Chapter 10
Computing Resource Allocation with PEADGA

In this chapter, for solving optimal allocation of computing resources (OACR) problem in cloud manufacturing (CMfg) [1], serial three-layer operation configuration and parallel configuration are both applied. Firstly, A new comprehensive model for OACR is proposed in CMfg system. In this model, all main computation, communication and reliability constraints in the special circumstances are considered. Secondly, niche strategy, immune heuristics, genetic operators and pheromone strategy are configured together to generate a hybrid niche immune algorithm (NIA) [2]. Based on NIA, we introduce an adaptive full mesh exchange scheme with population supervision and get a new parallel NIA (PNIA) for addressing the specific problem. From the perspective of algorithm parallelization, the supervision of population state is encapsulated as a module used before topology-based communication as an execution condition. Then the new module is configured together with full mesh topology in different generation.

10.1 Introduction

Nowadays, in the development of manufacturing, informatization 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 informatization modes, for example, agile manufacturing (AM) [3], application service provider (ASP) [4] and manufacturing grid (MGrid) [5] and so on, are proposed and used widely. Most of them are emphasis just on how to connect distributed resources by network with less considering 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 hire the computing power and storage by using virtualization technology. It combines multiple computing

F. Tao et al., *Configurable Intelligent Optimization Algorithm*,
Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-3-319-08840-2_10

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 by Li et al. [1] to expand the service mode in manufacturing informatization and improve its dynamic. 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 on-demand. In CMfg, there's a platform which combines core technologies of cloud computing, internet of things (IoT) 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 Internet to execute various manufacturing tasks [6, 12, 13]. That is to say, in manufacturing process, 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. 10.1. 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, how to implement efficient resources allocation and integration? Actually, they are all supported by computing resources, as shown in Fig. 10.1. Computing Resources, including CPU, processor, I/O, at the physical layer [1] is the core infrastructure of 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 CMfg platform through virtualization. Information sharing needs them, manufacturing devices invoking needs them and computing/simulation work needs them, too.

In other word, under the centralized management, various heterogeneous computing resources are integrated and re-divided as virtual machines by virtualization and assigned to user on-demand for computing and simulation. Meanwhile, when manufacturing equipments access in 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 [7, 55]. High virtualization of all kinds of manufacturing hardware/software resources and high heterogeneity

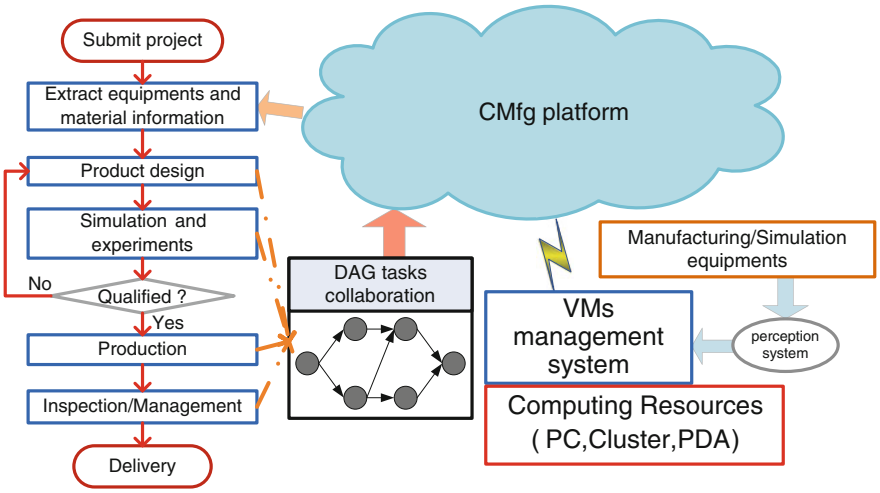

Fig. 10.1 The simplified manufacturing process in CMfg

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 multi-disciplinary 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 [8, 9] 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, focusing on OACR problem in CMfg, we proposed a systematic model for it from the point of packet communication and partition of computing power. The detailed running process of the allocation of computing resources for manufacturing tasks is shown. Classical intelligent algorithms are introduced and compared in solving the problem, and a new improved hybrid intelligent algorithm, NIA, is configured to solve OACR. Further, a new topology with pre-handling module is configured and applied in NIA. Simulation results on standard tests show that this new algorithm is pretty efficient to solve this kind of high dimensional complex problems.

10.2 Related Works

To perform larger-scale collaborative manufacturing, CMfg was firstly presented by Li et al. [1]. They specifically defined it and introduced the architecture of CMfg. Based on this, many studies about CMfg are started. Zhang et al. [7, 55] further described the key technologies for the construction of CMfg. He defined the dynamic cloud services center in CMfg as manufacturing cloud and classified it as public cloud and private cloud. Then, from the perspective of the structure of manufacturing cloud, he elaborated the types of manufacturing resources, the dynamic sensing and accessing of hardware/software and the method of information exchange in CMfg. And for further understanding and the research of CMfg, Zhang [10] then analyzed the differences and connections among CMfg and other related advanced manufacturing modes and then presented the target of CMfg, i.e., agility, servicesation, greening and intelligent in the whole manufacturing. Based on these researches, Li et al. [11, 49] specified the characteristics of CMfg and presented argument as a service (AaaS), design as a service (DaaS), fabrication as a service (FaaS), experiment as a service (EaaS), simulation as a service (SimaaS), management as a service (MaaS) and integration as a service (InaaS). These concepts is inspired by cloud computing but clearly distinguished CMfg from cloud computing. At the same time, Tao et al. [6, 12, 13] elaborated 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. All of these studies are macro-researches with less micro-analysis 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 hasn't been studied.

In manufacturing system, job-shop scheduling and workflow scheduling are much popular [61, 62] 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 pre-scheduling problem. It's more complex than several kinds of traditional job-scheduling or task scheduling problems [14–16]. In the existing task scheduling models, tasks can usually be expressed in four types: DAG (Directed Acyclic Graph) [17], HTG (Hierarchical Task Graph) [18], TIG (Task Interaction Graph) [19, 20] and Petri net [21]. The most commonly used is DAG, in which the nodes represent individual tasks and the directed arcs stand for communication overhead between tasks [22, 23]. 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 [17], and so on. The traditional DAG task scheduling problems have been proven to be NP-Complete [9]. It's far more complex in many kinds of manufacturing systems [1, 52, 63–65]. About the attributes of tasks, the concept of similarity is often

expressed as Granularity [24, 25], which indicates the ratio of communication overhead in a parallel program. The amount of communication edges is usually expressed as DAG density [26]. Besides, a variety of QoS (Quality of Service) indexes were also introduced in DAG, particularly in manufacturing task scheduling. Based on these QoS (Quality of Service) indexes, existing researches primarily focus on homogeneous cluster systems [27–29] 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 [30]. The studies about heterogeneous systems are seldom. Typically, end-point and network communication contention in heterogeneous systems are analyzed by O Sinnen [31]. The communication preparation, overhead and involvement of processors and communication mode of task scheduling are elaborated by Sinnen et al. [32] and Benoit et al. [33], and so on.

On the scheduling algorithms side, typical deterministic algorithms are list scheduling [34–36], clustering scheduling [24, 37, 38], linear programming [39], stochastic mapping [40], and several others. Yu-Kwong and Ishfaq compared and summarized 15 types of scheduling algorithms in [17], which is widely cited. After that, a few efficient approximate algorithms [41, 42], 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) [43–46], ant colony optimization (ACO) [11, 47–49], immune algorithms (IA) [50, 51] and so on and other new heuristic approaches [52–54, 64] have been paid attention and widely applied to this kind of scheduling problems for finding the Pareto optimal solutions especially in manufacturing application field [66, 67].

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) [1]. Virtual machines not only execute high performance computing 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 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 [7, 55]. 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 are presented in this chapter.

