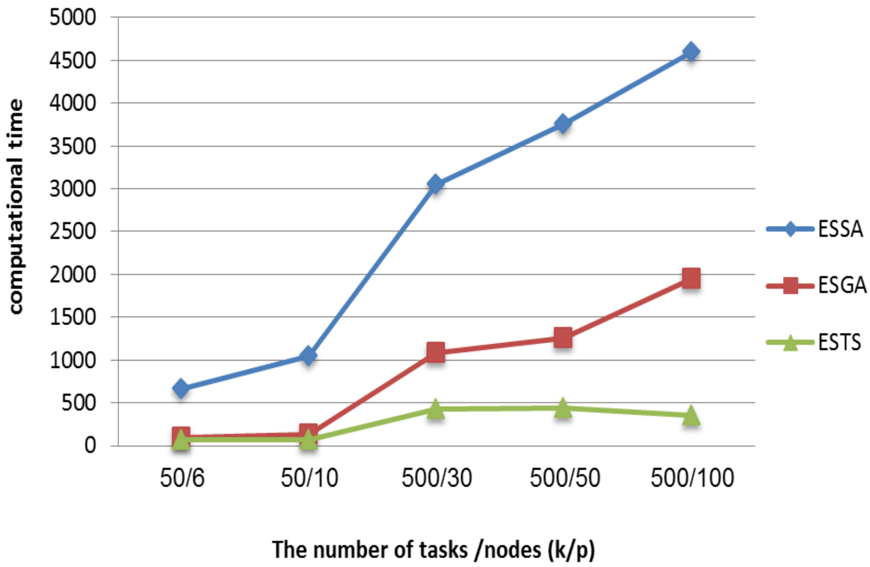


FIGURE 4. Comparison of the algorithms' computational time for obtaining the maximum profit (prf)

TABLE 9. The percentage of the computational time improvement of the proposed ESTS compared to ESGA and ESSA for obtaining the maximum profit (prf)

$\frac{k}{p} \rightarrow$	$\frac{50}{6}$	$\frac{50}{10}$	$\frac{500}{30}$	$\frac{500}{50}$	$\frac{500}{100}$
$ESGA \rightarrow$	130	175	255	288	543
$ESSA \rightarrow$	902	1396	723	860	1285

than ESGA and much more efficient than ESSA in terms of computational time. It seems that one of the most significant reasons for the superiority of the proposed algorithm over other algorithms is that it does not get trapped in local optimal due to the use of memory and the useful tabu list tool. Consequently, it can achieve good solutions faster. Furthermore, using appropriate mechanisms to generate good and mutated neighborhood solutions has led to faster convergence to good solutions. Therfore, the results indicate that the proposed algorithm is an efficient and fast approach and can probably be useful and successful for solving various scheduling problems.

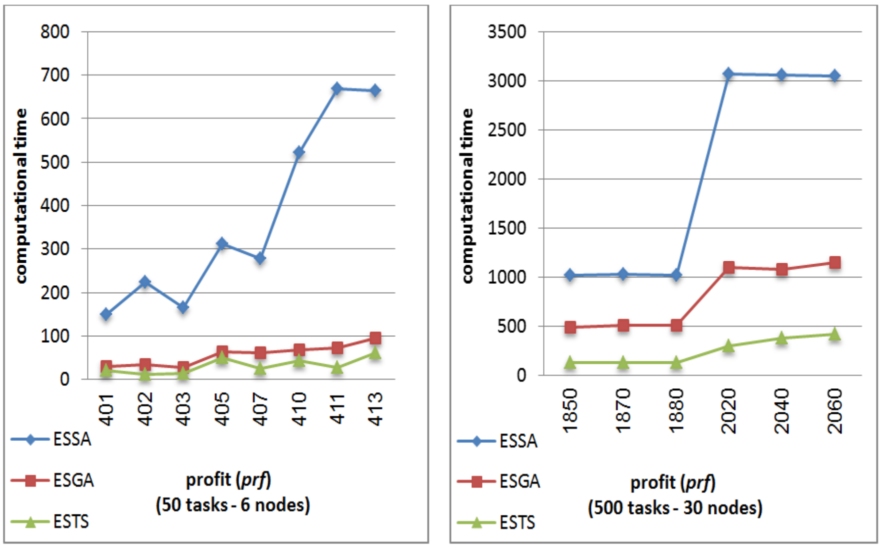


FIGURE 5. Comparison of the algorithms computational time for obtaining similar profit (prf)

