Future Generation Computer Systems 141 (2023) 414–432

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Multi-search-routes-based methods for minimizing makespan of homogeneous and heterogeneous resources in Cloud computing

Guangyao Zhou [a], Wenhong Tian [a,*], Rajkumar Buyya [b,a]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China, China
[b] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 17 January 2022
Received in revised form 22 November 2022
Accepted 26 November 2022
Available online 29 November 2022

Keywords:
Cloud computing
Resource scheduling
LPT-OneStep search
BFD-OneStep search
Makespan

ABSTRACT

Cloud computing, as a large-scale distributed computing system dynamically providing elastic services, is designed to meet the requirement of delivering computing services to users as subscription-oriented services. In general, the problems of resource scheduling in Cloud computing like minimizing makespan are usually NP-Hard problems. Various common algorithms including heuristic, meta-heuristic and machine learning are applied in resource scheduling of Cloud computing to obtain the solutions, which however are still probable and imperative to be optimized. Through innovatively applying heuristic algorithms namely LPT (Longest Processing Time) and BFD (Best Fit Decreasing) as the basic search routes and integrating these with neighborhood search algorithm namely OneStep, this paper proposes multi-search-routes-based algorithms containing LPT-OneStep, BFD-OneStep and their combinations for the sake of enhancing theoretical performance and improving solutions of scheduling schemes especially for problems of minimizing makespan for homogeneous and heterogeneous resources. Theoretical derivations prove that the proposed algorithms possess better theoretical approximation ratios for $P||C_{max}$. Extensive experiments on simulation environment demonstrate the proposed algorithms outperform than corresponding compared algorithms for minimizing makespan problems in both homogenous resources and heterogeneous resources, which validates the superiority of the proposed algorithms.

1. Introduction

The emerging trend of Industry 4.0 and 5G significantly enhances the number of tasks that the Internet-based computing systems need to process in real-time. The increase in data size proposes a demand for a large-scale distributed system such as Cloud computing which can provide flexible, reliable and dynamic services to users [1]. Cloud computing, also as a policy to provision high-performance computation services in a pay-as-you-go manner, has supported increasingly complex software systems and computing programs substantially, which has indicated the indispensability of Cloud computing. Currently, many IT companies, such as Amazon, Microsoft, Google, Alibaba and IBM [2–5], have established relatively mature Cloud computing systems. These systems usually have the stable structures such as Software-as-a-Service (SaaS), Platform-as-a-Service(PaaS) and Infrastructure-as-a-Service (IaaS), as well as can flexibly provide services to meet numerous requests from users [6,7].

Resource scheduling is defined by [6] as to find an "optimal" mapping "Tasks → Resources" to meet one or several given objectives. With the enlargement of the Cloud's user groups and facilities, extensive requests and reservations of users are challenging the resource scheduling of Cloud computing. Additionally, inappropriate resource scheduling will cause the waste of users' time, decrease QoS (quality of service), increase energy consumption, and increase carbon dioxide emissions. As the Cloud computing system is still expanding its scale and development of multitudinous industries depends on the reasonable operation of Cloud computing, therefore the research on its resource management is a prominent issue from the birth of Cloud computing to nowadays which will also affect the orientation and prospect of Cloud computing in the society [8,9].

One of the keys to address resource management of Cloud computing is the scheduler lying on the platform layer based on the resource scheduling algorithm. Cloud architecture and resource scheduling process are shown in Fig. 1. The users operate the clients in the application layer to submit task requests to the Cloud center through the high-speed networks of the connection layer; The Cloud center on the platform layer collects tasks, generates scheduling schemes leveraging scheduling algorithms, and allocates tasks to server nodes including VMs and

* Corresponding author.
E-mail addresses: guangyao_zhou@std.uestc.edu.cn (G. Zhou), tian_wenhong@uestc.edu.cn (W. Tian), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.future.2022.11.031
0167-739X/

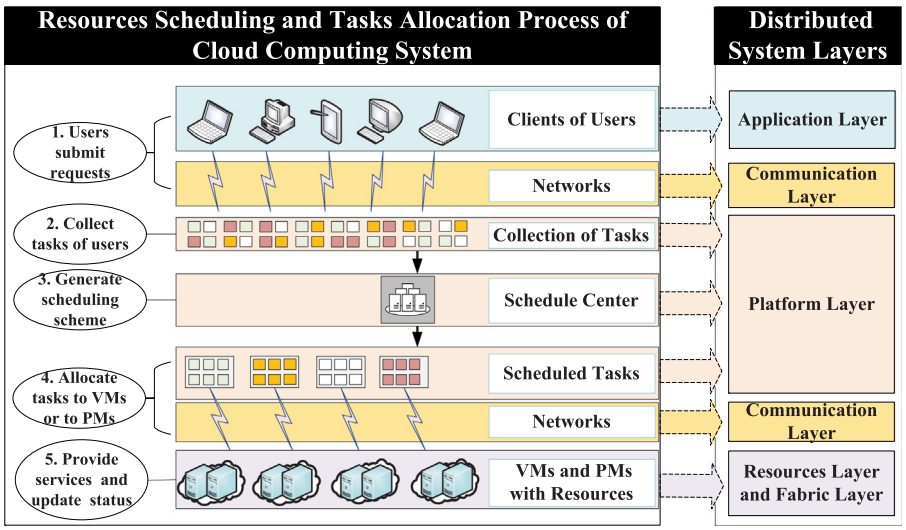


Fig. 1. Cloud architecture with resources scheduling process.

PMs on the resource layer and fabric layer; The server nodes then provide corresponding services to users. In Cloud computing, the scheduling algorithm is a crucial component affecting the quality of resource management. As the resource scheduling problem of Cloud computing is commonly an NP-Hard problem without a feasible method in polynomial time to ensure the global optimization of resource scheduling unless $P = NP$ [1,10,11], hence it is usually settled by heuristic, meta-heuristic, randomization and machine learning algorithms [12]. Some existing algorithms like Ant colony algorithm [13], NSGA II algorithm [14] and deep reinforcement learning algorithm [15,16] have achieved excellent performance in the research results. However, most of them with quite high computational complexity are unstable with randomness, which makes the worst case of scheduling results unpredictable with inevitable risk. For Cloud computing systems in the realistic scenario, some algorithms with determinacy and rapidity such as LPT (Longest Processing Time), FCFS (First Come First Server), RR (Round-Robin), BFD (Best Fit Decreasing), and SPT (Shortest Processing Time) are still practical, whose results nevertheless require to be optimized yet [16,17].

Considering iterative optimization properties of the search algorithm as well as the low complexity and analyzable approximation ratio of heuristic, we apply heuristic algorithms as search routes in the general local search algorithm based on the neighbors of dual resources and propose heuristic-based local search algorithms. For the heuristic-based search algorithm, the selection of the heuristic will directly affect its convergence and optimization performance. Hence, leveraging LPT and BFD as its search routes in light of the better theoretical approximation ratios of LPT and BFD than other heuristic algorithms such as RR, FCFS and SPT to the optimization problems studied in this paper. Based on the heuristic-based search algorithm, we propose multi-search-routes-based local search algorithms combining the One-Step search route (or K-Step search route) and heuristic-based search routes for the instances of minimizing makespan for homogeneous and heterogeneous resources. According to different combinations of search routes, we obtain various multi-search-routes-based algorithms containing LPT-One, BFD-One, LPT-BFD-One, etc, where the combination of multi-search routes is conducive to jumping out of the local optimum of the single search route to improve the performance of algorithms. As the proof of the theoretical approximation ratio of the search algorithm is rare in the previous research, this paper theoretically proves the approximation ratios of several multi-routes local search algorithms for the problem of minimizing makespan in homogenous resources (i.e. the scheduling problem of minimizing makespan on parallel machines abbreviated as $P||C_{max}$), which breaks the dilemma that the search algorithm rarely has approximation ratio proof in previous research. And then, this paper provides experimental results for minimizing makespan of both homogenous and heterogeneous resources to validate the performance of proposed algorithms.

For $P||C_{max}$, the approximation ratios of existing algorithms are as that $Ar_{LPT} \leq \frac{4}{3} - \frac{1}{3M}$, $Ar_{LPT-REV} \leq \frac{4}{3} - \frac{1}{3(M-1)}$ [18,19], $Ar_{Multifit} \leq \frac{72}{61} + 2^{-k}$ where k is the number of attempts to find the smallest number of machines (by binary search) [11]. A principle as PTAS (Polynomial-Time Approximation Scheme) revealed that the guarantee of $Ar \leq 1+\epsilon$ required complexity of $O\left((N/\epsilon)^{1/\epsilon^2}/M\right)$ and no FPTAS (Fully Polynomial-Time Approximation Scheme) exists for $P||C_{max}$ unless $P = NP$ [11]. Additionally, few existing studies proved the approximation ratio of local search algorithms. The approximation ratios of our proposed LPT-One and BFD-One are proved as $\frac{5}{4} - \frac{1}{4M}$ and that of LPT-K and BFD-K are not greater than $1 + \frac{M-1}{(3+K)M}$ for $P||C_{max}$ where M is the number of resources, which reveals the increasing K can ameliorate the upper limit of the approximation ratio even to 1.

The main contributions of this paper are as follows:

(1) Proposing a new framework of local search algorithms based on the neighbors of dual resources and heuristic-based search routes: this paper proposes a framework of the local search algorithm based on the neighbors of dual resources. With neighbors of dual resources, this paper defines various heuristic-based neighborhoods including neighborhoods of LPT, MLPT, BFD and Best-BFD, which correspond to heuristic-based search routes as LPTS, MLPTS, BFDS and BestBFDS respectively. Different from existing algorithms, heuristic-based search routes endowing heuristic algorithms with a novel role can significantly reduce the computational complexity of the local search algorithm and is advantageous to the theoretical derivation of the approximation ratio.

(2) Proposing multi-routes-based search algorithms: to further improve the performance of the scheduling algorithm, this paper proposes the multi-routes-based local search algorithms to resolve minimizing makespan of homogeneous and heterogeneous resources in Cloud based on the combination of heuristic-based search routes and OneStep (K-Step) search route. Different combinations obtain various multi-routes algorithms containing dual routes

search algorithms such as LPT-OneStep search (LPT-One, LPTO), BFD-OneStep search (BFD-One, BFDO) and MLPT-One search, as well as triple routes search algorithm as LPT-BFD-One search, which outperform existing algorithms.

(3) Providing theoretical proofs of approximation ratios for multi-routes-based search algorithms: in theory, it is not easy to prove the approximation ratio of search algorithms. This paper gives theoretical proofs of the approximation ratios for multi-routes-based search algorithms as that $Ar_{LPTO}, Ar_{BFDO} \leq \frac{5}{4} - \frac{1}{4M}$ and $Ar_{LPT-K}, Ar_{BFD-K} \leq 1 + \frac{M-1}{(3+K)M}$ for $P||C_{max}$, which are better than existing algorithms and have significance for the theoretical exploration of search algorithms.

(4) Simulation experiments: this paper executes experiments with abundant instances to verify the performance of the proposed algorithms in both homogeneous and heterogeneous resources.

The remainder of this paper is organized as follows. We review the related works in Section 2. The problem formulation and general local search algorithm are presented in Section 3. Based on various basic search routes, we propose the multi-search-routes-based algorithms and provide theoretical proof of approximation ratios of several specific algorithms in Section 4. The experiment results and discussion are given in Section 5. Finally, we conclude this paper in Section 6.

2. Related work

According to the focus of this paper, we review related work from two aspects, i.e. types of scheduling algorithms and the assumption of system model, to reflect the relationship between this paper and existing research.

2.1. Reviews of scheduling algorithms

To optimize the utilization of resources, energy consumption, response time, makespan, etc, multi-phase method [20,21], virtual machine migration [13,22], queuing model [10], joint optimization [4] and resource scheduling algorithm are frequent strategies used in Cloud computing, where scheduling algorithm is a critical component attracting scholars. Current scheduling algorithms in Cloud computing include local search algorithm, heuristic algorithm, meta-heuristic algorithm, randomization, machine learning algorithm and hybrid algorithm.

Machine Learning in scheduling algorithms mainly contains three types that deep learning (DL) such as DREP [23] and DLSC [24], reinforcement leaning (RL) such as QEEC [25], unified reinforcement learning (URL) [26], adaptive reinforcement learning (ARL) [27] and ADEC [28], as well as Deep reinforcement learning (DRL) such as DRM_Plus [16], A3C RL [29], MDRL [30], DPM [31], DQTS [32] and DQN [15].

A local search algorithm is to select the neighbor solution according to a strategy by comparing the current solution with the neighbor solution, where neighborhood structure and neighborhood selection (search route) are the basic components. In Cloud computing, some local search algorithms, including Neighborhood Search (NS) [33], Correlation-Aware Heuristic Search (CAHS) [34], IBGSS [35], Crow Search [36], Dynamic Grouping Integrated Neighboring Search (DGINS) [37] and Tabu Search [38], are applied to solve the problems of resource management.

The meta-heuristic algorithm is a combination of local search algorithm and randomization containing Ant Colony Optimization (ACO), Genetic Algorithm (GA), FireFly, Particle Swarm Optimization (PSO), etc [12].

ACO imitates ant colony to search for food as a search route. Liu et al. [39] proposed OEMACS combining OEM local search techniques and ACO to resolve VMs deployment in Cloud computing, which reduced the energy consumption and improved the effectiveness of different resources compared with conventional heuristic and other evolutionary-based approaches. Chakravarthy et al. [13] proposed two ant colony-based algorithms (TACO) to address VM scheduling and routing in multi-tenant Cloud data centers aiming at improving the utilization of energy in Cloud computing.

GA imitates the process of natural evolution as the search route of the local search algorithm. Refs. [14,40] improve the search strategy based on NSGA-II to reduce the energy consumption, response time, load imbalance and makespan in Cloud computing. Xu et al. [41] applied NSGA-III to optimize the execution time and energy consumption of IoT-enabled Cloud-edge computing. MOGA proposed by Jiang et al. [42] and MOEAs proposed by Cong et al. [20] improved the search route strategies based on NSGA-II and were utilized to settle resource scheduling in Cloud.

FireFly including FA [43] and FIMPSOA [44], PSO including MOPSO [45], APDPSO [46], and TSPSO [47], are other meta-heuristic algorithms applied in scheduling of Cloud.

For each instance I of a scheduling problem, assuming the solution of an algorithm Al as $Al(I)$ and the theory optimal solution as $OPT(I)$, if $\exists\tau$ (a function) for $\forall I$ **s.t.** $Al(I) \leq \tau(|I|) \cdot OPT(I)$ (or $Al(I) \geq \tau(|I|) \cdot OPT(I)$) and the running time of Al is bounded by a fixed polynomial in $|I|$, then Al is defined as an approximation algorithm with approximation ratio τ [48]. The approximation ratio is a momentous index to evaluate the performance of an algorithm.

The heuristic algorithm is a type of algorithm to solve an optimization problem based on intuitionistic or empirical construction. Currently, some heuristic algorithms in the scheduling of Cloud computing have given approximation ratios and other types of algorithms can rarely obtain the theoretical approximation ratio generally. For the justification of the huge scale of tasks in Cloud computing, higher computational complexity and randomness of meta-heuristic and machine learning, as well as timeliness requirements for processing tasks, heuristic algorithms are still widely applied in practical Cloud.

Zhang et al. [49] adopted Lagrange Relaxation based Aggregated Cost (LARAC) to reduce the energy consumption of Mobile Cloud computing. Dynamic Bipartition-First-Fit (BFF), a $(1+\frac{g-2}{k} - \frac{g-1}{k^2})$ competitive algorithm based on First-Fit algorithm, was proposed and its performance was proved theoretically by Tian et al. [50]. Hong et al. [51] proposed QoS-Aware Distributed Algorithm based on first-come-first-improve (FCFI) and all-come-then-improve (ACTI) algorithms to reduce computation time and energy consumption of Industrial IoT-Edge-Cloud Computing. Tian et al. [52] proposed Longest Loaded Interval First Algorithm (LLIF), a 2-approximation algorithm with theoretical proof, to minimize the energy consumption of VMs reservations in the Cloud. Other algorithms such as RR (Round-Robin) algorithm, greedy, BFD (Best Fit Decreasing), LPT (Longest Processing Time), and Jacobi Best-response Algorithm are frequent algorithms in realistic and have likewise become baselines in existing research [53].

