Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Two-stage hybrid genetic algorithm for robot cloud service selection

Lei Yin[1], Jin Liu[1], Yadong Fang[2], Ming Gao[3], Ming Li[1] and Fengyu Zhou[1*]

Abstract

Robot cloud service platform is a combination of cloud computing and robotics, providing intelligent cloud services for many robots. However, to select a cloud service that satisfys the robot's requirements from the massive services with different QoS indicator in the cloud platform is an NP hard problem. In this paper, based on the cost model between the cloud platform, cloud services and cloud service robotics, we propose a two-stage service selection strategy, namely, candidate services selection stage according to the specific QoS requirements of service robots and final cost optimization stage. Additionally, with respect to optimizing the final cost for the model, we propose a Dynamic Vector Hybrid Genetic Algorithm (DVHGA) that is integrated with local and global search process as well as a three-phase parameter updating policy. Specifically, inspired by momentum optimization in deep learning, dynamic vector is integrated with DVHGA to modify the weights of QoS and ensure the reasonable allocation of resources. Moreover, we suggest a linear evaluation method for the service robots and the cloud platform concerning time and final cost at the same time, which could be expected to be used in the real application environment. Finally, the empirical results demonstrate that the proposed DVHGA outperforms other benchmark algorithms, i.e., DABC, ESWOA, GA, PGA and GA-PSO, in convergence rate, total final cost and evaluation score.

Keywords DVHGA, Qos-aware, Cloud robotics, Genetic algorithm, Dynamic vector

Introduction

Service robots are considered to be cutting-edge technologies that improve human life. In recent years, they have gradually been used in homes, supermarkets, airports and other areas to provide patrols, inquiries, and navigation tasks. On the one hand, robots have limited computing resources, making it difficult to deploy complex intelligent algorithms. On the other hand, high prices discourage people from buying robots. High performance robots with low prices are what people need. The trade-off between performance and cost is difficult.

Many researchers have put forward methodologies to solve the above problems. Remote robots are an early typical example [1]. The remote nodes control the robots to perform specific tasks, which improves the intelligence of the robots. The remote end is mainly controlled by humans, and the robot cannot perform environmental data analysis and autonomous control. Subsequently, Kamei [2] put forward the concept of network robots, by building an Internet of robots, providing powerful data resources for robots to eliminate the problem of limited data. However, the data processing method and execution process have yet to be studied. Until 2010, James J. Kuffer [3] proposed the concept of cloud robotics and the basic framework of the robot cloud platform. The intelligence of the robot is encapsulated as a discrete cloud service [4]. The robot is simplified into an information collection terminal, and the

*Correspondence:
Fengyu Zhou
zhoufengyu@sdu.edu.cn
[1] School of Control Science and Engineering, Shandong University, Jingshi Road, 17923 Jinan, China
[2] Inspur Cloud Information Technology Co., Ltd, Inspur Group, Inspur Road, Jinan, China
[3] Academy of Intelligent Innovation, Shandong University, Shunhua Road, Jinan, China

robot cloud platform provides data analysis and decision services for the robot, which improves the robot's intelligence.

As a container for cloud services, the robot cloud platform deploys multiple types of intelligent cloud services, including dialogue, face recognition, and SLAM for a large number of robots [5]. The different performance of hardware resources causes the same cloud service to have different qualities of service(Qos), such as real-time, accuracy, effectiveness, etc. Robots have quality requirements for cloud services to ensure intelligence [6]. Based on the elasticity of cloud computing, more resources are allocated for cloud services to improve the quality of cloud services to satisfy the requirements of robots, which waste computing resources. Therefore, to select the cloud service required by the robot from the massive number of services with different Qos on the cloud platform has research significance.

Rule-based algorithms are often used to solve the scheduling and selection problems of cloud services, such as FCFS, Maxmin, SLA, etc. These strategies are based on the parameters and time characteristics of cloud services to match cloud services and users, and are more suitable for data processing cloud services or resource cloud services with few parameters and stable performance. However, the robot cloud service platform provides AI cloud services for robots. For the complexity of algorithms and the instability of computing resources, cloud service quality parameters are complicated, and robots have different requirements for cloud service quality. To select robot cloud service from cloud platform is an NP hard problem. Heuristic algorithms such as particle swarm algorithm, genetic algorithm, ant colony algorithm and other heuristic algorithms have been proven to be effective methods for cloud service selection. Most of these algorithms are evolved based on natural phenomena, have low requirements on cloud service parameters, and can dynamically track the optimal solution as the parameters change. Genetic algorithm is a natural evolutionary algorithm, which searches for the most individual through the selection of individual populations, crossover and mutation of chromosomes, and has been applied to find solutions in huge service search spaces composed of highly interacting parameters and shown its advantages in parallelism.

In this paper, a detailed model is investigated for the cloud platform as well as cloud robotics based on our previous work in [7]. In addition, we present a novel algorithm for solving the problems of cloud robotics in cloud service selection as well as relieving the above dilemmas. The main contributions of our work are summarized as follows.

(a) We aim at cloud robotics and propose the cost model for the cloud platform, cloud services and service robots, including cloud service scheduling cost, local computing cost, and communication cost. Specifically, the characteristic of cloud services, robots, as well as the cloud platform are investigated in System models section.
(b) We present a two-stage service selection strategy. That is, the candidate services are chosen according to the QoS requirements in the first stage, namely, local selection. Then, the service sequences will be optimized by DVHGA in the second stage called global selection.
(c) DVHGA integrated with local and global search process is proposed to overcome the existing deficits of the state-of-art algorithms. e.g., lacking for characteristic analysis in the relation of service robots, the cloud platform and cloud services, as well as falling into local optimal solution. Moreover, the dynamic vector is first proposed for cloud robotics to modify the weights of QoS requirements, specifically, composed of positive and negative attributes, and realize the dynamic weight coefficient evaluation.
(d) In terms of the real application in the cloud platform, we present a simple linear weight evaluation approach to quickly determine the appropriate algorithm under the consideration of time and cost at the same time. Besides, SLAM service, Dialog service, and face recognition service are shown as three representative applications in the experiment to validate the proposed DVHGA.

The remainder of this paper is organized as follows. Related work section provides the related works on QoS-aware cloud robotics service selection. In System models section, the final cost model of robotics, cloud service, and the cloud platform is discussed. Here, we aim to improve the quality of services as well as meet the tough QoS requirements. The detailed problem definition is presented in Problem definition section. In The proposed algorithm section, the proposed scheduling algorithm and service selection approach are presented. The experimental results are reported in Experiments section. Conclusions and future work are presented in Conclusion section.

Related work

The contradiction between the cost and intelligence of service robots has always affected the development of robots. S jia [8] proposed a robot architecture based on teleoperation, simplifying the intensive computing module of the robot based on manual remote operation.

Considering the small improvement of the robot's intelligence by teleoperating robots, M Sato [9] proposes a network robot system, establishes a robot's public network to share operating data, and enhances the robot's intelligence that robots can learn operating rules from between them. Robots still need to process a large amount of data and cannot be smart at low-cost. In 2010, James Kuffner proposed the concept of cloud robots, combining robots and cloud computing, offloading the robot's intensive computing modules to the cloud. The cloud platform provides robot cloud service of dialogue interaction, image recognition, positioning, navigation, control cloud services, etc for the robot through the cloud-robot interaction interface. Intelligent cloud services improve robot intelligence while uninstalling the machine's computing module to reduce hardware costs of robots.

A large number of cloud services with different parameters and qualities are deployed on the cloud platform. Each cloud service has different parameters and service quality. To select quality-matched cloud services from cloud platforms can improve the intelligence of robots [10–12]. Rule-based algorithms are widely used in cloud service selection due to their simplicity and high efficiency. In [13], researchers mainly focused on ensuring the QoS of the network to utilise the resources and reduce the average blocked users, as well as improve the throughput. In [6], by maximizing QoS and minimizing the cost, the author aimed to achieve efficient use of cloud resources and cloud service units, as well as ensure the normal operation of the system. Casalicchio et al. designed a detailed cloud brokerage based on legal-rule aware and QoS-aware [14]. The broker can evaluate the cloud services from four aspects, including F1-Legal-rule, F2-Legislation, F3-Qos and F4-Seamless service migration. Meanwhile, quality assurance and optimization were composed of the QoS assurance service and the QoS monitor service. While the runtime adaptation and accurate assessment of QoS still exist defects. In [15], a hybrid multi-criteria method was proposed, which combined K-Means with ANP to rank the cloud services and assign weights for clusters. Besides, CPU utilization ratio, memory utilization ratio, response time and cost were taken into consideration, while the evaluation process was too complicated and high-priced. Heidari et al. presented Graph Processing-as-a-Service (Gaas), which took service level agreement (SLA) requirements and quality of service into consideration to minimize the monetary cost and provide appropriate services [16]. Abdul [17] presented an efficient algorithm for identifying the cloud services based on the QoS metric given by the cloud consumer using decision tree classification algorithm to decreases the response time, CPU utilization and memory consumption for identifying and searching the cloud services and increases the accuracy of the CSPs list retrieved along with their QoS attributes. L. Stavrinides et al. presented an energy-efficient, QoS-aware and cost-effective scheduling approach, which applied an improved per-core Dynamic Voltage and Frequency Scaling (DVFS) approach to approximate computations [18]. For specific scenarios, some researchers study the preferred properties of services. In [19], An efficient trust management architecture for selecting cloud services is proposed, which is utilized by the Combined Preference Ranking Algorithm (CPRA) for an initial CSPs and their services ranking based on the requirements of the customer's cloud (CCs). This generates an accurate ranking list of CSPs with minimal execution time, ensuring that transactions are carried out effectively and efficiently.