10.3 Motivation Example of OACR

A CMfg system consists of manufacturing resources, manufacturing cloud (CMfg platform) and the whole lifecycle 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. 10.2 [7, 55]. 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 sub-tasks in accordance with the requirement number of VMs and devices and then forms them as a DAG. That means each sub-task in DAG can be executed by only one VM or one device without separation. After the task partition, manufacturing cloud need to find available resources for each sub-task and provide them as services for users. In fact, as introduced in Sect. 10.1, 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 specific the abstract workflow of task execution in CMfg as shown in Fig. 10.3 and consider the multi-disciplinary 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, multi-axis 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.

(1) *Requirement analysis of task DAG*: According to the users' requirement of DAG, analyze the communication and computation costs and the QoS (Quality of Services) constraints of tasks. Then check the accessed resources(include computing resources and manufacturing devices). If there's 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.

(2) *Optimal allocation and strategy sending*: In terms of the QoS and costs of tasks, manufacturing cloud determines which tasks need remote simulation physical devices. If the task needs physical device, then calculates the attribute values of device and maps 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.

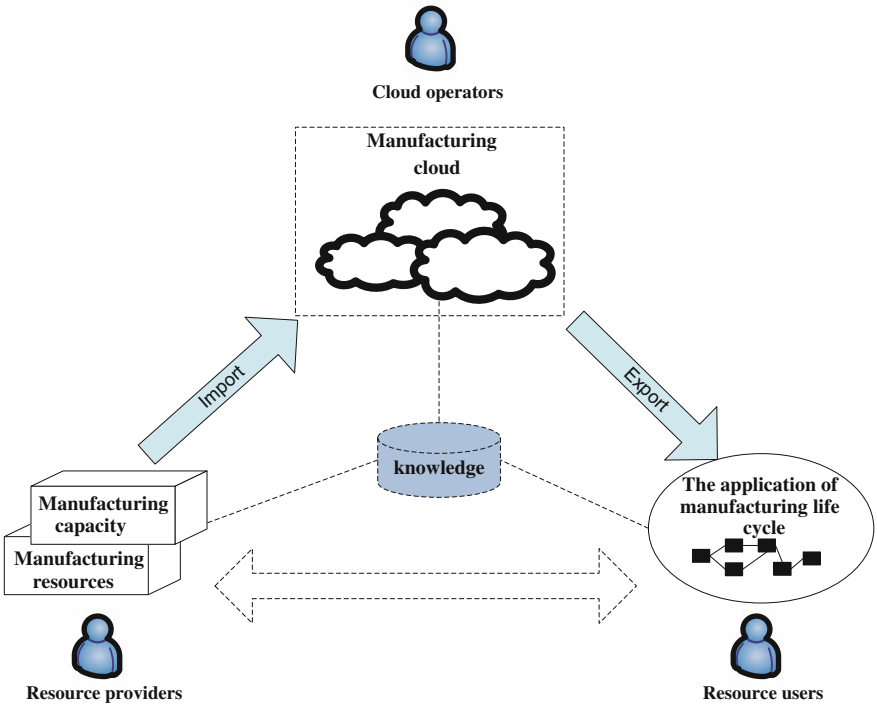

Fig. 10.2 The abstract operation principle of CMfg

(3) *Execution*: After the users' confirmation, 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 Internet. If unexpected error occurs during execution, the platform will call the fault-tolerant migration strategy automatically and try to execute tasks again.

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

10.4 Description and Formulation of OACR

According to the simplified manufacturing process shown in Sect. 10.3, 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

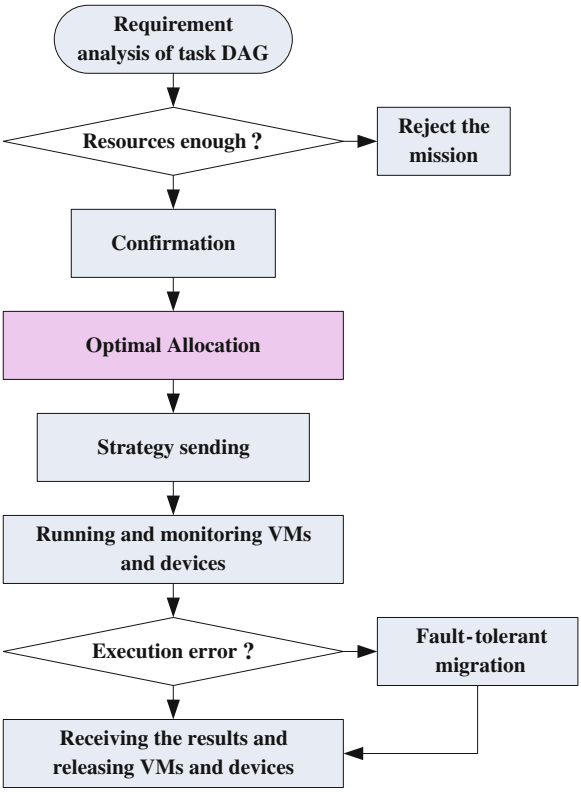

Fig. 10.3 The specific workflow of task execution

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.

10.4.1 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. 10.4.

On manufacturing task level, assume the tasks of a given MTs set are meta-tasks. *Meta-task* means that the task is inseparable for executing in VMs/CRs, as discussed in Sect. 10.3. 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 receive the demand information and allocate available VMs for

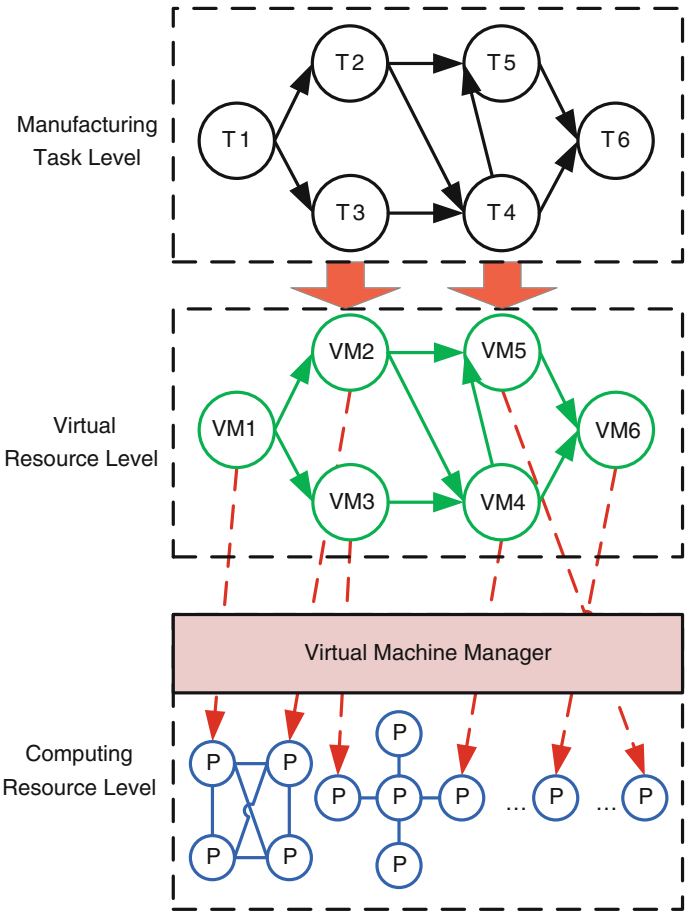

Fig. 10.4 The process framework of OACR

physical manufacturing resources. The physical manufacturing resources can be not only manufacturing/simulation equipments, but 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. 10.4, 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 chapter, 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 are emphasized.

10.4.2 The Characteristics of CRs in CMfg

In 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.

(1) Communication network
The CMfg network is different from other enterprise network or public network and compromised by many distributed manufacturing resource around the world. For the sake of facilitate 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 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 are all full-duplex.
(2) Communication ports
Generally, the port communication of master-slave system can be classified as single-port mode [56] and multi-port mode [57]. *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 *multi-port* 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, multi-port communication mode is adopted in CRs and the platform.

