Optimization Algorithm for Divisible Load Scheduling on Heterogeneous Star Networks

Xiaoli Wang, Yuping Wang* and Kun Meng
School of Computer Science and Technology, Xidian University, Xi'an, China
Email: wangxiaolibox@gmail.com, ywang@xidian.edu.cn

***Abstract*—Scheduling divisible loads on heterogeneous distributed computing systems is addressed in this paper. The platform considered here is more general and realistic, where processors are connected in star topology with arbitrary communication and computation speeds and non-zero start-up overheads. A new optimization algorithm, called WX-GA, is proposed to tackle the following four issues: (1) How many processors are needed in computation? (2) Finding the optimal distribution sequence among processors. (3) How much the load fraction should be assigned on each processor? (4) When workload is large enough, what is the sufficient and necessary condition for the minimum processing time? Finally, the experimental results indicate the efficiency and effectiveness of the proposed algorithm.**

***Index Terms*—divisible loads, distributed computing, start-up overheads, optimal distribution sequence, weight-based crossover operator**

I. INTRODUCTION

Divisible loads are parallel tasks which can be divided into arbitrary number of fractions [1]. There are no precedence relationships among fractions, thus they can be processed independently on the processors in parallel. Such workload model is useful in many real world applications, e.g., signal processing, image processing, experimental data processing and so on.

Divisible load theory has emerged as a powerful tool for modeling data-intensive computational problems, and a great amount of research on divisible load scheduling has been made in the last decades. Under certain conditions, closed-form expressions for the processing time and load fractions for processors involved in computation have been derived in both homogeneous systems and heterogeneous systems.

In the earlier studies, relatively simple models without start-up overheads have been proposed. For homogeneous networks in blocking mode of communication, a closed-form expression for the processing time of processors in linear topology was derived [2], and asymptotic performance analysis has been made for the cases of bus and tree topologies [3,4]. However, systems in reality are usually heterogeneous systems with arbitrary computation or communication speeds. For heterogeneous star networks, a closed-form expression for optimal processing time was derived by Ghose et al. [5]. It also has been proved that the sequence of load distribution should follow the order in which the communication speeds decrease in order to achieve the minimum processing time. In the case of heterogeneous tree networks, the effect of load distribution sequences on the processing time was analyzed by Kim et al. [6], and an algorithm which optimally determines the order of load distribution was developed. It was shown that the distribution order depends only on the communication speeds between processors but not on the computation speeds.

All the above models considered the blocking mode of communication. Kim [7] first introduced the nonblocking mode of communication in homogeneous systems with processors connected in star topology, and the results on the optimal sequencing and arrangement are presented by Mani et al. [8]. All the above works do not take start-up overheads into consideration. However, zero start-up overheads are quite not realistic for most distributed systems. In the case of constant start-up time on bus networks with blocking mode of communication, Suresh et al. [9], Bharadwaj et al. [10], Blazewicz [11] analyzed the influence of start-up overheads on the optimal processing time and studied the effect of changing the distribution sequence on the processing time. In particular, Bharadwaj et al. [10] gave a necessary and sufficient condition for the existence of the optimal processing time, and it was shown that the processing time is minimized when the load distribution sequence follows the decreasing order of the computation speeds. For heterogeneous star systems with non-blocking mode of communication, closed-form expressions for the processing time and load fractions have been derived by Shang [12].

Given that the start-up overheads and computation speeds of processors are with arbitrary values but that the communication speeds are all different, Beaumont et al. [13] proved that if load fractions were sent to processors according to the decreasing order of communication speeds, then when the processing time becomes large enough, the workload finished during time units is optimal among all possible orderings. For general cases with workload to be completed in arbitrary time, however, the optimal distribution sequence has not been addressed.

Based on the above study, many scholars have done a great deal of extended research [14-24], but the problem of deriving the optimal distribution sequence has not been solved.

doi:10.4304/jsw.9.7.1757-1766

In this paper, we propose a new optimization algorithm for divisible load scheduling on heterogeneous distributed systems in blocking mode of communication to address the following four issues:

• whether all processors are needed in computation. If not, how many and which processors should be selected;

• whether the distribution sequence has an effect on the processing time. If true, which order the load distribution should follow to achieve the minimum processing time;

• given a certain distribution sequence, how much the load fraction should be assigned on each processor;

•when workload is large enough, what is the sufficient and necessary condition for the minimum processing time.

The reminder of this paper is outlined as follows. Section II presents the optimization model for divisible load scheduling. Section III proves that when workload is large enough, the sufficient and necessary condition for the minimum processing time is that the distribution sequence follows the decreasing order of the communication speeds. A novel genetic algorithm for divisible load scheduling is proposed in Section IV. Experiments and their analysis appear in Section V. In the last section, conclusions are made.

II. Optimization Model for Divisible Load Scheduling

Before we give the scheduling model for divisible load, some relevant notations and assumptions should first be introduced. The platform considered in this paper is heterogeneous distributed systems. Processors are connected in a star topology, where P_0 is the master processor, while $\{P_1, P_2, ..., P_m\}$ are slave processors. P_0 connected with others by communication links $\{L_1, L_2, ..., L_m\}$. The entire workload, denoted as W_{total}, will be first partitioned into fractions. Then they will be distributed to slave processors in some order by the master processor P_0. Processor P_0 does not participate in computation itself but only takes the responsibility of assigning load to others.

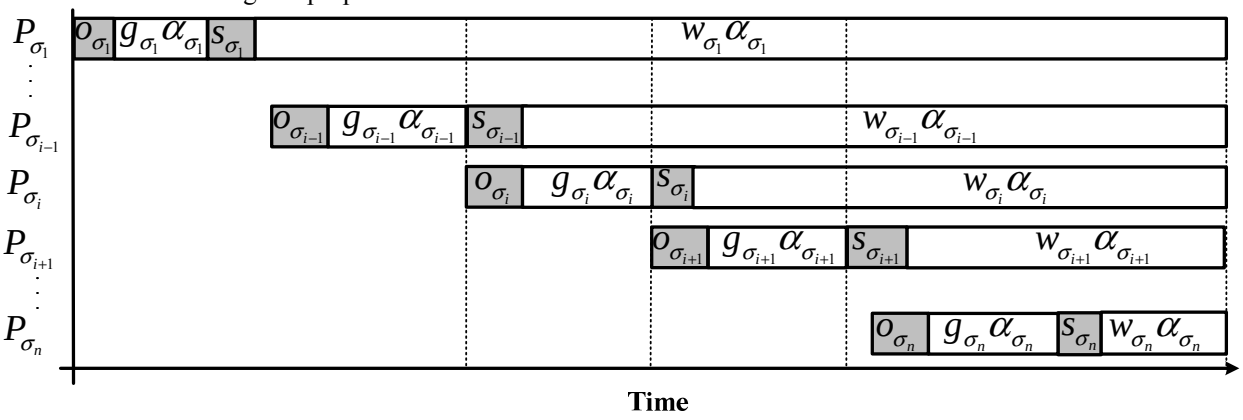


Figure 1. The time diagram of divisible load scheduling on heterogeneous distributed systems in blocking mode of communication.