The previous methods have made important contributions in optimizing computing costs and improving the utilization of cloud resources. However, the process of robot service selection not only considers the optimization of platform resources, more importantly, the selected cloud service indicators satisfy the task requirements of the robot, and the quality indicators of each cloud service are inconsistent. In [20], the author proposed a multi-robot service selection algorithm to provide service applications with the best service, namely, CAS (Circular Area Search algorithm), which is based on the SOA cloud platform. Dingju Zhu [21] presented a cooperative scheduling method for establishing the logistics delivery service model in the cloud robot system. When the number of robotic cloud services increases, the ability of rule-based algorithms to search for optimal services decreases. Shahab Mokarizadeh et al. proposed the basic architecture for service discovery planning based on the SOA paradigm, aiming at exploiting the ontological relations between services and their concepts to recommend high quality services via utilizing service parameters [22]. However, these approaches take neither the QoS requirements nor the characteristics of service robotics into consideration.

Qos-aware cloud service selection has become an important research. Li [23] proposed a hybrid architecture for cloud robotics to extend the capability of the service robots on their specified and predictable tasks without sacrificing QoS requirements. Brugali et al. [24] presented a model-based approach for building the relationship between the variability and Quality-of-Service (QoS) characteristics for autonomous robots. However, despite the achievements of these previous researches, there are two critical issues that need to be considered. For one thing, they focused on only one characteristic of QoS requirements without considering the positive and negative effect attributes. For another thing, when the number of services increases to a certain aspect, these algorithms may not work, even have a worse

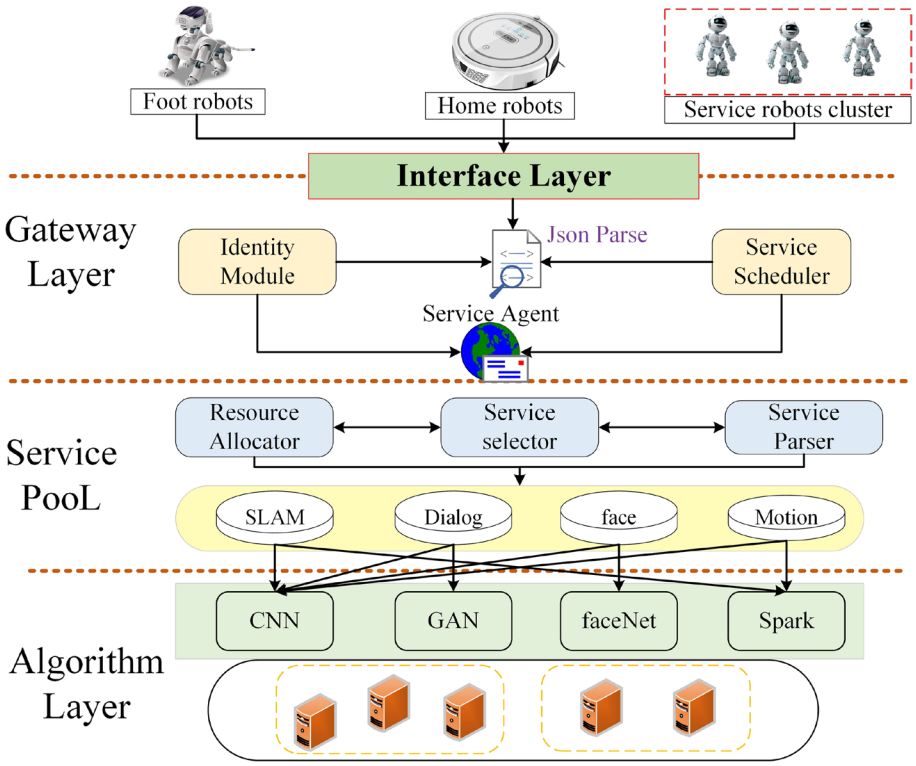

Fig. 1 An architecture of cloud platform

effect. Thus, inspired by the success of meta-heuristic algorithms in the cloud platform, e.g., Eagle Strategy with Whale Optimization Algorithm (ESWOA) [11], QoS-aware ABC algorithm [25], GA-based algorithms [18, 26, 27], etc., we propose a two-stage hybrid genetic algorithm integrated with dynamic vector to obtain the optimal services via global and local search process. Moreover, we are the first to provide a comprehensive study on cloud robot service selection under consideration of positive and negative attributes of QoS requirements, as well as taking the characteristics of service robots, the cloud platform and cloud services into account to optimize the final cost. More specifically, we present a simple linear weight evaluation approach to quickly determine the appropriate algorithm in consideration of execution time and final cost at the same time (Fig. 1).

System models

Cloud robot platform framework

In our previous work [7], we proposed the robot cloud service platform architecture shown in Fig. 1. The platform is composed of an interface layer, gateway layer, service pool, algorithm pool layer. The platform is developed based on the Inspur cloud serve. The interface layer is mainly responsible for providing interfaces, including peripheral devices, internal services. The gateway layer is used to deploy services, schedule services, and authenticate identification. Service pool composed of various cloud services, can allocate service resources, as well as select suitable services or combined services. Algorithm pool layer is formed of self-developed algorithms and running time dockers. For the robot cloud platform, two critical factors need to be considered, i.e., physical machine cost and service communication cost.

With various consumption requirements, cloud services are deployed on the computing nodes. Generally, CPU, Disk, and memory are three critical indicators that affect the Qos of robot cloud services. For a request for face recognition service from robot, the service data including JavaScript Object Notation(JSON) data, image data, and identification data can be represented as PCSD. The performance parameters of the computing node are defined as a triplet (PCU, PCD, PCM), where PCU represents the execution ratio of a single CPU, PCD is the data exchange ratio of Disk, and PCM is the identifier for the memory consumption compressed ratio. UCPC, UCDC, and UCMC are the corresponding costs in unit time, respectively. In this paper, PCSD/PCY $\leq$ 100ms is set on our platform to ensure the timely response of service and the reasonable CPU usage time, and the size of PCSD (L_{PCSD}) $\ll$ the memory of the platform (M_{cloud}) $<$ the storage disk(D_{cloud}). Thus, the specific cost for the face recognition service(*m-th* service) on *k-th* robot is defined in Eq. 1.

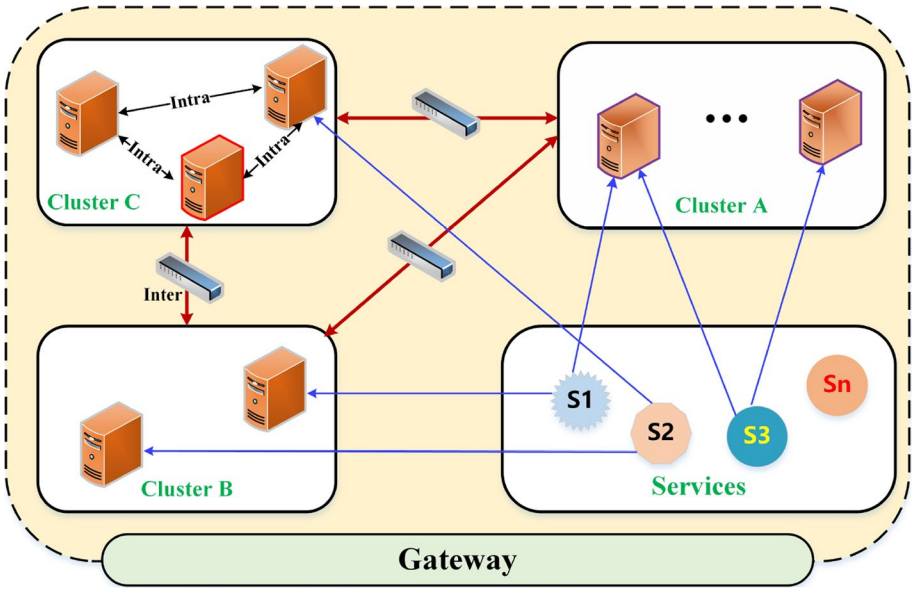

Fig. 2 Inter architecture of cloud platform

$$Fcost_k^m = UCPC_k * \frac{PCSD_m}{PCU_k} * pw_{1mk} + UCDC_k * \frac{PCSD_m}{PCD_k} * pw_{2mk} + UCMC_k * \frac{PCSD_m}{PCM_k} * pw_{3mk} \quad (1)$$

$$s.t. \quad \frac{PCSD}{PCY} \leq 100ms$$

$$L_{PCSD} \ll M_{cloud} < D_{cloud}$$

where pw is the weight of three factors that satisfy the condition in Eq. 2, and recorded into MySql for the registered services associated with robot.

