Int J Adv Manuf Technol (2012) 63:671–690
DOI 10.1007/s00170-012-3939-0

ORIGINAL ARTICLE

A study of optimal allocation of computing resources in cloud manufacturing systems

Yuanjun Laili · Fei Tao · Lin Zhang · Bhaba R. Sarker

Received: 16 August 2011 / Accepted: 16 January 2012 / Published online: 15 February 2012

Abstract As a new advanced service-oriented networked manufacturing model, cloud manufacturing (CMfg) has been proposed recently. The optimal allocation of computing resources (OACR) is a core part for implementing CMfg. High heterogeneity, high dynamism, and virtualization make the OACR problem more complex than the traditional scheduling problems in grid system or cloud computing system. In this paper, a new comprehensive model for OACR is proposed in the CMfg system. In this model, all main computation, communication, and reliability constraints in the special circumstances are considered. To solve the OACR problem, a new improved niche immune algorithm was presented. Associated with the niche strategy, new heuristics are designed flexibly based on the characteristics of the problem and pheromone is added for adaptive searching. Experiments demonstrate the effectiveness of the designed heuristic information and show NIA's high performances for addressing the OACR problem compared with other intelligent algorithms.

Keywords Optimal allocation · Computing resources · Cloud manufacturing (CMfg) · Intelligent algorithms

Y. Laili · F. Tao (✉) · L. Zhang
School of Automation Science and Electrical Engineering,
Beihang University,
Beijing 100191, China
e-mail: ftao@buaa.edu.cn

Y. Laili
e-mail: llyj0721@gmail.com

L. Zhang
e-mail: johnlin999@gmail.com

B. R. Sarker
Department of CM & Industrial Engineering,
Louisiana State University,
Baton Rouge,
LA 70803-6409, USA
e-mail: bsarker@lsu.edu

1 Introduction

Nowadays, in the development of manufacturing, informatics is important. It connects enterprises to work together, share resources, and improve the product efficiency. To fulfill the target of agility, high performance, and low cost among enterprises all over the world, many manufacturing informatics modes, for example agile manufacturing [1], application service provider [2], manufacturing grid [3], and so on, are proposed and used widely. Most of them are emphasis just on how to connect distributed resources by network with less consideration of resource management and generalized dynamic sharing. At the same time, cloud computing as a new network application mode is springing up. It constructs computing service center and hires the computing power and storage by using virtualization technology. It combines multiple computing resources and information as a strong "cloud" and divides computing power and storage quickly and freely from cloud to user on demand through network. Cloud is just like a huge repository (and management) of resources which reflects the generalized dynamic sharing and cooperative management of resources.

Inspired by this, cloud manufacturing (CMfg) was presented to expand the service mode in manufacturing informatics and improve its dynamic [4]. It is a new networked manufacturing mode which aims at achieving low-cost resource sharing and effective coordination. It transforms all kinds of manufacturing, simulation, and computing resources and abilities into manufacturing services to form a huge "manufacturing cloud" and distributes them to user the on demand. In CMfg, there's a platform which combines core technologies of cloud computing, internet of things, and high-performance computing (HPC) and so on to implement the intelligent management, efficient collaboration, and dynamic arbitrary service composition and division. All these resources and abilities are intelligently sensed and interconnected into "cloud" and automatically managed via

the Internet to execute various manufacturing tasks [5]. That is to say, in manufacturing process, the CMfg platform can analyze and divide users' requests and automatically search suitable information, available manufacturing devices, and computing resources, and intelligently integrate and provide them to users. Users here can hire remote large equipments and computing resources without buying; get more specific information about design, simulation, production, delivery, and recycle; and monitor the whole task execution process. Thus, the whole life cycle manufacturing process in CMfg can be simplified in Fig. 1. With high intelligence and information, it is a high-level extension of service-oriented manufacturing and cloud computing.

Based on this idea, people would ask how to transform large devices as services for hiring and how to implement efficient resources allocation and integration. Actually, they are all supported by computing resources, as shown in Fig. 1. Computing resources, including CPU, processor, and I/O, at the physical layer [4] are the core infrastructure of the CMfg platform. They not only provide computing power as in cloud computing but also control a variety of other manufacturing resources and abilities directly for collaboration and sharing. They locate in different places and form a big resource pool in the CMfg platform through virtualization. Information sharing needs them, manufacturing devices invoking needs them, and computing/simulation work needs them, too.

In other words, under the centralized management, various heterogeneous computing resources are integrated and redivided as virtual machines by virtualization and assigned to user on demand for computing and simulation. Meanwhile, when manufacturing equipments access in the CMfg platform through transducers, computing resources then become a kind of control and management media. They encapsulate and map these manufacturing equipments as virtual resources with virtualization technology to support the effective interoperation, collaboration, and monitoring of manufacturing tasks [6]. High virtualization of all kinds of manufacturing hardware/software resources and high heterogeneity and distribution of computing resources are two key characteristics of CMfg compared with cloud computing. Therefore, the optimal allocation of computing resources (OACR) which means efficient dividing and scheduling computing resources in manufacturing process for full utilization and high efficient operation is one of the most primary problems in CMfg.

Besides, oriented to the whole manufacturing life cycle, manufacturing tasks are very complex. They usually include multidisciplinary collaborative tasks such as mechanical, electronic, or control simulation and manufacturing. The demands of tasks for communication and computation power of manufacturing resources are high and different. Unlike the previous scheduling problems [7, 8] in parallel computing systems, in CMfg, computing resources are divided into virtual machines and allocated to different tasks according users' requirements. It has the characteristics of large-scale, high heterogeneity, dynamic interconnection and group collaboration, which has imposed a new challenge on the construction of CMfg platform.

So this paper focuses on the OACR problem in CMfg and proposed a novel systematic model for it. To address this dynamic multiobjective problem, the allocation architecture and the core principle of CMfg are introduced. The formal description of OACR has considered the main constrain conditions from all aspects. From the point of packet communication and partition of computing power, it shows the detailed running process of the allocation of computing resources for manufacturing tasks. Classical intelligent algorithms are introduced and compared in solving the problem, and a new improved hybrid intelligent algorithm is presented to solve OACR. Simulation results on standard tests show that this new hybrid algorithm is pretty efficient to solve this kind of high-dimensional complex problems.

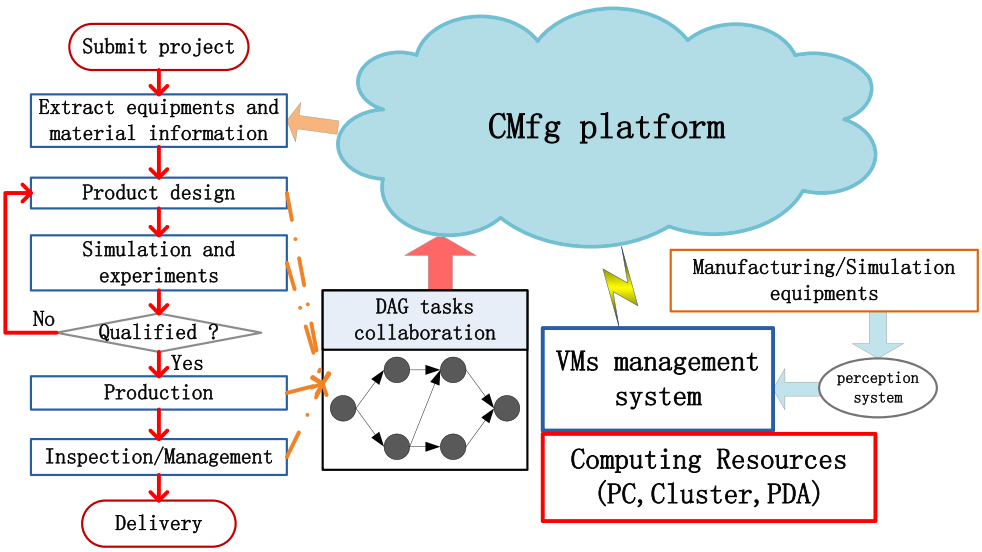

Fig. 1 The simplified manufacturing process in cloud manufacturing

The rest of this paper is organized as follows. Section 2 analyzes the related works on the newly CMfg research, the traditional modeling of computing resources, and the design of scheduling algorithms. Section 3 presents a simplified manufacturing process and a productive example to illustrate the problem. Section 4 describes the structure and characteristics of OACR in CMfg system. Section 5 elaborates the new model of OACR and gives the complexity proof for the problem. Section 6 investigates some classical intelligent algorithms and proposes a new improved algorithm for addressing the OACR problem. The paper ends with experiments and discussions in Section 7, and the conclusion is given in Section 8.

2 Related works

To perform larger scale collaborative manufacturing, CMfg was firstly presented in [4] in which a definition was given and the architecture of CMfg was introduced. Based on this, many studies about CMfg are started. Zhang et al. [6] further described the key technologies for the construction of CMfg, in which dynamic cloud services center in CMfg was defined as manufacturing cloud. Then, from the perspective of the structure of manufacturing cloud, the types of manufacturing resources, the dynamic sensing and accessing of hardware/software, and the method of information exchange in CMfg were elaborated. And for further understanding and the research of CMfg, Zhang et al. [9] then analyzed the differences and connections among CMfg and other related advanced manufacturing modes and then presented the target of CMfg, i.e., agility, servicing, greening, and intelligent in the whole manufacturing. Based on these research, Li et al. [10] specified the characteristics of CMfg and presented argument as a service, design as a service, fabrication as a service, experiment as a service, simulation as a service, management as a service, and integration as a service. These concepts are inspired by cloud computing but clearly distinguished CMfg from cloud computing. At the same time, the operational process of cloud manufacturing, the relation among resources, cloud service, and cloud platform and the importance of optimal allocation of whole manufacturing resources and tasks in CMfg were elaborated in [5, 11, 12]. All of these studies are macroresearches with less microanalysis in each key part. However, in detail, how to implement intelligent and agility in optimal allocation of computing resources for supporting these advanced manufacturing process, as one of the most important thing of constructing CMfg platform, still has not been studied.

In manufacturing system, job shop scheduling and workflow scheduling are much popular [13, 14] while the allocation of computing resources considered little. But from the global perspective, OACR is one of the most basic and important problem. OACR is a kind of prescheduling problem. It is more complex than several kinds of traditional job scheduling or task scheduling problems [15–17]. In the existing task scheduling models, tasks can usually be expressed in four types: directed acyclic graph (DAG) [18], hierarchical task graph [19], task interaction graph [20, 21], and Petri net [22]. The most commonly used is DAG, in which the nodes represent individual tasks and the directed arcs stand for communication overhead between tasks [23, 24]. Early DAG models were simplified as: the execution time of tasks are all the same, communication between tasks are excluding, the intercommunication interfaces between processors are enough, and multiple communications can be performed simultaneously [18], and so on. The traditional DAG task scheduling problems have been proven to be nondeterministic polynomial time (NP)-complete (Nondeterministic Polynomial-complete) [8]. It is far more complex in many kinds of manufacturing systems [25–27]. About the attributes of tasks, the concept of similarity is often expressed as granularity [28, 29], which indicate the ratio of communication overhead in a parallel program. The amount of communication edges is usually expressed as DAG density [30]. Besides, a variety of quality of service (QoS) indexes were also introduced in DAG, particularly in manufacturing task scheduling. Based on these QoS indexes, existing researches primarily focus on homogeneous cluster systems [31–33], scheduling more thread level tasks to less processors. The most frequently used topologies of the parallel systems are full interconnected network, hypercube network, grid network, public bus network, and so on [34]. The studies about heterogeneous systems are seldom. Typically, end point and network communication contention in heterogeneous systems are analyzed by Sinnen and Sousa [35]. The communication preparation, overhead, involvement of processors and communication mode of task scheduling are elaborated by Sinnen and Sousa [36] and Benoit et al. [37], and so on.

