Chapter 9
A Hybrid RCO for Dual Scheduling of Cloud Service and Computing Resource in Private Cloud

In this chapter, the idea of combining SCOS and OACR into one-time decision in one console is presented, named Dual Scheduling of Cloud Services and Computing Resources (DS-CSCR) [1]. For addressing large-scale DS-CSCR problem, Ranking Chaos Optimization (RCO) is configured. With the consideration of large-scale irregular solution spaces, new adaptive chaos operator is designed to traverse wider spaces within a short time. Besides, dynamic heuristic and ranking selection are hybrid to control the chaos evolution in the proposed algorithm.

9.1 Introduction

Newly developing cloud computing [2, 3] has brought about great benefits to both enterprises and individuals. With advanced technologies of virtualization and service, it incorporates various resources for user on-demand with open interfaces and transparent remote operations. While IBM, Google and Amazon are taking the lead in building general public cloud [4–6] under the modes of SaaS (Software as a Services), IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) [7], many conglomerates have also obtained cost reduction and higher flexibility of resource sharing with the establishment of their own private cloud.

Private cloud of conglomerate usually consists of a set of virtualized distributed infrastructures and application services which are provided by couples of sub-enterprises and partner-enterprises [8, 9], as shown in Fig. 9.1. For outside, such conglomerate could be a large SaaS provider. For inside, it turns to a shared resource pool. In a fairly secure environment, all resources are under the ownership and control of a single administrative domain. On one hand, the virtualization of multiple distributed infrastructures can greatly improve the computing capability for the whole organization with lower-cost. On the other hand, upper layer application services, no matter provided to outside Internet or inside members,

F. Tao et al., *Configurable Intelligent Optimization Algorithm*,
Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-3-319-08840-2_9

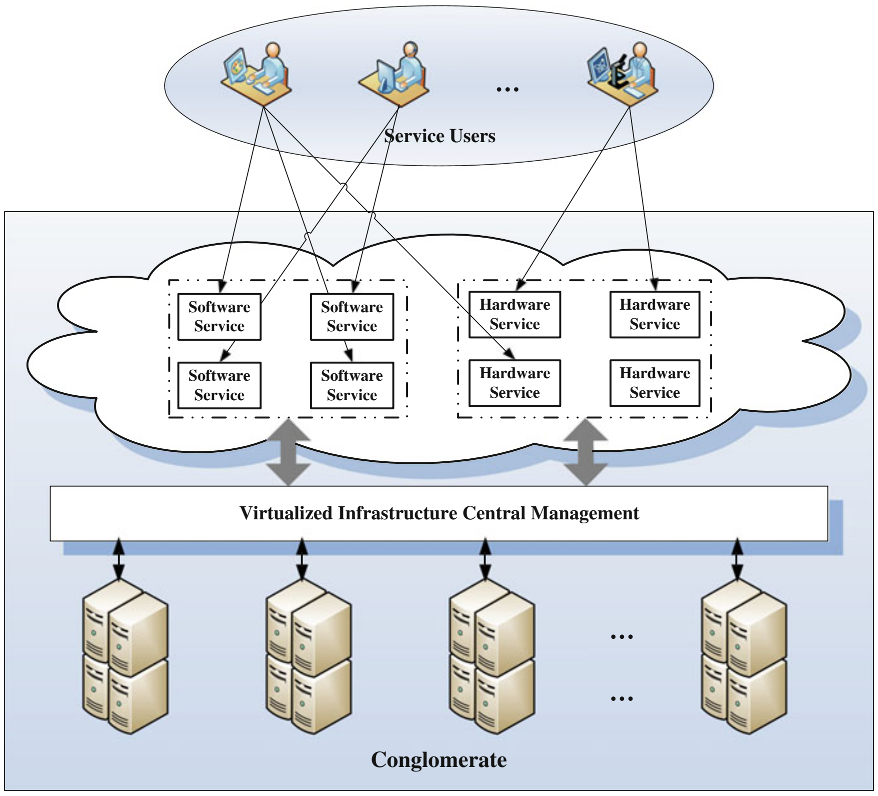


Fig. 9.1 Structure of private cloud and actors in conglomerate

need no longer to be deployed on a fixed computing resource with specific maintenance. Services with central control become more flexible with dynamic allocation. Thus, private cloud in conglomerate also contains two aspects of significance, one is the integration and sharing of underlying distributed infrastructure, another is the flexible deployment and usage of upper layer application services.

Besides, with the development of cloud, the concept of "service" in traditional Service-Oriented Architecture (SOA) is extended from software application to generalized "cloud service" with the inclusion of both software applications and hardware equipments with good interoperability, self-organization and scalability. The properties of cloud services have become more complex and most of them need higher computing ability to drive.

In such environment, when a composite project (which contains a set of tasks) is submitted, the console of conglomerate needs not only to aggregate suitable cloud services with different functionalities and generate service portfolio for user on-demand, but also choose available computing resources to support the running of cloud services. How to achieve high-quality and low-cost services composition optimal selection (SCOS) and optimal allocation of computing resources (OACR) simultaneously are critical for efficient project execution, green resource sharing and flexible service management.

At present, service composition and computing resource allocation in cloud have been studied preliminarily. Most researches are carried out according to the methodology of cluster computing, grid computing and high performance computing and consider the two problems independently. For one thing, computing availability and communication route of computing resources are analyzed. For another, QoS (Quality of Service) indexes and description languages are also discussed. In general public cloud, SCOS and OACR are performed in two steps and in the charge of different actors. Service providers are not infrastructure providers [3]. However, in private cloud of conglomerates with typical SaaS mode, they would provide suitable service portfolio and deploy corresponding services on their own infrastructure for customers on demand. The actors of SCOS and OACR turn out to be the same one.

With such two-step decision by a single administrator, the properties of upper layer selected cloud services in SCOS will limit the range of the underlying available computing resources for each service in OACR. Better portfolios of cloud services and computing resources are easily overlooked. Furthermore, as all knows, both SCOS and OACR are proved to be NP combinatorial optimization problem. Under the condition of large-scale cloud services and computing resources and complex relationship between them, addressing SCOS and OACR step by step with two different algorithms independently becomes very cumbersome and inefficient.

Therefore, we propose the idea of combining two stages decision-making into one and put forward the concept, Dual Scheduling of Cloud Service and Computing Resource (DS-CSCR), in private cloud of conglomerate. In the guidance of this idea, we analyze the complex features of hardware/software cloud service and computing resource in cloud computing in two levels and explore their mutual relations in-depth. Aiming at green efficient decision, the formulation of DS-CSCR with multi-objectives and multi-constraints is presented in this chapter. Additionally, in order to achieve high efficient one-time decision in DS-CSCR, a new Ranking Chaos Optimization (RCO) is designed in this chapter. Take the advantage of chaotic random ergodicity, this algorithm combines new adaptive chaos optimal strategy with ranking selection and dynamic heuristic mechanism to balance the exploration and exploitation in optimization. With adaptive control of chaotic sequence length, it's especially good at searching in large-scaled irregular solution space and shows remarkable performance for addressing DS-CSCR compared with other general intelligent algorithms.

9.2 Related Works

Nowadays, the most commonly used and analyzed cloud computing platforms are "Google cloud computing" platform, Amazon "elastic cloud" platform and IBM "blue cloud" platform. Private cloud with closed sharing are researched less and attracted criticism owing to the less hands-on management [4]. But it can notably reduce the cost of resources and improve the quality of services in large conglomerate. After years of development, large enterprises, academic institutions and new emerging internet service providers are building their own cloud platform, too, such as Eucalyptus, Red Hat's cloud, OpenNebula and so on. Though various platforms differ on their usage mode and openness, most of them share the same key technologies and target of resource sharing.

In cloud computing, two crucial optimization factors in determining resource sharing efficiency and platform application performance are SCOS and OACR exactly.

In recent years, researches on service composition are generally based on the environment of grid computing and other SOA mode [10]. These researches spread from service description language, service QoS indexes [11], reliability and trust evaluation [12], and optimal selection of services [13] and so on. Since cloud computing mode has been proposed, the concept and content of cloud service are broadened. According to the characteristics of cloud computing, semantic properties of cloud service are studied [14]. The classification, management, provision, storage and evaluation of cloud services are investigated widely. Pre-decision and online-decision of SCOS are also deliberated in different ways, such as [15]. Among these, QoS indexes of cloud services are discussed most widely. From the perspective of non-functional properties of cloud services, the existing indexes consider no more than cost, time and reliability factors. It's hard to describe various cloud services with different classification and attributes in a unified form. Thus the existing QoS indexes can't satisfy all types of cloud services.

For computing resource allocation, traditional researches mostly focus on the modeling and evaluation of computing resources based on homogeneous/heterogeneous cluster systems or distributed grid computing systems [16]. User's demand for resources, resources' costs and computation and communication capabilities of resources are the major considerations among these studies. In cloud computing mode, virtualization is the main support of flexible resource sharing [17]. In this context, Endo et al. introduced the concept, classification of resource allocation in distributed cloud [18]. Ma et al. [19] and Xiong et al. [20] investigated the management of cloud computing resources based on ontology and virtualization respectively. Zhang et al. [21] proposed a method for the deployment of upper layer software cloud services from virtual machines. Ghanbari et al. [22] have studied the feedback-based optimization problem including the allocation of resources especially in private cloud. Besides, considering the virtual division of computing resources and its influences on the quality of cloud services, researchers also built new models for computing resources from the rules, reliability and dynamic partition point of view, and so on, and presented various methods to solve OACR problem in cloud computing [23, 24]. Most of these studies concentrated on the expansion of characteristics of computing resources based on traditional models and the algorithm designing for OACR in cloud computing. However, the mutual relations between cloud services and the underlying computing resources and the influence of virtualization on quality of cloud services, as two of the key factors in cloud computing, have not been studied.

In addition, SCOS and OACR are both combinatorial optimization problems. For this kind of problems, the most widely used algorithms are intelligent algorithms due to its NP complexity. It includes Genetic Algorithm (GA) [25], Particle Swarm Optimization (PSO) [26] and so on and has the virtues of brachylogy, universality and rapidity. According to different specific problems, abundant researches mainly focus on the balance of exploration and exploitation in searching process based on evolutionary iteration of population and presented many kinds of improved hybrid intelligent algorithms such as [27]. Nevertheless, these improved hybrid intelligent algorithms are mostly problem-dependent with local convergence more or less. For addressing large-scaled DS-CSCR problem in private cloud of large conglomerate with irregular solution space efficiently, the design of high performance intelligent algorithm is imperative.

