Expert Systems with Applications 37 (2010) 1769–1775

Contents lists available at ScienceDirect

Expert Systems with Applications

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

# Multi-objective optimization based on ant colony optimization in grid over optical burst switching networks

Yahong Yang [a,b,*], Guiling Wu [a], Jianping Chen [a], Wei Dai [a]

[a] The State Key Laboratory on Advanced Optical Communication Systems and Networks, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
[b] Department of Automation, Nanchang University, Nanchang 330031, Jiangxi, China

ARTICLE INFO

*Keywords:*
Grid
OBS
Multi-objective optimization
Ant colony optimization

ABSTRACT

A multi-objective task scheduling approach for grid over optical burst switching (GOBS) networks is proposed. It takes into account the selection of both computational resource and network resource and is able to simultaneously satisfy three objectives representing the requirements of grid users and resource providers, namely completion time, payment and load balancing. An ant colony optimization algorithm (ACO) for this multi-objective GOBS optimization problem is designed. The convergence and diversity preserving of the algorithm is compared with the nondominated sorting genetic algorithm (NSGA-II) through three performance metrics. Simulations are carried out to compare grid and network performance of these two algorithms. Grid scheduling performance comparison between single-objective optimization and multi-objective optimization based on ACO is also taken by simulations.



## 1. Introduction

Grid computing platform enables the sharing of geographically distributed heterogeneous resources for solving large-scale problems in science, engineering and commerce. The grid based on optical burst switching (OBS) networks well suits to the grid applications due to their unique features such as low delay, large bandwidth with high utilization efficiency and simple control, and attracts more and more attention (Simeonidou et al., 2006). For grid applications, e.g. high resolution home video editing, high definition interactive television (HDTV), e-health, immersive interactive learning environments and online gaming (Breusegem, Leenheer, Cheyns, Turck, & Simeonidou, 2004; Simeonidou et al., 2006), objectives from users are to minimize completion time of task, minimize payment for the task and maximize security, while objectives of the grid resource providers aim to maximize resource utilization, minimize the wear down of resources and so on. Such problem is a multi-objective optimization problem, which requires the simultaneous optimization of two or multiple and often conflicting objectives. Obviously, in the multi-objective optimization problem, it is not realistic to search a single solution but to search a set of trade-offs, that is, nondominated solutions (Chankong & Haimes, 1983). Furthermore, most multi-objective optimization problems involve a large and complex search space, which is hard to be solved by exact methods such as linear programming and gradient search (Romero, 2009; Zitzler, Deb, & Thiele, 2000).

Evolutionary algorithms (EAs) can not only search intractably large spaces for multiple Pareto-optimal solutions, but also find many solutions of Pareto-optimal set in once algorithm run (Coello, 1999). In addition, EAs do not local on the shape and continuity of Pareto front and are easy to process discrete and concave Pareto front. Therefore, EAs have become established as the method for exploring the Pareto-optimal front of multi-objective optimization problems (Romero, 2009; Zitzler et al., 2000). Over the past decades, many EAs on multi-objective optimization have been proposed (Chang, Chen, & Liu, 2007; Lin & Gen, 2008; Lo, Chen, Huang, & Wu, 2008; Veldhuizen & Lamont, 1998). Most of them are based on genetic algorithm. Little attention has been paid to ant colony algorithm. Ant colony optimization (ACO) is a population-based, cooperative search algorithm that is derived from the behavior of real ants looking for food. The solution generation by ants is guided by pheromone trails and problem-specific heuristic information. Up to date, ACO has been widely applied to the solutions of many optimization problems such as the traveling salesman problem, job-shop scheduling and the quadratic assignment problem. In particular, by maintaining a population of solutions, ACO can search many nondominated solutions in parallel. This characteristic makes ACO very attractive for solving multi-objective optimization problem, which has been applied to resolve the discrete multi-objective optimization problem (Allahverdi & Anzi, 2008; Juang & Wang, 2008; Kindt, Monmarché, & Tercinet, 2002).

This paper proposes a multi-objective GOBS task scheduling approach that considers the selection of computational resource and

* Corresponding author. Address: The State Key Laboratory on Advanced Optical Communication Systems and Networks, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China. Fax: +86 21 34205140.
*E-mail address:* yahongyang@sjtu.edu.cn (Y. Yang).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.07.032

network resource in an overall sense and can simultaneously fulfil both users' requirement and resource providers' requirement such as completion time, payment, load balancing. An ACO algorithm is adopted to realize this multi-objective GOBS task scheduling because its positive feedback, parallelity and scalability are suitable for frequently changed GOBS. The idea of Pareto dominance is incorporated into ACO to search nondominated solutions. At the same time, a searching operator that combines the pheromone exchanging and shared niching method would motivate convergence toward globally nondominated solutions and preserve the diversity of solutions. We called this algorithm "multi-objective ant colony optimization" (MOACO). The convergence and diversity preserving performance of MOACO is compared with the nondominated sorting genetic algorithm (NSGA-II) (Deb, Pratap, & Agarwal, 2002) according to a suite of three performance metrics. Simulations are carried out on a centralized GOBS simulation platform to compare their grid and network performance. In addition, simulations are also implemented to make grid performance comparison between single-objective optimization and multi-objective optimization based on ACO for GOBS task scheduling. The remaining paper is organized as follows: Section 2 describes the multi-objective GOBS problem modeling and gives the multi-objective functions; Section 3 illustrates the MOACO implementation; Section 4 gives simulation results and performance analysis; and Section 5 concludes the paper.