On the scheduling algorithms side, typical deterministic algorithms are list scheduling [38–40], clustering scheduling [28, 41, 42], linear programming [43], stochastic mapping [44], and several others. Kwok and Ahmad compared and summarized 15 types of scheduling algorithms in [18], which is widely cited. After that, a few efficient approximate algorithms [45, 46] were presented for solving these problems in acceptable times. With the increase of tasks and processors scale, traditional deterministic algorithms and original approximate algorithms can no longer meet the demand. Thus intelligent algorithms, such as genetic algorithms (GA) [47–50], ant colony optimization (ACO) [51–53], immune algorithms (IA) [54, 55], and so on and other new heuristic approaches [25, 56, 57] have been paid attention and widely applied to this kind of scheduling problems for finding the Pareto optimal solutions especially in manufacturing application field [58, 59].

However, the above-mentioned models are not practicable to CMfg. First, unlike the previous thread level tasks, manufacturing tasks (MTs) are usually carried by virtual machines (VMs) [4]. VMs not only execute HPC tasks but also supervise and control manufacturing hardware resources such as simulation equipment and machine tools. Users have different demands on them. Multi-VMs can run in same processor. The more VMs are carried at one processor, the slower their run. More importantly, there are frequent interactions between users and VMs during tasks' execution. In the other word, VMs generally execute coarse grain manufacturing tasks. Second, computing resources (CRs) with high heterogeneity are composed of different kinds of cluster, PC, PDA, and so on. They are scattered around the world with dynamic access, so the system topology is dynamic and uncertain. Hence different areas have different access bandwidths, links, and communication buffers [60]. Third, on CMfg platform, CRs have larger scale while MTs have relatively smaller scale with higher and complex demands. Based on such a complex system, therefore, OACR is different from the original scheduling problems and a detailed analysis of its new model and algorithms is presented in this paper.

3 Motivation

A CMfg system consists of manufacturing resources, manufacturing cloud (CMfg platform) and the whole life cycle manufacturing applications. Like the traditional service-oriented manufacturing modes, three user types—resource providers, cloud operators, and resource users—are included in the platform, as shown in Fig. 2 [6]. Manufacturing cloud senses and manages the manufacturing resources (hardware/software) from resource providers all over the world. When users submit a manufacturing mission to manufacturing cloud, the platform analyzes the mission and intelligently divides it into subtasks in accordance with the requirement number of VMs and devices and then forms them as a DAG. That means each subtask in DAG can be executed by only one VM or one device without separation. After the task partition, the manufacturing cloud needs to find available resources for each subtask and provide them as services for users. In fact, as introduced in Section 1, all of the interactive and run processes among them are not only supported by knowledge but also by computing resources. In order to show the importance of OACR among the triple process, we specified the abstract workflow of task execution in CMfg as shown in Fig. 3 and consider the multidisciplinary physical collaborative simulation for example.

Normally, for an accurate design and modeling in industrial manufacturing (such as airplane and automobile), physical collaborative simulation is important. On one hand, it needs collaborative simulation of Matlab and Adams and so on. On the other hand, it also needs driving simulator, multiaxis table, and visual equipment to work together along with software. So it is a complex process in manufacturing. Assume there is a physical simulation task submitted to manufacturing cloud. After a series of intelligent divisions of task, the following steps are done in the CMfg platform:

Step 1. *Requirement analysis of task DAG*: According to the users' requirement of DAG, analyze the communication and computation costs and the QoS constraints of tasks. Then check the accessed resources (include computing resources and manufacturing devices). If there is no available resource or the resources are not enough, then reject the tasks. Or the system will send a confirmation message to users and then take the next step.

Step 2. *Optimal allocation and strategy sending*: In terms of the QoS and costs of tasks, the manufacturing cloud determines which tasks need remote simulation physical devices. If the task needs physical device, then calculate the attribute values of device and map it to the requirement attributes of controlling VM. Else the platform only needs to calculate the requirement attributes of computing VM for task. As soon as the platform establishes these VMs' requirement, it executes a scheduling algorithm for mapping these VMs to available computing resources, then gets the optimal allocation of computing resources strategy and sends it to users.

Step 3. *Execution*: After the users' confirmation, the manufacturing cloud then invokes these VMs and simulation hardware to execute. The simulation runtime process could be controlled and monitored by users through controlling VMs on the Internet. If unexpected error occurs during execution, the platform will call the fault-tolerant migration strategy automatically and try to execute tasks again.

Step 4. *Result receiving and resources release*: At the end of the workflow, the manufacturing cloud receives the simulation results and sends them to users. Then the devices and VMs (computing resources) are released accordingly.

All of the above-mentioned four steps need complex collaboration with CRs' participation as the infrastructure. In general, OACR (step 2) in this workflow is the most important step which decides the total efficiency of task execution and can ultimately reduces the execution error during the runtime.

For analyzing the OACR problem from both computation and communication perspective and finding better solution

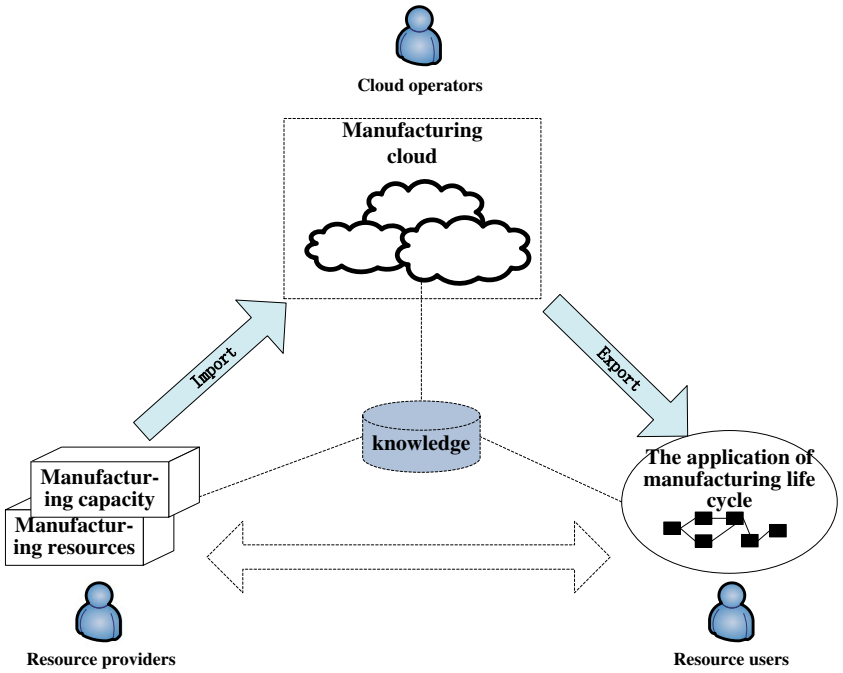

Fig. 2 The abstract operation principle of cloud manufacturing [5]

strategies to improve task execution and collaboration efficiency in CMfg, the following three issues are needed to be studied:

- The structure and characteristics of CRs in CMfg;
- The formalized descriptions of CRs and the OACR model with specific constraints and QoS; and
- The efficient intelligent algorithms for addressing the OACR problem.

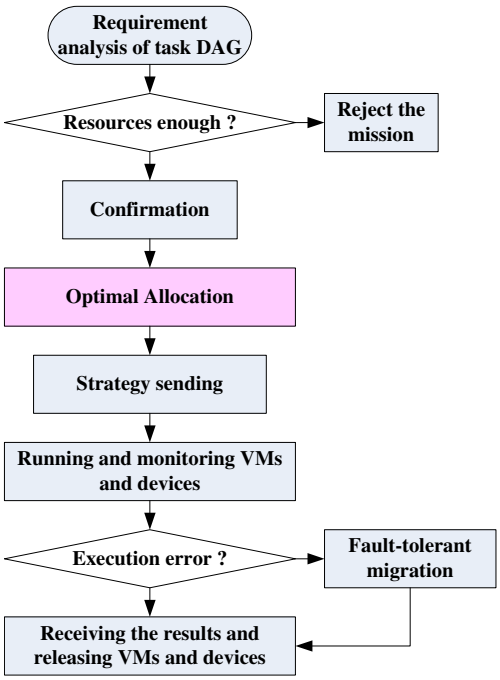

Fig. 3 The specific workflow of task execution

This paper will directly focus on these three issues.

4 The structure and characteristics of OACR

According to the simplified manufacturing process shown in Section 3, to build a practical model of optimal allocation of computing resources, the core allocation structure and its characteristics should be emphasized firstly. The structure of OACR gives the detailed allocation process of VM management and the distribution characteristic of CRs. Based on that, the communication and topology characteristics of CRs in CMfg are elaborated for further study of the model of OACR.

4.1 The structure of OACR

The OACR of CMfg is composed of three levels: manufacturing task level, virtual resource level, and computing resource level, as shown in Fig. 4. On the manufacturing task level, assume the tasks of a given MTs set are meta-tasks. *Metatask* means that the task is inseparable for executing in VMs/CRs, as discussed in Section 3. For instance, in a multidisciplinary collaborative simulation, each module runs on one VM with user's control and interaction. Each VM is inseparably running on only one CR. So MTs and VMs have the one-to-one mapping relationship.

When MTs' demands are abstracted as virtual resources' demands, the virtual machine manager receives the demand information and allocates available VMs for physical manufacturing resources. The physical manufacturing resources can be not only manufacturing/simulation equipments but

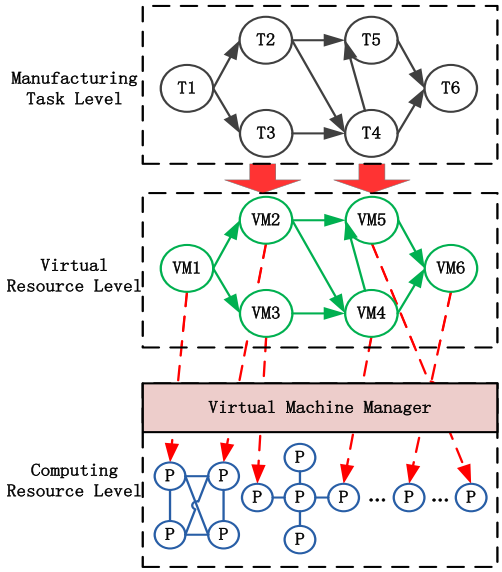

Fig. 4 The process framework of OACR

also computing resources. Each of manufacturing/simulation equipment needs a CR to control and monitor. Thus, all VMs are supported by CRs. They form the virtual resource level and support the running of MTs. Because the customized MTs are applied by user, the constraints of VMs (e.g., the demand of memory size, computing speed, communication link and bandwidth, etc.) could be obtained at the same time.

As shown in Fig. 4, the mapping of manufacturing task level and virtual resource level is the foundation of OACR, and the mapping of virtual resource level and physical resource level is central to the optimization. In this paper, the manufacturing task level and virtual resource level are merged, and the optimal allocation of MTs (or VMs) and CRs under the concrete computation and communication constraints is emphasized.

4.2 The characteristics of CRs in CMfg

In the actual operation of VM management, the topology and communication properties of CRs are very important. These factors determine which CRs are most suitable for MTs and which allocation scheme is the most efficient one, and almost all constraints of OACR come from the characteristics of CRs.

