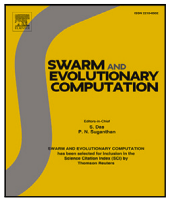
Swarm and Evolutionary Computation 87 (2024) 101575

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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



Information interaction and partial growth-based multi-population growable genetic algorithm for multi-dimensional resources utilization optimization of cloud computing

Guangyao Zhou [a], Yuanlun Xie [a], Haocheng Lan [a], WenHong Tian [a,*], Rajkumar Buyya [b], Kui Wu [c]

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

ARTICLE INFO

Keywords:
Multi-population evolution
Growable genetic algorithm
Cloud computing
Utilization optimization
Multi-dimensional resources
Elite sharing

ABSTRACT

Optimizing multi-dimensional resource utilization is a critical research area in distributed computing, particularly in cloud computing, where various heterogeneous resources are integrated to offer a wide range of services. Addressing this issue necessitates the simultaneous consideration of multiple resource bottlenecks. This paper presents a new solution, called the Multi-Population Growth Genetic Algorithm (MPGGA), which consists of a central management unit responsible for executing information interaction and growth quota reallocation, and multiple population evolution executors to perform crossover and regeneration within each population. The proposed MPGGA combines elite sharing and priority support for the weaker population (ESPW), resulting in better convergence and optimality than other combinations of strategies. This outcome is corroborated by extensive ablation experiments on various strategies. Furthermore, the experimental results for minimizing the maximum utilization of resources in each dimension indicate that MPGGA-ESPW outperforms other popular algorithms, such as GHW-NSGA II (1.363x), GHW-MOEA/D (1.339x), NSGA II (1.948x), and MOEA/D (2.151x) in terms of convergence speed. For energy consumption-related optimization problems, the experimental results demonstrate that the adaptability of a single algorithm in MPGGA family is limited by the algorithm of growth route, while also showing that the MPGGA framework is flexible to allow various algorithms as its growth route to adapt to various scenarios.

1. Introduction

The advent of Industry 4.0 has led to a significant increase in data volume, necessitating more advanced requirements for large-scale distributed computing systems [1]. Cloud computing, a prevalent high-performance distributed system paradigm, offers reliable, flexible, and dynamic services [2]. It integrates heterogeneous electronic components, such as CPUs, GPUs, and RAM, to accommodate various requests, including computing, caching, and storage [3]. In realistic scenarios, the bottleneck of any electronic component can impact the utilization of other components [4,5]. Consequently, the simultaneous optimization of various components has emerged as a critical research trend in cloud computing, which points to the optimization problem of multi-dimensional resource utilization [6–9]. Utilization optimization in distributed systems is typically an NP-hard problem, demanding considerable computational complexity to find optimal solutions. Moreover, the utilization optimization of multi-dimensional resources (abbreviated as UPMDR) is also a multi-objective optimization problem (MOP).

In cloud computing, existing scheduling algorithms include heuristics, meta-heuristics, and machine learning, all of which can achieve acceptable solutions in specific scenarios. Heuristic algorithms excel in single-objective optimization but often struggle with MOPs. Some machine learning algorithms, such as Deep Q Network (DQN) [10,11], ADEC [12] and DQTS [13], have demonstrated the potential to solve MOPs with a certain degree of optimality in their experimental scenarios. However, these approaches require exponentially larger training datasets as the number of tasks and server nodes increases. Meta-heuristic algorithms, with their ability to search for solution sets, offer comprehensive advantages in terms of complexity and optimality when solving MOPs, particularly in cloud scheduling [14]. Most

* Corresponding author.
E-mail addresses: guangyao_zhou@std.uestc.edu.cn (G. Zhou), ylxie@std.uestc.edu.cn (Y. Xie), 202122090620@std.uestc.edu.cn (H. Lan), tian_wenhong@uestc.edu.cn (W. Tian), rbuyya@unimelb.edu.au (R. Buyya), wkui@uvic.ca (K. Wu).

https://doi.org/10.1016/j.swevo.2024.101575
Received 8 November 2023; Received in revised form 27 March 2024; Accepted 8 April 2024
Available online 30 April 2024

meta-heuristic algorithms are evolutionary or swarm algorithms. In meta-heuristic algorithms, the NSGA (Non-dominated Sorting Genetic Algorithms) family [15–18] and the MOEA/D (Multiobjective Evolutionary Algorithm Based on Decomposition) family [19,20] are two frequently used algorithm families currently [21]. Two of the key factors affecting their application in UPMDR are convergence speed and optimality. A new method to increase the convergence speed and optimality in solving UPMDR is the GGA (Growable Genetic Algorithm) family that added an extra growth stage to enable individuals in GA to improve their genes through certain search algorithms before participating in the genetic process [22]. The shortcomings of GGA are that it only contains one population (single-population) and all individuals in GGA need to participate in the extra growth route (marked as all growth). Single-population will lead to premature local optima due to limited diversity, thereby reducing optimality. All growth will bring redundant computational complexity and imbalance within the population, thereby slowing down the convergence speed of the whole genetic system. Thus, the performance of GGA can still be further improved by addressing these two shortcomings.

It is widely recognized that even minor enhancements to large-scale cloud systems can bring significant benefits. Focusing on the shortcomings of GGA, we are committed to ongoing refinements within the GGA family to further enhance the management capability of cloud systems for multi-dimensional resources. We introduce the multi-population strategy into GGA to construct the multi-population growable genetic algorithm (MPGGA). Multi-population is a well-established method for nature-inspired optimization algorithms [23]. In multi-population evolutionary algorithms, information interaction between populations is a critical component [21,24–26]. Correspondingly, we develop three information interaction strategies between multi-populations. In order to enhance the evolutionary effectiveness of MPGGA, we propose a partial growth strategy to improve the growth efficiency of the genetic system, which can maintain balance within the population. Correspondingly, we propose three growth quota reallocation tactics to further optimize MPGGA's performance. Growth quota reallocation tactics can readjust the number of individuals in each population's current generation that can enter the growth route according to the evolutionary state of each population. With these strategies, MPGGA achieves better convergence speed and optimality than GGA. Another benefit of MPGGA is that it allows evolutions of multiple populations to perform parallel computing on multi-core or multithreaded platforms. In terms of some typical MOP indicators (hypervolume, Pareto solutions and C indicator) and statistic tests (Wilcoxon rank-sum test and Friedman test), rigorous experimentation in solving UPMDR confirms that our proposed MPGGA outperforms the baseline algorithms including GGA [22], NSGA II [27], MOEA/D [19], MP-NSGA II [25,28], MP-MOEA/D and their variants in solving UPMDR of cloud computing. Additionally, we also provide some experimental results of MPGGA for other problems including energy consumption-related problems and multi-objective asymmetric traveling salesman problems (MoATSP, in Appendix B), which can supplement the analysis to the adaptability of MPGGA framework.

The main contributions of this paper are summarized as follows.

(1) We propose the Multi-Population Growable Genetic Algorithm (MPGGA) framework to address UPMDR by incorporating multi-populations into the GGA, which is an exploration of a new architecture for evolutionary algorithms. To execute the program of MPGGA, we also design a multi-processes-based architecture with one management center and multi-population internal evolution executors.
(2) Building on the MPGGA framework, we develop three information interaction strategies for exchanging solutions between populations. This approach effectively leverages the information from multiple populations, resulting in enhanced convergence speed for MPGGA.
(3) To further optimize MPGGA's performance, we propose partial growth in lieu of all growth and introduce three growth quota reallocation strategies to dynamically adjust the number of individuals entering the growth route for each population. These strategies facilitate the full utilization of computational power, ultimately improving MPGGA's efficiency.
(4) We conduct comprehensive experiments to evaluate different combinations of interaction strategies and growth quota reallocation strategies. The experimental results demonstrate that MPGGA exhibits superior convergence speed and optimality compared to baseline algorithms. The experimental results also verify the adaptability of the MPGGA framework to diverse scenarios, allowing different algorithms to serve as growth routes of MPGGA to adapt to various problems.

The rest of this paper is organized as follows. We review the related work in Section 2. The system model and problem formulation are constructed in Section 3. The framework of MPGGA and its various strategies are proposed in Section 4. The extensive experiments to evaluate the proposed method are presented in Section 5. Finally, we conclude this paper in Section 6.

2. Related work

2.1. Optimization algorithms

The frequently used scheduling algorithms in cloud computing mainly include heuristic algorithms, meta-heuristic algorithms, machine learning algorithms, and hybrid algorithms [2,29].

Generally, heuristic algorithms were frequently used for some single objective optimization problems. Some popular heuristics include round-robin (RR), longest processing time first (LPT), greedy, random, first come first serve (FCFS) [30], etc. Heuristic algorithms can usually obtain an initial state of some search algorithms to accelerate convergence such as Jacobi Best-response Algorithm [31], FISTA [32], LARAC [33]. In recent studies, heuristic algorithms were also used as the search route of algorithms. In [34], multi-search route algorithms used heuristic algorithms as the search route to solve the minimizing makespan of heterogeneous cloud, which combined the advantages of heuristic and local search. Heuristics still have some adaptability in large-scale optimization problems due to their low computational complexity and analytical theoretical approximation. However, there are several defects of heuristics: a heuristic is often designed for one or few specific scenarios; heuristics are usually only suitable for single-objective problems or single-dimensional resources optimization; heuristics without search capability usually have worse solutions than search-based algorithms.

Machine learning (ML) algorithms of resource scheduling are mainly established by reinforcement learning or deep reinforcement learning such as QEEC [35], ADEC [12] and URL [36], Deep Q Network (DQN) [10,11], ADRL [37], DQTS [13]. Machine learning algorithms of resource scheduling have the ability to model for complex scenarios and optimization objectives. However, ML consumes large computing power in the progress of training, has unstable optimization results and has poor migration ability for scenarios due to its dependence on training datasets.

Common meta-heuristic algorithms include ant colony algorithms (such as MALO [38] and S-MOAL [39]), genetic algorithms (GA, such as NSGA II [27,40], NSGA III [17] and MOGA [5]), particle swarm optimization (such as MOPSO [41,42], TSPSO [43], and HAPSO [4]), bee colony algorithm [44], as well as firefly algorithm [45]. Liu et al. [46] proposed OEMACS combining OEM (order exchange and migration) local search techniques and ACO to resolve energy consumption of VMs deployment in Cloud computing, which significantly reduced the energy consumption and improved the effectiveness of different resources. Monge DA et al. [47] proposed an online multi-objective genetic autoscaler to optimize the scientific and engineering workflows

in cloud infrastructures with unreliable virtual machines. Shikha Mehta et al. [48] proposed two variants of the whale optimization algorithm for efficiently placing VMs on physical machines. Due to its adaptability, the meta-heuristic algorithm is a major type of algorithm to optimize resource utilization. NSGA proposed by Deb et al. [15,15] and MOEA/D proposed by Qingfu Zhang and Hui Li [19] are the two common families. To improve the optimality and convergence speed of GA, the growable genetic algorithm was proposed, whose GHW-NSGA II and GHW-MOEA/D showed a better convergence speed and optimization than NSGA II and MOEA/D [22]. Meta-heuristic algorithms are more applicable than heuristic algorithms suitable for more complex optimization scenarios because meta-heuristics are generally universal optimization strategies. In addition, based on search capability that continuously updates the optimization solution (or solution set), meta-heuristic can obtain better optimization solutions with higher probabilities. However, there are several inevitable defects of meta-heuristics: the convergence of the meta-heuristic cannot be guaranteed due to the presence of randomness; the randomness of the meta-heuristic also increases redundant computations; as the search space increases, the required search iterations must also increase accordingly, subsequently producing more redundant solutions.

At present, hybrid algorithms are based on a combination of meta-heuristic. For example, PSO-ACS [49] applied PSO for an optimal solution of task scheduling and ACO for the best migration path of VMs on PMs; HEFT-GA [50] adhibited HEFT to generate the initial population of GA. SFLA-GA algorithm [51] (shuffled frog leaping algorithm + GA) took advantage of the two algorithms to transmit information among groups hence the search route. A hybrid algorithm, with multiple heuristics or meta-heuristics as elemental algorithms, cannot exceed the scenarios that the elemental algorithms are suitable for. Additionally, various optimization algorithms also need to consider a balance between computation time and optimization performance.

2.2. Multi-population evolutionary algorithms

In evolutionary algorithms, the population is a basic factor [23]. From the existing Refs. [14,23,52–58], multi-population (MP) is a valid approach to improve the performance of evolutionary algorithms.

Sukanta Nama et al. [52] applied MP to improve the backtracking search algorithm and proposed ImBSA to solve optimization problems. Kaixi Yang et al. [59] used a dual-population evolutionary algorithm (dp-ACS) to deal with constrained multi-objective optimization problems, which can dynamically adjust constraint strength to improve the diversity. Liyun Fu et al. [53] proposed constrained cooperative adaptive multi-population differential evolutionary to solve economic load dispatch problems, which had constraint-handling efficiency and better global searching ability. Djaballah et al. [54] proposed a multi-population ABC algorithm for optimization problems, which divides the colony into multi sub-populations to increase the diversity of solutions. Yandi Zuo et al. [55] developed a novel multi-population artificial bee colony algorithm to minimize the makespan, total tardiness and total energy consumption (TEC) for energy-efficient hybrid flow shop scheduling problems. Yongjun Sun and Yu Chen [56] proposed an improved whale optimization algorithm with multi-population (MIWOA) to address high dimensional optimization, where the better group of individuals was used to improve exploitation performance and the poorer group was used to improve exploration performance. The multi-population enhanced the solution accuracy and convergence speed with less execution time. Li-Jiao Wu et al. [60] proposed a multiple population-based multi-objective ant colony system to solve cold chain logistics (CCL) scheduling problems. Guoqing Li et al. [61] proposed a grid search-based multi-population particle swarm optimization algorithm to handle multimodal multi-objective optimization problems. Xinming Zhang et al. [62] proposed MPBBO (multi-population biogeography-based optimization algorithm) which had stronger search ability and higher efficiency than WRBBO (Worst opposition learning and Random-scaled differential mutation Biogeography-Based Optimization) and performed well in image segmentation.

In the implementation of multi-population, multi-population genetic algorithm (MPGA) is a frequently used algorithm [14,25,63,64]. Zhiyong Xiao et al. [25] proposed a highly scalable hybrid parallel genetic algorithm (HPGA) with multi sub-population to solve large-scale optimization problems, which can effectively exploit individual diversity. Jia Luo et al. [26] applied the heterogeneous parallel genetic algorithm to solve the job shop scheduling problem, where GPUs were used to accelerate the execution time. Idir Aoudia et al. [28] implement the MPGA to enhance the QoS-Aware Service quality of fog-IoT healthcare environment. Huixian Qiu et al. [14] applied a dynamic multi-population genetic algorithm to solve multi-objective workflow scheduling in cloud, which simultaneously optimized makespan and energy consumption achieving a better Pareto solution set than two heuristic algorithms (MOHEFT and DCHG-TS) and three evolutionary algorithms (GA-PSO, HEFTGA, and HPSO).

In addition to other scenarios, Fatih Kılıç et al. [65] proposed MPPSO (a novel multi-population-based particle swarm optimization) for feature selection. For the clustering problem, K. Thirumoorthy and K. Muneeswaran [66] proposed ESAMPRO (an elitism-based self-adaptive multi-population Poor and Rich optimization algorithm) to solve grouping similar documents. Babak Rezaei et al. [67] combined genetic local search into a multi-population imperialist competitive algorithm to solve the vehicle routing problem. Hanghao Cui et al. [68] applied a greedy job insertion inter-factory neighborhood structure to improve the multi-population genetic algorithm to solve distributed heterogeneous hybrid flow shop scheduling problem.

According to the review of research, existing algorithms face various challenges in solving UPMDR or other multi-objective optimization problems. Heuristic algorithms are not optimal enough for solving multi-objective problems. Machining learning based on reinforcement learning needs to train for different scenarios and must be retrained when the parameters change. As the number of distributed nodes increases, its transfer learning ability will rapidly decrease. Meta-heuristic algorithms are popular to solve multi-objective optimization problems. Genetic algorithm (GA), multi-population genetic algorithm (MPGA) and their variant are widely applied to solve multidimensional resource optimization problems. They are also some frequently used baselines in optimization problems to evaluate the performance of other algorithms. However, in the multi-dimensional resource optimization of large-scale distributed systems, their convergence and optimality still require to be further improved, which attracts much interest. Therefore, we combine existing state-of-the-art methods and further propose various strategies for improvement. Referring to the previous research, we use the growable genetic algorithm family and multi-population approach to improve the solution of multi-dimensional resource utilization optimization, and then propose MPGGA. In order to support MPGGA, we introduce some novel information interaction strategies between different populations including balanced random mixing, elite supporting and elite sharing, etc. Additionally, we proposed partial growth and growth quota reallocation strategies to make full use of computational force and provide the opportunity for each individual to be optimized by the growth route in GGA. Our proposed MPGGA is also an extension of MPGA, replacing the genetic algorithm with the growable genetic algorithm and adding specific growth quota reallocation strategies. Especially, partial growth and growth quota reallocation strategies are unique features of MPGGA.

3. System model and problem

To assist with the system model and problem formulations, Table 1 lists the descriptions of some notations in this paper.

In this paper, we construct the system model of cloud with multiple types of resources. A diagram of the allocation of tasks or VMs to heterogeneous nodes with multi-dimensional resources is shown in Fig. 1

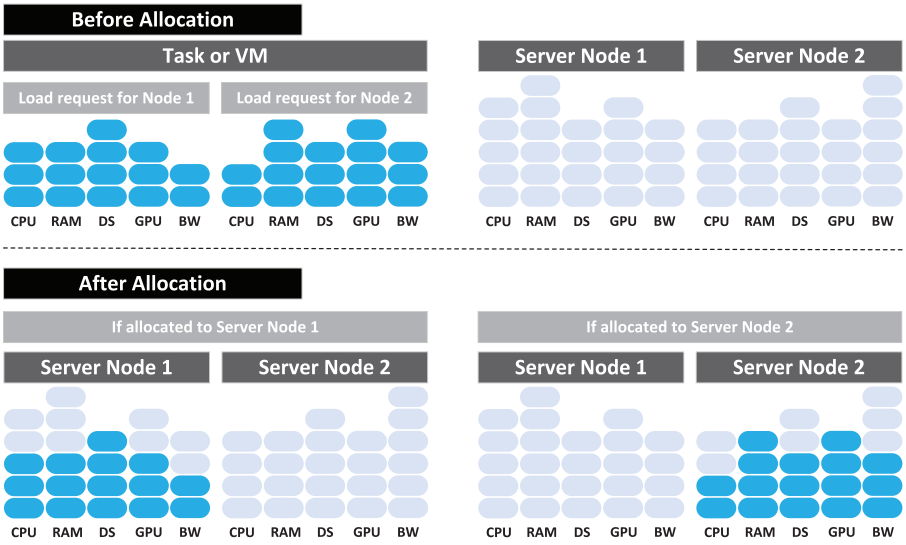


Fig. 1. Allocation of a task or VM to heterogeneous nodes with multi-dimensional resources.