The hybrid of two or more algorithms is also a strategy to improve the search route of the local search algorithm. Kumar et al. [54] proposed a hybrid algorithm called HGA-ACO combining GA and ACO to solve task allocation and ensure QoS parameters in the Cloud environment. Compared with the single search route of GA and ACO, HGA-ACO achieved better performance using the optimization solution of GA as the initial state of ACO [54]. Yang et al. [55] proposed a hybrid meta-heuristic called DAAGA combining GA and improved ACO by taking the solution of ACO as the seed of GA to address Cloud service composition

and the optimization problem. NN-DNSGA-II combining NSGA-II with neural networks [56], PSO-ACS applying PSO to find the optimal solution of task scheduling and applying ACO to find the best migration path of VMs on PMs [57], and FACO exploiting fuzzy module dedicated to pheromone evaluation [58], are also used in Cloud.

2.2. Review of system model

In Cloud Computing, the scheduling scenarios can be divided into dynamic scheduling and static scheduling [1,22].

Dynamic scheduling, usually applied in online scheduling, is generally regarded as more consistent with realistic Cloud computing where tasks usually vary with time and are unknown at the initial scheduling time. In the dynamic scheduling, the tasks are usually predicted with the random process [16,25] or machine learning [31] based on historical data to support the subsequent scheduling using an optimization algorithm, which makes dynamic scheduling rely on two aspects that the prediction models and optimization algorithms. Yuanjun et al. [20] focused on the hybrid tasks in Cloud and proposed multi-phase integrated scheduling with six representative multi-objective evolutionary algorithms. In [20], the orders of tasks are independent without processing constraints among different orders, each task whose processing process is non-preemptive is conducted continuously without interruption, and the resource is free for another step immediately once a processing step is finished. Ding et al. [25] focused on dynamic task scheduling on VMs for energy-efficient cloud computing and proposed a framework QEEC based on Q-learning and M/M/S queueing system. In [25], the energy is modeled as related to tasks' running times, each task is treated as integral which cannot be further split into smaller tasks, a task is conducted by only one VM, and each VM can only execute one task at any time.

Due to the uncertainty of prediction, it is difficult to obtain the theoretical performance of the solution in dynamic scheduling, whose frequently used algorithms are meta-heuristic algorithms [12,39] and machine learning algorithms [16,25,31]. Additionally, after getting the prediction of tasks or the real-time status of resources, dynamic scheduling can be converted to static scheduling by using a static scheduling algorithm as the optimization algorithm of dynamic scheduling. Therefore, there is still a lot of research focusing on static scheduling to explore the theory of scheduling algorithms.

In static scheduling that is usually leveraged in reservation, tasks and resources as known before scheduling and the algorithms aim at finding a scheme to allocate tasks to resources. Tian et al. [59] focused on the load balance of VMs reservations in data centers and proposed "Prepartition" to prepare migration in advance and set process time bound for each VM on a PM. In [59], the system model was modeled as all data are deterministic unless otherwise specified, time is formatted in slotted windows and there is no priority between VMs, which means the orders of VMs do not change their processing times. Zhang et al. [46] focused on the static load balancing in Cloud computing and proposed a novel adaptive Pbest discrete PSO (APDPSO). In [46], only the computation and bandwidth resources are considered, the network topology of hosts is deterministic, the unit data transmitting cost between each host pair is fairly unchanged, the constraints of resources and the required resources of tasks or VMs are known, and the amount of data to be transferred between federates is unchanged. Ghalami et al. [11] focused on the scheduling jobs on parallel identical machines to minimize makespan ($P||C_{max}$) and developed sequential approximation algorithms with various approximation guarantees. In modeling of [11], the processing times of jobs were available for processing at time zero, a job cannot be preempted once assigned to a processor for execution, and each machine cannot process more than one job at a time.

Table 1
Notations and descriptions.

Notation	Description	Nature
i	Index of task	Object
j	Index of resource	
T_i	The task with index i	
R_j	The resource with index j	
card(S)	The cardinal of set S	
N	Number of tasks	Given
M	Number of resources	
E_{ij}	General element of task T_i when executed in R_j	
ET_{ij}	The processing time of T_i when executed in R_j	
$MaxA_j$	General Upper limit of resource R_j	
TS_j	Set of tasks in resources R_j	Scheme-related
A_j	General aspect of resource R_j	
ω	General aspect of problem	
RT_j	The total processing time of resource R_j	
P_i	The parameter set of task T_i	
KS	The set of TS_j where $KS = \{TS_1, \ldots, TS_M\}$	

In this paper, we continue some assumptions about Cloud computing system modeling referring to reviewed literature to formulate the static scheduling of minimizing makespan in homogeneous and heterogeneous resources, which makes the improvement of the theoretical performance of our proposed algorithm in the minimizing makespan problem to be useful for resource scheduling of Cloud computing. According to the review of types of scheduling algorithms, it can be seen that our proposed multi-routes search algorithms using heuristic algorithms as search routes are distinct from the previous algorithms. For multi-routes search algorithms with a similar principle to hybrid algorithms, different search routes can jump out the inherent local optimum of a single search route resulting in optimization of the solution. Application of heuristic algorithm as search route enables the theoretical proof of approximate ratio of the search algorithm, which has not been carried out for search algorithms in the previous studies.

3. Problems formulation and general local search

In this section, we model the universal scheduling problem of Cloud computing and present objectives for minimizing makespan both for homogeneous and heterogeneous resources. And then, we establish the framework of a general local search algorithm based on the adjustment of tasks between two resources. The notations used in this paper are presented in Table 1, where the given parameters are known without the influence of scheme and scheme-related variables are the opposite.

3.1. Models of minimizing makespan in cloud computing

The resource in the problems of this paper refers to a physical machine or virtual machine that can process tasks with some specific component such as CPU, RAM, DS, etc, where task refers to a request from the user. In this paper, we mainly focus on the static scheduling and leverage the processing time to express a task that $T_i = \{\langle ET_{ij} \rangle\} = \langle ET_{i1}, ET_{i2}, \ldots, ET_{iM} \rangle$. For minimizing makespan, the features of resources and tasks are as follows referring to existing research.

(1) The set of tasks $\{T_1, T_2, \ldots, T_N\}$ is deterministic [59];
(2) All tasks are independent and preemptive without precedence constraints for the order of tasks [59];
(3) Each task is treated as an integral task and cannot be further split into smaller tasks [25];

(4) Each task can be fully fulfilled by one and only one resource (usually virtual machine) [25];
(5) When $A_j \leq MaxA_j$, the processing capacity of the resource remains unchanged, which means the parameter P_i of each task is fixed and unaffected by the status of the resource.
(6) The total processing time of all resources starts from 0;
(7) Each resource (i.e. VM or PM) is either idle or processing only one task [11,25];
(8) Once a task is finished, the occupied resource is free for another task immediately ignoring switching time [20];
(9) The processing time ET_{ij} is fixed without affection of the resource's status or the execution order of tasks;
(10) And the number of available resources is invariant.

Assuming the mapping between general element of task and general aspect of resources as function $h : E \rightarrow A$, the mapping between general aspect of resources and optimization objective as function $f : A \rightarrow \omega$, and taking $P_i = \langle E_{i1}, E_{i2}, \ldots, E_{iM} \rangle$, the universal resource scheduling problem of Cloud computing can be written as

$$\min \omega = f(A_1, A_2, A_3, \ldots, A_M) \tag{1}$$

where $A_j = h_j\left(E_{ij}|_{T_i \in TS_j}\right)$ subject to $A_j \leq MaxA_j$. In Cloud computing, the general element of task can be taken as processing time, volume, energy consumption, bandwidth, storage request and other elements of tasks, in addition the general aspect of resource can be taken as degree of load balance [20,43], makespan [32,43], energy consumption [13,30], cost [45,53], delay ratio [20], utilization of resource [25,43], throughput [24,44], SLA Violation [28,45], etc.

If $\forall l \neq k$ **s.t.** $h_l(x) = h_k(x)$, $MaxA_l = MaxA_k$ and $E_{il} = E_{ik}$, the resources are homogeneous, otherwise the resources are heterogeneous. In this paper, we consider homogeneous resources when studying the theoretical approximation ratio of the algorithms, while we also study the scheduling problems of heterogeneous resources to approach to real Cloud scenarios.

Solution of minimizing makespan is a way to reduce the working time of resources, where less working time may bring multifaceted benefits such as reduction of energy consumption, increase of resource utilization, prolong of devices' lifespan, improvement of processing capacity, etc. For problem of minimizing makespan, the decisive factors are time-related parameters including ET_{ij} and RT_j assuming time is slotted in the unit of time. Thus, we need to use ET_{ij} and RT_j to replace E_{ij} and A_j in problem then substitute the parameter set of tasks as $P_i = \langle ET_{i1}, ET_{i2}, \ldots, ET_{iM} \rangle$ where $ET_{ij} \in \mathbb{R}^+$ means the processing time of task T_i when it on the resource R_j.

According to the features of tasks and resources, the total processing time of R_j equals to the sum of processing time of all tasks executed on R_j that

$$RT_j = \sum_{T_i \in TS_j} ET_{ij}. \tag{2}$$

Along with minimizing makespan, the sum of the running time of all resources is considerable simultaneously. Therefore, two objectives that are minimizing the total running time and minimizing the makespan require considerations shown as

$$\begin{cases} \min \omega_{total-time} = \min \left(\sum_{j=1}^{M} \sum_{i=1}^{N} x_{ij} ET_{ij} \right) \\ \min \omega_{makespan} = \min \left(\max_{j=1,2,\ldots,M} \left(\sum_{i=1}^{N} x_{ij} ET_{ij} \right) \right) \end{cases}. \tag{3}$$

For heterogeneous resources, Eq. (3) is a bi-objective problem, and for homogeneous resources, total running time is a constant where Eq. (3) can degenerate into a single objective optimization problem. The constraints of Eq. (3) are subject to

$$\sum_{j=1}^{M} x_{ij} = 1, \forall i \in \{1, \ldots, N\}, \tag{4a}$$

$$x_{ij} \in \{0, 1\}, \forall i \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, M\}, \tag{4b}$$

$$A_j \leq MaxA_j, \forall j \in \{1, \ldots, M\}, \tag{4c}$$

where Eqs. (4a) and (4b) are the common constraints of zero–one integer programming, as well as Eq. (4c) is corresponding to the fifth features of system to ensure the parameters of tasks and resources unchanged.

In Eq. (3), the optimization solution of $\{x_{ij}\}$ necessarily and sufficiently occupies a unique set of KS, hence KS can express the solution corresponding to $\{x_{ij}\}$. When KS is determined, other scheme-related parameters in Table 1 can be calculated according to given parameters. Other scheduling problems can be formulated similar to these problems based on the universal mode of Eq. (1). Then, the formulated problems considered in this paper are in Table 2.

Table 2
Problems executed in experiments.

Sign	Description of problem
P_1	Minimizing makespan for homogenous resources
P_2	Minimizing makespan and total running time for heterogenous resources

3.2. General local search algorithm

Although the main problem studied in this paper is minimizing makespan, our proposed algorithms have generality for other optimization problems, such as load balancing, minimizing energy consumption, etc. Therefore, we still use the general parameters E_{ij} and A_j to present subsequent methodologies and theoretical proofs. When solving the makespan-related problems, they only need to be replaced by ET_{ij} and RT_j.

For scheduling problems, the local search algorithm is a considerable method, whose strategy is to search the optimal state by local neighborhoods [33,34]. In this paper, neighbors of dual resources is described as only two resources have different sets of tasks in two scheduling solutions, which can be defined by mathematical formulas as: assuming two solutions $KS_1 = \{TS_1, TS_2, \ldots, TS_M\}$ and $KS_2 = \{TS'_1, TS'_2, \ldots, TS'_M\}$, if $\exists j_1 \neq j_2 \in \{1, 2, \ldots, M\}$ **s.t.** $TS_{j_1} \neq TS'_{j_1}$, $TS_{j_2} \neq TS'_{j_2}$ and $TS_k = TS'_k$ for $\forall k \in \{1, 2, \ldots, M\} - \{j_1, j_2\}$, then KS_1 and KS_2 are neighbors of dual resources.

General Local Search Algorithm based on the neighbors of dual resources for the universal scheduling problem, utilizes a specified route to search for a better state in the neighbors of the current state until non-existent neighbors are better than the current state as Algorithm 1. General Local Search Algorithm based on the Neighbors of Dual Resources has four essential fundamentals as follows which vary with scenarios:

(1) The specified local search route which be substituted by various route strategies;
(2) The neighbor selection criteria consistent with $f : A \rightarrow \omega$;
(3) The initial state with initialization policy that determines which local optimal cluster to search in;
(4) The specific order strategies.

In these four essential fundamentals, the specified local search route directly influences the performance of the solution and is also a widely studied topic. This paper focuses on this to propose multi-search-routes-based algorithms.

Algorithm 1: General Local Search Algorithm based on the Neighbors of Dual Resources

Input : $\{T_1, T_2, \ldots, T_N\}$ and $\{R_1, R_2, \ldots, R_M\}$
Output: solution $KS = \{TS_1, TS_2, \ldots, TS_M\}$

Initially Allocate tasks to resources with **confirmed or random initialization policy** and gain the general aspect of resources by $A_j = h_j\left(E_{ij}|_{T_i \in TS_j}\right)$
while Exist_adjustment **do**
  Exist_adjustment = False
  Sort resources by their value A_j
  for j_1 *in resources of a* ***specific order strategies*** **do**
    $or\_j = j_1$, $\beta = f(A_1, A_2, \ldots, A_M)$
    for j_2 *in* $[0, j_1)$ **do**
      Use **specified local search route** (one or more of LPTS, BFDS, K-Step etc.) to adjust tasks belonging to TS_{j_1} and TS_{j_2} then gain TS'_{j_1} and TS'_{j_2} subject to $A_j \leq MaxA_j$ for $\forall j \in \{1, 2, \ldots, M\}$
      if $\beta > f\left(A_1, \ldots, A'_{j_2}, \ldots, A'_{j_1}, \ldots, A_M\right)$ **then**
        $pre\_j = j_2$ and Exist_adjustment = True
        $\beta = f\left(A_1, \ldots, A'_{j_2}, \ldots, A'_{j_1}, \ldots, A_M\right)$
    if Exist_adjustment **then**
      Update resources of or_j and pre_j based on the **specified local search route** as $TS_{or\_j} = TS'_{or\_j}$, $TS_{pre\_j} = TS'_{pre\_j}$
      Break (for repeat of j_1)

4. Multi search routes-based algorithm

4.1. Specified basic local search route

It can be seen from the flow of Algorithm 1 that any algorithm able to readjust the tasks of two resources can be applied as its **specified local search route** and the property of its convergence solution is affected by this route. This property of Algorithm 1 not only allows a heuristic algorithm to be a search route but also allows multiple search routes to be used simultaneously to jump out of the convergence points of a single search route. Based on the definition of different neighborhoods, we will present several specified local search routes as basic routes including K-Step search, LPT search and BFD search. Substituting these algorithms into Algorithm 1 as **specified local search route**, we can get various local search algorithms.

4.1.1. K-Step search route

On the basis of neighborhood of dual resources, neighbors of K-Step can be defined as: assuming KS and KS' of two states of solutions satisfy the neighborhood of dual resources, if $\mathbf{card}\left(TS_{j_1} - TS_{j_1} \cap TS'_{j_1}\right) \leq K$ and $\mathbf{card}\left(TS_{j_2} - TS_{j_2} \cap TS'_{j_2}\right) \leq K$, then KS and KS' are mutually neighbors of K-Step. The K-Step search route based on neighbors of K-Step is Algorithm 2.