4.2.1 Topology

The CMfg network is different from other enterprise network or public network and compromised by many distributed manufacturing resources around the world. For the sake of facilitated management and extension, master–slave (manager–service) mode is adopted in the platform. As the shoring of foundation, computing resources can dynamically access the platform via the Internet. They are managed and controlled by high stable VMs management system. According to their locations, CRs can be divided into multiple subsets. This topology is similar to the classical tree network. Each subset belongs to different provider who has full authority and obligation to operate and maintain it. The subsets could be mesh/star topology cluster or independent PCs. Due to the different topologies of CRs' subsets, the transmission in group can be half duplex, full duplex, or busses. With the development of the high-speed Ethernet switch, transmission among groups is all full duplex.

4.2.2 Communication ports

Generally, the port communication of master–slave system can be classified as single-port mode [61] and multiport mode [62]. *Single-port* mode means that the network central node can only send or receive limited byte message to/from one slave node in a given period of time. On the contrary, in *multiport* mode, the network central node can send or receive limited byte message to/from one or more slave nodes in a given period of time. In CMfg, multiport communication mode is adopted in CRs and the platform.

However, in this multiport mode, owing to the complex and frequent intercommunication among CRs for a large number of MTs, the amount of transmit data from multiport to the central node must be huge, which is called *periodic burst* or *data surge*. Periodic burst can cause packet loss and network congestion. In order to avoid this, the general port transmit mechanism of cloud computing is adopted in CMfg, that is large caches are allocated in the receive direction of switches while small caches are allocated in the send direction to control the flow burst. In this case, the critical cache in the receive direction for preventing data surge and the relationship with the communication time between CRs should be particularly considered.

4.2.3 Communication bandwidth

Associating with multiresources around the world, the core network protocol of CMfg is still TCP/IP mode (with two-sided communication type [36]). In TCP/IP protocol, data are usually divided into small packets and transmitted one by one. If one of the packets is not arriving, the packet will be resent or the congestion control strategy will be loaded in the network. Then the data transfer rate will be slower. Though the TCP/IP protocol is efficient in short-distance transmission, it may cause delay or packet loss in the large-scaled remote communication in CMfg. Thus, in gigabit network, long-distance communication between CRs will lead to delay at hundred milliseconds scale and small probability packets loss. This makes the actual transfer rate be only about one tenth of the original bandwidth or even smaller. Therefore, with existing communication technologies, the transfer rates of remote

communication among CRs can only reach tens to hundreds of megabits per second.

5 The formulation of the OACR problem

The above-mentioned structure and main characteristics clearly reflect the high heterogeneity and dynamics of optimal allocation of computing resources. It comes from the traditional models of task scheduling but is more complex than the traditional ones. For describing the model of OACR in formalization, the formal descriptions of tasks and computing resources in traditional task scheduling are shown as follows. And based on the traditional definitions, the new model of OACR is presented then.

5.1 Traditional models of tasks and CRs

In general task scheduling problem, the tasks and the multiprocessor system are defined as follows.

Definition 1 The tasks set in multiprocessor system can be presented as a weighted DAG, G=(V, E, c, w). The set is $V = \{v_i | i = 1 : v, v = |V|\}$, where v_i represents the task of the set V, and v is the cardinality of nodes. The set E=V×V, e=|E| is the number of edges, and $e(ij) \in E$ represents the communication between v_i and v_j. $w(i)$ represents the computation cost of v_i. $c(ij) \in c$ represents the communication cost of the directed edge $e(ij)$. If there is no communication between v_i and v_j, then $e(ij)$=$c(ij)$=0.

Let the predecessor tasks set of v_i be $pred(v_i)$, and the successor tasks set be $succ(v_i)$. The node with no predecessor task $pred(v_i)$=∅ is named *source* node, and the node with no successor task $succ(v_i)$=∅ is called *sink* node. They all strictly observe the tasks' priority rules. It means a node can only be started after all its parent (preceding) nodes are finished.

According to this task graph definition, the general computing resources model of multiprocessor system is defined as follows:

Definition 2 The multiprocessor system, M=(P, s, bw), consists of a finite set of processors $P = \{p_k | k = 1 : p, p = |P|\}$ which are connected by a communication network. The notation $s = \{s(k) | k = 1 : p, p = |P|\}$ represents the computing power of processors, bw=P×P represents the bandwidth between processors, and $bw(kl) \in bw$ is the bandwidth between p_k and p_l. If the system is homogeneous, the processor's computing power and their bandwidths are all equal, that is $\forall k, l \in [1, p], k \neq l \Rightarrow s(k) = s(l)$, $bw(k)$=$bw(l)$. Heterogeneous systems are then contrary.

In these models, processors are usually all directly connected, and the tasks are nonpreemptive. If two tasks are carried by the same processor, their communication cost is 0, and it assumed that the transmission rate of computing resources to be equal to the bandwidth (the ideal value).

5.2 The new models of OACR in CMfg

According to the characteristic of CRs, the uncertain topology can be simplified as shown in Fig. 5. The above-mentioned CRs subsets are simplified as different groups. Different topologies in groups can be reflected by the communication links among CRs. That is to say, with different topologies, CRs in the same group connected with each other through different communication links by local connection, and CRs in different groups are connected by switches via the Internet. Stand-alone PCs can be classified as a special group. They are connected with each other directly via the Internet.

In theory, the biggest difference between general computing tasks and MTs are whether they are controlled by and interacted with users during execution time. Control and supervision are generally implemented by multithread in CRs. The MTs' computation costs vary according with users' interactions. Because of the frequent control and supervision in MTs, the execution times might be much longer. How long it will be depends on how many interactions and supervisions during MTs' execution. For considering this, the new MTs model is defined as:

Definition 3 The MTs set in CMfg can be presented as a weighted DAG, $G = (V, E, c, w, oper\_p, su\_p)$. The definition of $V = \{v_i | i = 1 : v, v = |V|\}$, E=V×V, w, and c are the same as the traditional task model (definition 1). The set $oper\_p = \{oper\_p_i | i = 1 : v, v = |V|\}$ and $su\_p = \{oper\_p_i | i = 1 : v, v = |V|\}$ represent relative interoperation-to-computing ratio and relative supervision-to-computing ratio separately. That is to say, the estimated cost of interoperation $oper = c \times oper\_p$ and the estimated cost of supervision $su = c \times su\_p$.

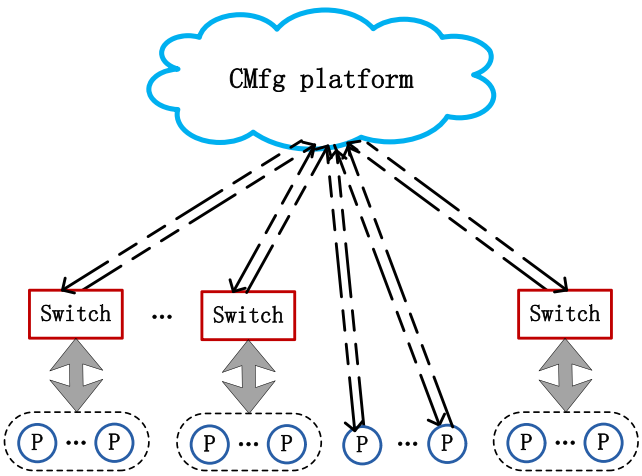

Fig. 5 The simplified topology of computing resources

Then the total cost of each node in G can be calculated as $W(i) = w(i) \times (1 + oper\_p(i) + su\_p(i))$. If there is no interaction or supervision in v_i, then $oper\_p(i) = 0$ or $su\_p(i) = 0$. These two factors can clearly reflect the user's demands for interaction in MTs, and the new model can then be more practical.

With the users' interaction and large-scaled computation and communication costs, the installments and involvements of VMs can be ignored from both communication and computation perspective. On the basis of the topology and the characteristics of CRs, the new CRs model can be defined as follows.

Definition 4 The CRs system model of CMfg is given by M=(P, s, rou, bw, mem, buf, rel), where

- $P = \{p_{kl} | k = 0 : m, l = 1 : n_k\}$ represents the CRs set, in which m is the number of CRs groups and n_k is the resources number in group k. Let k=0 represent the stand-alone PCs, and n_0 represent the number of these stand-alone PCs. Therefore, the total quantity of CRs is $|P| = \sum_{k=0}^{m} n_k$.
- $s = \{s(kl) | k = 0 : m, l = 1 : n_k\}$ represents the computing power (the computing speed) of the CRs set.
- $mem = \{mem(kl) | k = 0 : m, l = 1 : n_k\}$ represents the available memory volume of CRs, in which $mem(kl)$ varies dynamically with the task running. Its memory volume is reduced accordingly, when a task (VM) is assigned to the CR.
- $bw = \{bw(kl) | k = 1 : m, l = 1 : n_k\}$ represents the bandwidth between CRs and switch in each group. Considered the simplified topology (Fig. 2), the access bandwidths of switches to the Internet are defined as $BW = \{BW_i | i = 1 : m\}$. Due to the stand-alone PCs are connected via the Internet directly, let $bw_0 = \{bw_0 (kl) | k = 1, l = 1 : n_0\}$ be their bandwidth to the Internet. The bandwidths in groups are generally gigabit, so $\forall k \in [1, m], l \in [1, n_k] \Rightarrow BW_k \ll bw(kl)$.
- $rou = \{rou_i(kl) | i = 1 : m, k \neq l, k, l \in [1, n_i]\}$ represents the communication route between p_{ik} and p_{il} in local connection. Because the subsets of CRs are dynamic and complex, the route and bandwidths of the communication between two CRs needs to be calculated by a specific way when the subset is accessed. So the concrete topologies in groups are not considered in this model. It is assumed that the communication routes and bandwidths among CRs are previously figured out by some kinds of routing algorithms. The simplified communication route $rou_i(kl) = \{link_1, \cdots, link_r\}$ [36] varies with different topologies, and the bandwidth of rou_i (kl) is defined as $bw \times (rou_i(kl)) = \min\{bw\ (link_1), \cdots, bw(link_r)\}$.
- $buf = \{buf(kl) | k = 1 : m, l = 1 : n_k\}$ represents the buffer size of the switch communication ports in each group. According to [34], it assumed that the highest tolerable abrupt data of each port to be:

$$D(kl) = buf(kl) + buf(kl)\frac{BW_k}{bw(kl) - BW_k} = buf(kl)\frac{bw(kl)}{bw(kl) - BW_k} \quad (1)$$

- $rel = \{rel(kl) = (rel\_p(kl), rep\_t(kl)) | k = 0 : m, l = 1 : n_k\}$ represents the reliability of the CRs set. The reliability of CR means the probability that the computing resource fails to connected in the consequence of the communication link or occurrence of another MTs set which leads to pause computation for some times. So $rel\_p(kl)$ represents the probability, and $rep\_t(kl)$ represents the predicted failure duration time of CRs. Then $\forall k \in [0, m], l \in [1, n_k] \Rightarrow rel\_p(kl) \in [0, 1]$.

In this model, rou and bw are used to represent the local connections and the remote connections separately. Therefore, the OACR model can be described as S=(G, M), where $G = (V, E, c, w, oper\_p, su\_p)$ represents the MTs and $M = (P, s, rou, bw, mem, buf, rel)$ represents the CRs.

5.3 The constraints and objective function of OACR

Based on the structure described in Section 4.1, four issues of CRs are considered in this paper.

1. The minimum acceptable memory size $MEM_{\min}$ (i) for task v_i;
2. The minimum acceptable reliability $REL_{\min}$ (i) for task v_i;
3. The minimum acceptable computing speed $EXE\_SPEED_{\min}$ (i) for task v_i; and
4. The longest acceptable communication time $COM\_TIME_{\max}$ (ij) for task v_i, usually it is much looser than the above three constraints.