When distributing workload fractions, P_0 sends data to only one processor at a time and slave processors start computing when they have finished receiving their load fractions, that is, slave processors are in blocking mode of communication and cannot communicate and compute simultaneously.

Since we focus on heterogeneous systems, slave processors are assumed with arbitrary computation and communication speeds. Communication and computation time is proportional to the amount of workload assigned to each processor. Let w_i be the time of processor P_i computing a unit workload, while g_i be the time of link L_i communicating a unit workload. In this paper, we assume communication speeds are much faster than computation speeds; otherwise, only one or two processors should be enough to involve in computation [7].

Besides arbitrary communication speeds and computation speeds, there exists a start-up overhead during each communication from P_0 to P_i, denoted as o_i. Similarly, a start-up overhead s_i exists during each computation of processor P_i.

$$\sum_{i=1}^{n} \alpha_{\sigma_i} = W_{total},\ \alpha_{\sigma_i} > 0, i = 1, 2, ..., n,\ n \le m. \quad (1)$$

Bharadwaj, V., et al proved that for various network models the optimality criterion of scheduling divisible loads is that all processors have to finish computing at the same time. If all processors do not stop computing at the same time, certainly the load can be transferred from busy processors to idle processors to minimize the processing time [25]. Therefore, the following equation can be obtained intuitively.

$$s_{\sigma_i} + w_{\sigma_i}\alpha_{\sigma_i} = o_{\sigma_i} + g_{\sigma_i}\alpha_{\sigma_i} + s_{\sigma_{i+1}} + w_{\sigma_{i+1}}\alpha_{\sigma_{i+1}}, i = 1, 2, ..., n,$$

which can be rewritten as

$$\alpha_{\sigma_{i+1}} = \frac{s_{\sigma_i} - \left(o_{\sigma_{i+1}} + s_{\sigma_{i+1}}\right)}{g_{\sigma_{i+1}} + w_{\sigma_{i+1}}} + \frac{w_{\sigma_i}}{g_{\sigma_{i+1}} + w_{\sigma_{i+1}}}\alpha_{\sigma_i}. \quad (2)$$

Let

$$\varphi_{\sigma_{i+1}} = \frac{s_{\sigma_i} - \left(o_{\sigma_{i+1}} + s_{\sigma_{i+1}}\right)}{g_{\sigma_{i+1}} + w_{\sigma_{i+1}}},\ \delta_{\sigma_{i+1}} = \frac{w_{\sigma_i}}{g_{\sigma_{i+1}} + w_{\sigma_{i+1}}}. \quad (3)$$

Equation (2) can be written as

$$\alpha_{\sigma_{i+1}} = \varphi_{\sigma_{i+1}} + \delta_{\sigma_{i+1}}\alpha_{\sigma_i}, i = 1, 2, ..., n-1 \quad (4)$$

Expressing workload fraction α_{σ_i} in terms of α_{σ_1} as

$$\alpha_{\sigma_i} = \alpha_{\sigma_1}\Gamma_{\sigma_i} + \Phi_{\sigma_i},\ i = 2,3,...,n, \tag{5}$$

where

$$\Gamma_{\sigma_i} = \prod_{k=2}^{i}\delta_{\sigma_k},\ \Phi_{\sigma_i} = \sum_{k=2}^{i}(\varphi_{\sigma_k}\prod_{j=k+1}^{i}\delta_{\sigma_j}),\ i = 2,3,...,n.$$

Combing Eq. (1) and Eq. (5), one can get α_{σ_1} by

$$\alpha_{\sigma_1} = \frac{W_{total} - \left(\Phi_{\sigma_2} + \Phi_{\sigma_3} + ... + \Phi_{\sigma_n}\right)}{1 + \Gamma_{\sigma_2} + \Gamma_{\sigma_3} + ... + \Gamma_{\sigma_n}}. \tag{6}$$

Thus, the closed-form expression of the processing time is given by

$$\begin{aligned} T &= o_{\sigma_1} + s_{\sigma_1} + \left(g_{\sigma_1} + w_{\sigma_1}\right)\alpha_{\sigma_1} \\ &= o_{\sigma_1} + s_{\sigma_1} + \left(g_{\sigma_1} + w_{\sigma_1}\right)\frac{W_{total} - \sum_{k=2}^{n}\Phi_{\sigma_k}}{1 + \sum_{k=2}^{n}\Gamma_{\sigma_k}}. \end{aligned} \tag{7}$$

The main objective of scheduling divisible load on heterogeneous distributed systems is to minimize the processing time T. It can be seen from Eq. (7) that processing time T depends on parameters Φ and Γ, which are directly decided by the distribution sequence $\left(P_{\sigma_1}, P_{\sigma_2}, ..., P_{\sigma_n}\right)$ of the processors participating in computation. Thus the divisible load scheduling problem on heterogeneous distributed systems can be modeled as determining the number n of processors required in computation and the optimal distribution sequence so that the processing time is minimized, that is, an optimization model for scheduling divisible load on heterogeneous distributed systems in blocking mode of communication can be set up as follows.

$$\min T = \min\left[o_{\sigma_1} + s_{\sigma_1} + \left(g_{\sigma_1} + w_{\sigma_1}\right)\frac{W_{total} - \sum_{k=2}^{n}\Phi_{\sigma_k}}{1 + \sum_{k=2}^{n}\Gamma_{\sigma_k}}\right].$$

Subject to:

(1) $n \le m$;

(2) $vector\left(\sigma_1, \sigma_2, ..., \sigma_n\right)$ is a sequence of n different numbers from 1 to m;

(3) $\Gamma_{\sigma_i} = \prod_{k=2}^{i}\left[w_{\sigma_{k-1}} / \left(g_{\sigma_k} + w_{\sigma_k}\right)\right],\ i = 2,3,...,n;$