9.3 Motivation Example

Currently, the concept of cloud is studied and applied in almost every field. Based on the technology of cloud computing, manufacturing equipments and simulation software as cloud services can be realized [28, 29]. Various software and hardware can by dynamically shared for product customization of both inside or outside organizations without repeat-purchase. Under this background, we use “the design and NC (Numerical Control) machining of a complex surface part in conglomerate cloud” as a case to describe the whole process from tasks’ submission to tasks’ execution. As shown in Fig. 9.2, it can be divided into five sub-tasks: (1) technical and mathematical analysis, (2) CAD modeling and NC programming, (3) verification simulation and post-processing, (4) first NC machining and measuring, and (5) batch production.

During this process, task (1), (2) and (3) can be implemented directly by manufacturing software cloud services, such as CATIA, MasterCAM or Pro/E, etc., and task 4 and task 5 can be executed by manufacturing hardware cloud service with users’ supervision and control, such as 3-axis, 4-axis or 5-axis linkage CNC (Computer Numerical Control) machines, etc. When user submitted the tasks of designing and machining a customized part, four steps are needed to be done by centre console: (1) Requirement analysis of tasks, (2) Services composition optimal selection, (3) Optimal allocation of computing resources, (4) Execution.

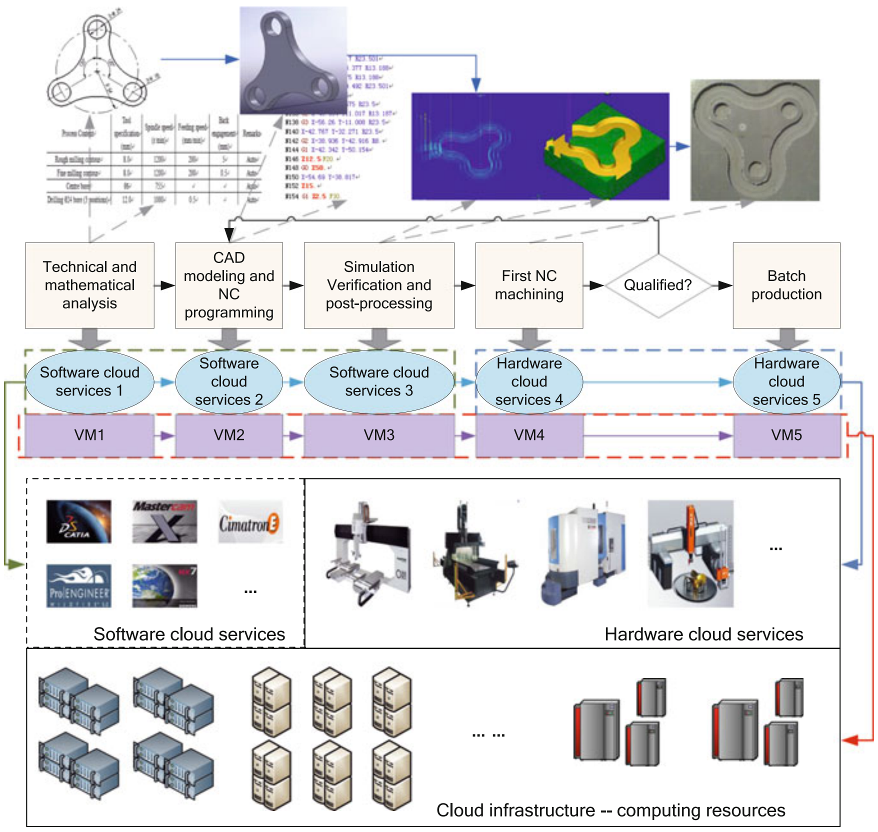

Technical and mathematical analysis
CAD modeling and NC programming
Simulation Verification and post-processing
First NC machining
Qualified?
Batch production
Software cloud services 1
Software cloud services 2
Software cloud services 3
Hardware cloud services 4
Hardware cloud services 5
VM1
VM2
VM3
VM4
VM5
Software cloud services
Hardware cloud services
Cloud infrastructure -- computing resources

Fig. 9.2 The design and NC machining of a customized part in conglomerate cloud

The scheduling of computing resources totally depends on the corresponding upper layer selected cloud services. With the distributed characteristics of services and infrastructures, the available computing resources are reduced and the OACR are constrained by the upper layer decision. For example, for task (4), assume the suitable CNC hardware service No. 1 and No. 2 are provided in Location A and Location B respectively. CNC service No. 1 is with higher QoS than CNC service No. 2. But the idle computing resources in Location A are less than Location B. If CNC service No. 1 is selected for task 4 in step 3, the low computing ability of computing resource in Location A and the remote communication overhead of computing resource in Location B would both cause the low execution efficiency of CNC service No. 1. If we select CNC service No. 2, the better available adjacent computing resource would then improve the overall execution efficiency of task 4. However, the decision of SCOS in step 2 usually disregards the influence of the underlying support computing resources due to the traditional binding mode of service and infrastructure. The latter strategy of choosing MasterCAM service

No. 2 is then overlooked. At this time, you might say, if SCOS and OACR are performed at the same time, then bad decision won't be happened.

Therefore, in order to reduce the time and improve the quality of decision, we merge SCOS and OACR into one dual-scheduling decision. With the purpose of efficient DS-CSCR decision, the following three issues are needed to be studied.

(1) QoS indexes of software/hardware cloud services and computing resources respectively and the mutual relation between them;
(2) The problem formulation of DS-CSCR with multi-objectives and multi-constraints in private cloud;
(3) The efficient scheduling algorithm for addressing large-scale DS-CSCR problem.

This chapter will directly focus on these three issues.

9.4 Problem Description

9.4.1 The Modeling of DS-CSCR in Private Cloud

In conglomerate, services and the support infrastructures are provided by distributed sub-enterprises and controlled by central head. Traditionally, service provider usually deploy the service to a fixed computer, put service and computing resource together to ensure the quality of service. The support computing resources are always occupied by fixed service and needed specific maintenance. With new cloud mode, services can be encapsulated and registered to cloud and deployed to virtual machines dynamically. Through the collaborative development of upper layer applications and underlying resources, all of the resources can be shared flexibly on-demand with more energy-saving, higher redundancy and reliability.

Moreover, based on such a flexible environment, cloud services with the support of VMs contain not only software cloud services, but also hardware services with further expansion. For hardware cloud services, the computing resources are no longer support carriers, but controlling and monitoring facilities for these manufacturing equipments.

(1) The characteristics and QoS indexes of cloud services

From the perspective of QoS evaluation, only simplified quantitative cost, time and reliability cannot comprehensively summarize the characteristics of software/hardware cloud services and their requirement for VMs' performance. With the consideration of the difference between software and hardware cloud services and their demands for VM configuration, this section gives new evaluation indexes for software/hardware cloud service and virtual resources respectively.

(a) The characteristics and QoS indexes of software cloud services

Software applications in cloud computing are running with the support of VMs. Each software service is deployed to a single VM and mapped to a corresponding computing resource. Thus the minimum requirements of VM which represents the required volume of services should be defined to facilitate the allocation of computing resources. Based on the functional description of services, we consider mainly the following non-functional factors of software cloud services in this chapter.

- s—service execution efficiency under the minimum required configuration of VM;
- c—the rent cost of service;
- r—trustiness of service, which is the ratio of the success execution time and the total execution time;
- v—the minimum required speed of VM.

Remarks The performance of the required VM is determined by many factors, such as the CPU and memory of the corresponding computing resources. In a computer, the speed of CPU is in proportion to the power supply voltage [30]. It's a constant value. The speed of VM can mainly be calculated by the number and speed of occupied CPUs. So that the minimum required speed of VM is adopted here for evaluation. The higher the speed of VM is, the faster the service runs.

(b) The characteristics and QoS indexes of hardware cloud services

Unlike the software service, hardware service is energy-consuming and needs supervision or control during execution. Real-time supervision or control will produce large amount of communication and increase service execution time (i.e. the time-consumption of data transmission). Different hardware service needs different amount of supervision and control. For this reason, based on the above four factors of software service, two more factors need to be considered.

- s—service execution efficiency under the minimum required configuration of VM;
- c—the rent cost of service;
- r—trustiness of service, which is the ratio of the success execution time and the total execution time;
- v—the minimum required speed of VM;
- e—the average energy-consumption of hardware service;
- ζ—the average control rate, which is the ratio of the amount of control commands and the amount of tasks;
- η—the transmission rate between VM (computing resources) and hardware service.

Remarks For hardware services, there are two conditions of control. One is inputting all control commands beforehand, and then executing tasks without interaction. Another is controlling during execution. Owning to the large amount of task in hardware service, ζ in the first condition can usually be ignored (i.e. $\zeta = 0$). We mainly focus on the second condition. Besides, if the hardware service needs no control or supervision any more, then $\zeta = 0$, too.

Usually, the transmission path of the control commands of software service is “user—VM”, while which of hardware service is “user—VM—hardware service”. Without the consideration of task interactions and energy-consumption of VMs, if the amount of submitted task is W, the total execution time T, the total cost C and the total energy-consumption E of the software and hardware service can be calculated as follows respectively.

For software services,

$$T = \frac{W}{s} \tag{9.1}$$

$$C = Tc = \frac{cW}{s} \tag{9.2}$$

For hardware services,

$$T = \frac{W}{s} + \frac{W\zeta}{\eta} = W\frac{\eta + s\zeta}{s\eta} \tag{9.3}$$

$$C = Tc = cW\frac{\eta + s\zeta}{s\eta} \tag{9.4}$$

$$E = Te = eW\frac{\eta + s\zeta}{s\eta} \tag{9.5}$$

(2) The characteristics and QoS indexes of VMs

VMs are the virtual division of the underlying computing resources. The performance of VM are mainly embodied in the running speed, transmission rate and energy consumption of the corresponding computing resources. It’s still hard to locate one VM into multiple computers by existing technologies of virtualization. Hence, we assume each VM maps into only one physical node. In accordance with the characteristics of cloud services, we primarily concentrate on four factors below.

