Applied Soft Computing 136 (2023) 110027

Contents lists available at ScienceDirect

Applied Soft Computing

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

Growable Genetic Algorithm with Heuristic-based Local Search for multi-dimensional resources scheduling of cloud computing

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

Article history:
Received 6 July 2022
Received in revised form 6 January 2023
Accepted 8 January 2023
Available online 1 February 2023

Keywords:
Cloud computing
Multi-Dimensional Resource
Pareto Solution
Multi-Objective Optimization
Heuristic-Based Local Search
Growable Genetic Algorithm

ABSTRACT

Multi-Dimensional Resources Scheduling Problem (MDRSP, usually a multi-objective optimization problem) has attracted focus in the management of large-scale cloud computing systems as the collaborative operation of various devices in the cloud affects resource utilization and energy consumption. Effective management of the cloud requires a higher performance method to solve MDRSP. Considering the complex coupling between multi-dimensional resources and focusing on virtual machines allocation, we propose GGA-HLSA-RW (GHW, a novel family of genetic algorithms) to optimize the utilization and energy consumption of the cloud. In GGA-HLSA-RW, we add a growth stage to the genetic algorithm and construct a Growable Genetic Algorithm (GGA) using the Heuristic-based Local Search Algorithm (HLSA) with Random multi-Weights (RW) as the growth route. Based on the GHW, we propose GHW-NSGA II and GHW-MOEA/D by applying the sorting strategies and population regeneration mechanism of NSGA II and MOEA/D. To evaluate the performance of GHW, we carry out extensive experiments on the simulation dataset and AzureTraceforPacking2020 for the problems of minimizing the maximum utilization rate of resources for each dimension and minimizing total energy consumption. Experiment results demonstrate the advantages of growth strategy and dimensionality reduction strategy of GHW, as well as validate the applicability and optimality of GHW in realistic cloud computing. The experiments also demonstrate our proposed GHW-NSGA II and GHW-MOEA/D have better convergence rates and optimality than state-of-the-art NSGA II and MOEA/D.

1. Introduction

The emerging trend of IoT and mobile communication accelerates the growth of Internet data urgently demanding large-scale software systems [1]. As a successful distributed computing paradigm, cloud computing is playing an important role in various industries. Cloud computing interconnects extensive heterogeneous server nodes to flexibly provide services for various requests from users including computing requests, storage requests, cache requests and mixed requests [2]. With services almost covering all industries, cloud computing has blossomed into an indispensable impetus in the new network era of IoT currently [3].

The main work units of cloud computing are the components integrated by various micro-circuits including CPU (Central Processing Unit), RAM (Random Access Memory), DS (Disk Storage), GPU (Graphics Processing Unit), BW (BandWidth), etc [4–6]. Constantly expanding requests from users extraordinarily enhance the difficulty and burden to manage resources of the cloud. Some factors noteworthy in resource management comprise heterogeneity of resources, timeliness of response, operation cost, quality of service, etc. These factors are complex and eventually result in the inferior utilization rate of resources and exceedingly massive energy consumption in realistic cloud computing systems.

In practice, the resource bottleneck in any dimension will limit the operating status of the cloud computing system, and then affects the quality of services. Additionally, the inferior utilization rate of resources usually accompanied by low energy conversion efficiency will also cause excessive CO_2 emissions. Therefore, targeting the preservation of social resources, the research on effective multi-dimensional resources scheduling in heterogeneous nodes of the cloud has become a hotspot.

Resource scheduling in cloud computing is defined by [7] as to find an "optimal" mapping "Tasks → Resources" to meet one or

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

https://doi.org/10.1016/j.asoc.2023.110027
1568-4946/

several given objectives. Multi-Dimensional Resource Scheduling Problem (MDRSP) in the heterogeneous nodes of cloud computing, as a multi-objectives problem involving resources in different dimensions, is an NP-hard problem and far more complex than single-objective resource scheduling. Some heuristic algorithms, such as LPT (Longest Processing Time First), FCFS (First Come First Serve) and BFD (Best Fit Decreasing) [8,9], are inappropriate to solve MDRSP. Therefore, meta-heuristic is a common type of algorithm in the existing research on MDRSP such as the modified binary pigeon-inspired algorithm [10], IBSMA (Improved Developed-Slime-Mould-Algorithm) [11], force-directed search [12], NSGA II (Non-dominated Sorting Genetic Algorithm II) [13]. For large-scale cloud computing with ultra-high energy consumption, small-scale improvements to these algorithms will bring considerable significance [14]. Thus, better methods are still necessary especially to obtain the Pareto boundary, where the Pareto boundary is frequently used to evaluate multi-objective optimization algorithms [15,16].

Focusing on the optimization of resource utilization and energy consumption in cloud computing, this paper considers the Virtual Machine (VM) allocation scenario in heterogeneous nodes with Multi-Dimensional Resources (MDRs). In this scenario, we formulate two types of MDRSPs that are minimizing the maximum utilization rate of each dimension of resources and minimizing energy consumption of the total system. Aiming to solve these problems, we apply the concept of stages to divide the classical genetic algorithm into four stages namely initialization stage, infancy stage, mature stage and genetic stage. Based on these four stages of GA (Genetic Algorithm), we add a growth stage for each individual and propose the Growable Genetic Algorithm (GGA) leveraging the Heuristic-based Local Search Algorithm (HLSA) as its growth route with Random multi-Weight-based dimensionality reduction (RW), which can be called GGA-HLSA-RW (abbreviated as GHW). In solving MDRSP, GHW selects the better part of the individual in each generation to generate the offspring, uses RW to reduce the dimensionality of MDRSP to Multi Single–Objective Problems (MSOPs), utilizes HLSA to gain the solution of these dimensionality-reduced MSOPs as the next individual and then updates the solution set of MDRSP. GHW can be regarded as a family of algorithms that allows the combination of various optimization strategies. To further improve the convergence rate and optimality of GHW, we proposed GHW-NSGA II and GHW-MOEA/D applying the sorting strategies and population regeneration mechanism of NSGA II and MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition). Extensive experiments on simulation dataset and experiments driven by the AzureTraceforPacking2020 [17] demonstrate the advantages of our proposed algorithms, where AzureTraceforPacking2020 is a popular public VMs traces representing part of the workload on Microsoft's Azure Compute and is provided by Microsoft Azure for VM allocation [18]. In the scenarios studied in this paper, several multi-objective optimization quality indicators, including hypervolume-over-time and the average probability of finding the theoretically optimal Pareto solutions, show that our proposed GHW family of algorithms has a much better convergence rate and optimality than the algorithms compared, such as NSGA II and MOEA/D.

The main contributions of this paper can be summarized as follows.

(1) GGA: Enlighten by the existing research and natural phenomenon, we use the concept of stages to divide the classical genetic algorithm into four stages called initialization stage, infancy stage, mature stage and genetic stage. Based on this, we add a growth stage to GA and propose the Growable Genetic Algorithm (GGA) which allows the individual in GA to grow through various growth routes.
(2) HLSA-RW: We propose the Heuristic-based Local Search Algorithm (HLSA) as the growth route of GGA and apply Random multi-Weights (RW) to decompose MDRSP to MSOPs. Using the heuristic algorithm as the search route of LSA (Local Search Algorithm) and using LSA as the growth route of the individual in GA are both novel perspectives. Combining GGA, HLSA and RW, we obtain a well-performed GGA-HLSA-RW (GHW) family of algorithms to solve MDRSP. GHW also has a flexible structure to adapt to the combination of various strategies.
(3) GHW-NSGA II and GHW-MOEA/D: We further apply the sorting strategy and population regeneration mechanism of NSGA II and MOEA/D to propose two instantiations of the GHW family i.e., GHW-NSGA II and GHW-MOEA/D, which have better convergence rate and optimality than NSGA II and MOEA/D.
(4) Extensive experiments on the simulation dataset and AzureTraceforPacking2020 [17] with various comparison sights demonstrate the superiority of the GHW family in solving MSRDPs.

The rest of this paper is organized as follows. We review the related work in Section 2. The system model and problem formulation of MDRSP in cloud computing are presented in Section 3. The proposed methodology GHW is presented in Section 4. The experiment design and evaluation results are presented in Section 5. Finally, we conclude this paper in Section 6.

2. Related work

In this section, we briefly review the related work from three aspects: scheduling algorithms in cloud computing, MDRSP and the existing approaches to Multi-objective Optimization Problem (MOP).

2.1. Scheduling algorithms in cloud computing

Approaches to optimize the resource utilization in cloud computing include VMs migration [19], queueing model [20], scheduling algorithm, etc. Among them, the scheduling algorithm is the core. In cloud computing, the existing common categories of scheduling algorithms include heuristic, machine learning and meta-heuristic algorithms.

Heuristic algorithms, generally of low computational complexity, are often used to obtain solutions with acceptable performance. Some classical heuristic algorithms include RR (Round-Robin), LPT, greedy, random, FCFS [8,9] etc. In other search algorithms, they can also be used to generate initial solutions to accelerate convergence, for example in JBA (Jacobi Best-response Algorithm) [21], FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) [22], and LARAC (Lagrange Relaxation based Aggregated Cost) [23].

Machine learning algorithms used for resource scheduling mostly belong to Reinforcement Learning (RL) or Deep Reinforcement Learning (DRL) categories. Some examples are QEEC (Q-learning based framework for Energy-Efficient Cloud) [24] and ADEC (Autonomic Decentralized Elasticity Controller) [25] from the RL category, as well as DQN (Deep Q Network) [26], ADRL (hybrid Anomaly-aware Deep Reinforcement Learning) [6], and DQTS (Deep Q-learning Task Scheduling) [5] from DRL. Combinations of machine learning and other algorithms also adapt to resource scheduling, examples of which are RL+Belief-Learning-Based Algorithm [27], DeepRM-Plus [3] and NN-DNSGA II (Neural Network with Dynamic NSGA II) [28].

The solution space of the NP-hard problem increases exponentially with the increase in data volume. Meta-heuristic is a common method to solve complex optimization problems, especially in big data systems such as the cloud systems [14].

Table 1
Summary of scheduling algorithms in literature from three categories i.e., heuristic, machine learning and meta heuristic.

Categories	Family	Algorithms	Scenarios
Heuristic		LPT [9], FCFS [8], et al.	SOP
Machine learning	RL DRL	QEEC [24], ADEC [25] et al. DQN [26], ADRL [6] et al.	SOP, MOP
Meta heuristic	GA ACO PSO	NSGA II [13], NSGA III [31] et al. MALO [29], S-MOAL [30] et al. MOPSO [33], HAPSO [34] et al.	SOP, MOP

Meta-heuristic algorithms (also evolutionary algorithms always inspired by natural phenomena) include ant colony algorithm such as MALO (Multi-objective AntLion Optimizer) [29] and S-MOAL (Spacing Multi-Objective AntLion algorithm) [30], genetic algorithms such as NSGA II [13], NSGA III [31], MOGA (Multi-Objective Genetic Algorithm) [32] and MOEAs (Multi-Objective Evolutionary Algorithms) [16], Particle Swarm Optimization (PSO) such as MOPSO (Multi-Objective PSO) [33] and HAPSO (Hybrid Adaptive PSO) [34], Artificial Bee Colony (ABC) [35], as well as Firefly Algorithm (FA) [36]. Searchability of solution enables meta-heuristics to utilize local search algorithm or other meta-heuristics as its input to accelerate the convergence, for example: OEMACS [37] leveraged OEM (Order Exchange and Migration) local search techniques; ACO-GA [38], HGA-ACO [39] and DAAGA (Dynamic Ant-colony Algorithm and Genetic Algorithm) [40] leveraged Ant Colony Optimization (ACO) and GA to optimize the search process.

For intuitive observation, we summarize the scheduling algorithms in Table 1.

2.2. MDRSP in cloud computing

For realistic cloud computing systems, many types of resources need to be arranged simultaneously. The working status of each resource may affect that status of others on the same physical machine, which increases the difficulty of research on MDRSP. Aiming at optimizing resource utilization, energy consumption and cost, researchers have carried out numerous studies on MDRSP in the cloud. In addition to the meta-heuristics reviewed in Section 2.1, we continue to review some other research on MDRSP.

Goudarzi and Pedram [12] proposed a force-directed search to solve the multi-dimensional SLA-based resource allocation problem in the cloud considering power, memory and bandwidth. Xie et al. [41] designed MPTMG (Multi-dimensional Pricing mechanism based on Two-sided Market Game) for distributed MDR allocation in mobile cloud computing considering storage, bandwidth and CPU in cloudset. Bao et al. [42] proposed MECC (Multi-dimensional resource allocation-Enabled Cloud Cache), a SLA-aware cloud cache framework, to achieve both the SLA ensurance and cost optimization for NVM-based cloud cache. Pan et al. [43] proposed a MDR sharing framework for heterogeneous nodes to reduce the total cost and task failure probability. Combining Lyapunov optimization and Lagrange dual decomposition, Yu et al. [44] proposed MERITS (Multi-timescale multi-dimEnsion Resource allocatIon and Task Splitting algorithm) to reduce energy consumption, queuing delay, queue backlog and increase connection success ratio. Gopu and Venkataraman [45] applied MOEA/D to solve optimal VM placement in the cloud considering wastage, power consumption and propagation delay simultaneously.

In addition, research on MDRSP is also a hot topic in other scenarios. For multi-dimensional knapsack problem, negative learning in ant colony optimization [46], sum-of-ratios-based decomposition [47], Modified-BPIO (Modified Binary Pigeon-Inspired Optimization) [10] and IBSMA [11] were proposed and achieved considerable performances. For multi-dimensional transport problems, Aktar et al. applied three ways, i.e., weighted sum technique, max–min Zimmermann technique and neutrosophic programming technique, to reduce multi-objective to single-objective and then used generalized reduced gradient method for solutions [48]. For diverse safety message transmissions in vehicular networks, Chen et al. [49] developed a MDR allocation scheme to jointly optimize the sensing resource allocation.

2.3. Existing approaches to MOP

MDRSP is actually one of the Multi-objective Optimization Problems (MOP, also known as multicriteria optimization) [50,51]. Solving a MOP generally requires two aspects: evaluation indicator of solutions and simplification of problems. These two aspects also extended various optimization algorithms to MOP. In this subsection, we will review them from these two aspects.

Evaluation indicator-based methods include two popular types: non-dominated sorting-based method [51,52] and hypervolume-based method [53,54].

For the Non-dominated Sorting (NS) based method, Srinivas and Deb [55] proposed Non-dominated Sorting Genetic Algorithms (NSGA). Based on the concept of Pareto optimization, NSGA stratifies the individuals according to their dominant and non-dominant relationships, which improves the convergence rate to solve MOPs [55]. Using elite strategy and congestion comparison operator, Deb et al. [56] proposed NSGA II, which guarantees the uniform distribution of the non-inferior optimal solution. Subsequently, NSGA III [31] applied reference points to replace congestion sorting of NSGA II, which is more suitable to high dimensional MOP. Other variants of NSGA include B-NSGA III, U-NSGA III [51], NSGA II-C [52], NN-DNSGA II algorithm [28], et al.

HyperVolume (HV), proposed by Zitzler et al. is an important indicator to evaluate the optimality of the Pareto solution set [57,58]. The indicator HV also extended a novel type of approach (HV-based method) to solve MOP. Some examples are R2HCA-EMOA (R2-based Hypervolume Contribution Approximation Evolutionary Multi-objective Optimization Algorithm) [53] and UHV-GOMEA (Uncrowded HyperVolume and Gene-pool Optimal Mixing Evolutionary Algorithm) [54].

Simplification of problems-based methods mainly includes some dimension reduction-based methods.