(4) $\Phi_{\sigma_i} = \sum_{k=2}^{n}\left[\frac{s_{\sigma_{k-1}} - (o_{\sigma_k} + s_{\sigma_k})}{g_{\sigma_k} + w_{\sigma_k}}\prod_{j=k+1}^{i}\frac{w_{\sigma_{j-1}}}{g_{\sigma_j} + w_{\sigma_j}}\right]$, where $i = 2,3,...,n$

(5) $\alpha_{\sigma_1} = \frac{W_{total} - \sum_{k=2}^{n}\Phi_{\sigma_k}}{1 + \sum_{k=2}^{n}\Gamma_{\sigma_k}};$

(6) $\alpha_{\sigma_i} = \alpha_{\sigma_1}\Gamma_{\sigma_i} + \Phi_{\sigma_i},\ i = 2,3,...,n;$

(7) $\alpha_{\sigma_i} > 0,\ i = 1,2,...,n.$

The key issue of solving the above model is to find the optimal distribution sequence. Considering the practical needs, the easier to obtain the optimal distribution sequence, the better. Fortunately, when workload is large enough, such optimal distribution sequence can be obtained directly by theoretical analysis, which is presented in the following section. For other workload cases, the optimal distribution sequence can be derived efficiently by the proposed genetic algorithm introduced in Section IV.

III. SCHEDULING LARGE ENOUGH WORKLOAD

In this section we will prove that when workload is large enough, the sufficient and necessary condition for the minimum processing time is that the distribution sequence follows the decreasing order of the communication speeds. Before we prove this conclusion, the following lemmas are first introduced.

Lemma 3.1 ([13]) *For heterogeneous systems in the blocking mode of communication with zero start-up overheads* $\left(o_i = s_i = 0\right)$ *and arbitrary communication and computation speeds, the distribution sequence should follow the decreasing order of communication speeds in order to achieve the minimum processing time.*

Lemma 3.2 ([13]) *For heterogeneous systems in the blocking mode of communication with start-up overheads, arbitrary computation speeds and all different communication speeds, if the distribution sequence follows the decreasing order of communication speeds, when the processing time T is large enough, the amount of processed workload during the time T is optimal among all possible orderings.*

Theorem 3.3 *For heterogeneous systems in the blocking mode of communication with startup overheads and arbitrary communication and computation speeds, when workload is large enough, the sufficient and necessary condition for the minimum processing time is that the distribution sequence follows the decreasing order of the communication speeds.*

Proof:

(1) Now we prove that when distribution sequence follows the decreasing order of the communication speeds, the corresponding processing time is minimized.

Since the workload is large enough, then all processors should take participate in computation. Assume that there are n processors in the system, and sort them in the decreasing order of their communication speeds, that is, the increasing order of g_i, where $i \in \{1,2,...,n\}$. Assume the distribution sequence is denoted as $\left(P_{\sigma_1}, P_{\sigma_2}, ..., P_{\sigma_n}\right)$ where $g_{\sigma_1} \le g_{\sigma_2} \le ... \le g_{\sigma_n}$. Let T_{opt} be the processing time for the entire workload in that sequence. Since all processors stop processing at the same time, T_{opt} is also the finish time T_{σ_i} of each processor, where $i = 1,2,...,n$.

According to the model shown in Fig.1, we have

$$\begin{aligned} T_{\sigma_i} = T_{opt} &= s_{\sigma_i} + w_{\sigma_i}\alpha_{\sigma_i} + \sum_{k=1}^{i}\left(o_{\sigma_k} + g_{\sigma_k}\alpha_{\sigma_k}\right) \\ &\quad i = 1,2,..,n. \end{aligned} \tag{8}$$

Now let $o_{\sigma_i} = s_{\sigma_i} = 0$ with $i = 1,2,...,n$, which means that the start-up overheads are ignored, thus the finish time T_{σ_i} of processor P_{σ_i} will be advanced by $s_{\sigma_i} + \sum_{j=1}^{i}o_{\sigma_i}$, namely

$$T_{\sigma_i} = T_{opt} - s_{\sigma_i} + \sum_{k=1}^{i}o_{\sigma_k},\ i = 1,2,..,n. \tag{9}$$

When $i=n$, $T_{\sigma_n}=T_{opt}-\left(s_{\sigma_n}+\sum_{k=1}^{n}o_{\sigma_k}\right)$. That is to say, at the time T_{σ_n}, processor P_{σ_n} just finished its workload fraction, while the other processors are still in computation because $T_{\sigma_i}>T_{\sigma_n}$ according to Eq.(8). As is proved in Lemma 3.1, in the case of zero start-up overheads, when the distribution sequence follows the decreasing order of communication speeds, the processing time is the minimum. Let $T_{opt}^{'}$ be the optimal processing time in this case, thus

$$T_{\sigma_n}=T_{opt}-\left(s_{\sigma_n}+\sum_{k=1}^{n}o_{\sigma_k}\right)\le T'_{opt}. \tag{10}$$

Meanwhile, since start-up overheads are ignored, so

$$T'_{opt}\le T_{opt}. \tag{11}$$

From Eq. (10) and Eq. (11), one can get

$$T'_{opt}\le T_{opt}\le T'_{opt}+\left(s_{\sigma_n}+\sum_{k=1}^{n}o_{\sigma_k}\right). \tag{12}$$

Since the workload is large enough, $T'_{opt}\to\infty$. Thus

$$1\le\frac{T_{opt}}{T'_{opt}}\le 1+\frac{\left(s_{\sigma_n}+\sum_{k=1}^{n}o_{\sigma_k}\right)}{T'_{opt}},\ T'_{opt}\to\infty.$$

Thus (1) is proved.

(2) Next we prove that when the processing time is optimal, the distribution sequence must follow the decreasing order of the communication speeds.

This will be proved by contradiction.

If the workload is large enough, then all processors should take part in computation. Without loss of generality, the optimal distribution sequence is denoted as $p_{seq}=\left(P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_n}\right)$. We will prove that $g_{\sigma_i}\le g_{\sigma_{i+1}}$ holds. If it is not true, one can interchange the sequences of P_{σ_i} and $P_{\sigma_{i+1}}$. Since workload size is large enough, and then the processing time T becomes large enough. According to Lemma 3.2, the amount of processed workload in the new order of workload distribution is larger than that of the former sequence, which contradicts the assumption that p_{seq} is the optimal sequence.