- ***p***—the running speed of VM, which depends on the occupancy rate and the speed of CPUs;
- ***q***—the transmission rate of VM;
- ***g***—the average energy-consumption of VM;
- ***f***—the failure probability of VM;
- ***u***—the recovery time of VM when fails.

Remarks q reflects the transmission rate between the occupied physical computing resources and the objects. If the transport object and the VM are in the same local network, then evaluate the transmission rate by local bandwidth. Else, the transmission rate is evaluated with the synthetic consideration of the transport object, the central console and the VM itself. Besides, the energy function of CPU per unit time can be represented as [29]: $P_0 = AV^2 f + Z$. Where A and Z are constant, V is the power supply voltage and f is the dominant frequency. Thus g is in proportion to p, too. In cloud platform, the way to handle the failures of physical nodes is usually dynamic migration of VMs. So, u is no longer the recovery time of the corresponding computing resource but the dynamic migration time. Computing resources with low reliability can easily cause dramatically increase of task execution time, cost and energy consumption.

Let the task execution time in the corresponding VM without failure be t, the average task execution time of VM can be evaluated as:

$$\tilde{t} = t(1 - f) + (t + u)f = t + fu \tag{9.6}$$

Assume the set of the predecessor nodes of the task i to be $\mathbf{L_i}$, and the input communication amount from the predecessor node j is U_{ij}, then the total communication time between the task and its predecessor nodes are:

$$U = \max_{j \in \mathbf{L_i}} \frac{U_{ij}}{q_j} \tag{9.7}$$

If the performance of VM can satisfy the minimum requirement of service, then the total execution time T, the total cost C and the total energy consumption E of the task can be calculated as follows.

(a) If the selected service is software cloud service, then

$$T = \frac{vW}{ps} + U + fu \tag{9.8}$$

$$C = Tc = \left(\frac{vW}{ps} + U + fu\right)c \tag{9.9}$$

$$E = Te = \left(\frac{vW}{ps} + U + fu\right)e \tag{9.10}$$

(b) If the selected service is hardware cloud service, then

$$T = \frac{vW(\eta + s\zeta)}{ps\zeta} + U + fu \tag{9.11}$$

$$C = Tc = \left(\frac{vW(\eta + s\zeta)}{ps\zeta} + U + fu\right)c \tag{9.12}$$

$$E = T(g + e) = \left(\frac{vW(\eta + s\zeta)}{ps\zeta} + U + fu\right)(g + e) \tag{9.13}$$

9.4.2 Problem Formulation of DS-CSCR in Private Cloud

According to the analysis of the characteristics and QoS indexes of cloud services and virtual resources, the abstract formal description of cloud services, VMs and computing resources are elaborated in this Section.

Definition 1 The set of tasks in cloud computing environment can be presented as a directed acyclic gragh (DAG) $G = (N, W, U, H_t, H_c, H_e, H_r)$. Where

- The set $\mathbf{N} = \{N_i | i = 1 : n\}$ represents tasks with serial numbers, where n is the total number of tasks.
- The set $\mathbf{W} = \{W_i | i = 1 : n\}$ indicates the size of tasks.
- The set $\mathbf{U} = \{U_{ij} | i = 1 : n, j = 1 : n\}$ represents the communication relationships among tasks, where U_{ij} reflects the communication from task N_i to task N_j. We should note that $U_{ij} \neq U_{ji}$. If there's no communication between the two tasks, then $U_{ij} = 0$.
- $\mathbf{H_t} = \{H_t(i) | i = 1 : n\}$, $\mathbf{H_c} = \{H_c(i) | i = 1 : n\}$, $\mathbf{H_e} = \{H_e(i) | i = 1 : n\}$ and $\mathbf{H_r} = \{H_r(i) | i = 1 : n\}$ represent the lowest time, cost, energy and reliability requirements of tasks respectively.

Besides, let the predecessor tasks set of N_i be $\mathbf{L_i}$, and the successor tasks set be $\mathbf{R_i}$. The node with no predecessor task $\mathbf{L_i} = \emptyset$ is named *source* node, and the node with no successor task $\mathbf{R_i} = \emptyset$ is called sink node. All tasks strictly observe the tasks' priority rules, that is to say, a node can only be started after all output communication data of its predecessor tasks are obtained.

According to the QoS indexes analyzed in the previous sections, the general model of cloud computing can be defined as follow.

Definition 2 The software/hardware cloud services in cloud computing mode can be presented respectively as

$$S : \begin{cases} \text{software service} : \ S_1 = (s, c, r, v) \\ \text{hardware service} : \ S_2 = (s, c, r, v, e, \zeta) \end{cases}$$

$\mathbf{S_1} = \{s_1(i) | i = 1 : n_{s_1}\}$ represents the set of software cloud services, where the number of services is $n_{s_1} = |\mathbf{S_1}|$. $\mathbf{S}_2 = \{s_2(i) | i = 1 : n_{s_2}\}$ represents the set of hardware cloud services, where the number of services is $n_{s_2} = |\mathbf{S}_2|$. Therefore the total number of cloud services is $n_s = n_{s_1} + n_{s_2}$. In the definition, s, c, r, v, e, and ζ represents the execution efficiency, rent cost, reliability, the minimum required speed of VM, energy-consumption and the average control rate of cloud services respectively. All of these attributes stored according to the type and the serial number of services.

Because the performance of VM is decided by the corresponding computing resources, so this chapter just define the formal description of computing resources as follow.

Definition 3 The computing resources in cloud computing mode can be presented as $P = (x, \varphi, \phi, \sigma, f, \lambda)$, where

- $\mathbf{P} = \{P_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ indicates the computing resources with different groups and different serial number, where k is the group number of the whole set, l is the number of computing resources in each group and d is the number of groups.
- $\mathbf{x} = \{x_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ represents the speed of computing resources. It's related to the configuration characteristics of these computers.
- $\Psi = \{\varphi_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ means the bandwidths of computing resources in local networks, and $\Phi = \{\phi_k | k = 1, 2, \ldots, d\}$ be the bandwidths between the switches of various sub-infrastructure groups and cloud centre console.
- $\boldsymbol{\sigma} = \{\sigma_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ represents the average energy-consumption per unit time of these computing resources. According to the analysis above, σ_{kl} is in proportional to x_{kl}.
- $\mathbf{f} = \{f_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ means the failure probability of each computing resource. This factor is changed after each time of task execution.
- $\boldsymbol{\lambda} = \{\lambda_{kl} | k = 1, 2, \ldots, d, l = 1, 2, \ldots, m_k\}$ represents the number of task loads in each computing resource at present. It changed during task execution. If multiple VMs map into one single computing resource, the running speed of the resource will be dramatically declined. For simplified the evaluation, we assume the VMs share the same computing resource with average division.

In the definition of computing resources, the failure recovery time is not defined. Because of the dynamic migration in cloud computing system, we assume the average dynamic migration time (i.e. the recovery time) as a constant $u = Const$.

For two tasks N_i and N_j, if the support VM are v_i and v_j, and the allocated computing resources are P_{kl} and $P_{k'l'}$, the running speed of v_i and v_j can be expressed as $p_i = x_{kl}/\lambda_{kl}$ and $p_j = x_{k'l'}/\lambda_{k'l'}$. If the selected computing resources are in the same group, i.e. $k = k'$, the transmission rate is $q_{ij} = \min(\varphi_{kl}, \varphi_{k'l'})$. If the allocated computing resources are distributed, the transmission rate can be represented as $q_{ij} = \min(\phi_k, \phi_{k'})$. In addition, the energy-consumption of the two VMs are $g_i = \sigma_{kl}/\lambda_{kl}$ and $g_j = \sigma_{k'l'}/\lambda_{k'l'}$. And the rent cost, failure probability and recovery time of VMs are defined the same as the attributes of computing resources.

Corresponding to Fig. 9.3, the DS-CSCR model can be defined as a quadric-tuple $M = (G, S, V, P)$. Based on the above definitions, the decision of DS-CSCR can be made and evaluated with multi objectives of the lowest execution time, energy-consumption and cost and the highest reliability for tasks.

Take the serial tasks as a case, let the number of tasks be n, the type of the selected cloud service for each task N_i is y_i. y_i can be 1 or 2 and represents

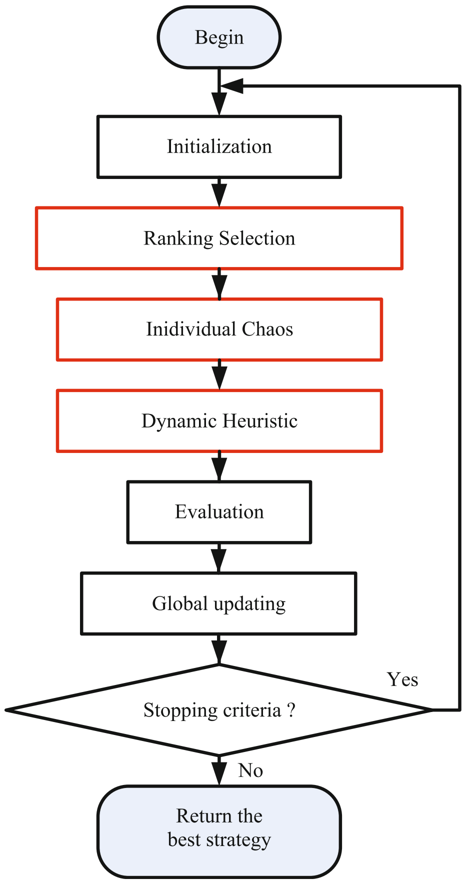


Fig. 9.3 The flowchart of RCO

software and hardware cloud service respectively. So that the serial number of the selected service is $S_{y_i}(i)$. Assume the allocated computing resource for the support VM v_i of each task is $P_{k_i l_i}$. Then the overall optimal objectives and constraints can be calculated as follows.

$$MAX\ Objective\ Function = w_1 \prod_{i=1}^{n} R_i + w_2 / \sum_{i=1}^{n} T_i + w_3 / \sum_{i=1}^{n} C_i + w_4 / \sum_{i=1}^{n} E_i \tag{9.14}$$

The variables in the objective function are calculated according to Table 9.1.