$$1 = \sum_{i=1}^{n} pw_n \quad (2)$$

the weight of pw between legal services and robots as a matrix are defined in Eq. 3.

$$pw = \begin{bmatrix} pw_{1,1} & pw_{1,2} & \cdots & pw_{1,m} \\ pw_{2,1} & pw_{2,2} & \cdots & pw_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ pw_{s,1} & pw_{s,2} & \cdots & pw_{s,m} \end{bmatrix} \quad (3)$$

where $s \in 1, \cdots, k$. k is the total number of quality impact factors of cloud services. $pw_{s,m}$ is the weight of the s-th factor of the m-th cloud service Consequently, costs of all the running services are expressed as follows.

$$POC = \sum_{k=1}^{k} \sum_{m=1}^{m} Fcost_k^m \quad (4)$$

Figure 2 shows the architecture inside the basic cloud platform. services are deployed on various virtual machines in clusters. Communication cost is the important factor, including three communication processes of robot-cloud(RC), cluster-cluster(CC) and node-node in cluster(NN).

Commonly, virtual machines on the same host communicate with high bandwidth, Cloud services in the same cluster can be combined into one cloud service. Thus, the inter-communication cost of cloud platform with e virtual machines and m services are as Eq. 5.

$$INTERC = \sum_{j=1}^{e} \sum_{i=1}^{m} PS_i * (\frac{SD_i}{PF_j} + \frac{SD_i^{'}}{PF_j}) \quad (5)$$

Where PS is the price for communication between different machines toward various services. SD is the data to be packaged and forwarded at the gateway layer. $SD^{'}$ is the processed data length. PF is data transfer ratio. Typically, the size of SD (L_{SD}) and $SD^{'}$ ($L_{SD'}$) is smaller than $PCSD$ after the JSON parser processes. For a set of services, the final cost on the cloud is T_{cloud} in Eq. 6.

$$T_{cloud} = INTERC + POC \quad (6)$$

$$s.t. \quad \frac{PCSD}{PCY} \leq 100ms$$

$$L_{PCSD} \ll M_{cloud} < D_{cloud}$$

$$L_{SD} \text{ and } L_{SD'} < L_{PCSD}$$

Cloud service model

QoS attributes of service are defined as $< q_1, q_2, \cdots, q_l >$, where q_l represent the *l-th* QoS attribute. The most critical factors for cloud services are service response time, best practices, availability, latency, reliability, and throughput. Each cloud service has different requirements for QOS attributes. For instance, face detection service is less time-sensitive than SLAM service. Therefore, for each service's bias towards service quality, we divide QoS attributes into two categories, i.e., positive attributes[availability, reliability, throughput, best practices] and negative attributes[response time, latency].

Maximum-minimum normalization based on [28] in Eqs. 7 and 8 is proposed to make different QoS values in a unified space. By this means, each QoS value is transferred into a value between 0 and 1.

$$q^p_{mi} = \begin{cases} \frac{q^{max} - q_i}{q^{max} - q^{min}} & if \;\; q^{max} - q^{min} \neq 0 \\ 1 - \epsilon & if \;\; q^{max} - q^{min} = 0 \end{cases} \tag{7}$$

$$q^n_{mi} = \begin{cases} \frac{q_i - q^{min}}{q^{max} - q^{min}} & if \;\; q^{max} - q^{min} \neq 0 \\ 1 - \epsilon & if \;\; q^{max} - q^{min} = 0 \end{cases} \tag{8}$$

where q^{max} and q^{min} are the normalized value of the maximum and minimum QoS attributes respectively. q^p_{mi} and q^n_{mi} represent the positive and negative normalized values of Qos attributes with respect to the *m-th* service. ϵ is the bias factor and set to *1e-2*.

Besides, when service robots request a suitable service from the cloud, a set of services SR can be represented as $< SR_1, SR_2, \cdots, SR_k, \cdots, SR_t >$. SR_k from *k-th* service robot is composed of the following sequence, $\{< s_{k1}, sp_{k1}, cp_{k1} >, < s_{k2}, sp_{k2}, cp_{k2} >, \cdots, < s_{km}, sp_{km}, cp_{km} >\}$. Where sp_{km} is the price set by cloud service providers and cp_{km} is the price set by cloud platform providers. s_{km} is the *m-th* service in SR_t of *k-th* service robot. Thus, the cloud service could be represented as $< s_{km}, sp_{km}, cp_{km}, q_{kml} >$, where q_{kml} represents the *l-th* QoS attribute of *m-th* service from *k-th* robot in SR.

Robotics cost model

As task execution terminal and processing terminal for raw data, the cost of a robot is an important factor to consider. Chen et al. proposed a QoS-aware robotic streaming workflow model in solving computation offloading problems based on the characteristics of the NCR and RSW for cloud robotics [29], while the model didn't consider the cost of the robot and service completion condition. In this paper, the local computing cost, communication cost, service completion ratio are introduced to establish the robot cost model.

Generally, the robot is equipped with Raspberry Pi or other mainboards, with ROS or Android. The cost of the robot is mainly related to the calculation attribute. We define the robot as a computing unit. RC is defined as the robot's computing consuming and RP as the power energy consuming at a specific service invocation per seconds. RCPU and RPC are the cost spent per unit time of CPU and power energy respectively. The cost in Δt are obtained in Eq. 9.

$$RLC_k = \sum_{j=1}^{m} (\alpha RC_j * RCPU_k * \Delta t_j + \beta RPC_k * \int_0^{\Delta t_j} RP_j) \tag{9}$$

where RLC_k represents the local computing cost of the *k-th* robot. α and β are the weights of CPU and energy cost.

The robot communication cost model consists of service invocation, data conversion and local bandwidth-consuming (ignored in this article). Based on our previous experience in building a robot cloud platform, for data timeout, JSON file format is used to transfer a small amount of data, and the meta-data denoted by FD is transferred after packing. FL and FL' are JSON data lengths of sending process and receiving process (i.e., JSON file), respectively. To simplify the model, the ratio of which the same kind robots transfer data is set to a constant number between cloud platform and specific robot, i.e., 5MB/s, 50MB/s, 100MB/s, is denoted by RF_k. Thus, the *k-th* robot communication cost can be expressed as Eq. 10.

$$RCC_k = \sum_{j=1}^{m} (SC_j * (\frac{FL_j}{RF_k} + \frac{FL'_j}{RF_k}) + DSC_j \frac{FD_j}{RF_k} + CLC_j) \tag{10}$$

where SC and DSC are the cost of the requesting stage and the data transfer stage, respectively. CLC is the local channel cost and set to 1e-3 considering the local channel's high-speed transfer ratio.

The final cost of robot is shown in Eq. 11 derived from Eqs. 9 and 10.

$$T_{robot} = \sum_{i=1}^{k} (RLK_k + RCC_k) \tag{11}$$

When robot requests a set of services, if the service can't be completed before its deadline, the robot will continue to apply for other similar cloud services to replace the previous service. the SCR(service completion ration, SCR) in a service set is defined as follows.

$$SCR = \frac{SFS}{TSS} \tag{12}$$

where SFS represents the successful completed service and TSS is the total number of a service set.