## 2. Multi-objective GOBS optimization problem modeling

Suppose there are $N$ jobs (denoted as $J_1, J_2, \ldots, J_N$) to be processed in OBS based grid environment, where there are $M$ different computational resources (denoted as $P_1, P_2, \ldots, P_M$). The grid clients wish to complete jobs as soon and cheap as possible, while the grid resource providers wish to maximize the utilization of resources. The main goal of the scheduling problem is to reasonably assign grid resources for these $N$ jobs in such way that the goals of the users and resource providers are met at one time. Obviously, this scheduling problem belongs to multi-objective optimization problem. In order to simplify problem without losing generality, we made the following assumptions: (1) Jobs are internally independent. (2) For each job, when it is assigned to one computational resource, the execution time can be known. (3) Once a job is scheduled on a machine, it will run to the finish without preemption.

We define the following multi-objective GOBS optimization functions associated with three objectives from the users and the resource providers, which minimize the time and the money to complete these $N$ jobs and the grid resources load difference at the same time. Each objective combines computational resource and network resource:

$$\begin{aligned}
&\text{Minimize} \quad f_1 = \max_{i,j}\{\phi_{ij} * (T_{ij}^{CR} + T_{ij}^{NR})\} \\
&\text{Minimize} \quad f_2 = \sum_{i=1}^{N}\sum_{j=1}^{M} c_j \phi_{ij} T_{ij}^{CR} + \partial \sum_{i=1}^{N}\sum_{j=1}^{M} \phi_{ij} T_{ij}^{NR} \\
&\text{Minimize} \quad f_3 = \sum_{j=1}^{M} |\kappa_j^{CR} - \kappa_{avg}| + \sum_{l=1}^{G} |p_l - p_{avg}| \\
&\text{s.t.} \qquad \phi_{ij} = 0 \text{ or } 1 \quad (i = 1, \ldots, N, j = 1, \ldots, M) \\
&\qquad \sum_{j=1}^{M} \phi_{ij} = 1 \quad (i = 1, \ldots, N) \\
&\qquad P_j > 0, \quad p_l \geq 0
\end{aligned} \tag{1}$$

The first objective function $f_1$ is the completion time, defined as the duration from sending out the first job to completing the last job. Given that job $i$ is assigned to computational resource $j$, then $T_{ij}^{CR}$ stands for the time to execute the $i$th job at resource $j$, $T_{ij}^{NR}$ stands for the time to send the $i$th job from its submitted client $u$ to resource $j$. They are given by:

$$T_{ij}^{CR} = b_i / P_j \tag{2}$$

$$T_{ij}^{NR} = \frac{d_i}{R_{uj} * \chi} \tag{3}$$

where $b_i$ and $d_i$ are computational quantity and data size of the $i$th job, respectively. $P_j$ is the computational capability of the $j$th computational resource. $R_{uj}$ denotes the TCP sending rate of the path from client node $u$ who submits job $i$ to resource node $j$, which denotes the segments sent per unit time. In OBS networks, the sending rate of fast TCP flows is the function of the burst loss ratio of the TCP path, $B_{u\to j}$, the average round trip time in absence of the burstifier, $RTT_0$, the ratio between the assembly time and $RTT_0$, $\alpha$, and the maximum congestion window, $W_{\max}$ (Detti & Listanti, 2002). $\chi$ is the TCP segment size.

$$R_{uj} = \begin{cases} \dfrac{B_{u\to j}^3 - B_{u\to j} + 1}{(1+\alpha) RTT_0 [B_{u\to j}(1 - B_{u\to j})^2 + B_{u\to j}^3 f(B_{u\to j})]} & \text{for } B_{u\to j} > \dfrac{1}{W_{\max}} \\ \dfrac{W_{\max} - B_{u\to j} W_{\max} + B_{u\to j}^2}{(1+\alpha) RTT_0 [(1 - B_{u\to j})^2 + B_{u\to j}^2 f(B_{u\to j})]} & \text{otherwise} \end{cases} \tag{4}$$

where

$$f(B_{u\to j}) = 1 + B_{u\to j} + 2B_{u\to j}^2 + 4B_{u\to j}^3 + 8B_{u\to j}^4 + 16B_{u\to j}^5 + 32B_{u\to j}^6 \tag{5}$$

$$B_{u\to j} = 1 - \prod_{l=1}^{L} (1 - p_l) \tag{6}$$

In the above equation, $L$ is the link number of the TCP path from client node $u$ to resource $j$. $p_l$ is the burst loss ratio of link $l$ in the TCP path.

The second objective function $f_2$ in Eq. (1) is the charge that grid users pay for using the resources. $\sum_{i=1}^{N}\sum_{j=1}^{M} c_j \phi_{ij} T_{ij}^{CR}$ is the total payment expended on using computational resource for all jobs. $\gamma \sum_{i=1}^{N}\sum_{j=1}^{M} \phi_{ij} T_{ij}^{NR}$ is the total payment expended on using network resource for all jobs, where $c_j$ is the cost per unit time of the $j$th computational resource, $\gamma$ is the cost per unit time of OBS networks.