With the adaptability of objectives, K-Step search can improve the current state of the solution. When $K = 1$, we can obtain One-Step search, which will be applied in our multi-route search subsequently as a special case of K-Step search. Generally, the computational complexity of finding the tasks sets B_{j_1} and B_{j_2} in One-Step search is $\left(card(TS_{j_1}) + card(TS_{j_2}) + card(TS_{j_1})card(TS_{j_2})\right)$. In our proposed One-Step search, we use a sort-based algorithm to optimize the process of finding task sets, which can reduce the computational complexity of finding the tasks sets to $\left(\left(card(TS_{j_1}) + card(TS_{j_2})\right)\log\left(card(TS_{j_1}) + card(TS_{j_2})\right)\right)$. And then, we obtain the algorithm of One-Step search route based on sort algorithm as Algorithm 3.

Algorithm 2: K-step search route

Input : tasks set TS_{j_1} and TS_{j_2} of R_{j_1} and R_{j_2}
Output: TS'_{j_1} and TS'_{j_2}

Find tasks set $B_{j_1} \subset TS_{j_1}$ and tasks set $B_{j_2} \subset TS_{j_2}$ **s.t.** $f\left(A'_{j_1}, A'_{j_2}\right) < f\left(A_{j_1}, A_{j_2}\right)$ where $A'_j = h_j\left(E_{ij}|_{T_i \in TS'_j}\right)$, $0 < \max\left(\mathbf{card}(B_{j_1}), \mathbf{card}(B_{j_2})\right) \leq K$, $TS'_{j_1} = TS_{j_1} - B_{j_1} + B_{j_2}$ and $TS'_{j_2} = TS_{j_2} + B_{j_1} - B_{j_2}$

Algorithm 3: One-step search route

Input : tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of R_{j_1} and R_{j_2} where it can be set as $A_{j_1} \geq A_{j_2}$
Output: TS'_{j_1} and TS'_{j_2}

Set$\gamma_{j_1} = \frac{\sum_{T_i \in TS} E_{ij_1}}{M}$, $C_{j_1} = A_{j_1} - \gamma_{j_1}$
$B_1 = \arg\min_{\{T_i\}}\left(\left|E_{ij_1} - C_{j_1}\right|_{T_i \in TS_{j_1}}, \left|E_{ij_2} + C_{j_1}\right|_{T_i \in TS_{j_2}}\right)$
$vl_1 = \min\left(\left|E_{ij_1} - C_{j_1}\right|_{T_i \in TS_{j_1}}, \left|E_{ij_2} + C_{j_1}\right|_{T_i \in TS_{j_2}}\right)$
Sort $\left\{E_{ij_1}|_{T_i \in TS_{j_1}}, E_{ij_2} + C_{j_1}|_{T_i \in TS_{j_2}}\right\} \rightarrow \{G_{\eta_1}, G_{\eta_2}, \ldots\}$
$B_2 = \arg\min_{\{T_{\eta_i}, T_{\eta_{i+1}}\}}\left|G_{\eta_i} - G_{\eta_{i+1}}\right|$, $vl_2 = \min\left|G_{\eta_i} - G_{\eta_{i+1}}\right|$ where $\{T_{\eta_i}, T_{\eta_{i+1}}\} \not\subset TS_{j_1}$ and $\{T_{\eta_i}, T_{\eta_{i+1}}\} \cap TS_{j_1} \neq \emptyset$
if $B_1 = \emptyset$ *and* $B_2 = \emptyset$ **then**
  Return Exist_adjustment = False
else
  if $vl_1 \leq vl_2$ **then**
    Update by $TS'_{j_1} = TS_{j_1} - B_1$, $TS'_{j_2} = TS_{j_2} - B_1$
  else
    Update by $TS'_{j_1} = TS_{j_1} \cup B_2 - TS_{j_1} \cap B_2$, $TS'_{j_2} = TS_{j_2} \cup B_2 - TS_{j_2} \cap B_2$
  Return TS'_{j_1}, TS'_{j_2} and Exist_adjustment = True

Substitution of K-Step search route to the **specified local search route** of Algorithm 1 (i.e. General Local Search Algorithm) can obtain K-Step Search Algorithm. This paper mainly applies One-Step to improve other search routes such as LPTS and BFDS presented subsequently.

For the sake of subsequently demonstrating proofs of approximate ratio, we can present the property of K-Step in Property 1 according to the definition of K-Step neighbors, which also applies to One-Step when $K = 1$.

Property 1 (K-*Step*). *Assume* $KS = \{TS_1, \ldots, TS_M\}$ *is the convergence solution of K-Step Search Algorithm. For* $\forall j_1 \neq j_2$, $\forall B_\alpha \subset T_{j_1}$, $\forall B_\beta \subset T_{j_2}$, *if* $0 < \max\left(\mathbf{card}(B_\alpha), \mathbf{card}(B_\beta)\right) \leq K$, *then:* $f\left(A'_{j_1}, A'_{j_2}\right) \geq f\left(A_{j_1}, A_{j_2}\right)$ *where* $A'_j = h_j\left(E_{ij}|_{T_i \in TS'_j}\right)$, $TS'_{j_1} = TS_{j_1} - B_\alpha + B_\beta$ *and* $TS'_{j_2} = TS_{j_2} + B_\alpha - B_\beta$.

4.1.2. LPT search route and modified LPT search route

LPT (Longest Processing Time) algorithm is proposed to solve minimizing makespan of parallel machines ($P||C_{max}$) [11]. Currently, LPT algorithm has approximation ratio $Ar_{LPT} \leq \frac{4}{3} - \frac{1}{3M}$ where $M \geq 2$ and LPT-REV has approximate ratio $Ar_{LPT-REV} \leq \frac{4}{3} - \frac{1}{3(M-1)}$ where $M \geq 3$ in $P||C_{max}$ [18,19]. Considering the merit of LPT, this paper applies LPT as the search route shown as Algorithm 4 by adjusting the classic LPT algorithm so that it

can be applied to Algorithm 1 as the search route. Based on the characteristics of LPT algorithm, neighborhoods of LPT-route can be defined as: assuming KS' and KS are the neighbors of dual resources and $TS'_{j_1} \cup TS'_{j_2} = TS_{j_1} \cup TS_{j_2} = \{T_{\alpha_1}, T_{\alpha_2}, \dots\}$ where $E_{\alpha_i} \geq E_{\alpha_{i+1}}$, if $T_{\alpha_i} \in \arg\min_{TS'} \left(\sum_{T_{\alpha_k} \in TS'_1} E_{\alpha_k}, \sum_{T_{\alpha_k} \in TS'_2} E_{\alpha_k}\right)$ where $k < i$ for $\forall T_{\alpha_i} \in TS'_{j_1} \cup TS'_{j_2}$ then KS' is an LPT route-based neighbor of KS, while by contraries KS may not be that of KS'.

Algorithm 4: LPT search route (based on LPT)

Input : tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of R_{j_1} and R_{j_2}
Output: TS'_{j_1} and TS'_{j_2}

$Mark_{j_1} = 0$, $Mark_{j_2} = 0$, $TS'_{j_1} = \emptyset$ and $TS'_{j_2} = \emptyset$
for T_α *in TS from largest to smallest* **do**
  if $Mark_{j_1} \leq Mark_{j_2}$ **then**
    $Mark_{j_1} += E_{\alpha j_1}$ and $TS'_{j_1} += \{T_\alpha\}$
  else
    $Mark_{j_2} += E_{\alpha j_2}$ and $TS'_{j_2} += \{T_\alpha\}$

Substitution of LPT search route to the **specified local search route** of Algorithm 1 can obtain LPT Search Algorithm (LPTS) that can adapt to various objectives corresponding to balance. LPTS inherits the approximation ratio $Ar_{LPTS} \leq \frac{4}{3} - \frac{1}{3M}$ for $P||C_{max}$ of LPT algorithm, and has better solutions in statistics, which means the solution of LPTS is not inferior to that of LPT algorithm.

Similarly for the demonstration of subsequent proofs, we present a property of LPTS as Property 2 according to the definition of LPT route-based neighbor.

Property 2 (*LPTS*). *$KS = \{TS_1, \dots, TS_M\}$ is the convergence solution of LPTS. For $\forall j_1 \neq j_2$, assuming $TS_{j_1} \cup TS_{j_2} = \{T_{\alpha_1}, T_{\alpha_2}, \dots\}$ where $E_{\alpha_i j_1} \geq E_{\alpha_{i+1} j_1}$, if $TS'_{j_1} \cup TS'_{j_2} = TS_{j_1} \cup TS_{j_2}$ and $T_{\alpha_i} \in \arg\min_{TS'} \left(\sum_{T_{\alpha_k} \in TS'_{j_1}} E_{\alpha_k j_1}, \sum_{T_{\alpha_k} \in TS'_{j_2}} E_{\alpha_k j_2}\right)_{k<i}$ for $\forall T_{\alpha_i} \in \{T_{\alpha_1}, T_{\alpha_2}, \dots\}$, then: $f\left(A'_{j_1}, A'_{j_2}\right) \geq f\left(A_{j_1}, A_{j_2}\right)$.*

However, the LPT algorithm is originally intended to resolve the problems in homogenous resources. For heterogeneous resources, a Modified LPT search (MLPT) algorithm seen in Algorithm 5 can be applied to address the problem of minimizing makespan. The Modified LPT search route considers the difference in the processing time or volume of a task between different resources and preferentially puts the task with the largest difference into a specific resource. In this way with adaptability for heterogeneous resources, the local search algorithm using MLPT route can obtain an optimized solution.

Algorithm 5: Modified LPT search route for heterogenous resources (Modification of LPT route)

Input : tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of R_{j_1} and R_{j_2}
Output: TS'_{j_1} and TS'_{j_2}

$Mark_{j_1} = 0$, $Mark_{j_2} = 0$, $TS'_{j_1} = \emptyset$ and $TS'_{j_2} = \emptyset$
while $TS \neq \emptyset$ **do**
  if $Mark_{j_1} \leq Mark_{j_2}$ **then**
    $c = j_1$, $b = j_2$
  else
    $c = j_2$, $b = j_1$
  Find $T_{\alpha_l} \in TS$ **s.t.** $T_{\alpha_l} = \arg\min_{T_i \in TS} (E_{ic} - E_{ib})$ to obtain a set of $\{T_{\alpha_1}, T_{\alpha_2}, \dots, T_{\alpha_s}\}$
  if $s \geq 2$ **then**
    Choose T_α **s.t.** $T_\alpha = \arg\max_{1 \leq l \leq s} E_{\alpha_l c}$
  $Mark_c += E_{\alpha c}$, $TS'_c += \{T_\alpha\}$ and $TS -= \{T_\alpha\}$

Algorithm 5 is a version convenient for comprehension. In realistic program of algorithm, we can use array operations on GPU to accelerate Algorithm 5. For the tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of two resources R_{j_1} and R_{j_2}, we can assume the set sorted in ascending order according to the value of $E_{ij_1} - E_{ij_2}$ in TS as $\{T_{\alpha_1}, T_{\alpha_2}, \dots, T_{\alpha_n}\}$. Then, the operation of Algorithm 5 equals to finding an index ζ in $\{0, 1, 2, \dots, n\}$ **s.t.** $\sum_{i=0}^{\zeta} E_{\alpha_i j_1}$ and $\sum_{i=\zeta+1}^{n+1} E_{\alpha_i j_2}$ as close as possible assuming $E_{\alpha_0 j} = E_{\alpha_{n+1} j} = 0$ where $j \in \{j_1, j_2\}$. And $\sum_{i=0}^{\zeta} E_{\alpha_i j_1} \approx \sum_{i=\zeta+1}^{n+1} E_{\alpha_i j_2}$ is equivalent to $\sum_{i=0}^{\zeta} E_{\alpha_i j_1} + \sum_{i=0}^{\zeta} E_{\alpha_i j_2} \approx \sum_{i=\zeta+1}^{n+1} E_{\alpha_i j_2} + \sum_{i=0}^{\zeta} E_{\alpha_i j_2} \rightarrow \sum_{i=0}^{\zeta} \left(E_{\alpha_i j_1} + E_{\alpha_i j_2}\right) \approx \sum_{i=0}^{n+1} E_{\alpha_i j_2} = Sum_{j_2}$. Setting $S_{j_1} = \{E_{\alpha_0 j_1}, E_{\alpha_1 j_1}, \dots, E_{\alpha_n j_1}\}$ and $S_{j_2} = \{E_{\alpha_0 j_2}, E_{\alpha_1 j_2}, \dots, E_{\alpha_n j_2}\}$, therefore, we can use the GPU-based program to quickly calculate a new array as $\left|cusum\left(S_{j_1} + S_{j_2}\right) - Sum_{j_2}\right|$ where $cusum(S)$ means the cumulative sum of S, choose the index at its minimum as ζ, and update $TS'_{j_1} = \{T_{\alpha_1}, T_{\alpha_2}, \dots, T_{\alpha_\zeta}\}$ and $TS'_{j_2} = TS - TS'_{j_1}$ where if $\zeta = 0$ then $TS'_{j_1} = \emptyset$. Using this idea, we can quickly adjust the tasks of two resources with GPU operation-based program, and even quickly adjust the tasks of multiple resources simultaneously. Similarly, other search routes can also be accelerated by GPU operation. As this is not the focus of this paper, we do not expand the explanation.

4.1.3. BFD search route

BFD (Best Fit Decreasing) is usually used to solve the bin packing problem and also applies to problems related to minimizing makespan. A search route based on BFD is Algorithm 6 and the neighborhood using the BFD search route can be called BFD route-based neighborhood similar to the definition of LPT route-based neighborhood. Using BFD search route to replace the **specified local search route** in Algorithm 1 obtains BFD search (BFDS) algorithm. Similar to the relationship between LPTS and LPT, BFDS, with a better statistical performance, inherits the approximation ratio of BFD. A property of BFDS is Property 3 according to its definition.

Algorithm 6: BFD search route (based on BFD)

Input : tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of R_{j_1} and R_{j_2}
Output: TS'_{j_1} and TS'_{j_2}

set $\gamma_j = \frac{\sum_{T_i \in TS} E_{ij}}{M}$ where $j \in \{j_1, j_2\}$
$Mark_{j_1} = 0$, $Mark_{j_2} = 0$, $TS'_{j_1} = \emptyset$, $TS'_{j_2} = \emptyset$
for T_α *in TS from largest to smallest* **do**
  if $\left|Mark_{j_1} + E_{ij_1} - \gamma_{j_1}\right| < \left|Mark_{j_2} + E_{ij_2} - \gamma_{j_2}\right|$ **then**
    $TS'_{j_1} += T_\alpha$, $Mark_{j_1} += E_{ij_1}$
  else
    $TS'_{j_2} += T_\alpha$, $Mark_{j_2} += E_{ij_2}$

Property 3 (*BFDS*). *$KS = \{TS_1, \dots, TS_M\}$ is the convergence solution of BFDS. For $\forall j_1 \neq j_2$, assuming $TS_{j_1} \cup TS_{j_2} = \{T_{\alpha_1}, T_{\alpha_2}, \dots\}$ where $E_{\alpha_i j_1} \geq E_{\alpha_{i+1} j_1}$, if $TS'_{j_1} \cup TS'_{j_2} = TS_{j_1} \cup TS_{j_2}$ and $T_{\alpha_i} \in \arg\min_{TS'} \left(\left|Q_{ij_1} + E_{\alpha_i j_1} - \gamma_{j_1}\right|, \left|Q_{ij_2} + E_{\alpha_i j_2} - \gamma_{j_2}\right|\right)$ for $\forall T_{\alpha_i} \in TS'_{j_1} \cup TS'_{j_2}$ where $Q_{ij_1} = \sum_{T_{\alpha_k} \in TS'_{j_1}, k<i} E_{\alpha_k j_1}$ and $Q_{ij_2} = \sum_{T_{\alpha_k} \in TS'_{j_2}, k<i} E_{\alpha_k j_2}$, then: $f\left(A'_{j_1}, A'_{j_2}\right) \geq f\left(A_{j_1}, A_{j_2}\right)$.*

An improved strategy of the BFD search route, which is called Best-BFD search route in Algorithm 7, is to record schemes and select the best scheme as the final solution.