Dimension reduction in MOP means using some methods to obtain problems with fewer objectives by decomposing MOP into Multi Lower-Dimensional objective Problems (MLDPs) [59–61]. Brockhoff and Zitzler [61] proposed an exact algorithm and fast heuristics to reduce the dimensions of objectives to assist evolutionary MOP with large dimensions. Ruochen Liu et al. [59] proposed a clustering and dimensionality reduction-based evolutionary algorithm for large-scale MOP with dimensions up to 5000. Zheng Tan [60] et al. proposed multi-stage dimension reduction to make surrogate-assisted evolutionary algorithms capable to handle sparse MOPs.

A specific case of dimension reduction is scalarization, which means decomposing the MOPs into Multiple Single-Objective Problems (MSOPs) [50]. Aggregating the objectives into a weighted sum is a frequent approach to scalar the MOP [50]. The weighted sum approach enables computation of the properly Pareto optimal in convex cases, while may work poorly in non-convex cases [50]. Other approaches including ϵ-constraint method, Benson's method, and compromise programming are applicable in non-convex solution space to transform MOP to SOPs [50]. MOEA/D, proposed by Qingfu Zhang and Hui Li [62], is a typical scalarization-based method, which combines genetic

Table 2
Summary of approaches to mops in literature from two aspects i.e., indicator- and simplification-based approaches.

Aspect	Approaches	Family	Algorithms
Indicator-based	NS-based HV-based	NSGA	NSGA II [56], NSGA III [31], B-NSGA III [51] et al. R2HCA-EMOA [53], UHVI [54] et al.
Simplification-based	Dimension reduction-based	MOEA/D Others	MOEA/D [62], MOEA/HD [63], MOEA/D-TS [64] et al. ϵ-constraint method, Benson's method [50] et al.

algorithms and a weighted sum method [50]. On the basis of MOEA/D, Hang Xu et al. [63] proposed a novel MOEA based on Hierarchical Decomposition (MOEA/HD) which decomposed the MOP into subproblems layered in different hierarchies. Jie Cao et al. [64] proposed MOEA/D-TS (a Two-Stage evolutionary strategy based MOEA/D) to improve MOEA/D. In MOEA/D-TS [64], the first stage focused on pushing the solutions into the area of the Pareto front to speed up its convergence ability, as well as the second stage conducted in the operating solution's diversity to make the solutions distributed uniformly.

For the sake of observation, we summarize these approaches in Table 2.

2.4. Analysis of related work

MDRSP, as a type of challenging MOPs, has complex problem features and discontinuous solution spaces. One common type of its method is the meta-heuristic algorithm. Among them, NSGA family and MOEA/D family are the most popular to solve MOPs. Some comparative studies between NSGA family and MOEA/D family [65,66] showed both these two families of algorithms have good convergence in continuous multi-objective optimization space. When in large-scale discrete solution space, the searching ability of these algorithms is insufficient. Therefore, they require abundant population size and generations to obtain an acceptable solution, which will cost a lot of computing time. Moreover, their local optimal solutions may be far from the theoretical optimal solutions due to their essential characteristics.

Conventionally, a genetic algorithm contains several processes: initializing the individuals, selecting the excellent individuals to participate in the pairing, and executing crossover and mutation to generate the children individuals, regenerate the population with a specific mechanism to generate the next generation. From the above review, the existing genetic algorithms mainly focus on the improvement of the population selection strategies and population regeneration mechanisms, such as non-dominated sorting, elite strategy, and competition mechanism [67–69], which have improved the searchability and local optimum of GA to some extent. However, as they do not pay special attention to the growth process of the individuals outside the crossover and mutation, the convergence rate and optimality need to be further improved.

Referring to the previous research, this paper reorganizes the process of genetic algorithms by the concept of stages and adds a growth stage for each individual to obtain a novel architecture of genetic algorithm called Growable Genetic Algorithm (GGA). The GGA allows the combination of various algorithms and the individuals have more flexible evolutionary routes. To solve the MDRSP in cloud computing, we propose the Heuristic-based Local Search Algorithm (HLSA) to instantiate the growth route of GGA and use Random multi-Weights (RW) as the growth direction of individuals. Combining the above components, we propose GGA-HLSA-RW (GHW), which can effectively solve MDRSP in the cloud.

3. System model and problem formulation of MDRSP

To assist with the system model and problem formulations, Table 3 lists the descriptions of some notations in this paper.

Table 3
Notations and descriptions.

Notation	Description
n	Number of tasks or VMs
m	Number of nodes
d	Number of dimensions of resources
i	Index of task or VMs
j	Index of nodes
k	Index of dimensions of resources
V_i	The task or VMs with index i
P_j	The node with index j
C_i	The property matrix of V_i
C_{ij}^{CPU}	The CPU capacity requested by T_i allocated on P_j in unit of MIPS (Million Instructions Per Second)
C_{ij}^{RAM}	The RAM capacity requested by T_i allocated on P_j in unit of Gigabytes
C_{ij}^{DS}	The Disk storage requested by T_i allocated on P_j in unit of Gigabytes
C_{ij}^{GPU}	The GPU capacity requested by T_i allocated on P_j in unit of Gigabytes
C_{ij}^{BW}	The bandwidth of network requested by T_i allocated on P_j in unit of Mbps (Million bits per second)
ψ_j	Set of tasks and VMs in node P_j
κ	The set of ψ_j where $\kappa = \langle \psi_1, \psi_2, \ldots, \psi_m \rangle$
x_{ij}	If $V_i \in P_j$ then $x_{ij} = 1$, otherwise $x_{ij} = 0$
L_{jk}	The limited capacity of resource in the kth dimension of the node P_j
S_{jk}	The load of resource in kth dimension of the node P_j
U_{jk}	The utilization rate in kth dimension of the node P_j
u_{ijk}	The resource occupancy rate of V_i for the kth dimension in P_j
$G_{jk}(S_{jk})$	The function between the load of resource in kth dimension of server node P_j and energy consumption
E_j	The total energy consumption the node P_j
E	The total energy consumption of the cloud system
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

3.1. Cloud system model with multi-dimensional resources

A cloud computing system usually consists of a large number of server nodes and integrates the resource layers of these nodes through the high-speed network as Fig. 1. We demonstrate cloud servers as heterogeneous nodes because of the default supportability of the cloud systems for heterogeneity. Then, we model it as a multi-dimensional system model and model the problem of VMs allocation in it as a MDRSP.

We consider a cloud system with m heterogeneous nodes (denoted as $P = \langle P_1, P_2, \ldots, P_m \rangle$) and each node with d-dimensions of resources such as CPU, RAM, disk storage, GPU, bandwidth etc. The set of tasks and VMs in a time slot $[t, t + \delta t)$ is denoted as $V = \langle V_1, V_2, \ldots, V_n \rangle$. The executions of the same request are different in different nodes and always need multi resources synergistically. Thus, we assume that a task or VM request from users equals a request for resources in multiple dimensions. Based on the above, we can set the property of a task or VM V_i as a matrix $C_i = \left\{C_{ijk}\right\}_{1 \le j \le m, 1 \le k \le d}$. The jth row $C_{ij} = \left\langle C_{ij}^{CPU}, C_{ij}^{RAM}, C_{ij}^{DS}, C_{ij}^{GPU}, C_{ij}^{BW}, \ldots \right\rangle$ denotes the capacity request for resources in each dimension when V_i is allocated to the jth node.

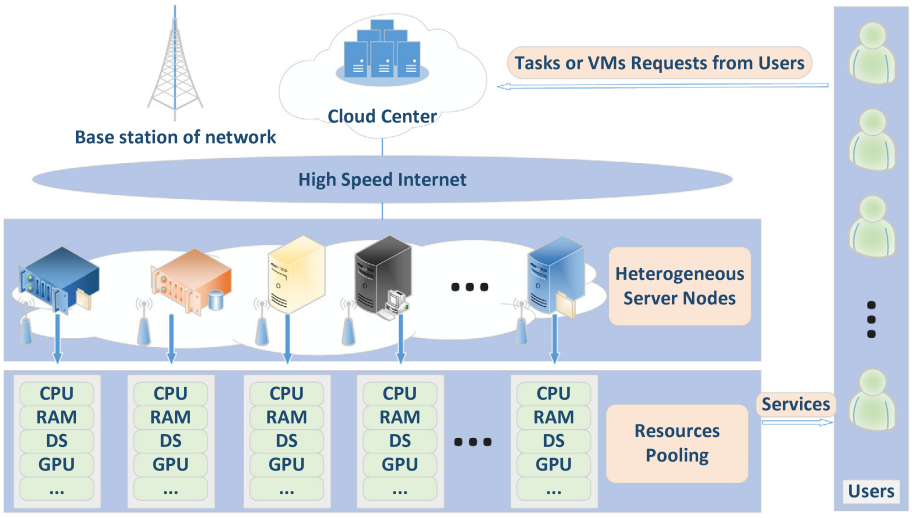


Fig. 1. Structure of cloud computing with various resources.

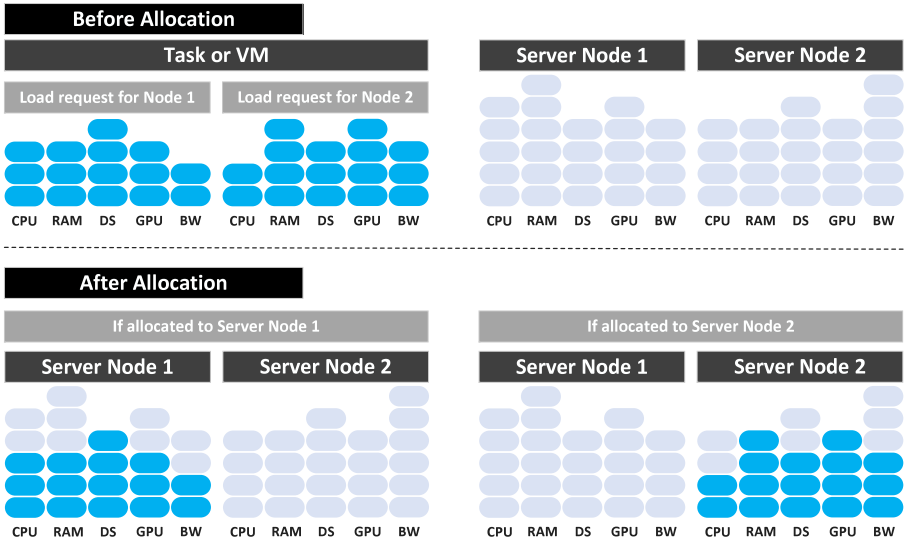


Fig. 2. Allocation of a task or VM to heterogeneous nodes with multi-dimensional resources.

Each node has limited capacity in each dimension (i.e., the maximum load for healthy operation of components) that can be set as $L = \{L_{jk}\}_{1\le j\le m, 1\le k\le d}$ where $L_j = \langle L_j^{CPU}, L_j^{RAM}, L_j^{DS}, L_j^{GPU}, L_j^{BW}, \ldots\rangle$. For example, L_j^{CPU} is the jth node's capacity of CPU and L_j^{DS} is its disk size. This model corresponds to various requests (computing requests, storage requests, cache requests, transmission requests, VMs requests, etc.) involved in cloud computing systems.

A diagram of the allocation of tasks or VMs to heterogeneous nodes with MDRs is shown in Fig. 2. Although VMs migration and task segmentation can also be leveraged to solve the resource scheduling in the cloud, they still cannot avoid the allocation of tasks or VMs. In view of this, we do not consider the VMs migration and task segmentation. Then, we mainly focus on the direct allocation of tasks or VMs where any task or VM cannot be further split into smaller ones. This also means any task or VM will be allocated to only one server node supporting affinity while one server node can process multiple tasks or VMs simultaneously. We denote the set of tasks and VMs in node P_j as ψ_j. If a task or VM V_i is allocated to the node P_j, we use $V_i \in \psi_j$. The ψ_j of each node constitutes a vector $\kappa = \langle \psi_1, \psi_2, \ldots, \psi_m\rangle$. Therefore, we can gain the relationships of ψ_j that $\bigcup_{j=1}^{m} \psi_j = V$, $\psi_j \subset \kappa$, and $\psi_j \bigcap \psi_l = \emptyset$ for $\forall 1 \le j \ne l \le m$. κ determines the unique allocation result corresponding to the solution of MDRSP.

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 $S_j = \langle S_{j1}, S_{j2}, \ldots, S_{jd}\rangle$. 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). A diagram of Eq. (1) is presented in Fig. 3.

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

3.2. Problem formulations for resources utilization and energy consumption

Cloud computing is based on the pay-as-you-go pattern [3] and regards the resources as ubiquitous "cloud". Generally, cloud has several targets: providing as many services as possible, ensuring flexibility in providing services, reducing the overall energy consumption, optimizing the resource utilization and prolonging

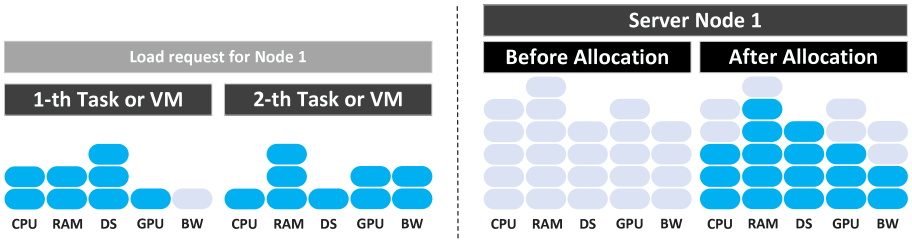


Fig. 3. Relationship of linearly superposition for multi-dimensional resources allocating two tasks or VMs to one server node.

the service life of components. In this paper, we transform its aims and focus on two problems:

(1) Minimizing the maximum utilization rate of resources for each dimension under all nodes;
(2) Minimizing the energy consumption for the whole system.

3.2.1. Minimizing the maximum utilization rate of resources for each dimension under all nodes

There are various indicators to evaluate balancing degree in the cloud such as variance or standard deviation of load [5,36], average success rate [70], coefficient of variance [71], degree of imbalance [28], etc. In this paper, we leverage the objectives of minimizing the maximum utilization of resources in each dimension. It is also a method to perform load balancing, improve resource utilization and ensure that the cloud system can process more VMs. The problem with multi-objectives can be formulated as Eq. (2), where we denote $\min \omega_k^{(1)} = \min \max (S_{1k}, S_{2k}, \ldots, S_{mk})$ and $S_{jk} \le L_{jk}$ for $\forall j \in \{1, 2, \ldots, m\}$ and $\forall k \in \{1, 2, \ldots, d\}$.

$$\begin{aligned} \min \omega^{(1)} &= \left(\min \max_{j=1,2,\ldots,m} S_{jk} \right) \Big|_{k=1,2,\ldots,d} \\ &= \min \begin{cases} \max (S_{11}, \ldots, S_{m1}) \\ \max (S_{12}, \ldots, S_{m2}) \\ \ldots \\ \max (S_{1d}, \ldots, S_{md}) \end{cases} \end{aligned} \tag{2}$$

Converting Eq. (2) to zero–one integer programming problem can obtain Eq. (3).

$$\min \omega_k^{(1)} = \min \left(\max_{j=1,2,\ldots,m} \left(\sum_{i=1}^{n} x_{ij} C_{ijk} \right) \right) \tag{3}$$

We assume the resource utilization rate of each dimension as the ratio of the occupied load to the limited capacity that is:

$$U_{jk} = \frac{S_{jk}}{L_{jk}} = \frac{\sum_{i=1}^{n} x_{ij} C_{ijk}}{L_{jk}} \tag{4}$$

In the system model of this paper, L_{jk} and C_{ijk} are given and invariant. Thus, we can denote a parameter $u_{ijk} = C_{ijk}/L_{jk}$ to express the occupancy rate of a single VMs V_i for the kth dimension of node P_j. The utilization rate u_{ijk} also satisfies the superposition relationship:

$$U_{jk} = \sum_{V_i \in \psi_j} u_{ijk} = \sum_{i=1}^{n} x_{ij} u_{ijk}. \tag{5}$$

Thus, if an algorithm can adapt to C_{ijk}, it can also apply to u_{ijk}, and vice versa. Then, a problem to reduce the utilization rate of resources for each dimension is as Eq. (6).