To improve the quality of service scheduling model on the cloud platform. RSLA(Robot Service-Level Agreement, RSLA) is defined in Eq. 13. RSLA reflects the satisfaction ratio of the service. In this paper, we set the following rules. If SCR is greater than 95%, the platform will record this service sequence and recommend it to a similar scene. It is the necessary premise for the selected services in our algorithms. When it is between 80% and 95%, the service scheduler will remove all the failure services and record it. Otherwise, the service set will be released all, and the service SLA will be registered. Once the service fails three times, the service provider and corresponding robot will be informed to troubleshoot the problem.

$$RSLA = \begin{cases} 1, & SCR \geq 95\% \\ 0, & 95\% > SCR \geq 80\% \\ -1, & SCR < 80\%. \end{cases} \tag{13}$$

Problem definition

The robot's service selection process is divided into two stages: local service selection and global service selection. As shown in Fig. 3. In the local selection process, a set of best candidate services were selected via dynamic QoS rank and randomly dropped. QoS is chosen through specific rules to satisfy the robot's preference requirements for cloud services, e.g., when the robot requests SLAM service, the scheduler would prefer more time-sensitive relative services. In the global selection process, candidate services circled in red line are formed as a service set, and the final cost, including communication cost, scheduling cost, and platform operation cost are minimized as well as hold a high service completion ratio.

In the first stage, many methods can be used to select candidate services. e.g., in [30], ANP was used to rank the QoS attribute for the first stage. In [24], AHP was used. In other conditions, fixed weight was introduced [11, 31]. In our algorithm, a performance formula was proposed as Eq. 14 to speed up the calculation efficiency. One service set can be computed in a whole matrix. The procedure to select the best top S performance candidate service of *k-th* service is shown as Eq. 15. Moreover, the *LPs* is obtained in parallel to accelerate the process.

$$LP_r = \sum_{g=1}^{l} W_{mg} * Q_{mg}^{nor} \qquad r \in t \quad [m * 1 \; size] \tag{14}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1l} \\ w_{21} & w_{22} & \dots & w_{2l} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{ml} \end{bmatrix}$$

$$Q^{nor} = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1l} \\ q_{21} & q_{22} & \dots & q_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \dots & q_{ml} \end{bmatrix}$$

where LP_r is obtained by the Hadamard product denotes the specific performance service sequence of SR, and the size of the result matrix is $m * 1$. Services possess different weight evaluation standard. Typically, robots present different cloud service QoS requirements that can be selected via the service orientation and dynamic weights. Q^{nor} represents the normalized value matrix of QoS. which is calculated via Eqs. 7 and 8. W denotes the dynamic weight of services.

$$S^{set} = rank_{topS}\{LP_r\} \qquad r \in t \tag{15}$$

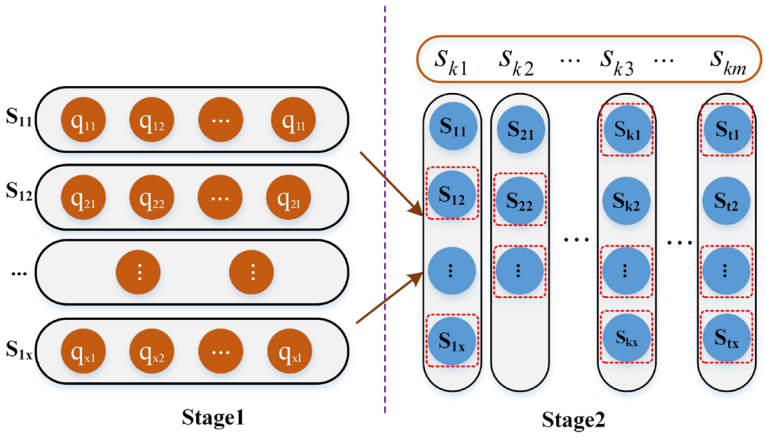

Fig. 3 Service selection process

For the second stage, minimizing total cost is an important optimization goal. Since all the cost have the same affect, the main goal is formulized as Eq. 16.

$$min \sum \{T_{cloud} + T_{robot}\} \tag{16}$$

$$s.t. \begin{cases} SCR \geq 95\% \\ s_{km} \in S^{set} \\ k, m = 1, 2, \ldots \\ PCSD/PCY \leq 100ms \\ L_{PCSD} \ll M_{cloud} < D_{cloud} \\ L_{SD} \; and \; L_{SD'} < L_{PCSD} \end{cases}$$

The proposed algorithm

To select cloud services to achieve the minimum total cost is the optimization goal of the algorithm. Meanwhile, a more efficient scheduler should be designed to shorten the service selection time. Therefore, the fast and accurate algorithm of Dynamic Vector Hybrid Genetic Algorithm(DVHGA) is presented.

Require: The set of service requests SR.
Ensure: A set of suitable scheduling services CSS.
1: Initialize the parameters using **Algorithm 2**
2: Detect the cluster condition and modify the weights using **Algorithm 3.**
3: Select top services set using **Eq.14** and **Eq.15.**
4: Initialize the population composed of PS chromosomes.
5: Leader_num,Leader_pos ← Evaluate the solutions using **Eq.16.**
6: **while** MG or Best solution has not been reached **do**:
7: Sort the population and get $rsf*PS$ mating parents.
8: Do crossover and mutation using **Algorithm 4.**
9: Local_num,Local_pos ← Evaluate the current populations using **Eq.16**
10: **if** Leader_num > Local_num **then**:
//Global modification
11: Leader_pos ← Local_pos.
12: Leader_num ← Local_num.
13: Combine the population with Leader_pos.
14: **else**:
15: Local serach using **Algorithm 5.**
16: Update the current search parameters using **Algorithm 6.**
17: Update the current step and store the best solution.
18: **end while**
19: **return** best suitable service set.

Algorithm 1 Procedure of the Basic DVHGAAlgorithm 1 shows the procedure of DVHGA composed of six main modules, e.g., parameter initialization, dynamic vector modification, crossover and mutation, current parameters updating, local search process, and global improvement.

For each set of service requests, all the system parameters will be initialized first. Once the gateway layer accepts the sequences, the cloud monitor will detect the cloud platform's condition and modify the weights. After that, the best *top-h* candidate services will be selected for the most suitable service requests. The main loop framework continues until the iteration reaches the threshold or the best solution is found. At each stage, the best position and fitness are determined by the evaluating procedure after mutation, crossover, local search, and global modification. The current algorithm parameters will be updated, and the current global best solution will be stored. Finally, the best suitable service set will be sent to the scheduler.

Paramter Initilization. The process in Algorithm 2 includes three steps. Firstly, all the QoS are normalized by using Eqs. 7 and 8. Secondly, global parameters that include the main run-time parameter are initialized properly. Thirdly, for the local search, the basic particle swarm optimization (PSO) framework is used to accelerate the search procedure.

1: Initialize service Qos using **Eq.7 and Eq.8.**
2: Initialize evolution weights $< w_1, w_2, \cdots, w_6 >$.
3: Set global parameters:
Initialize the set of CPU utilization ΔCPU, the memory utilization ΔMem, the network utilization $\Delta NetU$; The monitoring interval ratio ΔMR; he basic genetic algorithm parameters, including population size PS, the chromosome length CL, crossover probability CP, mutation probability MP, maximum generation MG**.**
4: Set local parameters
Initialize the acceleration coefficient c_1 and c_2 using random manipulation. Set reserve factor rsf. Set the times of local search MIL.
5: **return** the intialized parameters.

Algorithm 2 Parameter Initialization**Dynamic Vector Modification.** In our research, fixed weight factors can lead to a high cost, and the result will be analyzed in Experiments section. Thus, we propose the strategy of dynamic vector modification process inspired by gradient descent, and the vector moving direction may be positive or negative. Thus, the negative factor will be set to 0 here. The sum of all weights is 1, while an extreme case should be considered that the summation of the total factor becomes zero after the manipulation, which is solved in the following Eq. 17.

$$w = \begin{cases} \frac{w}{w_{sum}} & if \;\; w_{sum} \neq 0 \\ \frac{w}{w'_{sum}} & if \;\; w_{sum} = 0 \end{cases} \tag{17}$$

Where $w' = w + rand()$, $w'_{sum} = \sum w'$ and $w_{sum} = \sum w$. The random small perturbations are applied for the former w to eliminate the bad effect of zero summation.

1: Initialize vector velocity ratio VR
2: Detect the cluster condition.
3: **for** w_x in w **do**:
4: set Δ to the corresponding operation.
5: change w_x using $w_x = w_x - VR * rand * \Delta$.
6: **end for**
7: Calculate the summation of w as w_{sum}.
8: Modify the w using Eq.14.
9: **return** w.

Algorithm 3 Dynamic Vector Modification**Crossover and Mutation**. In Algorithm 4, multi-point crossover manipulation is designed for superior parents, and to produce more superior mutated individuals from the inferior mating pool.