However, in this multi-port mode, owing to the complex and frequent intercommunication among CRs for a large number of MTs, the amount of transmit data from multi-port 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.
(3) Communication bandwidth
Associating with multi-resources around the world, the core network protocol of CMfg is still TCP/IP mode (with two-sided communication type [32]. 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 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 ten to hundreds Mbps.

10.4.3 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.

(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 directed acyclic graph (DAG), $G = (V, E, c, w)$. The set $V = \{v_i | i = 1 : v, v = |V|\}$, where v_i represents the task of the set V, and v is the cardinality of nodes. The set $E = V \times V$, $e = |E|$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 $pred(v_i) = \emptyset$ is named *source* node, and the node with no successor task $succ(v_i) = \emptyset$ 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.

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, $bw = P \times P$ 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 non-preemptive. If two tasks are carried by the same processor, their communication cost is 0, and it assumed that the transmission rate of computing resources equal to the bandwidth (the ideal value).

(2) The new models of OACR in CMfg

According to the characteristic of CRs, the uncertain topology can be simplified as shown in Fig. 10.5. 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 Internet. Stand-alone PCs can be classified as a special group. They are connected with each other directly via 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 multi-thread 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 directed acyclic graph (DAG), $G = (V, E, c, w, oper\_p, su\_p)$. The definition of $V = \{v_i | i = 1 : v, v = |V|\}$, $E = V \times V$, 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 $su = c \times su\_p$.

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's no interaction or supervision in v_i, then $oper_p(i) = 0$or $su\_p(i) = 0$. These two factors can clearly reflect the users 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 follow.

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.

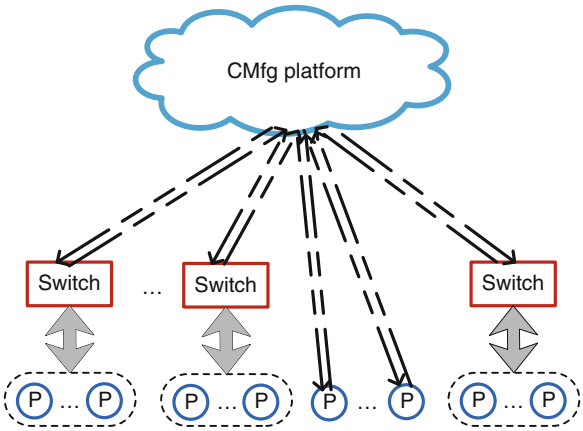

Fig. 10.5 The simplified topology of CRs

- $bw = \{bw(kl)|k = 1 : m, l = 1 : n_k\}$ represents the bandwidth between CRs and switch in each group. Considered the simplified topology (Fig. 10.2), the access bandwidths of switches to Internet are defined as $BW = \{BW_i|i = 1 : m\}$. Due to the stand-alone PCs are connected via Internet directly, let $bw_0 = \{bw_0(kl)|k = 1, l = 1 : n_0\}$ be their bandwidth to Internet. Owing to the bandwidths in groups are generally gigabit, $\forall k \in [1, m], l \in [1, n_k] \Rightarrow BW_k \ll bw(kl)$.
- $rou = \{rou_i(kl)|i = 1 : m, k \neq l, k, l \in [1, n_i]\}$ 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 $rou_i(kl) = \{link_1, \cdots, link_r\}$[32] varies with different topologies, and the bandwidth of $rou_i(kl)$ is defined as $bw(rou_i(kl)) = \min\{bw(link_1), \cdots, bw(link_r)\}$.
- $buf = \{buf(kl)|k = 1 : m, l = 1 : n_k\}$ represents the buffer size of the switch communication ports in each group. According to Davidovi et al. [30], it assumed that the highest tolerable abrupt data of each port to be:

$$D(kl) = buf(kl) + buf(kl)\frac{BW_k}{bw(kl) - BW_k} = buf(kl)\frac{bw(kl)}{bw(kl) - BW_k} \tag{10.1}$$

- $rel = \{rel(kl) = (rel\_p(kl), rep\_t(kl))|k = 0 : m, l = 1 : n_k\}$ 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.

(3) The constraints and objective function of OACR

Based on the structure described before, four issues of CRs are considered in this chapter.

(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;
(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 a MTs set $G = (V,E,c,w,oper\_p,su\_p)$ is applied to 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 $EXE\_TIME(i) = W(i) \times p\_load(k(i)l(i))/s(k(i)l(i))$;
- $\forall i \in [1,v]$, 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:

$$COM\_TIME\_s(ij) = COM\_TIME\_r(ij) = COM\_TIME(ij) = c(ij)/rou_k(l(i)l(j)) \tag{10.2}$$

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 $p_{k(j)l(j)}$ doesn't 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:

$$t(x) = (1 - rel\_p(k(j)l(j)) * (1 + t(x-1)) + rel\_p(k(j)l(j)) * (t(x) + rep\_t(k(j)l(j))) \tag{10.3}$$

$$t(x) = 1 + t(x-1) + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))} \tag{10.4}$$

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

$$t(x) = x(1 + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))}) \tag{10.5}$$

According to Eq. 10.5, the communication time between v_i and v_j can be expressed as:

$$COM\_TIME(ij) = (1 + \frac{rel\_p(k(j)l(j)) * rep\_t(k(j)l(j))}{1 - rel\_p(k(j)l(j))}) * \frac{c(ij)}{rou_k(l(i)l(j))} \tag{10.6}$$

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:

$$COM\_TIME\_s(ij) = c(ij)/bw(k(i)l(i)) \tag{10.7}$$

and the receiving time of task v_i is equal to:

$$COM\_TIME\_r(ij) = \left\{ \begin{array}{l} \frac{c(ij)}{\min\{BW_{k(i)}, BW_{k(j)}\}}, k(i) \neq k(j) \neq 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_{k(j)}\}}, k(i) = 0 \\ \frac{c(ij)}{\min\{BW_{k(i)}, BW_0(l(j))\}}, k(j) = 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)), BW_0(l(j))\}}, k(i) = k(j) = 0 \end{array} \right\} = COM\_TIME(ij) \tag{10.8.}$$

After adding the reliability factors, according to Eq. 10.5, the sending time of task is unchanged, but the receiving time is changed as:

$$\begin{aligned} COM\_TIME\_r(ij) &= (1+\frac{rel\_p(k(j)l(j))*rep\_t(k(j)l(j))}{1-rel\_p(k(j)l(j))}) \\ &* \left\{ \begin{array}{l} \frac{c(ij)}{\min\{BW_{k(i)},BW_{k(j)}\}}, k(i)\neq k(j)\neq 0 \\ \frac{c(ij)}{\min\{BW_0(l(i)),BW_{k(j)}\}}, k(i)=0 \\ \frac{c(ij)}{\min\{BW_{k(i)},BW_0(l(j))\}}, k(j)=0 \\ \frac{c(ij)}{\min\{BW_0(l(i)),BW_0(l(j))\}}, k(i)=k(j)=0 \end{array} \right\} \\ &= COM\_TIME(ij) \end{aligned} \tag{10.9}$$

- If $c(ij) > D(k(i)l(i))$, the sending rate of task v_i must be reduced. According to Eq. 10.1, 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. 10.9, 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:

$$START\_TIME(j) = \max_{i\in pred(v_j)}\{COM\_TIME\_r(ij)\} \tag{10.10}$$

$$\begin{aligned} FINISH\_TIME(j) &= START\_TIME(j) + EXE\_TIME(j) \\ &+ \max_{i\in succ(v_j)}\{COM\_TIME\_s(ij)\} \end{aligned} \tag{10.11}$$

Therefore the temporal relation between two adjacent tasks is as shown in Fig. 10.6. 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:

$$\begin{aligned} COM\_TIME(v) &= TASK\_SUBMISSION\_TIME(V) \\ &= \left\{ \begin{array}{ll} \frac{c(v)-buf(k(v)l(v))}{BW_{k(v)}} & , if\ c(v) > D(k(v)l(v)) \\ \frac{c(v)}{BW_{k(v)}} & , if\ c(v) < D(k(v)l(v)) \end{array} \right. \end{aligned} \tag{10.12}$$