Table 1
Notations and Descriptions.

Notation	Description
n	Number of tasks or VMs
m	Number of nodes
d	Number of dimensions of resources
V_i	The task or VM with index i
N_j	The node with index j
C_{ijk}	The capacity requested of by V_i for the kth dimensional resource in N_j
ψ_j	Set of tasks and VMs in node N_j
κ	The set of ψ_j where $\kappa = \langle \psi_1, \psi_2, \dots, \psi_m \rangle$
x_{ij}	If $V_i \in N_j$ then $x_{ij} = 1$, otherwise $x_{ij} = 0$
L_{jk}	The limited capacity of resource in the kth dimension of the node N_j
S_{jk}	The load of resource in kth dimension of the node N_j
U_{jk}	The utilization rate in kth dimension of the node N_j
u_{ijk}	The resource occupancy rate of V_i for the kth dimension in N_j
Y_{ij}	The vector of V_i for node N_j, i.e., $Y_{ij} = \langle u_{ij1}, u_{ij2}, \dots, u_{ijd} \rangle$
N_p	The number of individuals in each generation of genetic algorithm
N_g	The number of generations in genetic algorithm
G_{step}	The number of search steps of each individual in each generation through HLSA in GGA

where the ellipses are used to indicate the occupancy ratio. The allocation of tasks and VMs will also be related to other scenarios in cloud scheduling, such as virtual machine migration and task offloading, as allocation is one of the foundations of resource scheduling in cloud [69–71]. It can be set that a cloud system has m heterogeneous nodes (denoted as $N = \langle N_1, N_2, \dots, N_m \rangle$) and each node has d-dimensions of resources such as CPU, RAM, disk storage, GPU, bandwidth. The virtual machines (VMs) waiting to be allocated can be set as $V = \langle V_1, V_2, \dots, V_n \rangle$ where n is the number of VMs. When the ith VM is allocated to the jth node, the capacity required for the kth dimension of resource is denoted as C_{ijk} where $1 \le i \le n$, $1 \le j \le m$, $1 \le k \le d$. We denote the set of tasks and VMs in node N_j as ψ_j. If a task or VM V_i is allocated to the node N_j, we use $V_i \in \psi_j$. The ψ_j of each node constitutes a vector $\kappa = \langle \psi_1, \psi_2, \dots, \psi_m \rangle$. Therefore, we can gain the relationships of ψ_j that $\bigcup_{j=1}^{m} \psi_j = V$ and $\psi_j \bigcap \psi_l = \emptyset$ for $\forall 1 \le j \ne l \le m$. κ determines the unique allocation scheme. We use S_{jk} to denote the load of resource in the kth dimension of the jth node. Then, the load vector of the jth node is expressed as $S_j = \langle S_{j1}, S_{j2}, \dots, S_{jd} \rangle$. The occupancy of most components approximately satisfies linear superposition. Thus, resource occupation of each dimension on a node is equal to the sum of the requests of all VMs on it shown as Eq. (1).

$$S_{jk} = \sum_{V_i \in \psi_j} C_{ijk} \tag{1}$$

To better describe the characteristics of the system, the features of nodes and VMs are as follows considering static scheduling or offline reservation scheduling of VMs. The set of VMs is deterministic [72], which means the capacity $\{C_{ijk}\}$ or the utilization $\{u_{ijk}\}$ are given before allocation; All VMs are independent and preemptive without precedence constraints for the order of VMs [72]; Each VM cannot be further split into smaller tasks [35], which means a VM can only be allocated to one node; Each VM can be fully fulfilled by one and only one resource [35]; When $S_{jk} \le L_{jk}$ for $1 \le \forall k \le d$, the processing capacity of the jth node remains unchanged, which means the parameter $\{C_{ijk}\}$ or $\{u_{ijk}\}$ of each VM is fixed. Each node (i.e. server) can process more than one VM; And the number of available nodes is invariant. In fact, online scheduling of every time slot also requires solving UPMDR.

Each node has limited capacity in each dimension (i.e., the maximum load for the healthy operation of components) that can be set as $L = \{L_{jk}\}_{1 \le j \le m, 1 \le k \le d}$. Thus, for a solution of UPMDR, any dimension occupancy of resource in a node must be no larger than its corresponding limited capacity, i.e. for $\forall j$ and $\forall k$,

$$\sum_{V_i \in \psi_j} C_{ijk} \le L_{jk} \tag{2}$$

When L_{jk} and C_{ijk} are given, the utilization for a dimension of resource on a node required by a VM can be obtained as:

$$u_{ijk} = C_{ijk} / L_{jk} \tag{3}$$

Then, Eq. (2) can be rewritten as:

$$U_{jk} = \frac{S_{jk}}{L_{jk}} = \frac{\sum_{V_i \in \psi_j} C_{ijk}}{L_{jk}} \le 1 \tag{4}$$

where U_{jk} represent the total utilization of the kth dimension of resource for the jth node.

There are various indicators to evaluate balancing degrees in the cloud such as variance or standard deviation of load [13,45], average success rate [73], coefficient of variance [74], degree of imbalance [75], etc. In this paper, we leverage the objectives of minimizing the maximum utilization of resources in each dimension, which is a representative issue to ensure that distributed systems have better flexibility to respond to subsequent tasks or VM requests. We can set $\min \omega_k = \min \max \left(U_{1k}, U_{2k}, \dots, U_{mk} \right)$, which means minimizing the utilization of the kth dimension of resource for all nodes. To improve the effectiveness and flexibility of a cloud system, it is necessary to minimize the maximum utilization of resources in all dimensions, to

Table 2
Parameters u_{ijk} of VMs of example.

	V_1			V_2			V_3			V_4		
	CPU	RAM	DS	CPU	RAM	DS	CPU	RAM	DS	CPU	RAM	DS
N_1	0.1	0.3	0.3	0.2	0.3	0.2	0.2	0.1	0.3	0.3	0.2	0.1
N_2	0.2	0.3	0.1	0.1	0.2	0.3	0.0	0.3	0.2	0.2	0.1	0.3

provide services for subsequent VMs with greater robustness. Then, the problem with multi-objective can be written as:

$$\min \omega^{(1)} = \min \begin{cases} \max\left(U_{11}, \ldots, U_{m1}\right) \\ \max\left(U_{12}, \ldots, U_{m2}\right) \\ \ldots \\ \max\left(U_{1d}, \ldots, U_{md}\right) \end{cases} \tag{5}$$

Introducing the matrix $\{x_{ij}\}_{1\le i\le n, 1\le j\le m}$ as the unknown number, the sub-objective can be written as:

$$\min \omega_k = \min\left(\max_{j=1,2,\ldots,m}\left(\sum_{i=1}^{n} x_{ij}u_{ijk}\right)\right) \tag{6}$$

where $x_{ij} \in \{0,1\}$. $x_{ij} = 1$ means the ith VM is allocated to the jth node, i.e., $V_i \in \psi_j$, otherwise $V_i \notin \psi_j$.

The constraints are:

$$\text{s.t.} \begin{cases} \sum_{j=1}^{m} x_{ij} = 1, \sum_{i=1}^{n} x_{ij}u_{ijk} \le 1, x_{ij} \in \{0,1\}, \\ i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,m\}, \\ k \in \{1,2,\ldots,d\} \end{cases} \tag{7}$$

where $\sum_{j=1}^{m} x_{ij} = 1$ means a VM can only be allocated to one node.

In distributed computing systems, reducing the utilization of server nodes within a time slot is beneficial for increasing the flexibility of the system in providing services in subsequent time. The smaller the space occupied by resources in different dimensions, the more timely services can be provided for subsequent new task applications with less network congestion [72,76,77]. It is also beneficial to reduce the subsequent migration volume of virtual machines [78–80], balance the energy consumption of server nodes and reduce the total energy consumption of the system [81,82]. We can illustrate the application relevance of optimization problem $\omega^{(1)}$ in realistic cloud platforms through an example with $n = 4$ VMs, $m = 2$ server nodes and $d = 3$ dimensions of resources. The parameters u_{ijk} of VMs are listed in Table 2. There are 16 optional allocation schemes in the example of Table 2. The optional allocation schemes and their corresponding U_{jk} can be seen in Table 3. From Table 3, the Pareto solution set contains $(0.3, 0.6, 0.5)$, $(0.3, 0.5, 0.6)$, $(0.4, 0.4, 0.5)$, $(0.5, 0.6, 0.3)$ and $(0.5, 0.5, 0.4)$. In real cloud platforms or cloud–edge platforms, server nodes may be deployed in different places. It can be set that the server nodes N_1 and N_2 are far enough apart. If select the scheme of κ_1, the server node N_2 can be idle, which seems to be. However, when a user near node N_1 submits a new task or VM request with more than $(0.2, 0.1, 0.1)$ utilization requirement for N_1, it must be assigned to node N_2, which will consume redundant communication consumption. In addition, if VM migration is temporarily carried out once a new task closed to N_1 arrives, it is still necessary to send the migrated VM to N_2. When multiple users submit tasks at the same time slot (assuming unfortunately these users are all close to N_1 and far away from N_2), it will cause congestion or even paralysis to network communication, thereby increasing user waiting time and actually reducing the efficiency of the entire cloud system. The imbalanced utilization allocation schemes may face a similar risk. In actual cloud management, balancing the utilization of resources in each dimension is beneficial for reducing risks when the parameters for subsequent task submissions are unpredictable. Even if requiring VM migration, the migration volume is relatively lower. If selecting κ_6, i.e., $(0.3, 0.6, 0.5)$, any new task or VM request with the occupancy rate less than $(0.7, 0.4, 0.5)$ for each dimension on any node can be allocated nearby without the need for VM migration or long-distance communication, showing a larger tolerance range. If the occupancy rate of a task on one node is much higher than that of another node, a trade-off between the allocation cost and VM migration cost can guide subsequent schemes. In order to quantitatively illustrate the risk of different schemes, we can use Rt_1 and Rt_2 in Eq. (8) as two indicators for the new task tolerance of the schemes in the example of Table 2:

$$\begin{cases} Rt_1 = \prod_{k=1}^{d}\left(1 - \omega_k\right) \\ Rt_2 = \prod_{j=1}^{m}\prod_{k=1}^{d}\left(1 - U_{jk}\right) \end{cases} \tag{8}$$

Then, we can obtain the Rt for the example and list them in the last two rows of Table 3. It can be seen that the $\omega^{(1)}$-based Pareto solution set obtains better Rt_1 and Rt_2 than other solutions.

In addition to direct optimization of utilization, there is also energy consumption optimization based on utilization of multi-dimensional resources [22,82]. In the experimental evaluation of this paper, we will conduct experiments on the utilization problem ($\omega^{(1)}$) described in this section, and also on two energy consumption-related optimization problems ($\omega^{(2)}$ and $\omega^{(3)}$) based on utilization to verify the adaptability of the proposed algorithm and framework. For the convenience of presentations, we will present the objectives of $\omega^{(2)}$ and $\omega^{(3)}$ in the corresponding experimental section.

4. Methodology: MPGGA

The framework of MPGGA can be seen in Fig. 2. In MPGGA, each population executes a complete five stages of a growing genetic algorithm including initial stage, infancy stage, growth stage, mature stage and genetic stage.

Unlike in the growable genetic algorithm where all individuals will grow through a specific growth route, in one population of MPGGA, only a portion of individuals will enter the growth route, called GGA with partial growth. A certain quota is allocated to each population to select some individuals for growth, which can improve the computational efficiency of devices executing optimization algorithms. The flowchart of GGA with partial growth can be seen in Fig. 3.

Additionally, MPGGA also has the processes of multiple population evolution algorithms. In the mature stage, MPGGA selects a portion of individuals from each population according to a certain standard to enter the management center, where individuals from various populations are fused and adjusted (called information interaction between populations), and then returns the adjusted individuals to each population to participate in subsequent genetic stages.

The MPGGA introduces multi-population strategies to GGA and mainly includes four factors: growable genetic algorithm with partial population growth, multi-population, information interaction strategy and growth quota reallocation strategy. Next, we will provide descriptions of these factors.

4.1. GGA with partial growth

MPGGA follows the GGA [22] as the main part of the algorithm. The research [22] applies the concept of stages to divide the classical genetic algorithm into four stages namely the initialization stage, infancy stage, mature stage and genetic stage. Based on these four stages, GGA adds a growth stage for each individual, whose structure compared with that of classical GA is drawn in Fig. 3. GGA applies the direct growth route (such as the heuristic-based local search algorithm) to improve the infancy individuals and has better convergence speed and optimality than the classical GA.

In GGA, if all individuals are grown up through the direct growth route, it still requires much time. Therefore, GGA can still be improved

Table 3
The optional allocation schemes of example.

		κ_1	κ_2	κ_3	κ_4	κ_5	κ_6	κ_7	κ_8	κ_9	κ_{10}	κ_{11}	κ_{12}	κ_{13}	κ_{14}	κ_{15}	κ_{16}
V_1		N_1	N_1	N_1	N_1	N_2	N_1	N_1	N_2	N_1	N_2	N_2	N_1	N_2	N_2	N_2	N_2
V_2		N_1	N_1	N_1	N_2	N_1	N_1	N_2	N_1	N_2	N_1	N_2	N_2	N_1	N_2	N_2	N_2
V_3		N_1	N_1	N_2	N_1	N_1	N_2	N_1	N_1	N_2	N_2	N_1	N_2	N_2	N_1	N_2	N_2
V_4		N_1	N_2	N_1	N_1	N_1	N_2	N_2	N_2	N_1	N_1	N_1	N_2	N_2	N_2	N_1	N_2
								U_{jk}									
N_1	CPU	0.8	0.5	0.6	0.6	0.7	**0.3**	**0.3**	**0.4**	0.4	**0.5**	**0.5**	0.1	0.2	0.2	0.3	0
	RAM	0.9	0.7	0.8	0.6	0.6	**0.6**	**0.4**	**0.4**	0.5	0.5	0.3	0.3	0.3	0.1	0.2	0
	DS	0.9	0.8	0.6	0.7	0.6	**0.5**	**0.6**	**0.5**	0.4	0.3	0.4	0.3	0.2	0.3	0.1	0
N_2	CPU	0	0.2	0	0.1	0.2	0.2	0.3	0.4	0.1	0.2	0.3	0.3	0.4	0.5	0.3	0.5
	RAM	0	0.1	0.3	0.2	0.3	0.4	0.3	0.4	0.5	**0.6**	**0.5**	0.6	0.7	0.6	0.8	0.9
	DS	0	0.3	0.2	0.3	0.1	0.5	0.6	0.4	0.5	**0.3**	**0.4**	0.8	0.6	0.7	0.6	0.9
								Rt									
Rt_1 (%)		0.2	3	3.2	4.8	4.8	**14**	**16.8**	**18**	**15**	**14**	**15**	5.6	7.2	6	5.6	0.5
Rt_2 (%)		0.2	1.51	1.79	2.42	2.42	**3.36**	**3.29**	**3.89**	**4.05**	**3.92**	**4.41**	2.47	3.23	3.02	2.82	0.5

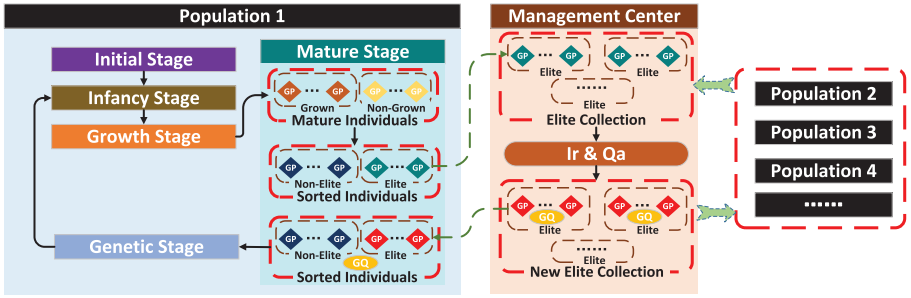


Fig. 2. The framework of MPGGA.