The third objective function $f_3$ in Eq. (1) represents the resource load difference (*RLD*) which is used to characterize the load balancing. The balanced load across heterogeneous computing and network architecture is critical for grid resource availability and user efficiency. The *RLD* includes two parts: computational *RLD* and network *RLD*. The computational resource load is represented by "the time spent by the computational resource to execute all jobs assigned to it". $\kappa_j^{CR}$ is the computational time that resource $j$ consumes to execute all jobs assigned to it. $\kappa_{avg}$ is the average occupation time on all computational resources. They are expressed as:

$$\begin{aligned}
\kappa_j^{CR} &= \sum_{i=1}^{N}\sum_{j=1}^{M} \phi_{ij} b_i / P_j \\
\kappa_{avg} &= \sum_{i=1}^{N}\sum_{j=1}^{M} \phi_{ij} T_{ij}^{CR} / M
\end{aligned} \tag{7}$$

For network resource, the difference between the burst loss ratio of each OBS link and the average burst loss ratio of all links in the network is summed to represent the load balancing degree of the network since the burst loss ratio reflects the status of traffic load in some sense. $p_{avg}$ is the average burst loss ratio of all links and $G$ is the total number of links in the OBS networks.

In constrained condition, $\varphi_{ij}$ equals 1 if resource $j$ is assigned to job $i$, otherwise $\varphi_{ij}$ is 0. $\sum_{j=1}^{M}\phi_{ij}=1$ guarantees each job only assigned to one resource.

## 3. MOACO implementation

A MOACO is designed to implement this multi-objective GOBS task scheduling. Each ant in current population independently searches grid resources for all jobs, and then the values of the three objectives for the solution found by each ant are calculated. Compare the solutions in current population with the external set of nondominated solutions using the idea of Pareto dominance to obtain new external set of nondominated solutions (Coello, Pulido, & Lechuga, 2004). This process is repeated until the maximum number of cycles is reached or the process goes into stagnation state. Fig. 1 illustrates the flow chart of MOACO. The main functionalities are as follows:

### 3.1. Initialization

Set the time counter and the cycle counter. Define initial pheromone of $M$ computational resource, $\tau_\mu(0), \mu\in\{1,\ldots,M\}$, according to their computational capability. The higher the computational capability is, the higher the initial pheromone is. Each ant randomly selects one resource from $M$ computational resources and assigns it to the first job. The pheromone of each selected resource is updated by the local pheromone updating rule that will be given by Eq. (10) below.

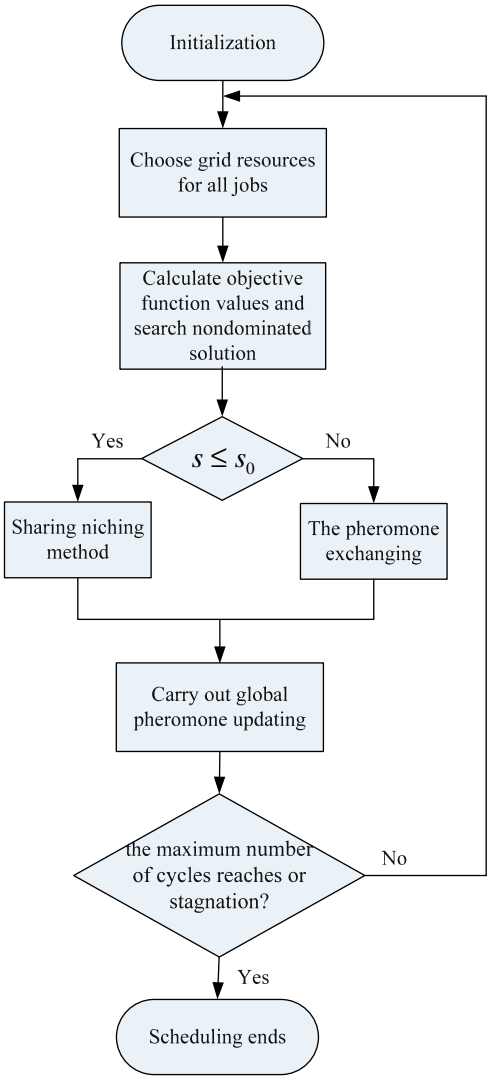


**Fig. 1.** The flow chart of MOACO.

### 3.2. Choose grid resources

Each ant selects grid resources according to the pseudo-random-proportional rule for the subsequent job, and updates the amount of pheromone on the selected resource by the local pheromone updating rule. The resource selection and update of each ant is independent with one another.

The pseudo-random-proportional rule is as follows (Dorigo & Gambardella, 1997):

$$j=\begin{cases}\arg\max\limits_{\tau_\mu(t)\ge\tau_\mu^{th}}\{[\tau_\mu(t)]^{\varepsilon}[\eta_\mu]^{\beta}\} & \text{if } q\le q_0 \text{ (exploitation)}\\ J & \text{otherwise (biased exploration)}\end{cases} \tag{8}$$

where $j$ is the number of computational resource selected for the current job. $q$ is a random number uniformly distributed in $[0\cdots1]$. $q_0$ $(0\le q_0\le 1)$ is a parameter determined by the relative importance of exploitation of accumulated knowledge about the problem versus exploration of new resource. $\tau_\mu(t)$ is the pheromone intensity of the $\mu$th computational resource at time $t$. $\eta_\mu$ is the heuristic information to select the $\mu$th resource, here we define it as follows:

$$\eta_\mu=\lambda_\mu^{CR}\times\tau_\mu(0)+\lambda_\mu^{NR}\times(1/\xi_\mu),\quad \lambda_\mu^{CR}+\lambda_\mu^{NR}=1 \tag{9}$$