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

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

$$TOTAL\_TIME(V) = FINISH\_TIME(v) - START\_TIME(1) = FINISH\_TIME(v) \tag{10.13}$$

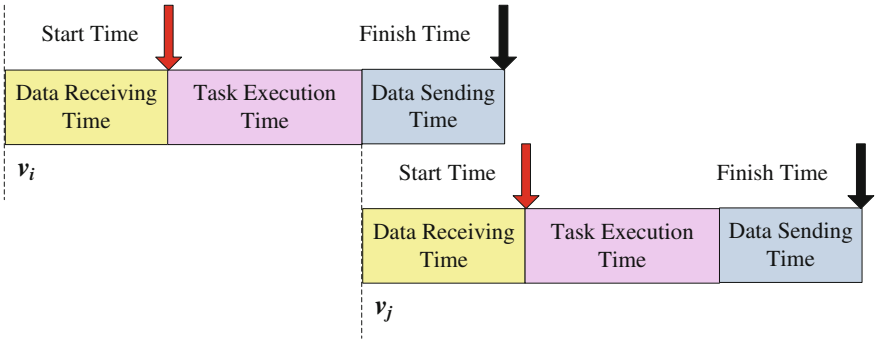

Fig. 10.6 The temporal relation between two MTs

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:

$$\begin{cases} MINIMIZE\ TOTAL\_TIME(V)\ SUBJECT\ TO \\ \forall i \in [1, v],\ \frac{s(k(i)l(i))}{p\_load(k(i)l(i))} \geq EXE\_SPEED_{\min}(i) \\ \forall i, j \in [1, v],\ COM\_TIME(ij) \leq COM\_TIME_{\max}(ij) \end{cases} \tag{10.14}$$

(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 Gawiejnowics [58].

Definition 5 [58] (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;
(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 [58] (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 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 multi tasks.

(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 meta-tasks and N_p resources turn into the traditional scheduling problem. In this situation the OACR problem can be reduced to the traditional scheduling problem. According to definition 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 meta-tasks and v computing resources can be converted to TSP (Traveling Salesman Problem), 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 some one want 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're 100 CRs and 5 MTs, the solution space is 5^{100}. It's a very huge number for calculation. At present, no deterministic algorithms can solve it in polynomial time. So, intelligent algorithms are introduced in this chapter.

10.5 NIA for Addressing OACR

The most frequently used intelligent algorithms for the traditional task scheduling are genetic algorithm (GA) [43, 44] and ant colony optimization algorithm (ACO) [47, 48]. Besides, immune algorithm (IA) has shown great potential in combinatorial optimization problems [59, 60]. They are widely used in various kinds of scheduling problems and their basic processes are shown in Fig. 10.7. 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 chapter for better solutions. These four algorithms will then be used and generally analyzed for addressing the OACR problem in detail.

10.5.1 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

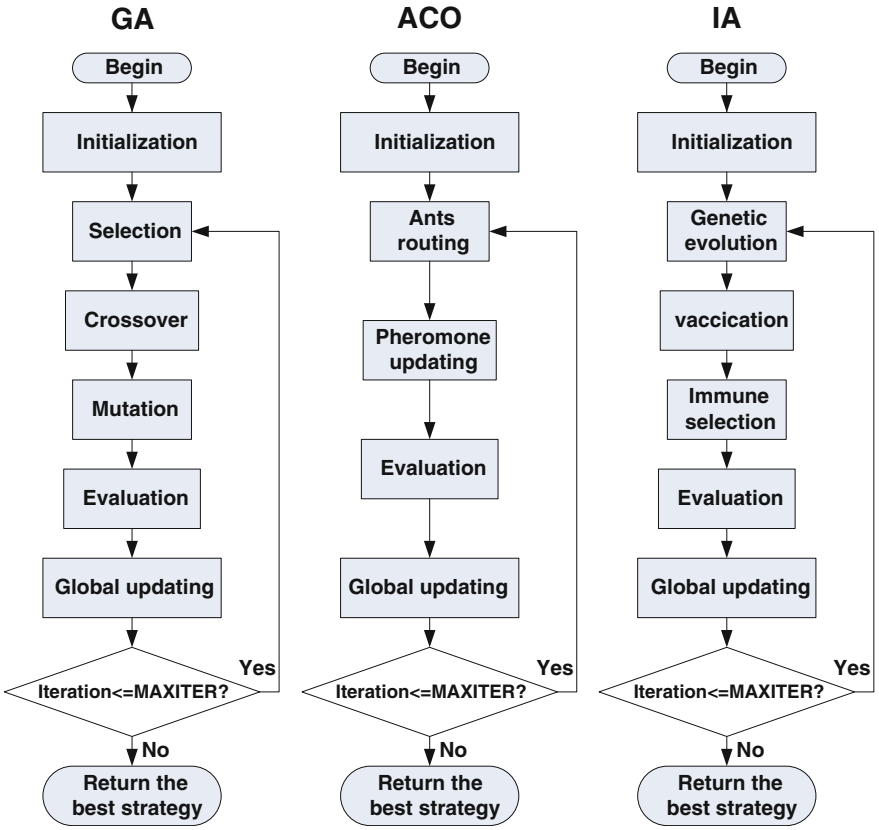

Fig. 10.7 The process of GA, ACO and IA

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 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 maximize generations are reached or the optimal solution is found.

ACO is a kind of swarm intelligence algorithms 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

Table 10.1 The characteristics of GA, ACO and IA

Algorithm	Year	Mechanism	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

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 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 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 reach 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 Lei Wang [59, 60] is a typical and efficient one. In this chapter, it will be applied in OACR and IA here just indicates the algorithm in [59, 60].

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 priori knowledge into the new populations in some degree for improving their fitness values. (The 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 10.1, only GA does not need the priori knowledge with less control parameters. Without other improvement strategies, its global convergence is weak, but its robustness is quite good. ACO and IA are both need the direction of priori knowledge with good global convergence. Yet the control parameters of ACO are more than IA's.

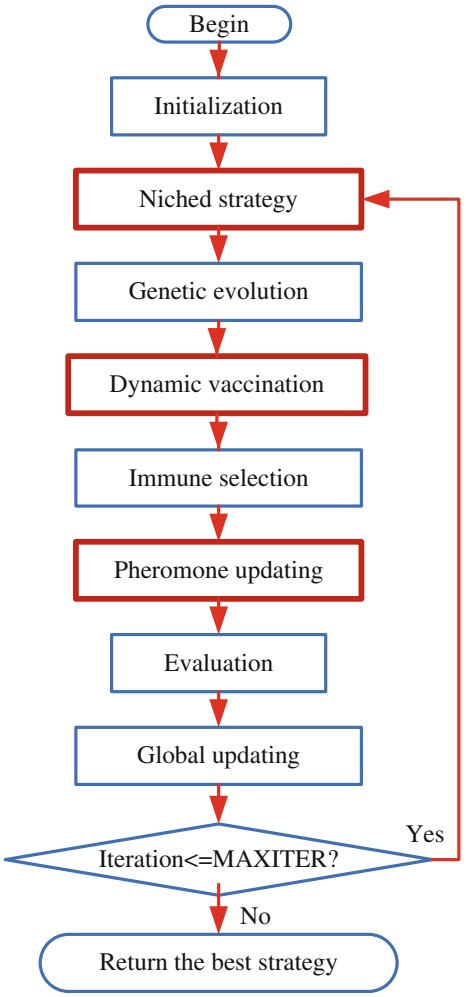

Fig. 10.8 The framework of NDIA for addressing OACR

10.5.2 The Configuration OfNIA 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. 10.8.

Compared with IA (as shown in Fig. 10.7), the niche strategy, 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 taked 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,