7. Conclusion and future work

This paper has investigated the efficiency and profitability of heterogeneous parallel systems. Proper use of the system's heterogeneous resource capacity is necessary to achieve high efficiency and profitability, and an efficient allocation and scheduling approach is very effective in achieving goals. In this study, the system resources are heterogeneous in terms of processing speed and cost. Input tasks have a specified profit and time limit. If a task is completed after the time limit, it will be penalized and its profit will be reduced. The penalty is proportional to delay. The main goal is to improve the system's profitability using an efficient solution with low computational time. The proposed approach is an extended tabu search-based scheduling algorithm (ESTS), which reaches suitable solutions in a low computational time. In order to evaluate the efficiency of the proposed solution, an experimental design was carried out in comparison with both an extended scheduling based on a genetic algorithm (ESGA) and an extended scheduling based on a simulated annealing algorithm (ESSA) under the same conditions. Benchmark files of different sizes were tested, and the results showed that the proposed ESTS obtained good-quality solutions in lower computational times. The results showed that, in terms of profitability, the proposed ESTS is about 0.7% to 5.2% better than ESGA and

0.5% to 5.3% better than ESSA for the various benchmarks. In terms of computational time, it is about 1.3 to 5.4 times better than ESGA and 7 to 13 times better than ESSA. Experiments on the benchmark with 500 tasks on 50 and 100 heterogeneous resources showed that the proposed solution can solve large-size problems well. This study has tried to consider the significant conditions and challenges of heterogeneous parallel systems. However, the existence of some other specific challenges in some real systems cannot be denied, such as the failure probability in processing resources, the cancellation of some tasks before or during processing, etc. These challenges as well as other potential ones can be explored in future research.

References

[1] Adamuthe, A. C., Bichkar, R. S. (2012). Tabu search for solving personnel scheduling problem. In 2012 International Conference on Communication. Information & Computing Technology (ICCICT). IEEE, 1–6. https://doi: 10.1109/ICCICT.2012.6398097.

[2] Alazzam, H., Alhenawi, E., & Al, R. (2019). A hybrid job scheduling algorithm based on tabu and harmony search algorithms. J. Supercomput, 75(12),7994-8011. https://doi: 10.1007/s11227-019-02936-0.

[3] Alkhateeb, F., & Abed-alguni, B. H. (2019). A hybrid cuckoo search and simulated annealing algorithm. Journal of Intelligent Systems, 28(4),683-698. https://doi: 10.1515/jisys-2017-0268.

[4] Ben Abdellafou, K., Hadda, H., & Korbaa, O. (2019). An improved tabu search metaheuristic approach for solving scheduling problem with non-availability constraints. Arab. J. Sci. Eng., 44:3369-3379. https://doi: 10.1007/s13369-018-3525-3.

[5] Bisht, J., & Vampugani, V. S. (2022). Load and cost-aware min-min workflow scheduling algorithm for heterogeneous resources in fog, cloud, and edge scenarios. International Journal of Cloud Applications and Computing (IJCAC), 12(1),1-20. https://doi: 10.4018/IJCAC.2022010105.

[6] Bozejko, W., Nadybski, P., & Wodecki, M., (2017). Two level algorithm with tabu search optimization for task scheduling problem in computing cluster environment. In 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), IEEE, 238–242. https://doi: 10.1109/MMAR.2017.8046831.