The selection of grid resources includes the selection of computational resource and network resource. Eq. (9) combines the inherent configuration of computational resource and the current state of end-to-end network path. $\lambda_\mu^{CR}(\lambda_\mu^{CR}\ge 0)$ is the parameter reflecting the relative importance of $\tau_\mu(0)$, i.e., the innate pheromone of the $\mu$th computational resource. $\lambda_\mu^{NR}(\lambda_\mu^{NR}\ge 0)$ is the parameter reflecting the relative importance of $1/\xi_\mu$, where $\xi_\mu$ achieved by Eq. (3), is the time to send the job to computational resource $\mu$ through the end-to-end network path. The end-to-end network path is determined by the shortest path rule. The larger $\xi_\mu$ is, the lower the possibility to select the path is. $\varepsilon(\varepsilon\geqslant 0)$ in Eq. (8) is the parameter on the relative importance of pheromone. $\beta(\beta\geqslant 0)$ is the parameter on the relative importance of grid resources attributes. $\tau_\mu^{th}$ is a threshold of each resource pheromone. The constrained condition $\tau_\mu(t)\ge\tau_\mu^{th}$ means the resource whose pheromone is less than corresponding threshold should not be chosen. The setting of the threshold is to avoid the overuse of computational resources, which will reduce the processing efficiency of servers. The threshold should be as small as possible to utilize resources sufficiently. $J$ is selected according to the following probability distribution:

$$\varphi_\vartheta(t)=\begin{cases}\dfrac{[\tau_\vartheta(t)]^{\varepsilon}*[\eta_\vartheta]^{\beta}}{\sum_{\tau_\mu(t)\ge\tau_\mu^{th}}[\tau_\mu(t)]^{\varepsilon}*[\eta_\mu]^{\beta}} & \text{if } \tau_\vartheta(t)\geqslant\tau_\vartheta^{th}\\ 0 & \text{otherwise}\end{cases} \tag{10}$$

After resource $j$ is selected, its pheromone is updated according to the following local pheromone updating rule (Dorigo & Gambardella, 1997):

$$\tau_j(t+1)=(1-\rho)\times\tau_j(t)+\rho\times(-b_i/P_j) \tag{11}$$

where $\rho$ is the evaporation rate of local pheromone $(0\leqslant\rho<1)$, $i$ is the number of the scheduled job. $b_i/P_j$ is the time to execute job $i$ at computational resource $j$, which represents the pheromone decrease of resource $j$.

The above procedure is repeated for the following jobs one by one until the last job is scheduled to one resource.

### 3.3. Calculate objective function values and search nondominated solution

After finding grid resources for all jobs, calculate the values of the three objectives for each solution in current ant population according to the model given in Section 2. Compare the solutions in current population using the idea of Pareto dominance with the external set of the nondominated solutions. If a solution in the current ant population is not dominated by any other solutions in the current population and the external set of nondominated solutions, this solution is added to the external set. Then all solutions dominated by the added one are eliminated from the external set.

### 3.4. Global pheromone updating

The global pheromone updating process includes two steps: search optimal solution and update corresponding pheromone. A searching operator proposed by Zhang and Huang (2005) is adopted to search the optimal solution for global pheromone updating. The adopted searching operator combines the pheromone exchanging and a shared niching method (Goldberg & Richardson, 1987) to motivate convergence toward globally nondominated solutions and preserve the diversity of solutions. If $s \leqslant s_0$, the algorithm uses the sharing niching method; otherwise it uses pheromone exchanging. $s$ is a random number uniformly distributed in $[0 \cdots 1]$. $s_0$ $(0 \leqslant s_0 \leqslant 1)$ is a constant value.

After finding the optimal solution, the global pheromone updating is carried out on each computational resource corresponding to the solution with highest pheromone or the minimum niche counts according to following rule (Dorigo & Gambardella, 1997):

$$\tau_\mu^g = (1-\theta) \times \tau_\mu(0) + \theta \times \Delta\tau_\mu \tag{12}$$

where $\theta$ $(0 \leqslant \theta < 1)$ is the evaporation rate of global pheromone, and

$$\Delta\tau_\mu = \begin{cases} \sum_{i=1}^{N} \phi_{i\mu} b_i / P_\mu & \text{if } \mu \in \text{solution with highest perhomone or minimum niche counts} \\ 0 & \text{otherwise} \end{cases}$$