Thus (2) is proved.

IV. A New Genetic Algorithm for Divisible Load Scheduling: WX-GA

Task scheduling problems are among the well-known hardest combinatorial optimization problems. Here we choose genetic algorithms(GAs), invented by Holland [26], to solve the above model for the simple reason that genetic algorithms have been proven to be a promising technique for many application problems, for example, optimal design, control, and machine learning, etc. [27,28] and they are suitable to solve scheduling optimization problems [29].

A. Population Initialization

The key point of finding the optimal distribution sequence by using genetic algorithms is to develop an encoding scheme that allows genetic operators to generate "legitimate children" without any constraint violation. Applying GA to the scheduling model proposed in this paper has an intrinsic issue: each sequence must contain exactly one instance of a processor and any omission or duplication of a processor or processors leads to an illegal distribution sequence.

In this paper, a sequence with m processors is directly represented as a permutation S_{pm} of m elements from 1 to m. Thus an individual I is denoted by $I=\left(n,S_{pm}\right)$, where n represents the number of processors taking part in computation.

After determining the encoding scheme, one can generate an initial population of N individuals by Algorithm 1.

Note that the first gene in each individual, which denotes the number of processors used in computation, will be initialized to m, the total number of processors. It will be modified during the calculation of its cost value.

Algorithm 1 Generate initial population

Ensure: N initialized individuals $\left(I^1,I^2,...,I^N\right)$

1: **for** $i=1,2,...,N$ **do**
2: generate individual $I^i=\left(c_0^i,c_1^i,...,c_m^i\right)$ as follows;
3: let $c_0^i=m$;
4: given an ordered list $L=1,2,...,m$;
5: **for** $j=1,2,...,m$ **do**
6: randomly take an element from L, assign it to c_j^i and then delete this element from L.
7: **end for**
8: **end for**

B. Cost Function

The cost function, also called fitness function, is defined over the genetic representation and measures the quality of the represented solution [30]. In general, a cost function is derived from the main objective of the problem and used in successive genetic operations. The main objective of scheduling divisible load on heterogeneous distributed systems is minimizing the processing time. How to map this objective to cost function is presented in Algorithm 2.

Algorithm 2 Cost function

Require: An individual $I=\left(c_0,c_1,c_2,...,c_m\right)$ and a workload W_{total}

Ensure: The cost function value $f(I)$ of individual I, the processing time T, the number c_0 of processors required in computation and the workload fraction α_i for each processor with $i=1,2,...,c_0$.

1: **for** $i=1,2,...,m-1$ **do**
2: calculate φ_{i+1} and δ_{i+1} by Eq. (3);
3: **end for**
4: **for** $i=2,...,m$ **do**
5: calculate Γ_i and Φ_{i+1} by Eq. (5);

6: **end for**
7: compute α_1 and the processing time T by Eq. (6) and Eq. (7) respectively;
8: **for** $i = c_0, c_0 - 1, ..., 1$ **do**
9: compute α_i according to Eq. (5);
10: **if** $\alpha_i \le 0$ **then**
11: let $c_0 = c_0 - 1$ and go to step7;
12 **endif**
13: **end for**

C. Weight-based Crossover Operator

Depending on how the chromosome represents the solution, a direct swap such as N-point crossover may not be possible since the recombination of chromosome may violate the constraint of ordering and thus need to be repaired. In this paper, a novel crossover operator called weight-based crossover is designed to generate offspring.

The easiest way to explain the weight-based crossover is by giving an example. Assume that there exists an ordered list L = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1), which serves as a reference sequence, and that the two parents are represented by

$$parent_1 = (8, 3, 1, 2, 5, 7, 8, 9, 6, 4, 10) \text{ and}$$

$$parent_2 = (7, 3, 1, 8, 6, 5, 2, 7, 10, 9, 4),$$

where the number of processors used in $parent_1$ is 8 and that in $parent_2$ is 7.

The weight-based crossover operator should be executed on $parent_1$ and $parent_2$ as follows.

(1) First according to the number of processors used in computation, assign weight values to each gene in $parent_1$ and $parent_2$ as follows. As is shown in Table I, for individual $parent_1 = (c_0, c_1, ..., c_m)$, since the number of processors needed in computation is $c_0 = 8$, then for the former 8 genes, from left to right, each of them takes the element of L at the same position as their weight values. That is to say, the weight value of c_i for $parent_1$ is assigned to L_i, where $i = 1, 2, ..., c_0$. For the remaining genes, which represent the processors that do not take part in computation, let their weight values be zeros. Finally, we have the weight list for $parent_1$: (10, 9, 8, 7, 6, 5, 4, 3, 0, 0). Similarly, the weight list (10, 9, 8, 7, 6, 5, 4, 0, 0, 0, 0) for $parent_2$ can be obtained.

TABLE I.

AN EXAMPLE OF CROSSOVER OPERATOR.

List		10	9	8	7	6	5	4	3	2	1
$parent_1$	8	3	1	2	5	7	8	9	6	4	10
$weight_1$		10	9	8	7	6	5	4	3	0	0
$parent_2$	7	3	1	8	6	5	2	7	10	9	4
$weight_2$		10	9	8	7	6	5	4	0	0	0
offspring	10	3	1	8	2	5	6	7	4	9	10
sumweight		20	18	13	13	13	10	10	4	0	0

(2) Based on the weight lists of $parent_1$ and $parent_2$, the weight value of each processor is calculated in the following way. For each processor, say P_1, since its weight in $parent_1$ is 9 and in $parent_2$ is also 9, take the sum of them 9 + 9 = 18 as the final weight of processor P_1. Repeat the above process until the weights of all processors are calculated. Finally, for processors $(P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10})$, the weight list (18, 13, 20, 4, 13, 10, 10, 13, 0, 0) can be obtained.

(3) Sort the processors in the descending order of their weights. Thus we have the sorted processor list $(P_3, P_1, P_2, P_5, P_8, P_7, P_6, P_4, P_9, P_{10})$ and its corresponding weight list (20, 18, 13, 13, 13, 10, 10, 4, 0, 0).

(4) Adjust the order of the processors with the same weight values. For the processors with the same non-zero weight, such as (P_6, P_7), compare the positions of them in their parents. In $parent_1$, P_7 is at position 5 and appears in front of P_6, while in $parent_2$, P_6 is at position 4 and appears in front of P_7. Thus we put P_6 with the smaller position indicator in front of P_7 in the final processor list. If processors are at the same position in their parents, such as (P_2, P_8), then we can randomly select one of them in front of the other. For the processors with zero weight values, since they do not take part in computation, their orders do not matter.