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

where the constraints are:

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

3.2.2. Minimizing the total energy consumption for system

Minimizing the number of working nodes is a frequent way to optimize the energy consumption of the cloud, whose optimization objective can be written as:

$$\min \omega^{(3)} = \min \sum_{j=1}^{m} \max_{i=1,2,\ldots,n} x_{ij} \tag{8}$$

The use of Eq. (8) requires the assumption that the operating energy consumptions of all nodes are similar. However, the ratio between load and energy consumption may be varying with the different nodes. Therefore, we consider the relationship between the energy consumption and load to lean closer to the actual scene. We assume $G_{jk}\left(S_{jk}\right)$ as the function between the load of resource in kth dimension of server node P_j and its required energy consumption where we denote E_j as the total energy consumption the node P_j. In reality, $G_{jk}\left(S_{jk}\right)$ is often non-linear related to the status of sever nodes such as temperature. In this paper, we do not address the issue of the relationship between load, energy consumption and status. Thus, we assume each $G_{jk}\left(S_{jk}\right)$ is a given function. Without losing generality, we also assume the energy consumption of all dimensions of resources is subject to superposition. Then, formulas for energy consumption can be obtained as:

$$E = \sum_{j=1}^{m} E_j = \sum_{j=1}^{m} \sum_{k=1}^{d} G_{jk}\left(S_{jk}\right) \tag{9}$$

The components in the computer mainly process the task by switching high-low voltage signals. According to Ohm's law, the electrical power is equal to the square of the voltage divided by the resistance. Thus, we set up the function $G_{jk}\left(S_{jk}\right)$ as a quadratic polynomial function in this paper:

$$G_{jk}\left(\sum_{k=1}^{q} S_{jk} \right) = a_{jk} S_{jk}^2 + b_{jk} S_{jk} + c_{jk} + sgn\left(S_{jk}\right) d_{jk} \tag{10}$$

where $sgn()$ is the signum function. When $\sum_{k=1}^{q} S_{jk} = 0$, $c_{jk} = E_{jk}^{idle}$ corresponds to the energy consumption of kth dimensional resource when node P_j is idle. When $\sum_{k=1}^{q} S_{jk} > 0$, $c_{jk} + d_{jk} = E_{jk}^{on}$ corresponds to the energy consumption of kth dimensional resource when node P_j is on working. Then, the total energy

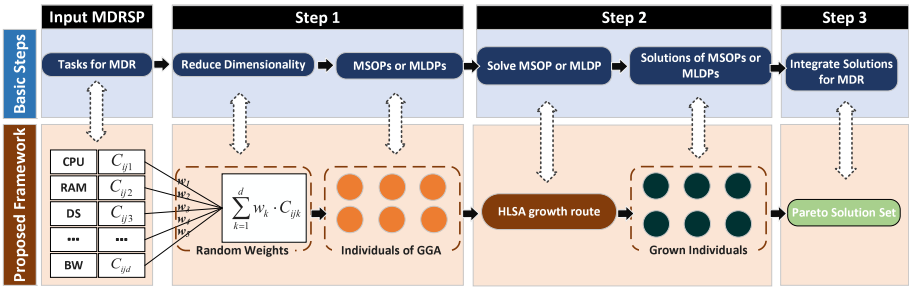


Fig. 4. Basic steps of our proposed framework to solve MDRSPs.

consumption of the system can be rewritten as follows by substituting x_{ij} and Eq. (10) into Eq. (9).

$$E = \sum_{j=1}^{m}\sum_{k=1}^{d}\left(a_{jk}\left(\sum_{i=1}^{n} x_{ij}C_{ijk}\right)^2 + b_{jk}\left(\sum_{i=1}^{n} x_{ij}C_{ijk}\right)\right) + \sum_{j=1}^{m}\sum_{k=1}^{d}\left(c_{jk}\right) + \sum_{j=1}^{m}\left(\max_{i=1}^{n}\left(x_{ij}\right)\sum_{k=1}^{d} d_{jk}\right) \tag{11}$$

Obviously, the objective of minimizing the total energy consumption is as:

$$\min \omega^{(4)} = \min E \tag{12}$$

where the constraints are also as Eq. (7).

4. Methodology for MDRSP: GGA-HLSA-RW (GHW) and its instantiations

For MDRSP, the commonly used three basic steps to obtain a solution set are as:

(1) Reducing dimensionality of the MDRSP to Multi Single-Objective Problems (MSOPs) or Multi Lower-Dimensional objective Problems (MLDPs);
(2) Solving the solutions of MSOPs or MLDPs;
(3) Integrating the solutions of MSOPs or MLDPs to obtain the solution set of the original MDRSP.

Therefore, we need to achieve these three steps by building corresponding methods to solve MDRSP. We map them into a growable genetic algorithm framework as shown in Fig. 4. In this framework, we leverage the Random multi-Weights method (RW) to achieve dimensionality reduction and use the individuals of GGA to represent the dimensionality-reduced MSOPs; propose the HLSA-based growth route to represent the solving process of each MSOP and obtain the mature individuals after HLSA-based growth as the solution of MSOP; finally, update the Pareto solution set of MDRSP by integrating the grown mature individuals.

The key to our proposed framework is the Growable Genetic Algorithm (GGA) using HLSA as the growth route. To illustrate its structure and highlight the difference between it and the classical GA, we present their flowchart in Fig. 5, where the flowchart of the classical GA is obtained by integrating the genetic-related literature [13,31,32,55,56].

As shown in Fig. 5, we apply a concept of stages to divide the classical GA into four stages namely initialization stage, infancy stage, mature stage and genetic stage. The classical GA firstly takes the initialized individuals as the early individuals of the infancy stage; secondly selects some excellent individuals in the mature stage according to some strategies such as fast non-dominated sorting algorithm [31,56]; then pairs the selected individuals; executes crossover and mutation to generate the children individuals; finally, choose a part of individuals as the individuals of the next infancy stage.

On the basis of the four stages of the classical GA, we add a growth stage in GGA shown as the part in the red box of Fig. 5. In the growth stage, we randomly entrust weights to the individuals as their directions of ability cultivations, use the Heuristic-based Local Search Algorithm (HLSA) to cultivate the individuals of the infancy stage, and then select the grown individuals for the subsequent genetic process. The application of a novel growth stage is conducive to quickly obtaining better optimization solutions for MDRSP. Finally, we obtain our proposed Growable Genetic Algorithm (GGA) using the HLSA-based growth route with Random multi-Weights (RW) named GGA-HLSA-RW (abbreviated as GHW).

Next, we will present the components of GGA-HLSA-RW (GHW) respectively including RW, HLSA, and GGA based on various growth strategies.

4.1. Random multi-weights-based dimensionality reduction

In this paper, we mainly consider transforming MDRSP into MSOPs. However, for strict consideration, the framework of this paper is also suitable for transforming MDRSP into MLDPs. Therefore, we still use dimensionality reduction (or decomposition) instead of scalarization in this paper.

Before discussing the method of dimensionality reduction, we first presuppose we have a valid and efficacious algorithm (marked as A_h) to solve the single-dimensional resource scheduling problem. Taking $\omega_k^{(2)}$ in Eq. (6) as an example, it can be regarded as a single-dimensional resource scheduling problem (also a single-objective optimization problem) regardless of other dimensions. A_h can gain a sufficiently optimized solution of Eq. (6) for the kth dimension based on our presuppose. Assuming minimizing the weighted sum of utilization in each dimension marked as $\bar{\omega}_{(w)}^{(2)}$, modification of Eq. (6) can gain a problem as:

$$\min \bar{\omega}_{(w)}^{(2)} = \min\left(\max_{j=1,2,\ldots,m}\left(\sum_{i=1}^{n} x_{ij}\sum_{k=1}^{d}\left(w_k \cdot u_{ijk}\right)\right)\right) \tag{13}$$

where $w = \langle w_1, w_2, \ldots, w_d\rangle$ is a vector of weights. It can be set $W_{ij} = \sum_{k=1}^{d} w_k \cdot u_{ijk}$. Because w and u_{ijk} are given and invariant, W_{ij} is a constant, which means Eq. (13) is a single-dimensional problem and has the same form as Eq. (6). More crucially, the algorithm A_h can apply to solve it and obtain a sufficiently optimized solution. Other problems are analogous to this property. For sake of the following discussion, we set the solution in each dimension of Eq. (13) obtained by A_h as $A_h\left(\bar{\omega}_{(w)}^{(2)}\right)$ where:

$$A_h\left(\bar{\omega}_{(w)}^{(2)}\right) = \left\langle \omega_1^{(2)}, \omega_2^{(2)}, \ldots, \omega_d^{(2)}\right\rangle \tag{14}$$

Therefore, $\omega_k^{(2)} = \max_{j=1,2,\ldots,m}\left(\sum_{i=1}^{n} x_{ij}\sum_{k=1}^{d}\left(w_k \cdot u_{ijk}\right)\right)$ where $\{x_{ij}\}$ is the obtained optimization solution of Eq. (13) solved by A_h.

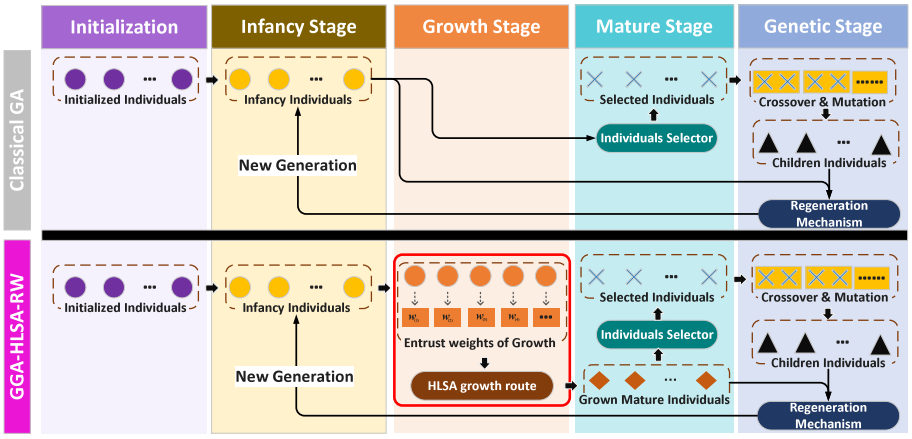


Fig. 5. The flowchart comparison between the classical GA and our proposed GGA-HLSA-RW (GHW).

The weight vector w is actually an angle to slice the multi-dimensions of resources and W_{ij} is the projection of this slice. This enlightens us to analyze whether the optimal solutions for the weights of all angles can cover all Pareto solutions. A general MDRSP can be assumed as:

$$\min \omega = \min \begin{cases} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_L \end{cases} \tag{15}$$

where:

$$\omega_k = f\left(\left\{x_{ij}Y_{ijk}\right\}_{1\le i\le n, 1\le j\le m}\right) \tag{16}$$

which expresses a function related to all the elements of the $n \times m$ matrix $\left\{x_{ij}Y_{ijk}\right\}_{1\le i\le n,1\le j\le m}$ where Y_{ijk} is given parameter, and the constraints are as Eq. (7). Assuming T_w as a weight set traversing all angles in Euclidean space $\mathbb{R}^L$, then we can denote the optimization solution set under each weight of T_w as B where:

$$B = \left\{A_h(\omega_{(w)}) | \forall w \in T_w\right\}, \tag{17}$$

theoretical Pareto solution set of Eq. (15) as D, and the set of solutions included in B as G subject to properties that: for $\forall g \in G$ and $\forall b \in B$, b does not dominate g; for $\forall b \in B - G$, $\exists g \in G$ **s.t.** g dominates b. Therefore, the key is to find a T_w **s.t.** $card(D - G)$ as small as possible, where $card(D - G)$ means the cardinality of set $D - G$.

This reveals a method to obtain the Pareto solutions of MDRSP which is to traverse weights of all angles. The way to approximately perform traversing weights is enumerated by cyclic isometrics based on a definite interval. However, the weight set T_w is continuous containing infinite elements, and the solutions $A_h(\omega_{(w_1)})$ and $A_h(\omega_{(w_2)})$ of two weights w_1 and w_2 expressing different angles may be the same. Therefore, differentiating from cyclic isometric, a random multi-weights is applied to obtain the slices of reduced dimension. It means to select multi-weights randomly to generate corresponding single-dimensional problems and respectively obtain the solutions of these problems, as Algorithm 1.

As the growth stage of GGA allows the combination of various algorithms, a multi-objective optimization algorithm (including GHW itself) can be nested in the growth stage. However, considering this paper focuses on the proposal of the GHW framework (a novel genetic-based framework) and the validation of its applicability in MDRSP, this paper mainly uses a group of random weights to transform a MOP into MSOPs at the growth stage of each generation, which also means the weights is variable after a generation.

Algorithm 1: Random multi-weights-based dimensionality reduction

Input : MDRSP with form as Eq. (15), constraints as Eq. (7) and the parameters of tasks as Y_{ij} where $Y_{ij} = \left\langle Y_{ij1}, Y_{ij2}, \ldots, Y_{ijd}\right\rangle$

Output: Solution set B of decomposed problems

1 **Generate** random weight set $T_w = \left\langle w_{(1)}, w_{(2)}, \ldots\right\rangle$ with multiple weights where $w_{(l)}$ is vector with length d

2 **Obtain** dot product $w_{(l)} \cdot Y_{ij}$ as new parameters

3 **Obtain** the problem after dimensionality reduction as:

$$\min \omega_{(w_l)} = \min f\left(\left\{x_{ij}\left(w_{(l)} \cdot Y_{ij}\right)\right\}_{1\le i\le n, 1\le j\le m}\right) \tag{18}$$

where Eq. (18) is single-dimensional problem in $w_{(l)}$

4 **Solve** the problem Eq. (18) by the given algorithm A_h

5 **Obtain** the solution set B as Eq. (17)

4.2. Heuristic-based local search algorithm

The next key is an effective algorithm A_h to gain the optimal solution for dimensionality-reduced problems, which directly determines the optimality of the solution for MDRSP. To improve the solution of A_h, we proposed the Heuristic-based Local Search Algorithm (HLSA) combining the superiorities of heuristic and local search.

Entrusting the heuristic algorithm the role of search route, we define a neighborhood in the heuristic-based local search algorithm as: It can be assumed that two solutions $\kappa = \langle \psi_1, \psi_2, \ldots, \psi_m\rangle$ and $\kappa' = \left\langle \psi'_1, \psi'_2, \ldots, \psi'_m\right\rangle$. If $\exists j_1 \neq j_2 \in \{1, 2, \ldots, m\}$ subject to that $\psi_l = \psi'_l$ for $\forall l \in \{1, 2, \ldots, m\} - \{j_1, j_2\}$. And if the result after reallocating the VMs set $\psi_{j_1} \cup \psi_{j_2}$ on two server nodes P_{j_1} and P_{j_2} through the specific heuristic algorithm is κ'. Then κ' is defined as the corresponding heuristic-based neighbor of κ and all heuristic-based neighbors of κ consist of a solution set as $Ner(\kappa)$.