Local Search Process. PSO is used to explore more feasible solutions as well as speed up coverage. The strategy of solutions to update velocity and position using

Eqs. 18 and 19 is similar to PSO. Finally, the global solution will be updated and returned to the main framework.

$$v_i^{k+1} = rand * v_i^k + c_1 * r_1 * (c\_pos - pos_i^k) + c_2 * r_2 * (Leader\_pos - pos_i^k) \quad (18)$$

$$pos_i^{k+1} = pos_i^k + v_i^{k+1} \quad (19)$$

1: Choose the parents from mating pool randomly.
2: Select crosspoint randomly for each pair of parents.
3: Apply multi-point crossover and generate crossover offsprings.
4: Choose the parents from inferior pool randomly.
5: Select mutation point for each individual.
6: Apply multi-point mutation and generate mutation offsprings.
7: Combine the three offsprings.
8: **return** new offsprings.

Algorithm 4 Crossover and Mutation

1: Initialize a population around Leader_pos, velocity v, random number r_1 and r_2.
2: c_num,c_pos ← Evaluate the fitness using **Eq.16.**
3: **while** has not been reached MIL **do**:
4: Update velocity using **Eq.18.**
5: Update positions using **Eq.19.**
6: Evaluate the fitness using **Eq.16.**
7: Update the local best solution c_pos and c_num.
8: **end while**
9: Update the global solution.
10: **return** best solution.

Algorithm 5 Local Search Process**Current parameter updating**. In Algorithm 6, improvement threshold (IT) is proposed to ensure that the excellent solution set increases faster in the primary stage. Then, escape threshold (ET) is proposed to prevent falling into local optimality. Finally, searching threshold (ST) is designed for exploring more superior solution. All the three threshold is shown in Eq. 20

$$\begin{cases} IT \leftarrow (round(MG/2), round(MG * 3/4)) \\ ET \leftarrow (round(MG * 3/4), round(MG)) \\ ST \leftarrow (round(MG * 4/5), round(MG)) \end{cases} \quad (20)$$

1: **switch** current step state
2: **case one :** within IT **then :**
3: Reserve factor will be set randomly.
4: **end case one**
5: **case two :** within ET **then :**
6: Reset CP and MP by adding a random number
7: **end case two**
8: **case three :** within ST **then :**
9: MIL will be updated by twice times than before.
10: **end case three**
11: **return** new parameters.

Algorithm 6 Current parameter updating**Complexity analysis of DVHGA**. The time computational complexity of of DVHGA can be described as $O(I \times (N + P + C) + Q)$, in which I represents the number of iterations; N denotes the calculation time of genetic algorithm; P is the number of particles in PSO; C denotes the calculation time in current parameter updating process; and Q is the calculation time in the first-stage for selecting the candidate service.

Moreover, the time computional complexity of the original GA-PSO algorithm can be defined as $O(I \times (N + P) + Q)$. We can observe that the computational complexity of DVHGA is slight higher than the GA-PSO due to local search process, updating process and first-satge calculation. However, DVHGA can obtain superior solutions in reasonable time significantly. Thus, in the Algorithm evaluation section, we will suggest a linear evaluation approach for choosing the appropriate algorithm in different circumstances.

Experiments

In this section, the experiment is designed to compare DVHGA and several state-of-art algorithms of discrete artificial bee colony (DABC) [25], eagle strategy whale optimization algorithm (ESWOA) [11], genetic algorithm (GA) [12], proposed genetic algorithm (PGA) [32], and GA-PSO. In this section, these contents will be analyzed, such as the convergence of DVHGA, various number of services, different cloud platforms (c1.medium, c1.large, c1.xlarge) features and robots features (v1 version, v2 version, v3 version) defined into low, middle and high levels, the parameter state of DVHGA and the algorithm evaluation.

Experimental setup

To reduce the influence of programming language on algorithm performance, all comparison algorithms are run in matlab environment. Moreover, the experiments are conducted on Intel(R) Core(TM) i7-8700 with 3.2GHZ and 32GB RAM running on the 64-bit of Windows 10.

Algorithm Parameters: The main parameters of the mentioned algorithms are shown in Table 1. The parameters are set to maximize the performance for all the comparable algorithms.

DataSet: the QWS dataset v2.0 including over 2500 services from the real world with QoS is used. Ten kinds of services that each contains only four concrete services are selected as the reference. To ensure the fairness of algorithm, 1000 cloud service data sets with Qos parameters are expanded by 30 specific service include general services (face recognition service, etc.) and specific services (typically time-oriented services, e.g., SLAM, reliability-oriented services, e.g., identification service). As is shown in Table 2, each service from the dataset contains

Table 1 The cloud platform simulation parameters

Parameter	Value
Number of particles (DNPSO)	25
Population size	30
Total maximum itertion	100
pw1,pw2,pw3	0.6,0.3,0.1
w[1..6]	0.2,0.1,0.2,0.2,0.1,0.2
topS	20
the number of robots	30

six attributes i.e., availability, reliability, throughput, best practice, response time, and latency.

Experiment on convergence

For the mechanism of coding and evolution, genetic algorithm has very good performance in service selection research, especially for solving discrete solutions. However, the direction of solving the optimal solution uncertain for the randomness of mutation and crossover. The following two problems are verified based on GA through experiment results in which the service length is set 100 and robot is low performance.

As can be seen from the Fig. 4(a). Although the final cost decreases in the previous iteration (left of pink line) and tends to convergence, the dispersion and volatility can't be neglected, which increase the cost of service scheduling. The result of Fig. 4(b) is worse. Algorithm can't keep the best solution, and the final cost gradually increases with volatility.

In this paper, algorithm of swarm optimization is interated with DVHGA to perform local search process. Figure 5 show the convergence result of DVHGA and other algorithms over 100 iterations. DVHGA converges quickly than ESWOA and DABC, and has better search results. Moreover, DVHGA can search the solution space efficiently via local search as well as three current parameter updating strategy, and improve the ability to search for global solutions. Figure 6 shows the computational time of the above four algorithms. Search time of DVHGA is longer than ESWOA and GAPSO due to its search strategy. However, search time of DVHGAhas the smallest variance compared to the other three algorithms, which could enable the scheduler to schedule in a reasonable way and minimize the overall system cost.

Experiment on fixed and dynamic weights of DVHGA

In this subsection, Service selection experiments of 100 services are presented to improve that the performance of the dynamic parameter algorithm in cost optimization is better than the fixed parameter algorithm. As shown in Fig. 7, DVHGA algorithm has a stronger search capability

Table 2 Description of the Qos Parameters

Parameter	Value	Units
Availability	Number of successful invocations/total invocations	%
Reliability	Ratio of the number of error messages to total messages	%
Throughput	Total Number of invocations for a given period of time	invokes/second
Best Practices	The extent to which a Web service follows WS-I Basic Profile	%
Response Time	Time taken to send a request and receive a response	ms
Latency	Time taken for the server to process a given request	ms

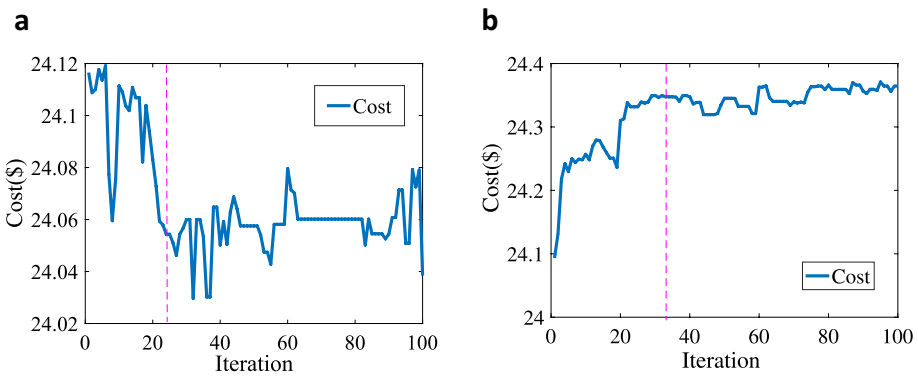

Fig. 4 Discretization of genetic algorithm. **a** descend trend, **b** ascend trend

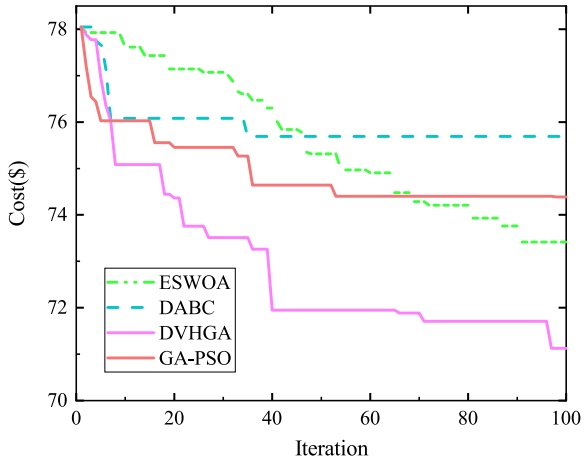

Fig. 5 The cost convergence curve

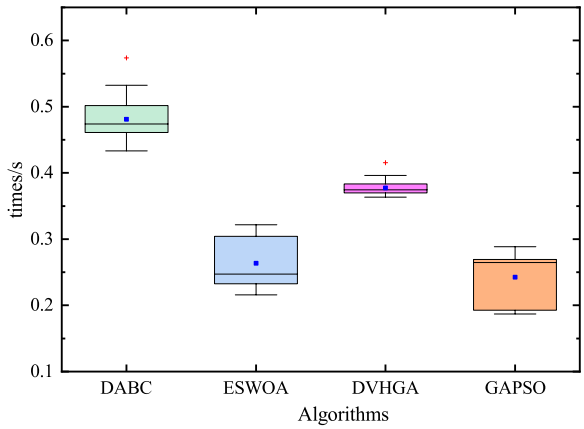

Fig. 6 The computational time

than the static GA algorithm. The reason for this result is that we used a matrix to calculate the weight. For the powerful processing of the cloud platform, the time-consuming could be neglected in this paper.