4.2. Combination of multi-routes and the flowchart

As mentioned above, the specified local search route of Algorithm 1 can be replaced by multiple search routes simultaneously. Therefore, the proposed search routes can be arbitrarily combined as the search routes of Algorithm 1. This paper presents three

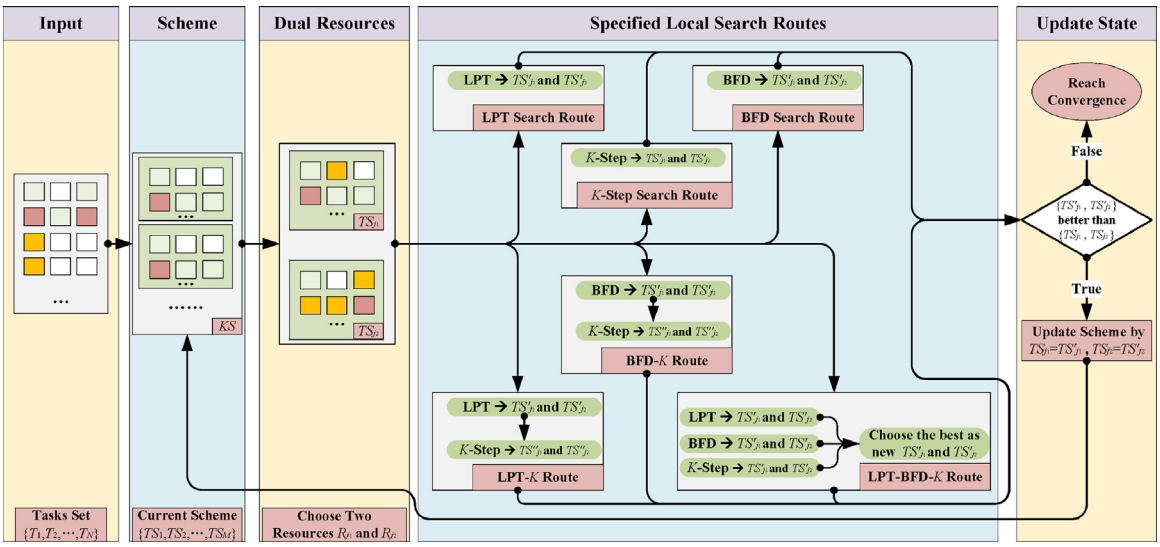


Fig. 2. Flowchart of local search algorithms based on the neighbors of dual resources with various search routes including LPT, BFD, K-Step and their combinations.

Algorithm 7: Best BFD search route

Input : tasks set $TS = TS_{j_1} \cup TS_{j_2}$ of R_{j_1} and R_{j_2}
Output: TS'_{j_1} and TS'_{j_2}

set $\gamma_{j_1} = \frac{\sum_{T_i \in TS} E_{ij_1}}{M}$, $k = 0$, $Mark_{j_1} = 0$, $TS^{(0)}_{j_1} = \emptyset$
for T_α *in TS from largest to smallest* **do**
  if $Mark_{j_1} + E_{ij_1} \le \gamma_{j_1}$ **then**
    $TS^{(0)}_{j_1} += T_\alpha$ and $Mark_{j_1} += E_{ij_1}$
  else
    $k++$
    $L_k = \left|Mark_{j_1} + E_{ij_1} - \gamma_{j_1}\right|$ and $TS^{(k)}_{j_1} = TS^{(0)}_{j_1}$
$L_0 = \left|Mark_{j_1} - \gamma_{j_1}\right|$
$TS'_{j_1} = \arg\min_{TS^{(\alpha)}_{j_1}} (L_0, L_1, \ldots, L_k)$, $TS'_{j_2} = TS - TS'_{j_1}$

combinations of them including LPT-One, BFD-One and LPT-BFD-One regarding LPT route, One-Step route and BFD route as the basic routes. These algorithms using various search routes can be uniformly called multi-search-routes-based algorithms and their operation processes corresponding to the specified search route in Algorithm 1 are as follows:

(1) **LPT-One** : $\left(TS_{j_1}, TS_{j_2}\right) \rightarrow \left(TS'_{j_1}, TS'_{j_2}\right)$ by LPTS and $\left(TS'_{j_1}, TS'_{j_2}\right) \rightarrow \left(TS''_{j_1}, TS''_{j_2}\right)$ by OneS; use (TS''_{j_1}, TS''_{j_2}) as the neighbor of (TS_j, TS_k) in Algorithm 1.
(2) **BFD-One**: $\left(TS_{j_1}, TS_{j_2}\right) \rightarrow \left(TS'_{j_1}, TS'_{j_2}\right)$ by BFDS and $\left(TS'_{j_1}, TS'_{j_2}\right) \rightarrow \left(TS''_{j_1}, TS''_{j_2}\right)$ by OneS; use (TS''_{j_1}, TS''_{j_2}) as the neighbor of (TS_{j_1}, TS_{j_2}) in Algorithm 1.
(3) **LPT-BFD-One**: Use LPTS, BFDS and OneS, then choose the best solution as the neighbor of (TS_{j_1}, TS_{j_2}).

Based on the above description, we can draw the flow charts of our proposed algorithms as Fig. 2 to intuitively demonstrate how the LPT, BFD, One-Step and their combinations act as search routes of Algorithm 1. From Fig. 2, we can see again the flexibility of Algorithm 1 that it allows various algorithms as its search routes. In the framework, we only need to change the search routes to obtain specific search algorithms with different convergence properties.

As LPT and BFD have increased performance and One-Step has a descending performance with the increasing number of tasks, therefore LPT-OneStep search (LPT-One) and BFD-OneStep search (BFD-One) occupy better approximation ratios respectively than LPTS and BFDS. Multi-routes search has various combination patterns of search routes where one is the combination with repeated iterations of search routes such as LPT_route+OneStep_route, as well as the other is the combination with repeated iterations of search algorithms such as LPT_algorithm+OneStep_algorithm. Theoretically, the two patterns have the same theoretical approximation ratio. In this paper, LPT-One and BFD-One appertain to that of search routes.

4.3. Theoretical proofs of approximation ratios

In consideration of the complexity and difficulty to prove the approximate ratio of heterogeneous resources, we only demonstrate the proofs of approximation ratio for homogeneous resources i.e. mainly focusing on $P||C_{max}$. Since the given general upper limits $MaxA_j$ of resources are the same in homogenous resources, which means the constraint is only related to the maximum aspects $\max\{A_1, A_2, \ldots, A_M\}$, so we do not need to consider the constraint of Eq. (3) in the proof.

As LPT-One and BFD-One are originated from the heuristic algorithms which have theoretical approximation ratios, hence their approximation ratios can be proved by referring to their original heuristic algorithms. Considering $E_{i1} = E_{i2} = \cdots = E_{iM}$ for $\forall i$ in homogeneous resources, we donate $G_i = E_{i1} = E_{i2} = \cdots = E_{iM}$ to represent the general elements of task T_i for the sake of demonstration of proofs. Next, we present the approximation ratios of several algorithms and their proofs.

Theorem 1. $Ar_{LPTO} = Ar_{LO} \le \frac{5}{4} - \frac{1}{4M}$ *for* $P||C_{max}$.

Proof. According to the convergence condition, the convergence solution of LPT-One simultaneously obeys Properties 1 and 2. Suppose the set of instances that do not satisfy Theorem 1 is H, the instance $I = \{T_1, T_2, \ldots, T_n\} \in H$ has minimum numbers of tasks, and $G_1 \ge G_2 \ge \cdots \ge G_n$, which means $Ar_{LO}\,(x|\forall x \in H) > \frac{5}{4} - \frac{1}{4M}$ and $\mathbf{card}(I) = \min \mathbf{card}\,(x|\forall x \in H)$.

Then, $T_n \in \arg\max_{TS_j}\left(A_j\right)$, otherwise $\exists I' = I - \{T_n\}$ **s.t.** $Ar_{LO}(I') > \frac{5}{4} - \frac{1}{4M}$ hence $I' \in H$ and $\mathbf{card}(I') < \mathbf{card}(I)$, which is contradicted to $\mathbf{card}(I) = \min \mathbf{card}\,(x|\forall x \in H)$.

Assuming $T_n \in TS_k$, when LPTS converges, $A_k - G_n \leq A_j$ for $\forall j \neq k$ from Property 2. Thus, $A_k \leq \frac{1}{M}\sum_{i=1}^{n} G_i + \frac{M-1}{M}G_n \leq OPT(I) + \frac{M-1}{M}G_n$ where $OPT(I)$ is the theoretical optimization.

As $Ar_{LO}(I) > \frac{5}{4} - \frac{1}{4M}$, so $\frac{5}{4} - \frac{1}{4M} < 1 + \frac{M-1}{M}\frac{G_n}{OPT(I)}$. $\therefore OPT(I) < 4E_n$. $\because G_n \leq G_i$ for $\forall i = 1, 2, \ldots, n$, $\therefore$ $\mathbf{card}(TS'_j) \leq 3$ for $\forall j = 1, 2, \ldots, M$ where $KS' = \{TS'_1, TS'_2, \ldots, TS'_M\}$ is the theoretical optimization scheme corresponding to $OPT(I)$.

It can be assumed that $n = 3M$.

Assuming $\mathbf{card}(TS_j) = 3$ for $\forall j = 1, 2, \ldots, M$ of instance I, $TS_k \cup TS_j$ can be set as $\{T_{\alpha_1}, T_{\alpha_2}, T_{\alpha_3}, T_{\alpha_4}, T_{\alpha_5}, T_n\}$ where $\forall j \neq k$ and $G_{\alpha_1} \geq G_{\alpha_2} \geq \cdots \geq G_{\alpha_5} \geq G_n$. Then, there are two situations for TS_k and TS_j considering $T_n \in TS_k$ that as:

$$\begin{cases} T_{\alpha_1} \in TS_k \\ T_{\alpha_2}, T_{\alpha_3} \in TS_j \end{cases} \quad or \quad \begin{cases} T_{\alpha_1} \in TS_j \\ T_{\alpha_2}, T_{\alpha_3} \in TS_k \end{cases} \tag{5}$$

If $T_{\alpha_1} \in TS_k$, there are two states that $T_{\alpha_4} \in TS_k$ or $T_{\alpha_5} \in TS_k$. When $T_{\alpha_4} \in TS_k$, the relationships of $\{T_{\alpha_1}, T_{\alpha_2}, T_{\alpha_3}, T_{\alpha_4}, T_{\alpha_5}, T_n\}$ based on Property 2 are:

$$\begin{cases} G_{\alpha_1} + G_{\alpha_4} \leq G_{\alpha_2} + G_{\alpha_3} + G_{\alpha_5} \\ G_{\alpha_1} + G_{\alpha_4} + G_n \geq G_{\alpha_2} + G_{\alpha_3} + G_{\alpha_5} \\ G_{\alpha_1} + G_n \leq G_{\alpha_2} + G_{\alpha_3} \end{cases} \tag{6}$$

$\because I \in H$, $\therefore \exists j \neq k$ **s.t.**:

$$\begin{cases} \frac{G_{\alpha_1}+G_{\alpha_4}+G_n}{G_{\alpha_1}+G_{\alpha_2}} > \frac{9}{8} \\ G_{\alpha_1} + G_{\alpha_2} \geq \frac{\sum_{i=1}^{5} G_{\alpha_i}+G_n}{2} \geq OPT(I) \end{cases} \tag{7}$$

derived from Property 1. Substitution Eq. (7) into Eq. (6) obtains:

$$\frac{53}{24}G_{\alpha_1} + \frac{45}{24}G_{\alpha_2} < \frac{\sum_{i=1}^{5} G_{\alpha_i} + G_n}{2} \leq OPT(I) \tag{8}$$

Eq. (8) contradicts $\frac{53}{24}G_{\alpha_1} + \frac{45}{24}G_{\alpha_2} > G_{\alpha_1} + G_{\alpha_2} \geq OPT(I)$.

When $T_{\alpha_5} \in TS_k$, the relationships are as follows based on Properties 1 and 2.

$$\begin{cases} G_{\alpha_1} + G_{\alpha_5} \geq G_{\alpha_2} + G_{\alpha_3} + G_{\alpha_4} \\ G_{\alpha_5} + G_{\alpha_6} > \frac{1}{8}G_{\alpha_1} + \frac{9}{8}G_{\alpha_2} \\ G_{\alpha_1} + G_{\alpha_2} \geq \frac{\sum_{i=1}^{5} G_{\alpha_i}+G_n}{2} \geq OPT(I) \end{cases} \tag{9}$$

Simplification of Eq. (9) can obtain $4OPT(I) > \frac{35}{8}G_{\alpha_1} + \frac{27}{8}G_{\alpha_2}$. However, $\because \frac{1}{8}G_{\alpha_1} + \frac{9}{8}G_{\alpha_2} < G_{\alpha_5} + G_{\alpha_6} \leq 2G_{\alpha_3}$ and $G_{\alpha_1} \geq G_{\alpha_2} + G_{\alpha_3}$, $\therefore 3G_{\alpha_1} > 5G_{\alpha_2}$. $\therefore \frac{35}{8}G_{\alpha_1} + \frac{27}{8}G_{\alpha_2} > 4\left(G_{\alpha_1} + G_{\alpha_2}\right) \geq 4OPT(I)$, which is constricted to $4OPT(I) > \frac{35}{8}G_{\alpha_1} + \frac{27}{8}G_{\alpha_2}$.

If $T_{\alpha_1} \in TS_j$, $\{T_{\alpha_1}, T_{\alpha_2}, T_{\alpha_3}, T_{\alpha_4}, T_{\alpha_5}, T_n\}$ satisfies that:

$$\begin{cases} G_{\alpha_1} + G_{\alpha_4} \leq G_{\alpha_2} + G_{\alpha_3} \\ G_{\alpha_1} + G_{\alpha_4} + G_{\alpha_5} \geq G_{\alpha_2} + G_{\alpha_3} \\ \frac{G_{\alpha_2}+G_{\alpha_3}+G_n}{G_{\alpha_1}+G_{\alpha_2}} > \frac{9}{8} \\ G_{\alpha_1} + G_{\alpha_2} \geq \frac{\sum_{i=1}^{5} G_{\alpha_i}+G_n}{2} \geq OPT(I) \end{cases} \tag{10}$$

Reduction of Eq. (10) can also obtain Eq. (8), which is constricted to $\frac{53}{24}G_{\alpha_1} + \frac{45}{24}G_{\alpha_2} > OPT(I)$.

Assuming $\exists \mathbf{card}(TS_j) \neq 3$, then $\exists \mathbf{card}(TS_j) = 2$ and $\exists \mathbf{card}(TS_l) \geq 4$. Thus, it can be suppose $\mathbf{card}(TS_l) = 4$ and $TS_l \cap TS_j = \{T_{\alpha_1}, T_{\alpha_2}, \ldots, T_{\alpha_6}\}$ where $G_{\alpha_1} \geq G_{\alpha_2} \geq \cdots \geq G_{\alpha_6}$. Four probable cases are $TS_j = \{T_{\alpha_1}, T_{\alpha_4}\}$, $TS_j = \{T_{\alpha_1}, T_{\alpha_5}\}$, $TS_j = \{T_{\alpha_1}, T_{\alpha_6}\}$ or $TS_j = \{T_{\alpha_2}, T_{\alpha_3}\}$, where One-Step search can reach the optimum in either case.