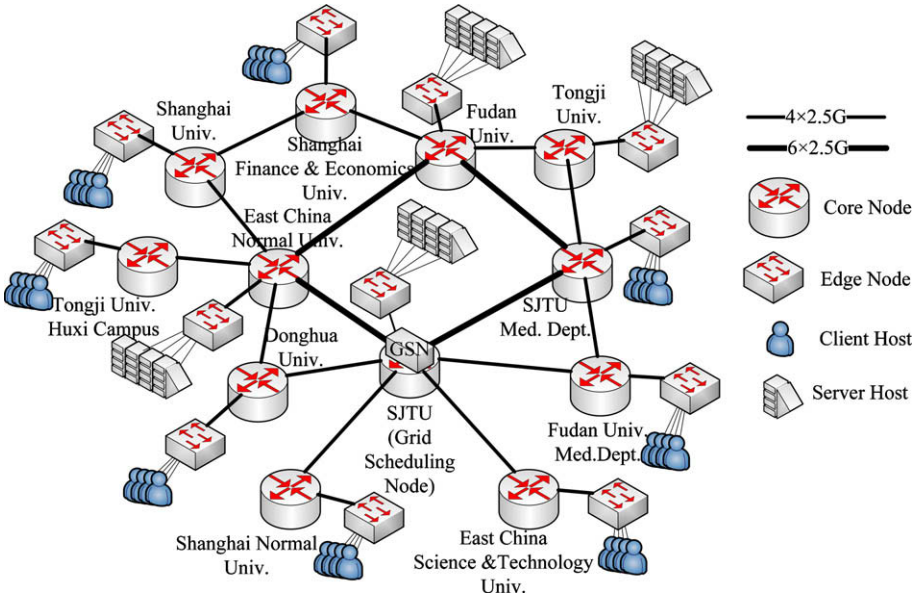


**Fig. 2.** Simulation topology.

**Table 1**
The computational capability description of each grid computational resource host.

| Resource host | Host1 | Host2 | Host3 | Host4 | Host5 | Host6 | Host7 | Host8 |
|---|---|---|---|---|---|---|---|---|
| Computational capability (MIPS) | $1.0 \times 10^3$ | $4.0 \times 10^2$ | $9.2 \times 10^2$ | $2.0 \times 10^2$ | $7.0 \times 10^2$ | $1.6 \times 10^3$ | $3.6 \times 10^2$ | $6.0 \times 10^2$ |
| | Host9 | Host10 | Host11 | Host12 | Host13 | Host14 | Host15 | Host16 |
| Computational capability (MIPS) | $8.7 \times 10^2$ | $1.2 \times 10^3$ | $2.5 \times 10^2$ | $5.0 \times 10^2$ | $3.8 \times 10^3$ | $6.9 \times 10^3$ | $3.0 \times 10^2$ | $1.7 \times 10^2$ |

**Table 2**
The performance metrics ONVG, SP and MDR comparison as job load varies under MOACO and NSGA-II.

| Algorithm | Parameters | Job load | | | | |
|---|---|---|---|---|---|---|
| | | 60 | 120 | 180 | 240 | 300 |
| MOACO | ONVG | 27 | 28 | 23 | 26 | 16 |
| | SP | 0.152566 | 0.246205 | 0.493494 | 0.421388 | 0.60809 |
| | MDR | 1.668637 | 2.693473 | 3.64752 | 3.328157 | 4.497933 |
| NSGA-II | ONGV | 12 | 10 | 11 | 10 | 8 |
| | SP | 0.30014 | 0.399936 | 0.535768 | 1.387545 | 1.142567 |
| | MDR | 1.22983 | 1.798143 | 2.341782 | 2.571842 | 3.144537 |

$\sum_{i=1}^{N} \phi_{i\mu} b_i / P_\mu$ is the time that resource $\mu$ consumed to complete all jobs assigned to it.

The process B–D is iterated until the pre-defined number of cycles (*MaxCount*) is reached, or all ants choose the same resource.

After the scheduling is finished, if there are several nondominated solutions, a solution belonging to the set of nondominated solutions is randomly chosen.

## 4. Simulation results

Simulations are conducted on a centralized grid OBS simulation platform (Dai, Wu, Yang, & Chen, 2008). The OBS based grid topology is from Shanghai Education & Research Network (SHERNET), shown in Fig. 2. It includes 12 edge nodes, 12 core nodes and 48 hosts. The 16 hosts linking to the four edge nodes, located at Shanghai Jiao Tong University (SJTU), Fudan University, Tongji University and East China Normal University, serve as computational resource hosts. The rest 32 hosts serve as grid client hosts. The grid scheduler node (GSN), located at SJTU, collects job requests to a certain value and sends the batch of job requests to the scheduler module in GSN, which carries out task scheduling according to MOACO scheduling strategy. After the scheduling of the batch of jobs is completed, the GSN sends back response packets to the corresponding clients through burst control channels. Once a client receives the response packet, it begins to send job data through allocated path to edge route, where the job data is assembled and sent to the destination edge router along the allocated network path by just enough time (JET) protocol (Yoo & Qiao, 1997). The destination edge router disassembles the burst and sends the job data to the allocated computational resource. The computational resources begin to deal with relevant jobs after receiving the data, and send the processing results back to the clients through traditional OBS protocol.

In our simulation, the interactive application adopted is same as that in Dai et al. (2008). The first job request is randomly generated. The mean interval time between generating a new job and receiving the reply is set to be 30 ms. The average data size is 2 M bytes, while the average computational quantity is 80 MI (Million Instructions).

The computational capability of each computational resource host is listed in Table 1. The initial pheromone of each computational resource, $\tau_\mu(0)$, is set as 10% of computational capability of each resource, respectively. The pheromone threshold of every resource, $\tau_\mu^{th}$, is set as $0.01 \times \tau_\mu(0)$. The unit price of computational resource increases with the computational capability, and the larger the computational capability is, the faster the unit price increases. The network resource price per unit time is assumed to be two units of money. The parameters of MOACO are set as: $\varepsilon = 1$, $\varphi = 1$, $\rho = 0.3$, $\theta = 0.3$, $s_0 = 0.7$, $m = 20$, $MaxCount = 10$, and $\lambda_\mu^{CR} = 0.7, \lambda_\mu^{NR} = 0.3$. The parameters of NSGA-II are set as: generation is 10, population size is 20, probability of crossover is 0.5, and probability of mutation is 0.2.