When an MTs set $G = (V, E, c, w, oper\_p, su\_p)$ is applied to the CMfg platform, the system M=(P, s, rou, bw, mem, buf, rel) will provide right CRs for it. Let $k(i)$ and $l(i)$ be the group number and the position of the selected CR for task v_i, and let $p\_load(k(i)l(i))$ be the load of the selected CR for task v_i, which is measured by MTs per CR. Then the constraints of each selected CR can be described as:

- When multi-MTs $\{v_1 \cdots v_n, n < v = |V|\}$ select the same CR p_s, if $mem(s) \geq \sum_{i=1}^{n} MEM_{\min}(i)$ then $p\_load(s)$=1, else MTs are needed to queue for execution, that is $p\_load(s)$=n;
- $\forall i \in [1, v]$, the computation speed of the selected CR for task v_i satisfied: $s(k(i)l(i))/p\_load(k(i)l(i)) \geq EXE\_SPEED_{\min}$ (i), and then the execution time of task v_i can

be expressed as $EXE\_TIME(i) = W(i) \times p\_load \times (k(i)l(i))/s(k(i)l(i))$;

- $\forall i \in [1, v]$, the reliability of the selected CR for task v_i satisfied: $rel\_p(k(i)l(i)) \geq REL_{\min}(i)$,

The constraints of the communication ability of the selected CRs can be represented as $COM\_TIME(ij) \leq COM\_TIME_{\max}(ij)$. It can be divided into two cases. $\forall i,j \in [1,v], i<j$ (v_i is the predecessor task of v_j), let $COM\_TIME\_s(ij)$ be the data sending time between the two tasks, and $COM\_TIME\_r(ij)$ be the data receiving time between two tasks.

Case 1 When v_i and v_j are in the same CRs group, $k(i) = k(j) = k \neq 0$. Without considering the reliability factors, the sending time of v_i is equal to the receiving time of v_j, that is:

$$COM\_TIME\_s(ij) = COM\_TIME\_r(ij) = COM\_TIME(ij) = c(ij)/rou_k(l(i)l(j)) \tag{2}$$

With the addition of the *rel* factor, let $t(x)$ be the average communication time between v_i and v_j which originally needs x seconds of processing. If the CR $p_{k(j)l(j)}$ does not fail halfway, then the communication time needs $1+t(x-1)$ seconds, but if it fails at midway [with the probability $rel\_p(k(j)l(j))$], then it needs to wait $rep\_t(k(j))(l(j))$ seconds and also need another $t(x)$ seconds to complete the communication. Therefore it has:

$$t(x) = (1 - rel\_p(k(j)l(j)) * (1 + t(x-1)) + rel\_p(k(j)l(j)) * (t(x) + rep\_t(k(j)l(j))) \tag{3}$$

$$t(x) = 1 + t(x-1) + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))} \tag{4}$$

Since $t(0)=0$, it can be written as:

$$t(x) = x\left(1 + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))}\right) \tag{5}$$

According to Eq. 5, the communication time between v_i and v_j can be expressed as:

$$COM\_TIME(ij) = \left(1 + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))}\right) * \frac{c(ij)}{rou_k(l(i)l(j))} \tag{6}$$

Case 2 When v_i and v_j are in different CRs groups,

- If $c(ij) < D(k(i)l(i))$, without considering the reliability, the sending time of task v_i is equal to:

$$COM\_TIME\_s(ij) = c(ij)/bw(k(i)l(i)) \tag{7}$$

and the receiving time of task v_i is equal to:

$$COM\_TIME\_r(ij) = \left\{ \begin{array}{l} \frac{c(ij)}{\min\{BW_{k(i)}, BW_{k(j)}\}}, k(i) \neq k(j) \neq 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_{k(j)}\}}, k(i) = 0 \\ \frac{c(ij)}{\min\{BW_{k(i)}, BW_0(l(j))\}}, k(j) = 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_0(l(j))\}}, k(i) = k(j) = 0 \end{array} \right\} = COM\_TIME(ij) \tag{8}$$

After adding the reliability factors, according to Eq. 5, the sending time of task is unchanged, but the receiving time is changed as:

$$COM\_TIME\_r(ij) = \left(1 + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))}\right) * \left\{ \begin{array}{l} \frac{c(ij)}{\min\{BW_{k(i)}, BW_{k(j)}\}}, k(i) \neq k(j) \neq 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_{k(j)}\}}, k(i) = 0 \\ \frac{c(ij)}{\min\{BW_{k(i)}, BW_0(l(j))\}}, k(j) = 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_0(l(j))\}}, k(i) = k(j) = 0 \end{array} \right\} = COM\_TIME(ij) \tag{9}$$

- If $c(ij) > D(k(i)l(i))$, the sending rate of task v_i must be reduced. According to Eq. 1, the sending rate $ssend(i) = c(ij) * BW_{k(i)}/(c(ij) - buf(k(i)l(i)))$. So the sending time of v_i is changed as $COM\_TIME\_s(ij) = (c(ij) - buf(k(i)l(i)))/BW_{k(i)}$. Yet the receiving time of v_j would remain as Eq. 9, and $COM\_TIME\_r(ij) < COM\_TIME\_s(ij)$.

Based on these constraints, let the start time of task v_i be $START\_TIME(i)$, the execution time of v_i be $EXE\_TIME(i)$, and the finish time of v_i be $FINISH\_TIME(i)$, then:

$$START\_TIME(j) = \max_{i \in pred(v_j)} \{COM\_TIME\_r(ij)\} \tag{10}$$

$$\begin{aligned} FINISH\_TIME(j) &= START\_TIME(j) \\ &\quad + EXE\_TIME(j) \\ &\quad + \max_{i \in succ(v_j)} \{COM\_TIME\_s(ij)\} \end{aligned} \tag{11}$$

Therefore the temporal relation between two adjacent tasks is as shown in Fig. 6. Note that the source node v_1 do not need to receive data, so $START\_TIME(1) = 0$, and the sink node's sending time is also the MTs submission time, that is:

$$\begin{aligned} COM\_TIME(v) &= TASK\_SUBMISSION\_TIME(V) \\ &= \begin{cases} \frac{c(v) - buf(k(v)l(v))}{BW_{k(v)}} & if\ c(v) > D(k(v)l(v)) \\ \frac{c(v)}{BW_{k(v)}} & if\ c(v) < D(k(v)l(v)) \end{cases} \end{aligned} \tag{12}$$

where $c(v)$ represents the submission data of MTs.

In conclusion, the execution time of the whole MTs set is:

$$\begin{aligned} TOTAL\_TIME(V) &= FINISH\_TIME(v) - START\_TIME(1) \\ &= FINISH\_TIME(v) \end{aligned} \tag{13}$$

As the constraint of memory size of CRs is embodied in the constraint of computing speed and the reliability factors is embodied in the constraints of communication in CRs, the optimal object function and the constraints of OACR $S=(G, M)$ can be summed up as:

$$\begin{cases} \text{MINIMIZE } TOTAL\ TIME(V) \text{ SUBJECT TO} \\ \forall i \in [1, v],\ \frac{s(k(i)l(i))}{p\_load(k(i)l(i))} \geq EXE\_SPEED_{\min}(i) \\ \forall i, j \in [1, v], COM\_TIME(ij) \leq COM\_TIME_{\max}(ij) \end{cases} \tag{14}$$

5.4 Problem complexity

Traditional task scheduling problems are proved to be NP-complete problems. To prove the complexity of OACR, two definitions are introduced in this section according to [63].

Definition 5 [63] (a polynomial time transformation) A polynomial time transformation of a decision problem P' into a decision problem P ($P' \propto P$) is a function f: $D_{p'} \rightarrow D_p$ satisfying the following two conditions:

(a) The function can be computed in polynomial time and
(b) For all instances $I \in D_{p'}$, there exists a solution to I if and only if there exists a solution to $f(I) \in D_P$.

Definition 6 [63] (An NP-complete problem) A decision problem P is said to be NP-complete, if $P \in NP$ and $P' \propto P$ for any $P' \in NP$.

Theorem 1 *The OACR problem is an NP-complete problem.*

Proof In the OACR problem, the task quantity of a MTs set $v=|V|$ is less than the processors number of a CRs set $p=|P|$. One processor can carry multitasks.

1. When $1<N_p<v$, choosing N_p suitable processors from p resources has $C_p^{N_p}$ solutions. After choosing these N_p processors, the mapping of v metatasks and N_p resources turn in to the traditional scheduling problem. In this situation the OACR problem can be reduced to the traditional scheduling problem. According to definition

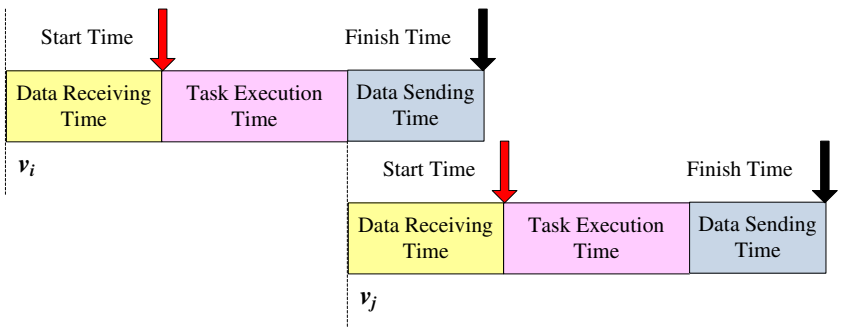

Fig. 6 The temporal relation between two MTs

6, the traditional scheduling problem is NP-complete, so the OACR problem is NP-complete.

2. When $N_p=v$, choosing v suitable processors from p resources has C_p^v solutions. Afterwards, the mapping between v metatasks and v computing resources can be converted to traveling salesman problem (TSP), which is the full permutation problem. In this situation, the OACR problem is reduced to TSP problem. From definition 6, TSP problem is NP-complete, thus the OACR problem is NP-complete, too.

The above discussions contain all cases of the OACR problem, so theorem 1 is true Q.E.D.

From the point of the solutions, let n be the total amount of CRs in the CMfg platform and let v be the task number of an applied MTs set. For the case of no time constraints, each task has n choices. So the size of the solution space is n^v. If someone wants to find the best solution one by one in the entire solution space, then they need $O(n^v)$ steps to complete. It is a huge calculation. For example, if there are 100 CRs and 5 MTs, the solution space is 5^{100}. It is a very huge number for calculation. At present, no deterministic algorithms can solve it in polynomial time. So, intelligent algorithms are introduced in this paper.

6 The intelligent algorithms for addressing OACR

The most frequently used intelligent algorithms for the traditional task scheduling are GA [47, 48] and ACO algorithm [51, 52]. Besides, IA has shown great potential in combinatorial optimization problems [64, 65]. They are widely used in various kinds of scheduling problems, and their basic processes are shown in Fig. 7. Based on these three classical intelligent algorithms and with the consideration of the complex of OACR, a new improved niche dynamic IA (NDIA) is proposed in this paper for better solutions. These four algorithms will then be used and generally analyzed for addressing the OACR problem in detail.

6.1 Review of GA, ACO, and IA

GA is an adaptive global optimization stochastic search algorithm which is inspired by the principle of evolution and natural genetics. With a set of structured populations which represented the candidate solutions of the problems, it combines the survival of the fittest among populations (selection), the structured yet randomized information exchange (crossover), and the random bit mutation.