Table 9.1 The calculation of elements in the objective function

Variables	Software services	Hardware services
R_i	$r_{s_1(i)}$	$r_{s_2(i)}$
T_i	$W_i \frac{v_{s_1(i)}\sigma_{k_i l_i}}{x_{k_i l_i} s_{s_1(i)}} + \max\limits_{j\in pred(i)} \frac{U_{ij}}{q_{ij}} + uf_{k_i l_i}$	$W_i \frac{v_{s_2(i)}\lambda_{k_i l_i}(\phi_{k_i} + s_{s_2(i)}\zeta_{s_2(i)})}{x_{k_i l_i} s_{s_2(i)}\phi_{k_i}} + \max\limits_{j\in pred(i)} \frac{U_{ij}}{q_{ij}} + uf_{k_i l_i}$
C_i	$T_i c_{s_1(i)}$	$T_i c_{s_2(i)}$
E_i	$T_i \frac{\sigma_{k_i l_i}}{\lambda_{k_i l_i}}$	$T_i \left(\frac{\sigma_{k_i l_i}}{\lambda_{k_i l_i}} + e_{s_2(i)}\right)$

The main constraints of DS-CSCR are shown as following

$$\forall i \in [1, n] \qquad 0 < \rho_i < 1 \tag{9.15}$$

$$\forall k \in [1, g], l \in [1, m_k] \qquad \sigma_{kl} \geq 0 \tag{9.16}$$

$$\forall i \in [1, n] \qquad T_i < H_t(i),\ C_i < H_c(i),\ E_i < H_e(i),\ R_i < H_r(i) \tag{9.17}$$

The first constraint means that the occupancy rates of VMs in computing resources are no less than 0 and no more than 1, that is to say, one VM can only be allocated in one computing resource with full occupancy at most. The second constraint indicates that the load of computing resources must be no less than 0. When $\sigma_{kl} = 0$, the computing resource is idle. When $0 < \sigma_{kl} < 1$, the computing resource is not fully occupied, the running speed can be hold. However, when $\sigma_{kl} \geq 1$, the tasks need to be executed in queue, the running speed of computing resource will be dramatically decreased. The third constraint represents that each attributes of cloud services and computing resources must satisfy the lowest requirement of tasks.

9.5 Ranking Chaos Algorithm (RCO) for DS-CSCR in Private Cloud

From the above analysis it's clear that the model of DS-CSCR is more complex than the traditional SCOS and OACR. The upper layer cloud services and the underlying computing resources interact with each other. Their complex attributes together directly determine the efficiency of task execution. In large-scale solution space, it's hard to find optimal solution of DS-CSCR by a deterministic algorithm. The general methods for solving these kinds of problems are searching for sub-optimal solutions by intelligent algorithms, such as GA, PSO and ACO and so on. ACO is designed particularly for path optimization. PSO is presented for continuous numerical optimization. GA is more universal but with serious local convergence. In the condition of complex mutual relations among the attributes of the problems with large-scaled irregular solution space, these typical algorithms are quite unsuitable.

Therefore, a new RCO is presented in this chapter for DS-CSCR. The flowchart of this algorithm is shown in Fig. 9.3. It contains three main operators: ranking selection operator, adaptive chaos operator and dynamic heuristic operator. All of them can be executed independently and hybrid arbitrarily. Their initialization (coding scheme), operators and evolutionary strategy for solving DS-CSCR are elaborated as follows.

9.5.1 Initialization

Usually, initialization in intelligent algorithm is very important. It determines the initial location and the coding scheme of population. The initial location ways of population include regular generation and random generation, and so on. For DS-CSCR, the solution space is quite complex, so that the random initialization scheme is selected in this chapter.

Additionally, different coding style has different contribution to algorithm. Coding scheme in intelligent algorithm not only directly reflects the characteristics of the problems, but also affects the performance of the operators. Suitable coding scheme can even improve the searching capability of algorithms. In this chapter, the real number coding scheme is adopted because of its characteristics of simplicity and intuitive.

Specifically, for the above mentioned DS-CSCR model, both service-genebit which represents the selected cloud services and resource-genebit which represents the allocated computing resources are needed to be set. One task corresponds to two genebits. Thus the real number coding is the most intuitive and space-saving scheme for DS-CSCR. When a set of tasks are submitted to cloud system, the system should choose suitable cloud services and computing resources with specific serial numbers at the same time. Assume the length of gene code be twice of the number of tasks, as shown in Fig. 9.4. Each two genebits represent the serial number of the selected service and the allocated computing resource for the corresponding task. It briefly demonstrates the relationship between cloud service and computing resource and makes the optimal process more convenient.

9.5.2 Ranking Selection Operator

In most chaos-based optimizations, chaotic operator is based on the individuals regardless of whether they are good or bad. In this case, the algorithm is easy to trap into bad conditions with large randomly searching range and extremely strong diversity. To obtain better seeds for chaotic random ergodicity, selection before it is needed.

The most commonly used selection operator in GA is roulette wheel selection. With high randomness, bad individuals may be selected more than good ones,

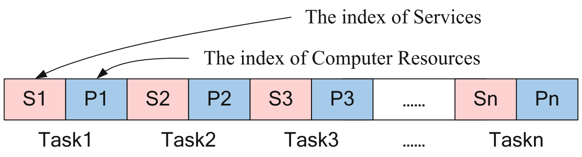


Fig. 9.4 The real number coding scheme for DS-CSCR problem

higher diversity can be achieved in population. But high diversity has been implemented by chaos and what we need before chaos is just a set of good seeds. In this condition, roulette wheel selection becomes unsuitable. To make sure the high quality of good individuals (i.e. seeds), a dynamic ranking selection operator is designed in this section.

Normally, ranking selection means selection according to the descending sort of individual fitness values under a constant proportion. That is to say, the numbers of individuals from best to worst are in arithmetic sequence. Here we adopt quick sort algorithm with the computation complexity O(*n*log*n*). Let $\mathbf{I} = \{I_i | i = 1, 2, N\}$ be the population with N individuals, and I_i in the population be the ith individual. Assume the sorted population to be $\mathbf{I}' = \{I'_i | i = 1, 2, N\}$ with the fitness value $F'_N < F'_{N-1} < \cdots < F'_1$. Define $P_{selection}$ to be the percentage of individuals to be selected on the whole. If $P_{selection} = 1$, then all individuals are selected at least once, if $P_{selection} = 0.5$, then only the first half individuals are selected, the other half individuals would not be selected any more. It represents the selection range in the sorted population. Thus the worst individual to be selected is the Kth individual where $K = NP_{selection}$. Under the selection range, let the number of times that the best individual to be selected as θ_1 and the number of times that the worst individual to be selected as θ_K. Then the difference between the numbers of two adjacent individuals can be calculated as follow.

$$\Delta\theta = \theta_{i-1} - \theta_i = \frac{\theta_1 - \theta_K}{K - 1} \tag{9.18}$$

$$\theta_i = \theta_1 - \Delta\theta(i-1) = \theta_1 - (i-1)\frac{\theta_K - \theta_1}{K-1} \quad \text{where} \quad 1 \le i \le K \tag{9.19}$$

It can be seen that $\sum_{i=1}^{K} \theta_i = N$. Therefore, we can deduce that,

$$\theta_1 + \theta_K = \frac{2N}{K} \tag{9.20}$$

Let $\theta_K = 1$, then

$$1 = \theta_K \le \theta_1 \le \frac{2N}{K} - 1 \tag{9.21}$$

To make the selection adaptively, a function for calculating θ_1 in the ranking selection is defined as follow.

$$\theta_1 = 1 + \left(\frac{2N}{K} - 2\right)\frac{F_{average}}{F_{best}} = 1 + \left(\frac{2N}{K} - 2\right)\frac{F_{average}}{F'_1} \tag{9.22}$$

$$\Delta\theta = \left(\frac{2N}{K} - 2\right)\frac{F_{average}}{F'_1}\frac{1}{(K-1)} = \frac{2(N-2K)F_{average}}{K(K-1)F'_1} \tag{9.23}$$

where $F_{average}$ represents of the average fitness value of the whole population. Thus the much closer $F_{average}$ and F' are, the bigger θ_1 is, the bigger the number of times the better individuals to be selected. Otherwise, the number of times the worse individuals would be bigger and the selection of K individuals becomes more balance. The pseudo-code of this operator is shown below as Algorithm 1.

```
Algorithm 1: Ranking Selection Operator
Ranking_Selection (I)
    Define the selection range according to $P_{selection}$
    Sort $I$ with quick sort algorithm and stored as $\mathbf{I}'$
    Calculate the number of times of $I_1$ to be selected, $\theta_1 = 1 + (2N/K - 2)F_{average}/F'_1$
    Calculate $\Delta\theta = 2(N-2K)F_{average}/K(K-1)F'_1$
    Calculate $\theta_2, \theta_3, \cdots, \theta_K$ for other $K-1$ individuals
    Select $N$ individuals according to $\theta_1, \theta_2, \cdots, \theta_K$ and generate new $I$
```

9.5.3 Individual Chaos Operator

Chaos is a universal non-linear phenomenon. It has the characteristics of strong randomness and internal regularity. With the generation of logistic chaos sequences, it can traverse almost all states in a certain range without duplication and cause great changes in output with rich dynamism. Thus it can improve population diversity in many typical intelligent algorithms and help them to avoid local optimization. Nevertheless, it is non-directional and hard to control.

In general, the searching process of typical chaos-based optimization can be divided into two stages. In the first stage, a bunch of chaotic sequences with certain length are generated by logistic chaos generating function. Then one or more gene-bits of individuals are changed according to the chaotic sequences and a series of new individuals are generated. After the selection of good solution among these new individuals, the second stage will introduce a small disturbance to the local optimum individuals for further exploitation. The iteration will continue until the terminate standards are satisfied.

However, two problems come up to restrain the performance of chaos for large-scale problems with irregular solution spaces. First, small disturbance will not help to exploit in complex and irregular spaces. Besides, the length of chaotic sequence directly decides the time consumption and searching ability of the algorithm.

For higher searching ability, the second problem is that fixed length of chaotic sequences may bring large time consumption in exploration. Thus, we design a new individual chaos operator in which the small disturbance is abandoned and adaptation of chaotic length is introduced for individuals with customization.