A HLSA-based neighbor solution of a given solution can be also described as the solution obtained by the given solution by calling the HLSA algorithm once. If $\forall \kappa' \in Ner(\kappa)$, the solution of κ' is not optimal than that of κ, stop the local search and regard κ as a local optimum. In the proposed GHW, we set additional criteria to stop the HLSA search of one individual in one generation, which can be called the number of growth steps (denoted as G_{step}). Then, the heuristic-based local search algorithm with G_{step} criteria can be seen in Algorithm 2.

Then, an appropriate heuristic algorithm as the search route is the critical factor. Because this framework has great adaptability

Algorithm 2: Heuristic-based local search algorithm

Input : Dimensionality reduced problem $\omega_{(w_l)}$ as Eq. (18) and the number of growth step (G_{step})
Output: Solution $A_h\left(\omega_{(w_l)}\right)$ of the problem

Initially Allocate tasks to resources with confirmed or random initialization policy and gain the general initial status of κ
Set $i = 0$, $Exists\_Ner = True$
while Exists_Ner and $i < G_{step}$ **do**
  $Exists\_Ner = False$, $i++$
  Search neighborhood $Ner(\kappa)$ of κ in the specific heuristic-based local search algorithm such as Algorithm 3 (modified LSPT-based local search algorithm)
  if $\kappa' \in Ner(\kappa)$ ***s.t.*** *the solution of* κ' *is optimal than* κ **then**
    $Exists\_Ner = True$
    Choose the optimal neighbor to update $\kappa = \kappa'$
Set κ as the solution $A_h\left(\omega_{(w_l)}\right)$ of the problem

Algorithm 3: Modified LSPT-based local search algorithm for heterogeneous nodes

Input : Tasks in $\psi = \psi_{j_1} \cup \psi_{j_2}$ of two server nodes P_{j_1} and P_{j_2}
Output : ψ'_{j_1} and ψ'_{j_2}
Initialize $Mark_{j_1} = 0$, $Mark_{j_2} = 0$, $\psi'_{j_1} = \emptyset = \psi'_{j_2}$
while $\psi \neq \emptyset$ **do**
  if $Mark_{j_1} \leq Mark_{j_2}$ **then**
    $\alpha = j_1$, $\beta = j_2$
  else
    $\alpha = j_2$, $\beta = j_1$
  Collect tasks $V_\tau \in \psi$ **s.t.** $w_{(l)} \cdot \left(Y_{\tau\alpha} - Y_{\tau\beta}\right) = \min\limits_{V_i \in \psi} w_{(l)} \cdot \left(Y_{i\alpha} - Y_{i\beta}\right)$ to obtain a set of $\left\langle V_{\tau_1}, V_{\tau_2}, \ldots, V_{\tau_s}\right\rangle$
  if $s \geq 2$ **then**
    Choose V_τ **s.t.** $w_{(l)} \cdot Y_{\tau\alpha} = \max\limits_{1 \leq p \leq s} w_{(l)} \cdot Y_{\tau_p\alpha}$
  $Mark_\alpha += w_{(l)} \cdot Y_{\tau\alpha}$, $\psi'_\alpha += \{V_\tau\}$ and $\psi - = \{V_\tau\}$

to various algorithms, almost any algorithm that can solve the resource scheduling of two nodes can be applied as its search route. Without losing generality, we propose the modified LSPT (modified Longest-Shortest Process Time) to search the heuristic-based neighbor considering the performance to address the single-dimensional resource scheduling of heterogeneous nodes. A modified LSPT-based neighborhoods can be defined as: It can be assume κ' is a HLSA-based neighbor of κ, $\psi'_{j_1} \cup \psi'_{j_2} = \psi_{j_1} \cup \psi_{j_2} = \left\{V_{\tau_1}, V_{\tau_2}, \ldots\right\}$ and $\xi_{\tau_i} \geq \xi_{\tau_{i+1}}$ where $\xi_{\tau_i} = w_{(l)} \cdot \left(Y_{\tau j_1} - Y_{\tau j_2}\right)$. If $V_{\tau_i} \in \arg\min_{\psi'}\left(\sum_{V_{\tau_k} \in \psi'_1} w_{(l)} \cdot Y_{\tau j_1}, \sum_{V_{\tau_k} \in \psi'_2} w_{(l)} \cdot Y_{\tau j_1}\right)$ 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'. Then, the modified LSPT-based local search algorithm is presented in Algorithm 3. In order to minimize the maximum utilization of all heterogeneous nodes, modified LSPT requires sorting the VMs in two nodes according to ξ_{τ_i} (i.e., the difference of the weighted utilization of a VM in two nodes) and $\psi = \left\{V_{\tau_1}, V_{\tau_2}, \ldots\right\}$ is an ascending set. In the process of putting VMs into nodes one by one: if the sum of weighted utilization of P_{j_1} is smaller than that of P_{j_2}, put the leftmost of the remaining ψ into ψ'_{j_1}; otherwise, put the rightmost of the remaining ψ into ψ'_{j_2}. Then, update ψ by removing the currently selected VM from ψ. The process of modified LSPT combines the characteristics of LPT and SPT so as to well solve the VMs allocation in two heterogeneous nodes.

4.3. Growable genetic algorithm based on growth strategies

As is known, the local search algorithm may converge to a non-global optimal solution. Thus, we consider using a genetic algorithm to increase the search scope and to improve the probability of jumping out of the local optimum. A feasible combination of local search and genetic algorithm is to use the local search algorithm as the growth route of the individuals of the genetic algorithm (i.e., GGA-HLSA).

Inspired by the existing research, we consider different individuals in the genetic algorithm to grow according to various growth strategies so as to increase the diversity of the population. This strategy can improve the global searchability of the genetic algorithm. The different growth processes of individuals can be regarded as the optimization process of individuals' ability to different weights. Additionally, if the individuals always inherit the ability weights of their parents, they may also fall into the local optimum. Therefore, we apply random multi-weights as the difference in the growth process of individuals. Due to the flexibility of the GGA framework, the solutions of HLSA can be used as mature individuals to participate in subsequent mature stage and genetic stage. Finally, we obtain our proposed GGA-HLSA-RW (GHW). The process of GHW has been shown in Fig. 5 and its algorithm is presented in Algorithm 4. The loops in Algorithm 3 and Algorithm 4 can be manipulated with matrices operations so as to utilize GPU for acceleration.

Algorithm 4: GGA-HLSA-RW (GHW)

Input : Tasks and VMs V_i and their parameters C_{ijk}, server nodes P_j, the limited capacities L_{jk} and the optimization problems ω
Output: Pareto solution set K

Set number of individuals in each generation as N_p, the number of generations as N_g
Generate initial individuals $\left\langle \kappa_1^{(0)}, \kappa_2^{(0)}, \ldots, \kappa_{N_p}^{(0)}\right\rangle$ with random genes and obtain initial Pareto solution set K
for l *in range*(N_g) **do**
  Generate a random weight vector $\boldsymbol{w}_q^{(l)}$ for each individual in the lth generation
  Obtain the problem after dimensionality reduction of each individual in the lth generation as Algorithm 1
  Using HLSA as Algorithm 2 (instantiated by Algorithm 3 in paper) to solve the problem after dimensionality reduction, obtain mature individuals as $\left\langle \bar{\kappa}_1^{(l)}, \bar{\kappa}_2^{(l)}, \ldots, \bar{\kappa}_{N_p}^{(l)}\right\rangle$, and update the solution set K
  Select and Pair the better mature individuals with specific sort strategies such as non-dominated sorting [55,56], congestion degree sorting [56] and ordering of subproblems corresponding to reference vectors [62]
  Generate the $l+1$th infancy individuals $\left\langle \kappa_1^{(l+1)}, \kappa_2^{(l+1)}, \ldots, \kappa_{N_p}^{(l+1)}\right\rangle$ with specific strategies such as elitist strategy
  Update the Pareto solution set K

4.4. Instantiation of GHW: GHW-NSGA II and GHW-MOEA/D

GGA-HLSA-RW (GHW) can be considered as a family of algorithms, which can incorporate various existing genetic optimization strategies to produce specific algorithm variants. This means that the strategies such as NSGA [55], NSGA II [13,56], NSGA III [31], MOEA/D [62], et al. can be applied in GHW to obtain new, well-performing algorithms. Specifically, this paper applies the NSGA II strategy [13] in GHW to obtain the GGA-HLSA-RW-NSGA II (GHW-NSGA II) algorithm and also applies the MOEA/D strategy [62] to obtain GGA-HLSA-RW-MOEA/D (GHW-MOEA/D) algorithm . Referring to the existing terms of genetic algorithms [56,72], the base components of GHW are defined as:

(1) **Gene and Individual (Chromosome)**: we regard the allocated node index of each VM (or task) as a gene, where the ith gene can be written 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 constructs an individual (also chromosome) corresponding to a solution of the optimization problem. Obviously, the vector I and the set of κ are interchangeable.
(2) **Individual selector**: GHW-NSGA II sorts the individuals in the mature stage by non-dominated ordering and congestion ordering, then selects better part of individuals to participate in pairing and crossover. GHW-MOEA/D initialize a group of vectors as reference vectors, selects part of individuals with the best solution under each reference vector and randomly selects two individuals of each reference vector to participate in pairing and crossover.
(3) **Crossover**: It can be assumed two individuals as $I_\alpha = \langle \lambda_{\alpha 1}, \lambda_{\alpha 2}, \ldots, \lambda_{\alpha n}\rangle$ and $I_\beta = \langle \lambda_{\beta 1}, \lambda_{\beta 2}, \ldots, \lambda_{\beta n}\rangle$. Their crossover is defined as separately extracting a part of genes from them to gain a new vector as the children individual, such as $\langle \lambda_{\alpha 1}, \lambda_{\beta 2}, \lambda_{\beta 3}, \ldots, \lambda_{\alpha n}\rangle$. In this paper, we randomly select the number of genes of I_β participating in crossover according to the uniform distribution, and then randomly select the corresponding number of genes from I_β with the same probability to replace the genes at the corresponding position of I_α.
(4) **Mutation**: Mutation is defined as that replacing some elements of an individual $I_\alpha = \langle \lambda_{\alpha 1}, \lambda_{\alpha 2}, \ldots, \lambda_{\alpha n}\rangle$ by randomly generated genes.
(5) **Population regeneration mechanism**: Both GHW-NSGA II and GHW-MOEA/D apply elitist strategy [56] to combine the parent individuals with their children individuals to jointly compete to produce the next generation.

Unlike MOEA/D whose individual respectively corresponds to a fixed reference line (or a vector), GHW-MOEA/D is only to select the optimal part of mature individuals corresponding to each preset reference line in the individual selector, and randomly select two of them for crossover to generate a new child. This means the individuals in GHW-MOEA/D will no longer be constrained by given fixed reference lines.

In order to intuitively demonstrate the process of GHW family, we take a visualized example of GHW-NSGA II for problem $\min \omega^{(2)}$ with two-dimensional resources (CPU and RAM for the sake of observation). Then, we draw its solution process in Fig. 6. In the example of Fig. 6, $n = 10$, $m = 4$, $N_p = 10$, $G_{step} = 10$ and the parameters u_{ijk} of VMs and the genes of individuals are randomly initialized.

As shown in Fig. 6 which is corresponding to each stage in Fig. 5:

(1) **Infancy Stage**: At the 1st generation, the initialized individuals are input as the infancy individuals whose solutions are as Fig. 6(a);
(2) **Growth Stage**: The infancy individuals are input in the growth stage and are improved by HLSA growth to generate mature individuals as Fig. 6(b). The vectors directed from circles to diamonds represent the growth process with HLSA;
(3) **Mature Stage**: After growth stage, mature individuals are screened by non-dominated sorting and congestion degree sorting to obtain some selected individuals as Fig. 6(c) for the subsequent crossover and mutation;
(4) **Genetic Stage**: The selected individuals participate in the crossover and mutation to generate children individuals as Fig. 6(d). Then, children individuals and mature individuals participate in screening together to generate the next infancy individuals as 6(f). At the same time, the Pareto solutions set is updated as Fig. 6(e);
(5) Repeat Infancy Stage → Genetic Stage.

4.5. Analysis of computational complexity of GHW

The computational complexity (denoted as ζ) of GHW mainly consists of two parts: the computational complexity of HLSA ($\zeta^{(H)}$) and that of genetic algorithm ($\zeta^{(G)}$).

The computational complexity of HLSA for one step search of one individual in one generation can be deduced as $\zeta^{(h)} = \mathrm{O}(mn \log n)$. Thus, for N_p individuals and N_g generations, the computational complexity of HLSA is $\zeta^{(H)} = \mathrm{O}\left(G_{step} \cdot N_p \cdot N_g \cdot mn \log n\right)$ where G_{step} is the max number of steps for HLSA search of each individual in each generation. Following, we demonstrate the proof of $\zeta^{(H)}$.

Proof. The computational complexity $\zeta^{(h)}$ corresponds to finding two nodes to execute Algorithm 3. The computational complexity of Algorithm 3 is $card\left(\psi_{j_1} \cup \psi_{j_2}\right) \log card\left(\psi_{j_1} \cup \psi_{j_2}\right)$ which mainly derives from the complexity of sorting. Thus the total computational complexity of enumerating the combinations of any two nodes is $\sum_{j=1}^{m}\sum_{l=j+1}^{m} card\left(\psi_j \cup \psi_l\right) \log card\left(\psi_j \cup \psi_l\right)$. The second derivative of $xlogx$ is $(xlogx)'' = \frac{1}{x} > 0$ and $\sum_{j=1}^{m} card\left(\psi_j\right) = n$, thus substitution into Jensen inequality can obtain:

$$\zeta^{(h)} \ge \mathrm{O}\left(\frac{m(m-1)}{2}\left(\frac{n}{m}\log\frac{n}{m}\right)\right) = \mathrm{O}(nm\log n) \tag{19}$$

In addition, $x_1 \log x_1 + x_2 \log x_2 < (x_1 + x_2)\log(x_1 + x_2)$ assuming $x_1, x_2 > 0$. Therefore:

$$\zeta^{(h)} \le \mathrm{O}((n(m-1)\log n(m-1))) = \mathrm{O}(nm\log n) \tag{20}$$

Combining the above equations, $\zeta^{(h)} = \mathrm{O}(nm\log n)$. Thus, $\zeta^{(H)} = \mathrm{O}\left(G_{step} \cdot N_p \cdot N_g \cdot mn\log n\right)$.