Using the same derivation method combining with mathematical induction, it can be proved that $\nexists I$ **s.t.** $Ar_{LO}(I) > \frac{5}{4} - \frac{1}{4M}$, which means $H = \emptyset$. Therefore, Theorem 1 is true.

From proof of Theorem 1, it can be observed that One-Step search can improve and optimize the solution when $\mathbf{card}(TS'_j) \leq 3$, $\forall j = 1, 2, \ldots, M$. If the processing time of tasks are as Eq. (11) with $3M$ tasks and M resources, then the allocation result of LPT algorithm is as Eq. (12) where $\varepsilon \in [0, \frac{M}{2})$ and the first row $TS_1 = \{3M - 1, M, M - \varepsilon\}$ means the resource R_1 has three tasks with processing time as $3M - 1$, M and $M - \varepsilon$.

$$\begin{cases} G_1 = 3M - 1; \\ G_2 = 2M - 1; \\ G_i = 2M - \left\lceil \frac{i}{2} \right\rceil, i = 2j + 1, 1 \leq j \leq M - 1; \\ G_{2M+1} = M; \\ G_i = 2M - \left\lceil \frac{i}{2} \right\rceil + \varepsilon, i = 2j, 2 \leq j \leq M; \\ G_i = M - \varepsilon, 2M + 2 \leq i \leq 3M \end{cases} \tag{11}$$

$$\begin{cases} TS_1 = \{3M - 1, M, M - \varepsilon\}; \\ TS_2 = \{2M - 1, M + 0 + \varepsilon, M - \varepsilon\}; \\ TS_3 = \{2M - 2, M + 1 + \varepsilon, M - \varepsilon\}; \\ TS_4 = \{2M - 3, M + 2 + \varepsilon, M - \varepsilon\}; \\ \vdots \\ TS_M = \{2M - (M - 1), M + (M - 2) + \varepsilon, M\} \end{cases} \tag{12}$$

If using the allocation result of LPT as initial allocation or coincidentally using Eq. (12) as the initial allocation, Eq. (12) will be the local optimum of LPT-One, where the approximate ratio is $Ar_{LO}(I^*) = \frac{5M-1-\varepsilon}{4M}$ and $\lim_{\varepsilon \to 0} Ar_{LO}(I^*) = \frac{5M-1}{4M}$. Thus, the instance of Eq. (11) with initial allocation as Eq. (12) is a worst-case of LPT-One when the number of resources is M.

Theorem 2. *$Ar_{BFDO} = Ar_{BO} \leq \frac{5}{4} - \frac{1}{4M}$ for $P||C_{max}$.*

Proof. For BFD-One, assuming $Ar_{BO}(x|\forall x \in H) > \frac{5}{4} - \frac{1}{4M}$, $I \in H$ and $\mathbf{card}(I) = \min \mathbf{card}(x|\forall x \in H)$. Then, the same conclusion as that in LPT-One is $OPT(I) < 4G_n$ for BFDS based on Property 3. Analogous to LPT-One algorithm, it can be proved that $H = \emptyset$ based on Properties 1 and 3 beneficial from combination of OneStep and BFDS.

From the similarities in proofs of Theorems 1 and 2, a novel theorem to improve the approximation ratio based on K-Step search can be gained as:

Theorem 3. *$Ar_{LPTK}, Ar_{BFDK} \leq 1 + \frac{M-1}{(3+K)M}$ in $P||C_{max}$.*

Proof. For LPT-K, similar to the proof of Theorem 1, assuming $Ar_{LPT-K}(x|\forall x \in H) > 1 + \frac{1}{3+K} - \frac{1}{(3+K)M}$, $I \in H$ and $\mathbf{card}(I) = \min \mathbf{card}(x|\forall x \in H)$. Then, a conclusion of I analogous to that in LPT-One search algorithm is $OPT(I) < (3 + K)G_n$ according to Property 2. Therefore, $\mathbf{card}(TS'_j) \leq 2 + K$, $\forall j = 1, 2, \ldots, M$ where $KS' = \{TS'_1, TS'_2, \ldots, TS'_M\}$ is the theoretical optimization scheme corresponding to $OPT(I)$. Based on the same principle of Property 1, K-Step search can optimize the solution when $\mathbf{card}(TS'_j) \leq 2+K$. Therefore, $H = \emptyset$ from mathematical induction.

The proof process also works for BFD-K algorithm.

Theorem 3 reveals an avenue of to optimize the approximate ratio to infinitely approach to 1 by increasing K of K-Step search. For K-Step Search algorithm, the limit of the approximate ratio is 2 in theory which cannot be promoted with increasing K because of the existence of a counterexample as Eq. (13), where $M\varepsilon = \delta$.

$$\underbrace{\underbrace{\{\varepsilon, \varepsilon, \ldots, \varepsilon\}}_{M+K}, \underbrace{\{\varepsilon, \varepsilon, \ldots, \varepsilon\}}_{M+K}, \ldots, \underbrace{\{\varepsilon, \varepsilon, \ldots, \varepsilon\}}_{M+K}}_{M-1}, \{\delta, \delta\} \tag{13}$$

For $\forall K$, $\exists M \gg K$ **s.t.** Eq. (13) is a local optimum where K-Step search algorithm cannot adjust tasks of each resource. Presently in Eq. (13), $Ar_{K-Step} \leq \frac{2M\delta}{(M+1)\delta+(M-1)K\varepsilon} \to 2$. However, combination of LPTS (or BFDS) and K-Step has a dominant performance to surmount the limitation of counterexamples on each search route according to Theorem 3.

Table 3
Summary of proposed algorithms and their corresponding problems evaluated in subsequent experiments.

Ascription	Algorithm	Category	Description	Problems
Single route	LPTS	Local Search	LPT-Search algorithm	$P_1 + P_2$
	MLPTS	Local Search	LPT-Search algorithm	P_2
	BFDS	Local Search	BFD-Search algorithm	P_1
	OneS	Local Search	OneStep-Search algorithm	P_1
	BestBFDS	Local Search	BestBFD-Search algorithm	P_1
Dual routes	LPT-One	Local Search	LPT-OneStep Search algorithm	$P_1 + P_2$
	MLPT-One	Local Search	MLPT-OneStep Search algorithm	P_2
	BFD-One	Local Search	BFD-OneStep Search algorithm	P_1
Triple routes	LPT-BFD-One	Local Search	Using the LPT, BFD and OneStep as search routes	P_1

Table 4
Comparison algorithms evaluated in experiments.

Algorithm	Category	Description	Problems
Random	Randomization	Randomly allocating tasks to resources	P_1
Greedy	Greedy	Scheduling with greed priory	P_1
RR	Heuristic	Round Robin algorithm	P_1
LPT	Heuristic	Longest Processing Time algorithm	P_1
BFD	Heuristic	Best Fit Decreasing algorithm	P_1
GA-Random	Meta-Heuristic	Genetic algorithm using random initialized state	P_1
PSO	Meta-Heuristic	Particle Swarm Optimization algorithm	$P_1 + P_2$
ACO	Meta-Heuristic	Ant Colony Optimization algorithm	P_2
GA-MinMin	Hybrid	Genetic algorithm using MinMin initialized state	$P_1 + P_2$
PSO-GA	Hybrid	Using the output of PSO as the input of GA	$P_1 + P_2$
ACO-GA	Hybrid	Using the output of ACO as the input of GA	P_2

The computational complexity of LPT-One (Cc_{LO}) consists of two parts. One is the number of iterations which can be deduced as O(M) and the other is the complexity of each iteration which is about O (MN) for $P||C_{max}$. Therefore, the computational complexity of LPT-One is $Cc_{LO} = \mathrm{O}\left(M^2N\right)$. BFD-One also satisfies this property. We will also verify Cc_{LO} through experiments in the next section. As the approximation ratios of existing algorithms are as that $Ar_{LPT} \leq \frac{4}{3} - \frac{1}{3M}$, $Ar_{LPT-REV} \leq \frac{4}{3} - \frac{1}{3(M-1)}$ [18,19] and $Ar_{Multifit} \leq \frac{72}{61} + 2^{-k}$ [11], the improvement and proof of approximate ratios of LPT-One and BFD-One still occupy theoretical significance, as well as the LPT-K and BFD-K, which provides a method to approach the upper bound 1.

4.4. Summary of proposed algorithms

Summary of proposed algorithms and corresponding problems in subsequent experiments are in Table 3. LPT-K and BFD-K are not listed in Table 3 as we have given the proof of their theoretical approximation ratio and will not verify them in subsequent experiments.

5. Experiment evaluation

5.1. Problems and simulated environment

Conduction of multi groups of experiments to the comprehensive evaluation of our proposed algorithms is essential. Therefore, we carried out experiments of problems shown in Table 2 for minimizing makespan under static scheduling for homogeneous and heterogeneous resources respectively.

For the problem of minimizing makespan in experiments, we simulate the Cloud environment as that each resource (especially VMs) can process only one task simultaneously, tasks are independent mutually and each task only has one working procedure, where the total processing time of a resource equals to the sum of the processing time of the tasks on this resource.

In consideration of that verification of algorithms' performance especially statistical performance requires abundant instances, we establish the simulation environment through randomly generating tasks, which is conducive to producing adequate instances and observing the performance of different algorithms under the same instance. The parameters of tasks are generated by a given uniform distribution and their specified parameters of the generation will be described in each instance. As the variation in the number of tasks or resources has an impact on the performance of the algorithm, we fix the number of resources or tasks and observe the trend of algorithms' performance with the number of the other.

5.2. Compared baselines and evaluation indexes

To assist the evaluation of the proposed algorithms for problems in Table 2, a variety of commonly used algorithms are regarded as baselines, including some random, greedy, heuristic, meta-heuristic and hybrid algorithms. The details of the proposed algorithms and the comparison algorithms in the experiments are in Table 3 and Table 4 respectively.

Assuming the set of algorithms participating in evaluation is $\left\{X_1, X_2, \ldots, X_p\right\}$, we conduct random experiments with 100 instances for each group of (M, N) donated as $\left\{I_1^{(M,N)}, I_2^{(M,N)}, \ldots, I_{100}^{(M,N)}\right\}$. Then, we donate the makespan obtained by algorithm X_k under the instance $I_l^{(M,N)}$ as $Y\left(X_k, I_l^{(M,N)}\right)$, the total running time as $Z\left(X_k, I_l^{(M,N)}\right)$, and the theoretical optimal makespan of $I_l^{(M,N)}$ as $OPT\left(I_l^{(M,N)}\right)$. Then, we can obtain several statistical indexes as Eq. (14) where λ_1 is the average makespan, λ_2 is the ratio between the average makespan and the least makespan (abbreviated as AM/LAM), λ_3 is probabilities achieving the least makespan (PALM), λ_4 is probabilities achieving the theoretical optimization (PATO), λ_5 is maximum approximate ratio, λ_6 is average of total running time, λ_7 is the ratio between the average total running time and the least total running time (AT/LAT), and

$\lambda_1^{X_k}|_{(M,N)}$ means the index λ_1 of algorithm X_k in the scenario of (M, N).

$$\begin{cases} \lambda_1^{X_k}|_{(M,N)} = \dfrac{\sum_{l=1}^{100} Y\left(X_k, I_l^{(M,N)}\right)}{100}; \lambda_2^{X_k}|_{(M,N)} = \dfrac{\lambda_1^{X_k}|_{(M,N)}}{\min_{q=1}^{p} \lambda_1^{X_q}|_{(M,N)}}; \\ \lambda_3^{X_k}|_{(M,N)} = \dfrac{\sum_{l=1}^{100} \left(Y\left(X_k, I_l^{(M,N)}\right) == \min_{q=1}^{p} Y\left(X_q, I_l^{(M,N)}\right)\right)}{100}; \\ \lambda_4^{X_k}|_{(M,N)} = \dfrac{\sum_{l=1}^{100} \left(Y\left(X_k, I_l^{(M,N)}\right) == OPT\left(I_l^{(M,N)}\right)\right)}{100}; \\ \lambda_5^{X_k}|_{(M,N)} = \max_{l=1}^{100} \left(\dfrac{Y\left(X_k, I_l^{(M,N)}\right)}{OPT\left(I_l^{(M,N)}\right)}\right); \\ \lambda_6^{X_k}|_{(M,N)} = \dfrac{\sum_{l=1}^{100} Z\left(X_k, I_l^{(M,N)}\right)}{100}; \lambda_7^{X_k}|_{(M,N)} = \dfrac{\lambda_6^{X_k}|_{(M,N)}}{\min_{q=1}^{p} \lambda_6^{X_q}|_{(M,N)}}. \end{cases} \tag{14}$$

Except for the above indexes, we will also apply the iterative process of makespan and Pareto scatter to comprehensively evaluate the performance of our proposed algorithms.

Then, simulation experiments are launched in a desktop computer with configurations as follows:

- CPU: Intel(R) Core(TM) i5-8400 CPU @ 2.8 GHZ;
- SSD: KINGSTON SA400S37 240 GB;
- GPU: NVIDIA GeForce GTX 1060 6 GB;
- Program version: Python 3.6;

5.3. Result and discussion

5.3.1. Minimizing makespan for homogeneous resources

Firstly for minimizing makespan of homogenous resources ($P||C_{max}$), we carry out extensive experiments to observe the iterative processes. Since a large number of experiments can obtain similar conclusions, we only plot the iterative process of the two instances $(M, N) = (50, 200)$ and $(M, N) = (100, 10000)$ in Fig. 3 for each algorithm with the property of searching solution in Table 4. In Fig. 3, we choose the minimum makespan of all iterations before the current iteration as the value of the current. As iterations reaching convergence of the proposed algorithms are far less than 100, we replenish them to 100 iterations by their convergence values. Fig. 3 shows that the proposed algorithms take about 25 iterations for instance of $(M, N) = (50, 200)$ and 50 iterations for instance of $(M, N) = (100, 10000)$ to reach an optimized state and their makespans of converging are evidently smaller than compared algorithms, which points out that the proposed algorithms, including OneS, LPTS, BFDS, BestBFDS, LPT-One and BFD-One, can reach a better state close to convergence with iterations about the half number of resources, and reach converges by less than 100 iterations. In Fig. 3, LPT-One possesses the fast convergence speed followed by LPTS and BFD-One.

To further investigate the statistical performance of proposed algorithms for $P||C_{max}$, we fix the number of resources or tasks, and randomly generate 100 instances for each combination (M, N) respectively where $ET_{ij} \in [1, 100]$ (unit of time). Similarly, we plot the average makespans (λ_1) of the 100 instances under each (M, N) for homogeneous resources as Fig. 4. For the sake of quantitative observation, we provide the numerical tables of Fig. 4 in Tables 5 and 6. As shown in Fig. 4, our proposed OneS, BFDS, BestBFDS, LPT-One, BFD-One and LPT-BFD-One achieve the lowest and almost coincident makespans, followed by LPTS, LPT, BFD. The makespans of other algorithms are significantly higher than those of our proposed algorithms. The results of Fig. 4 roughly shows that our proposed algorithms in Table 3 are better than the baselines in Table 4. Since the differences between our proposed OneS, BFDS, BestBFDS, LPT-One, BFD-One and LPT-BFD-One are far smaller than the whole ordinate span, it is difficult to distinguish which is better than others using Fig. 4. Therefore, we calculate the ratio between the average makespan and the least average makespan (AM/LAM, λ_2) and plot the box chart of OneS, BFDS, BestBFDS, LPT-One, BFD-One and LPT-BFD-One in Fig. 5. As Fig. 5, LPT-BFD-One has the lowest AM/LAM followed by LPT-One, BestBFDS and BFD-One. This is because LPT-BFD-One uses three search routes and needs to meet the convergence conditions of them simultaneously, which makes the performance of LPT-BFD-One better than the dual routes algorithms and single routes algorithms. LPT-One and BFD-One outperform LPTS, BFDS and OneS for similar reasons. Additionally, LPTS is better than LPT and BFDS is better than BFD also demonstrate the search algorithm with heuristic algorithm as the search route is better than corresponding heuristic algorithm.