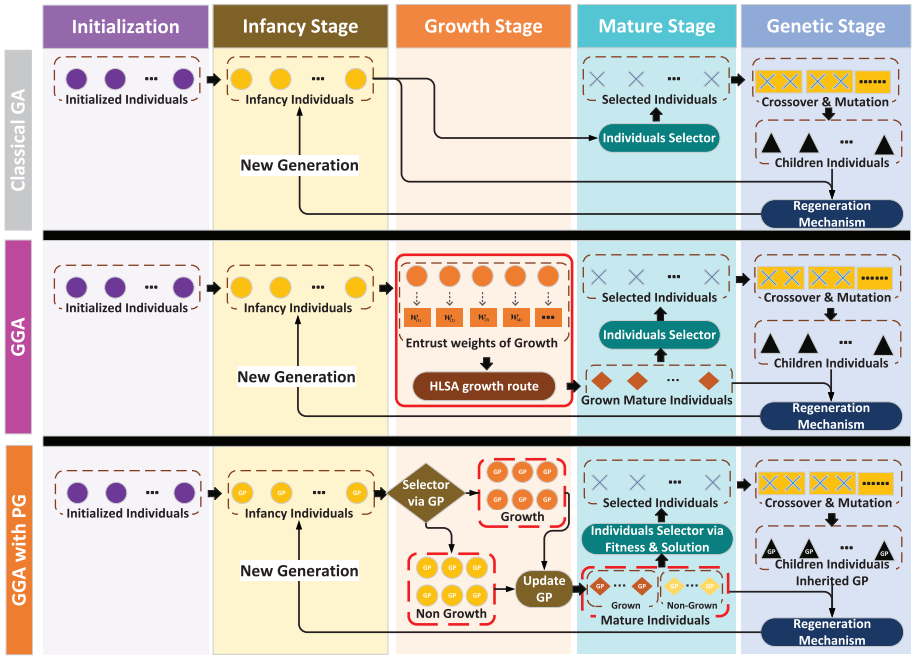
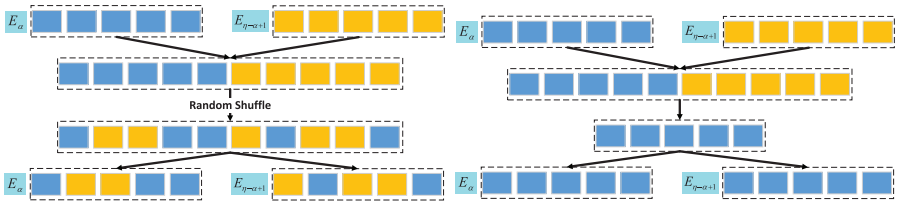


Fig. 3. The flowchart comparison between the classical GA, GGA, and GGA with partial growth (PG).

by partial growth (PG). In the growth stage of each generation, GGA with PG randomly re-selects partial individuals to grow instead of fixing a part of individuals to grow. In fact, if specific individuals are selected and fixed for growth in every generation of GGA, the diversity and overall optimality of the population will deteriorate. This is because the genes of these specific individuals will repeatedly appear during the crossover process so there is an increasing probability that all individuals will gradually have the same genes. Therefore, GGA with PG re-selects the individuals who enter the growth route in each generation and sets the attenuation mechanism for the probabilities of participating in growth (denoted as growth probability, abbreviated as GP) for each individual. In the initial stage, the growth probability of each individual is uniformly distributed, that is every individual has the same opportunity to participate in growth. In the growth stage, when an individual is selected to participate in the growth of a generation, the attenuation mechanism will adjust its probability of



(a) Balanced Randomly Mixing (Ir_1) (b) Elite Supporting (Ir_2) where the score of E_α is higher than $E_{\eta-\alpha+1}$

(c) Elite Sharing (Ir_3)

Fig. 4. Information Interaction Strategies between Populations.

being selected, and the reduced probability will be evenly distributed to other individuals not participating in growth. In the genetic stage, the children individuals will inherit the growth probability of their parents in an average way. The attenuation mechanism can eliminate polarization of the solution set to a certain extent.

Then, the framework of GGA with PG, compared with classical GA and GGA (with all growth), can also be seen in Fig. 3. The process of GGA with PG is as follows.

(1) **Infancy Stage**: At the 1st generation, input the initialized individuals as the infancy individuals with even GP.
(2) **Growth Stage**: Select one part of individuals to enter the growth route through the GP selector. Then, update the GP of each individual according to whether it participates in growth.
(3) **Mature Stage**: After the growth stage, mature individuals have two types of individuals: grown individuals and non-grown individuals. In the mature stage, screened mature individuals by non-dominated sorting and congestion degree sorting for the subsequent crossover and mutation;
(4) **Genetic Stage**: Execute crossover and mutation to generate children, which inherit GP from parents. Then, regenerate the next infancy individuals.
(5) Repeat Infancy Stage → Genetic Stage.

4.2. MP and information interaction strategies

In MPGGA with PG, we define the base components as follows referring to the existing research of GA and MPGA.

(1) **Gene and Individual (Chromosome)**: we define the ith gene as a vector $\lambda_i = \langle x_{i1}, x_{i2}, \ldots, x_{im} \rangle$ where $1 \le i \le n$ and $x_{ij} \in \{0, 1\}$. If $V_i \in \psi_j$, then $x_{ij} = 1$, otherwise $x_{ij} = 0$. A vector $I = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$ with n genes construct an individual (also chromosome) corresponding to a solution κ of the problem.
(2) **Crossover**: The crossover is defined as separately extracting a part of genes from two individuals to gain a new vector as the child individual.
(3) **Mutation**: Mutation is defined as replacing some genes of an individual with randomly generated genes.
(4) **Population**: A group of individuals constructs a population. Generally, the crossover is carried out within a population. The ith population can be set as $P_i = \left\langle I_1^{(i)}, I_2^{(i)}, \ldots, I_{\tau_i}^{(i)} \right\rangle$ where τ_i is the number of individuals in the ith population.
(5) **Regeneration Mechanism**: Each population applies elitist strategy [15] to combine the parent individuals with children individuals to jointly compete to produce the next generation.

With the basic components of MPGGA, an important factor is the interaction mechanism. Different populations exchange information through certain interaction mechanisms to share some optimal solutions. In this paper, we propose a novel group of information interaction strategies through elite solutions competition of different populations.

Some information interaction strategies can be seen in Fig. 4. MPGGA set up a management center to coordinate the entire process of multiple populations. Each population selects partial individuals with the best fitness (for example: according to the sorting strategies of NSGA II or MOEA/D) to enter the center to construct an elite collection E. Then, the center sorts the incoming individuals according to the non-dominated ranking and crowding ranking, scores each population according to the order of these individuals, as well as obtains the sorting of each population. Finally, the center distributes the information of the elite collection to each population according to the overall sorting of each population via a given interaction strategy.

It can be set the sorted populations set from better to worse is $P = \langle P_1, P_2, \ldots, P_\eta \rangle$ and the elite individuals selected from P_α to elite collection are $E_\alpha = \langle e_{\alpha 1}, e_{\alpha 2}, \ldots, e_{\alpha\zeta} \rangle$ where η is the number of populations and ζ is the number of elite individuals from each population to enter the elite collection. Then, several interaction strategies tested in this paper are called non-interaction, balanced randomly mixing, elite sharing and elite supporting, which are described as follows.

(1) **Non-Interaction** (denoted as Ir_0): Non-Interaction means each population evolves independently in the whole process without interaction with other populations.
(2) **Balanced Randomly Mixing** (Ir_1): One-to-one mixing the elite of a better population and a worse population (P_α and $P_{\eta-\alpha+1}$) as a set $\langle e_{\alpha 1}, \ldots, e_{\alpha\zeta}, e_{(\eta-\alpha+1)1}, \ldots, e_{(\eta-\alpha+1)\zeta} \rangle$, and randomly redistributing the individuals of this set to replace the elite of these two populations.
(3) **Elite Supporting** (Ir_2): Using the elite individuals E_α of the α-th ($\alpha \le \eta/2$) population to replace the elite individuals $E_{\eta-\alpha+1}$ in the $(\eta - \alpha + 1)$-th population.
(4) **Elite Sharing** (Ir_3): Using the best ζ individuals in E to replace the elite individuals of all the populations.

Other information interaction strategies are still applicable to MPGGA, and different information interaction strategies may have different diversity and convergence of solution sets. This paper mainly considers the construction of MPGGA and compares its advantages over GGA. Therefore, this paper only tests the performance of these three strategies of information interaction.

4.3. Growth quota reallocation strategies

PG requires a growth quota for each population. MPGGA sets up a global growth quota which means the total number of individuals that can enter the growth routes in each generation of all populations. MPGGA can control the convergence speed and optimality of all populations by adjusting the allocation of the growth quota to each population.

Algorithm 1: MPGGA family with various strategies

Generate the initial individuals and divide them into MPs as $P = \langle P_1, P_2, \dots, P_\eta \rangle$
for *l in range(generation number)* **do**
  Population evolution executor for the P_α **:**
    Randomly select Q_α individuals to enter the growth route according to the GP of individuals, and then obtain the grown individuals by calling Algorithm 2 or A.1
    Update the GP of individuals
    Select the best ζ individuals as the elite individuals E_α and send them to management center of evolutions
  Management center of evolutions:
    Receive the elite individuals from all the populations to construct an elite collection
    Sort the elite individuals in the elite collection and calculate the total score of each population
    Replace the elite individuals of populations based on the specific information interaction strategies
    Readjust the growth quota of populations based on specific quota reallocation strategies
    Send the new elite individuals E_α and growth quota to corresponding populations
  Population evolution executor for the P_α **:**
    Receive the elite individuals to replace the original elite individuals and update the growth quota
    Select and Pair the better mature individuals with specific sort strategies such as non-dominated sorting [15,16] and congestion degree sorting [15]
    Execute crossover and mutation to generate children inheriting the GP from parents
    Regenerate the infancy individuals of the $(l+1)$-th generation
Management center of evolutions:
  Output the Pareto solutions

It can be set the global growth quota as Q and the growth quota allocated to the α-th population is Q_α which constructs a vector as $\langle Q_1, Q_2, \dots, Q_\eta \rangle$. The growth quota reallocation strategy is to generate the growth quota vector $\langle Q_1, Q_2, \dots, Q_\eta \rangle$ according to the elite individuals $E_\alpha = \langle e_{\alpha 1}, e_{\alpha 2}, \dots, e_{\alpha\zeta} \rangle$ of each population P_α. We consider that when different populations evolve into the same generation, their evolutionary states may differ. If all populations retain equal growth quotas, there may be an imbalance in evolution among multiple populations, thereby affecting the performance of the entire genetic system. Overemphasizing the feasibility optimization may also lead to the search falling into local optimum [59]. According to different allocation principles of growth quotas, we propose three quota reallocation strategies as follows which will be tested in the experiments of this paper.

(1) **Balanced Quota Allocation** (denoted as Qa_1): Each population has the same growth quotas, i.e., $Q_\alpha = \lceil Q/\eta \rceil$.
(2) **Priority Supporting the Worse** (Qa_2): The population with the lower ranking can obtain more quota. In the priority support of this paper, we set the $Q_\alpha = \left\lceil \frac{2\alpha}{\eta(\eta+1)} Q \right\rceil$ for $1 \le \alpha \le \eta$.
(3) **Priority Supporting the Better** (Qa_3): It is contrary to Qa_2 and the $Q_\alpha = \left\lceil \frac{2(\eta-\alpha+1)}{\eta(\eta+1)} Q \right\rceil$ for $1 \le \alpha \le \eta$.

4.4. MPGGA and its system architecture

With the above strategies, we can obtain the framework of the MPGGA family as Fig. 2 and its pseudo code as Algorithm 1. In Algorithm 1, the population internal evolution executor corresponds to each population, which mainly performs the evolution process within each population. Corresponding to the number of populations, there are also η population internal evolution executors, which can be implemented through multi-threads or multi-machines (or multi-GPUs).

Algorithm 2: Growth route for one individual in MPGGA: Heuristic-based local search algorithm for heterogeneous nodes using MLSPT as search route

Input : Utilization vectors Y_{ij} for $\forall i, j$, random weight $w = \langle w_1, w_2, \dots, w_d \rangle$, solution κ for the individual, G_{step}
Output: Solution κ corresponding to $I = \langle \lambda_1, \lambda_2, \dots, \lambda_n \rangle$
Set $i = 0$, $Exists\_Ner = True$
while Exists_Ner and $i < G_{step}$ **do**
  $Exists\_Ner = False$, $i++$
  Choose the node N_{j_1} with the largest weighted utilization as $j_1 = \arg\max_{j=1,2,\dots,m} \left(\sum_{i=1}^{n} x_{ij} w \cdot Y_{ij} \right)$ where $w \cdot Y_{ij}$ means the vector inner product
  for $j_2 \ne j_1$ **do** ; `// Call the MLSPT algorithm in following loop`
    Initialize $Mark_{j_1} = 0$, $Mark_{j_2} = 0$, $\psi'_{j_1} = \emptyset = \psi'_{j_2}$
    while $\psi \ne \emptyset$ **do**
      if $Mark_{j_1} \le Mark_{j_2}$ **then**
        $\alpha = j_1$, $\beta = j_2$
      else
        $\alpha = j_2$, $\beta = j_1$
      Collect tasks $V_\tau \in \psi$ **s.t.** $w \cdot \left(Y_{\tau\alpha} - Y_{\tau\beta} \right) = \min_{V_i \in \psi} w \cdot \left(Y_{i\alpha} - Y_{i\beta} \right)$ to obtain a set of $\left\langle V_{\tau_1}, V_{\tau_2}, \dots, V_{\tau_s} \right\rangle$
      if $s \ge 2$ **then**
        Choose V_τ **s.t.** $w \cdot Y_{\tau\alpha} = \max_{1 \le p \le s} w \cdot Y_{\tau_p \alpha}$
      $Mark_\alpha += w \cdot Y_{\tau\alpha}$, $\psi'_\alpha += \{V_\tau\}$ and $\psi - = \{V_\tau\}$
      $\kappa_{j_1 j_2} = \kappa - \psi_{j_1} - \psi_{j_2} + \psi'_{j_1} + \psi'_{j_2}$
    if $\exists j_2 \ne j_1$ ***s.t.*** *the solution of* $\kappa_{j_1 j_2}$ *is optimal than* κ **then**
      $Exists\_Ner = True$
      Choose the optimal $\kappa_{j_1 j_2}$ to update $\kappa = \kappa_{j_1 j_2}$

In this paper, the growth route is also a Heuristic-based local search algorithm for heterogeneous nodes using Modified LSPT as search route as Algorithm 2 [22] where

$$\omega_{(w)} = \min \left(\max_{j=1,2,\dots,m} \left(\sum_{i=1}^{n} x_{ij} \sum_{k=1}^{d} \left(w_k \cdot u_{ijk} \right) \right) \right) \tag{9}$$

and $Y_{ij} = \langle u_{ij1}, u_{ij2}, \dots, u_{ijd} \rangle$. Eq. (9) utilizes weight summation to degrade multi-objective optimization problems to ensure that individuals can engage in additional growth through modified LSPT. The modified LSPT-based neighborhood can be defined as follows. It can be assumed that κ' is an HLSA-based neighbor of κ i.e., only two nodes have different VMs, $\psi'_{j_1} \cup \psi'_{j_2} = \psi_{j_1} \cup \psi_{j_2} = \left\{ V_{\tau_1}, V_{\tau_2}, \dots \right\}$ and $\xi_{\tau_i} \ge \xi_{\tau_{i+1}}$ where $\xi_{\tau_i} = w \cdot \left(Y_{\tau_{j_1}} - Y_{\tau_{j_2}} \right)$. If $V_{\tau_i} \in \arg\min_{\psi'} \left(\sum_{V_{\tau_k} \in \psi'_1} w \cdot Y_{\tau_{j_1}}, \sum_{V_{\tau_k} \in \psi'_2} w \cdot Y_{\tau_{j_1}} \right)$ where $k < i$ for $\forall V_{\tau_i} \in \psi'_{j_1} \cup \psi'_{j_2}$, then κ' can be called the modified LSPT-based neighbor of κ. While by contraries κ may not be that of κ'.

Table 4
GGA with various combinations of strategies.

Category	Models	MP	IR	QA
GGA	$Model_1$			
MPGGA with Non	$Model_2$	✓		
MPGGA with IR	$Model_3$	✓	Ir_1	
	$Model_4$	✓	Ir_2	
	$Model_5$	✓	Ir_3	
MPGGA with QA	$Model_6$	✓		Qa_1
	$Model_7$	✓		Qa_2
	$Model_8$	✓		Qa_3
MPGGA with IR & QA	$Model_9$	✓	Ir_1	Qa_2
	$Model_{10}$	✓	Ir_1	Qa_3
	$Model_{11}$	✓	Ir_2	Qa_2
	$Model_{12}$	✓	Ir_2	Qa_3
	$Model_{13}$	✓	Ir_3	Qa_2
	$Model_{14}$	✓	Ir_3	Qa_3

Algorithm 2 is a version convenient for comprehension. In the realistic program of the algorithm, we can use array operations on GPUs to accelerate the growth route and run MPGGA algorithm program in a distributed environment. For genetic algorithms to solve resource optimization problems, the main processes consuming computational time include fitness calculation, crossover and mutation processes for each individual in each generation (the details are provided in Appendix A). Resources optimization in cloud computing especially needs to consider large-scale scenarios. Using a single CPU for launching optimization algorithms will consume excessive calculational time, which is not conducive to executing extensive instances for comprehensive evaluation. Thus, in this paper, we will execute MPGGA and baseline algorithms on GPUs, which can leverage GPU-based matrix operations to accelerate the execution speed of algorithms. GPU-based genetic programming is also one of the current hotspots in meta-heuristics [26,83,84].

With various strategies of MPGGA, we can get different models under different strategy combinations as shown in Table 4.

In this section of experiments, we will verify the performance of each model through experiments in UPMDR to determine policy options and also compare our proposed algorithm to state-of-the-art methods.

5. Performance evaluation in UPMDR of cloud

5.1. Experiments setting

In this section, we mainly present and discuss the experimental results of MPGGA algorithm in the UPMDR problem of cloud computing as Eq. (5). Additionally, our proposed MPGGA framework also adapts to other problems considering multi-dimensional resources. To supplement the analysis to the adaptability of our proposed MPGGA framework, we provide partial experimental results of MPGGA for solving multi-objective asymmetric traveling salesman problems (MoATSP) in Appendix B and some results of MPGGA for energy consumption-related problems in Ex_7. For the sake of the comprehensive evaluations of MPGGA, we carry out seven groups of experiments from various aspects including:

(1) Ex_1: evaluation of the MP strategy without other strategies;
(2) Ex_2: comparison of information interaction strategies;
(3) Ex_3: comparison of quota reallocation strategies;
(4) Ex_4: comparison of various combinations of strategies;
(5) Ex_5: evaluation of different growth quotas;
(6) Ex_6: comparison with the state-of-the-art methods for the problem of minimizing the maximum utilization of resources in all dimensions;
(7) Ex_7: comparison with the state-of-the-art methods for the energy consumption-related problems to evaluate the adaptability of MPGGA.

All these experiments are based on the control variable method. Ex_1 fixes the structure of the algorithm as GGA and varies the number of populations. Ex_2 fixes the structure of the algorithm as MPGA and varies the information interaction strategies. Ex_3 fixes the structure of the algorithm as MPGGA and varies the quota reallocation strategies. Ex_4 compares the performance of different combinations of information interaction and quota reallocation strategies. Ex_5 fixes the structure of the algorithm as MPGGA with Ir_3 (elite sharing) and Qa_2 (priority supporting the worse) and varies the growth quotas. When growth quota $Q = 0$, MPGGA will degrade to multi-population NSGA II (MP-NSGA II). Thus, the comparison between $Q > 0$ and $Q = 0$ in Ex_5 is also equivalent to the comparison between MPGGA and MP-NSGA II. Ex_6 compares our proposed MPGGA with baselines for the problem of minimizing the maximum utilization of resources in all dimensions. Ex_7 compares our proposed MPGGA with baselines for the problems of minimizing the maximum energy consumption of each server node ($\min \omega^{(2)}$) and minimizing the total energy consumption of the whole system ($\min \omega^{(3)}$) to evaluate the adaptability of MPGGA to scenarios and optimization objectives. Considering that the comparison algorithms need to be representative and compatible with the multi-dimensional resources optimization problem studied in this paper, we choose various state-of-the-art methods as baselines, including some existing well-performed multi-dimensional resource optimization algorithms (GHW-NSGA II, GHW-MOEA/D, NSGA II and MOEA/D) and some existing advanced multi-population evolutionary algorithms (MP-NSGA II, MP-MOEA/D, and their variants, e.g., MP-NSGA II-ES, MP-MOEA/D-ES, MP-NSGA II-BM, MP-MOEA/D-BM).

