Research Article

Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Reassignment

Pradeep Kumar Yadav,¹ M. P. Singh,² and Harendra Kumar²

¹ Department of Research Planning and Business Development, Central Building Research Institute, Roorkee 247667, Uttarakhand, India

² Department of Mathematics and Statistics, Gurkula Kangari University, Harwar 249404, Uttarakhand, India

Correspondence should be addressed to Harendra Kumar, kumar_balyan@rediffmail.com

Received 5 September 2007; Revised 17 January 2008; Accepted 19 March 2008

Recommended by Y. -M. Huang

Distributed computing systems [DCSs] offer the potential for improved performance and resource sharing. To make the best use of the computational power available, it is essential to assign the tasks dynamically to that processor whose characteristics are most appropriate for the execution of the tasks in distributed processing system. We have developed a mathematical model for allocating “M” tasks of distributed program to “N” multiple processors ($M > N$) that minimizes the total cost of the program. Relocating the tasks from one processor to another at certain points during the course of execution of the program that contributes to the total cost of the running program has been taken into account. Phasewise execution cost [EC], intertask communication cost [ITCT], residence cost [RC] of each task on different processors, and relocation cost [REC] for each task have been considered while preparing a dynamic tasks allocation model. The present model is suitable for arbitrary number of phases and processors with random program structure.

Copyright © 2008 Pradeep Kumar Yadav et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Over the last several years, distributed computing systems have become popular as a very attractive option for fast computing and information processing. Distributed computing system is used to describe a system whenever there are several computers interconnected in some fashion so that a program or procedure runs on the system with multiple processors. In a heterogeneous computing system, the tasks of an application program must be assigned to the heterogeneous processors to utilize the computational capabilities and resources of the system efficiently. However, it is a difficult problem to compute an assignment of tasks to processors for which the total cost of running the program is minimized. Static and dynamic algorithms can be extremely efficient in homogeneous systems. If all processors are identical then the choice “to which node should we assign the next task?” falls on the processor with the fewest outstanding jobs [1]. However in heterogeneous distributed environments, where not all nodes have the same parameters, selecting the optimal target processor requires more elaborate algorithms.

Stone [2] has suggested an efficient optimal algorithm for the problem of assigning tasks to two processors by making use of the well-known network flow algorithm in the related two-terminal network graphs. If the structure of a distributed program is constrained in certain ways, it is possible to find the optimal assignment over a system made up of any number of processors in polynomial time. When the structure is constrained to be a tree, the shortest tree algorithm developed by Bokhari [3] yields the optimal assignment. Towsley [4] generalized the results of Bokhari to the case of series-parallel structures. A further generalized program structure is a partial K-tree in which trees are partial 1-tree and series-parallel structures are partial 2-tree [5] assuming distributed system as fully interconnected, that is, there exists a communication link between any two machines. However, the general N -processors problem ($N > 3$) in a fully interconnected system is known to be NP-complete [6, 7]. Lee et al. [8] and Cho and Park [9] have suggested an optimal algorithm for the general structure problem in a linear array network with any number of processors.

All the above assignment models are static because they assume that once a task is assigned to a processor, it remains there statically during the execution of a distributed program. These models deal with the global information, which is assumed to be constant during the program execution, of program's activity. However the static model assumes that the problem can be partitioned into several tasks each of which contains a unique computational characteristic. In a general case each task may have multiple computational characteristics due to the dynamic control behavior of the task. Such tightly and medium-coupled portions of code task will be more difficult to break up and assign to different processors. When the characteristics of the computation change, a new assignment must be computed. By "characteristics of the computation" we mean the ratios of the times that the program spends in different parts of the program. In order to make the best use of computational characteristics and resources in a distributed heterogeneous computing system, it is essential to reassign tasks dynamically during program execution, so as to take advantage of changes in the local reference patterns of the program. Dynamic algorithms are employed more frequently because they offer a better efficiency, having the flexibility of tasks migration. In addition, usually dynamic algorithms are neither very complicated to construct nor costly to run. Hence dynamic algorithms have the potential to outperform static algorithms in general. Clearly, the gains from relocating tasks must outweigh the cost of relocation. For this relocation, detailed information on a program's activity over its lifetime can be collected at each time slot of lifetime. We use phase as the unit of the time slot for gathering more detailed information of program activity. Bokhari [10] analyzed the problem of dynamic assignment in a two-processor system, which permits relocation of tasks from one processor to the other at certain points during the execution of the program. This relocation incurs a predefined relocation cost that contributes to the total cost of running the program and code-conversion overheads. Also it is shown that an optimal dynamic assignment corresponds to a "dynamic" partition imposed by a minimum weight cut in an extended graph. In this paper, we consider the dynamic task assignment problem for a general program structure and a heterogeneous N -processors in DCS.

The rest of the paper is organized as follows. Definitions and assumptions are defined first in Section 2. In Section 3, tasks allocation problem has been discussed. The proposed method and algorithm have been explained in Sections 4 and 5 concludes this paper.

2. DEFINITIONS AND ASSUMPTIONS

2.1. Definitions

- (i) Phase: a phase of a distributed program is a contiguous period of time during which only one task is executed. During this period, the executing task may communicate with any number of the remaining task. A task may not be moved from one processor to another during a phase; it may be moved only between

them. A task may complete its execution in more than two phases; therefore, the number of phases of the executing program is always more than the number of the tasks.

- (ii) Execution cost: the execution cost e_{kij} is the amount of the work to be performed by the executing task t_i on the processor p_i during the k th phase.
- (iii) Intertask communication cost: the intertask communication cost c_{kij} is incurred due to the data units exchanged between the executing task t_i and residing task t_j if they are on different processors during the process of execution in k th phase.
- (iv) Residence cost: residence cost r_{kij} (where $1 \leq i \leq M$ & $1 \leq j \leq N$) is the amount of time spent by the residing task (other than executing task) t_i on the processor p_i in k th phase. These costs may come from the use of storage.
- (v) Relocation cost: relocation cost rel_{ik} is the cost of reassigning the i th task from one processor to another processor at the end of k th phase.

2.2. Assumptions

- (i) The program to be executed is assumed to be collection of tasks that are free, in general, to reside on either of the processors. A task may be a module of executable program.
- (ii) The given program runs in various phases with one of its task executing on a processor in each phase and each of the other tasks residing on any of the processors.
- (iii) In each phase, execution cost (EC), residence cost (RC) of nonexecuting tasks, intertask communication cost (ITCC) of executing task with nonexecuting tasks, and relocation cost (REC) for reassigning each task from one processor to the others at the end of phases are known.
- (iv) Some of the tasks have restrictions for being allocated on some particular processor because of the nonavailability of desired facilities which may include access to particular peripheral device, access to a particular task, and high speed arithmetic capability of the processor. The execution costs of such tasks are taken to be infinite on those processors where these tasks can not be executed.

3. PROBLEM STATEMENT

Consider a distributed program consisting of a set $T = \{t_1, t_2, t_3, \dots, t_M\}$ of M tasks to be allocated to a set $P = \{p_1, p_2, p_3, \dots, p_N\}$ of N processors divided into K phases. The basis for dynamic program model is the concept of the phases of a task program. With each phase the following information is associated:

- (1) the executing task during this phase and its execution cost on each processor in a heterogeneous system;
- (2) residence costs of the remaining tasks, except for the executing task, on each processor. These costs may come from the use of storages;

Step-1: Input: $K, M, N, ECM(,), RCM(,), RECM(,), ITCCM(,)$
 Step-2: Start with phase $k = 1$.
 Step-3: Calculate $NECM(,)$ by adding the $ECM(,)$ with $RCM(,)$ for k th phase.
 Step-4: Assign the executing task t_s to processor p_r for which cost is minimum say TEC_k
 Step-5: Add the effect of communication cost of executing tasks t_s with other residing tasks by adding k th column of $ITCCM(,)$ to all the column of $NECM(,)$ except the r th column at which this task is assigned.
 Step-6: Assign the residing tasks t_g ($g = 1, 2, \dots, M, g \neq s$) to processors p_h at which time is minimum say $TRC(g)_k$
 Calculate:

$$TRC_k = \sum_{\substack{i=1 \\ i \neq s}}^M TRC(i)_k$$

 Step-7: Store executing and residing tasks and their position on processors in a linear array $POSS(,)_k$.
 Calculate:

$$(TCOST)_k = TEC_k + TRC_k$$

 Step-8: $k = k + 1$.
 Step-9: Repeat step-3.
 Step-10: Modify the cost of $NECM(,)$ by adding the relocation cost rel_{sk-1} to all e_{ksj} ,
 $j = 1, 2, \dots, N, j \neq r$.
 Step-11: Modify the cost of $NECM(,)$ by adding the relocation cost rel_{gk-1} to e_{kgj} ,
 $j = 1, 2, \dots, N, j \neq h$.
 Step-12: Repeat step-4 to step-6.
 Step-13: Store executing and residing tasks and their position on processors in linear array $POSS(,)_{k+1}$.
 Calculate:

$$(TCOST)_{k+1} = TEC_{k+1} + TRC_{k+1}$$

 Step-14: If $k \neq K$
 then
 repeat step-8 to step-14.
 else
 Step-15: Calculate:

$$TCOST = \sum_{k=1}^K (TCOST)_k$$

 Step-16: End.

ALGORITHM 1: To assign a distributed program consisting of a set $T = \{t_1, t_2, t_3, \dots, t_M\}$ of M tasks which is divided into K phases to a set of n processors $P = \{p_1, p_2, p_3, \dots, p_N\}$ in order to minimize the total cost.

- (3) intertask communication cost between the executing task and all other tasks if they are on different processors;
- (4) an amount of relocation cost for reassigning each task from one processor to the others at the end of the phases.

In general, the objective of task assignment is to minimize the completion cost of a distributed program by properly mapping the tasks to the processors. The cost of an assignment A , $TCOST(A)$, is the sum of execution, intertask communication, residence, and relocation costs, that is,

$$TCOST(A) = EC(A) + RC(A) + ITCC(A). \quad (1)$$

In order to make the best use of the resources in a DCS, we would like to find an assignment having minimum cost.

4. PROPOSED METHOD AND ALGORITHM

The problem for allocating a program dynamically that has been divided into K number of phases is to determine the optimal task allocation strategy by considering the phasewise EC for executing task and RC for nonexecuting tasks during the phases. The phasewise processing efficiency of individual processor is given in the form of matrix $EMC(,) = (e_{kij})$ of order $(K \times M \times N)$ and RC of each residing task is reflected through matrix $RCM(,) = (r_{kij})$ of order $(K \times M \times N)$. ITCC between executing task and another tasks is taken in the form of matrix $ITCCM(,) = (c_{kij})$ of order $(K \times M)$. The k th column of this matrix represents the ITCC between the executing task and all other residing tasks for the k th phase. Relocation costs for reassigning the tasks from one processor to another at the end of the phases is shown by matrix $RECM(,) = rel_{ik}$ of order $(M \times K)$ where rel_{ik} is the relocation cost for reassigning the i th task from one processor

TABLE 1: Phasewise ECM(,) and RCM(,).

Phases	Tasks	Execution cost ECM(,)		Residence cost RCM(,)	
		p_1	p_2	p_1	p_2
1	t_1	2	6	—	—
	t_2	—	—	3	2
	t_3	—	—	2	2
	t_4	—	—	4	1
2	t_1	—	—	2	1
	t_2	∞	4	—	—
	t_3	—	—	1	3
	t_4	—	—	2	3
3	t_1	—	—	2	1
	t_2	—	—	1	2
	t_3	4	6	—	—
	t_4	—	—	2	1
4	t_1	—	—	1	3
	t_2	—	—	3	2
	t_3	—	—	1	2
	t_4	3	2	—	—
5	t_1	2	∞	—	—
	t_2	—	—	3	1
	t_3	—	—	2	1
	t_4	—	—	1	2

to the another at the end of k th phase. In this paper, we have considered that the relocation cost rel_{ik} is a constant when i th task relocates from one processor to another at the end of the k th phase. The model starts from the first phase for allocating the tasks and moves to the next phase. Let t_s be the s th executing task and t_g ($s \neq g$) be the g th residing tasks during k th phase. The execution for the executing task t_s and residing task t_g to the processors in the k th phase have been depicted in Figure 1.

The executing task t_s (represented by *) must be assigned to processor p_r during k th phase for which sum of its execution and relocation costs is minimum. This minimum cost for the task t_s is represented by TEC_k . When residing task t_g resides on a processor, it has three-type costs, that is. residing, relocation, and communication cost with the executing task t_s . Therefore, the residing task t_g must be assigned to that processor p_h where it has minimum costs mentioned above. This minimum cost for the task t_g is represented by $TRC(g)_k$. Similarly, all other residing tasks are assigned to the processors. The total cost TRC_k due to the residing tasks in the k th phase will be:

$$TRC_k = \sum_{\substack{i=1 \\ i \neq s}}^M TRC(i)_k. \quad (2)$$

The total cost $(TCOST)_k$ for the k th phase is the sum of TEC_k and TRC_k , that is,

$$(TCOST)_k = TEC_k + TRC_k. \quad (3)$$

TABLE 2: ITCCM(,) and RECM(,).

Phasewise communication cost between executing task and other residing tasks ITCCM(,)					Relocation cost of the tasks at the end of phase RECM(,)					
Phases										
	1	2	3	4	5	1	2	3	4	5
Tasks										
t_1	—	6	2	1	—	2	3	3	2	—
t_2	2	—	4	2	0	2	3	2	3	—
t_3	3	2	—	2	2	1	2	2	4	—
t_4	2	4	0	—	3	2	4	2	2	

The total optimal cost, TCOST, for the program will be as follows:

$$TCOST = \sum_{k=1}^K (TCOST)_k. \quad (4)$$

Using the arrays representation for the execution, residing, intertask communication, and relocation costs, a mathematical model has been developed for allocating the distributed program dynamically which has been splitting into any number of tasks partition and completing their execution in any number of phases. The process of allocating the tasks to the processors is given in the form of Algorithm 1.

Implementation of the Model

Example 1. Consider a program made up of four tasks $\{t_1, t_2, t_3, t_4\}$ to be allocated on two processors $\{p_1, p_2\}$ in five phases. Task t_1 completes its execution in two phases (in phase 1 and phase 5). The various matrices to be taken as input are described phasewise in Tables 1 and 2.

On applying the present tasks allocation model, we get the following optimal assignment as follows:

$$\begin{aligned} POSS(,)_1 &= \{(t_1, p_1), (t_2, p_1), (t_3, p_1), (t_4, p_2)\}, \\ TCOST_1 &= TEC_1 + TRC_1 = 10. \\ POSS(,)_2 &= \{(t_1, p_2), (t_2, p_1), (t_3, p_1), (t_4, p_2)\}, \\ TCOST_2 &= TEC_2 + TRC_2 = 15. \\ POSS(,)_3 &= \{(t_1, p_2), (t_2, p_1), (t_3, p_1), (t_4, p_2)\}, \\ TCOST_3 &= TEC_3 + TRC_3 = 12. \\ POSS(,)_4 &= \{(t_1, p_2), (t_2, p_2), (t_3, p_1), (t_4, p_2)\}, \\ TCOST_4 &= TEC_4 + TRC_4 = 12. \\ POSS(,)_5 &= \{(t_1, p_1), (t_2, p_2), (t_3, p_1), (t_4, p_1)\}, \\ TCOST_5 &= TEC_5 + TRC_5 = 10. \end{aligned} \quad (5)$$

$$TCOST = \sum_{k=1}^K (TCOST)_k = 59.$$

TABLE 3: Phasewise ECM($\cdot$, $\cdot$) and RCM($\cdot$, $\cdot$).

Phases	Tasks	Execution costs ECM($\cdot$, $\cdot$)			Residence costs RCM($\cdot$, $\cdot$)		
		p_1	p_2	p_3	p_1	p_2	p_3
1	t_1	—	—	—	3	1	2
	t_2	2	6	4	—	—	—
	t_3	—	—	—	3	4	2
	t_4	—	—	—	4	3	2
	t_5	—	—	—	4	3	2
	t_6	—	—	—	2	1	1
	t_7	—	—	—	3	1	2
2	t_1	6	4	8	—	—	—
	t_2	—	—	—	2	1	3
	t_3	—	—	—	3	2	2
	t_4	—	—	—	3	4	2
	t_5	—	—	—	3	3	2
	t_6	—	—	—	1	4	4
	t_7	—	—	—	2	3	2
3	t_1	—	—	—	2	3	1
	t_2	—	—	—	2	2	2
	t_3	—	—	—	3	4	2
	t_4	6	3	∞	—	—	—
	t_5	—	—	—	3	2	1
	t_6	—	—	—	3	2	2
	t_7	—	—	—	3	1	1
4	t_1	—	—	—	2	1	2
	t_2	—	—	—	1	1	2
	t_3	—	—	—	3	3	1
	t_4	—	—	—	1	2	4
	t_5	—	—	—	2	2	1
	t_6	8	6	4	—	—	—
	t_7	—	—	—	1	1	3
5	t_1	—	—	—	2	1	1
	t_2	—	—	—	2	1	2
	t_3	6	6	8	—	—	—
	t_4	—	—	—	2	2	2
	t_5	—	—	—	4	3	2
	t_6	—	—	—	1	2	3
	t_7	—	—	—	2	1	2
6	t_1	—	—	—	3	2	2
	t_2	—	—	—	1	4	1
	t_3	—	—	—	1	2	1
	t_4	—	—	—	2	1	1
	t_5	5	4	2	—	—	—
	t_6	—	—	—	2	2	1
	t_7	—	—	—	1	1	2
7	t_1	—	—	—	1	2	1
	t_2	∞	6	8	—	—	—
	t_3	—	—	—	2	3	1
	t_4	—	—	—	2	3	1
	t_5	—	—	—	1	2	2
	t_6	—	—	—	1	3	2
	t_7	—	—	—	2	2	1

TABLE 3: Continued.

Phases	Tasks	Execution costs ECM(, ,)			Residence costs RCM(, ,)		
		p_1	p_2	p_3	p_1	p_2	p_3
8	t_1	—	—	—	2	1	1
	t_2	—	—	—	1	1	2
	t_3	—	—	—	2	4	2
	t_4	—	—	—	2	3	3
	t_5	—	—	—	1	1	4
	t_6	—	—	—	3	2	2
	t_7	4	7	8	—	—	—

TABLE 4: ITCCM(, ,) and RECM(, ,).

Phases	Phasewise communication cost of executing task with other residing tasks ITCCM(, ,)								Relocation cost of the tasks at the end of phases RECM(, ,)							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Tasks																
t_1	2	—	3	2	4	2	4	0	3	3	4	1	3	2	3	—
t_2	—	0	2	2	2	0	—	3	2	1	5	3	3	3	4	—
t_3	4	3	0	4	—	2	2	2	1	1	3	2	3	4	4	—
t_4	2	2	—	2	1	3	2	3	1	4	3	2	4	3	2	—
t_5	2	1	0	2	2	—	0	2	2	3	2	2	2	4	2	—
t_6	6	2	2	—	2	0	2	2	2	3	4	3	2	4	3	—
t_7	2	3	3	4	2	0	3	—	3	4	4	3	3	4	2	—

The result of Example 1 is shown by Figure 2. Here circles labeled with tasks number represent tasks (tasks represented by * are executive task and others are residing tasks). Costs associated with tasks are their execution costs (for tasks represented by *) or residence costs. The edges connecting the executing task with other residing tasks during the same phase represent the communication and weights of these edges represent communication costs between the executing task and other residing tasks. The edges connecting the task of a phase to the task of the next phase represent the relocation of the tasks and the weights of these edges represent the costs of relocating the tasks. For example, the task t_2 is relocating from processor p_2 to processor p_1 between the phase 2 and phase 3. The above result shows that the assignments of the tasks are changing phase to phase. Optimal cost for this task program is 59.

Example 2. Consider a program made up of seven tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ to be executed on three processors $\{p_1, p_2, p_3\}$ in eight phases. The various matrices to be taken as input are described phasewise in Tables 3 and 4.

The present tasks allocation model gives the following optimal assignment as follows:

$$\begin{aligned}
& \text{POSS}_1 = \{X\}, \\
& \text{TCOST}_1 = \text{TEC}_1 + \text{TRC}_1 = 21, \\
& \text{POSS}_2 = \{B\}, \\
& \text{TCOST}_2 = \text{TEC}_2 + \text{TRC}_2 = 22, \\
& \text{POSS}_3 = \{C\}, \\
& \text{TCOST}_3 = \text{TEC}_3 + \text{TRC}_3 = 22, \\
& \text{POSS}_4 = \{D\}, \\
& \text{TCOST}_4 = \text{TEC}_4 + \text{TRC}_4 = 31, \\
& \text{POSS}_5 = \{E\}, \\
& \text{TCOST}_5 = \text{TEC}_5 + \text{TRC}_5 = 20, \\
& \text{POSS}_6 = \{F\}, \\
& \text{TCOST}_6 = \text{TEC}_6 + \text{TRC}_6 = 21, \\
& \text{POSS}_7 = \{G\}, \\
& \text{TCOST}_7 = \text{TEC}_7 + \text{TRC}_7 = 24, \\
& \text{POSS}_8 = \{H\}, \\
& \text{TCOST}_8 = \text{TEC}_8 + \text{TRC}_8 = 24, \\
& \text{TCOST} = \sum_{k=1}^K (\text{TCOST})_k = 185,
\end{aligned} \tag{6}$$

FIGURE 1: Allocation of the tasks in kth phase.

FIGURE 2: Optimal assignment graph (TCOST = 59).

where

$$\begin{aligned}
 X &= (t_1, p_2), (t_2, p_1), (t_3, p_1), (t_4, p_3), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 B &= (t_1, p_2), (t_2, p_1), (t_3, p_2), (t_4, p_3), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 C &= (t_1, p_2), (t_2, p_2), (t_3, p_3), (t_4, p_2), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 D &= (t_1, p_2), (t_2, p_2), (t_3, p_3), (t_4, p_2), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 E &= (t_1, p_2), (t_2, p_2), (t_3, p_2), (t_4, p_2), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 F &= (t_1, p_2), (t_2, p_1), (t_3, p_3), (t_4, p_2), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 G &= (t_1, p_2), (t_2, p_2), (t_3, p_3), (t_4, p_2), (t_5, p_3), (t_6, p_1), (t_7, p_2), \\
 H &= (t_1, p_2), (t_2, p_2), (t_3, p_3), (t_4, p_1), (t_5, p_1), (t_6, p_1), (t_7, p_1).
 \end{aligned}
 \tag{7}$$

The results of Example 2 are shown by Figure 3. Task t_2 completes its execution in two phases (in phase 1 and phase 7). The execution cost for the task t_2 is 2 in phase 1 and 4 in phase 7. The above result for the executing task t_2 shows

that the assignments for both the phases are different, that is; it depends on the characteristic of the computation. The optimal cost for tasks program is 185.