Specifically, the length of chaotic sequence for each individual is determined by its current evolutionary state. Let $\mathbf{I} = \{I_i | i = 1, 2, N\}$ be the population with N individuals, and I_i be the ith individual. It includes its gene-bit values $G_i = \{G_i(1), G_i(2), \ldots, G_i(M)\}$ and fitness value F_i, where M represents the length of gene code (i.e. twice of the number of tasks). The specific pseudo-code is shown as Algorithm 2.

```
Algorithm 2: Individual Chaos Operator
    Individual_Chaos (I)
    For ( $i = 1$ to $N$ )
        $L_{chaos} = A + B(F_{best} - F_i) / (F_{best} - F_{worst} + 1)$
        Generate $X_1[L_{chaos}]$, $X_2[L_{chaos}] \in [0,1]$ by using Logistic chaos function
        For ( $j = 1$ to $L_{chaos}$ )
            Map $X_1(j)$ as genebit serial number $k \in [1, M]$
            If ( $k$ corresponds to service-bit)
                Map $X_2(j)$ as genebit value $v \in [1, n_s]$
            Else
                Map $X_2(j)$ as genebit value $v \in [1, n_p]$
            End if
            Generate j new temporary individuals $\{r(1), r(2), \cdots, r(j)\}$ by replacing the value of $G_i(k)$ with v
            Choose the best individual $r_{best}$ from the temporary individuals
            If ( $F_{r_{best}} > F_i$ )
                $I_i = r_{best}$
            Else
                If ( $\exp((F_{r_{best}} - F_i) / t^o) > \gamma$ )
                    Replace $I_i$ with $r_{best}$
                End if
            End if
        End for
        $t^o = Dt^o$
    End for
```

Go in detail, the evolutionary state of the ith individual is defined as Q_i:

$$Q_i = \frac{F_{best} - F_i}{F_{best} - F_{worst}} \tag{9.24}$$

$$L_{chaos} = A + (B - A)Q \tag{9.25}$$

where A and B is the lower bound and upper bound of L_{chaos}, respectively. F_{best} and F_{worst} represent the serial numbers of the individual with the best and the worst fitness value. To be exact, the closer the average fitness value to the best fitness value in population, the better the evolutionary state is, and the shorter the length of chaotic sequence L_{chaos} is, so that the smaller the searching range is. Otherwise, the closer the average fitness value to the worst fitness value in population, the smaller the searching range is.

With the initialized definition of the length of chaotic sequences L_{chaos}, the operator generates two chaotic sequences $X_1[L_{chaos}]$, $X_2[L_{chaos}]$ for each individual $I_i(i = 1, 2, \ldots, N)$ by Logistic mapping chaotic function, as shown in Eq. (9.26).

$$z_{l+1} = \mu z_l(1 - z_l) \tag{9.26}$$

where $\mu = 4$ according to general chaotic strategy. Then X_1 and X_2 are mapped to the serial number k and the value v of gene-bits respectively. If $k \in [1, M]$ corresponds to service gene-bit, we should map X_2 to relative service number and store it in v. Or we should map X_2 to relative computing resource number and store it. In the pseudo-code, n_s and n_p represents the number of cloud services and the number of computing resources respectively. After the chaotic mapping step, new neighbor solutions $\{r(1), r(2), \ldots, r(j)\}$ can be generated by changing the value of $G_i(j)$ into v. Further, choose the individual r_{best} with the best fitness value and accept it as new individual with probability $P_{annealing} = \exp(\frac{F_{r_{best}} - F_i}{t^0})$, where t^0 is the annealing temperature and the initial value is 100. In the algorithm, the rate of t^0 drop D is set to be 0.95 to gradually narrow down the accept probability. On the whole, in the individual chaos operator, searching is carried out with the adaptive changing of the length of chaotic sequences for each individual according to its evolutionary state Q_i. Chaos states can finally be controlled by population state.

9.5.4 Dynamic Heuristic Operator

For further improving the searching direction in chaos optimization, dynamic heuristic is introduced in this algorithm after ranking selection and adaptive individual chaos. The principle of this operator is dynamically guiding the algorithm for local search with right direction by using some priori knowledge of the problem.

To be specific, for each individual, the operator randomly chooses a gene-bit, traverses part of the available values for the single gene-bit and dynamically calculates the heuristic of each value, then picks the most suitable value with the highest heuristic and generates new individual. It's quite like the mechanism of pheromone in ACO. Compared with the pheromone, dynamic heuristic here does not contain empirical information. It uses just the priori knowledge which is dynamically calculated according to the states or the gene-bit values of individuals in each generation. Define the traverse range for one gene-bit to be hn, where $h \in [0, 1]$, and n can be n_s or n_p. The specific pseudo-code is shown as follow.

```
Algorithm 3: Dynamic Heuristic Operator
Dynamic_Heuristic (I)
    For ($i = 1$ to $N$)
        r = $I_i$
        Randomly choose a genebit $k \in [1:M]$
        If (p corresponds to service-bit)
            Randomly choose $hn_s$ values from 1 to $n_s$
            Choose the service $s_i$ with the highest heuristic $Y_s = \max_j y_s(j)$ $(j \in [1, hn_s])$
            $G_r(k) = s_i$
        Else
            Randomly choose $hn_p$ values from 1 to $n_s$
            Choose the computing resource $p_i$ with the highest heuristic $Y_p = \max_j y_p(j)$ $(j \in [1, hn_p])$
            $G_r(k) = p_i$
        End if
        Accept r with simulation annealing probability
    End for
```

During the process, h can be set as 0.3. And p represents the randomly selected gene-bit for each individual. If p corresponds to service-genebit, search available services and calculate dynamic heuristic of each available service $y_s(j)(j \in [1, hn_s])$ by service heuristic function. Then choose the service s_i with the highest heuristic Y_s to replace the original value of *k*th gene-bit. If k corresponds to computing resource gene-bit, search available computing resources and calculate dynamic heuristic of each computing resource $y_p(j)(j \in [1, hn_p])$ by computing resource heuristic function. Then choose the computing resource p_i with the highest heuristic Y_p to replace the value of *k*th gene-bit. After these steps, a new individual r is generated for each individual. Then replace I_i with r in simulation annealing probability as well as in adaptive chaos operator $P_{annealing} = \exp(\frac{F_r - F_i}{t^0})$.

In this process, how to design service heuristic function and computing resource heuristic function is very important. Unsuitable heuristic function can cause wrong searching direction in algorithm and easily lead the algorithm to serious premature convergence. In this chapter, the service and computing resource heuristic function are simply designed as follow.

$$y_s(j) = \begin{cases} \alpha_1 \frac{s_{s_j}}{v_{s_j}} + \alpha_2 \frac{1}{r_{s_j}} + \alpha_3 \frac{1}{e_{s_j}} + \alpha_4 \frac{1}{c_{s_j}} + \alpha_5 r_{s_j}, & \text{if } s_j \in S_1 \\ \alpha_1 \frac{s_{s_j}}{v_{s_j}} + \alpha_2 \frac{1}{c_{s_j}} + \alpha_3 r_{s_j}, & \text{if } s_j \in S_2 \end{cases} \tag{9.27}$$

$$y_p(j) = \alpha_1 \frac{x(j)}{} \lambda(j) + \alpha_2 \max(\varphi(j), \phi(j)) + \alpha_3 \frac{1}{\sigma(j)} + \alpha_4 \frac{1}{f(j)} \tag{9.28}$$

where α_1, α_2, α_3, α_4, α_5 represent the weights of services/computing resources attributes respectively which corresponds to the weights setting in the objective function. Through the adjustment of weights, small range of local search in a single gene-bit could be guided in the algorithm according to the dynamic heuristics.

9.5.5 The Complexity of the Proposed Algorithm

Generally, the time complexity of the intelligent algorithms is dynamically varied with different problems. Let n be the scale of the population, m be the size of tasks, s be the number of the available cloud services for each task and p be the total scale of computing resources. The algorithms' complexities in each generation are shown in Table 9.2.

In GA, typical roulette wheel selection needs n times roulette operations to generating new population. Each roulette operation contains at least 1 and at most n times comparison according to the relative fitness values of individuals. Thus the average complexity of selection operator is $O(n^2)$. In RCO, the complexity of ranking individuals in selection is $O(n\log n)$ (with quick sort method) and the selection step according to selective pressure needs at most n times. Thus the complexity of ranking selection is $O(n\log n)$.

Besides, crossover and mutation operation in GA are just executed once for each individual. The complexity are both at least $O(n)$ and at most $O(mn)$. In RCO, chaotic sequences with constant length are generated for each individual. From the pseudo-code it can be seen that the complexity of individual chaos operator is $O(nL_{chaos}) = O(n)$. Because the adaptation of chaotic length is in a limited area, the complexity of chaos operator is also $O(n)$. It is lower than crossover operator. In addition, dynamic heuristic randomly chooses a gene-bit for each individual, traverse part of available value of this gene-bit with heuristics. If all of the selected gene-bits are service-bit, then the complexity is $O(n_s)$, else if all of the selected gene-bits are computing resource-bit, then the complexity is $O(n_p)$. Thus the average complexity of dynamic heuristic operator is $O(n(s + p)/2) = O(n\max(p, s))$.

In theory, if $s \to \infty$ and $p \to \infty$, the complexity of RCO is a little higher than GA. But in the condition of $n \to \infty$ and $m \to \infty$, the complexity of RCO is lower than GA.