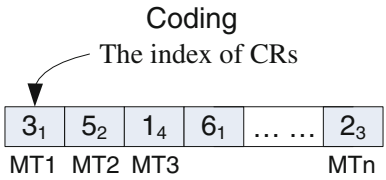

Fig. 10.9 The real number coding for OACR

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. 10.9, 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 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:

$$\max f = \frac{Const}{TOTAL\_TIME(V)} \tag{10.15}$$

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

(3) Niche strategy

For improve 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. 10.16.

$$D_{ij} = \left\| X_i - X_j \right\| = \sqrt{\sum_{k=1}^{N} (x_{k,i} - x_{k,j})^2} \tag{10.16}$$

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 pre-set 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 priori knowledge (i.e. heuristic information) of the problems. If the priori knowledge is extracted inappropriately, the algorithm will evolve in the wrong direction and no feasible solution can be found. However, the 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 needs 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:

$$\eta_{ij} = \frac{\left(\frac{s(k(i)l(i))}{p\_load(k(i)l(i))}\right)^{\alpha} * (bw(\mathrm{k(i)}l(i)))^{\beta}}{\left(1 + \frac{rel\_p(k(j)l(j) * rep\_t(k(j)l(j)}{1 - rel\_p(k(j)l(j)}\right)^{\gamma}} \tag{10.17}$$

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 $\alpha, \beta, \lambda \in [0, 1]$. In experiments, the value of these parameters will be tested and discussed for better solution.

Besides, for further improving the searching direction, the pheromone of ACO is brought in NDIA in this chapter. That is to say, the improved NDIA extracts antigen and vaccine not only by the 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 priori knowledge (i.e. the heuristic factor) of the problem. The vaccine can then be expressed as:

$$vaccine = (\tau_{ij})^{\varphi} (\eta_{ij})^{\phi} \tag{10.18}$$

where τ_{ij} is updating as the same as in ACO, and $\varphi, \phi \in (0, 1)$ 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.10.17 in this chapter are set the same as ACO (i.e. $\varphi = 1$, $\phi = 5$).

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.

10.5.3 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 10.2.

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 10.2 just mean the worst case complexities.

In ACO, because 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 10.2. 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 10.2 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 10.2 The time complexities of the three algorithms

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

10.6 Configuration and Parallelization of NIA

As mentioned in Chap. 5, there are many topologies for the parallelization of intelligent optimization algorithm. No matter with large-scaled or small-scaled computing resources, communication on one hand is a critical issue for preserving the overall performance of parallel algorithm, on the other hand is also a decisive factor the total time consumption of searching process. Therefore, the control of individual exchange is the most important thing in the parallelization configuration.

As is known, during the evolutionary process, if the sub-populations in different nodes are consistent and distributed in a small solution space, the communication is needless. Too frequent exchange in this situation is very likely to get premature convergence. On the contrary, if the sub-populations in different nodes are dynamic and distributed in a large solution space, then the communication is required. In this case, foreign excellent individuals will always bring good information and push the whole population evolving in a good direction. Otherwise, sub-population will do evolution on their own with a lot of repetitions and no convergence at all. Inspired by the use of prior knowledge in many adaptation strategies, we present a new adaptive ways for control the communication step and make exchange only when the whole population is high diversity.

Firstly we need to select a suitable connected topology.

1. Connected topology

The common used topologies include ring, grid and full-mesh and so on. Normally, with fixed number of individuals in the whole population, dense connection topologies such as full-mesh and grid can obtain higher collaborationamong subgroups but with higher communication overhead,vice versa. However, based on general parallel tools - MPI (Message Passing Interface), we found that-MPI_Allgather (the full-connected interaction way) is more efficient than the use of MPI_Send/MPI_Recv or MPI_Isend/MPI_Irecvto implement full communications during$n$ nodes due to the inside optimization by the tools itself. In this situation, full-mesh is quite advantageous.

Thus, full-mesh module is selected to generate the new PNIA. Nonetheless, full-mesh cannot make sure low time consumption during iterations. Considering the time cost of MPI_Allgather, the design of efficient migration mechanism is imperative.

2. New adaptive strategy

In each period, whether to exchange individuals is decided by the overall evolutionary state of the sub-populations. Specifically, migration is not always needed in iterations. If the sub-populations execute many times of iterations but with low evolution and diversity, then migration is needed to introduce outside excellent genes to improve the quality and diversity of sub-populations. Otherwise, migration is needless at all because good evolutionary direction of sub-populations

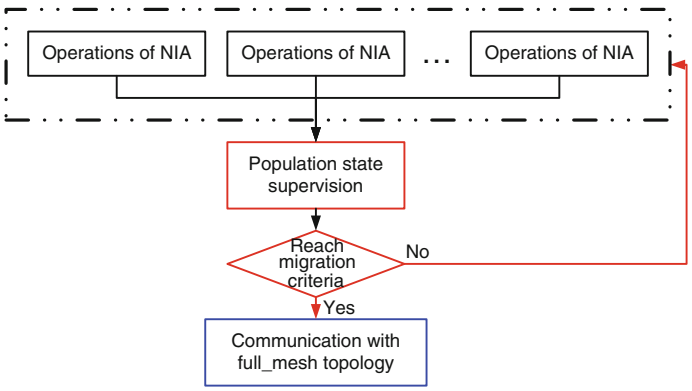

Fig. 10.10 The configuration of population supervision module with full_mesh topology

could easily be disturbed. Not only is the redundant migration wasting time, but also it does not bring good benefits.

Thus, from the whole population point of view, we set one computing node to take in charge of supervising and gathering the state parameter of all sub-populations, as well as master-slave mode, and broadcasting the overall state to all others with Map/Reduce operations in MPI. Whether to do interaction depends directly on the overall state parameter.

As we know, one of the key factors to reflect the evolutionary state is the total number of progression-free generations. It implies that, since the last generation of the improvement of historical best solutions, the generations have been executed without the change of the historical best record. Let the progression-free generation of each sub-population i be G_inva_i. If better solution has been searched in the current generation, then $G\_inva_i = 0$. Otherwise in each generation, execute $G\_inva_i = G\_inva_i + 1$. To avoid extra large value of G_inva_i, we could set $G\_inva_i \in [0, G_{\max}]$ in which G_{max} represents the upper bound of G_inva_i.

According to such variable, the evolutionary state of each sub-population can be easily obtained. With fixed topology modules in configuration, here we only need the overall evolutionary state to control the exchange process. For balance among different sub-population, set the total progression-free generations G_inva_total to be calculated as follows.

$$G\_inva\_total = \sum_{i=0}^{n} G\_inva_i \tag{10.19}$$

where n is the total number of sub-groups. If the total progression-free generation parameter becomes large,then carry out the migration with full mesh topology and specific migration mechanism. Otherwise, stop the migration and keep self-evolution going.

Moreover, set the migration condition (migration frequency) to be MT, then the migration with full-mesh topology is allowed with the probability E_c.

$$E_c = \exp(-\frac{G\_inva\_total}{n \times G_{\max}}) \tag{10.20}$$

It can be seen that if G_inva_total is smaller, $E_c$will be increased, the time of migrations will be decreased, vice versa.

From the perspective of parallelization, the specific pseudo-code of the adaptive stragey with full mesh topology can be represented as follows.

```
For (each sub-population i in parallelization)
    Initialize subpopulation
    generation = 0
    While(generation<= MAX_generation or convergence criterion satisfied)
            generation ++
            Apply algorithm's operators
            Evaluate solutions in the subpopulation
            If (generation % MT == 0)
                    If (i == supervision_node)
                            Reduce G_inva_total = sum_i G_inva_i
                            Broadcast G_inva_total
                    End if
                    If (rand()>exp(-1*G_inva_total / (n*G_max)))
                            Migration with full_mesh_topology
                    End if
            End if
    End while
```

From this, the total parallel efficiency can be improved under the mode of MPI_Allgather (full-connection topology) by means of supervision and adaptive communication.The module of population supervision can be shown in Fig. 10.10.