Based on the above experimental settings, the number of services will be increased from 500 to 1000 at intervals of 50. the result shows in Fig. 8. For the fluctuation of platform and robot parameters in its state, the calculation time fluctuates as well with several numerical spikes. However, the longest extra time is only 1.4ms and small enough to be neglected. Additionally, the vast majority of the extra time is less than 0.2ms, which could help the scheduler work in a short time and cut the final cost.

Experiment on various number of services

In this section, We demonstrate the performance of different algorithms by setting up different numbers of 100, 500,700,1000 cloud service selection experiments.

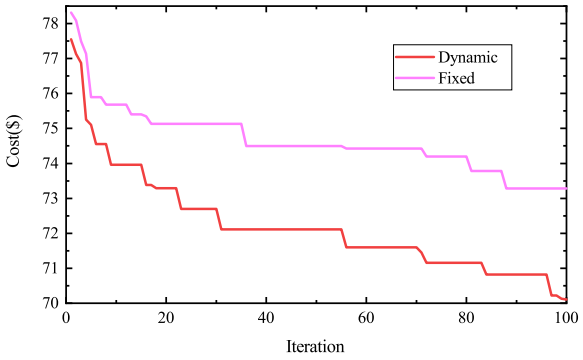

Fig. 7 The algorithm cost

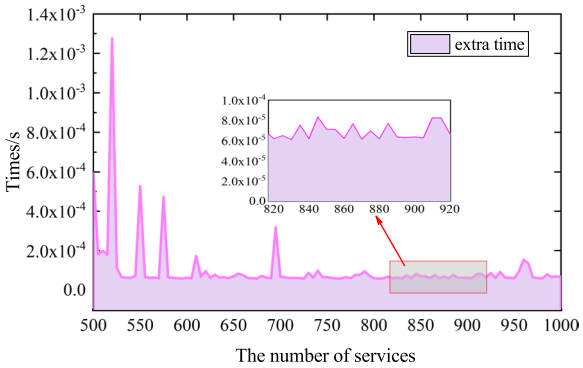

Fig. 8 The time fluctuation

All the algorithms run 30 times in a total of 100 iterations without early stopping. As shown in Fig. 9 of experimental results. Since the advantages of global and local search, DVHGA outperforms than other comparison algorithms in the optimization of service selection costs with or without dynamic vectors. we can also find that the final cost increases as the number of services becomes large. Intuitively, DVHGA shows significant improvements compared with other benchmarks, especially GA-PSO, when the number of services rises from 700 to 1000. This mainly due to that DVHGA adopts the three-stage update strategy, i.e.,primary stage, middle stage, late stage, and the multi-point crossover for superior parents, which makes it easier for the DVHGA to find the optimal strategy. Additionally, compared Fig. 9(a) with (b), all the algorithms are improved in different degrees due to dynamic vector.

To illustrate the affect of dynamic vector on the algorithms, we define the difference of cost in Eq. 21. According to the equation, the result is shown in Fig. 10.

$$\delta_{cost} = C_{without} - C_{with} \tag{21}$$

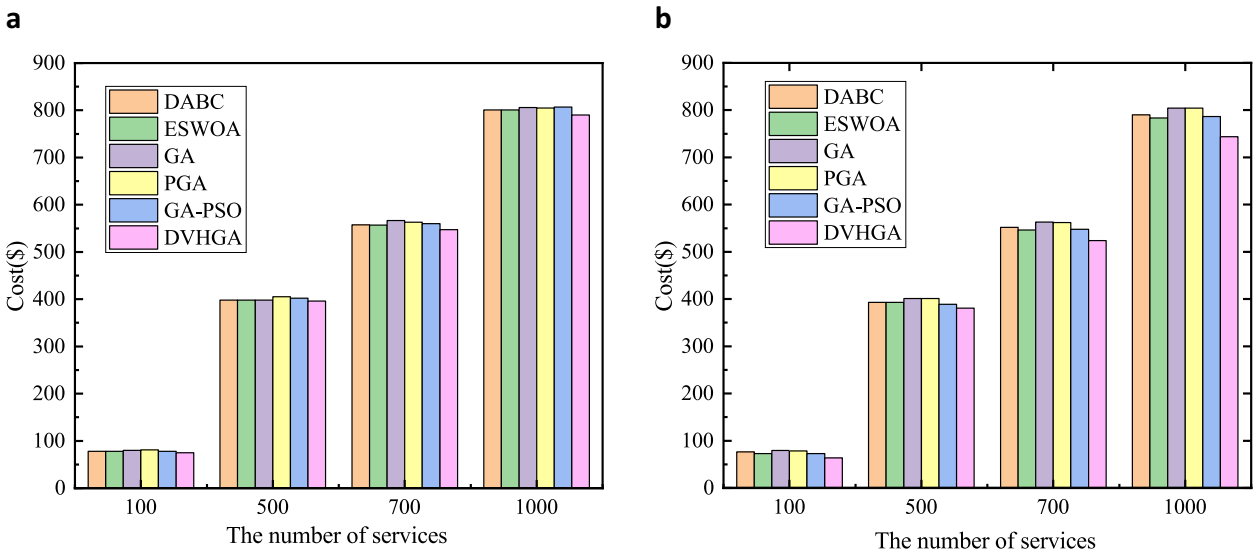

Fig. 9 Experiment results of different number of services. **a** services without dynamic vector, **b** services with dynamic vector

According to the equation, the experiment results are shown in Fig. 10. The dynamic vector has less influence on GA and PGA. Dynamic vector obtains approximately the same effect on GA-PSO when the number of services is 500 and 700. In contrast, DVHGA has a significant increase. dynamic vector can enhance the distinct performance of algorithms in swarm algorithms and slight in genetic algorithms. Since the uncertainty of genetic direction, GA and PGA are difficult to search for the optimal solution. For the introduction of a three-stage parameter update strategy, DVHGA can optimize the process of chromosome compilation to quickly capture the optimal solution. The empirical results show that when the number of services increases to 1,000 or more, DVHGA can obtain satisfactory performance.

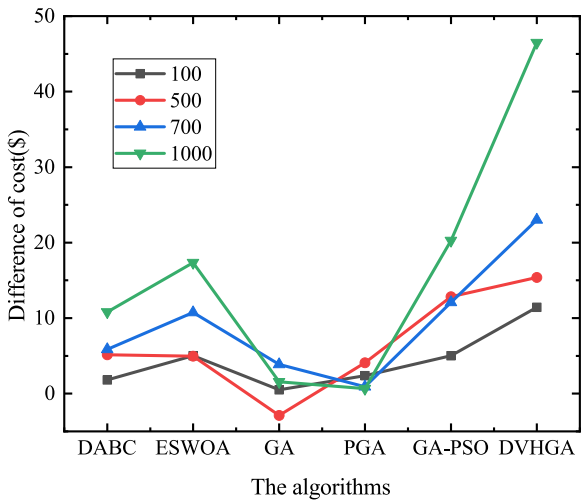

Fig. 10 The results of cost difference for algorithms

Experiments with different configurations of robots and cloud platforms

Features of robots

In this experiment, low, mid and high features are chosen for robots. The number of services is set as 100 and 500. All the algorithms are integrated with dynamic vector to obtain superior solutions. The results of final cost shows in Fig. 11.