9.6 Experiments and Discussions

Based on the case "the design and NC (Numerical Control) machining process of a complex surface part" mentioned before, three typical DAG: two DAGs as shown in Fig. 9.5 [21] and the "j30" DAG of Resource-Constrained Project Scheduling Problem (RCPSP) in PSPLIB [31], are used as three task graphs in our experiments. In practical application of private cloud in manufacturing conglomerate or large-scale manufacturing service providers, a composite project contains multiple complex surface parts' machining. Thus a composite project can be divided into far more than 5 tasks. Those tasks have several functional and non-functional

Table 9.2 The complexity of the operators in GA and RCO

Algorithms	The time complexities of operators			$n \to \infty$	$m \to \infty$	$s \to \infty$	$p \to \infty$
GA	Roulette wheel Selection	Crossover	Mutation	O(n^2)	O(m)	O(1)	O(1)
	O(n^2)	O(nm)	O(nm)				
RCO	Ranking Selection	Individual Chaos	Dynamic Heuristic	O ($n$log$n$)	O(1)	O(s)	O(p)
	O($n$log$n$)	O(n)	O(n*(max (p, s)))				

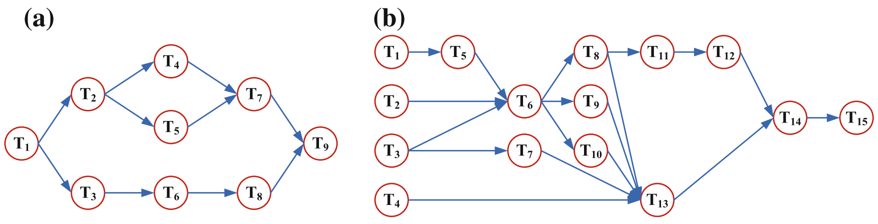


Fig. 9.5 Two typical DAG with 9 and 15 tasks respectively. **a** DAG1 **b** DAG2

requirements for cloud services. Some of them need hardware cloud services, some need software cloud services. In order to evaluate the performance of dual-scheduling optimization compared with the traditional two-level decision, we use basic real-coding GA uniformly to simulate the decision process in theory. At the OACR step, each gene-bit represents the selected computing resource number for the above selected service. The lengths of gene-bits at the two steps are equal. Furthermore, At the SCOS step, we consider only the properties of cloud services and then set the objective function as following according to Eq. (9.14) and [32].

At the OACR step, we also use the objective function in Eq. (9.14) with the fixed properties of cloud services. In Eq. (9.18), let the weight to be $w_1 = w_2 = w_3 = w_4 = 100n$.

For simplifying the optimization process, we set that each task in a composite project has the same number of available cloud services. In the three cases, 3 composite scales of DS-CSCR are tested, as shown in Table 9.3. And in each scales, computing resources are equally divided into 5 distributed groups.

Assume the available number of cloud services for each task is s and the available number of computing resources is p, then the size of solution space is $s^n p^n$. From Scale 1 to Scale 9, it's range from $10^9 \times 20^9$ to $50^9 \times 100^9$. Most deterministic algorithms can't handle these situations due to composite exposition.

Table 9.3 The selected 4 composite scales of cloud services and computing resources

	Scale 1	Scale 2	Scale 3	Scale 4	Scale 5	Scale 6	Scale 7	Scale 8	Scale 9
Number of tasks	9	9	9	15	15	15	30	30	30
Number of available cloud services	10	20	50	10	20	50	10	20	50
Number of available computing resources	20	50	100	20	50	100	20	50	100

Table 9.4 The property ranges of cloud services and computing resources

	s	c	r	v	e	ζ
Software service	[1, 10]	[1, 10]	(0, 1)	[1, 10]		
Hardware service	[1, 10]	[1, 10]	(0, 1)	[1, 10]	[1, 10]	[0, 1]
.	x	φ	ϕ	σ	f	λ
computing resource	[1, 10]	[1, 10]	[1, 5]	[1, 10]	(0, 1)	0

Moreover, because of the restriction of experimental environment, we set the ranges of properties of cloud services and computing resources as shown in Table 9.4.

For theoretical analysis, all the values are randomly generated with normalization and idealization and stored in a *txt* file. In order to distinguish the bandwidths inter-group and intra-group, the range of φ is set to be slightly larger than ϕ. Initially, task load of all computing resources are 0.

Based on DS-CSCR with 9 scales, standard GA, chaos GA (CGA), typical chaos optimization (CO), chaos optimization with only individual chaos operator designed in this chapter (RCO^{-2}), chaos optimization with ranking selection and individual chaos operator (RCO^{-1}) and chaos optimization with the addition of dynamic heuristics (RCO) are compared together for further testing the performance of the above designed algorithm. In the experiments, the classical roulette wheel selection operator, multiple-point crossover operator and single-point mutation operator are adopted in GA. And the crossover and mutation probabilities are set to be the typical values, i.e. 0.8 and 0.15, respectively. In chaos strategy of CGA and CO, the length of chaotic sequences is set as a constant 10. For a fairer comparison, in the new RCO, let $A = 5$ and $B = 15$ to make sure the same level of chaotic operation. Besides, the iterations of all experiments are set as 2000 uniformly and population sizes are all 20. Due to the randomness of intelligent algorithms, a total of 100 runs of each experiment are conducted and the average fitness value of the best solutions throughout the run is recorded.

9.6.1 Performance of DS-CSCR Compared with Traditional Two-Level Scheduling

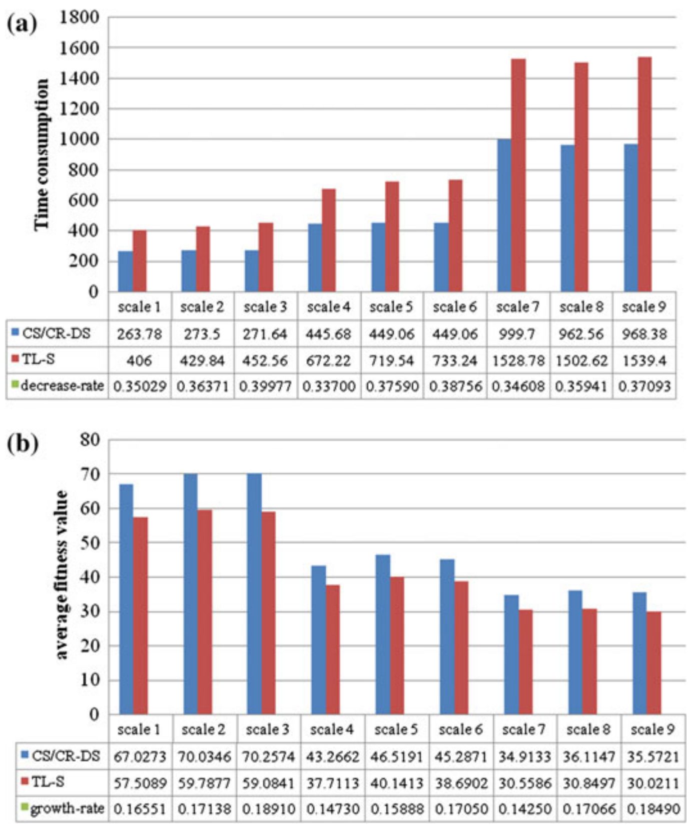
Let TL-S to be the abbreviation of traditional Two-Level Scheduling, we compared it with new DS-CSCR in the above 9 scales of solution space. Figure 9.6 shows the testing results from the perspectives of time consumption and solution quality respectively.

Firstly, we define the *decrease-rate* to be $\tau_d = \frac{T_{TL-S} - T_{CS/CR-DS}}{T_{TL-S}}$ in Fig. 9.6a. As we have analyzed previously, the time consumption of SCOS and OACR in traditional TL-S are reduced by about 35–40 % in DS-CSCR. Although the length of individual and the size of solution space are only half that of DS-CSCR. Traditional TL-S takes almost twice the time of DS-CSCR. For each task graph, as the numbers of cloud services and computing resources are enlarged, the *decrease-rate* increases gradually. Thus it can be seen, with the same algorithm (no matter deterministic algorithm or intelligent algorithm), TL-S is more and more time-consuming with the increase of solution space while DS-CSCR always maintains a relatively low level of time consumption. It proved that, with the same algorithm, no matter using determistic or intelligent, two level decision is cumbersome.

Secondly, from the angle of solution quality in Fig. 9.6b, we define the *growth-rate* to be $\tau_g = \frac{\bar{F}_{CS/CR-DS} - \bar{F}_{TL-S}}{\bar{F}_{TL-S}}$, where $\bar{F}_{CS/CR-DS}$ and $\bar{F}_{TL-S}$ represent the average result of the best fitness value in experiments. It increases along with the expansion of solution spaces in each kinds of task graph. For all of scales, the total level of quality in TL-S is improved by about 14–19 % in DS-CSCR. In theory, the service properties are static in the second step of TL-S. With the splitting of SCOS and OACR under unified console, the mutual relations between cloud service and the underlying computing resources are ignored. This tells us the conclusion that in private cloud, the underlying support infrastructure must be considered in the process of SCOS. With fast development of dynamic network, service with dynamic deployment are more and more common. SCOS with the consideration of QoS only are inpractical for many of the advanced system in large SaaS mode.

9.6.2 Searching Capability of RCO for Solving DS-CSCR

For addressing DS-CSCR more efficiently, we designed RCO especially aiming at the situation of large-scale solution space. Figure 9.7 recorded the average fitness value of the best solution during 2000 generations in 100 runs for 9 scales of DS-CSCR (i.e. the average evolutionary trend of the 6 algorithms in 100 runs). Figure 9.8 shows the fitness value of the best solution, the worst solution and the average result in 100 runs for 9 scales of DS-CSCR. Note that the fitness value is the assessment value of each individual according to the objective function. So from the perspective of searching capability, the sort of the six algorithms from



	scale 1	scale 2	scale 3	scale 4	scale 5	scale 6	scale 7	scale 8	scale 9
🟦 CS/CR-DS	263.78	273.5	271.64	445.68	449.06	449.06	999.7	962.56	968.38
🟥 TL-S	406	429.84	452.56	672.22	719.54	733.24	1528.78	1502.62	1539.4
🟩 decrease-rate	0.35029	0.36371	0.39977	0.33700	0.37590	0.38756	0.34608	0.35941	0.37093

	scale 1	scale 2	scale 3	scale 4	scale 5	scale 6	scale 7	scale 8	scale 9
🟦 CS/CR-DS	67.0273	70.0346	70.2574	43.2662	46.5191	45.2871	34.9133	36.1147	35.5721
🟥 TL-S	57.5089	59.7877	59.0841	37.7113	40.1413	38.6902	30.5586	30.8497	30.0211
🟩 growth-rate	0.16551	0.17138	0.18910	0.14730	0.15888	0.17050	0.14250	0.17066	0.18490

Fig. 9.6 Comparison of DS-CSCR and TL-S based on GA. **a** The average solution of DS-CSCR and TL-S based on GA in 9 scales **b** The average solution of DS-CSCR and TL-S based on GA in 9 scales

bad to good is: GA<CGA<CO<RCO^{-2}<RCO^{-1}<RCO. The step-by-step improvement from the design of individual chaos operator to the introduction of ranking selection and dynamic heuristic operators can be clearly observed.