10.7 Experiments and Discussions

For testing the OACR models, the DAG in Fig. 10.4 and other two kinds of DAGs (the e-Economic DAG and the e-Protein DAG) introduced from [48] are selected, as shown in Fig. 10.11.

Based on the above mentioned four algorithms and the new OACR models, the *CCR* [31] is introduced as the testing factor in this chapter. It is defined as the communication to computation ratio.

$$CCR = \frac{\sum_{e \in E} c(e)}{\sum_{n \in V} w(n)} \tag{10.21}$$

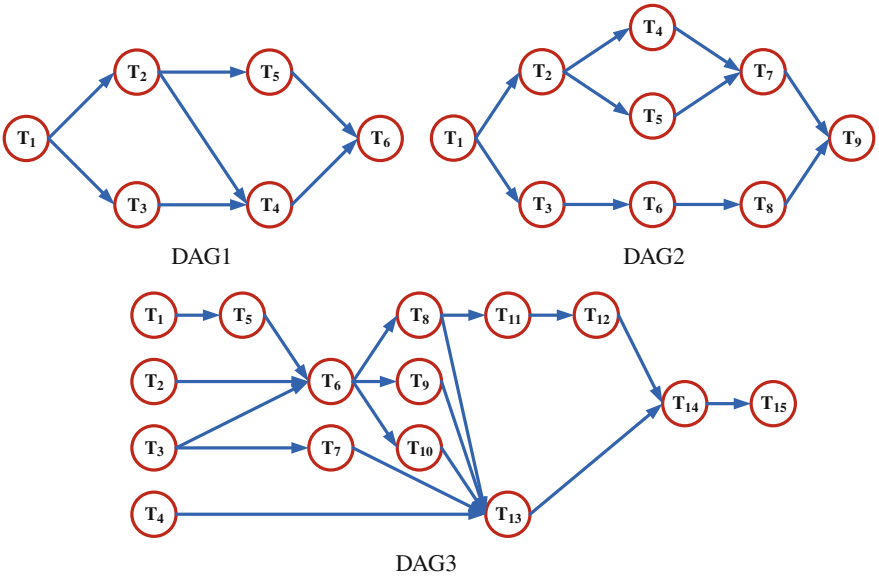

Fig. 10.11 DAGs of the selected MTs

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.10.17. Other information of CRs (e.g., the bandwidths of communication routes in group among CRs and the bandwidths among groups) are generated before allocation and they are all constant value during the solution process. The extraction of the 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 4 groups separately, and group. 1 represents the stand-alone CRs group. The quantities of CRs per group are {7, 6, 4, 3}. According to Eq.10.15, 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.10.15 is set as 1000 to make the object function results not too small. Then the units of best fitness value in the experiments arems^{-1}. In the algorithm, the maximum time of iteration is set to be 1000, 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.

Table 10.3 Experiment results with different heuristic parameters

(α, β, γ)	Average minimum make span of MTs (when $CCR = 1$) (units: ms^{-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 $CCR = 1/10$) (units: ms^{-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 $CCR = 10$) (units: ms^{-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

10.7.1 The Design of the Heuristic Information in the Intelligent Algorithms

Owing to GA does not need the heuristic information, in this section, experiments just be carried out on ACO, IA and NDIA. According to Eq. 10.17, 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, 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.10.18 are the same as ACO (i.e. $\varphi = 1$, $\phi = 5$).

Table 10.3 and Fig. 10.12 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.

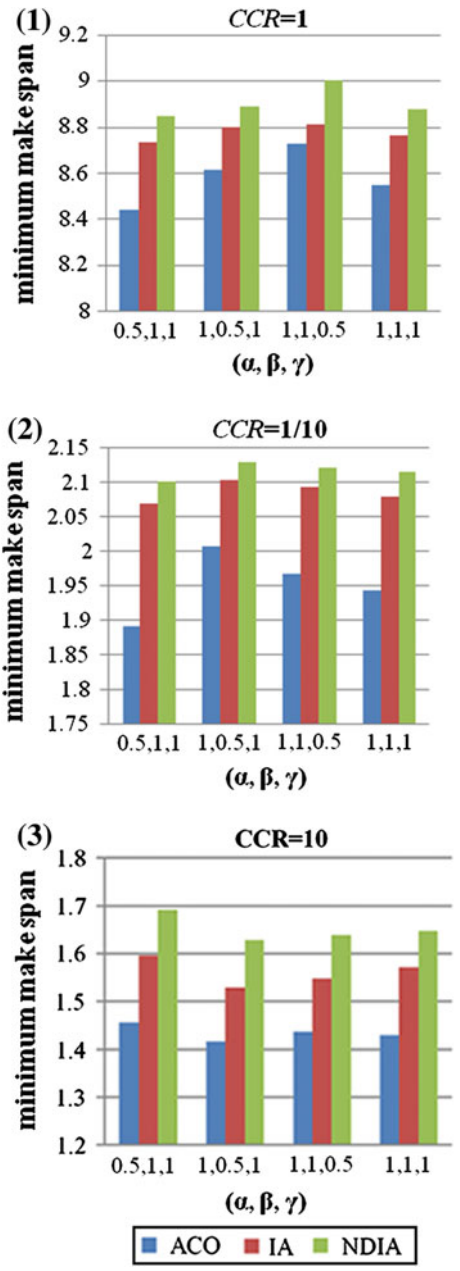

Fig. 10.12 The effect of different heuristic information in OACR

Second, in medium communication situation ($CCR = 1$), it is 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 their direct influencing factors separately are equally important. So reducing the

rate of the indirect acting factor (the reliability information) and emphasize the other two can promote searching efficiency in algorithms. However, with the same proportion of these three kind 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 Sect. 10.5.3, 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. 10.12.

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.

10.7.2 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 10.13 and Table 10.4 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 wouldn't 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 can not 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.

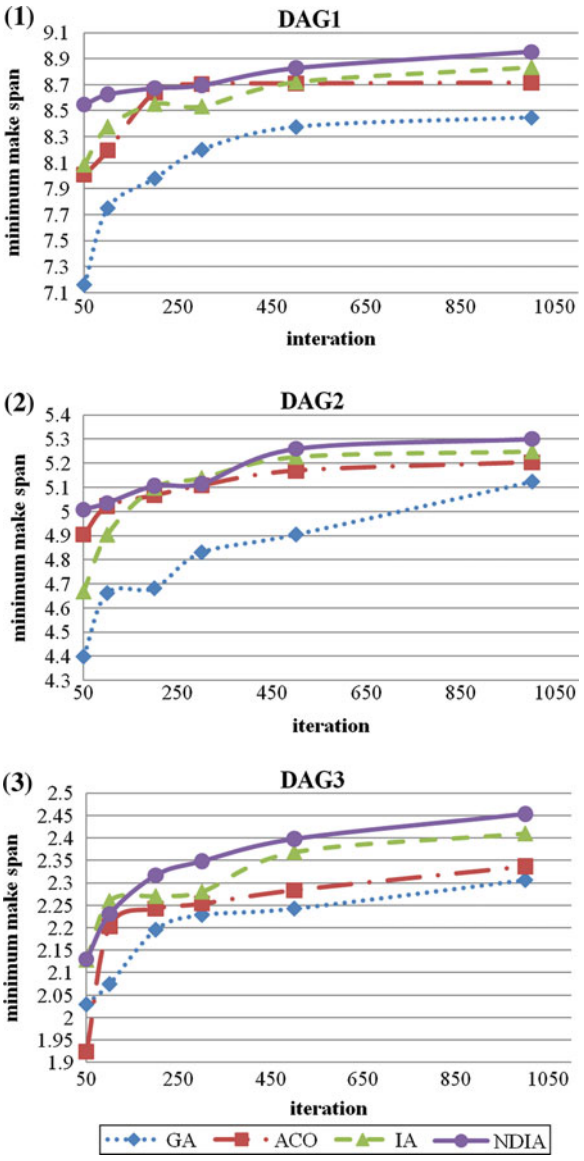

Fig. 10.13 Evolutionary trend of the four intelligent algorithms for addressing OACR

From the climbing ability point of view, GA is the best, NDIA is the next. However, due to the basic stochastic crossover and mutation, GAis easy to trap into local optimum and finally couldn't 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

Table 10.4 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	3250.44
	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	6731.88
	IA	4.7512	5.8265	5.2483	0.2574	326.61
	NDIA	4.7775	5.9687	5.3008	0.2086	1324.07
DAG3 (100 runs)	GA	1.7043	2.6405	2.3064	0.1688	419.71
	ACO	2.2205	2.6516	2.3367	0.1407	9512.52
	IA	2.046	2.7276	2.41	0.1538	538.28
	NDIA	2.1535	2.7791	2.4533	0.1371	2019.99

the ants 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.