Firstly, roulette wheel selection is used commonly in the standard GA. It is performed by randomly picking a certain amount of populations according to their fitness values to form a new group of populations. The one with higher fitness value occupied higher probability of selected. Secondly, each two of the selected populations exchange parts of their gene bits in the crossover operation. So two new genetic chromosomes are generated, and the better gene bits go into the next generation. Thirdly, the mutation operation randomly changes some gene bits of populations with a certain probability for increasing the diversity. After the

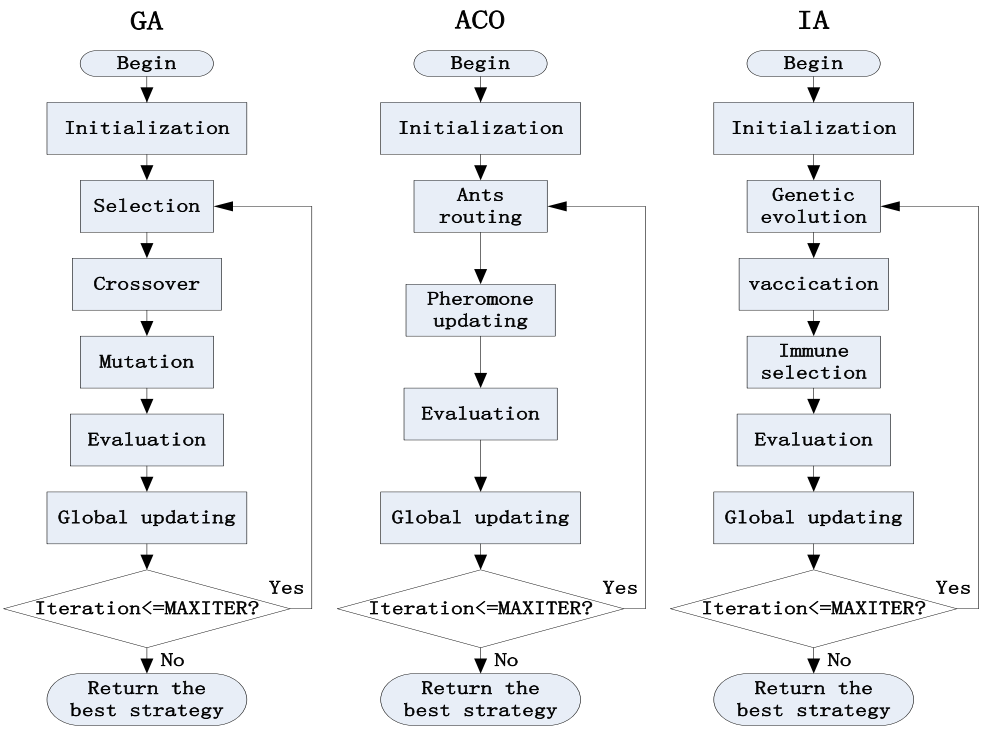

Fig. 7 The process of GA, ACO, and IA

above three steps, if the new best population is better than the old one, then it would evaluated by the new one, or the best population record will remain unchanged. This process will repeat and then terminate when the maximized generations are reached or the optimal solution is found.

ACO is a kind of swarm intelligence algorithm which takes inspiration from the social behaviors of ant colony. It combines ants' routing and pheromone update. The original intention of ACO is to solve the complicated path optimization problems, such as TSP, and edge scheduling.

In the process of routing and finding foods, ants deposit pheromone on the path they have walked in order to mark some favorable path and broadcast the information. The longer the path, the lower the density of pheromone is. Other ants can perceive this pheromone and recognize its density. They have a large probability to select the path which has the greater pheromone density. Then a kind of information positive feedback is formed. The pheromone density on the optimal path will become higher, while the pheromone density on other paths will reduce as the time goes by. Finally the whole colony will find the optimal path.

With this inspiration, the a priori knowledge is introduced and formed the standard ACO. That is to say, ants are finding path not only in the light of the pheromone, but also according to the a priori rules (knowledge) of the problems. As the same with GA, the whole process continues many evolution times until the ants find the optimal path (solution) or the number of evaluation steps reaches a predefined value.

IA is a kind of evolutionary programming which based on the immune system in biotic science. With the introduction of the concepts and the characteristics of antigen recognition, immunological memory, and immune regulation, diversified immune algorithms are presented. The immune algorithm proposed by Wang et al. [65] is a typical and efficient one. In this paper, it will be applied in OACR and IA here just indicates the algorithm in [65].

More specifically, IA is a convergence of immune theory and genetic algorithm. It contains genetic evolution, immune vaccination, and immune selection, but first of all, antigen extract and vaccine selection according to the feature information of problem is the most important part of this algorithm. It is a core rule to lead the population evolution in the right direction. Then, the population initialization and the genetic evolution are all the same as the standard genetic algorithm. After selection, crossover, and mutation, new populations are vaccinated by antibodies. That is injecting a priori knowledge into the new populations in some degree for improving their fitness values (the a priori knowledge in IA is the same as in ACO.) Then, new populations are selected by immune selection operation according to the choosing rules of simulated annealing. Three steps will repeat until the ending conditions are meeting.

All of the above-mentioned intelligent algorithms are evolved with a number of cycles by their own mechanism. As shown in Table 1, only GA does not need the a priori knowledge with less control parameters. Without other improvement strategies, its global convergence is weak, but its robustness is quite good. ACO and IA both need the direction of a priori knowledge with good global convergence. Yet the control parameters of ACO are more than the IA's. Their real effect in solving the OACR problem will be shown in Section 7.

6.2 The improved niche IA for the OACR problem

Inspired by the above three algorithms, the improved niche IA takes the techniques of pheromone guide from ACO and the ecological niche strategy. Its framework is shown in Fig. 8.

Compared with IA (as shown in Fig. 7), the niche strategy, and the dynamic vaccination and pheromone updating strategy are added in NDIA. Niche strategy is used for improving exploration during searching, dynamic vaccination and pheromone updating strategy is taken for further improving the exploitation and searching direction with the dynamical consideration of both computation and communication in OACR. The genetic evolution just adopts the standard roulette wheel strategy, single-point crossover and mutation. The improvement of initialization, the object function, and new improved strategies in NDIA for solving OACR are elaborated as follows.

1. Initialization

   For solving OACR problem, real number coding is used in the experiments. Real number coding can avoid the encoding/decoding process, improve the accuracy, and reduce the complexity of the algorithm. As shown in Fig. 9, the sequence number of gene bits is denoted as the serial number of MTs, the numbers in the gene bits represent the index of CRs, and their subscripts

Table 1 The characteristics of GA, ACO, and IA

Algorithm	Year	Mechanism	A priori knowledge	Global convergence	Control parameters
GA	1975	Biological evolution	Needless	Weak	Less
ACO	1992	Ants behavior	Need	Strong	More
IA	2000	immune system	Need	Strong	Medium

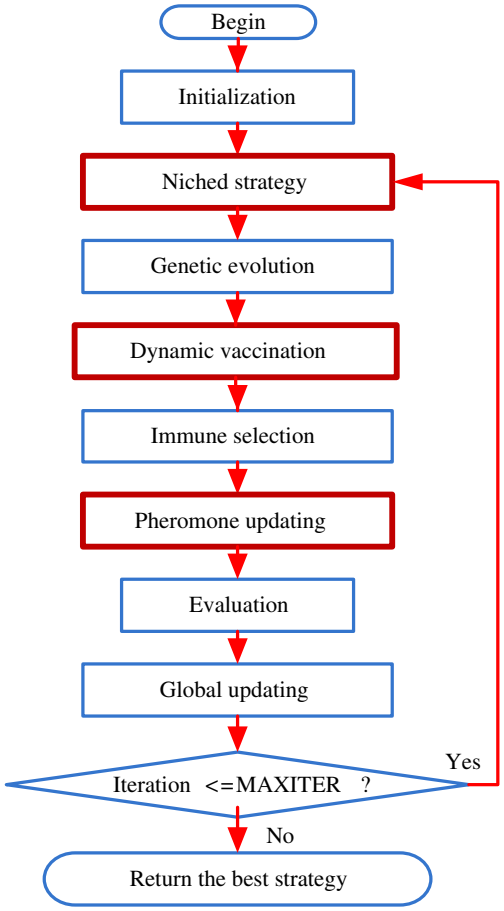

Fig. 8 The framework of NDIA for addressing OACR

represent the group indexes of CRs. In other word, each gene bit occupies two integer bits. This kind of coding method takes less space and is more intuitive and simple.

2. Object function in evolution

   Because the standard GA with the roulette wheel strategy is commonly used to find the individual with the maximize fitness value, the fitness evaluation function of OACR in all intelligent algorithms is set as:

$$\max f = \frac{Const}{TOTAL\_TIME(V)} \tag{15}$$

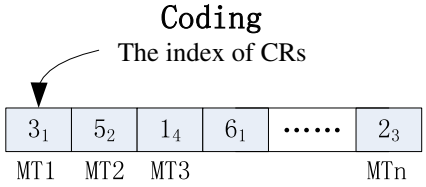

Fig. 9 The real number coding for OACR

Const is a constant which makes the object fitness value in the algorithms neither too large nor too small.

3. Niche strategy

   For improving the diversity and balance the exploration and exploitation of the algorithms, the technology of ecological niche is introduced in it. In NIA, the hamming distances D_{ij} between two individuals should be calculated before the implementation of genetic evolution, as shown in Eq. 16.

$$D_{ij} = \|X_i - X_j\| = \sqrt{\sum_{k=1}^{N} (x_{k,i} - x_{k,j})^2} \tag{16}$$

   where X_i and X_j represent individual i and individual j, and $x_{k,i}$ and $x_{k,j}$ represent the gene bits of each individuals separately. If D_{ij} (between individual i and j) is less than a preset parameter L, then the individual with lower fitness value will multiple a penalty function to make it more lower. This action could wipe off the similar individuals and protect the diversity of the population to improve the search ability.

   Because of the definition of the maximum object function, here the penalty function is set as $f=10^{-5}$ and let the parameter L to be v which is the size of the individual in algorithms (MTs' number).

4. Dynamic vaccination and pheromone updating

   As is known to all, antigen extraction and vaccine selection is the core factor of IA and it usually comes only from the a priori knowledge (i.e., heuristic information) of the problems. If the a priori knowledge is extracted inappropriately, the algorithm will evolve in the wrong direction and no feasible solution can be found. However, the a priori knowledge in the complex problem is usually complex and varying with different situations. For example, both the computation and communication (node and edge factors) should be considered in OACR especially when the computation rate of MTs is equal to its communication rate. The incidence relations among tasks are very complex, and the computation and communication power of CRs are varying dynamically. The extracting and the rate selection of the two factors are therefore hard. This directly influences the efficiency of IA in solving the global optimal solutions.

To avoid this problem, and considering the dynamic change of the memory size, communication bandwidth, and reliability constraints of CRs, we present the new dynamic vaccination strategy. That is, extraction and calculating the heuristic information (η_{ij}) of allocating the CR p_j to the MT v_i need real time in each evolutionary cycle. It is time-consuming but can obtain higher accuracy result in the

scheme, and for simplification, the heuristic information function is set as:

$$\eta_{ij} = \frac{\left(\frac{s(k(i)l(i))}{p\_load(k(i)l(i))}\right)^{\alpha} * (bw(\mathrm{k(i)}l(i)))^{\beta}}{\left(1 + \frac{rel\_p(k(j)l(j) * rep\_t(k(j)l(j)}{1 - rel\_p(k(j)l(j)}\right)^{\gamma}} \quad (17)$$

where α, β, and γ represent the importance of the execution speed, communication bandwidth, and reliability of CR in the heuristic information and they satisfied α, β, $\lambda \in [0, 1]$. In Section 7, the value of these parameters are tested and discussed for better solution.