For the computational complexity of the genetic algorithm, if using NSGA II strategies to select mature individuals and to generate the new generations, the computational complexity is $\zeta^{(G)} = \mathrm{O}\left(N_p^3 \cdot N_g \cdot d\right)$ [56]. Then, the computational complexity of GHW-NSGA II can be obtained as:

$$\zeta = \zeta^{(H)} + \zeta^{(G)} = \mathrm{O}\left(N_p \cdot N_g \cdot \left(G_{step} \cdot mn\log n + N_p^2 \cdot d\right)\right). \tag{21}$$

GGA-HLSA is essentially a multi-route search algorithm by adding a search route (HLSA) to the genetic algorithm. The local convergence points of various routes are usually different, so multiple routes can help each other to jump out of the local convergence points of other routes so as to improve the optimality of solutions. Generally, the relationship between multiple performances (such as optimality and computational complexity) of the algorithms are varying with the change of parameters, which may not have an explicit expression. However, from the perspective of qualitative analysis based on information theory, GHW receives more effective information from GA-based route and HLSA-based route. This helps GHW spend less time finding

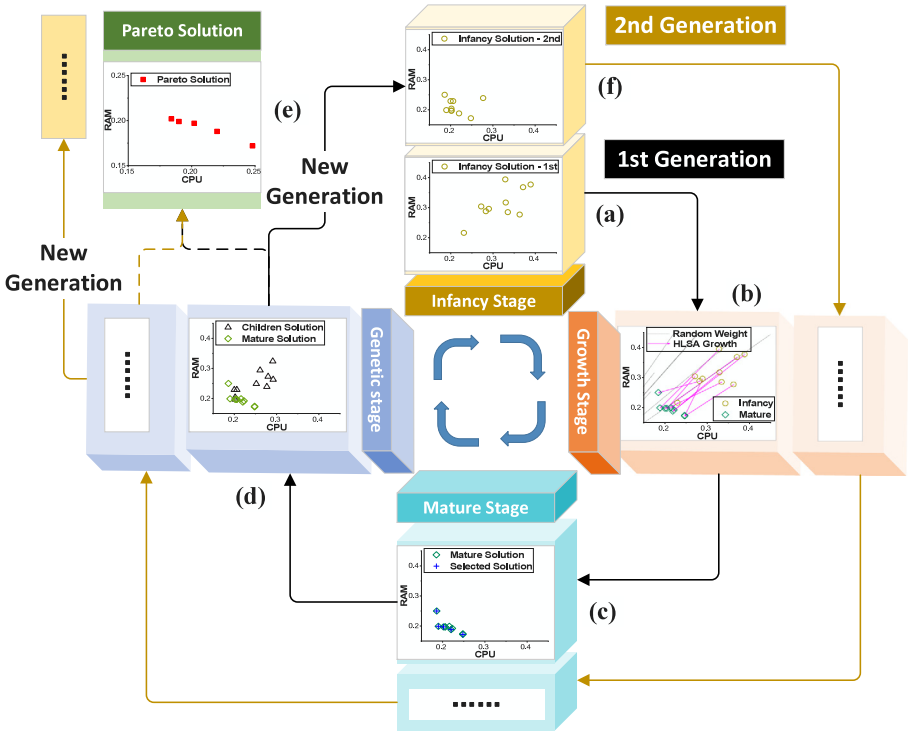


Fig. 6. The visualized example of GHW-NSGA II with actual results in each stage.

Table 4
Different Strategies of GGA.

Category	Strategy	Description
Growth Strategies	HLSA growth	Using HLSA as growth route of individuals in GGA
	Random growth	Using randomization as growth route of individuals
	Non growth	Direct genetic without growth
Dimension Reduction	RW	Random multi-weights for each individual
	EW	Enumeration weights for each individual
	RD	Random dimensions for each individual
	RRD	Round-robin dimensions for each individual
	TRD	Taboo round-robin dimensions for each individual
	RWS	Random dimensionality reduction strategies for each individual

better solutions than the existing algorithms such as NSGA II and MOEA/D. Thus, the algorithms of the GHW family theoretically have certain advantages in terms of convergence performance.

5. Performance evaluation

5.1. Experiments setting

For the sake of the comprehensive evaluations to the proposed algorithms, this section carries out four groups of experiments from various aspects including:

(1) EX_1: comparison of growth strategies for GGA;
(2) EX_2: comparison of dimensionality reduction strategies for GGA-HLSA;
(3) EX_3: evaluation of practicability on a real public trace-driven dataset (AzureTraceforPacking2020 [17]);
(4) EX_4: verification on the advantages of proposed algorithms in convergence and optimality by comparing with state-of-the-art.

All these experiments are based on the control variable method. Because this paper mainly focuses on discussing the influence of adding a directed growth in the GGA instead of that of the type of algorithms for the growth stage, we select modified LSPT-based HLSA as the growth route of GGA. Fixing the algorithm process as GGA and regarding the growth strategy as the variable, EX_1 presents the performance of adding HLSA growth, random growth (non-directed growth) and no growth respectively as shown in Table 4. Except for random multi-weights, other strategies for dimensionality reduction include Enumeration Weights (EW), Random Dimensions (RD), Round-Robin Dimensions (RRD), Taboo Round-robin Dimensions (TRD) and Random dimensionality and Weights Strategies (RWS) as shown in Table 4. For example, Random Dimensions (RD) means randomly choosing one dimension to generate the single-dimensional optimization problem of each individual in each generation and using HLSA to solve the problem of the chosen dimension. Other compared strategies for dimensionality reduction are similar to RD and RW. EX_2 fixes the algorithm process as GGA-HLSA and regards the dimensionality reduction strategy as the variable. To validate the applicability of our proposed algorithms in realistic, EX_3 is a group of trace-driven experiments based on AzureTraceforPacking2020 [17]. EX_4 presents some indicators over time to validate the superiority of our proposed algorithms of GHW family compared with the state-of-the-art.

With the exception of EX_3, which is performed on the trace-based dataset, other groups of experiments are executed on the random simulation dataset, which is conducive to generating adequate data for comprehensive verification. The descriptions and operations of the datasets used in experiments are as follows.

(1) Random Simulation Dataset: is generated randomly by a Python-based simulation environment. In the simulation environment, we set up the server nodes to be heterogeneous and the parameters of VMs obey a uniform distribution.
(2) AzureTraceforPacking2020 [17]: represents part of the workload on Microsoft's Azure Compute. AzureTraceforPacking2020 provides the required CUP, RAM, SSD (Solid-State

Table 5
Setup of experiments.

Groups	Comparison	Dataset	Objectives	Scenarios (servers, VMs)
EX_1	Growth Strategies	Simulation Dataset	$\min \omega^{(2)}$ $\min \omega^{(4)}$	(8, 80) (16, 20-200), (20, 20-200)
EX_2	Dimensionality Reduction	Simulation Dataset	$\min \omega^{(2)}$ $\min \omega^{(4)}$	(20, 200) (16, 20-200), (20, 20-200)
EX_3	Generations	AzureTracefor Packing2020	$\min \omega^{(2)}$ $\min \omega^{(4)}$	(400, 1000), (700, 2000) (400, 1000), (700, 2000)
EX_4	SOTA: NSGA II MOEA/D	Simulation Dataset	$\min \omega^{(2)}$	Large Scale Small Scale

Drive) and NIC (Network Bandwidth) allocation for 487 types of VMs on 35 types of machines. Since AzureTraceforPacking2020 provides the utilization of MDRs by VMs on heterogeneous machines, it is very suitable to verify the performance of the proposed methods. If a certain dimension of a VM occupies too much utilization on all types of machines, it will directly cause this VM to occupy one machine alone, which has no impact on the optimization of other VMs and machines. Thus we choose to ignore some types of VMs with large utilization. In the experiments of this paper, 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.

EX_1 to EX_3 are executed for two problems, i.e. $\min \omega^{(2)}$ (minimizing the maximum utilization rate of resources for each dimension under all nodes) and $\min \omega^{(4)}$ (minimizing energy consumption for total cloud system), while EX_4 is performed only for the $\min \omega^{(2)}$ problem. For the sake of presenting the Pareto solution in the same figure simultaneously, we choose three dimensions of resources, that is CPU, RAM and DS, to execute the experiments. Since the results from various setups of clouds allowed us to draw the same conclusions, we only present results for parts of them corresponding to specific scenarios. The experimental setup is shown in Table 5.

We use the Mxnet framework to enable the program to run in GPU. Then, the experiments are launched on a GPU desktop computer with configurations as:

- CPU: Intel(R) Core(TM) i5-8400 CPU @ 2.8 GHZ;
- SSD: KINGSTON SA400S37 240 GB;
- GPU: NVIDIA GeForce GTX 1060 6 GB;
- Program version: Python 3.6 + mxnet-cu90 1.5.0.

5.2. EX_1: Comparison of the growth strategies for GGA

EX_1 is carried out to evaluate the performance of different growth strategies in the genetic algorithm including non-growth, random growth and HLSA growth shown as Table 4. In EX_1, we set $N_g = 100$, $N_p = 50$, mutation rate is 0.2, $G_{step} = 10$ as well as the individual selector and regeneration mechanism of Fig. 5 are conducted with random screening with equal probability rather than using survival of the fittest.

5.2.1. Minimizing the maximum utilization of resources

For the problem of $\min \omega^{(2)}$ (i.e., minimizing the maximum utilization rate of resources for each dimension under all nodes), we present the experiment results under the scenarios with 8 nodes and 80 VMs. And the utilization of VMs obeys the uniform distribution:

$$u_{ijk} \sim \mathrm{U}(0, 12)\,\% \tag{22}$$

where $\mathrm{U}(0, 12)$ means the uniform distribution in $[0, 12]$ generated by calling the function of mxnet as mxnet.nd.random.randint(low = 0, high = 121, shape = (n, d, m)/1000.

For sake of observation, Fig. 7 plots the Pareto solutions of two-dimensional resources (the boundary of projection on each coordinate plane of 3D Pareto solution) for 8 server nodes with 80 VMs.

In Fig. 7, the Pareto set of HLSA growth strategy outperforms random growth and non-growth. That means for each solution of non-growth and random growth strategies, there is still at least one solution of HLSA growth strategy dominating it. The comparison between random-growth and non-growth illustrates that the GGA can optimize the result of the non-growth genetic algorithm although only using randomization as the growth route. The comparison between HLSA-growth and random growth illustrates that directional growth (i.e., using the HLSA as the growth route of the GA) can obtain better Pareto solutions than that of the random-growth route. Further, the HLSA-growth can eliminate the instability of the random growth strategy.

5.2.2. Minimizing energy consumption

Next, we carry out the experiments in the scenarios of $\min \omega^{(4)}$ (i.e., minimizing energy consumption for the total cloud system) with 16 and 20 server nodes respectively and set the numbers of VMs from 20 to 200 where the capacities requested of each VMs are $C_{ijk} \sim \mathrm{U}(75, 150)$. In the simulated experiments, we randomly generate the coefficients of Eq. (11) as integers according to uniform distribution that:

$$\begin{cases} a_{jk} \sim \mathrm{U}(1, 10),\ b_{jk} \sim \mathrm{U}(0, 100), \\ c_{jk} \sim \mathrm{U}(100, 200),\ d_{jk} \sim \mathrm{U}(500, 1000). \end{cases} \tag{23}$$

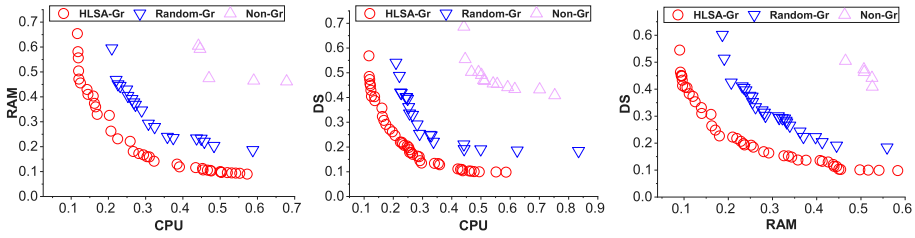
Then, we plot the results of each growth strategy in Fig. 8.

Fig. 8(a) plots the minimum energy consumption from three strategies for 16 server nodes and Fig. 8(b) plots that for 20 server nodes of the 100th generations. From Fig. 8, the HLSA-Growth strategy achieves the lowest energy consumptions compared with random-growth and non-growth for all the sets of (n, m). In addition, random-growth achieves lower energy consumption than non-growth. The sorting of the performance of these three strategies in Fig. 8(b) demonstrates GGA outperforms non-growable one and the directional growth outperforms the random growth. For all experimental combinations, HLSA reduces energy consumption on average by 95.21% and 333.7% than random-growth and non-growth respectively.

Overall, EX_1 verifies that adding a growth stage in the process of the genetic algorithm solving MDRSPs can significantly improve solutions. This may be because GGA actually applies a multi-route, which usually has better local optimal solutions than the single-route search.

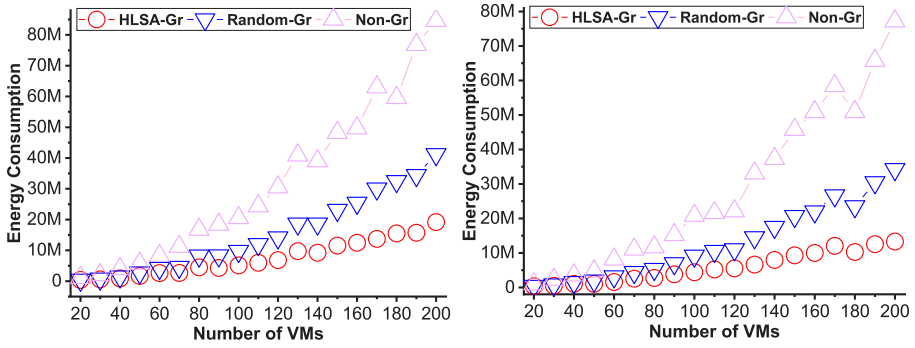
5.3. EX_2: Comparison of dimensionality reduction strategies for GGA-HLSA

To evaluate the performance of different dimensionality reduction strategies, we carry out experiments respectively using:



(a) CPU and RAM (b) CPU and DS (c) RAM and DS

Fig. 7. 2D Pareto solution of minimizing the maximum utilization of each dimensional resources under non-growth, random growth and HLSA growth strategies of GGA with random crossover and regeneration for 8 server nodes and 80 VMs.



(a) 16 server nodes (b) 20 server nodes

Fig. 8. Energy consumption under non-growth, random growth and HLSA growth strategies of GGA with random crossover and regeneration.

RW, EW, RD, RRD, TRD and RWS as shown in Table 4. Same as EX_1, we set $N_g = 100$, $N_p = 50$, mutation rate is 0.2, $G_{step} = 10$, as well as the individual selector and regeneration mechanism, are also conducted with random screening with equal probability.

5.3.1. Minimizing the maximum utilization of resources

For the problem of $\min \omega^{(2)}$, we only plot the experiment results under the scenarios with 20 server nodes and 200 VMs, as similar conclusions can also be obtained under other combinations of (n, m). The parameters of VMs also obey Eq. (22). After 100 generations, the Pareto solution sets of each dimensionality reduction strategy are plotted in Fig. 9. In each subfigure of Fig. 9, the Pareto solutions of RWS, EW and RW are obviously better than that of RD, RRD and TRD, which shows that the strategy only by switching a single-dimension is far from finding a better solution hence weighted solutions are necessary. For RWS, EW and RW with relatively close Pareto solutions, the solutions of EW and RW are better than that of RWS, which demonstrates RWS has more uncertainty resulting worse solution set. The Pareto boundaries of EW and RW almost coincide, which demonstrates that RW is sufficient to search the Pareto solutions set. The computational complexities of EW are too large because each generation in the genetic algorithm needs to enumerate as many weights as possible. However, the Pareto solutions corresponding to different weights may be the same, so EW costs extra computation, but may miss some solutions as it cannot really enumerate all weights.

To more quantitatively evaluate the performance of EW and RW, we combine the three-dimensional Pareto solution sets of EW and RW for 20 server nodes and 200 VMs into a new Pareto solution set to calculate the proportion of EW and RW in the new solution set. The new Pareto solutions set has 621 solutions in total where 474 solutions originate from RW and 149 solutions from EW. That means RW solutions account for 76.08% and EW for 23.92% in the combined Pareto solutions set. These results illustrate that the RW strategy has a higher probability of obtaining a better Pareto solution set than EW. It means the usage of RW can not only reduce the computational complexity but also improve the Pareto solution set.

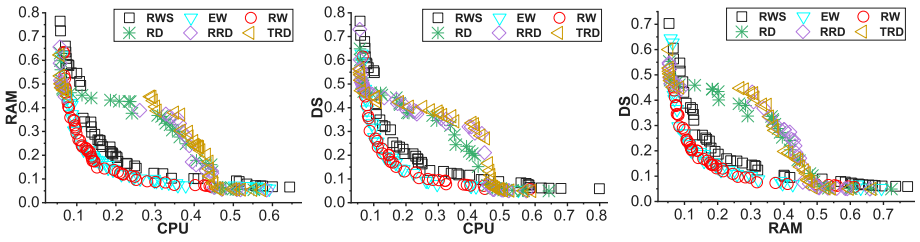
5.3.2. Minimizing energy consumption

To further verify the performance of different dimensionality reduction strategies, we carry out the experiments in the scenarios of minimizing energy consumption $(\min \omega^{(4)})$ with 16 and 20 server nodes respectively. We set the numbers of VMs from 20 to 200 where the capacities requested of each VMs are $C_{ijk} \sim \mathrm{U}(75, 150)$. Then, the minimum energy consumptions of each dimensionality reduction strategy are plotted in Fig. 10.