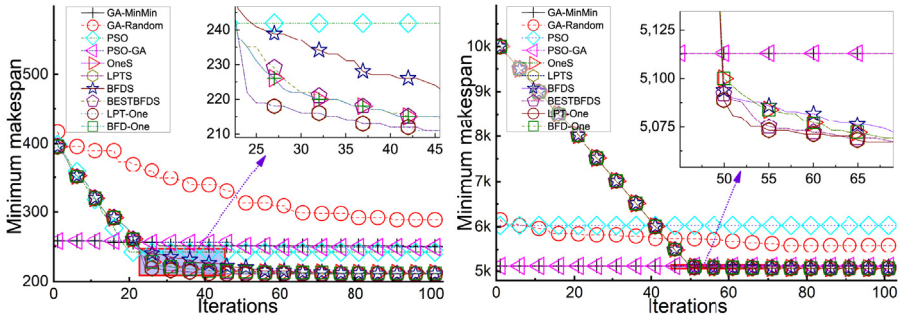
Figs. 4 and 5 verify the performance of our proposed algorithms from the perspective of the average value of makespan. In addition to considering the average performance in practical applications, we usually consider the probability of an algorithm obtaining the best optimization solution. Therefore, we plot the probabilities achieving the least makespan (PALM, λ_3) under each (M, N) in Fig. 6. Consistent with the conclusion form Figs. 4 and 5, LPT-BFD-One has the highest probabilities to obtain the least makespan in Fig. 6 also followed by LPT-One and BestBFDS. On the whole, the probability of LPT-BFD-One achieving the least makespan remains above 70%, that of LPT-BFD-One remains above 65%, and BestBFDS above 60%.

This group of experiments not only demonstrate our proposed multi-search-routes-based algorithms outperform than baseline, but also demonstrate increasing the types of search routes can improve the optimization solution.

To further evaluate the performance of our proposed algorithms, we compare the solution of the proposed algorithms with the theoretical optimal solution obtained by the enumerative algorithm. Considering the computation complexities of the enumerative algorithm are too large for instances with more resources, we only present the results of 3 resources and 4 resources. Then, we plot the probabilities achieving the theoretical optimal makespan (PATO, λ_4) in Fig. 7, and plot the maximum approximation ratio of makespan (λ_5) in Fig. 8.

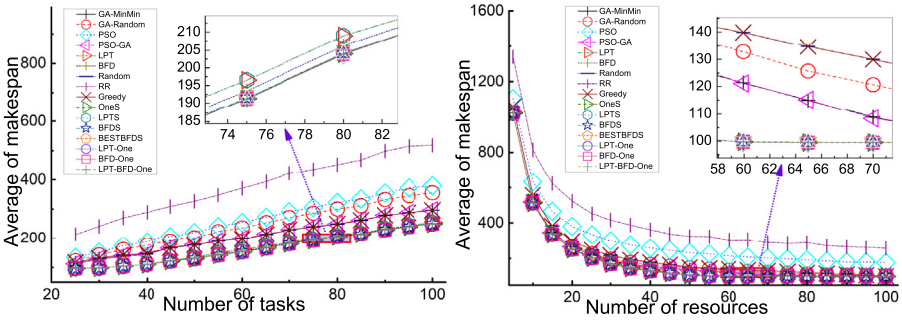
From Fig. 7, LPT-One, BFD-One and their combination LPT-BFD-One occupy higher probabilities to achieve the theoretical optimal makespan than other algorithms under all the combination of (M, N) in Fig. 7. Concurrently, they keep the approximation ratio closest to 1 better than other algorithms from Fig. 8. The results of Figs. 7 and 8 verifies our proofs of Section 4.3 to some extent. From Fig. 7(a), when the number of tasks is 4 times the number of resources, LPT-One, BFD-One and LPT-BFD-One obtain their lowest PATO about 40%, however the PATO of other algorithms such as GA, LPTS, PSO-GA etc are less than 10%, which shows that our proposed multi-routes algorithms have made significant improvement in the probability to achieve the theoretical optimum for NP-Hard problem. Fig. 8 shows that the approximate ratios of our proposed LPT-One, BFD-One and LPT-BFD-One have been lower than 1.1 in experiments, which verifies the stability of these algorithms and can provides a reliable scheme for the task allocation or resource scheduling in realistic.

With above evaluation from several aspects, we continue to verify the calculational complexity. As our proposed multi-route algorithms are based on the general local search algorithm of Algorithm 1, which makes the complexity of these algorithms similar, so we choose to analyze the calculation complexity of LPT-One, whose complexity has been deduced as $Cc_{LO} = O\left(M^2N\right)$ in



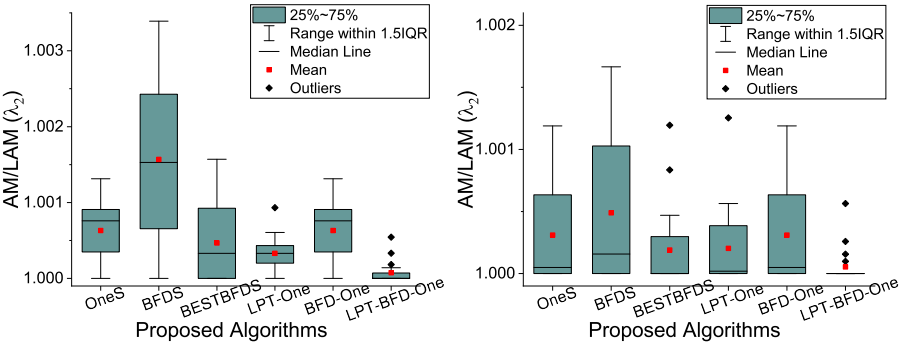
(a) $M = 50, N = 200$ (b) $M = 100, N = 10000$

Fig. 3. Iterative processes of makespan with 100 iterations for the problem of minimizing makespan for homogeneous resources.



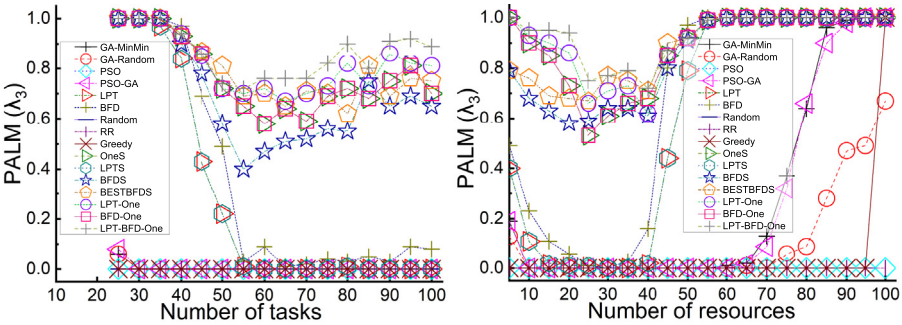
(a) $M = 20, N \in [25, 100]$ (b) $M \in [5, 100], N = 100$

Fig. 4. The average makespans (λ_1) under each (M, N) with 100 instances respectively for problem of minimizing makespan for homogeneous resources.



(a) $M = 20, N \in [25, 100]$ (b) $M \in [5, 100], N = 100$

Fig. 5. The box chart of ratio between average makespan and the least average makespan (AM/LAM, λ_2) for our proposed algorithms corresponding to the experiments of Fig. 4.



(a) $M = 20, N \in [25, 100]$ (b) $M \in [5, 100], N = 100$

Fig. 6. The probabilities achieving the least makespan (PALM, λ_3) corresponding to the experiments of Fig. 4.

Table 5
The average makespans (λ_1) under ($M = 20, N \in [30, 100]$) for homogeneous resources corresponding to Fig. 4(a).

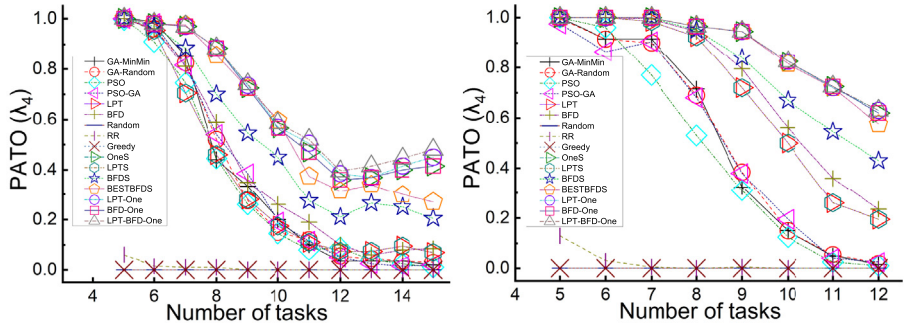
Algorithm	($M = 20, N =?$)							
	30	40	50	60	70	80	90	100
GA-MinMin	127.14	144.08	176.17	197.35	229.1	248.63	278.75	296.06
GA-Random	141.73	174.89	202.07	236.06	271.48	300.08	332.68	355.74
PSO	151.49	187.89	219.97	254.25	291.03	322.66	354.76	379.61
PSO-GA	127.99	143.88	176.71	197.97	229.34	248.57	279.1	296.34
LPT	97.74	110.09	133.35	160.41	185.84	208.98	233.51	254.17
BFD	97.74	109.73	130.24	154.49	182.57	206.17	231.43	250.88
Random	131.12	147.03	176.45	197.55	229.38	249.2	278.86	296.06
RR	241.35	288.8	326.64	371.4	422.43	451.16	498.04	517.94
Greedy	131.12	147.03	176.45	197.55	229.38	249.2	278.86	296.06
OneS	97.74	109.77	128.93	152.46	180.28	203.96	229.26	249.21
LPTS	97.74	110.09	133.35	160.41	185.84	208.98	233.51	254.17
BFDS	97.74	109.86	129.23	152.73	180.55	204.17	229.41	249.29
BESTBFDS	97.74	109.76	**128.83**	152.33	180.16	204.09	229.31	249.3
LPT-One	97.74	**109.75**	128.95	152.3	180.15	203.86	229.14	249.05
BFD-One	97.74	109.77	128.93	152.46	180.28	203.96	229.26	249.21
LPT-BFD-One	97.74	**109.75**	128.9	**152.26**	**180.09**	**203.77**	**229.09**	**248.99**

Table 6
The average makespans (λ_1) under ($M \in [10, 100], N = 100$) for homogeneous resources corresponding to Fig. 4(b).

Algorithm	($M =?, N = 100$)									
	10	20	30	40	50	60	70	80	90	100
GA-MinMin	517.49	266.26	190.86	154.67	135.39	121.39	108.65	101.4	99.44	99.34
GA-Random	518.89	270.42	197.16	164.65	145.47	132.77	120.7	111.55	104.92	101.38
PSO	632.81	381.12	300.31	259.12	233.53	214.75	202.43	191.35	179.5	176.26
PSO-GA	517.51	265.61	190.86	154.26	134.14	121.16	108.18	101.29	99.44	99.34
LPT	514.55	255.78	174.94	134.49	106.84	99.75	99.53	99.21	99.42	99.34
BFD	513.36	253.2	170.57	129.39	106.41	99.75	99.53	99.21	99.42	99.34
Random	556.29	297.61	218.16	178.43	149.41	140.05	129.94	119.93	109.74	99.34
RR	814.32	522.41	416.89	362.53	324.51	304.46	291.22	289.45	266.72	257.43
Greedy	556.29	297.61	218.16	178.43	149.41	140.05	129.94	119.93	109.74	99.34
OneS	512.22	251.27	168.48	127.68	106.47	99.75	99.53	99.21	99.42	99.34
LPTS	514.55	255.78	174.94	134.49	106.84	99.75	99.53	99.21	99.42	99.34
BFDS	512.45	251.44	168.54	127.68	106.47	99.75	99.53	99.21	99.42	99.34
BESTBFDS	512.37	251.36	168.33	**127.59**	**106.46**	99.75	99.53	99.21	99.42	99.34
LPT-One	512.19	251.14	168.38	127.75	106.47	99.75	99.53	99.21	99.42	99.34
BFD-One	512.22	251.27	168.48	127.68	106.47	99.75	99.53	99.21	99.42	99.34
LPT-BFD-One	**512.17**	**251.06**	**168.32**	127.61	106.47	99.75	99.53	99.21	99.42	99.34

Table 7
The parameter ξ and evaluation index R^2 to fit the average computational complexities $Cc_{LO} \approx \xi N$ of LPT-One for $P||C_{max}$.

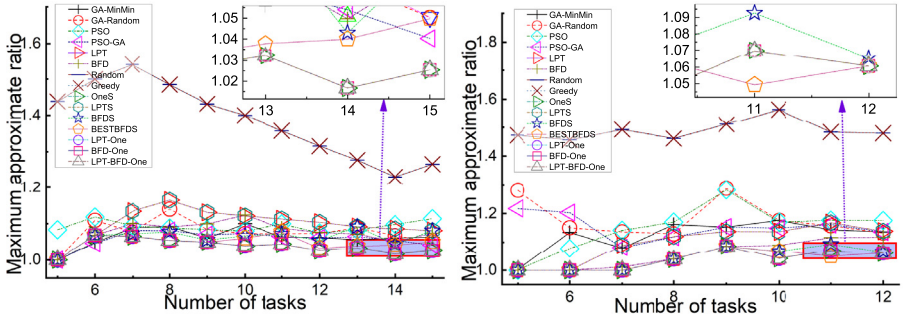
M	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	30	40	50
ξ	15.0	28.7	42.5	56.9	72.2	88.3	105.1	122.6	141.1	161.3	181.9	203.7	226.2	248.6	274.9	300.9	327.4	353.9	381.5	730.4	1193	1760
R^2	0.996	0.997	0.997	0.997	0.995	0.998	0.998	0.997	0.997	0.997	0.997	0.996	0.995	0.998	0.992	0.991	0.988	0.996	0.998	0.992	0.981	0.965



(a) $M = 3, N \in [5, 15]$ (b) $M = 4, N \in [5, 12]$

Fig. 7. The probabilities achieving the theoretical optimal makespan (PATO, λ_4) under each (M, N) with 100 instances respectively for problem of minimizing makespan for homogeneous resources.

Section 4.3. Similar to the indexes of λ_1 to λ_7, we record the calculational complexity of LPT-One in each $I_i^{(M,N)}$ and calculate its average complexities in 100 instances i.e. $\left\{I_1^{(M,N)}, I_2^{(M,N)}, \ldots, I_{100}^{(M,N)}\right\}$. Then, we plot the average complexities of LPT-One in Fig. 9 under the scenarios of ($M \in \{2, 3, \ldots, 7\}, N \in [M, 1000]$). From Fig. 9, the complexity is approximately proportional to the number of tasks for each group of experiment. Thus, we assume the complexity $Cc_{LO} \approx \xi N$ and utilize linear regression to fit the complexities of more groups of experiments. The parameter ξ and the



(a) $M = 3, N \in [5, 15]$ (b) $M = 4, N \in [5, 12]$

Fig. 8. Maximum approximation ratios of makespan (λ_5) corresponding to the experiments of Fig. 7.

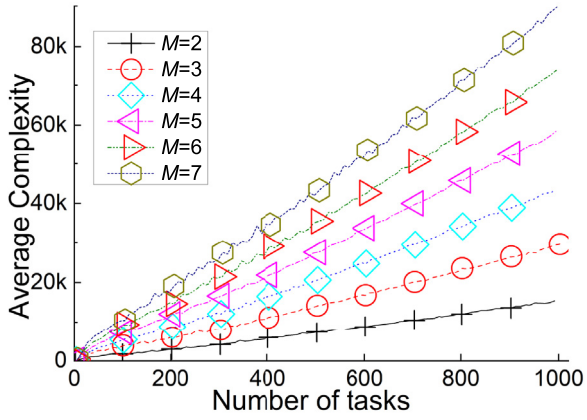


Fig. 9. Average Complexities of LPTO for $P||C_{max}$.