(5) Generate an offspring $O = (c_0', c_1', c_2', ..., c_m')$ according to the sorted processor list $(P_3, P_1, P_8, P_2, P_5, P_6, P_7, P_4, P_9, P_{10})$, where c_0' is initialized to m. Finally we can get $O =$ (10, 3, 1, 8, 2, 5, 6, 7, 4, 9, 10).

The proposed weight-based crossover operator has two advantages: first, it keeps the optimal subsequence of the parents into offspring, such as (P_3, P_1); second, it takes the order of processors in the distribution sequence into consideration because the more front the position of a processor is in the distribution sequence, the much load fraction it will be assigned, and thus the more important the processor is.

D. Mutation Operator

If we randomly change one number in a chromosome, we are left with one integer duplicated and another missing. The simplest solution is to randomly choose a chromosome to mutate, and then randomly choose two positions within that chromosome to exchange until a local optimal is found. The process of the mutation operator is shown in Algorithm 3 by the pseudo-code.

Algorithm 3 Mutation operator

Require: An individual $I = (c_0, c_1, c_2, ..., c_m)$ and mutation probability p_{mut}.

Ensure: An offspring $h = (h_0, h_1, ..., h_m)$ generated by mutation.

1. Randomly generate a real number $r \in [0,1]$;
2: **if** $r \le p_{mut}$ **then**
3: let $h = I$;

4: evaluate the cost value $f(h)$ of individual h by Algorithm 2;
5: let $count = 0$ and $f^* = f(h)$;
6: randomly generate two integers $a, b \in [1, m]$;
7: swap h_a and h_b;
8: evaluate the cost value $f(h)$ of individual h by Algorithm 2;
9: **if** $f(h) \geq f^*$ **then**
10: swap back h_a and h_b and let $count = count + 1$;
11: if $count > m$ stop; otherwise go to step 6;
12: **else**
13: go to step 5;
14: **endif**
15: **endif**

E. A new GA for scheduling divisible loads: WX-GA

Once the encoding scheme and the cost function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover and selection operators. Algorithm 4 presents the process of the proposed genetic algorithm: WX-GA.

Algorithm 4 A new genetic algorithm for scheduling divisible load on heterogeneous systems

1: (**Initialization**) Choose population size N, proper crossover probability p_{cros} and mutation probability p_{mut} etc. Randomly generate initial population $P(0)$ by Algorithm 1. Let the generation number $t = 0$.
2: (**Crossover**) Choose the parents for crossover in $P(t)$ with probability p_{cros}. If the number of parents chosen is odd, then randomly choose an additional one from $P(t)$. Afterwards, randomly match every two parents as a pair and the specific-designed weight based crossover operator is used on each pair to generate one offspring. All the new offspring constitute a set denoted by O_1.
3: (**Mutation**) Select the parents for mutation from set O_1 with probability p_{mut}. For each chosen parent, the proposed mutation operator is used to generate a new offspring. These new offspring constitute a set by O_2.
4: (**Selection**) The best E solutions from the set $P(t) \cup O_1 \cup O_2$ are maintained in the next generation so that the convergence is faster. The roulette-wheel selection is used to select $N - E$ individuals among the set $P(t) \cup O_1 \cup O_2$ as the next generation population $P(t+1)$.
5: **Stopping Criteria:** If termination conditions hold, then stop, and keep the best solution obtained as the global optimal solution of the problem; otherwise, go to step 2.

V. Experiments and Analysis

Several experiments are presented in this section to show the effectiveness and efficiency of the proposed algorithm. The parameters of the heterogeneous distributed system are shown in Table II [12]. In the proposed genetic algorithm, the following parameters are adopted: population size $N = 100$, crossover probability $p_{cros} = 0.6$, mutation probability $p_{mut} = 0.02$, elitist number $E = 5$ and stop criterion $t = 2000$.

TABLE II.
PARAMETERS OF THE HETEROGENEOUS DISTRIBUTED SYSTEM

P	o	s	g	w
p_1	70.554750	57.951860	0.533424	2.895625
p_2	30.194800	1.4017640	0.774740	7.607236
p_3	81.449010	4.535275	0.709038	4.140327
p_4	86.261930	37.353620	0.790480	9.619532
p_5	87.144580	94.955670	0.056237	3.640187
p_6	52.486840	5.350452	0.767112	5.924582
p_7	46.870010	62.269670	0. 298165	6.478212
p_8	26.379290	82.980160	0. 279342	8. 246021
p_9	58.916300	91.096430	0. 986093	2.268660
p_{10}	69.511550	24.393140	0. 980003	5.338731
p_{11}	10.636970	67.617590	0. 999415	1.570390
p_{12}	57.518380	10.302260	0. 100052	7.988844
p_{13}	28.448030	29.577290	0. 045649	3.820107
p_{14}	30.090500	97.982940	0. 948571	4.013743
p_{15}	27.828000	16.282160	0. 160442	6.465871

In the first experiment, the workload size ranges from 100 to 10000. For convenience, let GA represents the algorithm WX-GA proposed in this paper, IG indicates the algorithms given by Beaumont et al. [13] and Shang [12], which schedules divisible load in the sequence of increasing value of g_i, while IW given by Bharadwaj et al. [10], which schedules workload in the sequence of increasing value of w_i. The processing time and used machine numbers for different workload by these three scheduling algorithms are recorded in Table III.

It can be seen from Table III that the processing time used by the proposed algorithm is much less than that used by other two compared algorithms for all test cases.

Fig. 2(a) intuitively shows the processing time difference between algorithm IW and WX-GA, while Fig. 2(b) shows the time difference between IG and WX-GA. As is shown in Fig. 2(a), with the increasing size of workload, the difference of processing time is found in linear growth between the proposed algorithm IW and WX-GA. In other words, the proposed algorithm is much more effective in finding the optimal distribution sequence. Similarly, it can be seen from Fig. 2(b) that when workload size ranges from 100 to 9900, the processing time obtained by IG is larger than that by WX-GA. What's more, we show the numbers of processors that are selected to participate in computation obtained by algorithm WX-GA and IG in Fig. 3.

TABLE III.

PROCESSING TIME FOR DIFFERENT WORKLOAD FROM 100 TO 9900.