### 4.1. The MOACO performance assessment

Table 2 shows overall nondominated vector generation (ONVG) (Veldhuizen, 1999), Spacing (SP) (Schott, 1995) and maximum distribution range (MDR) (Zitzler, 1999) under MOACO and NSGA-II as the function of the job load, namely, the number of job requests in each batch. From the results, we can see that the

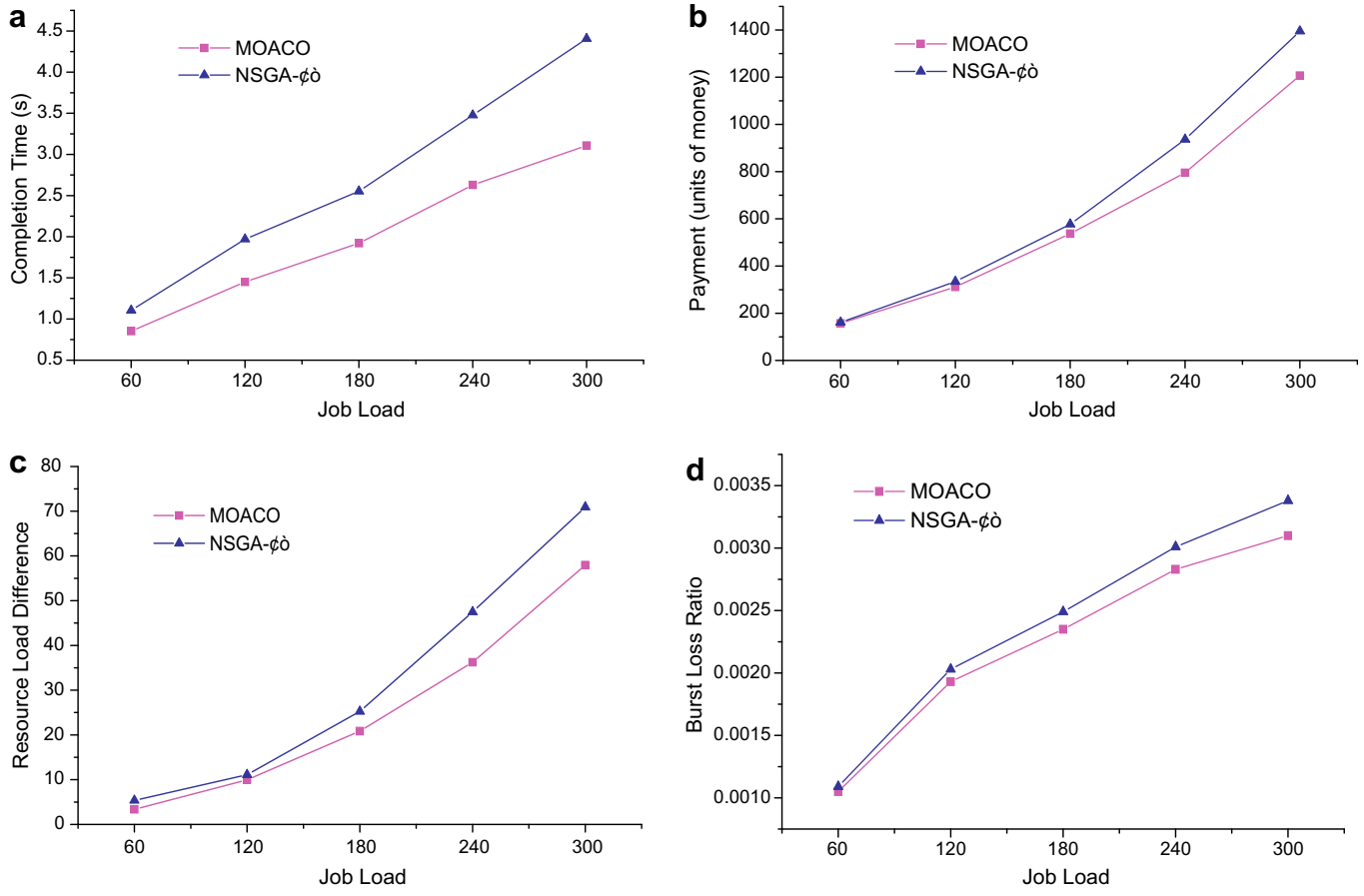


**Fig. 3.** Effects of the job load on the performance of: (a) completion time, (b) payment, (c) resource load difference and (d) burst loss ratio.

convergence and diversity preserving properties of MOACO is better than NSGA-II. This is due to the fact that for MOACO, initial pheromone is directly relevant to the computational capability of each resource, and hence it can quickly search solution. At the same time, through continuous updating of pheromone, it can efficiently converge to the optimal solution. For NSGA-II, however, it randomly allocates computational resource and thus the searching has some blindness. At the same time, it does not sufficiently utilize the feedback information of grid resources and load, which results in the low efficiency of searching the optimal solution.