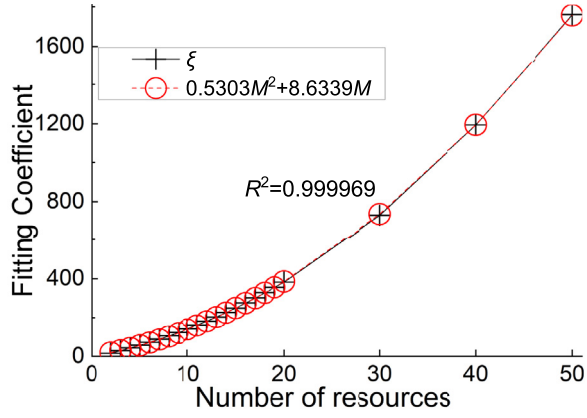


Fig. 10. The relationship between ξ and M.

evaluating indexes *R-Square* are as Table 7. From Table 7, positive scale function $Cc_{LO} = \xi N$ can fit the computational complexity of LPTO well, whose evaluating indexes of R^2 are almost about 0.99. Furthermore, as the coefficient ξ increases monotonically with respect to the number of resources, we leverage quadratic polynomial regression to fit the relation between ξ and M, and then gain the expression as $\xi \approx 0.5303M^2 + 8.6339M$ with evaluation index $R^2 = 0.999969$ shown in Fig. 10. Fig. 10 means it is reliable to use $0.5303M^2 + 8.6339M$ to fit ξ. Therefore, we can obtain the complexity $Cc_{LO} \approx (0.5303M^2 + 8.6339M)N$ which is identical to $Cc_{LO} = \mathrm{O}\left(M^2N\right) + \phi(M, N)$ deduced in Section 4.3. Specifically, the coefficient 0.5303 of M^2 is consistent with the appearance in Fig. 3 that LPTO can achieve the convergence through iterations with about half number of resources. Using similar experimental process can derive similar conclusion for other proposed algorithms. This group of experiments shows that the average computational complexity of our proposed LPT-One belongs to quadratic polynomial.

In summary, this section verifies our proposed algorithms perform well in homogeneous resources from several aspects: convergence, optimality and computational complexity.

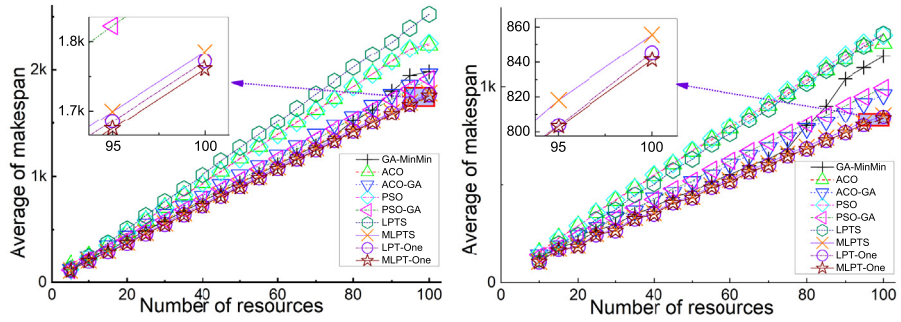
5.3.2. Minimizing makespan and total running time for heterogeneous resources

The above experiments have demonstrated the advantages of the proposed multi-search-routes-based algorithms for minimizing the makespan of homogenous resources. Following, we execute experiments to observe that in heterogeneous resources. As LPT is designed to resolve $P||C_{max}$ for homogeneous resources, the LPT search cannot adapt to the problem of minimizing makespan for heterogeneous resources. However, LPT search can be modified by policy seen in Algorithm 5. Combining with OneStep Search, MLPT-One is also applied to solve problems of minimizing makespan and total running time for heterogeneous resources. In addition, the algorithms of Greedy, RR and Random do not have advantages to solve the problem of minimizing makespan according to the results of the above experiments. Thus, without losing representativeness, we only choose several meta-heuristic algorithms, i.e. GA, ACO, PSO and their combinations, as the baselines of the experiments for heterogeneous resources.

Similarly, extensive experiments can gain the same conclusion, so we only present two groups of experiments that $(M = 5, N \in [5, 100])$ and $(M = 10, N \in [10, 100])$, where each different combination of (M, N) also has 100 random instances generated by simulation systems and the processing time of each task on any resource is a random integer as $ET_{ij} \in [75, 150]$ (unit of time). Then, we plot the average of makespan (λ_1) in Fig. 11 and the average of total running time (λ_5) in Fig. 12 respectively.

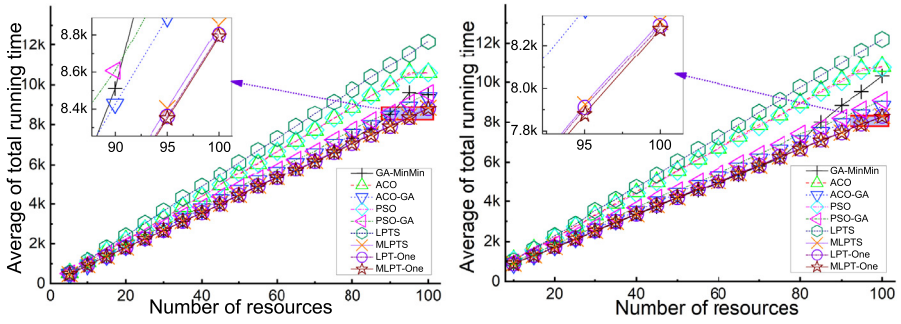
Tables 8 and 9 provide the partial numerical values of Fig. 11. As shown in Figs. 11 and 12, MLPT-One obtains the lower average makespan and lower total running time than other algorithms followed by LPT-One and MLPTS.

For the sake of more clear observation for the results, we plot the AM/LAM (λ_2) and AT/LAT (λ_7) in Fig. 13 and Fig. 14 respectively. To clearly observe the performance of MLPT-One, LPT-One and LPTS, we also plot their box charts in Figs. 15 and 16. It can be clearly seen from Figs. 13 and 14 that MLPT-One, LPT-One, and MLPTS are obviously superior to other algorithms. LPTS has the highest average makespan and average total running time, which illustrates again that LPTS is not suitable for heterogeneous resources. However, combining LPTS with One-Step, LPT-One greatly improves the performance, which is because One-Step adds a convergence condition to ensure optimization.



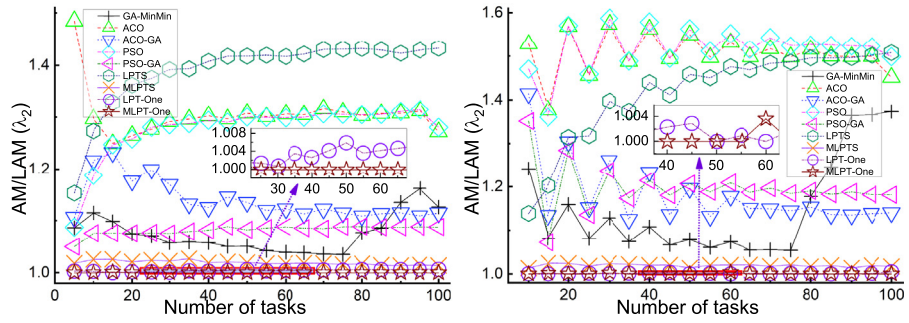
(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 11. The average makespans (λ_1) under each (M, N) with 100 instances respectively for the problem of minimizing makespan and total running time for heterogenous resources.



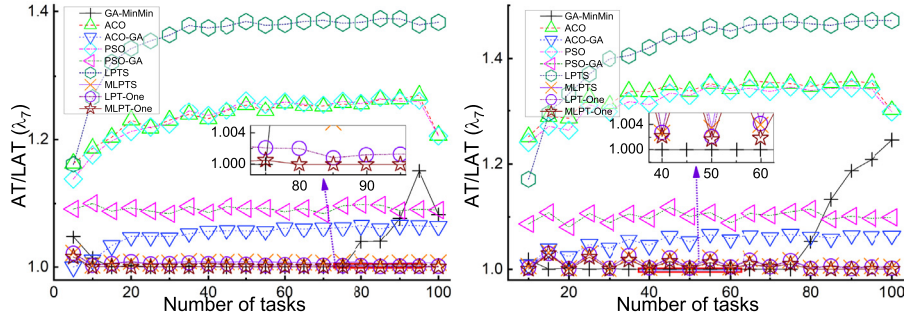
(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 12. The average of total running time (λ_6) under each (M, N) with 100 instances respectively corresponding to the experiments of Fig. 11.



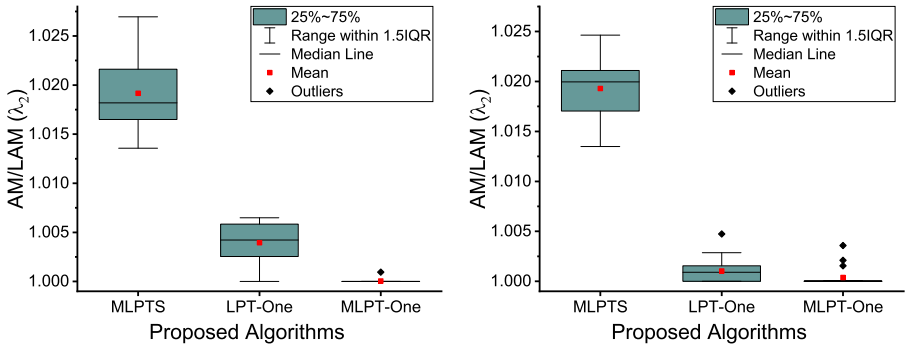
(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 13. The ratio between average makespan and the least average makespan (AM/LAM, λ_2) corresponding to the experiments of Fig. 11.



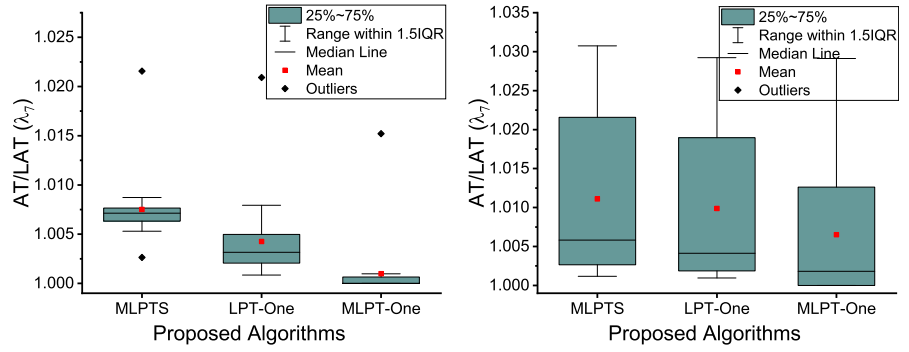
(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 14. The ratio between average total running time and the least average total running time (AT/LAT, λ_7) corresponding to the experiments of Fig. 11.



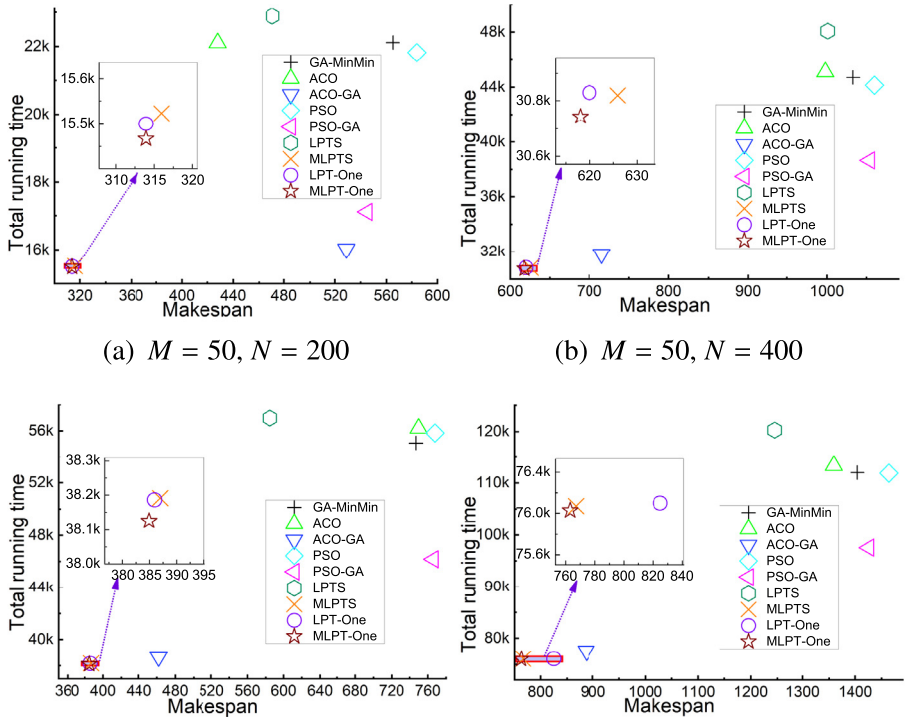
(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 15. The box chart of ratio between average makespan and the least average makespan (AM/LAM, λ_2) corresponding to the experiments of Fig. 11.



(a) $M = 5, N \in [5, 100]$ (b) $M = 10, N \in [10, 100]$

Fig. 16. The box chart of ratio between average total running time and the least average total running time (AT/LAT, λ_7) corresponding to the experiments of Fig. 11.



(a) $M = 50, N = 200$ (b) $M = 50, N = 400$

(c) $M = 100, N = 500$ (d) $M = 100, N = 1000$

Fig. 17. Pareto scatter of makespan and total running time for heterogeneous resources.

Table 8
The average makespans (λ_1) under ($M = 5, N \in [10, 100]$) for heterogenous resources corresponding to Fig. 11(a).

Algorithm		GA-MinMin	ACO	ACO-GA	PSO	PSO-GA	LPTS	MLPTS	LPT-One	MLPT-One
($M = 5, N = ?$)	10	219.31	255.22	239.33	233.47	211.6	250.54	201.79	197.25	**196.78**
	20	400.75	471.78	438.52	469.28	401.04	507.89	380.8	373.68	**373.05**
	30	579.63	708.25	641	704.85	589.47	763.06	560.65	548.81	**548.37**
	40	761.01	937.54	809.71	931.28	778.08	1012.89	733.4	721.7	**719.75**
	50	940.75	1170.94	1015.95	1164.48	968.62	1269.27	912.96	900.77	**895.45**
	60	1110.59	1396.08	1199.26	1388.09	1162.41	1514.34	1087.58	1073.44	**1068.9**
	70	1286.3	1635	1393.58	1623.01	1346.48	1776.3	1264.44	1248.65	**1242.17**
	80	1522.28	1843.21	1567.81	1845.77	1538.95	2025.61	1438.53	1423.11	**1414.35**
	90	1804.21	2079.12	1772.32	2084.7	1728.17	2262.41	1611.54	1599.37	**1589.96**
	100	1981.98	2239.36	1961.87	2254.51	1914.42	2522.67	1785.03	1771.97	**1760.73**

Table 9
The average makespans (λ_1) under ($M = 10, N \in [20, 100]$) for heterogenous resources corresponding to Fig. 11(b).