Alg	Size	No	Time	Size	No	Time	Size	No	Time	Size	No	Time
GA		5	209.255		7	411.123		9	572.316		10	700.423
IG	100	4	285.379	300	8	475.671	500	10	609.918	700	10	735.025
IW		2	295.481		3	581.707		5	809.344		7	1004.95
GA		11	818.945		12	930.701		12	1041.08		13	1149.98
IG	900	11	859.494	1100	11	980.935	1300	13	1096.19	1500	13	1207.3
IW		8	1199.39		9	1386.48		10	1569.33		10	1748.7
GA		13	1256.35		13	1362.33		13	1468.04		14	1573.51
IG	1700	15	1318.42	1900	15	1426.03	2100	15	1526.87	2300	15	1627.71
IW		10	1928.07		12	2102.19		12	2274.43		13	2444.43
GA		14	1677.21		14	1779.98		14	1882.75		14	1985.41
IG	2500	15	1728.54	2700	15	1829.38	2900	15	1930.22	3100	15	2031.06
IW		13	2613.90		13	2783.37		13	2952.85		14	3119.95
GA		15	2087.43		15	2189.35		15	2291.26		15	2393.17
IG	3300	15	2131.89	3500	15	2232.73	3700	15	2333.57	3900	15	2434.4
IW		15	3286.37		15	3451.4		15	3616.43		15	3781.46
GA		15	2494.61		15	2595.9		15	2697.19		15	2798.48
IG	4100	15	2535.24	4300	15	2636.08	4500	15	2736.91	4700	15	2837.75
IW		15	3946.49		15	4111.52		15	4276.55		15	4441.58
GA		15	2899.77		15	3001.06		15	3102.35		15	3203.63
IG	4900	15	2938.59	5100	15	3039.43	5300	15	3140.26	5500	15	3241.10
IW		15	4606.61		15	4771.64		15	4936.67		15	5101.70
GA		15	3304.92		15	3406.21		15	3507.5		15	3608.79
IG	5700	15	3341.94	5900	15	3442.77	6100	15	3543.61	6300	15	3644.45
GA		15	5266.73		15	5431.76		15	5596.79		15	5761.82
IG		15	3710.07		15	3811.3		15	3912.54		15	4013.77
IW	6500	15	3745.28	6700	15	3846.12	6900	15	3946.96	7100	15	4047.79
IG		15	5926.85		15	6091.88		15	6256.91		15	6421.94
IW		15	4115.00		15	4216.23		15	4317.47		15	4418.70
IW	7300	15	4148.63	7500	15	4249.47	7700	15	4350.31	7900	15	4451.14
IW		15	6586.97		15	6752.00		15	6917.03		15	7082.06
IW		15	4519.93		15	4621.17		15	4722.4		15	4823.63
IW	8100	15	4551.98	8300	15	4652.82	8500	15	4753.65	8700	15	4854.49
IW		15	7247.09		15	7412.12		15	7577.15		15	7742.17
IW		15	4924.86		15	5026.10		15	5127.33		15	5228.56
IW	8900	15	4955.33	9100	15	5056.16	9300	15	5157.00	9500	15	5257.84
IW		15	7907.2		15	8072.23		15	8237.26		15	8402.29
IW		15	5329.79		15	5431.03						
IW	9700	15	5358.67	9900	15	5459.51						
IW		15	8567.32		15	8732.35						

From Fig. 2(b) and Fig. 3, we can come to the following conclusions: (1) For divisible load scheduling problems on heterogeneous distributed systems in blocking mode of communication, the proposed algorithm is much more effective than other compared algorithms because processors involved in computation after scheduling by WX-GA is fewer than that by IG and meanwhile the processing time obtained by WX-GA is smaller than that by other compared algorithms. (2) Distribution sequence plays a significant influence on the processing time, since algorithms with different distribution schemes lead to distinct experimental results.

In the second experiment, the workload size ranges from 1×10^4 to 2×10^5. The processing time and used machine numbers for different workload by Algorithm WX-GA, IW and IG are recorded in Table IV.

It can be seen from Table IV that the processing time used by the proposed algorithm is less than or at least equal to that used by other two compared algorithms for all test cases. Fig. 4(a) shows the processing time difference between algorithm IW and WX-GA, while Fig.4(b) shows the time difference between IG and WX-GA.

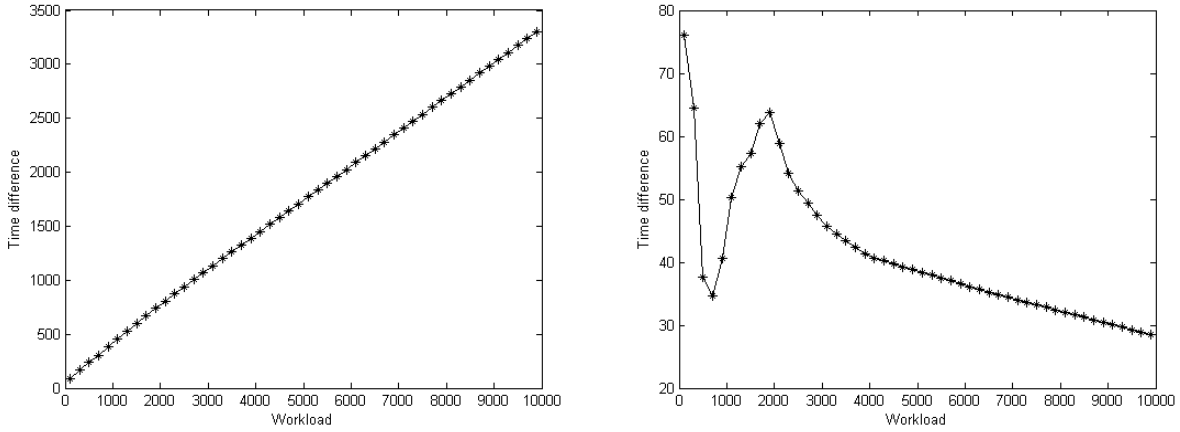


Figure 2. Comparative experiment 1: (a) the processing time difference between IW and WX-GA. (b) the time difference between IG and WX-GA.