These experiments are executed on the random simulation dataset. In the simulation, we set up the server nodes of cloud to be heterogeneous and the parameters of VMs obey uniform distribution:

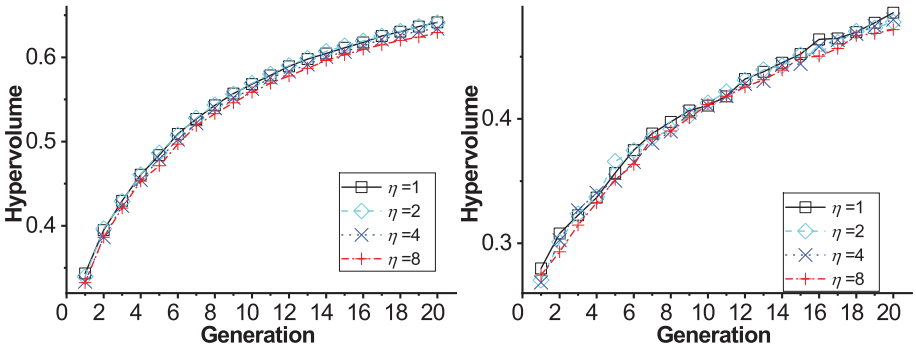
$$u_{ijk} \sim \mathrm{U}(2, 10)\,\%, \tag{10}$$

which are generated by torch.randint(low=20, high=100, size=(n, d, m))/1000, where $\mathrm{U}(2, 10)$ means the uniform distribution in the region $[2, 10]$. In simulated datasets with other ranges or the public dataset (such as AzureTraceforPacking2020 [76]), the experimental comparison results are similar. Therefore, this paper mainly provides results on simulation datasets in ablation studies and parameter lectotype studies (Ex_2 to Ex_5), and only provides experimental results on public trace (AzureTraceforPacking2020) in Ex_6 to supplement the usability of the proposed algorithm for realistic scenarios.

The main multi-objective indicators for measuring multi-dimensional resource utilization optimization in this paper include the Pareto solution set, C indicator and HyperVolume (HV) [85,86] of $\omega^{(1)}$ with multi-dimensional resources utilizations. For the sake of the evaluation, we choose three dimensions of resources, i.e., CPU, RAM and DS, to execute the experiments. In order to maintain the comparability of the results, we use the absolute HV of these three-dimensional resources' utilization by calling the function **pymoo.indicators.hv. Hypervolume** [87], whose settings are as ref_points = $(1, 1, 1)$, zero_to_one = False, ideal = $\left(\min_j U_{j1}, \min_j U_{j2}, \min_j U_{j3}\right)$, nadir = $\left(\max_j U_{j1}, \max_j U_{j2}, \max_j U_{j3}\right)$. HV of $\omega^{(1)}$ converts the vector with maximum utilization rates of multi-dimensional resources (a multi-objective) into a scalar indicator, which is often used to measure the optimality and also able to evaluate the convergence of multi-objective optimization algorithms [85,86]. In experiments, we also use Wilcoxon rank-sum test and Friedman test to evaluate the significance of the performance differences between baseline algorithms and our proposed MPGGA.

We uniformly set the mutation rate as 0.2 and growth steps in GGA as $G_{step} = 10$. In order to show more coverage of results, we try to present them corresponding to different combinations of (n, m).

For optimization problems in large-scale scenarios, using the CPU to execute evolutionary algorithms will usually cost far more time. To generate extensive instances for comprehensive evaluation and discussions of our proposal, we execute MPGGA and baseline algorithms



(a) $(n, m) = (500, 200)$ (b) $(n, m) = (2000, 1000)$

Fig. 5. The HV-over-generation for MPGGA-without-additional-strategies under different number of populations: all growth, non information interaction, non growth quota reallocation, $N_p \times \eta = 800$, $\eta \in \{1, 2, 4, 8\}$, $N_g = 20$.

on GPUs in experiments, which is also beneficial for evaluating the performance of algorithms at the current advanced computing power level. Then, the experiments are launched on a cluster environment with multiple servers or desktops. The center server (or desktop) acts as the management center of MPGGA. All algorithms (including baseline algorithms and our proposed algorithms) participating in comparative experiments are running on the same number of GPUs (in the same server or desktop) to ensure that they are compared in the environment with the same computing power. Except for specified explanations, the experiments are performed on servers (GPU cluster of V100). The main configurations of the cluster environments are as follows.

- Communication Network: Gigabit, Full Duplex;
- Communication Backend: Gloo;
- Program version: Python 3.7, Pytorch 1.10.2, Numpy 1.21.6, Pymoo 0.6.0.1;
- Servers (GPU Cluster of V100):
  - Operation System: Ubuntu 18.04.5;
  - CPU: Intel i9 10850K @ 3.6 GHz, 10 cores;
  - SSD: Samsung 980 NVMe M.2 @ 1TB;
  - RAM: LPX 64 GB DDR4 3200;
  - GPU: NVIDIA TESLA V100 @ 32 GB × 2, where × 2 means one server has two GPUs).
- Desktops (GPU Cluster of RTX 3060 Ti):
  - Operation System: Windows 10;
  - CPU: Intel i5 13600KF @ 3.5 GHz, 14 cores;
  - SSD: P7000Z NVMe M.2 SSD @ 2TB;
  - RAM: LPX 64 GB DDR4 3200;
  - GPU: NVIDIA GeForce RTX 3060 Ti @ 8 GB × 2.

5.2. Ex_1: Evaluation of the MP strategy

Firstly, we evaluate the effect of the MP strategy without additional information interactions by comparing $Model_2$ to $Model_1$. All individuals will participate in the growth route. Considering experiments for various combinations of (n, m) have a similar conclusion, we only present two groups of experiments respectively under $(n, m) = (500, 200)$ and $(n, m) = (2000, 1000)$. We change the number of populations from 2^0 to 2^3. Because all individuals will grow through the growth route, the time consumed by each generation of the algorithm program is independent of the population number. Therefore, we use the generation as the abscissa and plot the results under the different number of populations in Fig. 5.

As shown in Fig. 5, with the increase of generation, the growth trends of HVs corresponding to each population number $\eta = 1, 2, 4, 8$ are very close. This indicates that only increasing the number of populations without additional information interactions between populations has no obvious impact on the convergence rate of MPGGA. This also laterally reflects the significance of information interactions between populations in MPGGA.

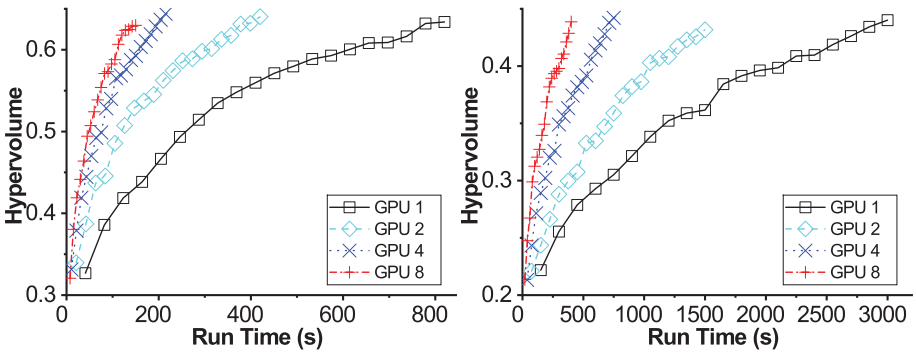
Then, we evaluate the performance of the proposed system architecture to execute MPGGA which is shown in Fig. A.1. We set the number of individuals $N_p = 100$ for each population, the number of populations is $\eta = 8$, and all individuals can be improved through the growth stage. Then, we execute the MPGGA respectively on 1 GPU, 2 GPUs, 4 GPUs and 8 GPUs. As the architecture with multi GPUs mainly focuses on improving the computing speed of MPGGA, we use the running time as the abscissa. Then, we plot the HVs-over-time for the scenarios $(n, m) = (500, 200)$ and $(n, m) = (2000, 800)$ respectively under different sizes of GPUs in Fig. 6.

The overall trend in Fig. 6 shows that increasing the number of GPUs can improve the computation speed of MPGGA algorithm. In each generation, the HVs corresponding to different numbers of GPUs are very close, which shows that the number of GPUs has no effect on the evolution process of each generation. In detail to $(n, m) = (500, 200)$, it takes 820 s for 1 GPU to implement 20 generations, 420 s for 2 GPUs, 215 s for 4 GPUs and 150 s for 8 GPUs. From 1 GPU to 4 GPUs, the time is approximately inversely proportional to the number of GPUs. In this range, the acceleration effect of multiple GPUs is more obvious than that for 8 GPUs. From 4 GPUs to 8 GPUs, the time is not inversely proportional. This may be because, in addition to the time required for the growth route of each generation, the selection at the mature stage, the crossover and regeneration at the genetic stage will also consume a certain time, which is less affected by the number of GPUs than growth route.

Additionally from Fig. 6, we can observe that increasing the number of GPUs may reduce the efficiency of GPUs, so we choose to apply 2 GPUs for the subsequent experiments, which does not affect the conclusion of comparative experiments on different strategies and algorithms. In fact, as long as the memory of the GPU is sufficient and the requirements for computing speed are relaxed, only one GPU is needed to perform the evolution process of multiple populations. However, a phenomenon in our experiments with large-scale tasks and resource counts is that a single GPU is not sufficient to support the simultaneous evolution and computation of 8 populations. Thus, we select 2 GPUs for the experiment. In the experiment, all baseline algorithms and our proposed methods were running on the same number of GPUs to ensure that they are compared in the environment with same computing power. If there are more GPUs, the calculation process can still be accelerated accordingly. However, more GPUs may introduce additional communication consumption, thus it may not necessarily fully comply with linear acceleration.

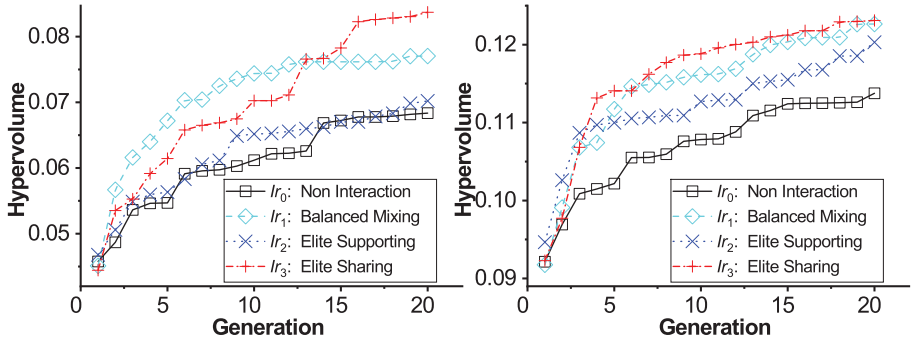
5.3. Ex_2-Ex_4: Ablation studies

To test the performance of different information interaction strategies, we carry out a group of experiments (Ex_2) by comparing $Model_2$



(a) $(n, m) = 500, 200)$ (b) $(n, m) = (2000, 800)$

Fig. 6. The hypervolume-over-time for MPGGA-without-additional-strategies under the different size of GPUs: all growth, non information interaction, non growth quota reallocation, $N_p = 100$ for each population, $\eta = 8$, $N_g = 20$, $GPUs \in \{1, 2, 4, 8\}$.



(a) $(n, m) = 4000, 1000)$ (b) $(n, m) = (8000, 3000)$

Fig. 7. The HV-over-generation for MPGA under different interaction strategies: non growth, various information interactions, non growth quota reallocation, $N_p = 100$ for each population, $\eta = 8$, $N_g = 20$.

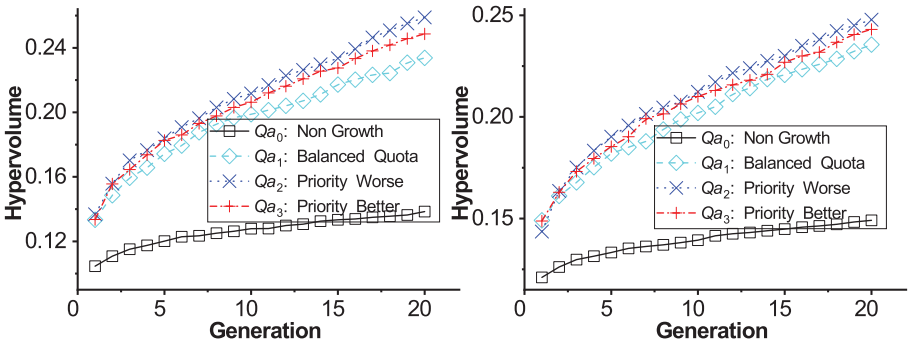
to $Model_3$, $Model_4$ and $Model_5$. In this group of experiments, we set non-growth for all individuals (MPGA), $N_p = 100$ for each population, $\eta = 8$ and $N_g = 20$. Because different information interaction strategies do not affect the time of each generation, we also use generation as the abscissa. Similarly, a large number of experiments will draw the same conclusion, thus we only present two scenarios of $(n, m) = (4000, 1000)$ and $(n, m) = (8000, 3000)$, which have sufficient virtual machines and nodes for a representative conclusion. Then, we plot the HV-over-generation for MPGA under different interaction strategies in Fig. 7.

For the instance of Fig. 7(a), the HVs of Ir_1 (balanced mixing) before the 13-th generation are higher than that of Ir_3 (elite sharing), while Ir_3 (elite sharing) achieves the highest HVs at the 20-th generation. For the instance of Fig. 7(b), the curve of Ir_3 (elite sharing) is higher than others from the 4-th generation to the 20-th generation. This may be because Ir_3 shares the better solutions to all populations so that each population can evolve the solutions on the basis of the better solutions to accelerate the optimizations. From the two instances of Fig. 7, the curves of Ir_1 and Ir_3 are higher than that of Ir_2 (elite supporting). This illustrates directly using the elite of the better populations to replace that of the worse one cannot effectively interact information between populations. This may be because Ir_2 (elite supporting) abandons some appreciable solutions of the lower ranking population accompanied by the reduction of population diversity, resulting in a reduction in the convergence speed of the optimization solution. In Fig. 7, the curves of Ir_1, Ir_2 and Ir_3 are obviously higher than that of non-interaction, which demonstrates applying the information interaction strategies can improve the convergence speed of the evolution process in MPGGA to a certain extent. Other unlisted experiments still have similar conclusions. The comparative experiments of information interaction strategies also illustrate again the necessity of information interaction in MPGGA.

MPGGA, as one of the variants of MPGA, also conforms to the effectiveness of information interaction strategies. The difference is that some individuals in MPGGA can get extra improvement through the growth route. Then, we continually evaluate the performance of quota reallocation strategies by comparing $Model_2$ to $Model_6$, $Model_7$ and $Model_8$. In this group of experiments (Ex_3), we set total growth quota $Q = 80$, non interaction strategy, $N_p = 100$ for each population, $\eta = 8$ and $N_g = 20$. The different growth quota reallocation strategies also do not affect the time of each generation for the whole population set, thus we use generation as the abscissa. Similarly, as the conclusion is representative, we only present two scenarios of $(n, m) = (3000, 1000)$ and $(n, m) = (5000, 2000)$. Then, we plot the HV-over-generation for MPGGA under different growth quota reallocation strategies in Fig. 8.

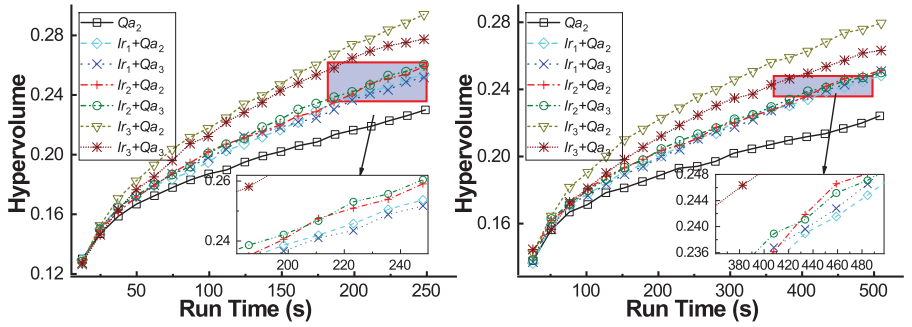
From Fig. 8, the curves of non-growth are far lower than that with growth quotas, which illustrates extra growth of populations has significant improvement to the convergence and optimality. Among the three growth quota reallocation strategies, Qa_2 corresponds to the highest curves, followed by Qa_3 and Qa_1. This demonstrates preferably giving additional growth quotas to the populations with lower ranking can improve the overall convergence speed. This may be because balanced allocation or priority better may cause some populations to enter the local optimum prematurely and priority worse strategy can make full use of the limited growth quotas.

Combining the results of Figs. 7 and 8, we can roughly predict that the combination of Ir_3 and Qa_2 in MPGGA will have better performance than others. To validate this, we carry out a group of experiments (Ex_4) by comparing $Model_9$-$Model_{14}$. In Ex_4, we change the combinations of interaction strategies and growth quota strategies. Other parameters, including total growth quotas, number of populations, number of generations, number of individuals of each population, etc, are the same as the experiments of Figs. 7 and 8. We present the



(a) $(n, m) = (3000, 1000)$ (b) $(n, m) = (5000, 2000)$

Fig. 8. The HV-over-generation for MPGGA under different growth quota reallocation strategies: partial growth with total quota $Q = 80$, non information interactions, various growth quota reallocation, $N_p = 100$ for each population, $\eta = 8$, $N_g = 20$.