In Fig. 10, EW, RW and RWS obtain far lower energy consumptions than RD, RRD and TRD, which shows the strategies on weights are still better than that on a single resource dimension in the MDRSP of minimizing energy consumption. RW achieves the best performance among all the strategies, followed by RWS and EW. Although EW has better Pareto solutions than RWS in $\min \omega^{(2)}$, EW has larger energy consumption than RWS in $\min \omega^{(4)}$. This is because the objective of $\min \omega^{(4)}$ has nonlinear terms as Eq. (11), which makes balancing the utilization of dimensional resources not equal to the minimum energy consumption. This phenomenon shows again EW cannot really enumerate all weight combinations. Overall, RW reduces the energy consumption on average by 29.63%, 35.30%, 122.8%, 129.3%, and 129.2% respectively than RWS, EW, RD, RRD and TRD.

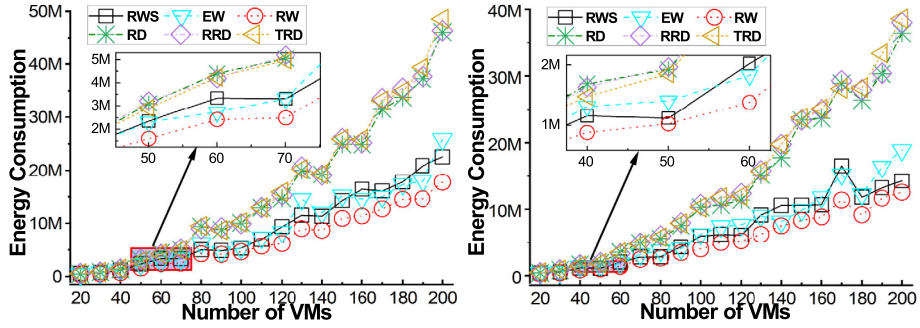
The results of EX_2 can conclude that using random weight in GGA-HLSA can gain better solutions than other dimensionality reduction strategies through the same generations for MDRSPs.

Firstly, when using dimension reduction strategies based on resource dimensions, including random dimensions, round-robin dimensions, and taboo round-robin dimensions, GGA-HLSA evolves each individual in only one dimension at a time. In fact, RD, RRD and TRD are unable to balance the utilization rates of resources in multiple dimensions. Thus, they cannot obtain



(a) CPU and RAM (b) CPU and DS (c) RAM and DS

Fig. 9. 2D Pareto solution using different dimensionality reduction strategies of GGA-HLSA with random crossover and regeneration for 20 server nodes and 200 VMs.



(a) 16 server nodes (b) 20 server nodes

Fig. 10. Energy consumption using different dimensionality reduction strategies of GGA-HLSA with random crossover and regeneration.

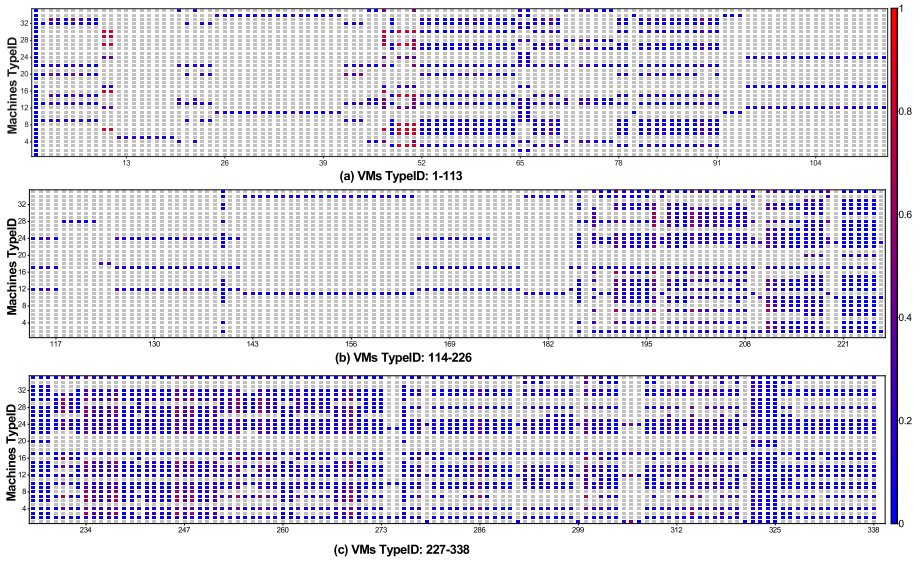


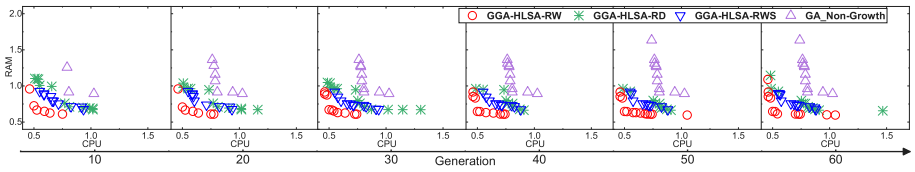
Fig. 11. Heat-map of the CPU utilization required by our selected 338 types of VMs on 35 types of machines, where the gray represents the VM of the specified type cannot run on the corresponding machine.

good solutions with the same optimization level as RW when multiple dimensions all have a certain weight proportion. Consequently, in Fig. 9, the Pareto solutions of RD, RRD, and TRD in the middle are far worse than that of RW. The same reason works in the experiments of Fig. 10. That is, the optimization of resources in a single-dimension often cannot reduce the total energy consumption of the cloud system.

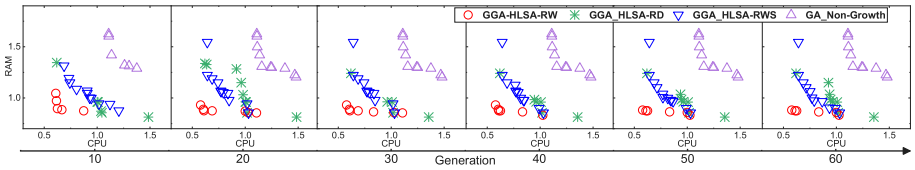
Then, RW, EW and RWS in GGA-HLSA can obtain a close solution set, but that of EW and RWS are worse than RW. This may be because: in RWS, poor strategies have shared the running time; and in EW, some close weights actually get repeated solutions. Both RWS and EW make GGA-HLSA not fully utilized to obtain more information in the process of searching for solution space. Similarly, they failed to obtain the solution with lower energy consumption than RW.

5.4. EX_3: Evaluation of practicability on azure trace

To further evaluate the practicality and the convergence of our proposed GHW in solving MDRSP, we carry out a group of trace-driven experiments EX_3 based on AzureTraceforPacking2020 [17]. As mentioned above, we select 338 types of VMs with $\min_{j=1}^{m} u_{ijk} \leq 30\%$ for $\forall i, k$, and renumber these types of VMs. Then we can draw a heat-map of the CPU utilization required by our selected 338 types of VMs on 35 types of machines as shown



(a) 1000 VMs and 400 machines



(b) 2000 VMs and 700 machines

Fig. 12. Pipeline of Pareto solution sets within 60 Generations for $\min \omega^{(2)}$ of Azure Trace.

in Fig. 11. The heat maps of their RAM and SSD are similar to that of the CPU.

In EX_3, we set each generation to have 20 individuals to observe the pipeline of Pareto solutions in $\min \omega^{(2)}$ and that of energy consumptions in $\min \omega^{(4)}$ within 60 generations. We also set the mutation rate is 0.2, $G_{step} = 10$, as well as the individual selector strategies and regeneration mechanism of Fig. 5 are based on NSGA II. We conducted extensive experiments under different numbers of VMs and machines. Since each experiment can lead to the same qualitative conclusion, we only present the results under two sets of parameters (400 machines, 1000 VMs) and (700 machines, 2000 VMs).

Using GGA-HLSA-RD-NSGA II (GHD-NSGA II), GGA-HLSA-RWS-NSGA II (GHWS-NSGA II) and NSGA II as baselines, the pipelines of Pareto solution sets of $\min \omega^{(2)}$ on the plane of (CPU, RAM) are plotted in Fig. 12. Fig. 12(a) is for 1000 VMs and 400 machines, as well as Fig. 12(b) for 2000 VMs and 700 machines. Because most types of VMs in AzureTraceforPacking2020 [17] cannot be allocated on many types of machines, the Pareto solution sets of the trace-driven experiments are not as regular as that on the simulation dataset (EX_1 and EX_2) which approximately form the continuous curves. However, both in Figs. 12(a) and 12(b), GHW-NSGA II not only still obtains better Pareto solution set at the 60th generation, but also achieves convergence faster in $\min \omega^{(2)}$ than baselines.

To continually evaluate the performance of GHW-NSGA II in other objectives, we plot the pipeline of energy consumption within 60 generations of the problem $\min \omega^{(4)}$ in Fig. 13. For this problem, we also assume energy consumption satisfies Eq. (24) by replacing C (capacity) with u (utilization) and the coefficients also satisfy Eq. (23) considering AzureTraceforPacking2020 [17] did not provide energy consumption data of resources.

$$E = \sum_{j=1}^{m}\sum_{k=1}^{d}\left(a_{jk}\left(\sum_{i=1}^{n} x_{ij}u_{ijk}\right)^2 + b_{jk}\left(\sum_{i=1}^{n} x_{ij}u_{ijk}\right)\right) + \sum_{j=1}^{m}\sum_{k=1}^{d}\left(c_{jk}\right) + \sum_{j=1}^{m}\left(\max_{i=1}^{n}\left(x_{ij}\right)\sum_{k=1}^{d} d_{jk}\right) \tag{24}$$

As shown in Fig. 13, GHW-NSGA II obtains lower solutions of energy consumption than the baselines since the first generation. In the iteration of 60 generations, the curve of GHW-NSGA II is still the lowest compared with the baselines. At the end of the 60th generation for the two scenarios, the energy consumptions of GHW-NSGA II are (844633, 1545821), which are respectively reduced by (1.14, 1.68)%, (1.03, 1.11)% and (1.71, 2.43)% respectively compared with the baselines GHD-NSGA II, GHWS-NSGA II and NSGA II.

EX_3 on AzureTraceforPacking2020 verifies our proposed GGA-HLSA-RW not only has fast convergence but also applies to the MDRSP in realistic cloud computing.

5.5. EX_4: Comparison with the state-of-the-art

To further evaluate the advantages of our proposed algorithms, we execute a group of experiments EX_4 to compare the proposed GHW family with the state-of-the-art in the terms of convergence and optimality. The state-of-the-art participating in comparison are NSGA II and MOEA/D which are two well-performed and frequent baselines in MOPs. For the algorithms of the GHW family, we verify three algorithms including GHW-RCE (GGA-HLSA-RW with Random Crossover and rEgeneration mechanism), GHW-NSGA II and GHW-MOEA/D.

The previous subsections have fully verified the practicability of our proposed algorithms in both the simulation dataset and public trace-driven dataset for both the problems of $\min \omega^{(2)}$ and $\min \omega^{(4)}$. Therefore, we only evaluate their performance in the simulation dataset for the problem of $\min \omega^{(2)}$ (minimizing the maximum utilization rate of resources for each dimension under all nodes) in EX_4, which does not lose generality.

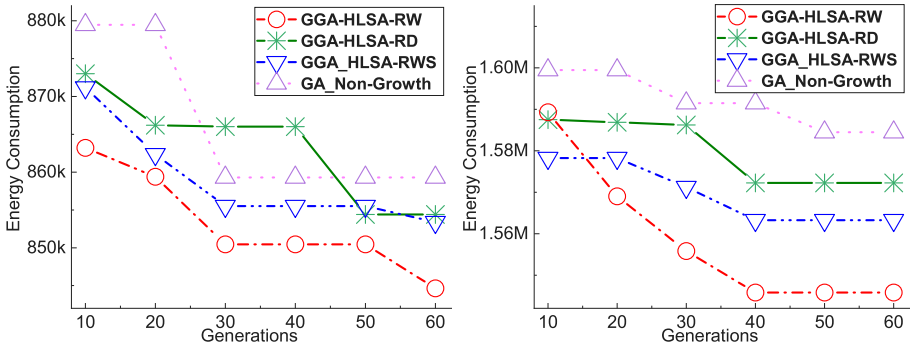
In addition, we select different parameters from those in the EX_1, EX_2 and EX_3 to increase the diversity of experimental results. The distribution of the utilization is:

$$u_{ijk} \sim \mathrm{U}\,(5,\,12)\,\%, \tag{25}$$

generated by mxnet.nd.random.randint(low = 50, high = 121, shape
= (n, d, m)/1000.

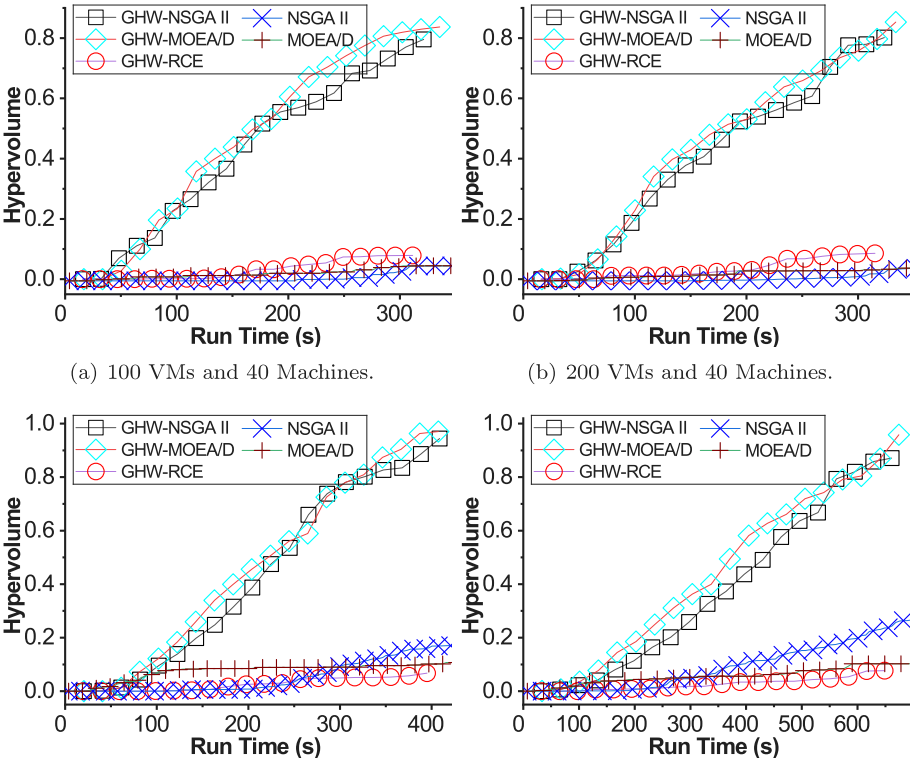
For the sake of quantitative analysis of convergence, we apply HVs-over-time [57,58] as the evaluation indicator. Considering to observe the HyperVolume (HV) of each algorithm into the range [0, 1], the experiments call the function **pymoo.indicators.hv**. **Hypervolume** [73] with the following settings:
(1) ref_points = (1, 1, 1),
(2) norm_ref_point = False,
(3) zero_to_one = True,
(4) ideal = $\left(\min_j U_{j1}, \min_j U_{j2}, \min_j U_{j3}\right)$,
(5) nadir = $\left(\max_j U_{j1}, \max_j U_{j2}, \max_j U_{j3}\right)$.
According to the calculation time of each algorithm, we convert the generation number into the corresponding running time to obtain the HVs-over-time of each algorithm.