From Fig. 4(a), one can see that the larger the workload, the greater the difference of the processing time between IW and WX-GA. On the contrary, Fig. 4(b) shows that the larger the workload, the smaller the difference of the processing time between IG and WX-GA. Especially, when workload is large enough, the processing time obtained by WX-GA and IG are the same, because all processors are needed in computation and the two scheduling algorithms get the same distribution sequence, which follows the decreasing order of the communication speeds. The experimental results are consistent with the conclusion given by Theorem 3.3 in Section 3. That is to say, the experimental results reconfirmed the validity of the conclusion.

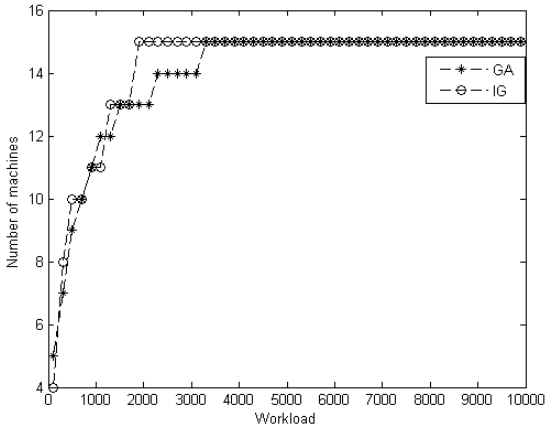


Figure 3. Numbers of processors selected to participate in computation obtained by algorithm WX-GA and IG.

TABLE IV.

PROCESSING TIME FOR DIFFERENT WORKLOAD FROM 1×10^4 TO 2×10^5 .

Alg.	Size	No.	Time	Size	No	Time	Size	No.	Time
GA		15	5481.64		15	10535.0		15	15579.5
IG	1×10^4	15	5509.93	2×10^4	15	10551.8	3×10^4	15	15593.6
IW		15	8814.87		15	17066.4		15	25317.8
GA		15	20622.8		15	25665.7		15	30708.6
IG	4×10^4	15	20635.5	5×10^4	15	25677.3	6×10^4	15	30719.2
IW		15	33569.3		15	41820.8		15	50072.3
GA		15	35751.5		15	40794.5		15	45837.4
IG	7×10^4	15	35761	8×10^4	15	40802.9	9×10^4	15	45844.7
IW		15	58323.8		15	66575.3		15	72429.2
GA		15	50880.3		15	55923.2		15	60966.2
IG	1×10^5	15	50886.6	1.1×10^5	15	55928.4	1.2×10^5	15	60970.2
IW		15	83078.3		15	91329.7		15	99581.2
GA		15	66009.1		15	71052.0		15	76094.9
IG	1.3×10^5	15	66012.1	1.4×10^5	15	71053.9	1.5×10^5	15	76095.8
IW		15	107833		15	116084		15	124336
GA		15	81137.6		15	86179.5		15	91221.3
IG	1.6×10^5	15	81137.6	1.7×10^5	15	86179.5	1.8×10^5	15	91221.3
IW		15	132587		15	140839		15	149090
GA		15	96263.2		15	101305			
IG	1.9×10^5	15	96263.2	2×10^5	15	101305			
IW		15	157342		15	165593			

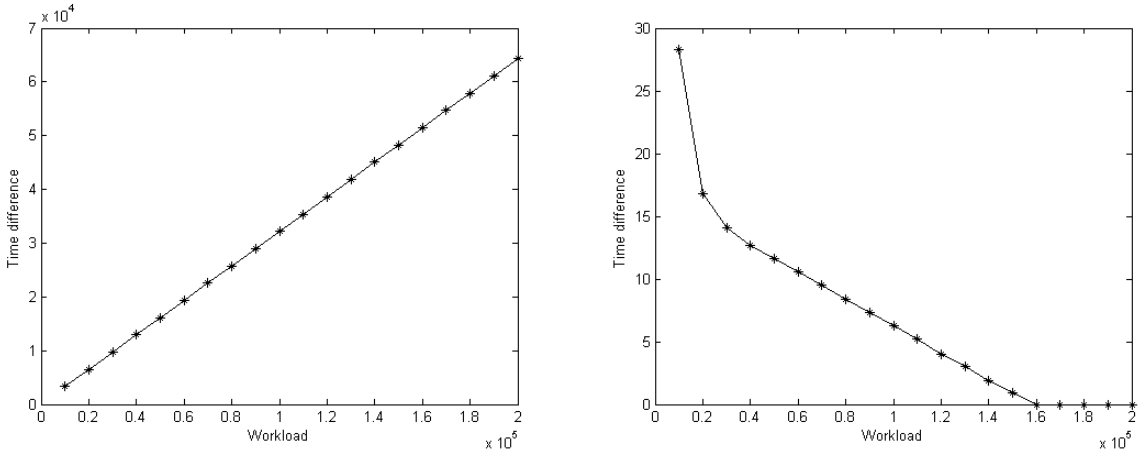


Figure 4. Comparative experiment 2: (a) the processing time difference between IW and WX-GA. (b) the time difference between IG and WX-GA.

VI. CONCLUSIONS

The goal of this paper was to find an optimal scheduling for divisible load on heterogeneous distributed systems in blocking mode of communication. The goal was successfully achieved by designing a novel genetic algorithm WX-GA. In order to examine the performance of the proposed algorithm, a set of experiments were carried out. From the experimental results, the following conclusions can be drawn. First, distribution sequence plays a significant role in processing time. Second, the proposed algorithm greatly decreases the processing time. Third, when workload is large enough, in order to achieve minimum processing time, the optimal distribution sequence should follow the decreasing order of the communication speeds.

ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China (No.61272119).

REFERENCES

[1] T.G. Robertazzi, "Ten reasons to use divisible load theory," *Computer*, vol. 26, no. 5, pp. 63-68, 2003.

[2] V. Mani, D. Ghose, "Distributed Computation in Linear Networks: Closed–Form Solutions," *IEEE transactions on aerospace and electronic systems*, vol. 30, no. 2, pp. 471–483, 1994.

[3] S. Bataineh, T.G. Robertazzi, "Distributed Computation in Linear Networks: Closed–Form Solutions," *Proc. Conf. Information Sciences and Systems, Princeton Univ., Princeton*, vol. 30, no. 2, pp. 794–799., 1992.

[4] D. Ghose, V. Mani, "Distributed Computation with Communication Delays: Asymptotic Performance Analysis," *Journal of parallel and distributed computing*, vol. 23, no. 3, pp. 293–305, 1994.

[5] V. Bharadwaj, D. Ghose, and V. Mani, "Optimal Sequencing and Arrangement in DistributedSingle–Level Networks with Communication Delays," *IEEE transactions on parallel and distributed systems*, vol. 5, no. 9, pp. 968–976, 1994.