(2) Stability

From Table 10.4 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 10.5.

(3) Time consuming

From the testing results it is clear that ACO is the most time consuming algorithm. According to the analysis in Sect. 10.5.3, 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 10.2.

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

Table 10.5 Testing results of different parallel methods in solving OACR

Mode\Proc_num	Average_fitness(1000 s^{-1})								
	1	2	3	6	9	12	15	18	21
Independent	7.6968	7.4651	7.2909	6.5734	6.9808	6.3716	6.3622	5.8155	5.6282
Ring	7.6968	7.3982	7.3849	7.4501	7.3661	7.3693	7.4034	7.3637	7.3031
Neighbor	7.6968	7.3724	7.405	7.4985	7.3782	7.4081	7.3625	7.4506	7.3154
Gridsingle	7.6968	7.3951	7.4704	7.5114	7.4066	7.4228	7.3705	7.3958	7.3265
Griddouble	7.6968	7.5106	7.4436	7.5239	7.4735	7.4859	7.3992	7.4293	7.3917
Total	7.6968	7.4814	7.5039	7.4241	7.5468	7.4285	7.4339	7.4626	7.3521
Random	7.6968	7.4287	7.5428	7.4639	7.5213	7.5036	7.4761	7.378	7.4267
Master	7.6968	7.2723	7.3367	7.3523	7.4225	7.2896	7.319	7.2639	7.2578
New	7.6968	7.4384	7.4761	7.5032	7.4862	7.5302	7.4497	7.5216	7.4539
Mode\Proc_num	Average_time(s)								
	1	2	3	6	9	12	15	18	21
Independent	23.594	6.4976	3.7991	2.0284	1.6629	1.4305	1.0965	1.0004	0.9519
Ring	23.594	10.547	7.6201	5.3207	4.4047	3.8511	3.6547	3.6573	3.3041
Neighbor	23.594	10.794	7.8972	5.5503	4.7451	4.1131	3.7188	3.6104	3.3748
Gridsingle	23.594	10.7703	7.4568	5.0895	4.2516	3.6961	3.4102	3.5991	3.2851
Griddouble	23.594	10.1286	7.684	5.1871	4.3461	3.9787	3.3344	3.6379	3.2632
Total	23.594	10.3114	7.0071	4.4832	4.2734	4.1316	3.3544	3.2454	3.0284
Random	23.594	11.0542	9.3598	5.6021	4.6448	4.1999	4.7577	5.1886	5.7958
Master	23.594	12.1286	7.932	8.8628	8.7232	8.7056	8.8172	9.5644	9.5623
New	23.594	10.7358	6.9579	5.1083	4.1406	3.6668	3.2832	3.2339	2.8566

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.

10.7.3 The Performance of PNIA

In this section, we mainly apply master-slave, independent, single-ring, double-ring, single-mesh, double-mesh, full-mesh and random topologies mentioned in Chap. 5 based on the configured NIA for addressing OACR problem in CMfg. In the experiments, only DAG3 in Fig. 10.11 is used. For uniformity, all of the above topologies are implemented with 'The best-replace-the worst' migration mechanism and in each period only one individual is migrated. Also, in this test, the total generation number is set to be 2000, *MT* is set to be 20. Other parameters still follow the settings in above section. The parallel environments are one computer with 4-cores and three computing resources with 16-cores in each.

(1) **Time consumption**

According to the tests, the time consumption of independent parallel algorithm (i.e., there is no individual exchange between any computing nodes, the results are received after iteration) is exponentially reduced along with the increase number of sub-processors (i.e. the number of sub-populations). When the number of sub-populations is below to 6, we can get linear speedup directly. When the number is increased further, the time reduction becomes less. The most consumption scheme is master-slave mode. Compared with independent scheme, the time consumption from low to high is: single-ring $<$ double-ring $<$ single-mesh $<$ double-mesh, correspondingly. The difference between them is still small. With MPI_Allgather, full-mesh topology in the experiments performs a little better than the above four topologies.

As the increase of sub-processors, random topology performs well when the processors below 6 and bounce back again when the processors continue to increase. The large time consumption of random topology in the case of large processors mainly ascribes to the production and broadcasting of random control matrix in each period. The point-to-point communication mechanism is also partly responsible for the large time consumption when the processors are continue to increase.

To see the performance of the new configured adaptive full-mesh mechanism, we could find that before the processors come to 9, the time consumption is near to the general full-mesh topology. When the number of processors is larger than 9, the time consumption of it is significantly reduced. In the case with 21 processors, its time consumption becomes the minimum. Therefore, compared with the general full-mesh topology, the adaptive mechanism, just as analyzed above, can effectively reduce the communication load.

(2) **Solution quality**

From the perspective of solution quality, the independent one without any collaboration gets the worst solution result. For the reason that in each sub-population, less individuals is much powerless without the communication with others. Among ring topology and mesh topology, their solution quality from good to bad can be listed as: single-ring < double-ring < single-mesh < double-mesh. During the whole test, single-ring topology shows obviously low searching capability in the specific problem. With the increase of individual exchange in the whole population, we can see that the searching capability of mesh-topology mechanisms is better than that of ring-topology mechanisms. Likely, double-side exchange scheme performs always better than single-side exchange for solving OACR problem. Moreover, from the solution stability point of view, single-ring topology with lower communication and high diversity is the most unstable one of the four schemes. In contrast, double-mesh with the most collaboration is quite stable than the others.

Compared with the four schemes, full-mesh topology performs slightly better. But its stability is worse than double-grid topology. High communication especially when more processors are adopted makes the whole population have low diversity and is partly responsible for its low stability. As we analyzed before, too frequent communication is apt to disrupt the searching direction of each sub-population.

Different with the full-mesh one, random topology performs quite well in solving OACR. The overall solution quality keeps a high level near to the full-mesh one. It has high stability along with the increase of processor number. The random collaboration in each period not only makes high diversity in each sub-population, but also avoids disrupting the whole searching pace and preserves good solution quality. Combined with its time performance, it can be seen that random-topology tradesits searching time for solution quality and stability to some extent.

Further, inspired by the general adaptive mechanisms in improved intelligent optimization algorithm, the new adaptive full-mesh topology has better solution capability than the random one. With population supervision, the amount of data connected and broadcasted in each period is much less than which in random-topology. And the population state can better reflect the evolutionary process and guide the individual exchange. Moreover, with MPI_Allgather, which performs more efficient than the other MPI point-to-point schemes, the searching capability of the whole algorithm is improved without the increase of time consumption.

On the whole, the newly presented PNIA with adaptive full-mesh topology shows high performance in solving OACR problem. Its time consumption with 21 processors is 2.1923 s, which is 10 times lower than the searching with single machine. The speed up ratio of it keeps linear when the processor number is below to 4. The solution accuracy is also largely improved to a high level compared with the other traditional topology. And with the full use of the collective communication of MPI, the new configured topology is also easy to implement.

However, we should notice that in different problems and different computing environments, the solution capability of each topology is changing and quite

unstable. The tests above are only focus on solving the OACR problem in CMfg. With changing environment, it does not always work well to other problems.In this case, we can configure other topology module and adopt multiple algorithms if required. That is the advantage of configuration ways. It is also suggested that in such small cluster environment, the improved configuration based on a single scheme performs better than the topology configuration mentioned in Chap. 5. For adjusting changing environments, multiple configured serial intelligent optimization algorithm can also adopted to improve the whole searching efficiency.