(a) $(n, m) = (3000, 1000)$ (b) $(n, m) = (5000, 2000)$

Fig. 9. The HV-over-time for MPGGA under different combinations of information interaction and growth quota reallocation strategies: partial growth with total quota $Q = 80$, various information interactions, various growth quota reallocation, $N_p = 100$ for each population, $\eta = 8$, $N_g = 20$.

results of two instances of $(n, m) = (3000, 1000)$ and $(n, m) = (5000, 2000)$, and use the run time as the abscissa. Then, we plot the HV-over-time for MPGGA under different combinations of information interaction strategies and growth quota reallocation strategies in Fig. 9.

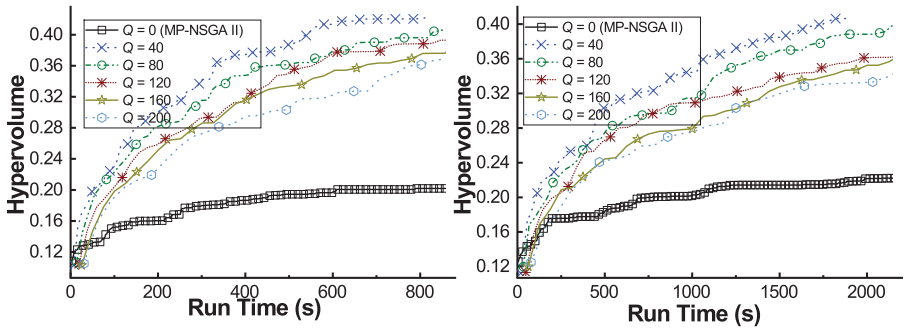
The curves in Fig. 9 can be divided into four grades from top to bottom: first includes $Ir_3 + Qa_2$ and $Ir_3 + Qa_3$, second is $Ir_1 + Qa_2$ and $Ir_1 + Qa_3$, third is $Ir_2 + Qa_2$ and $Ir_2 + Qa_3$, as well as the last is Qa_2 ($Ir_0 + Qa_2$). The order of these four grades is basically consistent with the ranking of information interaction strategies in the experiments of Fig. 7. In each grade, the results corresponding to Qa_2 are better than those corresponding to Qa_3, which is consistent with the conclusion of the experiments in Fig. 8. This grading shows that the gap between different information interaction strategies is greater than the gap between different quota reallocation strategies. This not only demonstrates the importance of information interaction strategies but also laterally shows that the growth quota proposed in this paper has greatly improved MPGGA, because the closer to the theoretical optimal solution in the optimization problem, the more difficult it is to be further optimized. As shown in Fig. 9, the curves corresponding to the $Ir_3 + Qa_2$ are the highest, which validates again that the combination of Ir_3 and Qa_2 in MPGGA has better performance than others. Based on this conclusion, we select $Model_{13}$ (with $Ir_3 + Qa_2$, abbreviated as MPGGA-ESPW) as the practical strategies combination in the following experiments.

5.4. Ex_5: Evaluation of growth quotas

In the above experiments, we observe that the performance of $Model_{13}$ (MPGGA-ESPW) was better than other combinations of information interaction and growth quota reallocation, so we conducted incremental experiments on the growth quota based on $Model_13$ to observe the performance of $Model_{13}$ (MPGGA-ESPW) under different growth quotas, so as to verify the advantages of partial growth strategy. Referring to Taguchi methods [88,89], the experimental process of Ex_1 to Ex_5 can also be used to determine the algorithm parameters for specific optimization scenarios and problems.

With the increase in growth quota, MPGGA will spend more time completing 100 generations. Thus, we only present the time regions when MPGGA with 40 growth quota completes 100 generations. Then, we plot the results for the scenarios of $(n, m) = (3000, 1000)$ and $(n, m) = (5000, 2000)$ in Fig. 10.

From Fig. 10, $Q = 40$ achieves the highest HV-over-time. From $Q = 0$ to $Q = 40$, the curve has a significant rise, which shows that growth quota has a significant improvement effect on MPGGA. As the growth quota increases from 40 to 200, the curves gradually decrease. This proves that more growth quotas are not necessarily better. In fact, compared with the process of conventional genetic algorithms, the local search algorithm corresponding to the growth route often consumes a lot of computing time, thus excessive growth quotas will consume additional computing force. While, if there is no additional growth route, the search process of GA has great uncertainty, resulting in the slow evolution of each generation, and then may also consume redundant computational force. The genetic algorithm with partial growth quotas combines the advantages of both. It allocates limited computational force to make some individuals gain additional growth through HLSA [22], and these grown individuals participate in the crossover and mutation to help the population evolve. Additionally, when the growth quota $Q = 0$, the growable genetic algorithm will degenerate into the conventional genetic algorithm, which indicates that MPGGA-ESPW with 0 growth quota equals multi-population NSGA II (MP-NSGA II). Thus, comparing the results of $Q = 0$ and $Q > 0$ in Fig. 10 can also demonstrate our proposed MPGGA has better performance in terms of convergence and optimality than the other



(a) $(n, m) = 3000, 1000)$ (b) $(n, m) = (5000, 2000)$

Fig. 10. The HV-over-time for $Model_{13}$ (MPGGA-ESPW) with different growth quotas: partial growth with different total growth quotas $Q = 0, 40, 80, \ldots, 200$, elite sharing ($Ir_3$) information interaction, priority supporting the worse (Qa_2) growth quota reallocation, $N_p = 100$ for each population, $\eta = 8$, $N_g = 100$, 5 skip points for the symbols in the curves. When the growth quota $Q = 0$, MPGGA-ESPW equals multi-population NSGA II (MP-NSGA II).

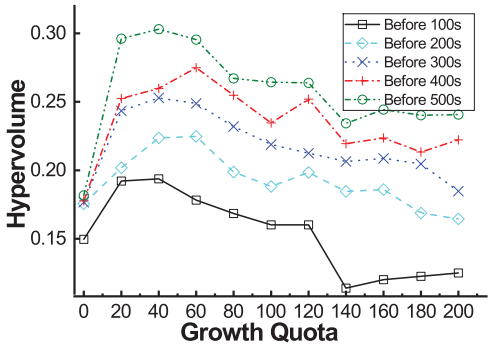


Fig. 11. HV with respect to growth quota for different each time profile under the scenario of Fig. 10(b) $(n, m) = (5000, 2000)$.

multi-population evolutionary algorithm (i.e., MP-NSGA II). This indicates that GGA has a greater advantage than NSGA II when combined with the multi-population strategy.

To further observe the effect of growth quota on performance, we intercept five time-profiles 100 s, 200 s, 300 s, 400 s and 500 s of Fig. 10(b), record the HV of each growth quota corresponding to the nearest generation of individuals before time profile, and then plot the curve of HV with respect to growth quota for different each time profile in Fig. 11.

It can be clearly seen from Fig. 11 that with the increase of growth quota, HV shows a trend of first increasing and then decreasing in each time profile. The HV of $Q = 60$ is highest at the 400 s profile and that of $Q = 40$ is highest at other profiles. The possible reason for this is that the iteration process for different time profiles varies with the growth quotas resulting in certain volatility. While it does not affect to draw an important conclusion: there is the most appropriate growth quota of MPGGA at each time point, i.e., limited time corresponds to an appropriate allocation of computing force. In addition, this also verifies the advantages of partial growth of MPGGA in this paper.

5.5. Ex_6: Comparison with the state-of-the-art for the problem $\min \omega^{(1)}$

To further evaluate the advantages of our proposed algorithms in the problem $\min \omega^{(1)}$ (minimizing the maximum utilization of resources in each dimension), we execute a group of experiments Ex_6 comparing the MPGGA with the state-of-the-art in the terms of convergence. According to the above experiments, we select the strategies combination $Ir_3 + Qa_2$, (ESPW), which outperforms other combinations, to participate in the comparison with the state-of-the-art.

In research [22], it has demonstrated that GHW family (including GHW-NSGA II and GHW-MOEA/D) outperforms two state-of-the-art methods NSGA II and MOEA/D in solving the problem $\min \omega^{(1)}$. Therefore, we also regard GHW family as the state-of-the-art in this group of experiments in addition to NSGA II and MOEA/D. The number of total individuals is set as 800 for these algorithms. The MPGGA also includes $\eta = 8$ populations and each of them has 100 individuals. These algorithms are all executed on two GPUs respectively with 4 processes. To observe the algorithm performance over a long time, we set 100 generations for MPGGA and GHW family. Because the NSGA II and MOEA/D have no additional growth, which makes each generation consume less time than MPGGA and GHW family, we set 2000 generations for NSGA II and MOEA/D. Then, we only present the indicators of each algorithm in the time corresponding to the 100 generations of MPGGA.

To increase the coverage of the experimental results, we show the results of four scenarios: two have less size of VMs and nodes that $(n, m) = (500, 100)$ and $(n, m) = (1000, 300)$; as well as two have larger size that $(n, m) = (5000, 1500)$ and $(n, m) = (6000, 2500)$. Then, we plot the results of these scenarios in Fig. 12.

From the overall trend of the curves in Fig. 12, the HVs of the algorithm MPGGA-ESPW proposed in this paper are significantly higher than that of the compared algorithms in the time range shown in the figures. The concrete order of these algorithms is: MPGGA-ESPW is better than GHW family, followed by NSGA II and MOEA/D. This not only proves the superiority of the proposed algorithm MPGGA-ESPW in terms of optimality and convergence compared with state-of-the-art in both small and large-scale scenarios but also shows that the introduction of multiple populations with its counterpart information interaction strategies and partial growth quota reallocation strategies based on GGA can further improve the convergence speed.

Concretely, we list the generations and HVs of each algorithm corresponding to the final time of each sub-figure in Table 5. Taking $(n, m) = (500, 100)$ as the example: It can be calculated that with 291 s the average increment of HV per generation is 0.00524 for algorithm MPGGA, 0.0225 for GHW-NSGA II, 0.0229 for GHW-MOEA/D, 0.00056 for NSGA II and 0.00051 for MOEA/D. This means that the average increment of GHW per generation is the largest, followed by MPGGA. However, the average time consumed by each generation of GHW is 14.10 s for GHW-NSGA II and 14.00 s for GHW-MOEA/D, which is the longest, also followed by MPGGA (2.91 s). Thus, MPGGA is actually to minimize the time consumption of each generation and simultaneously enhance the increment of HV per generation. According to the HV of the four experiments, the average convergence speed of MPGGA is 1.363, 1.339, 1.948 and 2.151 times that of GHW-NSGA II, GHW-MOEA/D, NSGA II and MOEA/D respectively.

To evaluate the significance of differences between baselines and our proposed algorithm, we can use Wilcoxon rank-sum test. The Wilcoxon rank-sum test tests the null hypothesis that two sets of measurements are drawn from the same distribution. At the 5% test

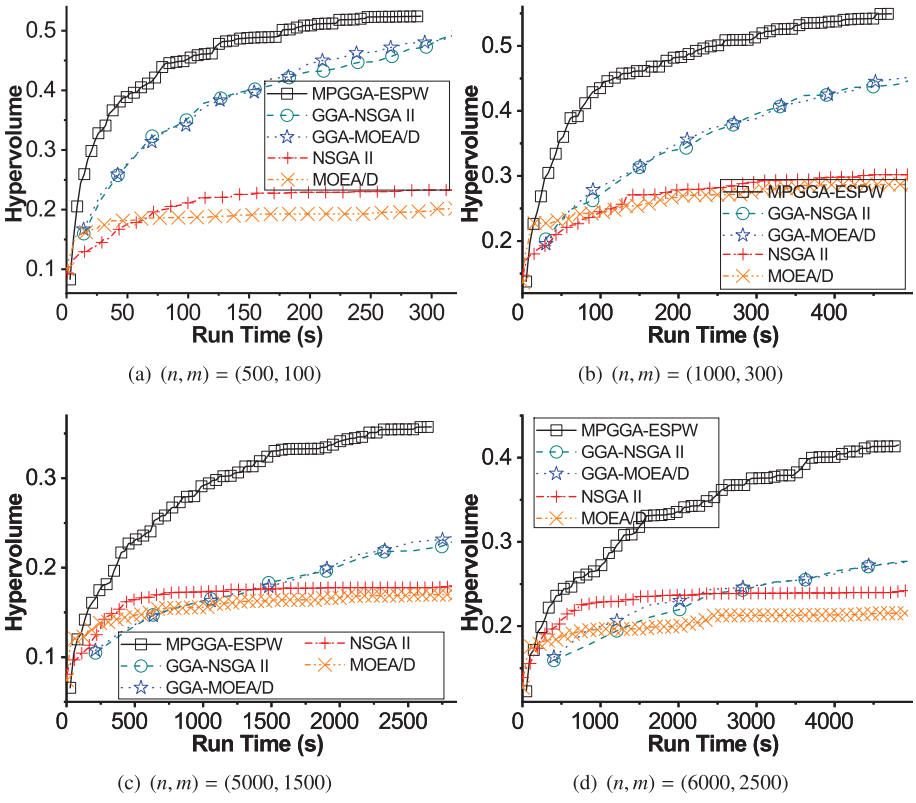


Fig. 12. The HV-over-time of MPGGA-ESPW compared with baselines GHW-NSGA II, GHW-MOEA/D, NSGA II, MOEA/D: partial growth with growth quotas $Q = 80$, elite sharing (Ir_3, ES) information interaction, priority supporting the worse (Qa_2, PW) growth quota reallocation, $N_p = 100$ of each population, $\eta = 8$, $N_g = 100$ for MPGGA; $N_p = 800$ for GHW-NSGA II, GHW-MOEA/D, NSGA II, and MOEA/D; 5 skip points for the symbols in the curves of MPGGA-ESPW, GGA-NSGA II and GGA-MOEA/D, 20 skip points for NSGA II and MOEA/D in figures; all algorithms are executed on two GPUs using 8 processes.

Table 5
The generations and HVs of each algorithm corresponding to the final time of each sub-figures in Fig. 12, where ratio means $\mathrm{HV}^{MPGGA}/\mathrm{HV}^{X}$ and X is the corresponding row's algorithm.

Algorithms	(500, 100)				(1000, 300)				(5000, 1500)				(6000, 2500)				Ave. Ratio
	Gen.	Time	HV	Ratio	Gen.	Time	HV	Ratio	Gen.	Time	HV	Ratio	Gen.	Time	HV	Ratio	
G-NSGA II	21	296	0.473	1.107	16	479	0.440	1.248	13	2746	0.224	1.594	12	4831	0.276	1.504	1.363
G-MOEA/D	21	294	0.481	1.089	16	482	0.450	1.220	13	2752	0.232	1.539	12	4840	0.275	1.509	1.339
NSGA II	420	297	0.233	2.249	680	479	0.302	1.818	1000	2730	0.178	2.006	1030	4853	0.241	1.722	1.948
MOEA/D	390	296	0.197	2.660	670	491	0.287	1.913	900	2721	0.170	2.100	920	4893	0.215	1.930	2.151
MPGGA	100	291	**0.524**	–	100	473	**0.549**	–	100	2670	**0.357**	–	100	4818	**0.415**	–	–

level, if p-value ≤ 0.05 indicates that two algorithms have obvious differences in a function, otherwise the difference is not obvious. The alternative hypothesis is that values in one sample are more likely to be larger than the values in the other sample. For the results of Fig. 12, we calculate the HVs of baseline algorithms at the corresponding time points for each generation of MPGGA-ESPW. If baseline algorithms do not have results at the corresponding time point, we use linear interpolation to obtain the corresponding values. Then, the results of the Wilcoxon rank-sum test for the HVs are listed in Table 6, where if statistics> 0 indicates the HVs of the proposed MPGGA are less than the corresponding baseline algorithm.

From Table 6, the HVs of MPGGA are significantly less than those of the baseline algorithms during the time within 100 generations of MPGGA. However, as the time range tested by Wilcoxon rank-sum test shortens, the advantages and the significance of MPGGA will decrease. When only testing the time within the 10 generations of MPGGA, the differences between MPGGA and G-NSGA II (or G-MOEA/D) are not significant where the p-value remains larger than 0.05. This phenomenon indicates that manifesting the advantages of MPGGA relies on sufficient evolutionary generations (corresponding to sufficient computing time), and MPGGA is not significantly better than baseline algorithms when the computing time is small.

To further observe the solution of each algorithm, we plot the Pareto scatters in CPU-RAM projection of the final time for scenarios $(n, m) = (500, 100)$ and $(n, m) = (1000, 300)$ in Fig. 13. From Fig. 13, the Pareto solution set in CPU-RAM projections of our proposed algorithm also obviously dominates the solution set of other comparison algorithms. We use C indicator [90] to quantify the advantages and disadvantages of the dominating relationship between 3D Pareto solution sets of different algorithms. Table 7 shows the C indicator between the algorithms for the Pareto solution set of Fig. 12, where the C indicator is computed by the following formula [90].

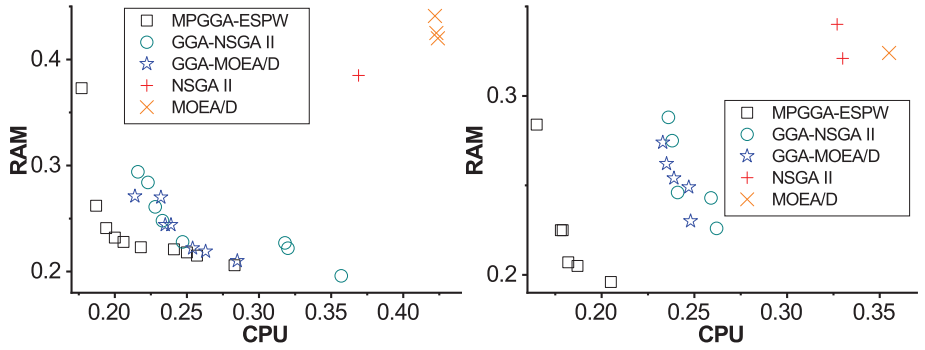
$$C(X, Y) = \frac{|y \in Y | \exists x \in X : x \preceq y|}{Y} \tag{11}$$

From Table 7, MPGGA has a higher proportion of Pareto solutions that are not dominated by the solutions of other algorithms, followed by GHW-MOEA/D, GHW-NSGA II, NSGA II and MOEA/D. From the perspective of Pareto solution dominance, the results of Fig. 13 and Table 7 again verify that the proposed MPGGA is superior to the compared algorithms.

From the experiments of Fig. 12, MPGGA can maintain advantages over the baseline evolutionary algorithms that have one population in large-scale scenarios for the problem of $\omega^{(1)}$. Next, we compare MPGGA

Table 6
Wilcoxon rank-sum test for the HVs-over-time of Fig. 12.