[7] Bozejko, W., Gnatowski, A., Pempera, J., & Wodecki, M. (2017). Parallel tabu search for the cyclic job shop scheduling problem. Computers & Industrial Engineering, 113: 512–524. https://doi: 10.1016/j.cie.2017.09.042.

[8] Chandran, R., & Kumar, S. R., (2020). Genetic algorithm-based tabu search for optimal energy-aware allocation of data center resources. Soft Comput., 24(7),1-14. https://doi: 10.1007/s00500-020-05240-9.

[9] Chawra, V. K., & Gupta, G. P. (2022). Optimization of the wake-up scheduling using a hybrid of memetic and tabu search algorithms for 3D-wireless sensor networks. International Journal of Software Science and Computational Intelligence (IJSSCI), 14(1), 1-18. https://doi: 10.4018/IJSSCI.300359.

[10] Chen, C., Fathi, M., Khakifirooz, M., & Wu, K., (2022). Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. Comput. Ind. Eng., 165, 1-11. https://doi: 10.1016/j.cie.2021.107915.

[11] Chen, L., & Li, X. (2017). Cloud workflow scheduling with hybrid resource provisioning. J. Supercomput., 74, 6529-6553. https://doi: 10.1007/s11227-017-2043-5.

[12] Cruz-Chávez, M. A., Martínez-Rangel, M. G., & Cruz-Rosales, M. H. (2017). Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. International Transactions in Operational Research, 24(5),1119–1137. https://doi: 10.1111/itor.12195.

[13] Dai, H., Cheng, W., & Guo, P., (2018). An improved tabu search for multi-skill resource-constrained project scheduling problems under step-deterioration. Arab. J. Sci. Eng., 43,3279-3290. https://doi: 10.1007/s13369-017-3047-4.

[14] Grabowski, J., & Wodecki, M. (2004). A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Comput. Oper. Res., 31 (11), 1891-1909. https://doi.org/10.1016/S0305-0548(03)00145-X.

[15] Hajibabaei, M., & Behnamian, J. (2021). Flexible job-shop scheduling problem with unrelated parallel machines and resources-dependent processing times: a tabu search algorithm. Int. J. Manag. Sci. Eng. Manag., 16(4), 242–253. https://doi: 10.1080/17509653.2021.1941368.

[16] Huang, K.C., Hung, C. H., & Hsieh, W. (2018). Revenue maximization for scheduling deadline-constrained mouldable jobs on high-performance computing as a service platform. Int. J. High Perform. Comput. Netw., 11(1),1–13. https://doi: 10.1504/IJHPCN.2018.088874.

[17] Juraszek, J., Sterna, M., & Pesch, E. (2009,December). Revenue maximization on parallel machines. Operations Research Proceedings, 0-21. https://doi: 10.1007/978-3-642-00142-0.

[18] Krim, H., Zufferey, N., Potvin, J. Y., Benmansour, R., & Duvivier, D. (2022). Tabu search for a parallel-machine scheduling problem with periodic maintenance, job rejection and the weighted sum of completion times. J. Sched., 25, 89-105. https://doi: 10.1007/s10951-021-00711-9.

[19] Liu, Y., Meng, L., & Tomiyama, H. (2019). A genetic algorithm for scheduling of data-parallel tasks on multicore architectures. IPSJ Trans. Syst. LSI Des. Methodol., 12, 74–77. https://doi: 10.2197/ipsjtsldm.12.74.

[20] Mathlouthi, I., Gendreau, M., & Potvin, J. (2021). A metaheuristic based on tabu search for solving a technician routing and scheduling problem. Comput. Oper. Res., 125, 105079. https://doi: 10.1016/j.cor.2020.105079.

[21] Momenikorbekandi, A., & F.Abbod, M. (2023). A novel metaheuristic hybrid parthenogenetic algorithm for job shop scheduling problems: Applying an optimization model. IEEE, 11:56027–56045. https://doi: 10.1109/ACCESS.2023.3278372.