Besides, for further improving the searching direction, the pheromone of ACO is brought in NDIA in this paper. That is to say, the improved NDIA extracts antigen and vaccine not only by the a priori knowledge, but also by the pheromone which is released by the previous best individual of the whole populations. As a supplement, the pheromone increased the experiential guidance and made a positive feedback in IA. To put it more specifically, let τ_{ij} be the density of pheromone of mapping MT v_i to CR p_j and η_{ij} be the a priori knowledge (i.e.m the heuristic factor) of the problem. The vaccine can then be expressed as:

$$vaccine = \left(\tau_{ij}\right)^{\varphi} \left(\eta_{ij}\right)^{\phi} \quad (18)$$

where τ_{ij} is updating as the same as in ACO, and φ, $\phi \in (0, 1)$ represent the strength factors of τ_{ij} and η_{ij} separately. If φ is too larger than ϕ, then vaccine will be directed by the experience of populations and result in low searching ability and low convergence. But on the contrary, too larger ϕ will also changed vaccine to static and lead to premature. So, according to ACO, the rates of pheromone and heuristic information in Eq. 17 in this paper are set the same as ACO (i.e., φ=1, ϕ=5).

Then at the vaccination step, the one or more gene bits of the selected populations will be changed as the one with the highest *vaccine* at a certain rate. This strategy can improve the convergence, increase the robustness, and simplify the previous vaccine extraction and calculating in algorithms.

6.3 The time complexity of the proposed algorithms

The time complexity of the intelligent algorithms is dynamically varied with different problems. Let n be the scale of the population, m be the scale of CRs in CMfg platform, and v be the scale of the MTs set applied by user. The algorithms' complexities in each cycle (or generation) are shown in Table 2.

GA does not need the heuristic information to direct its evolution. In selection, the complexity of the roulette wheel strategy in the best situation is O(n), and its worst complexity is O(n^2). Due to the worst case, complexity of the algorithm is the upper bound of run time. The complexities in Table 2 just mean the worst case complexities.

In ACO, because the ants' routing needs to calculate the priori knowledge in each cycle, finding a suitable CR for each task then needs m step to get all of the heuristic information of CRs. With n populations and v tasks, its complexity is O(nmv).

The same as the ACO, IA needs to find a certain number of population and vaccinates them. In vaccination, the load and memory of each CR should be calculated according to the mapping of MTs, so the complexity of this operator is O($n(m+v)$). Then the immune selection will decide if the new populations can be kept in the next generation according to the choosing rules of simulated annealing. For n new populations (at most) and v tasks, the complexity is O(nv).

Based on IA, the complexities of additional strategies in NDIA are also shown in Table 2. Owing to the calculation of hamming distance among populations, the niche strategy's complexity is O(n^2). The dynamic vaccination in each evolutionary cycle is O(m), and the pheromone updating strategy is O(mv). So in theory, the additional operators in NDIA did not increase the complexity of the algorithm.

The complexities of above-mentioned four algorithms when $n \to \infty$ and $m \to \infty$ and $v \to \infty$ are also proposed as Table 2 shows. If the population size of the algorithms is large, the complexity of ACO (O(n)) is the lowest. When the scale of CRs $m \to \infty$, then the lowest complexity is O(1) in GA. However, when the scale of MTs $v \to \infty$, then the complexities of the four algorithms are all the same (i.e., O(v)).

Table 2 The time complexities of the three algorithms

Algorithms	The time complexities of operators			$n \to \infty$	$m \to \infty$	$v \to \infty$
GA	Selection O(n^2)	Crossover, O(n)	Mutation, O(nv)	O(n^2)	O(1)	O(v)
ACO	Ants' routing, O(nmv)		Pheromone updating, O(mv)	O(n)	O(m)	O(v)
IA	Genetic evolution (pending)	Vaccination, O($n(m+v)$)	Immune selection, O(nv)	O(n^2)	O(m)	O(v)
NDIA	IA evolution (pending)	Dynamic vaccination and pheromone updating, O(mv)	Niched strategy, O(n^2)	O(n^2)	O(m)	O(v)

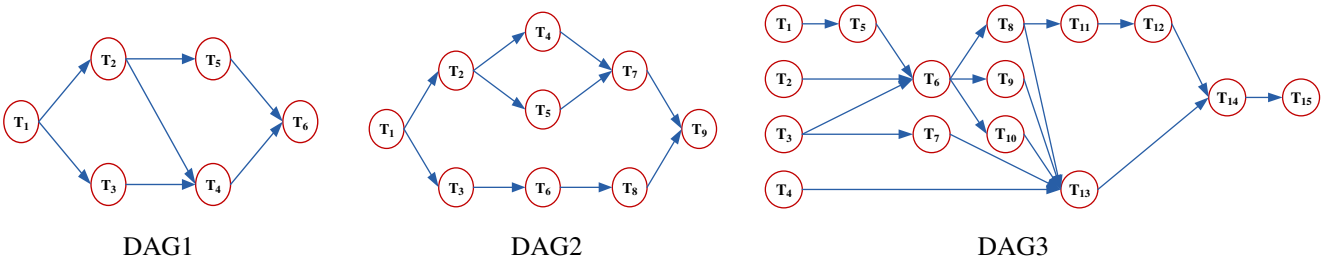

Fig. 10 DAGs of the selected MTs

7 Experiments and discussion

For testing the OACR models, the DAG in Fig. 4 and other two kinds of DAGs (the e-Economic DAG and the e-Protein DAG) introduced from [52] are selected, as shown in Fig. 10. Based on the above-mentioned four algorithms and the new OACR models, the *CCR* [35] is introduced as the testing factor in this paper. It is defined as the communication-to-computation ratio.

$$CCR = \frac{\sum_{e \in E} c(e)}{\sum_{n \in V} w(n)} \tag{19}$$

In the experiments, all of the communication costs and the computation costs are randomly generated. Due to the looser communication time constraints, the experiments focus mainly on the effect of the computing speed, memory, and reliability constraints of CRs, as Eq. 17. Other information of CRs (e.g., the bandwidths of communication routes in group among CRs and the bandwidths among groups) is generated before allocation, and they are all constant value during the solution process. The extraction of the a priori knowledge and the effect of constraints would be tested in three cases: $CCR=1/10$, $CCR=1$, and $CCR=10$ in different MTs' DAGs.

More specifically, it assumed that there are 20 available CRs in four groups separately, and group. One represents the stand-alone CRs group. The quantities of CRs per group are $\{7, 6, 4, 3\}$. According to Eq. 15, the best fitness values in tests are the inverse of the minimum make spans of MTs. So the optimal objection is finding the maximum fitness value. Because the make spans of MTs (in seconds) are usually big, the parameter *Const* in Eq. 15 is set as 1,000 to make the object function results not too small. Then the units of best fitness value in the experiments are the reciprocal of millisecond. In the algorithm, the maximum time of iteration is set to be 1,000, and the population size is set to be 50. A total of 100 runs of each experimental setting are conducted, and the average fitness of the best solutions throughout the run is recorded.

7.1 The design of the heuristic information in the intelligent algorithms

Owing to GA does not need the heuristic information, in this section, experiments are just be carried out on ACO, IA, and NDIA. According to Eq. 17, choosing a set of suitable parameters α, β, and γ is critical in these three algorithms which need the direction of the heuristic information, and in the parameter sets α, β, and γ, low value indicated the low effect in heuristic. Different heuristics may lead different results. With multiple constraints, how to extract suitable heuristic information for better solutions in different situation is very important.

In this experiment, the initial pheromone value of ACO is 1 and its evaporation factor is 0.5. The rates of pheromone and heuristic information in ACO are 1 and 5 separately, and in IA, the crossover and mutation rates are 0.8 and 0.15 separately. Then the initial annealing temperature and its decay factor are 100 and 0.95 separately. Based on the preferences in IA, the rates of pheromone and heuristic information in Eq. 18 are the same as ACO (i.e., $\varphi=1$, $\phi=5$).

Table 3 and Fig. 11 visualize the effect of different heuristic information on the average minimum make span of MTs figured by the three algorithms. In these experiments, DAG1 is adopted and tests are carried in three situations of *CCR*.

First, in low communication situation ($CCR=1/10$), the set (1, 0.5, 1) can get the best results while the set (0.5, 1, 1) can get the worst, that means low bandwidth information with high speed and reliability information can guide the algorithms to a better solution, and with low computing speed information, the algorithms are leaded to worse solutions. This is quite reasonable that computing speed is the most important information and bandwidth is the least important one. Because in this situation, computation accounted for larger proportion and then the effect of bandwidth is minor.

Second, in medium communication situation ($CCR=1$), it can be seen that the heuristic with low reliability can get the best results. By now, both computation and communication in MTs are important. Bandwidth and computing speed as

Table 3 Experiment results with different heuristic parameters

(α, β, γ)	Average minimum make span of MTs (when CCR=1) (m s^{-1})			
	(0.5, 1, 1)	(1, 0.5, 1)	(1, 1, 0.5)	(1, 1, 1)
ACO	8.4378	8.6126	8.7284	8.5469
IA	8.7347	8.7962	8.8082	8.7593
NDIA	8.8455	8.8863	9.0034	8.8773
(α, β, γ)	Average minimum make span of MTs (when CCR=1/10) (m s^{-1})			
	(0.5, 1, 1)	(1, 0.5, 1)	(1, 1, 0.5)	(1, 1, 1)
ACO	1.8891	2.0065	1.9658	1.9422
IA	2.0678	2.1012	2.0913	2.0787
NDIA	2.1006	2.1277	2.1201	2.1139
(α, β, γ)	Average minimum make span of MTs (when CCR=10) (m s^{-1})			
	(0.5, 1, 1)	(1, 0.5, 1)	(1, 1, 0.5)	(1, 1, 1)
ACO	1.456	1.4163	1.4371	1.4299
IA	1.5961	1.5294	1.5481	1.572
NDIA	1.6914	1.6286	1.6392	1.6469

their direct influencing factors are equally important. So reducing the rate of the indirect acting factor (the reliability information) emphasizes that the other two can promote searching efficiency in algorithms. However, with the same proportion of these three kinds of information, worse results would be gotten as the result of the interference of unimportant factor.

Third, in high communication situation (CCR=10), bandwidth information seems to be the most important information with the high proportion of communication in MTs. At this time, low computing speed information can lead a better evolution, and when bandwidth heuristic is lower the solution is lower, too. Hence the reliability heuristic is also an important factor in this situation. According to the constraints description in Section 5.3, the reliability factor only effects the communication time when allocating CRs for MTs. So it has large influence in high communication situation and has small influence in low communication situation, just as shown in Fig. 11.

Therefore, we can draw a conclusion that the computation speed influences much in low communication situation while the bandwidth and reliability have large effect in high communication situation, but in all of the three situations, equal proportions of three factors could not obtain good solutions. With CCR=1 and its best parameter set $(\alpha,\beta,\gamma) = (1,1,0.5)$, the comparison of the four algorithms (i.e., GA, ACO, IA, NDIA) is carried in the next section.

7.2 The comparison of GA, ACO, IA, and NDIA for addressing OACR

In this experiment, algorithms are tested in three DAGs, and the preferences of GA are the same as IA and NIA, that is $p_c = 0.8, p_m = 0.15$. Figure 12 and Table 4 show the performance results of the four intelligent algorithms for addressing the OACR problems. The run time, standard deviation, average best fitness, the best solution results, and the worst solution results in 100 runs are listed.