Algorithm	(500, 100)		(1000, 300)		(5000, 1500)		(6000, 2500)	
	p-value	statistics	p-value	statistics	p-value	statistics	p-value	statistics
The time within 100 generations of MPGGA								
MPGGA	–	–	–	–	–	–	–	–
G-NSGA II	7.7E−13	+7.167	5.7E−20	+9.150	2.9E−22	+9.705	5.5E−21	+9.400
G-MOEA/D	2.8E−11	+6.658	8.5E−20	+9.107	7.1E−22	+9.612	1.0E−20	+9.268
NSGA II	8.6E−32	+11.73	6.7E−29	+11.16	2.7E−26	+10.61	5.7E−28	+10.96
MOEA/D	4.6E−31	+11.59	9.0E−29	+11.13	4.3E−25	+10.35	2.7E−24	+10.17
$W/T/L$	4/0/0		4/0/0		4/0/0		4/0/0	
The time within 50 generations of MPGGA								
MPGGA	–	–	–	–	–	–	–	–
G-NSGA II	6.4E−7	+4.977	4.4E−10	+6.239	1.1E−10	+6.460	2.6E−11	6.666
G-MOEA/D	3.3E−7	+5.108	9.3E−10	+6.122	7.6E−11	+6.508	5.0E−10	+6.218
NSGA II	2.0E−15	+7.942	2.0E−13	+7.349	1.6E−11	+6.742	8.7E−13	+7.149
MOEA/D	3.1E−15	+7.887	1.5E−13	+7.390	1.2E−10	+6.446	3.4E−10	+6.280
$W/T/L$	4/0/0		4/0/0		4/0/0		4/0/0	
The time within 20 generations of MPGGA								
MPGGA	–	–	–	–	–	–	–	–
G-NSGA II	4.1E−3	+2.867	8.8E−4	+3.327	7.2E−4	+3.381	1.1E−4	+3.868
G-MOEA/D	4.1E−3	+2.867	1.4E−3	+3.192	8.8E−4	+3.327	1.4E−3	+3.192
NSGA II	1.2E−5	+4.382	1.1E−4	+3.868	1.7E−3	+3.138	2.9E−4	+3.625
MOEA/D	2.9E−6	+4.680	1.3E−5	+4.355	8.0E−4	+3.354	2.9E−3	+2.976
$W/T/L$	4/0/0		4/0/0		4/0/0		4/0/0	
The time within 10 generations of MPGGA								
MPGGA	–	–	–	–	–	–	–	–
G-NSGA II	**0.131**	+1.512	**0.150**	+1.436	**0.096**	+1.663	0.028	+2.192
G-MOEA/D	**0.151**	+1.436	**0.150**	+1.436	**0.131**	+1.512	**0.131**	+1.512
NSGA II	0.016	+2.419	**0.082**	+1.739	**0.257**	+1.134	**0.096**	+1.663
MOEA/D	2.5E−3	+3.024	0.005	+2.797	0.023	+2.268	**0.096**	+1.663
$W/T/L$	2/2/0		1/3/0		1/3/0		1/3/0	



(a) $(n, m) = (500, 100)$ of Fig 1(a) (b) $(n, m) = (1000, 300)$ of Fig 1(b)

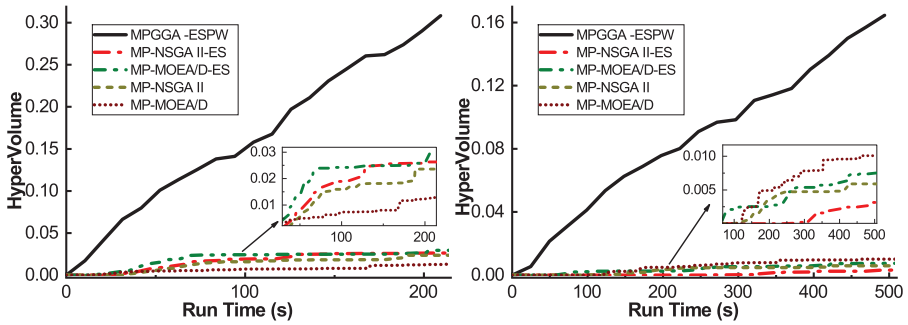
Fig. 13. The 2D Pareto solutions (CPU-RAM projections) of utilization-ESPW compared MPGGA with baselines including GHW-NSGA II, GHW-MOEA/D, NSGA II, MOEA/D at the final time of Figs. B.1(a) and B.1(b).

with other baseline multi-population algorithms including MP-NSGA II, MP-MOEA/D, MP-NSGA II-ES (using elite sharing) and MP-MOEA/D-ES. In order to enrich the diversity of the experimental scenario, we chose to perform the optimization algorithm on the AzureTraceforPacking2020 [91] driven dataset and in other GPU clusters (RTX 3060 Ti). Same as the experiments of GGA [22], we screen out some types of VMs with a minimum resource utilization of CPU, RAM and SSD greater than 0.3 on all types of machines, and finally retain 338 types of VMs, which means $\min_{j=1}^{m} u_{ijk} \leq 30\%$ for $\forall i, k$. Then, we randomly select the given numbers of VMs and machines from the 338 types of VMs and 35 types of machines. For two scenarios $(n, m) = (500, 100)$ and $(n, m) = (800, 300)$, we can plot the absolute HVs-over-time in Fig. 14 where the reference point of HV is $(1, 1, 1)$.

Fig. 14 shows that MPGGA has far higher HVs than other multi-population algorithms including MP-NSGA II, MP-MOEA/D, MP-NSGA II-ES, MP-MOEA/D-ES. The experiments in the public trace-based dataset also verify the feasibility of MPGGA in realistic scenarios. Due to the fact that a type of VM in AzureTrace could only be suitable for partial categories of machines, the feasible solution space is smaller than that in ideal scenarios. Therefore, the solutions corresponding to the genes of all individuals in a certain generation may be outside the feasible solutions space without using a constraint to guarantee the solutions in the feasible solution space. In this situation (e.g., baseline algorithms in Fig. 14), HVs will be close to 0 when using $(1, 1, 1)$ as the reference point to calculate the HV of the Pareto solution set. The results in Fig. 14 also illustrate that the solving ability of the baseline multi-population evolutionary algorithms (if without additional strategies) will significantly deteriorate when the feasible solution space range is reduced. However, the multi-population growth genetic algorithm (MPGGA), introducing additional growth routes, allows evolutionary algorithms to quickly find some of the feasible solution regions and continuously optimize solutions in these regions.

Table 7
The C indicator between the algorithms for the Pareto solution set of Fig. 12.

$C(X,Y)$ / X \ Y	MPGGA	G-NSGA II	G-MOEA/D	NSGA II	MOEA/D
$(n,m)=(500,100)$ of Fig. B.1(a)					
MPGGA	-	0	0	0	0
G-NSGA II	0.980	-	0.588	0	0
G-MOEA/D	1	0.123	-	0	0
NSGA II	1	1	1	-	0
MOEA/D	1	1	1	1	-
$(n,m)=(1000,300)$ of Fig. B.1(b)					
MPGGA	-	0	0	0	0
G-NSGA II	1	-	0.241	0	0
G-MOEA/D	1	0.207	-	0	0
NSGA II	1	1	1	-	0
MOEA/D	1	1	1	1	-
$(n,m)=(5000,1500)$ of Fig. 12(c)					
MPGGA	-	0	0	0	0
G-NSGA II	1	-	0.867	0	0
G-MOEA/D	1	0	-	0	0
NSGA II	1	1	1	-	0
MOEA/D	1	1	1	0.4	-
$(n,m)=(6000,2500)$ of Fig. 12(d)					
MPGGA	-	0	0	0	0
G-NSGA II	1	-	0.579	0	0
G-MOEA/D	1	0.136	-	0	0
NSGA II	1	1	1	-	0
MOEA/D	1	1	1	1	-



(a) $(n,m)=(500,100)$ (b) $(n,m)=(800,300)$

Fig. 14. The HV-over-time of MPGGA-ESPW compared with baselines MP-NSGA II, MP-MOEA/D, MP-NSGA II-ES, MP-MOEA/D-ES in the Azure Trace driven dataset: executed on two GPUs that RTX 3060 Ti, partial growth with growth quotas $Q=80$, $N_p=100$ of each population, $\eta=4$, $N_g=20$ for MPGGA.

5.6. Ex_7: Adaptation test for MPGGA

In Ex_6, our proposed MPGGA outperforms the compared algorithms in the problem $\omega^{(1)}$. However, MPGGA needs to design specific growth routes for corresponding problems. If using MLSPT as the growth route to solve other optimization algorithms, MPGGA may fail to obtain competitive solutions.

To test the adaptability of MPGGA, we execute experiments in two problems related to energy consumption, including minimizing the maximum energy consumption of each server node (denoted as $\min\omega^{(2)}$) and minimizing the total energy consumption of the whole system (denoted as $\min\omega^{(3)}$). The Ex_7 adopts the calculation formula for energy consumption in research of GGA [22], mainly to introduce nonlinear terms. Then, the problems of $\min\omega^{(2)}$ and $\min\omega^{(3)}$ can be respectively written as Eqs. (12) and (14).

$$\min\omega^{(2)}=\min\max_{j=1}^{m}E_j \tag{12}$$

$$\min\omega^{(3)}=\min\sum_{j=1}^{m}E_j \tag{13}$$

where E_j means the energy consumption of the jth server node, and

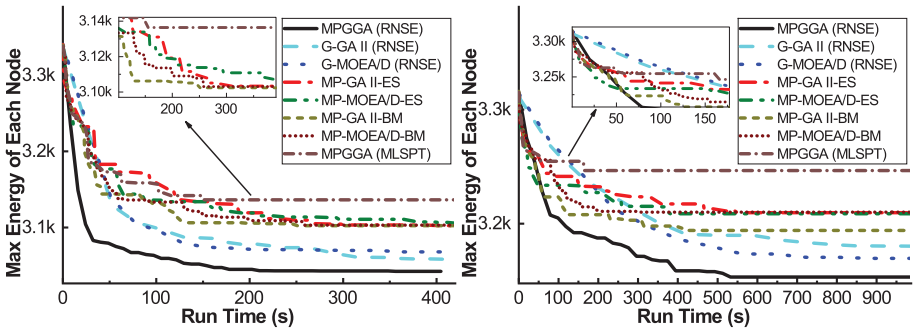
$$E_j=\sum_{k=1}^{d}\left(a_{jk}\left(\sum_{i=1}^{n}x_{ij}u_{ijk}\right)^2+b_{jk}\left(\sum_{i=1}^{n}x_{ij}u_{ijk}\right)+c_{jk}+d_{jk}\max_{i=1}^{n}(x_{ij})\right) \tag{14}$$

where a_{jk}, b_{jk}, c_{jk} and d_{jk} are the coefficients of energy consumption for quadratic polynomials. In the simulation experiments, we generate the coefficients as integers according to the uniform distributions as

$$\begin{cases} a_{jk}\sim \mathrm{U}(1,10), b_{jk}\sim \mathrm{U}(0,100), \\ c_{jk}\sim \mathrm{U}(100,200), d_{jk}\sim \mathrm{U}(500,1000). \end{cases} \tag{15}$$

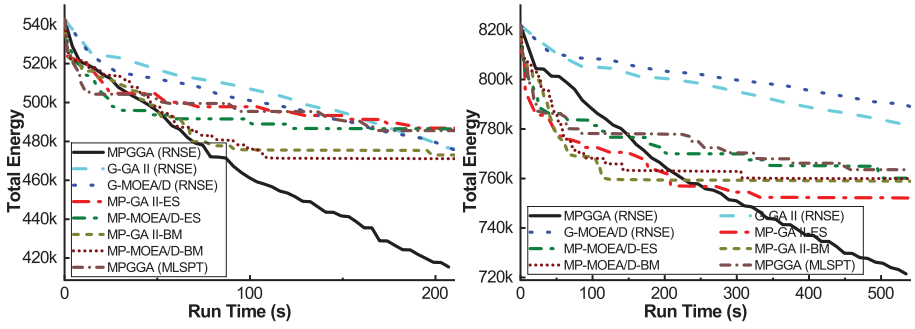
In experiments related to energy consumption, we use two MPGGAs with different growth routes as follows.

- MPGGA (MLSPT), using MLSPT algorithm as the growth route of MPGGA;
- MPGGA (RNSE), using random neighborhood search for energy consumption (RNSE) as the growth route of MPGGA. The neighborhood based on only changing of the server node of virtual machine V_i is defined as: for two solutions $\kappa=\langle\psi_1,\psi_2,\dots,\psi_m\rangle$ and $\bar{\kappa}=\langle\bar{\psi}_1,\bar{\psi}_2,\dots,\bar{\psi}_m\rangle$, if $\exists j_1\neq j_2$ satisfy $\psi_{j_1}-\bar{\psi}_{j_1}=\bar{\psi}_{j_2}-\psi_{j_2}=\{V_i\}$ and $\psi_j=\bar{\psi}_j$ for $\forall j\in\{1,2,\dots,m\}-\{j_1,j_2\}$, then κ and $\bar{\kappa}$ can be referred to as mutual neighborhoods based on only changing of the server node of V_i. RNSE randomly chooses a VM V_i, calculates $\omega^{(2)}$ (or $\omega^{(3)}$, depending on the current optimization objective) corresponding to the solutions of all neighborhoods, and chooses the neighborhood that has the minimum $\omega^{(2)}$ (or $\omega^{(3)}$) to update the current solution.



(a) $(n, m) = (800, 200)$ (b) $(n, m) = (2000, 300)$

Fig. 15. The maximum energy consumptions of each node for $\omega^{(2)}$ compared MPGGAs to baselines: executed on RTX 3060 Ti, partial growth with growth quotas $Q = 80$, $N_p = 100$ of each population, $\eta = 4$, $N_g = 50$ for MPGGAs; all algorithms are executed on two GPUs using 4 processes.



(a) $(n, m) = (800, 200)$ (b) $(n, m) = (1500, 300)$

Fig. 16. The total energy consumption of system for $\omega^{(3)}$ compared MPGGAs to baselines: executed on RTX 3060 Ti, partial growth with growth quotas $Q = 80$, $N_p = 100$ of each population, $\eta = 4$, $N_g = 50$ for MPGGAs; all algorithms are executed on two GPUs using 4 processes.

Among them, MLSPT is a growth route specifically designed for problem $\omega^{(1)}$ as Algorithm 2, and RNSE is a growth route specifically designed for problems $\omega^{(2)}$ and $\omega^{(3)}$. The baseline algorithms include G-GA II (RNSE), G-MOEA/D (RNSE), MP-GA II-ES, MP-MOEA/D-ES, MP-GA II-BM, MP-MOEA/D-BM. $\omega^{(2)}$ and $\omega^{(3)}$ are single-objective problems but still need to consider multi-dimensional resources' utilizations. The GA II in baselines represents using the value of a single objective as fitness and utilizing the elite strategy in NSGA II for population regeneration, which is a variant of NSGA II in single-objective problem; G-GA II (RNSE) and G-MOEA/D (RNSE) are two instantiations growable genetic algorithm with all growth using RNSE as the growth route, where the growable genetic algorithm is one of the latest evolutionary algorithms with excellent performance in multi-dimensional resource scheduling problems; MP-GA II-ES means the multi-population genetic algorithm using elite sharing strategy to interact information among multiple populations; BM means using balanced randomly mixing to interact information among multiple populations which is one of the important strategies in the latest multi-population algorithm [52,53,55,56]. The results of energy consumptions for $\omega^{(2)}$ and $\omega^{(3)}$ are respectively plotted in Figs. 15 and 16.

From Fig. 15 (minimizing the maximum energy consumption of each node) and Fig. 16 (minimizing the total energy consumption of the system), it can be seen that the energy consumption curve obtained by MPGGA using MLSPT as the growth route is not better than the baseline algorithms. This indicates that MLSPT is not as suitable for energy consumption problems as utilization optimization problems, which is because energy consumption is a high power polynomial in this experiment and MLSPT is mainly designed for load balancing of resource utilization. The comparison between MPGGA (MLSPT) and baseline algorithms also reveals that MPGGA with only one certain growth route will have limited adaptability for different scenarios. In addition, MPGGA (RNSE) uses a more adaptive neighborhood search to solve energy consumption-related optimization problems, and its optimization results in the 50th generation are significantly better than the baseline algorithms. This indicates that although the adaptability of a single algorithm of MPGGA (i.e., MPGGA using MLSPT as the growth route) is limited, the MPGGA architecture has appreciable flexibility, allowing various algorithms to serve as its growth routes to increase the adaptability to optimization scenarios and problems. For two scenarios of Fig. 15, MPGGA (RNSE) reduces the maximum energy consumption by 1.93% and 1.28%; for that of Fig. 16, MPGGA (RNSE) reduces the total energy consumption by 11.83% and 7.07% compared to the best baseline algorithm. Specifically, in Fig. 16(b), within 200 s, the energy consumption corresponding to the optimized solution of MPGGA (RNSE) is larger than that of the baseline algorithms. This indicates that the advantage of MPGGA-RNSE depends on sufficient computation time. This is because the computational complexity of the RNSE growth route used by MPGGA is relatively high. When the overall optimization of the current solution set of the evolutionary algorithm is not high, the cost-effectiveness of using RNSE for additional growth is lower, and it is not as effective as directly conducting the genetic evolutionary search. After the optimization of the solution set is improved, evolutionary algorithms such as genetic algorithms may fall into local convergence and find it difficult to continue improving the solution. However, from the experimental results over a long period of time, it can be seen that the optimization solution obtained by MPGGA (RNSE) remains better than the baseline algorithms when the computing time is long enough. This indicates that the MPGGA architecture can improve the optimality of local convergence solutions of evolutionary algorithms.

Table 8
Wilcoxon rank-sum test for the energy consumption of Fig. 15 and Fig. 16 within 50 generations of MPGGA (RNSE).