When the robot cloud service deploys a fixed number of cloud services, as the robot configuration increases, the cost of cloud service selection increases. In this case, when the number of cloud services increases from 100 to 500, the cost of choosing cloud services will increase greatly. The result is due to the limited computing resources of the robot. although the final cost rises in the experiment, for the global and local search as well as updating strategy, DVHGA has a lower cost compared with the above-mentioned benchmark algorithm.

Features of cloud platforms

In this experiment, a low-configuration robot is used as a cloud service call terminal. The cloud service platform has three configurations of low, medium and high. The number of services is also 100 and 500. Besides, the algorithms dynamic vector to improve their performance as well.

Figure 12 shows the cloud service cost under different cloud platform configurations. When the number of services increases from 100 to 500, the increase in cost of services is much smaller than that shown in Fig. 11. Moreover, when the number of services is 500, the final cost remaines virtually unchanged near 400$. This mainly because of the sufficient computing power and resources,

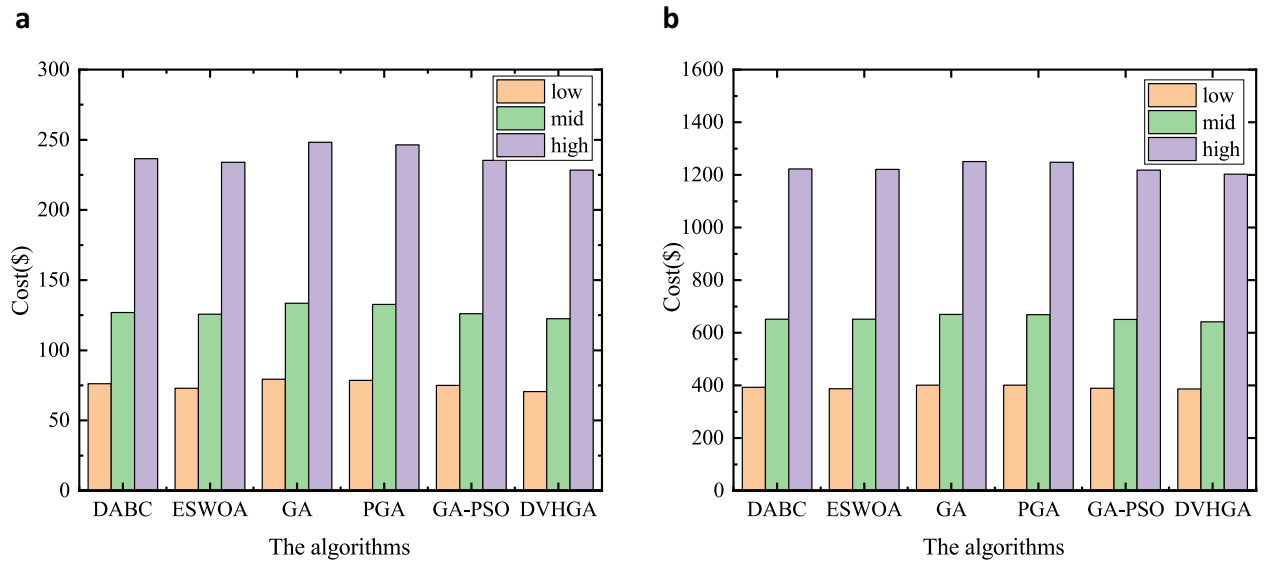

Fig. 11 Different features of robots. **a** 100 services, **b** 500 services

which could accelerate the operation and reduce the final cost for service robots and cloud platform. Sufficient resources to maximize the algorithm performance of cloud services. In addition, the final cost of DVHGA grows in an acceptable range by setting different performance platforms.

The cloud service of the platform adopts the pay-as-you-go model. Cloud services will not occupy physical hosts for a long time. It can be inferred from the above two experiments that the architecture of the cloud robot is a low-performance robot with high-performance cloud providers, which can shift the cost to the cloud provider and promote the birth of more efficient scheduling.

Algorithm evaluation

In the application of the robot cloud service platform, the cloud platform must satisfy the robot's cloud service QOS Request to improve robot intelligence and QoE (Quality of Experience) for users. Different cloud services require different QOS attributes. The simultaneous localization and mapping(SLAM) cloud service cares more about response time. Face recognition cloud services pay

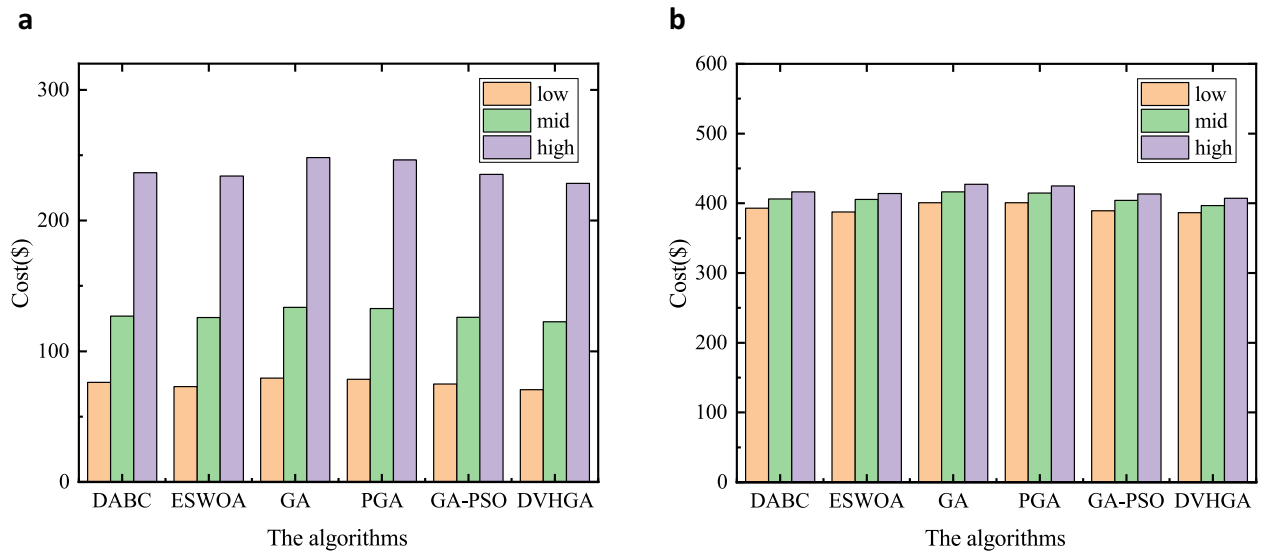

Fig. 12 Different features of cloud platform. **a** 100 services, **b** 500 services

more attention to accuracy and requires the least cost. In this experiment, we proposed a simple idea in our design to evaluate the algorithm on for robot and the cloud platform side, which is shown in Eq. 22.

$$T = \alpha * time_a + \beta * cost_a - \gamma * (var(time) + var(cost)) \quad (22)$$

Where α, β, and γ represents the weight coefficients, satisfying $\alpha + \beta + \gamma = 1$. $time_a$ and $cost_a$ are the average of time and cost. $var(time)$ and $var(cost)$ denote the variance of time and cost, respectively. All the basic service information is stored in information repository for both robots and cloud platform. As for time-sensitive cloud service, α can be more significant than β, and γ is a bias for balancing the cost and time. To improve the user's QOE, the priority of the robot is defined as higher than the priority of the cloud platform. When there are no other constraints, we give priority to ensuring the response time of cloud services.

In this experiment, Low-profile cloud platforms and robots are used. The number of cloud services is set to 100 and 500 respectively, and algorithms equipped with dynamic vector. Additionally, three kinds of services, namely, dialog service (α=0.3, β=0.3, γ=0.4), SLAM service (α=0.7, β=0.2, γ=0.1), and face recognition service (α=0.2, β=0.7, γ=0.1), are selected as three typical robot cloud service.

Tables 3 and 4 illustrate the results for different evaluation approaches toward 100 and 500 services. As can be seen from the tables, DVHGA remaines the best performance comparing with the benchmarks. Moreover, different algorithms rank differently with various service requirements evaluation. Thus, based on the evaluation approach, we could select a suitable algorithm under various circumstances, whereas it is linearly related to the cost and time.

Conclusion

In this paper, we propose a cloud service selection method, including two-stage selection strategy, dynamic weight and cloud service quality evaluation method, and introduce a new search strategy into local and global search to ensure the solving efficiency of the optimal solution.

In the experiment, we evaluated DVHGA algorithms with four state-of-art algorithms, i.e. DABC, ESWOA, PGA, and GA-PSO, and a traditional common algorithm, i.e. GA. The empirical results demonstrate that DVHGA outperforms the benchmarks in cost-savings with diverse characteristic of service robots and cloud platform, converges faster and is more stable than other algorithms. Furthermore, the experimental results also suggest that The combination of low-configuration robots and cloud services on high-configuration cloud platforms will result in lower cloud service costs. Fixed-weights and dynamic-weights vector were stduied via several experiments to verify the superiority in dynamic vector modi- fication.