1. Search capability

   As shown by results, in precision, NDIA get the best solutions compared with the other three algorithms while IA takes the second place, and the standard GA is the worst. In the aspect of the worst fitness, ACO is the best, and NDIA is the next. In ACO, ants find route from the initiation so their initiate population would not be so bad. The pheromone provides the posterior information to ants to achieve cooperation searching, but it is also easy to make the algorithms trapped into local optimum. So the best solution of ACO is not really good. In NDIA, the niche strategy after the initiation and before the selection increases the diversity of the population. Its good climbing ability makes the algorithm's worst solution in 100 runs better than others. With the incorporation of genetic evolution and niche strategy, the pheromone and dynamic vaccination in NDIA cannot only increase the robustness of the heuristics searching but also avoid the local optimum. So it can always find the best solution compared with other three algorithms.

   From the climbing ability point of view, GA is the best, NDIA is the next. However, due to the basic stochastic crossover and mutation, GA is easy to trap into local optimum and finally could not find the best solutions. On the contrary, NDIA can keep a better evolutionary trend because of its dynamic vaccination strategy. ACO is the worst just because the simple pheromone and heuristic direction cause the ants to be gathered quickly into a local optimal solution. And based on dynamic IA's evolution, the niche strategy eliminates the similar individuals and keeps the population searching new area. Thus the climbing ability of NDIA is quite good.

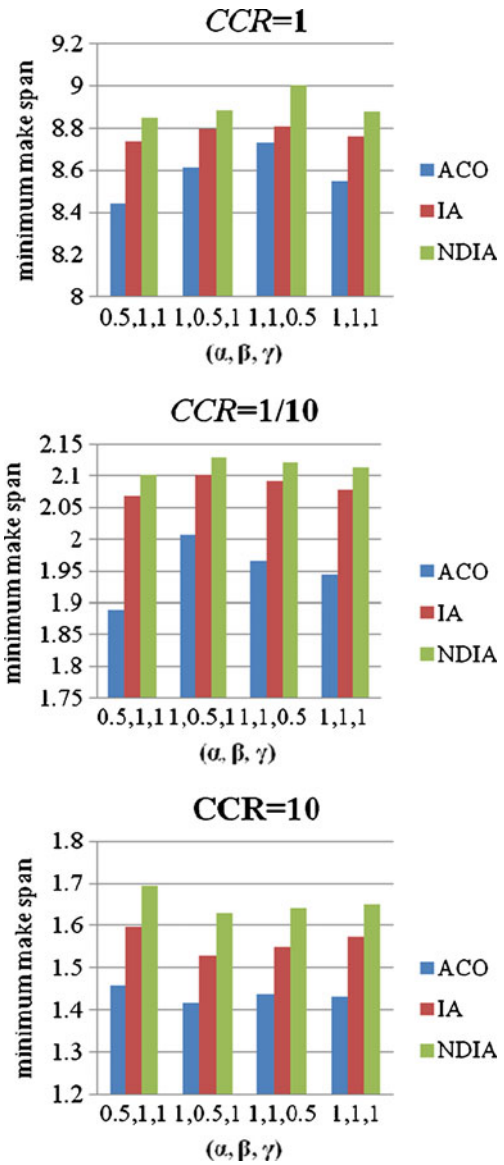

Fig. 11 The effect of different heuristic information in OACR

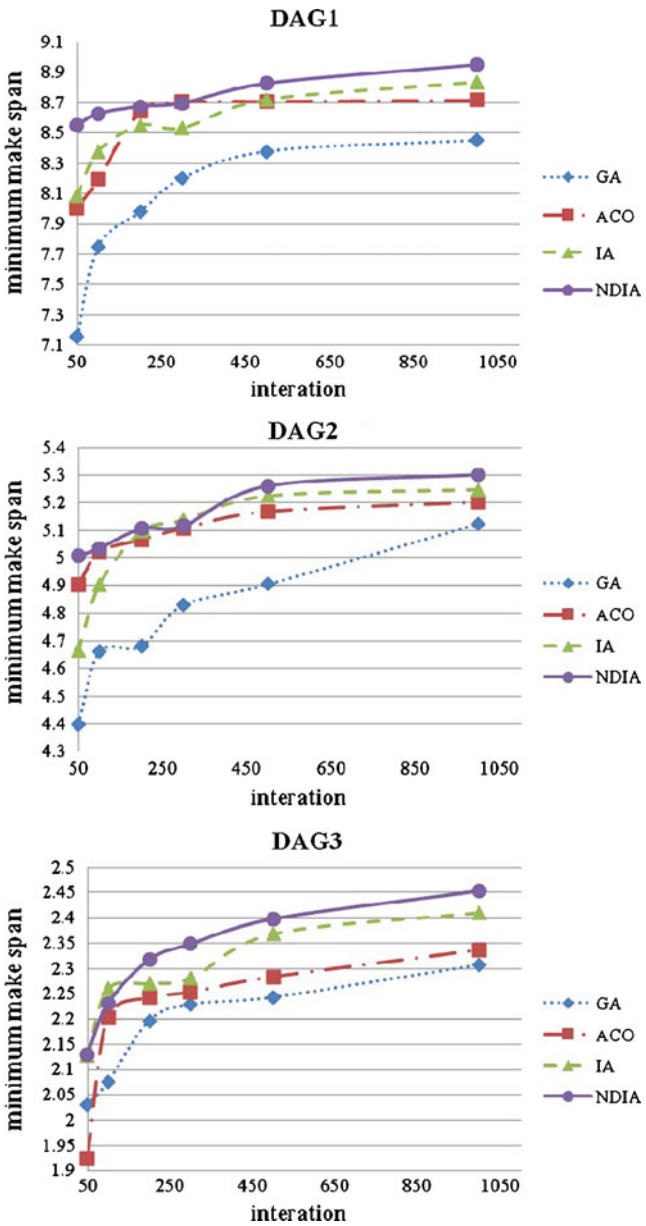

Fig. 12 Evolutionary trend of the four intelligent algorithms for addressing OACR

With the increase of the MTs' scales (from DAG1, DAG2 to DAG3), ACO's searching ability decreases along with the increase of the problem scale while the IA's searching ability increases. Stochastic vaccination added the heuristic guidance to genetic evolution and makes IA avoid the degradation of the population. At the same time, the convergence speed of NDIA becomes slower. That is due to the niche strategy, too, but compared with other three algorithms, it always can search the best solutions at certain times.

2. Stability

From Table 4 it can be seen that GA's convergence speed is slow and its stability is the worst of all. Based on the genetic strategy, IA is the next. The initiation in genetic is totally stochastic without heuristic. The evolutions in certain generations are not very stable. ACO's convergence rate is quite good. Because of the pheromone and heuristic, ants can always gather quickly to some extent. With the constant initial pheromone and heuristic, the initial paths founded by ants are fairly stable. Thus the fast convergence and stable initiation makes ACO the most stable algorithm in solving OACR. In NDIA, with the injection of pheromone, dynamic vaccination in IA could be more stable like ACO, and the ability of skipping the local optimum from the niche strategy makes it less stable than ACO.

Table 4 Performance of the four intelligent algorithms for addressing OACR

Graph number	Algorithms	The worst fitness	The best fitness	Average fitness	Standard deviation	Time
DAG1 (100 runs)	GA	7.5653	9.3467	8.4485	0.4262	163.13
	ACO	8.1071	9.1214	8.716	0.218	3,250.44
	IA	7.987	9.5617	8.8352	0.403	226.86
	NDIA	8.0657	9.5617	8.9529	0.3074	811.55
DAG2 (100 runs)	GA	4.6547	5.7213	5.1237	0.2958	255.3
	ACO	5.0828	5.5556	5.2184	0.1313	6,731.88
	IA	4.7512	5.8265	5.2483	0.2574	326.61
	NDIA	4.7775	5.9687	5.3008	0.2086	1,324.07
DAG3 (100 runs)	GA	1.7043	2.6405	2.3064	0.1688	419.71
	ACO	2.2205	2.6516	2.3367	0.1407	9,512.52
	IA	2.046	2.7276	2.41	0.1538	538.28
	NDIA	2.1535	2.7791	2.4533	0.1371	2,019.99

3. Time-consuming

   From the testing results it is clear that ACO is the most time-consuming algorithm. According to the analysis in Section 5.3, ACO needs to compute the heuristic values for all of CRs in every iteration. The complexity of ants' routing is O(*mnv*). With limited MTs, CRs, and populations, ACO is the most complex one compared with the other three, and with the addition of pheromone updating and niche strategy, NDIA is more time-consuming than IA and GA. However, the complexity of NDIA does not increase significantly in theory, just as shown in Table 2.

From the global perspective, NDIA showed high performance in all scales of MTs in OACR. Niched strategy improved the algorithm's exploration, and the introduction of experiential pheromone and dynamic heuristics improved the algorithm's exploitation. NDIA got a better balance between exploration and exploitation by these two strategies for addressing OACR in CMfg. From the perspective of solution quality and stability, NDIA has big potential in solving this kind of allocation problem without the increase of time complexity.

8 Conclusions and future works

Optimal allocation of computing resources is one of the most important and basic problem in CMfg. Current works related to the allocation (scheduling) model and algorithms are either unsuitable or inefficient. Based on the authors' previous works on service-oriented manufacturing such as manufacturing grid [66–74], this paper presented a new model with considering the characteristics of CMfg thoroughly and then designed a highly efficient intelligent algorithm for OACR in CMfg. In detail, the primary works and contribution of this paper can be concluded as follows:

1. In the new OACR model, user's interaction (control and supervision) in manufacturing tasks was fully considered for the first time. The difference between coarse-grained MTs and traditional processor/thread tasks is clearer, and it is more practical to simulation and manufacturing processes in application.
2. From the computing aspect, dynamic computing speed was presented associated with processor memory. This technique can clearly reflect the characteristics of resource partitioning in virtualization. Then, from the communication aspect, new cache technique for avoiding data surge, local and remote communication, and the communication reliability (rate and recovery time) are introduced in the OACR model. These factors are firstly fused together and make the model more close to the real integrated system in CMfg. With the full consideration of dynamic computation and communication, the process of OACR in CMfg can be more flexible and practical.
3. For solving the new complex model, an improved new intelligent algorithm (NDIA) is presented with the introduction of other three most commonly used intelligent algorithms (i.e., GA, ACO, IA). By the introduction of efficient strategies, NDIA showed high performances in terms of searching ability, stability, and time complexity in solving the OACR problem compared with the other three algorithms. At the same time, extensive experiments demonstrated the design of heuristic information in OACR and the suitable heuristics in different situations.

Future work includes intensive study on CMfg environment and research on the details of the relations among CRs'

computation and communication factors, such as the influence of memory volume on computing speed and the characteristics and differences of local communication and remote communication. Due to the NP-completeness characteristic, how to find a feasible solution which can satisfy all users demand with the minimum resources utilization in a shorter time is the most important thing in resources management in CMfg. So it would also be very interesting and important to find more efficient algorithms for OACR. Researches into algorithms not only needed in the operation improvement but also needed in the perspectives of parallelization and simplification for achieving agile allocation in CMfg.

Acknowledgment This work is partly supported by the Fundamental Research Funds for the Central Universities, National Hi-Tech. R&D (863) Program (No.2011AA040501) and the NSFC projects (No.61074144 and No.51005012) in China.