On the basis of GA, the average fitness value of the best solutions of CGA is about 30 % higher than GA. At this moment, the average best fitness value of CO with single chaos optimal operator is about 1.5 times higher than GA. From here we can come to the conclusion that the basic operators of GA constrained the searching ability of chaos optimal operator in CGA to some degree. Simple chaos optimization can get much better solution than the traditional GA and improved CGA. Furthermore, the adaptive strategy adapts chaotic sequences according to the state of the whole population. When the population is in a good state, the adaptive strategy will reduce the chaotic sequences, so as to reduce the complexity of the algorithm. Compared with CO, the average best fitness value

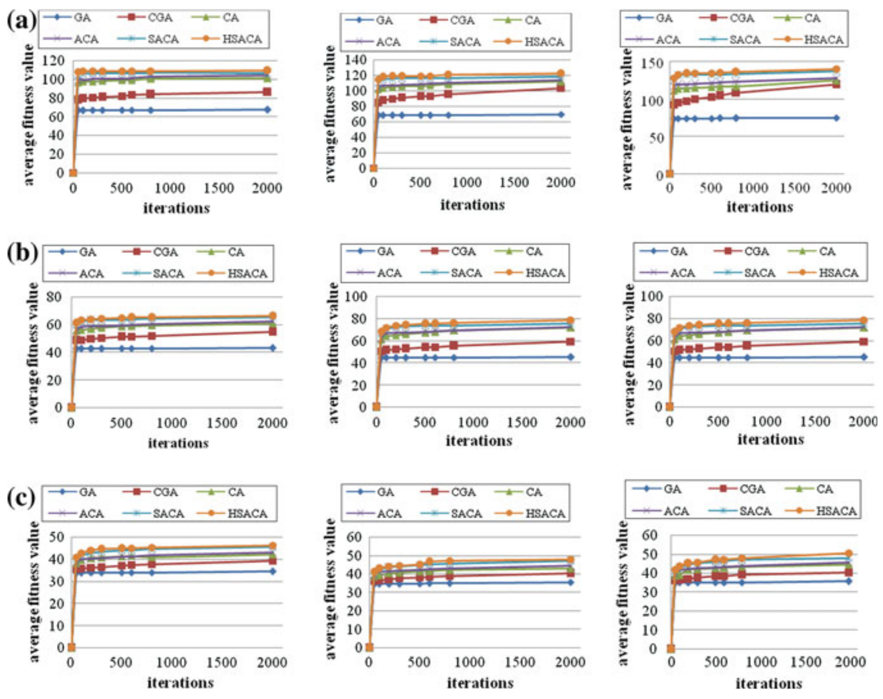
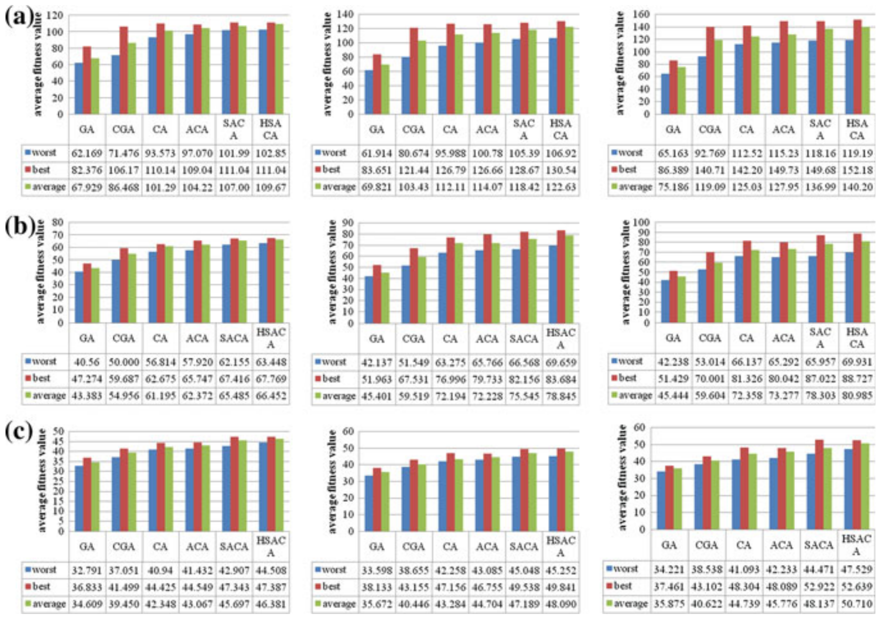


Fig. 9.7 The average evolutionary trend of the 6 algorithms in 100 runs for 9 scales of DS-CSCR. **a** DAG1 with 9 tasks in the scale 1, 2 and 3 **b** DAG2 with 15 tasks in the scale 4, 5 and 6 **c** DAG3 with 30 tasks in the scale 7, 8 and 9

of RCO^{-2} in the 9 scales of DS-CSCR has been raised by about 3 %. Afterwards, ranking selection was put in the front of RCO^{-2}. With the collaboration of selection and chaos, the average best fitness value of RCO^{-1} is improved again. Hence, it can be learned that the operation and collaboration of individual chaos operator and the "the survival of the fittest" ranking selection strategy can not only reduce the complexity of algorithm, but also improve the searching capability remarkably. Because the effect of mutation is similar to chaos operator, it may conclude that the crossover operator in GA mainly restrained the capability of chaos strategy in CGA. Based on the improved RCO^{-1}, for guiding chaos optimization further, dynamic heuristic operator was introduced at last. From Figs. 9.7 and 9.8 we can see that the new RCO performs better than RCO^{-1} with the guidance of heuristics. On the whole, the average best fitness value of RCO in 100 runs is about 2 times higher than GA. The overall improvements are extremely considerable.



(a)	GA	CGA	CA	ACA	SAC A	HSA CA
🟦 worst	62.169	71.476	93.573	97.070	101.99	102.85
🟥 best	82.376	106.17	110.14	109.04	111.04	111.04
🟩 average	67.929	86.468	101.29	104.22	107.00	109.67

	GA	CGA	CA	ACA	SAC A	HSA CA
🟦 worst	61.914	80.674	95.988	100.78	105.39	106.92
🟥 best	83.651	121.44	126.79	126.66	128.67	130.54
🟩 average	69.821	103.43	112.11	114.07	118.42	122.63

	GA	CGA	CA	ACA	SAC A	HSA CA
🟦 worst	65.163	92.769	112.52	115.23	118.16	119.19
🟥 best	86.389	140.71	142.20	149.73	149.68	152.18
🟩 average	75.186	119.09	125.03	127.95	136.99	140.20

(b)	GA	CGA	CA	ACA	SACA	HSAC A
🟦 worst	40.56	50.000	56.814	57.920	62.155	63.448
🟥 best	47.274	59.687	62.675	65.747	67.416	67.769
🟩 average	43.383	54.956	61.195	62.372	65.485	66.452

	GA	CGA	CA	ACA	SACA	HSAC A
🟦 worst	42.137	51.549	63.275	65.766	66.568	69.659
🟥 best	51.963	67.531	76.996	79.733	82.156	83.684
🟩 average	45.401	59.519	72.194	72.228	75.545	78.845

	GA	CGA	CA	ACA	SAC A	HSA CA
🟦 worst	42.238	53.014	66.137	65.292	65.957	69.931
🟥 best	51.429	70.001	81.326	80.042	87.022	88.727
🟩 average	45.444	59.604	72.358	73.277	78.303	80.985

(c)	GA	CGA	CA	ACA	SACA	HSAC A
🟦 worst	32.791	37.051	40.94	41.432	42.907	44.508
🟥 best	36.833	41.499	44.425	44.549	47.343	47.387
🟩 average	34.609	39.450	42.348	43.067	45.697	46.381

	GA	CGA	CA	ACA	SACA	HSAC A
🟦 worst	33.598	38.655	42.258	43.085	45.048	45.252
🟥 best	38.133	43.155	47.156	46.755	49.538	49.841
🟩 average	35.672	40.446	43.284	44.704	47.189	48.090

	GA	CGA	CA	ACA	SACA	HSAC A
🟦 worst	34.221	38.538	41.093	42.233	44.471	47.529
🟥 best	37.461	43.102	48.304	48.089	52.922	52.639
🟩 average	35.875	40.622	44.739	45.776	48.137	50.710

Fig. 9.8 The statistical results of the 6 algorithms in 100 runs for 9 scales of DS-CSCR. **a** DAG1 with 9 tasks in the scale 1, 2 and 3 **b** DAG2 with 15 tasks in the scale 4, 5 and 6 **c** DAG3 with 30 tasks in the scale 7, 8 and 9

9.6.3 Time Consumption and Stability of RCO for Solving DS-CSCR

Next, based on the above mentioned 9 scales with 3 kinds of task graphs (Table 9.3), the time efficiency and stability of the 6 algorithms are discussed below. Note that the time consumption are tested in millisecond (ms) and the stability is measured by the standard deviation of the average fitness values in 100 runs.