We present four groups of experiments respectively as $(n = 100, m = 40)$, $(n = 200, m = 40)$, $(n = 500, m = 100)$ and $(n = 1000, m = 100)$, which are sufficient to illustrate the convergence of the algorithms. Since GHW adds a growth stage



(a) 1000 VMs and 400 Machines (b) 2000 VMs and 700 Machines

Fig. 13. Pipeline of Energy Consumption within 10 Generations for $\min \omega^{(4)}$ of Azure Trace.



(a) 100 VMs and 40 Machines. (b) 200 VMs and 40 Machines.

(c) 500 VMs and 100 Machines. (d) 1000 VMs and 200 Machines.

Fig. 14. HVs-over-time of proposed GHW family, NSGA II and MOEA/D for the problem $\min \omega^{(2)}$ in simulation dataset.

in each generation, the time consumption of each generation will be longer than that of NSGA II and MOEA/D. In order to balance the running time of algorithms so as to facilitate comparison at the approximate time point, we set the $N_g = 20$, $N_p = 100$ for the algorithms of GHW family, as well as $N_g = 400$, $N_p = 500$ for NSGA II and MOEA/D. The mutation rate is set as 0.2 and $G_{step} = 10$. Then, we plot the HVs-over-time of each algorithm in Fig. 14.

Comparing GHW-RCE with NSGA II and MOEA/D in Figs. 14(a) and 14(b), GHW-RCE reaches higher HVs than NSGA II and MOEA/D in each same time point. Concretely in Fig. 14(a), the HV of GHW-RCE at the 20th generation (corresponding to 311.41 s) is 0.079. Correspondingly, NSGA II and MOEA/D at the 90th generation obtain lower HVs both as 0.056. In Figs. 14(c) and 14(d) with larger scale of VMs and server nodes, NSGA II and MOEA/D obtain higher HVs than GHW-RCE. Changes in comparison results from Figs. 14(a) to 14(d) show that the performance of GHW-RCE degrades faster than that of NSGA II and MOEA/D with the increasing number of VMs and nodes. This is because random crossover and population regeneration counterproductively make GHW-RCE fail to inherit the excellent solutions found before. In fact, GHW plays the role of optimizing solutions, as well as random crossover and regeneration play the role of degrading solutions. Even so, GHW-RCE still maintains advantages to a certain extent in Fig. 14(a) and Fig. 14(b), which reversely demonstrates the powerful advantages of GHW.

Comparing GHW-NSGA II and GHW-MOEA/D with NSGA II and MOEA/D in Fig. 14, the results obviously and solidly demonstrate the superiorities of GHW family. In Fig. 14, the HVs of GHW-NSGA II and GHW-MOEA/D are far higher than that of compared algorithms. Concretely in Fig. 14(a), GHW-NSGA II and GHW-MOEA/D only take 48.20 s and 67.16 s to obtain the HVs as 0.070 and 0.099 respectively, which are higher than that of NSGA II and MOEA/D got in 320 s. This illustrates GHW-NSGA II and GHW-MOEA/D have much faster convergence rates. At 11th generation, GHW-NSGA II (at 176.73 s) and GHW-MOEA/D (at 184.67 s)

Table 6
The HVs and corresponding time compared the algorithms of GHW family with state-of-the-art.

Algorithms	Final HV					When one achieves HV ≥ 0.5		
	Gen.	Time	HV	ϵ_1	ϵ_2	Gen.	Time	HV
Fig. 14(a): $(n, m) = (100, 40)$								
NSGA II	90	335.3	0.056	48.2	67.1	50	186.3	0.000
MOEA/D	90	349.4	0.056	48.2	67.1	45	174.7	0.021
GHW-RCE	20	311.4	0.079	64.2	67.1	11	171.2	0.033
GHW-NSGA II	20	321.3	**0.796**	–	–	11	176.7	**0.516**
GHW-MOEA/D	20	335.7	**0.837**	–	–	11	184.6	**0.532**
Fig. 14(b): $(n, m) = (200, 40)$								
NSGA II	80	331.2	0.043	64.7	66.7	43	178.0	0.002
MOEA/D	80	308.6	0.041	64.7	66.7	45	173.5	0.022
GHW-RCE	20	315.2	0.086	80.8	83.4	11	173.3	0.025
GHW-NSGA II	20	323.5	**0.802**	–	–	11	177.9	**0.463**
GHW-MOEA/D	20	333.8	**0.853**	–	–	11	183.6	**0.514**
Fig. 14(c): $(n, m) = (500, 100)$								
NSGA II	120	403.8	0.169	143.0	122.2	66	222.1	0.017
MOEA/D	120	349.4	0.119	122.5	101.8	60	226.8	0.089
GHW-RCE	20	395.0	0.068	102.1	81.4	11	217.2	0.027
GHW-NSGA II	20	408.6	**0.945**	–	–	11	224.7	**0.475**
GHW-MOEA/D	20	407.3	**0.970**	–	–	11	224.0	**0.506**
Fig. 14(d): $(n, m) = (1000, 200)$								
NSGA II	200	674.3	0.357	363.6	303.4	120	404.6	0.296
MOEA/D	200	727.9	0.151	231.4	202.2	110	400.3	0.114
GHW-RCE	20	648.6	0.077	165.3	134.8	12	389.2	0.034
GHW-NSGA II	20	661.2	**0.871**	–	–	12	396.7	**0.437**
GHW-MOEA/D	20	674.2	**0.959**	–	–	12	404.5	**0.581**

Note: ϵ_1 (s) and ϵ_2 (s) respectively express the time when GHW-NSGA II and GHW-MOEA/D achieve corresponding HV.

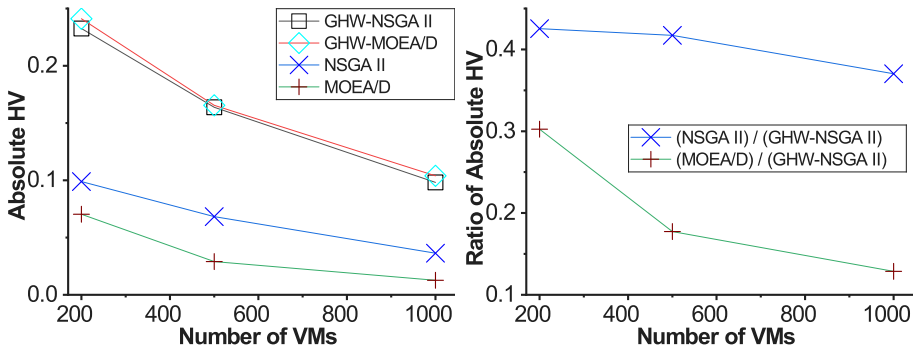
achieve 0.516 and 0.532 HV respectively more than 0.5. And at 20th generation, the HVs of GHW-NSGA II (at 321.33 s) and GHW-MOEA/D (at 335.78 s) achieve 0.796 and 0.837 respectively, which are both ten times more than that of NSGA II and MOEA/D. The other figures in Fig. 14 also have similar phenomena. With the increase in the number of VMs and nodes, GHW-NSGA II and GHW-MOEA/D still maintain obvious advantages steadily.

For quantitative description, we list the values of HVs and the corresponding time of each group of experiments in Table 6. The quantitative comparison of Table 6 shows that the HVs of our proposed algorithms keep higher than compared algorithms. Moreover, our proposed GHW-NSGA II and GHW-MOEA/D reduce more than 200 s to achieve the approximate HV that the comparison algorithms achieve at the end of each figure in Fig. 14. And, our proposed algorithms only spend about 11 generations to achieve the HVs close to 0.5 in each scenario of Fig. 14. With the expansion of the scale, the HVs of all algorithms show an increasing trend continuously. This is because we normalize the solutions to [0, 1] by setting zero_to_one = True when calculating the HV, thus raising the value of the comparison algorithm. This also leads to the narrowing of the gap between our algorithm and the comparison algorithm. Different settings when computing HV will cause different numerical trends, but their qualitative results will not change. Thus, the results of Table 6 can at least intuitively prove our proposed algorithms are significantly better than compared algorithms.

In fact, the advantages of our algorithm are expanding with the increase of the scale of (n, m). To verify it, we compute the absolute HV of the three scales of VMs and nodes $(200, 40)$, $(500, 100)$, $(1000, 200)$ by setting zero_to_one=False. Then, we plot the absolute HVs in Fig. 15(a) and the ratio of absolute HV between compared algorithms and GHW-NSGA II in Fig. 15(b). In Fig. 15, the numbers of VMs are all 5 times of nodes. Under the same proportion between VMs and nodes, large scale has more feasible solutions and smaller scheduling granularity. Thus, the theoretical optimal absolute HVs will increase with scale. However, in Fig. 15(a), the absolute HVs of all algorithms decrease with the increase of scale. This is because the search space of the solution set of the NP-hard problem increases exponentially with the increase of scale, so that the optimality of the algorithms' solution set will decline in the exponential increase of search time. To measure the decline amplitude of different algorithms, Fig. 15(b) computes the ratio of absolute HVs between compared algorithms and GHW-NSGA II under each combination of (n, m). In Fig. 15(b), the ratios are decreasing with the increase of VMs, which means the optimality of compared algorithms decreases more than that of GHW-NSGA II. This may be because the searchability of NSGA II and MOEA/D cannot keep up with the exponential increase of solution space with the increase of VMs and nodes.

Additionally Fig. 16 plots the results with a larger time range of Fig. 14(c) and Fig. 14(d). From Fig. 16, increasing the time range cannot make the HVs of NSGA II and MOEA/D reach the level of GHW-NSGA II and GHW-MOEA/D. This demonstrates our proposed GHW-NSGA II and GHW-MOEA/D have advantages not only in convergence rate but also in optimality.

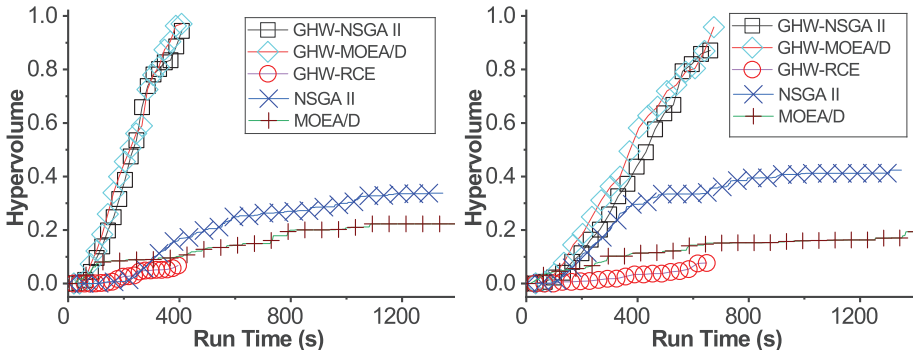
In order to further illustrate the performance of our proposed algorithms in the theoretical optimal solution, we carry out several groups of experiments in small scales. We use an enumerative algorithm (marked as EnA) to obtain the theoretical optimal Pareto solutions further to obtain theoretical optimal HVs as reference. Considering multi groups of experiments have similar conclusion, we only present two groups of experiments respectively under $(n, m) = (10, 3)$ and $(n, m) = (10, 4)$. Then, the HVs-over-time are shown in Fig. 17. In the experiments of Fig. 17, we set $N_p = 100$ for GHW-RCE, GHW-NSGA II and GHW-MOEA/D, as well as set $N_p = 500$ for NSGA II and MOEA/D. The N_g of all the algorithms is 20 and the mutation rate is 0.2. In Fig. 17, GHW-NSGA II and GHW-MOEA/D are the two fastest algorithms to approach or even reach the theoretical optimal HV followed by GHW-RCE, while the state-of-the-art NSGA II and MOEA/D do not reach the theoretical optimum in Fig. 17. This shows that our proposed GHW family of algorithms is more likely to find



(a) Absolute HVs.

(b) Ratio of absolute HVs between compared algorithms and GHW-NSGA II.

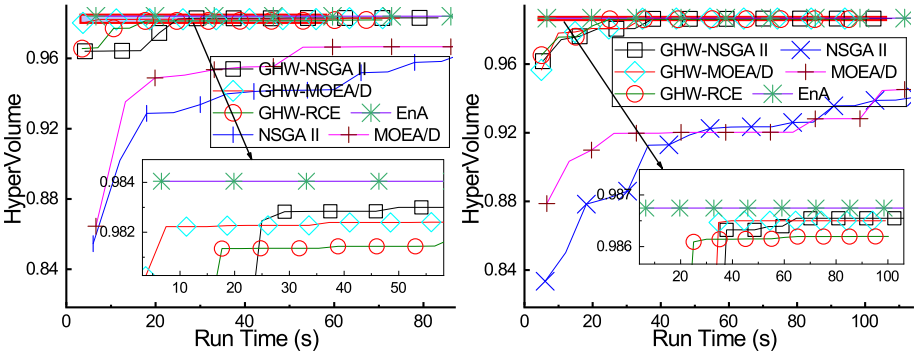
Fig. 15. Absolute HVs and ratio over number of VMs where (200, 40), (500, 100), (1000, 200) and zero_to_one=False.



(a) 500 VMs and 100 Machines.

(b) 1000 VMs and 200 Machines.

Fig. 16. Extended results of Fig. 14 with larger time range.



(a) 10 VMs and 3 Machines.

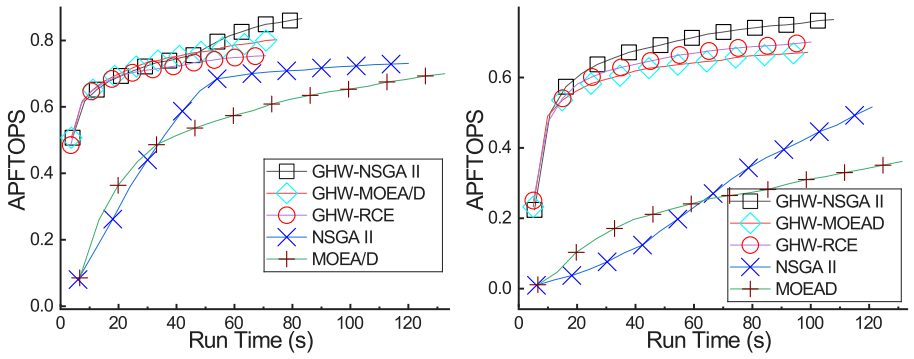
(b) 10 VMs and 4 Machines.

Fig. 17. HVs-over-time of proposed GHW family, NSGA II and MOEA/D for the problem $\min \omega^{(2)}$ with enumerative algorithm as reference in small scale simulation dataset.

the theoretical optimal solution in the small-scale dataset than compared algorithms.

To further comprehensively validate this conclusion, we conduct 100 instances under each combination of parameters $(n, m) = (10, 3)$ and $(n, m) = (10, 4)$ respectively, record the proportion of each algorithm finding the theoretical optimal solution at the corresponding time, and compute the average proportion corresponding to each time as the Average Probability-over-time of each algorithm Finding the Theoretical Optimal Pareto Solutions (denoted as APFTOPS). Then, we plot the results in Fig. 18.