[6] H. J. Kim, G.I. Jee, and J.G. Lee, "Optimal Load Distribution for Tree Network Processors," *IEEE transactions on aerospace and electronic systems*, vol. 32, no. 2, pp. 607–612, 1996.

[7] H.J. Kim, "A novel optimal load distribution algorithm for divisible load," *Cluster computing-the journal of networks software tools and applications*, vol. 6, no. 1, pp. 41–46, 2003.

[8] H.J. Kim, V. Mani, "Divisible load scheduling in single–level tree networks: optimal sequencing and arrangement in the nonblocking mode of communication," *Computers & mathematics with applications*, vol. 46, no. (10–11), pp. 1611–1623, 2003.

[9] S. Suresh, V. Mani, and S.N. Omkar, "The effect of start–up delays in scheduling divisible load on bus networks: an alternate approach," *Computers & mathematics with applications*, vol. 46, no. (10–11), pp. 1545–1557, 2003.

[10] V. Bharadwaj, Xiaolin Li, and C.C. Ko, "On the influence of start–up costs in scheduling divisible load on bus networks," *IEEE transactions on parallel and distributed systems*, vol. 11, no. 12, pp. 1288–1305, 2000.

[11] J. Blazewicz, M. Drozdowski, "Distributed processing of divisible jobs with communication startup costs," *Discrete Applied Mathematics*,, vol. 76, no. (1-3), pp. 21-41, 1997.

[12] M.S. Shang, "Optimal algorithm for scheduling large divisible workload on heterogeneous system," *Applied mathematical modelling*,, vol. 32, no.9, pp. 1682–1695, 2008.

[13] O. Beaumont, A. Legrand, and Y. Robert, "Scheduling divisible workloads on heterogeneous platforms," *Parallel computing*,, vol. 29, no.9, pp. 1121–1152, 2003.

[14] J. Berlinska, M. Drozdowski, M. Lawenda, "Experimental study of scheduling with memory constraints using hybrid methods," *Journal of Computational and Applied Mathematics*, vol. 232, no.2, pp. 638–654, 2009.

[15] J. Berlinska, M. Drozdowski, "Heuristics for multi-round divisible loads scheduling with limited memory," *Parallel Computing*, vol. 36, no.4, pp. 199–211, 2010.

[16] J. Berlinska, M. Drozdowski, "Scheduling Divisible MapReduce Computations," *Journal of Parallel and Distributed Computing*, vol. 71, no.3, pp. 450–459, 2011.

[17] T.G. Robertazzi, "A Product Form Solution for Tree Networks with Divisible Loads," *Parallel Processing Letters*, vol. 21, no.1, pp. 13-20, 2011.

[18] K. Li, "New Divisible Load Distribution Methods using Pipelined Communication Techniques on Tree and Pyramid Networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no.2, pp. 806–819, 2011.

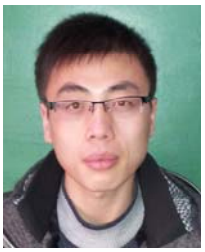
[19] A. Shokripour, M. Othman, H. Ibrahim, and S. Subramaniam, "A New Method for Scheduling Divisible

Data on a Heterogeneous Two-Levels Hierarchical System," *Procedia Computer Science*, vol. 4, pp. 2196–2205, 2011.
[20] A. Shokripour, M. Othman, H. Ibrahim, and S. Subramaniam, "A Method for Scheduling Heterogeneous Multi-Installment Systems," *Lecture Notes in Artificial Intelligence, Springer*, vol. 6592, pp. 221–230, 2011.
[21] A. Shokripour, M. Othman, H. Ibrahim, and S. Subramaniam, "A New Method for Job Scheduling in a Non-Dedicated Heterogeneous System," *Procedia Computer Science, Elsevier*, vol. 3, pp. 271–275, 2011.
[22] C.F. Gamboa, T.G. Robertazzi, "Simple Performance Bounds for Multicore and Parallel Channel Systems," *Parallel Processing Letters*, vol. 21, no. 4, pp. 439–459, 2011.
[23] G. Barlas, "Cluster-Based Optimized Parallel Video Transcoding," *Parallel Computing*, vol. 38, pp. 226–244, 2012.
[24] A. Shokripour, M. Othman, H. Ibrahim, S. Subramaniam, "New method for scheduling heterogeneous multi-installment systems," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1205–1216, 2012.
[25] V. Bharadwaj, D. Ghose, V. Mani, T.G. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems," *IEEE Computer Society Press*, *Los Alamitos,CA,*1996.
[26] J.H. Holland, "Adaptation in Natural and Artificial Systems," *University of Michigan Press, Ann Arbor*, 1975.
[27] Liu, D. L., Chen, X. H., & Du, J. L. A Hybrid Genetic Algorithm for Constrained Optimization Problems. Journal of Computers, 8(2), 272-278, 2013.
[28] Zebin, Tahmina, and M. S. Alam. Modeling and Control of a Two-link Flexible Manipulator using Fuzzy Logic and Genetic Optimization Techniques. Journal of Computers 7.3: 578-585. 2012.
[29] ZHAO, Jian-Hua; LI, Wei-Hua. Intrusion Detection Based on Improved SOM with Optimized GA. Journal of Computers, 8(6): 1456-1463.2013.
[30] C.R. Reeves, J.E. Rowe, "Genetic algorithms–principles and perspectives: a guide to GA theory," *Kluwer Academic Publish*, 2012.

Xiaoli Wang and is currently a Ph.D. candidate at Xidian University, and major in computer science and technology.

She was born in ShanDong Province. She received her BS degree in software engineering from Xidian University in 2004. She is a member of the ACM. Her research interests include parallel and distributed systems, grid computing and cloud computing.

Yuping Wang is a professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. He got Ph.D from the Department of Mathematics, Xi'an Jiaotong University, China in 1993. He is the Senior member of IEEE, and visited Chinese University of Hong Kong, City University of Hong Kong, and Hong Kong Baptist University as a research fellow many times from 1997 to 2010. He has authored and co-authored over 100 research papers in journals and conferences. His current research interests include evolutionary computation, non-bilevel programming, global optimization, and multi-objective programming.



Kun Meng is currently a postgraduate student at Xidian University, and major in computer science and technology, Xi'an, China. His current research interests include task scheduling and evolutionary algorithms.