[22] Orr, M., & Sinnen, O. (2020). Optimal task scheduling benefits from a duplicate-free state-space. journal of Parallel and Distributed Computing, 146, 158-174. http://arxiv.org/abs/1901.06899.

[23] Romero, M. A. F., García, E. A. R., Ponsich, A. & Gutiérrez, R. A. M. (2018). A heuristic algorithm based on tabu search for the solution of flexible job shop scheduling problems with lot streaming. In Proceedings of the Genetic and Evolutionary Computation Conference, 285–292. https://doi: 10.1145/3205455.3205534.

[24] Singh, R. (2014). Task scheduling in parallel systems using genetic algorithm. Int. J. Comput. Appl., 108(16), 34–40. https://doi: 10.5120/18999-0470.

[25] Singh, S., Kumar, R., & Rao, U. P. (2022). Multi-objective adaptive manta-ray foraging optimization for workflow scheduling with selected virtual machines using time-series-based prediction. International Journal of Software Science and Computational Intelligence (IJSSCI), 14(1),1-25. https://doi: 10.4018/IJSSCI.312559.

[26] Sun, H., Elghazi, R., Gainaru, A., Aupy, G., & Raghavan, P. (2018). Scheduling parallel tasks under multiple resources: list scheduling vs. pack scheduling. In 2018 IEEE 32nd International Parallel and Distributed Processing Symposium (IPDPS), 194–203. https://doi: 10.1109/IPDPS.2018.00029.

[27] Toshev, A. (2019). Particle swarm optimization and tabu search hybrid algorithm for flexible job shop scheduling problem – analysis of test results. Cybernetics and InformationTechnologies, 19(4):26–44. https://doi: 10.2478/cait-2019-0034.

[28] Vela, C. R., Afsar, S., Jose, J., González-rodríguez, I., & Puente, J., (2020). Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling. Computers and Operations Research, 119:104931. https://doi: 10.1016/j.cor.2020.104931.

[29] Umam, M. S., Mustafid, M., & Suryono, S. (2022). A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem. J. King Saud Univ. Comput. Inf. Sci., 34(9),7459-7467. https://doi: 10.1016/j.jksuci.2021.08.025.

[30] Wang, S., & Ye, B. (2019). Exact methods for order acceptance and scheduling on unrelated parallel machines. Comput. Oper. Res., 104,159–173. https://doi: 10.1016/j.cor.2018.12.016.

[31] Wei, H., Li, S., Jiang, H., & Hu, J. (2018). Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion. Appl. Sci., 8(12),2621. https://doi:10.3390/app8122621.

[32] Wu, G., Cheng, C., Yang, H., & Chena, C. (2017). An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines. Appl. Soft Comput. J., 71, 1072-1084. https://doi: 10.1016/j.asoc.2017.10.015.

[33] Zhang, G., Zhang, L., Song, X., Wang, Y., & Zhou, C. (2018). A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. Cluster Comput., 22, 11561-11572. https://doi: 10.1007/s10586-017-1420-4.

[34] Zorin, D. A., & Kostenko, V. A. (2014). Simulated annealing algorithm in problems of multiprocessor scheduling. Automation and Remote Control, 75, 1790-1801. https://doi: 10.1134/S0005117914100063.

Saeedeh Bakhoda
Orcid number: 0009-0002-4001-9075
Department of Computer Engineering
Miyaneh Branch
Islamic Azad University
Miyaneh, Iran
Email address: saeedeh_ bakhoda@m-iau.ac.ir

Mohammad Abdollahi Azgomi
Orcid number: 0000-0002-9605-8412
School of Computer Engineering
Iran University of Science and Technology
Tehran, Iran
Email address: azgomi@iust.ac.ir

Mohammad Reza Ebrahimi Dishabi
Orcid number: 0000-0003-1963-9560
Department of Computer Engineering
Miyaneh Branch
Islamic Azad University
Miyaneh, Iran
Email address: mrebrahimy@m-iau.ac.ir