Figure 9.9a shows the average time-consumption of the 6 algorithms in 9 scales with 100 runs. The step-by-step improvement from CO to RCO compared with GA and CGA, the variation trends of time in all scales are the same. In CGA, there are four operators (selection, crossover, mutation and chaos), with lower searching capability, its time-consumption is the highest in these 6 algorithms. After wiping out the three operators of GA, the times of CO are just lower than CGA. It is clear that the most time-consuming operator in CGA is chaos operator. Only narrowing down the chaotic traverse range can reduce the total execution time of algorithm. Along with the decrease of chaotic sequences, the searching ability of algorithm will be reduced, too. Therefore, in order to reduce the time complexity of algorithm with the maintaining of the searching ability, individual chaos operator customized for individuals is designed in this chapter. Experiments in RCO^{-2}

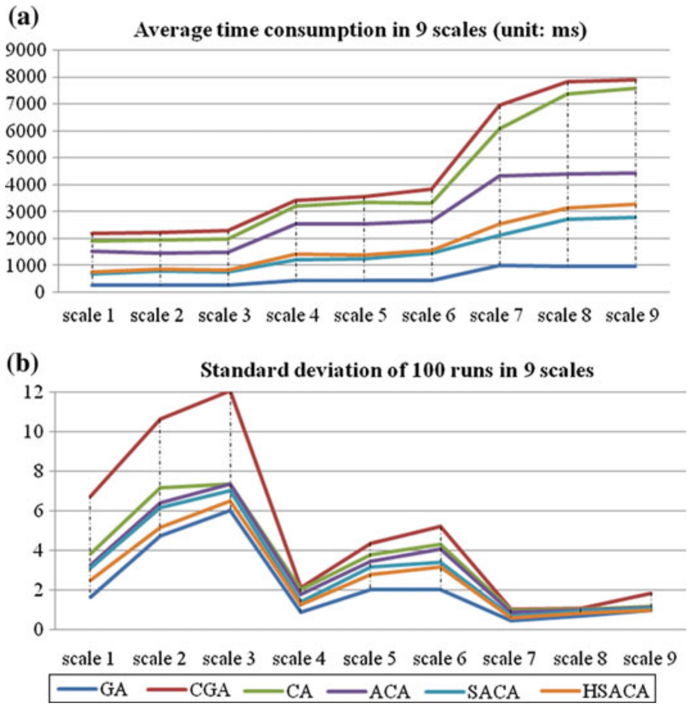


Fig. 9.9 The average time-consumption and standard deviation of the 6 algorithms in 100 runs. **a** Average time consumption of 6 algorithms **b** Standard deviation of 6 algorithms

show that the time-consuming is effectively reduced by about 20 % based on CO with the improvement of searching ability. Especially in scale 7, 8 and 9 with very large solution spaces, time-consuming of chaotic operations are sharply reduced.

Moreover, the introduction of ranking selection not only improved the searching capability of RCO^{-2}, but also reduced the time. The reason is that, based on ranking selection, the difference between the best fitness value and the average fitness value in the population is shortened, the population can always be adapted to a better state with “the survival of the fittest” strategy, then the chaotic sequences are shortened accordingly. With shorter chaotic sequences, the population can be guided to better areas based on fitter individuals and then find better solutions more quickly. In terms of the time measuring, the prominent performance of the collaborative operation of ranking selection and individual chaos operator has been verified again as RCO^{-1}. At the next step, the introduction of dynamic heuristic operator increase the time slightly based on RCO^{-1}, but the new complete RCO is much faster than RCO^{-2}, CGA and CO as a whole.

From the perspective of stability, as shown in Fig. 9.9b, the six algorithms in the 9 problem scales changed irregularly. But from the 9 scales of tests, we can obtain the sort of stability of the six algorithms from bad to good is:

CGA<CO<RCO^{-2}<RCO^{-1}<RCO<GA. Traditional GA is the most stable while the stability of CGA is the worst. With the adaptive improvement, RCO^{-2} is more stable than CO. That is because in large-scale solution space, chaotic sequences are generated based on no matter good or bad individuals, the population is easy to be lead to bad areas during searching and the states of population in each generations are not stable any more. After the introduction of ranking selection, the stability of the algorithm has greatly improved. Each time of selection in iteration maintained the population state and reduced the chaotic sequences, so that the population can always be evolved based on fitter individuals with higher stability. Besides, the design of dynamic heuristic operator with the priori knowledge of DS-CSCR can always guide the population into better areas during evolution and then improve the stability further.

Thus it can be seen that the new designed RCO possesses plenty of advantages in searching capability, time-consumption and stability for addressing DS-CSCR no matter with large or small scales solution spaces in private cloud.

9.7 Summary

Service composition optimal selection (SCOS) and optimal allocation of computing resource (OACR) are both very critical in cloud system. Current works found that the two steps decision of SCOS and OACR in private cloud are quite cumbersome and the mutual relations between cloud services and underlying computing resources are always ignored. Thus this chapter deeply analyzed the characteristics of these two problems and their interactions. Based on this, the idea of one-time decision of SCOS and OACR was presented accordingly. To sum up, the primary works of this chapter can be concluded as follows.

(1) New DS-CSCR model was presented in private cloud for high efficient one-time decision. Properties of software/hardware cloud services, VMs and computing resources are deeply analyzed. The formulation of DS-CSCR was clarified according to the aim of high efficient and low cost resource sharing.
(2) For addressing the complex dual scheduling problem (DS-CSCR), a new intelligent algorithm—RCO was presented. Individual chaos operator was designed as the backbone operator of the algorithm. Then a new adaptive ranking selection was introduced for control the state of population in iteration. Moreover, dynamic heuristics were also defined and introduced to guide the chaos optimization. RCO with these three operators showed remarkable performances in terms of searching ability, time complexity and stability in solving the DS-CSCR problem in such private cloud compared with other algorithms.

References

1. Laili YJ, Tao F, Zhang L, Cheng Y, Luo Y, Sarker BR (2013) A ranking chaos algorithm for dual scheduling of cloud service and computing resource in private cloud. Comput Ind 64(4):448–463
2. Boss G, Malladi P, Quan D, Legregni L, Hall H (2007) Cloud computing. IBM White Paper, 2007. http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2009) Above the clouds: a berkeley view of cloud computing. University of California, Berkeley
4. Xia TZ, Li Z, Yu NH (2009) Research on cloud computing based on deep analysis to typical platforms. Lect Notes Comput Sci 5931:601–608
5. Xu X (2012) From cloud computing to cloud manufacturing. Robot Comput Integr Manuf 28(1):75–86
6. Wu D, Thames L, Rosen D, Schaefer D (2012) Towards a cloud-based design and manufacturing paradigm: looking backward, looking forward. In: Proceedings of the ASME 2012 international design engineering technical conference and computers and information in engineering conference, Chicago
7. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2009) A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 39(1):50–55
8. Li BH, Zhang L, Wang SL, Tao F, Cao JW, Jiang XD, Song X, Chai D (2010) Cloud manufacturing: a new service-oriented networked manufacturing model. Comput Integr Manuf Syst 16(1):1–16
9. Nick JM, Cohen D, Kaliski BS (2010) Key enabling technologies for virtual private clouds. Handb Cloud Comput 1:47–63
10. Tan W, Fan YS, Zhou MC (2010) Data-driven service composition in enterprise SOA solution: a petri net approach. IEEE Trans Autom Sci Eng 7(3):686–694
11. Tao F, Hu YF, Zhao D, Zhou ZD, Zhang HJ, Lei ZZ (2009a) Study on manufacturing grid resource service QoS modeling and evaluation. Int J Adv Manuf Technol 41 (9-10):1034–1042
12. Tao F, Hu YF, Zhou ZD (2009b) Application and modeling of resource service trust-QoS evaluation in manufacturing grid system. Int J Prod Res 47(6):1521–1550
13. Tao F, Zhao D, Hu YF, Zhou ZD (2010) Correlation-aware resource service composition and optimal-selection in manufacturing grid. Eur J Oper Res 201(1):129–143
14. Fujii K, Suda T (2005) Semantics-based dynamic service composition. IEEE J Sel Areas Commun 23(12):2361–2372
15. Ferrer AJ, Hernandez F, Tordsson J, Elmroth E, Ali-Eldin A, Zsigri C, Sirvent R, Guitart J, Djemame RM, Ziegler W, Dimitrakos T, Nair SK, Kousiouris G, Konstanteli K, Varvarigou T, Hudzia B, Kipp A, Wesner S, Corrales M, Forgo N, Sharif T, Sheridan C (2012) OPTIMIS: a holistic approach to cloud service provisioning. Future Gener Comput Syst 28(1):66–77
16. Mika M, Waligora G, Weglarz J (2011) Modeling and solving grid resource allocation problem with network resources for workflow applications. J Sched 14(3):291–306
17. Tordsson J, Montero RS, Moreno-Vozmediano R, Liorente IM (2012) Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener Comput Syst 28(2):358–367
18. Endo PT, Palhares AVD, Pereira NN, Goncalves GE (2011) Resource allocation for distributed cloud: concepts and research challenges. IEEE Netw 25(4):42–46
19. Ma YB, Jang SH, Lee JS (2011) QoS and ontology-based resource management in cloud computing environment. Inf Int Interdisc J 14(11):3707–3715
20. Xiong PC, Chi Y, Zhu SH, Moon HJ, Pu C, Hacigumus H (2011) Intelligent management of virtualized resources for database systems in cloud environment. In: Proceedings of the 27th IEEE international conference on data engineering

21. Zhang YH, Li YH, Zheng WM (2011) Automatic software deployment using user-level virtualization for cloud-computing. Future Gener Comput Syst 29(1):323–329
22. Ghanbari H, Simmons B, Litoiu M, Iszlai G (2012) Feedback-based optimization of a private cloud. Future Gener Comput Syst 28(1):104–111
23. Laili YJ, Tao F, Zhang L, Sarker BR (2012) A study of optimal allocation of computing resources in cloud manufacturing systems. Int J Adv Manuf Technol 63(5–8):671–690
24. Nathani A, Chaudhary S, Somani G (2012) Policy based resource allocation in IaaS cloud. Future Gener Comput Syst 28(1):94–103
25. Ma Y, Zhang CW (2008) Quick convergence of genetic algorithm for QoS-driven web service selection. Comput Netw 52(5):1093–1104
26. Yin PY, Wang JY (2008) Optimal multiple-objective resource allocation using hybrid particle swarm optimization and adaptive resource bounds technique. J Comput Appl Math 216(1):73–86
27. Wada H, Suzuki J, Yamano Y, Oba K (2011) Evolutionary deployment optimization for service-oriented clouds. Softw Pract Exp 41(5):469–493
28. Tao F, Zhang L, Venkatesh VC, Luo YL, Cheng Y (2011) Cloud manufacturing: a computing and service-oriented manufacturing model. Proc Inst Mech Eng Part B J Eng Manuf 225(10): 1969–1976
29. Schaefer D, Thames L, Wellman RD, Wu D (2012) Distributed collaborative design and manufacture in the cloud–motivation, infrastructure and education. In: Proceedings of the annual conference and exposition (ASEE), Texas
30. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener Comput Syst 28(5):755–768
31. Kolisch R, Sprecher A (1997) PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. Eur J Oper Res 96(1):205–216
32. Tao F, Zhao DM, Hu YF, Zhou ZD (2008) Resource service composition and its optimal-selection based on swarm optimization in manufacturing grid system. IEEE Trans Ind Inf 4(4):315–327