Algorithm		GA-MinMin	ACO	ACO-GA	PSO	PSO-GA	LPTS	MLPTS	LPT-One	MLPT-One
($M = 10, N = ?$)	20	215.24	291.32	242.09	291.66	238.4	243.97	190.19	185.77	**185.69**
	30	299.84	418.38	335.25	421.97	329.04	371.35	272.11	**265.83**	266.39
	40	386.74	545.67	430.93	551.18	423.88	502.63	356.09	349.92	**349.12**
	50	465.11	668.23	515.39	673.83	520.06	628.53	437.01	**431.03**	**431.03**
	60	551.09	785.96	604.27	796.62	620.45	757.35	523.62	**513.19**	515.03
	70	630.84	906.64	686.45	922.34	714.86	886.76	608.05	597.84	**597.56**
	80	800.08	1030.2	783.7	1037.46	808.76	1016.81	690.98	680.25	**679.64**
	90	1039.54	1142.18	865.15	1161.15	904.97	1140.58	772.6	**762.14**	762.19
	100	1155.25	1221.15	959.23	1260.96	993.74	1268.47	855.65	845.3	**841.31**

The comparison between LPTS and LPT-One also demonstrates that multi-routes can make the algorithm adapt to its originally unsuited scene. From Figs. 15 and 16, MLPT-One outperforms LPT-One and MLPTS, which is because MLPT-One improves the LPT neighborhood compared to LPT-One and adds a search route One-Step compared to MLPTS. Additionally, the observation, that LPT-One performs better than MLPTS in scenarios of heterogeneous resources, confirms the combination of multi-search-routes like LPT-One is more effective than modification of the single algorithm like MLPT.

Pareto Scatter is usually used to evaluate the solution of multi-objective problems [32]. Furthermore, we execute four instances to demonstrate the Pareto Scatter of total running time and makespan as Fig. 17. From Fig. 17, the solution of MLPT-One satisfies Pareto Optimality better than compared algorithms, which gets benefits from the assistance of One-Step and shows again the advantages of multi-search-routes.

Overall, these experimental results validate the feasibility and superiority of using multi routes-based algorithms to address problems of heterogeneous resources.

5.4. Summary

In multi groups of experiments with abundant simulation instances, the proposed algorithms based on multi-search routes outperform the compared baselines. LPT-One, BFD-One and LPT-BFD-One achieve higher probabilities to obtain the best solutions and with lower approximation ratios of the worst cases for minimizing the makespan of homogenous resources. LPT-One and its modified algorithm MLPT-One achieve better solutions for minimizing makespan and total running time of heterogeneous resources than compared baselines.

6. Conclusions and future work

In this paper, we propose local search algorithms, LPT-Search, BFD-Search, and OneStep-Search, using heuristic algorithms LPT and BFD as basic search routes to solve resource scheduling problems in Cloud computing. Based on the basic search routes, we also propose multi-search-routes-based algorithms combining various search routes including LPT-One, BFD-One and LPT-BFD-One.

By theoretical deductions, we prove the approximation ratios of LPT-One and BFD-One as $\frac{5}{4} - \frac{1}{4M}$ as well as that of LPT-K- and BFD-K as $1 + \frac{M-1}{(3+K)M}$, which are better than approximation ratios of LPT, LPT-REV and other existing algorithms for $P||C_{max}$. Moreover, in extensive simulation experiments for minimizing makespan for homogenous and heterogeneous resources, the proposed algorithms based on multi-search-routes outperform the compared algorithms with observations of various indexes, which demonstrates the fact that the proposed algorithms can achieve better solutions in fewer iterations also with better optimization results.

In addition to improving the theoretical approximation ratio of the algorithm, the dominant meaning of proposed algorithms is that they demonstrate the significant potential of applying heuristic algorithms as the search routes of search algorithms and combining different search routes to increase the theoretical analyzability and comprehensive performance of algorithms. Along this research direction as part of future work, we plan to apply the search route to other algorithms such as meta-heuristic algorithms and machine learning algorithms, to explore more search routes-based algorithms and combinations of multi-routes to optimize the performance of scheduling algorithms for more objectives and complex scenarios in Cloud computing. We will also explore whether LPT-K can improve the existing PTAS. In theory, it is also a meaningful work to explore and prove the theoretical approximation ratio of MLPT-One and other search algorithms in heterogeneous resources.

CRediT authorship contribution statement

Guangyao Zhou: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Wenhong Tian:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Rajkumar Buyya:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is funded by the National Natural Science Foundation of China, with Grant ID 61672136 and 61828202.

References

[1] M. Adhikari, T. Amgoth, S.N. Srirama, A survey on scheduling strategies for workflows in cloud environment and emerging trends, ACM Comput. Surv. 52 (4) (2019) 68:1–68:36.
[2] M. Helft, Google confirms problems with reaching its services, in: The New York Times, 2009.
[3] J. Markoff, Software via the internet: Microsoft in cloud computing, in: New York Times, Vol. 3, 2007.
[4] J. Chase, D. Niyato, Joint optimization of resource provisioning in cloud computing, IEEE Trans. Serv. Comput. 10 (3) (2017) 396–409.
[5] R. Yang, Y. Zhang, P. Garraghan, Y. Feng, J. Ouyang, J. Xu, Z. Zhang, C. Li, Reliable computing service in massive-scale systems through rapid low-cost failover, IEEE Trans. Serv. Comput. 10 (6) (2017) 969–983.
[6] Z. Zhan, X.F. Liu, Y. Gong, J. Zhang, H.S. Chung, Y. Li, Cloud computing resource scheduling and a survey of its evolutionary approaches, ACM Comput. Surv. 47 (4) (2015) 63:1–63:33.
[7] T. Welsh, E. Benkhelifa, On resilience in cloud computing: A survey of techniques across the cloud domain, ACM Comput. Surv. 53 (3) (2020) 59:1–59:36.
[8] P. Cong, G. Xu, T. Wei, K. Li, A survey of profit optimization techniques for cloud providers, ACM Comput. Surv. 53 (2) (2020) 26:1–26:35.
[9] A.R. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in cloud computing: A literature survey, Future Gener. Comput. Syst. 91 (2019) 407–415.
[10] J. Mei, K. Li, Z. Tong, Q. Li, K. Li, Profit maximization for cloud brokers in cloud computing, IEEE Trans. Parallel Distrib. Syst. 30 (1) (2019) 190–203.
[11] L. Ghalami, D. Grosu, Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm, J. Parallel Distrib. Comput. 133 (2019) 221–231.
[12] Y. Laili, F. Tao, F. Wang, L. Zhang, T. Lin, An iterative budget algorithm for dynamic virtual machine consolidation under cloud computing environment, IEEE Trans. Serv. Comput. 14 (1) (2021) 30–43.
[13] S.C. A, C. Sudhakar, T. Ramesh, Energy efficient VM scheduling and routing in multi-tenant cloud data center, Sustain. Comput. Inform. Syst. 22 (2019) 139–151.
[14] A.S. Sofia, P. Ganeshkumar, Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II, J. Netw. Syst. Manag. 26 (2) (2018) 463–485.
[15] M. Li, F.R. Yu, P. Si, W. Wu, Y. Zhang, Resource optimization for delay-tolerant data in blockchain-enabled iot with edge computing: A deep reinforcement learning approach, IEEE Internet Things J. 7 (10) (2020) 9399–9412.
[16] W. Guo, W. Tian, Y. Ye, L. Xu, K. Wu, Cloud resource scheduling with deep reinforcement learning and imitation learning, IEEE Internet Things J. 8 (5) (2021) 3576–3586.
[17] J. Mao, Q. Pan, Z. Miao, L. Gao, An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance, Expert Syst. Appl. 169 (2021) 114495.
[18] Y.J. Kim, J.W. Jang, D.S. Kim, B.S. Kim, Batch loading and scheduling problem with processing time deterioration and rate-modifying activities, Int. J. Prod. Res. (2021) 1–21.
[19] F.D. Croce, R. Scatamacchia, The longest processing time rule for identical parallel machines revisited, J. Sched. 23 (2) (2020) 163–176.
[20] Y. Laili, S. Lin, D. Tang, Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment, Robot. Comput.-Integr. Manuf. 61 (2020) 101850.
[21] S. Guo, J. Liu, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, IEEE Trans. Mob. Comput. 18 (2) (2019) 319–333.
[22] M. Kumar, S.C. Sharma, A. Goel, S.P. Singh, A comprehensive survey for scheduling techniques in cloud computing, J. Netw. Comput. Appl. 143 (2019) 1–33.
[23] C. Bitsakos, I. Konstantinou, N. Koziris, DERP: A deep reinforcement learning cloud system for elastic resource provisioning, in: 2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2018, Nicosia, Cyprus, December (2018) 10-13, IEEE Computer Society, 2018, pp. 21–29.
[24] S.S. Haytamy, F.A. Omara, A deep learning based framework for optimizing cloud consumer qos-based service composition, Computing 102 (5) (2020) 1117–1137.
[25] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, J. Zeng, Q-learning based dynamic task scheduling for energy-efficient cloud computing, Future Gener. Comput. Syst. 108 (2020) 361–371.
[26] C. Xu, J. Rao, X. Bu, URL: A unified reinforcement learning approach for autonomic cloud management, J. Parallel Distrib. Comput. 72 (2) (2012) 95–105.
[27] K. Lolos, I. Konstantinou, V. Kantere, N. Koziris, Elastic management of cloud applications using adaptive reinforcement learning, in: 2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December (2017) 11-14, IEEE Computer Society, 2017, pp. 203–212.
[28] S.M.R. Nouri, H. Li, S. Venugopal, W. Guo, M. He, W. Tian, Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications, Future Gener. Comput. Syst. 94 (2019) 765–780.
[29] J. Feng, F.R. Yu, Q. Pei, X. Chu, J. Du, L. Zhu, Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach, IEEE Internet Things J. 7 (7) (2020) 6214–6228.
[30] K. Karthiban, J.S. Raj, An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm, Soft Comput. 24 (19) (2020) 14933–14942.
[31] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang, A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, in: 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June (2017) 5-8, IEEE Computer Society, 2017, pp. 372–382.
[32] Z. Tong, H. Chen, X. Deng, K. Li, K. Li, A scheduling scheme in the cloud computing environment using deep Q-learning, Inform. Sci. 512 (2020) 1170–1191.
[33] C. Li, Y. Zhang, Y. Luo, Neighborhood search-based job scheduling for iot big data real-time processing in distributed edge-cloud computing environment, J. Supercomput. 77 (2) (2021) 1853–1878.
[34] C. Luo, B. Qiao, W. Xing, X. Chen, P. Zhao, C. Du, R. Yao, H. Zhang, W. Wu, S. Cai, B. He, S. Rajmohan, Q. Lin, Correlation-aware heuristic search for intelligent virtual machine provisioning in cloud systems, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February (2021) 2-9, AAAI Press, 2021, pp. 12363–12372.
[35] M.R.G. Raman N. Somu and, A. Kaveri, A.R. K., K. Krithivasan, V.S.S. Sriram, IBGSS: an improved binary gravitational search algorithm based search strategy for qos and ranking prediction in cloud environments, Appl. Soft Comput. 88 (2020) 105945.
[36] K.R.P. Kumar, K. Kousalya, Amelioration of task scheduling in cloud computing using crow search algorithm, Neural Comput. Appl. 32 (10) (2020) 5901–5907.
[37] C. Chen, L. Hung, S. Hsieh, R. Buyya, A.Y. Zomaya, Heterogeneous job allocation scheduler for hadoop mapreduce using dynamic grouping integrated neighboring search, IEEE Trans. Cloud Comput. 8 (1) (2020) 193–206.
[38] M. Diallo, A. Quintero, S. Pierre, A tabu search approach for a virtual networks splitting strategy across multiple cloud providers, Int. J. Metaheuristics 7 (3) (2020) 197–238.
[39] X.F. Liu, Z. Zhan, J.D. Deng, Y. Li, T. Gu, J. Zhang, An energy efficient ant colony system for virtual machine placement in cloud computing, IEEE Trans. Evol. Comput. 22 (1) (2018) 113–128.
[40] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, N. Linge, A speculative approach to spatial–temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, Secur. Commun. Netw. 9 (17) (2016) 4002–4012.
[41] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, L. Qi, A computation offloading method over big data for iot-enabled cloud–edge computing, Future Gener. Comput. Syst. 95 (2019) 522–533.
[42] H. Jiang, J. Yi, S. Chen, X. Zhu, A multi-objective algorithm for task scheduling and resource allocation in cloud-based disassembly, J. Manuf. Syst. 41 (2016) 239–255.
[43] M. Adhikari, T. Amgoth, S.N. Srirama, Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach, Appl. Soft Comput. 93 (2020) 106411.

[44] A.F.S. Devaraj, M. Elhoseny, S. Dhanasekaran, E.L. Lydia, K. Shankar, Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments, J. Parallel Distrib. Comput. 142 (2020) 36–45.
[45] H. Li, G. Zhu, Y. Zhao, Y. Dai, W. Tian, Energy-efficient and qos-aware model based resource consolidation in cloud data centers, Clust. Comput. 20 (3) (2017) 2793–2803.
[46] M. Zhang, Y. Peng, M. Yang, Q. Yin, X. Xie, A discrete pso-based static load balancing algorithm for distributed simulations in a cloud environment, Future Gener. Comput. Syst. 115 (2021) 497–516.
[47] R. Jena, Multi objective task scheduling in cloud environment using nested pso framework, Procedia Comput. Sci. 57 (2015) 1219–1227.
[48] V.V. Vazirani, Approximation Algorithms, Springer, 2001.
[49] W. Zhang, Y. Wen, Energy-efficient task execution for application as a general topology in mobile cloud computing, IEEE Trans. Cloud Comput. 6 (3) (2018) 708–719.
[50] W. Tian, Q. Xiong, J. Cao, An online parallel scheduling method with application to energy-efficiency in cloud computing, J. Supercomput. 66 (3) (2013) 1773–1790.
[51] Z. Hong, W. Chen, H. Huang, S. Guo, Z. Zheng, Multi-hop cooperative computation offloading for industrial iot-edge-cloud computing environments, IEEE Trans. Parallel Distrib. Syst. 30 (12) (2019) 2759–2774.
[52] W. Tian, M. He, W. Guo, W. Huang, X. Shi, M. Shang, A.N. Toosi, R. Buyya, On minimizing total energy consumption in the scheduling of virtual machine reservations, J. Netw. Comput. Appl. 113 (2018) 64–74.
[53] Z. Guan, T. Melodia, The value of cooperation: Minimizing user costs in multi-broker mobile cloud computing networks, IEEE Trans. Cloud Comput. 5 (4) (2017) 780–791.
[54] A.M.S. Kumar, M. Venkatesan, Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment, Wirel. Pers. Commun. 107 (4) (2019) 1835–1848.
[55] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, X. Shu, A dynamic ant-colony genetic algorithm for cloud service composition optimization, Int. J. Adv. Manuf. Technol. 102 (1–4) (2019) 355–368.
[56] G. Ismayilov, H.R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing, Future Gener. Comput. Syst. 102 (2020) 307–322.
[57] M. M, J. T, Combined particle swarm optimization and ant colony system for energy efficient cloud data centers, Concurr. Comput. Pract. Exp. 33 (10) (2021).
[58] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, M. Rida, FACO: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing, J. Ambient Intell. Humaniz. Comput. 11 (10) (2020) 3975–3987.
[59] W. Tian, M. Xu, Y. Chen, Y. Zhao, Prepartition: A new paradigm for the load balance of virtual machine reservations in data centers, in: IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June (2014) 10-14, IEEE, 2014, pp. 4017–4022.

Guangyao Zhou received Bachelor's degree and Master's degree from School of architectural engineering, Tianjin University, China. He is now a Ph.D candidate at School of information and software engineering, University of Electronic Science and Technology of China. His research interests include scheduling algorithms in Cloud Computing, facial expression recognition, algorithmic theory of machine learning and BigData processing.

Wenhong Tian received a Ph.D. degree from the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. He is now a professor at the University of Electronic Science and Technology of China (UESTC). His research interests include scheduling in Cloud computing and Bigdata platforms, and image recognition by deep learning. He has more than 110 journal/conference publications and 5 books in related areas.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012–2016. He received the Ph.D. degree in Computer Science and Software Engineering from Monash University, Melbourne, Australia, in 2002. He has authored over 750 publications and seven text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=152, gindex=331, 120600+ citations). He is recognized as a "Web of Science Highly Cited Researcher" for six consecutive years since 2016, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing and distributed systems. He has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of him are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC.