METHODS AND MEANS FOR SOLUTION PROBLEM OF DISTRIBUTED PARALLEL DATA PROCESSING

Camofalov K., Simonenko V.
Department of Computer Science, National Technical University of Ukraine, Kiev,Ukraine.

Abstract

This paper presents a new approach to solve the problem of job scheduling in parallel heterogeneous systems. We introduce a classification for the given job scheduling problem by the heterogeneity of the systems, from the view of the schedulers' eyes. Then, according to this analysis, a new scheduling strategy for so-called "*Strictly-Heterogeneous*" systems is proposed.

Keywords.
Job scheduling, optimal criteria, time complexity, multi-processor systems.

1. Introduction.

In large parallel computing systems, as the number of the users and the resources increases, processing scheduling becomes more important as a means to achieve the best of parallelism by an efficient processing and high system utilization.

For the last few years, a number of scheduling algorithms for uniform systems has been published. However, heterogeneous systems and uniform systems are different, therefore scheduling strategies for them, consequently, are also different. Besides, a lot of algorithms for uniform systems sometimes are specialized on a particular or a standard uniform type of the system architecture [1][2][3].

In this paper we study job scheduling for parallel heterogeneous systems, where not only computing nodes are heterogeneous but also tasks (jobs) can belong to different classes as well. We provide the Objective-Oriented Algorithm based on a strategy which has not yet been commonly used before in scheduling.

2. Heterogeneous Computing Model.

2.1. Description of System Model.

As stated above, we will study scheduling strategy for " *Strictly-Heterogeneous* " systems. The computing model, in which our scheduling algorithm works, has been derived from a large parallel and distributed system and has been studied before in [4]. In the real world, this system consists of : First, *different resources,* they are heterogeneous nodes-users Ui (e.g. different computers, or processors) and the commonly-used resources CRj (e.g. the severs) ; Second, *tasks of different kinds* T_l, which come from users-nodes or from outside (e.g. from other systems). As in any parallel systems, these so-called mother-tasks are to be maximally parallelized. They are divided into son-tasks (small computing modules) which we will call jobs. These *jobs are different* as well because they come from different tasks.

Therefore, the given system can be represented by:

- A set of **Q** tasks ST={T_1, ...,T_Q} with all their heterogeneous features.

- a set of **N** resources SR={R_1, ...,R_N} with all their heterogeneous features;
- A matrix of communication channels between them MCR[1..N,1..N], where MCR[i,j] is the weight (cost) of communication between R_i and R_j , and MCR[i,j] $\in \Re^+$. We say there is "no connection" between R_i and R_j when MCR[i,j] > Ω_0 **(**some given number)**.**

2.2. Statement for Problem of Job Scheduling.

As is discussed above, the whole job scheduling process is a complex of procedure-steps. In this paper we will focus on the most important step — step of scheduling jobs onto resources. In more detail, it can be stated as follows:

At a moment in time there are:

- N heterogeneous resources of the system, which are represented by a graph $G_R=(V_R,E_R,W_{VR},W_{ER})$, where:
  - V_R= {R_1, R_2, ..., R_N}is the set of N resources-nodes, $R_i \in \mathbb{N}|$, i=1..N.
  - E_R={E_1, E_2, ..., E_d}is the set of edges, which represent the physical communication links between resources E_i = {R_i,R_j}, where R_i, $R_j \in V_R$ and $0 \le d \le N^2$.
  - W_{VR}={ WVR_1, WVR_2, ..., WVR_N }is the set of nodes' weights, where WVR_i ={RE_i, RT_i}. For $\forall$ i=1..N, (i): $RE_i \in \Re^+$ is a ratio that characterizes the capacity (e.g. by the speedup, by local memory) of the given resource R_i ; (ii): $RT_i \in \{0,1\}$ is the state of the resource (free or occupied).
  - W_{ER}={ WER_1, WER_2, ..., WER_p } is the set of edges' weights and it can be represented by a matrix RC=RC[i,j] $\in \Re^+$, where i=1..N, j=1..N, and $0 \le p \le M*N$.
- M independent (no precedent relationship between each other) and different jobs, which are represented by a set V_J={J_1, J_2, ..., J_M}. The heterogeneity of jobs is characterized by data sets J_i ={JN_i, JE_i, JL_i}, i=1..M, where:
  - $JN_i \in \mathbb{N}|$ is ID of the job (e.g. the number).
  - $JE_i \in \Re^+$ is the work amount for executing the job.
  - JL_i ={(R^1, φ_1), ..., (R^q, φ_q) } represent the logical communication links of the given job to the resources, where $R^l \in V_R$ is the resource which the given job need to communicate with (it is determined by the resource where the predecessor-job of the job has been executed) , $\varphi_1 \in \Re^+$ is the amount of data transfer, l=1..q, $q \in \mathbb{N}|$.
  - $JP_i \in \Re^+$ is the priority of the job *i* if the priority system for jobs exists.
  - The requirement for the given problem is to find out a schedule for M jobs onto N resources so that the processing achieves the two following optimization goals: minimum total actual execution time or shortest response time for each job.

3. Solution Basis.

3.1. Related work.

It has been shown that the given problem is NP-hard [5]. Therefore, most of scheduling algorithms use heuristic or genetic approaches[6][8][13][15]. Besides, there are two important issues that are be to considered, while solving a scheduling problem. They are: *solution quality* (how the received schedule is near to the optimum one) and *solving time* (scheduler execution time).

In dealing with solving time: The requirement for solving time depends on the type of scheduling. Static scheduling algorithms can have a long solving time while the Dynamic ones always have a short solving time.

As is said above, because of the complexity of the problem, most of the scheduling algorithms are heuristic or genetic. In the algorithms that have been published recently, a genetic method called Simulated Annealing (SA) is used very commonly [6][8][7]. This

method is good because of its flexibility. By resetting the values of the parameters for the simulation (the initial and freezing temperatures, the ratio for decreasing temperatures) we can receive many kinds of schedulers, which are different by the ratio between solution time and solution quality as follows:

- From the fastest scheduler, which gives a schedule with a random quality for the minimum solution time;
- To the slowest one, which checks all possible variants of schedules and gives us the exact solution (real optimum schedule).

Usually, the simulation parameters are set so that the result is in the middle between these two extremists. However, the solution time for achieving an acceptable-optimum schedule is too long, especially when the problem size (M,N) is great., and also, the solution quality is unpredictable.

3.2. New Approach with Hungarian method.

As it has been reviewed above, in order to achieve two conflicting goals: short scheduling time and good quality of the schedule, we always have a trade-off between solving time and solution quality. The point is how to achieve the “golden mean”.

Scheduling Strategy for dealing with solving time: For the system that is described above, our goal is to develop a balanced algorithm, which gives us a schedule of good quality for an acceptable solving time. In order to see explicitly the difference of our approach from the existing ones, let us analyze again (but now from the view of strategies) the above approach of scheduling by using simulated annealing (SA):

Suppose that all possible variants (**Ri**) of schedules can be set as the “balls” in a “box”. The variants-balls are located in a chaos. Among them, there is a real optimum one that is to be found. Now, see how it is found by SA and by our algorithm with using Hungarian method [10][11].

In SA, all the variants are put in the Markovian chain as all the balls are connected each to another, with a visual “thread” (Figure 1.a). The search is started with a random variant-ball in the “pocket” and is guided by this thread. During the search, the next ball is compared with the one in the pocket; If the first is better than the second then it will occupy the pocket, and so on. After a given number of steps, the variant-ball in the pocket is thought of the optimum one of all the balls in the box. However, if the real optimum variant-ball is not the searched area then the received variant is not optimum, obviously.

The key idea of the new approach is to detach from all possible variants (the box) a zone, which contains exactly the real optimum variant **Vop** (for the given optimization criteria) and is less than the box (Figure 1.b.). Then, the search is carried out only in this area until it reaches the real optimal variant **Vop**. Therefore, we will receive the exact solution while the solving time is much less than it is in SA for finding a solution of the same quality.

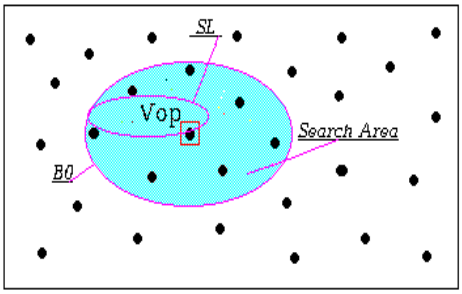

Figure 1.b. New scheduling strategy.

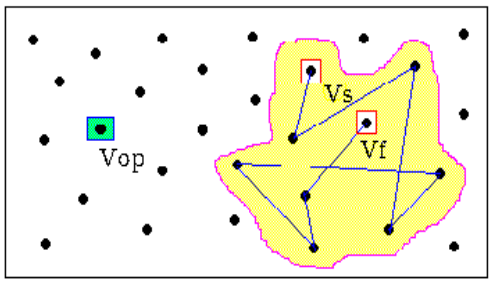

Figure 1.a. Scheduling strategy in SA

Moreover, using the Hungarian method (which will be described in detail later) for such an approach allows us to carry out not a random search (as it is in SA) but a so-called "*objective-oriented search"* in the chosen area, where the objective is the real optimum variant **Vop.** That is why we name the algorithm "Objective-Oriented" (OOA). The selection of the Hungarian method is not by chance but it is based on a careful investigation that is studied in [12].

Scheduling Strategy for dealing with solution quality: We have to determine the optimization criterion and the scale of optimization area.

- Heterogeneous systems usually are distributed. Therefore, we think it is necessary to consider communication costs in scheduling for such systems. To escape the conflicting goals: minimizing execution time of jobs (by maximally parallelizing them on the maximum number of resources); minimizing response time and communication costs (by decreasing the number of using resources), we determine one balanced optimization criterion as to parallelize jobs as far as possible, then to minimize the total system executing time (while the execution time of jobs is included with the communication costs).
- From the feature of Hungarian method that works very efficiently with problem size range 10-100, we propose two variants of optimization:
  - First variant; when M>100 , the jobs are randomly grouped into a slice of N jobs. Then the local optimization is carried out for N jobs and N resources. The size of the optimization zone n = N.
  - Second variant; when M<100, all the jobs are in a global optimization with K multiple groups of N resources, where $K \in \mathbb{N}|$ and K>[M/N]. The size of the optimization area n = N.

3.3. Algorithm Basis.

With the chosen strategies for scheduling, which are mentioned above, there are three important issues, whch now will be studied in detail:

- Forming the "box".
- Determining the searching zone.
- Building a rule-guide for the objective-oriented search.

A. Box Forming.

It means the way to represent the data for scheduling. For the initial data, we have: a set of M independent jobs with the data about their heterogeneity and communication requirements; a graph of N resource-nodes with their heterogeneity and communication links.

Now, for forming the box, we have to reform this separate initial data into a form that represents directly the relationship between jobs and resources: a matrix of size M*N and that we will call Job-Resource matrix (JR). Each element JR[i,j] of this matrix is the weight of the assignment of a job *i* onto a resource *j.* These weights are determined by an optimization function $\Delta(i,j)$. The formation of this function is based on the optimization criteria.

Suppose that the jobs and the resources are identified respectively by two sets $V_J=\{J_1, J_2, ..., J_M\}$, $V_R=\{R_1, R_2, ..., R_N\}$. In general, the optimization function can be determined as the following:

$$\Delta(J_i, R_j) = \delta_{i,j} = \prod_{k=1}^{K} P_k^{i,j} \times \prod_{x=1}^{H} C_x^{i,j} \times \sum_{y=1}^{G} L_y O_y^{i,j} \qquad (1)$$

Where,

- $\prod_{k=1}^{K} P_k^{i,j}$ is called absolute *priority* of assignment (J_i,R_j). It is calculated by multiplication all Kp relative priorities $P_k^{i,j} \in \Re^+$ which are derived from different factors (e.g. permitted waiting time of the job, desirable frequency for usage of the resource). Suppose that $p^d \le P_k^{i,j} \le p^u$, for ∀ k=1..Kp. Then we have: $P^d \le \prod_{k=1}^{K} P_k^{i,j} \le P^u$.
- $\prod_{x=1}^{H} C_x^{i,j}$ is the final result of analyzing all H *processing requirements*, $C_x^{i,j}$ is the degree of satisfying the processing requirement *x* for the assignment (J_i,R_j), $C_x^{i,j} \in \{0,1\}$.(e.g. the requirement about memory, interface environment), $C_x^{i,j}$ =1 if the resource R_j is satisfied by a requirement *x* of the job J_i , otherwise, $C_x^{i,j}$ =0.
- $\sum_{y=1}^{G} L_y O_y^{i,j}$ is the final result of analyzing all G *optimization requirements*, where $O_y^{i,j} \in \Re^+$ and $O^d \le O_y^{i,j} \le O^u$ is the degree of satisfying the optimization requirement *y* for the assignment (J_i,R_j),; $L_y \in \Re^+$ and $L^d \le L_y \le L^u$ is the weight ratio for the optimization criterion *y*. (Figure 2)

RESOURCES

		1	2	3	4	5	6
	1	**8**	**5**	**71**	**7**	**6**	**9**
J	2	**6**	**3**	**9**	**1**	**47**	**7**
O	3	**5**	**59**	**3**	**6**	**8**	**8**
B	4	**9**	**1**	**8**	**95**	**5**	**4**
S	5	**1**	**2**	**1**	**80**	**2**	**3**
	6	**6 9**	**70**	**3**	**5**	**3**	**10**

Figure 2. BOX with Matrix JR (n=6; δ_{max}.=100; ψ_{exist}=30;)

Therefore, finally, for all elements of the matrix JR[i,j]= $\psi_{i,j}$, i=1..M, j=1..N, we have : $\mathbf{0} \le \psi_{i,j} \le \delta_{max}$ and the value $\psi_{exist} = \delta_{max} - \delta_{exist}$ as the threshold for determining if the assignment(Ji,Rj) is possible or not. The size of the box *n* is determined as in section 2 (based on the value of M and N) and according to the optimization area

B. Determining searching zone.

After determining job-resource matrix JR as the "box", the next important step is to determine the zone for searching the real optimum variant of possible schedules.For further study we provide some definitions about the assignments and the schedules as the follows:

Definition 1: There are two given sets: $V_J=\{J_1, J_2, ..., J_n\}$ and $V_R=\{R_1, R_2, ..., R_n\}$. The pair $a=(J_i,R_j)$ is called *Assignment* of the job $J_i \in V_J$ onto the resource $R_j \in V_R$. Each assignment has its *weight* that is determined as $\Psi(a)=\Psi(Ji,Rj)$.

Definition 2: For two given sets: $V_J=\{J_1, J_2, ..., J_n\}$ and $V_R=\{R_1, R_2, ..., R_n\}$, a set $A=\{a_1,$

a_2, ..., a_{n^*}}={(R^1, J^1), (R^2, J^2), ..., (R^{n^*}, J^{n^*})} is called *Maximum matching* for such jobs and resources if:

- $\forall$ i=1.. n*, $\Psi(a_i)=\Psi(J^i, R^i) < \psi_{exist}$.
- $\forall$ i=1..n*, $R^i \notin AR \setminus R^i$, $J^i \in AJ \setminus J^i$, where AR={R^1, R^2, ..., R^{n^*}}, AJ={J^1, J^2, ..., J^{n^*}}.
- n* is maximized as far as it can be.

Definition 3: A maximum matching A* with the size n* for two given sets: V_J={J_1, J_2,.., J_n} and V_R={R_1, R_2, ..., R_n}is a possible *Schedule* for such jobs and resources if n* = n. The weight of the schedule then is determined by: $D(A^*) = \sum_{i=1}^{n} \Psi(a_i^*)$.

Definition 4: Suppose that Xp={A_1 , A_2 , ..., A_p }, $p \in N|$ is the set of all possible schedules. A schedule A* is the *real optimum variant* of schedules if :

$$D(A^*) = \sum_{i=1}^{n} \Psi(a_i^*) = min\{D(A_1), D(A_2), \ldots, D(A_p)\} = \min_{j=1}^{p}\{D(A_j)\}$$

According to the Hungarian method, the searching zone SZ is limited by so-called minimum assignments a*=(J*,R*) which have the minimum weights Ψ(J*,R*) in comparison with other assignments in the same rows or in the same columns of the matrix JR with size n. These minimum assignments create the bounder called B0 for SZ which can be found in the following way:

For i=1.. n, j=1..n, if :

- ($\Psi(a^*)=\Psi(J^*,R^*) < \psi_{exist}$) and
- ($\Psi(a^*)=\min_{i=1}^{n}\Psi(J_i,R_j)$ or $\Psi(a^*)=\min_{j=1}^{n}\Psi(J_i,R_j)$)

then a*= (J*,R*) $\in$ B0.

Therefore, the bounder B0 of the searching zone SZ is determined by a set of minimum assignments A_b = {a_1*,a_2* ,..a_{b0}* }.

In the Hungarian method, the bounder is marked by making the minimum assignments A_{b0} on it become the so-called zero assignments. It is obvious that if we subtract the same number S from all elements of a row (or of a column) in the matrix JR then the minimum assignments still remain as the minimum ones. In other words, if we subtract the weight of the minimum assignment (S) from all elements, for each row and each column, we will still have the same bounder B0 with the same minimum assignments A_b = {a_1*,a_2* ,..a_{b0}* }. The only difference is that these assignments will have zero weights.

In summery, after all, the found bounder B0 of the searching zone is a set of zero assignments A_b = {a_1*,a_2* ,..a_{b0}* }.

C. Objective-Oriented Search.

After determining the bounder of the searching zone, we have to carry out the last and the most important step - to search the optimum variant of the schedules in the determined searching zone. The search starts from the found bounder B0, which is marked by zero assignments and goes to the so-called current “Searching Line“ (SL) by the following steps:

(1) First, check if the optimum variant Vop is on the bounder B0 or not. That means to check if there is a schedule from the minimum assignments A_{b0}={a_1*,a_2*,..a_{b0}*}on B0. If so, go to (2). If not, go to (3).

(2) If any schedule is found, it is the real optimum variant Vop. According to definition 3, we have a schedule on SL if there are such n minimum assignments A*={a_1*,a_2* ,..a_n* } $\subseteq$ A_{b0} that A* is the maximum matching for the given jobs and the given resources. Suppose that for two given sets: V_J={J_1, J_2, ..., J_n} and V_R = {R_1, R_2, ..., R_n}, we have A*={(R^1, J^1), (R^2, J^2), ..., (R^n, J^n)} and AR={R^1, R^2, ..., R^n}, AJ={J^1, J^2, ..., J^n}. By definition 2 this means:

- $\forall$ i=1.. n, $\Psi(a_i^*)=\Psi(J^i, R^i) < \psi_{exist}$.
- $\forall$ i=1..n, $R^i \notin AR \setminus R^i$, $J^i \notin AJ \setminus J^i$.

After that, the real optimum variant Vop is found. Therefore the search has to be stopped.

(3) If we can not find any schedule from the first-minimum assignments $A_{b0}=\{a_1^*,a_2^*,..,a_{b0}^*\}$, this means there is no real optimum variant Vop on SL. Therefore, the search must be continued in the searching zone.The new SL inside the search zone SZ is found from CB by the procedure of Making New Zeros by the Hungarian method [10] as the set of second-minimum assignments (Figure 5.b.). Then go to (1) and repeat the same procedures for the new SL as for the current ones.

		RESOURCES					
		1	2	3	4	5	6
	1	**3**	**0**		**2**	**6**	**2**
J	2	**5**	**2**	**8**	**0**		**4**
O	3	**2**		**0**	**3**	**8**	**3**
B	4	**8**	**0**	**7**		**5**	**1**
S	5	**0**	**1**	**0**		**2**	**0**
	6			**1**	**3**	**0**	**6**

Figure 3.a. B0 of "0"s in Matrix JR (n=6)

		RESOURCES					
		1	2	3	4	5	6
	1	**2**	**0**		**1**	**0**	**1**
J	2	**5**	**3**	**8**	**0**		**4**
O	3	**2**		**0**	**3**	**5**	**3**
B	4	**7**	**0**	**6**		**3**	**0**
S	5	**0**	**2**	**0**		**1**	**0**
	6			**0**	**2**	**0**	**6**

Figure 3.b. SL of "0"s in Matrix JR (n=6)

Note that there are two conditions for finding Vop: First, there is a set of the minimum assignments; Second, there is a schedule (a maximum matching) in this set. In the steps described above, the search is continued not randomly inside the whole area of SZ but only with the assignments on the line SL which characterizes the primary condition for having Vop. Therefore, *the search is oriented to Vop* all the time, unlike random search in simulated annealing method. And that is why we call the algorithm objective-oriented (OOA).

4. Analysis of Simulation Results.

Besides the theoretical analysis, the practical analysis that is based on simulation results is another way to show the advantage of the new algorithm and to examine its the performance in comparison with the existing ones.

An algorithm with random scheduling and an algorithm using SA technique [7] are executed together with OOA for comparing their performance. The comparison criteria are: (i) the solving times (the time for finding the schedule) and (ii) solution quality of received schedules (the time for executing M jobs with N resources by the found schedule in the given system).

Briefly speaking, the input data is formed of the so-called basic data sets (BDS) of M jobs and N resources. There are two factors that have an influence on solving time and solution quality:

- The size *n* of the system which is characterized by M and N.
- The heterogeneity H_{et} of the system which is characterized by how different the elements in matrix JR can be. H_{et} is the ratio (in%) of the number of the different elements in JR.

To illustrate the efficiency of OOA, we use 50 random BDS, where the number of jobs $M \in [20,120]$; the number of resources $N \in [3,30]$.

Firstly, the problem size n is changed from 3 to 30. For each given value n, we investigate solving times and solution quality of Random, SA, and OOA. The simulation results are shown in Figure 7.a and 7.b. They show us that the quality of received schedule by OOA is much better than the ones by SA and, off course, by Random scheduling (Figure 4.a.). Although the solving time of OOA is more than the

one of Random scheduling, it is much less than the solving time of SA (Figure 4.b.). We will have the similar graphics if the problem size is increasing. We do not provide the graphics of comparing with larger size than 30 because of the great solving time of SA. Indeed, the larger the problem size (n) is, the better OOA is, in both scheduling issues: solution quality and solving time..

Secondly, when the problem size is fixed, in our case n = 30, while the heterogeneity is changed by $H_{et} \in [10\%,90\%]$, we study solving times and solution quality of Random, SA, and OOA. The simulation results show us that while solving time of all algorithm (Random,SA,OOA) do not depend on system heterogeneity. However, the heterogeneity of

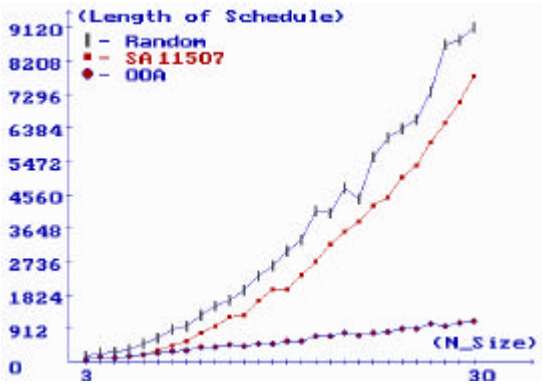

Figure 4.a. Investigation of Solution Quality.

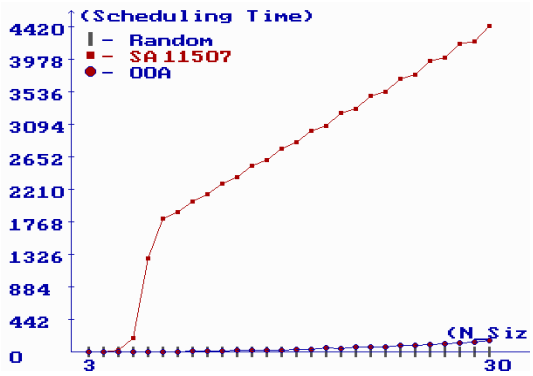

Figure 4.b. Investigation of Solving Time.

the systems has an influence on solution quality.

5. Conclusions.

We have presented a new approach to solve the problem of job scheduling in heterogeneous systems. The new approach using the Hungarian method provides a quick and objective-oriented search for the best schedule by two optimization goals: (i) minimum total execution time including communication costs and (ii) shortest response time for each job. In addition, using the modified algorithm (which has been provided in [12]) for this method, the solving time is decreased to O(n(E+nlogn) The explanation of this fact is that the modified algorithm [12] which is applied for searching Vop on current SL (in 4.3.) is very good fit for such kind of searching.

The advantages of the proposed algorithm OOA are: (i) achieving a good balancing of several conflicting optimization goals: minimum execution time, minimum communication costs, short response time; (ii) scheduling for a relatively short time; (iii) flexibility in application because scheduling is carried out in a general computing model, where there is no requirement for the system architecture; and finally, (iv) especially good for so-called “Strictly-Heterogeneous systems, where either the jobs and the resources can be very different, even unrelated.

For future work, there are two ideas for decreasing the solving time and increasing the solution quality of the proposed algorithm OOA: (i) combining OOA with SA technique for creating a new algorithm having less solving time while controlling the solution quality so that it is acceptable; (ii) simplifying the local optimization in current OOA and adding a simple optimization function for global optimization (by considering workload balance in the global scale).

1. T.L.Casavant and J.G. Kuhl, “A taxonomy of scheduling in general-purpose distributed computing systems”, *IEEE Trans. Softw.Eng.14* (1988)pp141-154.
2. Reinhard Riedl and Lutz Richter, “Classification of Load Distribution Algorithms” 1066-6192/96 *IEEE, Proceeding of PDP’96* (1996)404-413.
3. J.Blazevicz, M. Drozdowski, G. Schmidt, and D. De Werra, “Scheduling independent multiprocessor tasks on a uniform k-processor system”, *Parallel Computer* 20(1994),pp15-28.
4. Pham H. Hanh and V. Simonenko,“A new algorithm and simulation for task assignment in parallel distributed systems”, *Proc. of ESM’96*, Hungary,Budapest, (1996), pp95-99.
5. M.R. Garey and D.S. Johnson, Computer and Intractability-A guide to the Theory of NP- completeness, Freeman New York (1979).
6. A. A. Elsadek and B.E. Wells, “Heuristic model for task allocation in a heterogeneous distributed systems”, *proc. of PDPTA’96*, Vol.2, (1996), pp 659-671.
7. Kirkpatrick, s., Gelatt, C.D., Vecchi, M.P., “Optimization by simulated annealing”, *Science*, Vol.220, N.4589, 13 May (1983).
8. P.Shroff, D.W,Watson, N.F. Flann, and R.F.Freund, “Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments”, *Heterogeneous Computing Workshop ’96*,(1996), pp.98-104.
9. Richard F.Freund, B.R.Carter, Daniel Watson, E. Keith, and F. Mirable, ”Generational Scheduling for Heterogeneous Computing Systems”, *proc. of PDPTA’96*, Vol.2, (1996), pp 769-778.
10. Valery Simonenko and Pham Hong Hanh, “Adaptation of algorithm for job- resource assignment in heterogeneous distributed systems”, *proc. of PDPTA "96*, California USA, Vol.2, (1996), pp 835-845.
11. Efe K., “Heuristic models for task assignment scheduling in distributed systems”, *IEEE Computer*, June (1982)
12. El-Rewini H., Lewis T.G., “Scheduling Parallel tasks onto Arbitrary Target Machines”, *Journal of Par. and Distr. Com.*,Vol.9,(1990), pp 138-153.
13. Bultan T. and Aykanat C. , “A new mapping heuristic based on mean field annealing”, *Journal of Parallel and Distributed Computing*, Vol. 16, n4, Dec 1992.
14. Berge C. Theorie des graphes et ses applications. Dunod, Paris (1958).
15. Shen Shen Wu and David Sweeting, “Heuristic algorithms for task assignment and scheduling in a processor network”, *Parallel Computing* 20 (1994) 1-14.