In Fig. 18, the curves of GHW-NSGA II and GHW-MOEA/D are higher than that of the compared algorithms. Concretely, in Fig. 18(a), the APFTOPSs of GHW-RCE (0.754), GHW-NSGA II (0.866) and GHW-MOEA/D (0.802) reach more than 0.75 at about 80 s (20th generation), while that of NSGA II (0.711) and MOEA/D (0.623) are less than 0.72; in Fig. 18(b), that of GHW-RCE (0.701), GHW-NSGA II (0.765) and GHW-MOEA/D (0.671) reach more than 0.65 at about 100 s (20th generation), while that of NSGA II (0.446) and MOEA/D (0.310) are less than 0.45. The statistical significance of Fig. 18 shows that our proposed algorithms have higher probabilities to find the theoretical Pareto solutions,



(a) 10 VMs and 3 Machines. (b) 10 VMs and 4 Machines.

Fig. 18. APFTOPS of the proposed GHW family, NSGA II and MOEA/D for the problem $\min \omega^{(2)}$ with enumerative algorithm as reference in small scale simulation dataset where each combination of (n, m) has 100 instances.

which means our proposed algorithms have better optimality than compared algorithms in solving the problem $\min \omega^{(2)}$.

These results in EX_4 prove that the advantages of GHW are comprehensive in the aspects of faster convergence rate and better optimality. This is consistent with the analysis in Section 4.5 that the combination of two search routes (i.e., HLSA route and genetic route) has more diverse information and can make full use of effective information to find better optimization solutions.

5.6. Summary of experiments

Through the multiple groups of experiments from various sights in this section, we can observe that GHW, as a novel family of genetic algorithms, has significant advantages in addressing the MDRSP that is usually NP-hard problem. Among these experiments:

- EX_1 evaluates the effect of different growth routes on GGA. It demonstrates adding a growth stage can significantly improve the performance of the genetic algorithm in solving MDRSPs. It also demonstrates a directional growth route using HLSA has better solutions than compared random growth and non-growth. The order of algorithms by performance is: GGA-HLSA > GGA-RandomGrowth > GA (GGA-Non Growth), where GGA-HLSA > GGA-RandomGrowth means GGA-HLSA outperforms GGA-RandomGrowth;
- EX_2 evaluates the effect of different dimensionality reduction strategies on GGA-HLSA. It demonstrates using RW (Random Weight) in GGA-HLSA can obtain more comprehensively better solutions than compared dimensionality reduction strategies including EW (Enumeration Weights), RD (Random Dimensions), RRD (Round-Robin Dimensions), TRD (Taboo Round-robin Dimensions) and RWS (Random dimensionality and Weights Strategies). The order by performance is: GGA-HLSA-RW > (GGA-HLSA-EW, GGA-HLSA-RWS) > (GGA-HLSA-RD, GGA-HLSA-RRD, GGA-HLSA-TRD). (GGA-HLSA-EW, GGA-HLSA-RWS) means GGA-HLSA-EW and GGA-HLSA-RWS have performance with approximate level, or GGA-HLSA-EW is not obviously superior to GGA-HLSA-RWS;
- Based on the previous experimental conclusions, EX_3 validates the feasibility of the algorithm we proposed to solve the MDRSP in realistic cloud computing. In addition, it demonstrates that combining the strategies of NSGA II with GHW can obtain better solutions than compared algorithms. The order by performance is: GHW-NSGA II > (GHD-NSGA II, GHWS-NSGA II) > NSGA II;
- Finally, EX_4 compares the GHW family (GHW-NSGA II and GHW-MOEA/D) with the state-of-the-art NSGA II and MOEA/D by observing the HVs-over-time in some large-scale dataset and observing the average probability to find theoretical optimal Pareto solutions over time in some small scale dataset. EX_4 demonstrates that the GHW family algorithms we propose have a faster convergence rate and better optimality than NSGA II and MOEA/D, that is (GHW-NSGA II, GHW-MOEA/D) > (NSGA II, MOEA/D). The results of EX_4 show that the algorithms we propose provide a comprehensive and significant improvement compared to the reference algorithms.

6. Conclusion and future work

The Multi-Dimensional Resources Scheduling Problem (MDRSP) in cloud computing, a multi-objective optimization problem, is challenging because the resources of each dimension are usually heterogeneous and coupled. The solution of MDRSP requires simultaneous consideration of multi types of resources, which makes MDRSP far more complex than the single-dimensional resource scheduling problem.

In this paper, we focus on the allocation of VMs in heterogeneous multi-dimensional resources of cloud computing and formulate several MDRSPs including minimizing the maximum utilization rate of each dimension of resources and minimizing the energy consumption of the total system. To solve these MDRSPs, we firstly use the concept of stages in genetic algorithm to divide its processes into four stages namely initialization stage, infancy stage, mature stage and genetic stage; secondly add a growth stage to the genetic algorithm and propose GGA-HLSA-RW (GHW, i.e., Growable genetic algorithm using the Heuristic-based local search algorithm with random Weights as growth route). To concretize HLSA, we proposed a modified LSPT algorithm, which can improve the solutions of MSOPs in heterogeneous nodes. GHW, a hybrid algorithm combining meta-heuristic, heuristic and local search, has strong adaptability and optimality for various MDRSPs. The proposal of the GHW family allows the flexible combinations of various algorithms. To further improve the performance of GHW family of algorithms, we finally propose GHW-NSGA II and GHW-MOEA/D applying the sorting strategies and regeneration mechanism of NSGA II and MOEA/D into GHW.

To validate the performance of the GHW family, we carried out extensive experiments on the simulation dataset and AzureTraceforPacking2020-driven dataset. The experiments not only validate the growth strategy and dimensionality reduction strategy of GHW outperforming baselines, but also validate the

feasibility and superiority of GHW in realistic cloud computing. Compared with state-of-the-art NSGA II and MOEA/D in experiments, GHW-NSGA II and GHW-MOEA/D have better convergence rate and optimality, which shows the comprehensive advantages of the GHW family of algorithms.

The other significance of the GHW family is that it shows the great potential of adding a growth stage to GA and demonstrates that combining with multi-search routes may be able to improve convergence and optimality.

On this basis, it is a worthwhile direction to explore the architecture and theory of more stable genetic algorithms or other multi-search route algorithms. The processing speed and energy consumption of electronic components are always affected by many factors, such as network congestion, temperature, continuous working time et al. We in the future plan to explore the variants of GHW and apply them to address MDRSPs in more complex scenarios considering the capacities of resources and conversion formula of energy consumption are time-varying in dynamic systems. In addition, accelerating GHW through distributed computing is also an important topic affecting the development of GHW family in the big data era.

CRediT authorship contribution statement

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

Declaration of competing interest

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

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 with Grant ID 2021YFG0325, and National Natural Science Foundation of China with Grant ID 61672136 and 61828202.

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] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT, Future Gener. Comput. Syst. 93 (2019) 278–289.
[3] W. Guo, W. Tian, Y. Ye, L. Xu, K. Wu, Cloud resource scheduling with deep reinforcement learning and imitation learning, IEEE Internet Things J. 8 (5) (2021) 3576–3586.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] L. Ghalami, D. Grosu, Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm, J. Parallel Distrib. Comput. 133 (2019) 221–231.
[10] A.L. Bolaji, F.Z. Okwonu, P.B. Shola, B.S. Balogun, O.D. Adubisi, A modified binary pigeon-inspired algorithm for solving the multi-dimensional knapsack problem, J. Intell. Syst. 30 (1) (2021) 90–103.
[11] M. Abdel-Basset, R. Mohamed, K.M. Sallam, R.K. Chakrabortty, M.J. Ryan, BSMA: A novel metaheuristic algorithm for multi-dimensional knapsack problems: Method and comprehensive analysis, Comput. Ind. Eng. 159 (2021) 107469.
[12] H. Goudarzi, M. Pedram, Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems, in: L. Liu, M. Parashar (Eds.), IEEE International Conference on Cloud Computing, CLOUD 2011, Washington, DC, USA, 4–9 July, 2011, IEEE Computer Society, 2011, pp. 324–331.
[13] 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.
[14] C. Dhaenens, L. Jourdan, Metaheuristics for Big Data, 2016, http://dx.doi.org/10.1002/9781119347569.
[15] D. Gabi, A.S. Ismail, A. Zainal, Z. Zakaria, A. Abraham, N.M. Dankolo, Cloud customers service selection scheme based on improved conventional cat swarm optimization, Neural Comput. Appl. 32 (18) (2020) 14817–14838.
[16] Y. Laili, S. Lin, D. Tang, Multi-phase integrated scheduling of hybrid tasks in cloud manufacturing environment, Robot. Comput.-Integr. Manuf. 61 (2020) 101850.
[17] 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.
[18] AzurePublicDataset, https://github.com/Azure/AzurePublicDatase.
[19] A. Sudarshan Chakravarthy, C. Sudhakar, T. Ramesh, Energy efficient VM scheduling and routing in multi-tenant cloud data center, Sustain. Comput. Inform. Syst. 22 (2019) 139–151.
[20] A. Roy, S. Midya, K. Majumder, S. Phadikar, Distributed resource management in dew based edge to cloud computing ecosystem: A hybrid adaptive evolutionary approach, Trans. Emerg. Telecommun. Technol. 31 (8) (2020).
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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).
[27] Q. Li, H. Yao, T. Mai, C. Jiang, Y. Zhang, Reinforcement-learning- and belief-learning-based double auction mechanism for edge computing resource allocation, IEEE Internet Things J. 7 (7) (2020) 5976–5985.
[28] 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.
[29] 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.
[30] 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.
[31] 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.
[32] 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.
[33] 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.

[34] 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.
[35] 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.
[36] 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.
[37] 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.
[38] W. Xia, L. Shen, Joint resource allocation at edge cloud based on ant colony optimization and genetic algorithm, Wirel. Pers. Commun. 117 (2) (2021) 355–386.
[39] A.M.S. Kumar, M. Venkatesan, Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment, Wirel. Pers. Commun. 107 (4) (2019) 1835–1848.
[40] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, X. Shu, A dynamic ant-colony genetic algorithm for cloud service composition optimization, Int. J. Adv. Manuf. Technol. 102 (1–4) (2019) 355–368.
[41] K. Xie, X. Wang, G. Xie, D. Xie, J. Cao, Y. Ji, J. Wen, Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing, IEEE Trans. Serv. Comput. 12 (6) (2019) 925–940.
[42] N. Bao, Y. Chai, Y. Zhang, C. Wang, D. Zhang, More space may be cheaper: multi-dimensional resource allocation for NVM-based cloud cache, in: 38th IEEE International Conference on Computer Design, ICCD 2020, Hartford, CT, USA, October 18–21, 2020, IEEE, 2020, pp. 565–572.
[43] Y. Pan, L. Gao, J. Luo, T. Wang, J. Luo, A multi-dimensional resource crowdsourcing framework for mobile edge computing, in: 2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7–11, 2020, IEEE, 2020, pp. 1–7.
[44] H. Yu, Z. Zhou, Z. Jia, X. Zhao, L. Zhang, X. Wang, Multi-timescale multi-dimension resource allocation for NOMA-edge computing-based power IoT with massive connectivity, IEEE Trans. Green Commun. Netw. 5 (3) (2021) 1101–1113.
[45] A. Gopu, N. Venkataraman, Optimal VM placement in distributed cloud environment using MOEA/D, Soft Comput. 23 (21) (2019) 11277–11296.
[46] T. Nurcahyadi, C. Blum, Negative learning in ant colony optimization: application to the multi dimensional knapsack problem, in: S. Deb (Ed.), ISMSI 2021: 2021 5th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, Victoria, Seychelles, April 10-11, 2021, ACM, 2021, pp. 22–27.
[47] M. Yu, C. Wu, B. Ji, J. Liu, A sum-of-ratios multi-dimensional-knapsack decomposition for DNN resource scheduling, in: 40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10–13, 2021, IEEE, 2021, pp. 1–10.
[48] M.S. Aktar, M. De, S.K. Mazumder, M. Maiti, Multi-Objective Green 4-dimensional transportation problems for breakable incompatible items with different fixed charge payment policies, Comput. Ind. Eng. 156 (2021) 107184.
[49] J. Chen, H. Wu, F. Lyu, P. Yang, X.S. Shen, Multi-dimensional resource allocation for diverse safety message transmissions in vehicular networks, in: ICC 2021 - IEEE International Conference on Communications, Montreal, QC, Canada, June 14-23, 2021, IEEE, 2021, pp. 1–6.
[50] M. Ehrgott, Multicriteria Optimization, second ed., Springer, 2005.
[51] J.K. Mandal, S. Mukhopadhyay, P. Dutta (Eds.), Multi-Objective Optimization - Evolutionary to Hybrid Framework, Springer, 2018.
[52] J. Yang, H. Zhu, T. Liu, Secure and economical multi-cloud storage policy with NSGA-II-C, Appl. Soft Comput. 83 (2019).
[53] K. Shang, H. Ishibuchi, A new hypervolume-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 24 (5) (2020) 839–852.
[54] S.C. Maree, T. Alderliesten, P.A.N. Bosman, Uncrowded hypervolume-based multiobjective optimization with gene-pool optimal mixing, Evol. Comput. 30 (3) (2022) 329–353.
[55] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evol. Comput. 2 (3) (1994) 221–248.
[56] 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.
[57] 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.
[58] 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.
[59] R. Liu, R. Ren, J. Liu, J. Liu, A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems, Appl. Soft Comput. 89 (2020) 106120.
[60] Z. Tan, H. Wang, S. Liu, Multi-stage dimension reduction for expensive sparse multi-objective optimization problems, Neurocomputing 440 (2021) 159–174.
[61] D. Brockhoff, E. Zitzler, Dimensionality reduction in multiobjective optimization: the minimum objective subset problem, in: K. Waldmann, U.M. Stocker (Eds.), Operations Research, Proceedings 2006, Selected Papers of the Annual International Conference of the German Operations Research-Society (GOR), Jointly Organized with the Austrian Society of Operations Research (ÖGOR) and the Swiss Society of Operations Research (SVOR), Karlsruhe, Germany, September 6–8, 2006, 2006, pp. 423–429.
[62] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[63] 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.
[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] H. Li, K. Deb, Q. Zhang, P.N. Suganthan, L. Chen, Comparison between MOEA/D and NSGA-III on a set of novel many and multi-objective benchmark problems with challenging difficulties, Swarm Evol. Comput. 46 (2019) 104–117.
[66] H. Li, Q. Zhang, Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.
[67] L. Shao, M. Ehrgott, Discrete representation of non-dominated sets in multi-objective linear programming, European J. Oper. Res. 255 (3) (2016) 687–698.
[68] L. Liu, T. Wang, An evolvable hardware method based on elite Partheno-Genetic Algorithm, Appl. Soft Comput. 113 (Part) (2021) 107904.
[69] J. Yang, Y. Hu, K. Zhang, Y. Wu, An improved evolution algorithm using population competition genetic algorithm and self-correction BP neural network based on fitness landscape, Soft Comput. 25 (3) (2021) 1751–1776.
[70] V. Priya, C.S. Kumar, R. Kannan, Resource scheduling algorithm with load balancing for cloud service provisioning, Appl. Soft Comput. 76 (2019) 416–424.
[71] 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.
[72] K.S. Pal, P.P. Wang, Genetic Algorithms for Pattern Recognition, CRC Press, Inc., 1996.
[73] J. Blank, K. Deb, Pymoo: multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.

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, big data processing, parallel training of large-scale model and evolution algorithms especially genetic algorithms.

Wenhong Tian received a Ph.D. degree from the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. He is now a professor at the University of Electronic Science and Technology of China (UESTC). His research interests include scheduling in cloud computing and big data platforms, image recognition by deep learning, algorithmic theory of machine learning, parallel training of large-scale model 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=154, gindex=331, 125200+ 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.