Fig. 3 shows the completion time, payment, resource load difference and burst loss ratio as a function of the job load under MOACO and NSGA-II. It can be observed from Fig. 3, the grid and network performance in the case of MOACO is superior to that in the case of NSGA-II. The reason is that grid resources allocation under MOACO combines the inherent configuration of computational resource and the current state of end-to-end network path and simultaneously incorporates continuous updating of pheromone, therefore it can find more appropriate grid resources and achieve better grid and network performance. Comparatively, grid resources allocation under NSGA-II does not consider the current status of grid and sufficiently employ the feedback information of grid, which lead to its low grid and network performance.

### *4.2. Grid scheduling performance comparison between single-objective optimization and multi-objective optimization*

Fig. 4 shows the completion time, payment and resource load difference as a function of the job load under MOACO and three single-objective ACO optimization algorithms, i.e., time minimization (TM) algorithm, cost minimization (CM) algorithm and load difference minimization (LDM) algorithm. From Fig. 4a, we can see that TM produces the shortest completion time, while CM produces the longest completion time. This is due to the fact that computational resources with superior capabilities are selected to minimize the completion time in TM and computational resources with inferior capabilities are chosen to minimize fee in CM. The completion time of LDM and MOACO are between those of the above two since both algorithms have a trade-off objective between objectives of TM and CM. At the same time, we can also see that the completion time of MOACO is shorter than that of LDM when the job load is light. As the job load increases, the completion time of MOACO becomes larger than that of LDM. The reason is that when the job load is light, the effect of load balance is less important. MOACO has shorter completion time since it considers the completion time. As the job load increases, the effect of load balance is reflected, then the completion time in LDM becomes shorter since it selects balanced load to avoid congestion efficiently. In Fig. 4b, the payment in CM is the least since it minimizes the expenses spent on completing jobs. TM produces the highest payment because the grid scheduler chooses the more powerful (expensive) computational resources to minimize the completion time. When the job load becomes heavy, the payment of MOACO is less than that of LDM since the payment is considered in MOACO. Although balanced grid resources are selected in LDM, they are perhaps less cheap. The impact of the job load on resource load difference is illustrated in Fig. 4c. Comparatively, the load in the case of LDM is the most balanced because the light load resources are selected in this case. The load balance under MOACO is better than that under TM and CM, whose load balance is close to each other. This is reasonable since MOACO considers load balance factor, while the other two do not consider it.

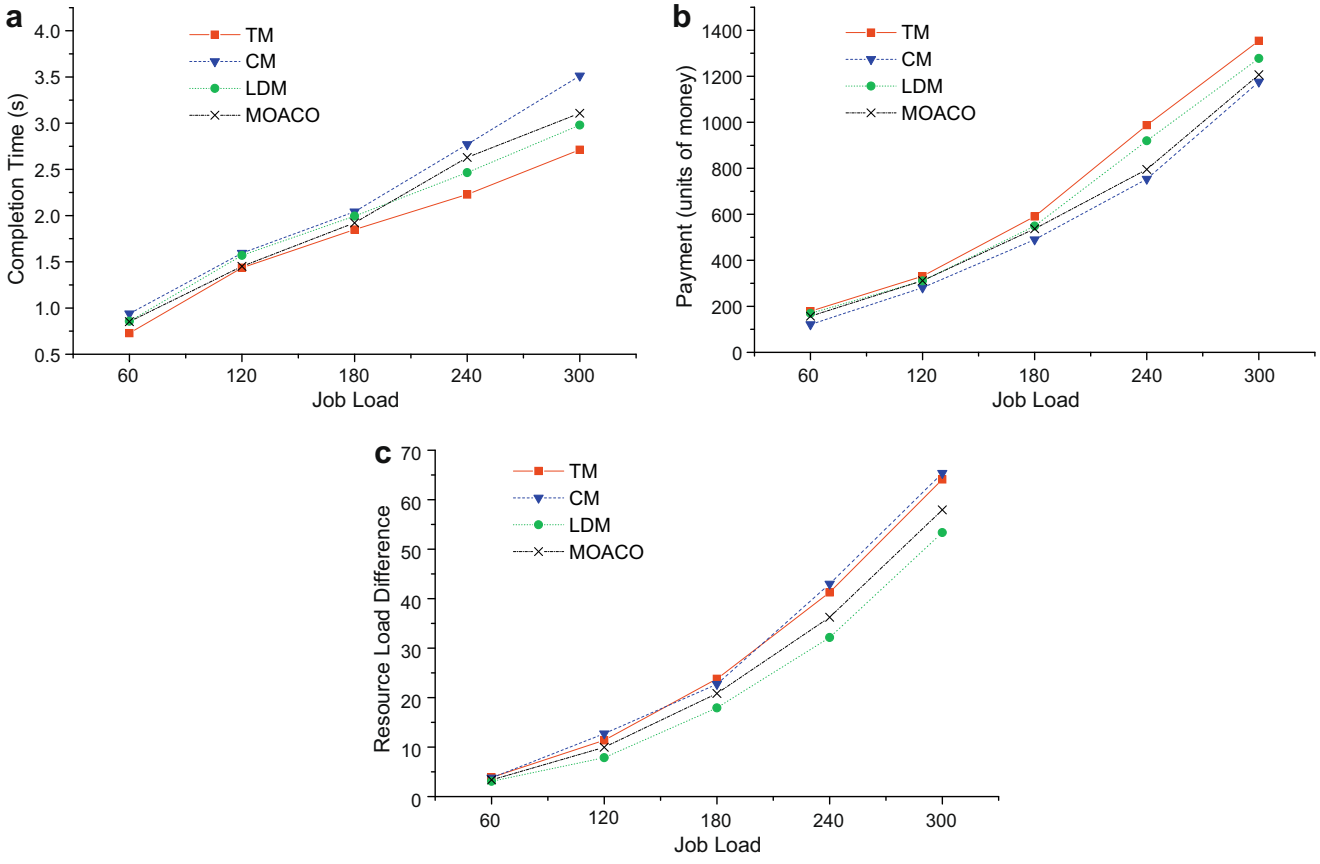


**Fig. 4.** Effects of the job load on the performance of: (a) completion time, (b) payment and (c) resource load difference.