Algorithm	$\omega^{(2)}$				$\omega^{(3)}$			
	(800, 200)		(2000, 300)		(800, 200)		(1500, 300)	
	p-value	statistics	p-value	statistics	p-value	statistics	p-value	statistics
MPGGA (RNSE)	–	–	–	–	–	–	–	–
G-GA II	1.1E−7	+5.304	1.5E−7	+5.250	2.6E−5	+4.206	2.2E−8	+5.598
G-MOEA/D	5.9E−10	+6.194	4.4E−6	+4.427	9.7E−5	+3.899	5.6E−10	+6.201
MP-GA II-ES	1.2E−14	+7.720	4.7E−14	+7.539	1.2E−6	+4.849	**0.214**	+1.241
MP-MOEA/D-ES	9.4E−15	+7.747	6.8E−14	+7.492	2.0E−5	+4.260	1.1E−3	+3.256
MP-GA II-BM	2.3E−14	+7.633	1.7E−12	+7.057	4.9E−4	+3.484	**5.1E−2**	+1.951
MP-MOEA/D-BM	2.3E−14	+7.633	5.2E−14	+7.526	1.6E−3	+3.149	1.6E−2	+2.406
MPGGA (MLSPT)	1.9E−15	+7.948	5.9E−15	+7.807	3.6E−6	+4.635	1.1E−3	+3.390
$W/T/L$	7/0/0		7/0/0		7/0/0		5/2/0	

Table 9
Wilcoxon rank-sum test and Friedman test for $\omega^{(3)}$ compared MPGGAs to baseline algorithms where each combination of (m, n) has 30 instance.

Algorithm	(300, 50)				(500, 100)				(700, 250)			
	ϵ_1	ϵ_2	Score	Rank	ϵ_1	ϵ_2	Score	Rank	ϵ_1	ϵ_2	Score	Rank
MPGGA (RNSE)	–	0/0/30	1	**1**	–	0/2/28	1.00	**1**	–	0/2/28	1.00	**1**
G-GA II	30/0/0	3/12/15	3.23	3	30/0/0	2/5/23	3.90	3	30/0/0	27/3/0	6.83	7
G-MOEA/D	30/0/0	2/14/14	3.03	2	30/0/0	1/4/25	3.47	2	30/0/0	29/1/0	7.07	8
MP-GA II-ES	30/0/0	14/5/11	5.93	6	30/0/0	13/6/11	5.57	6	29/1/0	14/3/13	5.00	4
MP-MOEA/D-ES	30/0/0	12/9/9	6.37	8	30/0/0	12/5/13	6.23	7	30/0/0	11/4/15	5.30	6
MP-GA II-BM	30/0/0	13/3/14	5.67	5	30/0/0	6/5/19	4.27	4	24/6/0	3/3/24	2.63	2
MP-MOEA/D-BM	30/0/0	8/4/18	4.70	4	29/0/0	12/2/17	5.17	5	29/1/0	4/0/26	3.10	3
MPGGA (MLSPT)	30/0/0	–	6.13	7	30/0/0	–	6.40	8	28/2/0	–	5.07	5

To evaluate the significance of the performance differences between baseline algorithms and MPGGA (RNSE), we use Wilcoxon rank-sum test to show the differences in the energy consumption-related objective values obtained by each algorithm for Figs. 15 and 16. The results are shown in Table 8. Table 8 shows that in the experiments of Figs. 15 and 16, the results obtained by MPGGA (RNSE) are significantly better than the baseline algorithms.

To further evaluate the performance of MPGGA in statistics, we respectively conduct 30 instances on each combination of $(m, n) = (300, 50)$, $(m, n) = (500, 100)$ and $(m, n) = (700, 250)$ for problem $\omega^{(3)}$, and recorded the results of Wilcoxon rank-sum test and Friedman test. The results are shown in Table 9, where ϵ_1 means $W/T/L$ using MPGGA (RNSE) as the standard of Wilcoxon rank-sum test, ϵ_2 means $W/T/L$ using MPGGA (MLSPT) as the standard, score and rank corresponds to Friedman test. In the experiments, we set the number of populations as $\eta = 4$, each population has $N_p = 100$ individuals, and test the results within the time of $N_g = 50$ generations of MPGGA (RNSE). $\epsilon_1 = 30/0/0$ for the row of G-GA II represents there are 30 instances where the results of MPGGA (RNSE) are significantly better than G-GA II; $\epsilon_2 = 3/12/15$ represents there are 3 instances where the results of MPGGA (MLSPT) are significantly better than G-GA II, 12 instances for MPGGA (MLSPT) approaching G-GA II, and 15 instances for MPGGA (MLSPT) worse than G-GA II.

From the statistical test results of multiple instances in Table 9, it can be seen that MPGGA (RNSE) can maintain better than other algorithms in various scenarios, while the performance of MPGGA (MLSPT) fluctuates greatly. The results shows that the adaptability of a specific algorithm (i.e., MPGGA (MLSPT)) in the MPGGA series is limited by the growth route. However, the performance of MPGGA (RNSE) demonstrates the flexibility of MPGGA framework, allowing various algorithms as its growth route to adapt to various scenarios. The MPGGA framework also has the potential for solving other problems, and its experimental results for the multi-objective asymmetric traveling salesman problem can be seen in Appendix B.

5.7. Summary of experiments

Through the multiple groups of experiments, we have not only completed the performance test of various strategies of MPGGA but also verified that MPGGA outperforms the compared algorithm in the problem of multi-dimensional resource utilization optimization. Among these experiments:

- Ex_1 validates that only increasing the number of populations without additional information interactions between populations has no obvious impact on the convergence of GGA. Ex_1 also validates the proposed system architecture with multi GPUs can accelerate the computation of MPGGA.
- Ex_2 evaluates the performance of different information interaction strategies between populations and demonstrates that: the addition of information interaction can improve the performance of MPGGA and elite sharing strategy (Ir_3) has the most significant improvement, followed by balanced mixing (Ir_1) and elite supporting (Ir_2).
- Ex_3 evaluates the performance of different growth quota allocation strategies in MPGGA and demonstrates that: preferably giving more additional growth quotas to the populations with lower ranking (priority worse, Qa_2) can most significantly improve the overall convergence speed.
- Ex_4 evaluates the performance of different combinations of growth quota reallocation strategies and growth quota allocation strategies in MPGGA and demonstrates that: the combination of elite sharing and priority worse ($Ir_3 + Qa_2$) outperforms other combinations.
- Ex_5 tests the performance of MPGGA under different growth quotas, and shows that: with the increase of growth quota, the performance of MPGGA shows a trend of first increasing and then decreasing in each time profile.
- For the problem of $\omega^{(1)}$ (minimizing the maximum utilization of the resources in each dimension), Ex_6 compares the proposed MPGGA-ESPW with the some baseline algorithms including some single-population evolutionary algorithms (GHW-NSGA II, GHW-MOEA/D,

NSGA II, MOEA/D) and some multi-population evolutionary algorithms (MP-NSGA II, MP-MOEA/D, MP-NSGA II-ES, MP-MOEA/D-ES). The results demonstrate our proposed MPGGA-ESPW has a faster convergence rate and better optimality than compared algorithms.

- For other multi-dimensional resources utilization optimization problems (energy consumption-related problems $\omega^{(2)}$ and $\omega^{(3)}$), Ex_7 compares the proposed MPGGA with baseline algorithms to evaluate the adaptability of MPGGA to different scenarios. The results show that MPGGA with MLSPT growth route designed for problem $\omega^{(1)}$ is unable to adapt to problem $\omega^{(2)}$ and problem $\omega^{(3)}$, which also shows that the adaptability of a specific algorithm in the MPGGA series is limited by the algorithm corresponding to its growth route. Additionally, using the neighborhood search (a more adaptive algorithm) as its growth route, MPGGA (RNSE) still has better performances than the baseline algorithms in terms of convergence speed and optimality, which further demonstrates the flexibility of the MPGGA framework. In realistic optimization problems, we can select a suitable algorithm as the growth route to improve the adaptability of MPGGA to specific scenarios or select some universal algorithms (e.g., neighbor search) to increase the universality of MPGGA for different scenarios.

6. Conclusion

Optimizing multi-dimensional resource utilization in cloud computing presents a formidable challenge, as it necessitates the concurrent consideration of multiple resource bottlenecks. By incorporating multi-populations with information interaction strategies and growth quota reallocation strategies into growable genetic algorithms, this paper proposed a new algorithm called the multi-population growable genetic algorithm (MPGGA). To streamline and expedite MPGGA, we also presented a system architecture comprising two types of executors: population internal evolution executors and a management center of evolutions. The former is primarily responsible for the internal evolution process of each GGA population, while the latter oversees information interaction and growth quota reallocation.

To evaluate MPGGA's performance, we conducted multiple sets of experiments. Through these experiments, we first identified the optimal combination of information interaction strategy and growth quota allocation strategy, namely ESPW (elite sharing and priority worse, $Ir_3 + Qa_2$). Subsequently, we demonstrated that partial growth in MPGGA outperforms both non-growth and all growth. Then, we confirmed that MPGGA-ESPW exhibits superior convergence speed and optimality compared to baseline algorithms. Lastly, we test the adaptability of MPGGA to different scenarios of multi-dimensional utilization optimizations. The results show that the adaptability of a specific algorithm in the MPGGA series is limited by the algorithm corresponding to its growth route and also demonstrate the MPGGA framework is flexible enough to allow various algorithms as its growth route to adapt to various scenarios.

MPGGA represents the advancement and extension of the growable genetic algorithm family. It showcases the potential of employing multiple populations, information interaction strategies, and partial growth quota reallocation strategies to further enhance GGA's convergence speed. This research also underscores the value of continued development within the GGA algorithm family. Potential future work includes applying MPGGA to other complex problems or hierarchical distributed computing systems (e.g., edge-cloud, fog-cloud), as well as exploring the combination of multi-population strategies and growth genetic algorithms to enable distinct growth trajectories and information interaction rules for different populations. In the realistic optimization problems of diverse cloud systems or other fields, we can select a suitable algorithm as the growth route to improve the adaptability of MPGGA to specific scenarios, alternatively select some universal algorithms to increase the universality of MPGGA for varying scenarios. Auxiliary strategies can be adjusted according to actual requirements. It may also be effective to combine the extra growth route and dynamic allocation of growth quotas with other evolutionary algorithms. Additionally, a theoretical analysis of the feasible solution clusters belonging to the same local convergent solution in growable genetic algorithms is also a possible direction.

CRediT authorship contribution statement

Guangyao Zhou: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Yuanlun Xie:** Validation, Software, Investigation, Conceptualization. **Haocheng Lan:** Validation, Software, Investigation, Conceptualization. **WenHong Tian:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization. **Rajkumar Buyya:** Writing – review & editing, Conceptualization. **Kui Wu:** Writing – review & editing, Conceptualization.

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 research is partially supported by National Key Research and Development Program of China with Grant ID 2018AAA0103203, Project of Key Research and Development Program of Sichuan Province, China with Grant ID 2021YFG0325, the Chengdu Science and Technology Project with Grant ID 2022-YF05-02014-SN, and National Natural Science Foundation of China with Grant ID 61672136 and 61828202.

Appendix A. GPU based acceleration and architecture for MPGGA

In this appendix, we present some acceleration methods for MPGGA program based on array operation and distributed system. The array operations on GPUs can be leveraged to accelerate the procedure of 'Call the MLSPT algorithm in the following loop' (i.e., line 5–20) in Algorithm 2 to eliminate 'for loop' (line 5) and 'while loop' (line 8). For $\psi = \psi_{j_1} \cup \psi_{j_2}$ of two nodes N_{j_1} and N_{j_2}, we can assume the set sorted in ascending order according to the value of $w \cdot \left(Y_{ij_1} - Y_{ij_2}\right) = Z_{ij_1} - Z_{ij_2}$ in ψ as $\left\{V_{\alpha_1}, V_{\alpha_2}, \ldots, V_{\alpha_h}\right\}$, where we set $Z_{ij} = w \cdot Y_{ij}$ for $\forall i, j$. Then, the operation of MLSPT algorithm equals to finding an index φ in $\{0, 1, 2, \ldots, h\}$ **s.t.** $\sum_{i=0}^{\varphi} Z_{\alpha_i j_1}$ and $\sum_{i=\varphi+1}^{h+1} Z_{\alpha_i j_2}$ as close as possible, assuming $Z_{\alpha_0 j} = Z_{\alpha_{h+1} j} = 0$ for $j \in \{j_1, j_2\}$. And $\sum_{i=0}^{\varphi} Z_{\alpha_i j_1} \approx \sum_{i=\varphi+1}^{h+1} Z_{\alpha_i j_2}$ is equivalent to $\sum_{i=0}^{\varphi} Z_{\alpha_i j_1} + \sum_{i=0}^{\varphi} Z_{\alpha_i j_2} \approx \sum_{i=\varphi+1}^{h+1} Z_{\alpha_i j_2} + \sum_{i=0}^{\varphi} Z_{\alpha_i j_2} \to \sum_{i=0}^{\varphi} \left(Z_{\alpha_i j_1} + Z_{\alpha_i j_2}\right) \approx \sum_{i=0}^{h+1} Z_{\alpha_i j_2} = Sum_{j_2}$. Setting $S_{j_1} = \left\langle Z_{\alpha_0 j_1}, Z_{\alpha_1 j_1}, \ldots, Z_{\alpha_h j_1} \right\rangle$ and $S_{j_2} = \left\langle Z_{\alpha_0 j_2}, Z_{\alpha_1 j_2}, \ldots, Z_{\alpha_h j_2} \right\rangle$, 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 φ, and update $\psi'_{j_1} = \left\{V_{\alpha_1}, V_{\alpha_2}, \ldots, V_{\alpha_\varphi}\right\}$ and $\psi'_{j_2} = \psi - \psi'_{j_1}$ where if $\varphi = 0$ then $\psi'_{j_1} = \emptyset$. Using this idea, we can quickly adjust the tasks or VMs of two nodes. On this basis, we make further improvement and we propose the array operation-based growth route as Algorithm A.1, which can quickly adjust the tasks or VMs of multiple nodes simultaneously in MLSPT growth route. The introduction of array operations allows growth route to be accelerated by running on GPUs.

Algorithm A.1: Growth route accelerated by array operation (improved algorithm for Algorithm 2)

Input : Y_{ij} for $\forall i, j$, $w = \langle w_1, w_2, \dots, w_d \rangle$, solution $X = \{x_{ij}\}_{n\times m} = \langle X_1, \dots, X_m \rangle$, G_{step}
Output: Solution $X = \{x_{ij}\}_{n\times m}$ corresponding to $I = \langle \lambda_1, \lambda_2, \dots, \lambda_n \rangle$

$Z = \{Z_{ij}\}_{n\times m} = \langle Z_1, \dots, Z_m \rangle$
Set $i = 0$, $Exists\_Ner = True$, $Mark = \max(sum(X * Z, dim = 0), dim = 0)$
while Exists_Ner and $i < G_{step}$ **do**
  $Exists\_Ner = False$, $i++$
  $j_1 = \arg\max(sum(X * Z, dim = 0), dim = 0)$ where $X * Z$ means Hadamard product between X and Z ; // Calculate the node with the largest weighted utilization
  $X^{new} = X + \bar{X}_{j_1}$ where $\bar{X}_{j_1} = \left\langle X_{j_1}, \dots, X_{j_1} \right\rangle_m$, $F_1 = X^{new} * \bar{Z}_{j_1}$, $F_2 = X^{new} * Z$
  $D = \text{argsort}(F_1 - F_2, dim = 0)$ and $Rm = range(m)$, $F_1 = F_1[D[:, Rm], Rm]$, $F_2 = F_2[D[:, Rm], Rm]$; // Rearrange each column of F_1 and F_2 according to the sorting order of corresponding column in $F_1 - F_2$
  Expand F_1 as $EF_1[0] = \langle 0, \dots, 0 \rangle_m$, $EF_1[1 : n+1] = F_1[1 :]$, similar to EF_2
  $CS_1 = Tril_{(n+1)\times(n+1)} \times EF_1$ and $CS_2 = Triu_{(n+1)\times(n+1)} \times EF_2$ where $Tril$ is lower triangular matrix and $Triu$ is upper triangular matrix, $\times$ means matrix product ; // Quickly calculate cumulative sums of EF_1 and EF_2 by triangular matrix
  $EF_2 = CS_2 - EF_2$; // adjust EF_2 by cumulative sums
  $CS_3 = \text{abs}(EF_2 - EF_1)$, $D_2 = \arg\min(CS_3, dim = 0)$; // Obtain the index φ for each column
  $F_1 = CS_1[D_2[Rm], Rm]$, $F_2 = CS_2[D_2[Rm], Rm]$, $M^{new} = \min(maximum(F_1, F_2), dim = 0)$, $j_2 = \arg\min(maximum(F_1, F_2), dim = 0)$, $\varphi = D_2[j_2]$; // Obtain the optimal adjustment result combining N_{j_1} with all other nodes
  if $M^{new} < Mark$ **then**
    $Exists\_Ner = True$, $Mark = M^{new}$
    $X'_{j_1} = X'_{j_2} = \langle 0, \dots, 0 \rangle_n^{\mathrm{T}}$ $X'_{j_1} = X^{new}_{j_2}[D[j_2, : \varphi]] = 1$, $X'_{j_2} = X^{new}_{j_2}[D[j_2, \varphi :]] = 1$, ; // Reallocate tasks or VMs between N_{j_1} and N_{j_2} according to φ
    $X_{j_1} = X'_{j_1}$ and $X_{j_2} = X'_{j_2}$; // Update the solution X

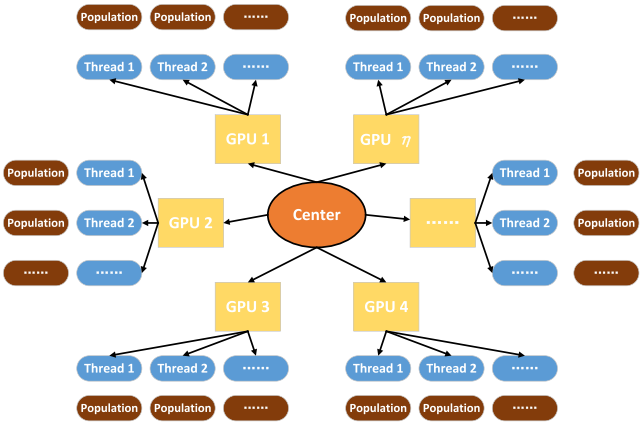


Fig. A.1. The distributed system architecture with multi-GPUs to execute the MPGGA.

Due to the growth route being able to run on GPUs as Algorithm A.1, we use the method combining multi-thread and multi-GPU, i.e., each GPU corresponds to multiple threads (or multiple processes) and each thread (or process) corresponds to one population. As each population has only partial individuals to grow in each generation, so each thread (also each population) must have idle time without using GPU. Therefore, the method combining multi-thread and multi-GPU can improve the utilization of GPU.