10.8 Summary

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. This chapter presented a new model with considering the characteristics of CMfg thoroughly and then designed a high efficient intelligent algorithm for OACR in CMfg. In detail, the primary works and contribution of this chapter can be concluded as follows.

(1) In the new OACR model, user's interaction (control and supervision) in manufacturing tasks were fully considered. 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 introducedin the OACR model. With the full consideration of dynamic computation and communication, the process of OACR in CMfg can be more flexible and practical.
(2) For solving the new complex model, an intelligent optimization algorithm (NIA) is configured with the introduction of niche strategy, immune heuristics, genetic operators and pheromone strategy. Experiments demonstrated the design of heuristic information in NIA for OACR and the suitable heuristics in different situations.
(3) For improve the searching efficiency further, a new adaptive population supervision mechanism is designed and configured with full-mesh topology based on coarse-grained parallelization and MPI collective communication. By using the population degradation state, the individual exchange in each period is controlled to preserve the solution quality with less time consumption. Compared with traditional serial intelligent algorithms and classical parallel intelligent algorithms respectively, the searching capability and time efficiency of the new PNIA was fairly remarkable in the experiments for OACR in CMfg.

References

1. 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
2. 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
3. Yusuf YY, Sarhadi M, Gunasekaran A (1999) Agile manufacturing: The drivers, concepts and attributed. Int J Prod Econ 62(1–2):33–43
4. Flammia G (2001) Application service providers: challenges and opportunities. IEEE Intell Syst Appl 16(1):22–23
5. 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
6. 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 (2011, March 10, Accepted)
7. Zhang L, Luo LY, Tao F, Ren L, Guo H (2010) Key technologies for the construction of manufacturing cloud. Comput Integr Manuf Syst 16(11):2510–2520
8. He K, Zhao Y (2005) Research of grid resource management and scheduling. J WuHan Univ Technol (Information and Management Engineering) 27(4): 1–5
9. Ullman JD (1975) NP-complete scheduling problems. J Comput Syst Sci 10(3):384–393
10. 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
11. 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
12. Tao F, Cheng Y, Zhang L, Luo YL, Ren L (2011) Cloud manufacturing. The 2nd international conference on manufacturing service and engineering (ICMSE)
13. 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
14. 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
15. Zou ZM, Li CX (2006) Integrated and events-oriented job shop scheduling. Int J Adv Manuf Technol 29(5–6):551–556
16. 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
17. Kwok YK (1999) Benchmarking and comparison of the task graph scheduling algorithms. J Parallel Distrib Comput 59(3):381–422
18. Polychronopoulos CD (1991) The hierarchical task graph and its use in auto-scheduling. In: Proceedings of the 5th international conference on supercomputing (ICS' 91)
19. Bokhari SH (1979) Dual processor scheduling with dynamic reassignment. IEEE Trans Software Eng 5(4):341–349
20. Stone HS (1977) Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans Software Eng 3(1):85–93
21. Madhukar M, Leuze V, Dowdy V (1995) Petri net model of a dynamically partitioned multiprocessors system. In: Proceedings of the 6th international workshop on petri nets and performance models (PNPM' 95)
22. Buyya R, Abramson D, Venugopal S (2005) The grid economy. Proc IEEE 93(3):698–714
23. 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 WWW 1(3):281–308
24. Yang T, Gerasoulis A (1993) DSC: Scheduling parallel tasks on an unbounded number of processors. IEEE Trans Parallel Distrib Syst 5(9):951–967

25. Gerasoulis A, Yang T (1993) On the granularity and clustering of directed acyclic task graphs. IEEE Trans Parallel Distrib Syst 4(6):686–701
26. Gerasoulis A, Yang T (1994) Performance bounds for parallelizing Gaussian-Elimination and Gauss-Jordan on message-passing machines. Applied Numerical Mathematics Journal 16:283–297
27. 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
28. Hamscher V, Schwiegelshohn U, Streit A, Yahyapour V (2004) Evaluation of job-scheduling strategies for grid computing. Grid Computing at the 7th International Conference on High Performance Computing 191–202
29. Ememann C, Hamscher V, Yahyapou V (2002) On effects of machine configurations on parallel job scheduling in computational grids. In: Proceedings of the international conference on architecture of computing systems (ARCS 2002), 169–179
30. 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
31. Sinnen O, Sousa LA (2005) Communication contention in task scheduling. IEEE Trans Parallel Distrib Syst 16(6):503–515
32. Sinnen O, Sousa LA, Sandnes FE (2006) Toward a realistic task scheduling model. IEEE Trans Parallel Distrib Syst 17(3):263–275
33. Benoit A, Marchal L, Pineau JF (2010) Scheduling concurrent bag-of-tasks applications on heterogeneous platforms. IEEE Trans Comput 59(2):202–217
34. Adam TL, Chandy KM, Dickson JR (1974) A comparison of list schedules for parallel processing systems. Commun ACM 17(12):685–690
35. Sinnen O, Sousa LA (2004) List scheduling: Extension for contention awareness and evaluation of node priorites for heterogeneous cluster architectures. Parallel Comput 30(1):81–101
36. Wu MY, Gajski DD (1990) Hypertool: a programming aid for message-passing systems. IEEE Trans Parallel Distrib Syst 1(3):330–343
37. Sarkar V (1989) Partitioning and scheduling of parallel programs for multiprocessors. Research Monographs in Parallel Computing, MIT Press
38. Chen S, Eshaghia MM, Wu Y (1995) Mapping arbitrary non-uniform task graphs onto arbitrary non-uniform system graphs. In: Proceedings of the international conference on parallel processing
39. Yang L, Gohad T, Ghosh P, Sinha D, Sen D, Richa A (2005) Resource mapping and scheduling for heterogeneous network processor systems. In: Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems *(ANCS' 05)*, 19–28
40. Weng N, Wolf T (2005) Profiling and mapping of parallel workloads on network processors. In: proceedings of the 20th annual ACM symposium on applied computing (sac) 890–896
41. Huang JG, Chen JE, Chen SQ (2004) Parallel-job scheduling on cluster computing system. Chin J Comput 27(6):765–771
42. Huang JG (2008) Approximation algorithm on multi-processor job scheduling. Comput Eng Appl 44(32):26–28
43. 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
44. Correa RC, Ferreira A, Rebreyend P (1999) Scheduling multiprocessor tasks with genetic algorithms. IEEE Trans Parallel Distrib Syst 10(8):825–837
45. 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

46. 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
47. 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 Circuits Syst 26(6):1010–1029
48. Chen WN, Zhang J (2009) An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans Syst Man Cyber 39(1):29–43
49. 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
50. 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
51. 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
52. 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
53. Laha D, Chakraborty UK (2008) An efficient heuristic approach to total flowtime minimization in permutation flow shop scheduling. Int J Adv Manuf Technol 38(9–10):1018–1025
54. 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
55. Zhang JX, Gu ZM, Zheng C (2010) Survey of research progress on cloud computing. Appl Research Comput 27(2): 429–433
56. Hong B, Prasanna VK (2004) Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In: Proceedings of the 18th international parallel and distributed processing symposium (IPDPS' 04)
57. Bhat PB, Raghavendra CS, Prasanna VK (2003) Efficient collective communication in distributed heterogeneous systems. J Parallel Distrib Comput 63(3):251–263
58. Gawiejnowics S (2008) Time-dependent scheduling. Springer, Berlin
59. Wang L, Pan J, Jiao LC (2000) The immune programming. Chin J Comput 23(8): 806–812
60. Wang L, Pan J, Jiao LC (2000). The immune algorithm. Acta Electronica Sinica, 28(7): 74–77
61. 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
62. 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
63. 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
64. 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
65. 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
66. Jerald J, Asokan P, Saravanan R, Delphin R, 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
67. 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