Table 3 100 services(†:first, *:second)

	Dialog service		SLAM service		Face recognition	
Project	**results**	**rank**	**results**	**rank**	**results**	**rank**
DABC	24.27	4	16.42	4	56.46	4
ESWOA	23.47	2*	15.85	2*	55.07	2*
GA	25.12	6	16.91	6	58.04	5
PGA	23.97	5	16.82	5	58.75	6
GA-PSO	23.68	3	15.98	3	55.61	3
DVHGA	22.82	1†	15.43	1†	53.10	1†

Table 4 500 services(†:first, *:second)

	Dialog service		SLAM service		Face recognition	
Project	**results**	**rank**	**results**	**rank**	**results**	**rank**
DABC	125.24	4	84.44	5	291.81	5
ESWOA	123.43	3	83.32	2*	289.74	2*
GA	125.7	6	85.11	6	298.42	6
PGA	125.35	5	16.82	4	296.96	4
GA-PSO	123.34	2*	15.98	3	289.41	3
DVHGA	120.34	1†	15.43	1†	284.86	1†

In our future work, we intend to integrate the concept of DVHGA with other meta-heuristic algorithms. Additionally, since different services have distinct requirements for service quality parameters in actual experiments, we will concentrate on studying robot service selection under varied preferences.

Abbreviation

DVHGA Dynamic vector hybrid genetic algorithm

Acknowledgements

No further acknowledgments to provide than those already listed in the authors' section. In addition, the authors thank the anonymous reviewers for providing valuable comments to improve this paper.

Authors' contributions

L. Yin contributed on the design of algorithms. F. Zhou presented the ideal and the performance evaluation. Y. Fang and M. Gao designed the experiment. J. Liu and M. Li participated in the design and optimization of framework. The authors read and approved the final manuscript.

Funding

This work was supported in part by the National Key R & D Program of China under Grant 2017YFB1302400, the Jinan "20 New Colleges and Universities" Funded Scientific Research Leader Studio under Grant 2021GXRC079, the Major Agricultural Applied Technological Innovation Projects of Shandong Province under Grant SD2019NJ014, the Shandong Natural Science Foundation under Grant ZR2019MF064, and the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2019IRS19.

Availability of data and materials

The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Ethics approval and consent to participate

No ethical approval is required, and the authors express their consent to participate in the paper.

Competing interests

The authors declare no competing interests.

Received: 25 January 2023 Accepted: 18 May 2023
Published online: 22 June 2023

References

1. Fong T, Thorpe Charles, Baur Charles (2003) Multi-robot remote driving with collaborative control. IEEE Trans Ind Electron 50(4):699–704
2. Kamei K, Nishio S, Hagita N et al (2012) Cloud networked robotics. IEEE Netw 26(3):28–34
3. Kuffner J 2010) Cloud-enabled humanoid robots. In humanoid robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, 2010
4. Mohanarajah G, Hunziker D et al (2015) Rapyuta: A Cloud Robotics Platform. IEEE Trans Autom Sci Eng 12(2):481–493. https://doi.org/10.1109/TASE.2014.2329556
5. Waibel M, Beetz M, Civera J et al (2011) RoboEarth - A World Wide Web for Robots. IEEE Robot Autom Mag 18(2):69–82
6. Wu Z, Lu Z, Hung P et al (2018) QaMeC: A QoS-driven IoVs application optimizing deployment scheme in multimedia edge clouds. Futur Gener Comput Syst 92:17–28
7. Liu J, Zhou F, Yin L et al (2019) A Novel Cloud Platform For Service Robots. IEEE Access PP(99):1–1
8. Jia S, Takase K (2002) Internet-based robotic system using CORBA as communication architecture. J Int Robot Syst 34(1):121–134
9. Sato M, Kamei K, Nishio S, Hagita, N (2011, December) The ubiquitous network robot platform: common platform for continuous daily robotic services. In 2011 IEEE/SICE International Symposium on System Integration (SII). IEEE, p 318–323
10. Dong F, Luo J, Jin J, Shi J, Yang Y, Shen J (2017) Accelerating skycube computation with partial and parallel processing for service selection. IEEE Transactions on Services Computing 13(6):969–984
11. Gavvala SK, Jatoth C et al (2019) QoS-aware cloud service composition using eagle strategy. Futur Gener Comput Syst 90:273–290
12. GoldbergD E (1989) GeneticA igorithm in search, optim ization and machinelearning. Reading: Addison2wes2 ley.
13. Khan M, Fakhri ZH, Al-Raweshidy HS (2017) Semi-Static Cell Differentiation And Integration With Dynamic BBU-RRH Mapping In Cloud Radio Access Network. IEEE Trans Netw Serv Manag PP(99):1–1
14. Casalicchio E, Cardellini V, et al (2018) Research challenges in legal-rule and QoS-aware cloud service brokerage. Futur Gener Comput Syst 78(3). https://doi.org/10.1016/j.future.2016.11.025
15. Al-Faifi A, Song B, Hassan MM et al (2018) A Hybrid Multi Criteria Decision Method for Cloud Service Selection from Smart Data. Futur Gener Comput Syst 94(APR):43–57
16. Heidari S, Buyya R (2019) Quality of Service (QoS)-driven resource provisioning for large-scale graph processing in cloud computing environments: Graph Processing-as-a-Service (GPaaS). Futur Gener Comput Syst 96:490–501
17. Quadir MdA, Varadarajan V, Mandal K (2019) Efficient algorithm for identification and cache based discovery of cloud services. Mob Netw Appl 24(1):1181–1197
18. Jatoth C, Gangadharan GR, Buyya R (2015) Computational Intelligence based QoS-aware Web Service Composition: A Systematic Literature Review. IEEE Trans Serv Comput 10(3):475–492
19. Qadir MdA, Vijayakumar V (2020) Combined preference ranking algorithm for comparing and initial ranking of cloud services. Recent Adv Electr Electron Eng 13(2):260–275
20. Zhou HB, Zhang JJ, Liu ZZ et al (2019) Research on Circular Area Search algorithm of multi-robot service based on SOA cloud platform. Appl Soft Comput 88:105816
21. Zhu D (2018) IOT and big data based cooperative logistical delivery scheduling method and cloud robot system. Futur Gener Comput Syst 86(SEP):709–715
22. Mokarizadeh S, Dokoohaki N, Matskin M, et al (2010) Trust and Privacy Enabled Service Composition Using Social Experience. Conference on e-Business, e-Services and e-Society. Springer, Berlin, Heidelberg
23. Li Y, Wang H, Ding B, Shi P, Liu X (2016) Toward QoS-aware cloud robotic applications: a hybrid architecture and its implementation. 2016 Intl IEEE conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). IEEE, p 33–40
24. Brugali D, Capilla R, Mirandola R, Trubiani C (2018, January) Model-based development of qos-aware reconfigurable autonomous robotic systems. 2018 second ieee international conference on robotic computing (irc). IEEE 2018:129–136
25. Wang X, Xu X, Sheng QZ et al (2019) Novel Artificial Bee Colony Algorithms for QoS-Aware Service Selection. IEEE Trans Serv Comput 12(2):247–261
26. Jatoth C, Gangadharan GR, Buyya R (2019) Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm. Future Generations Comput Syst 94:185–198
27. Seghir F, Khababa A (2018) A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. J Intell Manuf 29:1773–1792
28. Mabrouk NB, Beauche S, Kuznetsova E, et al (2009) QoS-Aware Service Composition in Dynamic Service Oriented Environments. ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, Berlin, Heidelberg
29. Chen W, Yaguchi Y, Naruse K, Watanobe Y, Nakamura, K (2018) QoS-aware robotic streaming workflow allocation in cloud robotics systems. IEEE transactions on services computing 14(2):544–558

30. Zheng X, Martin P, Brohman K et al (2014) Cloud Service Negotiation in Internet of Things Environment: A Mixed Approach. IEEE Trans Ind Inf 10(2):1506–1515
31. Liu ZZ, Sheng QZ, Xu X et al (2019) Context-aware and Adaptive QoS Prediction for Mobile Edge Computing Services. IEEE Trans Serv Comput PP(99):1–9
32. Abdulhamed AA, Tawfeek MA, Keshk AE (2018) A genetic algorithm for service flow management with budget constraint in heterogeneous computing. Futur Comput Inf J 3(2):341–347

Publisher's Note