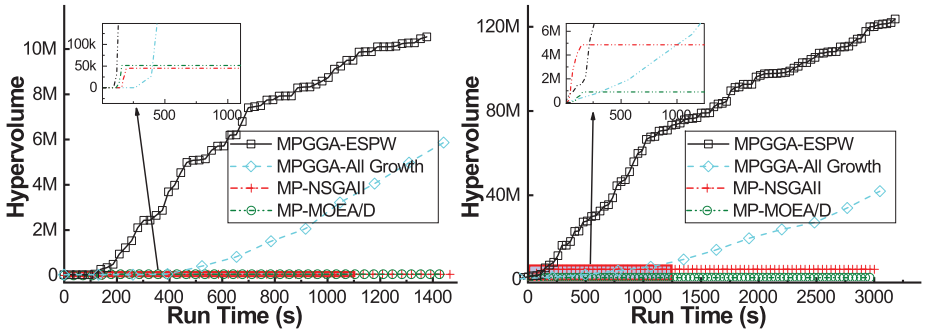
The management center of evolutions mainly performs information interaction and growth quota reallocation. In this paper, the management center of evolutions is deployed on the central server. It receives information of elite individuals from each population through the network and delivers the selected information to each population according to specific rules and strategies. Since the management center of evolutions receives the elite information of each generation from all populations, it is also responsible for calculating and recording the Pareto solution set and outputting the final Pareto solution set at the end of the MPGGA. With the above description, the system architecture to execute the MPGGA can be seen in Fig. A.1.

Appendix B. Performance evaluation of MPGGA in MoATSP

In this appendix, we present some experimental results of MPGGA in solving multi-objective asymmetric traveling salesman problem (MoATSP). It targets at supplementing that our proposed evolutionary algorithm MPGGA is also applicable to other multi-objective optimization problems.

MoATSP is a complex problem in TSP. There are multiple dimensional distances (or costs) between cities in MoATSP. And the round-trip distances between two cities may not be equal, i.e., asymmetric distances. When visiting each city once and returning to the starting city, the optimization objectives of MoATSP are respectively minimizing total distances of each dimension.

In experiments, we use simulation program to randomly generate all distances of MoATSP by a uniform distribution U[20, 100]. We use the order in which cities are visited to represent each gene. The crossover method between two individuals is partial mapping crossover, i.e., randomly selecting one segment for exchange and using missing genes to randomly replace repetitive genes that are not involved in exchange. The mutation method of one individual is to randomly select a portion of the genes for sequence exchange. The growth route is based on local search, whose neighborhood solutions are that there are only two different genes. Other parameters of MPGGA include $G_{step} = 10$, $\eta = 8$, $N_p = 100$ for each population. Then, we carry out experiments comparing MPGGA with multi-population NSGA-II, multi-population MOEA/D, and MPGGA with all growth. The results of two experiments respectively with 200 cities and 500 cities are plotted in Fig. B.1. From the experimental results in Fig. B.1, it is intuitive that our proposed MPGGA still has better convergence speed and optimization performance in MoATSP. This demonstrates the applicability of our proposed MPGGA for other types of multi-objective optimization problems.



(a) 200 cities, $N_g = 60$ for MPGGA-ESPW (b) 500 cities, $N_g = 90$ for MPGGA-ESPW

Fig. B.1. The HV-over-time of MPGGA-ESPW in solving MoATSP compared with baselines multi-population NSGA-II, multi-population MOEA/D, and MPGGA with all growth: 2-dimensional distances (costs), partial growth with different total growth quotas $Q = 80$ for MPGGA-ESPW; $N_p = 100$ of each population and $\eta = 8$ for each algorithm; 5 skip points for the symbols in the curves of multi-population NSGA-II and multi-population MOEA/D in figures.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.

[2] 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.

[3] 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.

[4] S. Midya, A. Roy, K. Majumder, S. Phadikar, Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach, J. Netw. Comput. Appl. 103 (2018) 58–84.

[5] 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.

[6] W. Wei, H. Gu, K. Wang, J. Li, X. Zhang, N. Wang, Multi-dimensional resource allocation in distributed data centers using deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. 20 (2) (2023) 1817–1829.

[7] M.I. Khaleel, A fault tolerance aware green IoT workflow scheduling algorithm for multi-dimensional resource utilization in sustainable cloud computing, Internet Things 23 (2023) 100909.

[8] Y. Sun, S. Chen, Z. Wang, S. Mao, A joint learning and game-theoretic approach to multi-dimensional resource management in fog radio access networks, IEEE Trans. Veh. Technol. 72 (2) (2023) 2550–2563.

[9] J. Huang, K.B. Kent, J. Yen, Y. Wang, Hestia: A cost-effective multi-dimensional resource utilization for microservices execution in the cloud, in: K. Ye, L. Zhang (Eds.), Cloud Computing - CLOUD 2022 - 15th International Conference, Held As Part of the Services Conference Federation, SCF 2022, Honolulu, HI, USA, December 10-14, 2022, Proceedings, in: Lecture Notes in Computer Science, vol. 13731, Springer, 2022, pp. 22–38.

[10] 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.

[11] T. Dong, F. Xue, C. Xiao, J. Li, Task scheduling based on deep reinforcement learning in a cloud manufacturing environment, Concurr. Comput. Pract. Exp. 32 (11) (2020).

[12] 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.

[13] 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.

[14] H. Qiu, X. Xia, Y. Li, X. Deng, A dynamic multipopulation genetic algorithm for multiobjective workflow scheduling based on the longest common sequence, Swarm Evol. Comput. 78 (2023) 101291.

[15] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[16] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evol. Comput. 2 (3) (1994) 221–248.

[17] A.J. Miriam, R. Saminathan, S. Chakaravarthi, Non-dominated sorting genetic algorithm (NSGA-III) for effective resource allocation in cloud, Evol. Intell. 14 (2) (2021) 759–765.

[18] J.K. Mandal, S. Mukhopadhyay, P. Dutta (Eds.), Multi-Objective Optimization - Evolutionary to Hybrid Framework, Springer, 2018.

[19] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.

[20] H. Xu, W. Zeng, D. Zhang, X. Zeng, MOEA/HD: A multiobjective evolutionary algorithm based on hierarchical decomposition, IEEE Trans. Cybern. 49 (2) (2019) 517–526.

[21] C. Guerrero, I. Lera, C. Juiz, Genetic-based optimization in fog computing: Current trends and research opportunities, Swarm Evol. Comput. 72 (2022) 101094.

[22] G. Zhou, W. Tian, R. Buyya, K. Wu, Growable genetic algorithm with heuristic-based local search for multi-dimensional resources scheduling of cloud computing, Appl. Soft Comput. (ISSN: 1568-4946) 136 (2023) 110027.

[23] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, H. Zhou, Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey, Swarm Evol. Comput. 44 (2019) 365–387.

[24] K. Shen, T.D. Pessemier, L. Martens, W. Joseph, A parallel genetic algorithm for multi-objective flexible flowshop scheduling in pasta manufacturing, Comput. Ind. Eng. 161 (2021) 107659.

[25] Z. Xiao, X. Liu, J. Xu, Q. Sun, L. Gan, Highly scalable parallel genetic algorithm on Sunway many-core processors, Future Gener. Comput. Syst. 114 (2021) 679–691.

[26] J. Luo, D.E. Baz, R. Xue, J. Hu, Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm, Future Gener. Comput. Syst. 108 (2020) 119–134.

[27] 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.

[28] I. Aoudia, S. Benharzallah, L. Kahloul, O. Kazar, A multi-population genetic algorithm for adaptive QoS-aware service composition in fog-IoT healthcare environment, Int. Arab J. Inf. Technol. 18 (3A) (2021) 464–475.

[29] G. Zhou, W. Tian, R. Buyya, Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions, 2021, CoRR abs/2105.04086.

[30] S.G. Domanal, R.M.R. Guddeti, R. Buyya, A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment, IEEE Trans. Serv. Comput. 13 (1) (2020) 3–15.

[31] 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.

[32] T. Chen, A.G. Marqués, G.B. Giannakis, DGLB: Distributed stochastic geographical load balancing over cloud networks, IEEE Trans. Parallel Distrib. Syst. 28 (7) (2017) 1866–1880.

[33] 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.

[34] G. Zhou, W. Tian, R. Buyya, Multi-search-routes-based methods for minimizing makespan of homogeneous and heterogeneous resources in Cloud computing, Future Gener. Comput. Syst. 141 (2023) 414–432.

[35] 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.
[36] 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.
[37] S. Kardani-Moghaddam, R. Buyya, K. Ramamohanarao, ADRL: A hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds, IEEE Trans. Parallel Distrib. Syst. 32 (3) (2021) 514–526.
[38] L. Abualigah, A. Diabat, A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments, Cluster Comput. (2020) 1–19.
[39] A. Belgacem, K.B. Bey, H. Nacer, S. Bouznad, Efficient dynamic resource allocation method for cloud computing environment, Clust. Comput. 23 (4) (2020) 2871–2889.
[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] 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.
[42] F. Ramezani, J. Lu, F.K. Hussain, Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization, in: Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings, in: Lecture Notes in Computer Science, vol. 8274, Springer, 2013, pp. 237–251.
[43] R. Jena, Multi objective task scheduling in cloud environment using nested PSO framework, Procedia Comput. Sci. 57 (2015) 1219–1227.
[44] J. Li, Y. Han, A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system, Clust. Comput. 23 (4) (2020) 2483–2499.
[45] 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.
[46] 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.
[47] D.A. Monge, E. Pacini, C. Mateos, E. Alba, C.G. Garino, CMI: An online multi-objective genetic autoscaler for scientific and engineering workflows in cloud infrastructures with unreliable virtual machines, J. Netw. Comput. Appl. 149 (2020).
[48] S. Mehta, P. Kaur, P. Agarwal, Improved whale optimization variants for SLA-compliant placement of virtual machines in cloud data centers, Multimedia Tools Appl. 83 (1) (2024) 149–171.
[49] 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).
[50] H. Aziza, S. Krichen, A hybrid genetic algorithm for scientific workflow scheduling in cloud environment, Neural Comput. Appl. 32 (18) (2020) 15263–15278.
[51] S. Kayalvili, M. Selvam, Hybrid SFLA-GA algorithm for an optimal resource allocation in cloud, Clust. Comput. 22 (Supplement) (2019) 3165–3173.
[52] S. Nama, A.K. Saha, A bio-inspired multi-population-based adaptive backtracking search algorithm, Cogn. Comput. 14 (2) (2022) 900–925.
[53] L. Fu, H. Ouyang, C. Zhang, S. Li, A.W. Mohamed, A constrained cooperative adaptive multi-population differential evolutionary algorithm for economic load dispatch problems, Appl. Soft Comput. 121 (2022) 108719.
[54] C.B. Djaballah, W. Nouibat, A new multi-population artificial bee algorithm based on global and local optima for numerical optimization, Clust. Comput. 25 (3) (2022) 2037–2059.
[55] Y. Zuo, Z. Fan, T. Zou, P. Wang, A novel multi-population artificial bee colony algorithm for energy-efficient hybrid flow shop scheduling problem, Symmetry 13 (12) (2021) 2421.
[56] Y. Sun, Y. Chen, Multi-population improved whale optimization algorithm for high dimensional optimization, Appl. Soft Comput. 112 (2021) 107854.
[57] X. Zhang, Z. Zhan, W. Fang, P. Qian, J. Zhang, Multipopulation ant colony system with knowledge-based local searches for multiobjective supply chain configuration, IEEE Trans. Evol. Comput. 26 (3) (2022) 512–526.
[58] Y. Tian, R. Liu, X. Zhang, H. Ma, K.C. Tan, Y. Jin, A multipopulation evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems, IEEE Trans. Evol. Comput. 25 (3) (2021) 405–418.
[59] K. Yang, J. Zheng, J. Zou, F. Yu, S. Yang, A dual-population evolutionary algorithm based on adaptive constraint strength for constrained multi-objective optimization, Swarm Evol. Comput. 77 (2023) 101247.
[60] L. Wu, Z. Chen, C. Chen, Y. Li, S. Jeon, J. Zhang, Z. Zhan, Real environment-aware multisource data-associated cold chain logistics scheduling: A multiple population-based multiobjective ant colony system approach, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 23613–23627.
[61] G. Li, W. Wang, W. Zhang, Z. Wang, H. Tu, W. You, Grid search based multi-population particle swarm optimization algorithm for multimodal multi-objective optimization, Swarm Evol. Comput. 62 (2021) 100843.
[62] X. Zhang, S. Wen, D. Wang, Multi-population biogeography-based optimization algorithm and its application to image segmentation, Appl. Soft Comput. 124 (2022) 109005.
[63] C. Dhaenens, L. Jourdan, Metaheuristics for big data, 2016.
[64] J. Cao, J. Zhang, F. Zhao, Z. Chen, A two-stage evolutionary strategy based MOEA/D to multi-objective problems, Expert Syst. Appl. 185 (2021) 115654.
[65] F. Kiliç, Y. Kaya, S. Yildirim, A novel multi population based particle swarm optimization for feature selection, Knowl.-Based Syst. 219 (2021) 106894.
[66] T. Karpagalingam, K. Muneeswaran, An elitism based self-adaptive multi-population Poor and Rich optimization algorithm for grouping similar documents, J. Ambient Intell. Humaniz. Comput. 13 (4) (2022) 1925–1939.
[67] B. Rezaei, F.G. Guimarães, R. Enayatifar, P.C. Haddow, Combining genetic local search into a multi-population imperialist competitive algorithm for the capacitated vehicle routing problem, Appl. Soft Comput. 142 (2023) 110309.
[68] H. Cui, X. Li, L. Gao, An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem, Expert Syst. Appl. 222 (2023) 119805.
[69] J. Huang, J. Wan, B. Lv, Q. Ye, Y. Chen, Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning, IEEE Syst. J. 17 (2) (2023) 2500–2511.
[70] J. Wang, J. Hu, G. Min, A.Y. Zomaya, N. Georgalas, Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Trans. Parallel Distrib. Syst. 32 (1) (2021) 242–253.
[71] J. Wu, W. Yang, X. Han, Y. Qiu, A. Gudkov, J. Song, Hotspot resolution in cloud computing: A Γ-robust knapsack approach for virtual machine migration, J. Parallel Distrib. Comput. 186 (2024) 104817.
[72] 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 10-14, 2014, IEEE, 2014, pp. 4017–4022.
[73] V. Priya, C.S. Kumar, R. Kannan, Resource scheduling algorithm with load balancing for cloud service provisioning, Appl. Soft Comput. 76 (2019) 416–424.
[74] A. Ghasemi, A.T. Haghighat, A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning, Computing 102 (9) (2020) 2049–2072.
[75] 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.
[76] O. Hadary, L. Marshall, I. Menache, A. Pan, E.E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich, T. Moscibroda, Protean: VM allocation service at scale, in: 14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020, USENIX Association, 2020, pp. 845–861.
[77] W. Lin, C. Xiong, W. Wu, F. Shi, K. Li, M. Xu, Performance interference of virtual machines: A survey, ACM Comput. Surv. 55 (12) (2023) 254:1–254:37.
[78] M.C.S. Filho, C.C. Monteiro, P.R.M. Inácio, M.M. Freire, A distributed virtual-machine placement and migration approach based on modern portfolio theory, J. Netw. Syst. Manag. 32 (1) (2024) 2.
[79] M. Xu, W. Tian, R. Buyya, A survey on load balancing algorithms for virtual machines placement in cloud computing, Concurr. Comput. Pract. Exp. 29 (12) (2017).
[80] 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.
[81] Z. Mahmoodabadi, M. Nouri-Baygi, An approximation algorithm for virtual machine placement in cloud data centers, J. Supercomput. 80 (1) (2024) 915–941.
[82] W. Lin, W. Wu, L. He, An on-line virtual machine consolidation strategy for dual improvement in performance and energy conservation of server clusters in cloud data centers, IEEE Trans. Serv. Comput. 15 (2) (2022) 766–777.
[83] L. Fenaux, T. Humphries, F. Kerschbaum, Gaggle: Genetic algorithms on the GPU using pytorch, in: S. Silva, L. Paquete (Eds.), Companion Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2023, Companion Volume, Lisbon, Portugal, July 15-19, 2023, ACM, 2023, pp. 2358–2361.
[84] J.R. Cheng, M. Gen, Accelerating genetic algorithms with GPU computing: A selective overview, Comput. Ind. Eng. 128 (2019) 514–525.
[85] E. Zitzler, D. Brockhoff, L. Thiele, The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration, in: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4403, Springer, 2007, pp. 862–876.
[86] K. Shang, H. Ishibuchi, L. He, L.M. Pang, A survey on the hypervolume indicator in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 25 (1) (2021) 1–20.
[87] J. Blank, K. Deb, Pymoo: Multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.
[88] T. Goh, Taguchi methods: some technical, cultural and pedagogical perspectives, Qual. Reliab. Eng. Int. 9 (3) (1993) 185–202.

[89] X. Sun, Z. Shi, J. Zhu, Multiobjective design optimization of an IPMSM for EVs based on fuzzy method and sequential taguchi method, IEEE Trans. Ind. Electron. 68 (11) (2020) 10592–10600.
[90] M. Li, X. Yao, Quality evaluation of solution sets in multiobjective optimisation: A survey, ACM Comput. Surv. 52 (2) (2019) 26:1–26:38.
[91] AzurePublicDataset. https://github.com/Azure/AzurePublicDatase.

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 or Edge Computing, image recognition especially facial expression recognition, algorithmic theory of machine learning, BigData processing, parallel training of large-scale model and evolution algorithms especially genetic algorithms.

Yuanlun Xie is now a Ph.D. candidate at the School of information and software engineering, University of Electronic Science and Technology of China. His main research interests include algorithmic theory of machine learning, facial expression recognition by deep learning, evolution algorithms, attention mechanism and BigData processing.

Haocheng Lan received Bachelor's degree from the School of information and software engineering, University of Electronic Science and Technology of China. He is now a Master candidate at the School of information and software engineering, University of Electronic Science and Technology of China. His main research interests include NLP, machine learning, evolution algorithms 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 School of information and software engineering, University of Electronic Science and Technology of China (UESTC). His research interests include scheduling in Cloud computing and Bigdata platforms, image recognition by deep learning, algorithmic theory of machine learning, parallel training of large-scale model, mixture of experts and evolution algorithms. 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=160, gindex=340, 134600+ 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.

Kui Wu received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada, in 2002. In 2002, he joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, where he is currently a professor. His current research interests include network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.