5. CONCLUSION

We have presented an efficient solution technique for tasks execution dynamically in DCS. Starting with the definitions of the phase of a modular program, we have developed a mathematical-based assignment technique for allocating the tasks of a program, divided into k -phases to $N > 2$ processors. The present model is compared with Kumar et al. [11] and Bokhari [10]. Kumar et al. [11] have developed a mathematical model for allocating “ M ” tasks of a program to “ N ” processors ($M > N$), divided into any number of phases. In this paper, authors have considered only execution, residence, and intertask communication costs. It is also observed that the model [11] does not work for the system program

FIGURE 3: Optimal assignment graph (TCOST = 185).

having relocation cost for reassigning each task from one processor to the others at the end of phases. Bokhari [10] has developed a graph theoretical approach for finding out the optimal dynamic assignment of a modular program for the two-processor system considering execution, residence, intertask communication, and relocation costs. The model [10] can not find out the solution when the system is NP-hard ($N > 2$) while the present model is applicable for the DCS having any number of phases and processors with random program structure. The study of the time complexity of the present model has also been made and compared with Bokhari [10] and Cho and Park [9]. For Bokhari [10], the time complexity of the solution of the dynamic task assignment problem for the two-processor system is $O(M^3K^3)$. The time complexity of the present tasks model for any number of processors is $O(M^2N^2K)$ which is time-saving as compared to $O(M^2N^2K^2\log NMK)$ for the Cho and Park [9].

REFERENCES

- [1] S. H. Bokhari, *Assignment Problem in Parallel and Distributed Computing*, Kluwer Academic Publishers, Boston, Mass, USA, 1987.
- [2] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 1, pp. 85–93, 1977.
- [3] S. H. Bokhari, "A shortest tree algorithm for optimal assignments across space and time in a distributed processor system," *IEEE Transactions on Software Engineering*, vol. SE-7, no. 6, pp. 583–589, 1981.
- [4] D. F. Towsley, "Allocating programs containing branches and loops within a multiple processor system," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 10, pp. 1018–1024, 1986.
- [5] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.
- [6] A. Ghafoor and J. Yang, "A distributed heterogeneous supercomputing management system," *Computer*, vol. 26, no. 6, pp. 78–86, 1993.
- [7] M. P. Singh, P. K. Yadav, and H. Kumar, "A tasks allocation model for efficient utilization of processor's capacity in heterogeneous distributed systems," in *Proceeding of the 9th Conference of International Academy of Physical Sciences*, Agra, India, February 2007.
- [8] C.-H. Lee, D. Lee, and M. Kim, "Optimal task assignment in linear array networks," *IEEE Transactions on Computers*, vol. 41, no. 7, pp. 877–880, 1992.
- [9] S.-Y. Cho and K. H. Park, "Dynamic task assignment in heterogeneous linear array networks for metacomputing," in *Proceeding of the Heterogeneous Computing Workshop*, pp. 66–71, Cancun, Mexico, April 1994.
- [10] S. H. Bokhari, "Dual processor scheduling with dynamic reassignment," *IEEE Transactions on Software Engineering*, vol. SE-5, no. 4, pp. 341–349, 1979.

-
- [11] V. Kumar, M. P. Singh, and P. K. Yadav, "An efficient algorithm for multiprocessor scheduling with dynamic reassignment," in *Proceeding of the 6th National Seminar on Theoretical Computer Science*, pp. 105–118, Banasthali, Rajasthan, India, August 1996.