References

1. Yusuf YY, Sarhadi M, Gunasekaran A (1999) Agile manufacturing: the drivers, concepts and attributed. Int J Prod Econ 62(1–2):33–43
2. Flammia G (2001) Application service providers: challenges and opportunities. IEEE Intel Syst Appl 16(1):22–23
3. Tao F, Hu YF, Zhou ZD (2008) Study on manufacturing grid & its resource service optimal-selection system. Int J Adv Manuf Technol 37(9–10):1022–1041
4. Li BH, Zhang L, Wang SL, Tao F, Cao JW, Jiang XD, Song X, Chai XD (2010) Cloud manufacturing: a new service-oriented networked manufacturing model. Comput Integr Manuf Syst 16(1):1–16
5. Tao F, Zhang L, Venkatesh VC, Luo YL, Cheng Y (2011) Cloud manufacturing: a computing and service-oriented manufacturing model. Proc Inst Mech Eng B, J Eng Manuf 225(10):1969–1976
6. Zhang L, Luo YL, Tao F, Ren L, Guo H (2010) Key technologies for the construction of manufacturing cloud. Comput Integr Manuf Syst 16(11):2510–2520
7. He K, Zhao Y (2005) Research of grid resource management and scheduling. J WuHan Univ Technol (Inf Manag Eng) 27(4):1–5
8. Ullman JD (1975) NP-complete scheduling problems. J Comput Syst Sci 10(3):384–393
9. Zhang L, Luo YL, Fan WH, Tao F, Ren L (2011) Analysis of cloud manufacturing and related advanced manufacturing models. Comput Integr Manuf Syst 17(3):458–468
10. Li BH, Zhang L, Chai XD, Tao F, Luo YL, Wang YZ, Yin C, Huang G, Zhao XP (2011) Further discussion on cloud manufacturing. Comput Integr Manuf Syst 27(3):449–457
11. Tao F, Zhang L, Hu YF (2011) Resource Services Management in Manufacturing Grid System. Wiley-Scrivener Publishing, Dec. 2011.
12. Tao F, Zhang L, Luo YL, Ren L (2011) Typical characteristic of cloud manufacturing and several key issues of cloud service composition. Comput Integr Manuf Syst 17(3):477–486
13. Park J, Kang M, Lee K (1996) An intelligent operations scheduling system in a job shop. Int J Adv Manuf Technol 11(2):111–119
14. Jiao LM, Khoo LP, Chen CH (2004) An intelligent concurrent design task planner for manufacturing system. Int J Adv Manuf Technol 23(9–10):672–681
15. Liang JJ, Pan QK, Chen TJ, Wang L (2011) Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int J Adv Manuf Technol 55(5–8):755–762
16. Zou ZM, Li CX (2006) Integrated and events-oriented job shop scheduling. Int J Adv Manuf Technol 29(5–6):551–556
17. Hu PC (2005) Minimizing total flow time for the worker assignment scheduling problem in the identical parallel-machine models. Int J Adv Manuf Technol 25(9–10):1046–1052
18. Kwok YK, Ahmad I (1999) Benchmarking and comparison of the task graph scheduling algorithms. J Parallel Distrib Comput 59 (3):381–422
19. Polychronopoulos CD (1991) The hierarchical task graph and its use in auto-scheduling. Proceedings of the 5th International Conference on Supercomputing (ICS' 91).
20. Bokhari SH (1979) Dual processor scheduling with dynamic reassignment. IEEE Trans Softw Eng 5(4):341–349
21. Stone HS (1977) Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans Softw Eng 3(1):85–93
22. M Madhukar, M Leuze, L Dowdy. Petri net model of a dynamically partitioned multiprocessors system. Proceedings of the 6th International Workshop on Petri Nets and Performance Models (PNPM' 95), 1995.
23. Buyya R, Abramson D, Venugopal S (2005) The grid economy. Proc IEEE 93(3):698–714
24. Cardoso J, Sheth A, Miller J, Arnold J, Kochut K (2004) Quality of service for workflows and web service processes. Web Semant Sci, Serv Agents World Wide Web 1(3):281–308
25. Saravanan M, Haq AN (2008) Evaluation of scatter-search approach for scheduling optimization of flexible manufacturing systems. Int J Adv Manuf Technol 38(9–10):978–986
26. Chaudhry IA, Drake PR (2009) Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. Int J Adv Manuf Technol 42(5–6):581–594
27. Wang LY, Wang JB, Gao WJ, Huang X, Feng EM (2010) Two single-machine scheduling problems with the effects of deterioration and learning. Int J Adv Manuf Technol 46(5–8):715–720
28. Yang T, Gerasoulis A (1993) DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans Parallel Distrib Syst 5(9):951–967
29. Gerasoulis A, Yang T (1993) On the granularity and clustering of directed acyclic task graphs. IEEE Trans Parallel Distrib Syst 4 (6):686–701
30. Gerasoulis A, Yang T (1994) Performance bounds for parallelizing Gaussian-Elimination and Gauss-Jordan on message-passing machines. Appl Numer Math J 16:283–297
31. Jones WM, Pang LW, Ligon WB, Stanzione D (2005) Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. J Supercomput 34(2):135–163
32. Hamscher V, Schwiegelshohn U, Streit A, Yahyapour R (2004) Evaluation of job-scheduling strategies for grid computing. Grid Computing at the 7th International Conference on High Performance Computing, 191–202
33. Ememann C, Hamscher V, Yahyapou R (2002) On effects of machine configurations on parallel job scheduling in computational grids. Proceedings of the International Conference on Architecture of Computing Systems (ARCS 2002), 169–179
34. Davidovi T, Hansen P, Mladenovi N (2005) Permutation based genetic, tabu and variable neighborhood search heuristics for multiprocessor scheduling with communication delays. Asia Pac J Oper Res 22(3):297–326
35. Sinnen O, Sousa LA (2005) Communication contention in task scheduling. IEEE Trans Parallel Distrib Syst 16(6):503–515
36. Sinnen O, Sousa LA, Sandnes FE (2006) Toward a realistic task scheduling model. IEEE Trans Parallel Distrib Syst 17 (3):263–275

37. Benoit A, Marchal L, Pineau JF (2010) Scheduling concurrent bag-of-tasks applications on heterogeneous platforms. IEEE Trans Comput 59(2):202–217
38. Adam TL, Chandy KM, Dickson JR (1974) A comparison of list schedules for parallel processing systems. Commun ACM 17 (12):685–690
39. Sinnen O, Sousa LA (2004) List scheduling: extension for contention awareness and evaluation of node priorities for heterogeneous cluster architectures. Parallel Comput 30(1):81–101
40. Wu MY, Gajski DD (1990) Hypertool: a programming aid for message-passing systems. IEEE Trans Parallel Distrib Syst 1 (3):330–343
41. Sarkar V (1989) Partitioning and scheduling of parallel programs for multiprocessors (Research Monographs in Parallel Computing). MIT Press, Cambridge
42. Chen S, Eshaghian MM, Wu Y (1995) Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs. Proceedings of the International Conference on Parallel Processing
43. Yang L, Gohad T, Ghosh P, Sinha D, Sen A, Richa A (2005) Resource mapping and scheduling for heterogeneous network processor systems. Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems (ANCS' 05), 19–28
44. Weng N, Wolf T (2005) Profiling and mapping of parallel workloads on network processors. Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC), 890–896.
45. Huang JG, Chen JE, Chen SQ (2004) Parallel-job scheduling on cluster computing system. Chin J Comput 27(6):765–771
46. Huang JG (2008) Approximation algorithm on multi-processor job scheduling. Comput Eng Appl 44(32):26–28
47. Yin GF, Luo Y, Long HN, Cheng EJ (2004) Genetic algorithms for subtask scheduling in concurrent design. J Comput-Aided Des Comput Graph 16(8):1122–1126
48. Correa RC, Ferreira A, Rebreyend P (1999) Scheduling multiprocessor tasks with genetic algorithms. IEEE Trans Parallel Distrib Syst 10(8):825–837
49. Tsai JT, Liu TK, Ho WH, Chou JH (2008) An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. Int J Adv Manuf Technol 38(9–10):987–994
50. Chen YW, Lu YZ, Yang GK (2008) Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling. Int J Adv Manuf Technol 36(9–10):959–968
51. Wang G, Gong WR, DeRenzi B, Kastner R (2007) Ant colony optimizations for resource and timing constrained operation scheduling. IEEE Trans Comput-Aided Des Integr Circuit Syst 26(6):1010–1029
52. Chen WN, Zhang J (2009) An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans Syst Man Cybern 39(1):29–43
53. Li JQ, Pan QK, Gao KZ (2011) Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. Int J Adv Manuf Technol 55(9–12):1159–1169
54. Xu XD, Li CX (2007) Research on immune genetic algorithm for solving the job-shop scheduling problem. Int J Adv Manuf Technol 34(7–8):783–789
55. Agarwal R, Tiwari MK, Mukherjee SK (2007) Artificial immune system based approach for solving resource constraint project scheduling problem. Int J Adv Manuf Technol 34(5–6):584–593
56. Tao F, Zhao D, Hu YF, Zhou ZD (2008) Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Transactions on Industrial Informatics, 4(4):315–327.
57. Maheswaran R, Ponnambalam SG, Aravindan C (2005) A meta-heuristic approach to single machine scheduling problems. Int J Adv Manuf Technol 25(7–8):772–776
58. Jerald J, Asokan P, Saravanan R, Delphin A, Rani C (2006) Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic algorithm. Int J Adv Manuf Technol 29(5–6):584–589
59. Shukla SK, Son YJ, Tiwari MK (2008) Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. Int J Adv Manuf Technol 36(9–10):982–995
60. Zhang JX, Gu ZM, Zheng C (2010) Survey of research progress on cloud computing. Appl Res Comput 27(2):429–433
61. Hong B, Prasanna VK (2004) Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS' 04)
62. Bhat PB, Raghavendra CS, Prasanna VK (2003) Efficient collective communication in distributed heterogeneous systems. J Parallel Distrib Comput 63(3):251–263
63. Gawiejnowics S (2008) Time-dependent scheduling. Springer, Berlin
64. Wang L, Pan J, Jiao LC (2000) The immune programming. Chin J Comput 23(8):806–812
65. Wang L, Pan J, Jiao LC (2000) The immune algorithm. Acta Electronica Sinica 28(74):7–77
66. Tao F, Zhang L, Nee A Y C (2011) A review of the application of grid technology in manufacturing. International Journal of Production Research, 49(13): 4119–4155
67. Tao F, Zhao D, Zhang L (2010) Resource service optimal-selection based on intuitionistic fuzzy set and non-functionality QoS in manufacturing grid system. Knowledge and Information Systems, 25(1):185–208
68. Tao F, Zhao D, Hu YF, Zhou ZD (2010) Correlation-aware resource service composition and optimal-selection in manufacturing grid, European Journal of Operational Research, 201(1):129–143
69. Tao F, Hu YF, Zhao D, Zhou ZD (2010) Study of failure detection and recovery in manufacturing grid resource service scheduling. International Journal of Production Research,48(1):69–94
70. Tao F, Hu YF, Zhou ZD (2009) Application and modeling of resource service trust-QoS evaluation in manufacturing grid system. International Journal of Production Research, 47(6):1521–1550
71. Tao F, Hu YF, Zhao D, Zhou ZD (2009) Study on resource service match and search in manufacturing grid system.International Journal of Advanced Manufacturing Technology, 43(3-4):379–399
72. Tao F, Hu YF, Zhao D, Zhou ZD (2009) An Approach to Manufacturing Grid Resource Service Scheduling based on Trust-QoS. International Journal of Computer Integrated Manufacturing, 22 (2):100–111
73. Tao F, Hu YF, Zhao D, Zhou ZD (2009) Study on Manufacturing Grid Resource Service QoS Modeling and Evaluation. International Journal of Advanced Manufacturing Technology, 41(9–10):1034–1042
74. Tao F, Hu YF, Zhou ZD (2008) Study on Manufacturing Grid & Its Resource Service Optimal-Selection System. International Journal of Advanced Manufacturing Technology, 37(9–10):1022–1041