## 5. Conclusion

In this paper, a multi-objective GOBS task scheduling approach is proposed. It combines the selecting process of computational resource and network resource and can simultaneously satisfy the requirements of grid users and resource providers, such as completion time, payment and load balancing. A multi-objective ant colony optimization algorithm is designed to deal with this multi-objective GOBS task scheduling problem. Pareto dominance is incorporated into the algorithm to search nondominated solutions, on the other side, a searching operator that combines the pheromone exchanging and the shared niching method would motivate convergence toward globally nondominated solutions and maintain the diversity of solutions. The convergence and diversity preserving properties of the implemented algorithm are compared against NSGA-II through the three performance metrics. Simulations are conducted to compare their grid and network performance in terms of completion time, payment, *RLD* and burst loss ratio. The results show that MOACO performs better than NSGA-II. Additionally, grid scheduling performance comparison between the single-objective and multi-objective optimization based on ACO is also taken. The results show that the single-objective optimization maximizes the interests of some objective, while multi-objective optimization weighs the requirements of multiple objectives and the global performance is better.

## Acknowledgements

The paper is partially supported by National Science Foundation of China (NSFC) (ID90704002 and 60877012), 863 Project (ID2006AA01Z242 and 2007AA01Z275), Dawn Program for Excellent Scholars by the Shanghai Municipal Education Commission, and the Key Disciplinary Development Program of Shanghai (T0102).

## References

Allahverdi, A., & Anzi, F. S. A. (2008). The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time. *International Journal of Advanced Manufacturing Technology, 37*(1–2), 166–177.

Breusegem, E. V., Leenheer, M. D., Cheyns, J., Turck, F. D., Simeonidou, D. (2004). An OBS architecture for pervasive grid computing. In Workshop on optical burst switching. BROADNETS. California.

Chang, P., Chen, S., & Liu, C. (2007). Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems. *Expert Systems with Applications, 33*(3), 762–771.

Chankong, V., & Haimes, Y. Y. (1983). Multiobjective decision making: Theory and methodology. North-Holland: Amsterdam Press.

Coello, C. A. C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems, 1*(3), 269–308.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transaction on Evolutionary Computation, 8*(3), 256–279.

Dai, W., Wu, G., Yang, Y., & Chen, J. (2008). Resource scheduling in centralized OBS-based grids. *Journal of Optical Networking, 7*(2), 111–118.

Deb, K., Pratap, A., & Agarwal, S. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Detti, A., & Listanti, M. (2002). Impact of segments aggregation on TCP Reno flows in optical burst switching networks. In *Proceedings of IEEE Infocom* (pp. 1803–1812). New York.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithm with sharing for multimodal function optimization. In Grefenstette (Ed.), *Proceedings of the second international conference on genetic algorithms* (pp. 41–49). San Mateo, CA: Morgan Kaufman.

Juang, C., & Wang, C. ( (2008). A self-generating fuzzy system with ant and particle swarm cooperative optimization. *Expert Systems with Applications, 36*(3), 5362–5370.

Kindt, V. T., Monmarché, N., & Tercinet, F. (2002). An ant colony optimization algorithm to solve a 2-machine bicriteria flow-shop scheduling problem. *European Journal of Operational Research, 142*(2), 250–257.

Lin, C., & Gen, M. (2008). Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. *Expert Systems with Applications, 34*(4), 2480–2490.

Lo, S., Chen, R., Huang, Y., & Wu, C. (2008). Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system. *Expert Systems with Applications, 34*(3), 2071–2081.

Romero, C. (2009). Evolutionary algorithms for subgroup discovery in e-learning: A practical application using moodle data. *Expert Systems with Applications, 36*(2), 1632–1644.

Schott, J. R. (1995). Fault Tolerant Design using single and multicriteria genetic algorithm optimization. MS thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Simeonidou, D., Nejabati, R., Ciulli, N., Battestilli, L., Carrozzo, G., Castoldi, P. (2006). Grid optical burst switched networks (GOBS). Grid High Performance Networking Research Group.

Veldhuizen, D. A. V. (1999). Multiobjective evolutionary algorithms: Classifications, analyses, and new Innovations. PhD thesis, Grad. School of Eng. of the Air Force Institute of Technology, Air University.

Veldhuizen, D. A. V., & Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. Technical Report, Air Force Institute of Technology.

Yoo, M., & Qiao, C. (1997). Just-Enough-Time (JET): A high speed protocol for bursty traffic in optical networks. IEEE/LEOS Technologies for a global information infrastructure (pp. 26–27).

Zhang, Y., & Huang, S. (2005). On ant colony algorithm for solving multiobjective optimization problems. *Control and Decision, 20*(2), 171–173.

Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8*(2), 173–195.