A Heuristic Approach for Optimal Task Allocation for The System Cost Analysis in Distributed Systems

Avanish Kumar[1] and Tariq Ahmad Bhat[2]

[1] Associate Professor and Head & [2]Research Scholar
Deptt. of Mathematical Science and computer Applications
Bundelkhand university, Jhansi-284128(U.P.), India

[1]Email: dravanishkumar@yahoo.com, Ahmad4u11@gmail.com

Abstract: The problem of the task allocation in distributed computing system is to need to allocate a number of tasks to different processors for execution. In this paper we have developed a task allocation model and have proposed a heuristic algorithm for task allocation that will find an optimal solution to the problem. The proposed algorithm try to minimize the inter processor communication cost (IPCC) by assigning those task first, which has the heaviest communication link sum (CLS). Using this approach it has been seen that the system cost will minimize more than other heuristic.

Keywords: Distributed computing systems, Heuristic Approach, Task Allocations, execution costs, Communication costs, communication link sum.

1. INTRODUCTION

A distributed computing system (DCS) consists of a set of multiple processors interconnected by communication links. A very common interesting problem in DCS is the task allocation. This problem deals with finding an optimal allocation of tasks to the processors so that the system cost (i.e. the sum of execution cost and communication cost) is to be minimized without violating any of the system constraints [1]. In DCS, an allocation policy may be either static or dynamic, depending upon the time at which the allocation decisions are made. In a static task allocation, the information regarding the tasks and processor attributes is assumed to be known in advance, before the execution of the tasks [2]. Here we shall be considering static task allocation policy in this paper task allocation problem is known to be NP- hard problem in complexity (The task assignment problem for more than three processors is known to be NP-hard), when we required an optimal solution to this problem. The easiest way to finding an optimal solution to this problem is an exhaustive enumerative approach. But it is impractical, because there are n^o ways for allocating m- tasks to n-processors [1].

Many research efforts on the task allocation problem have been identified in the past with the main concern on the performance measures such as minimizing the total sum of execution and communication costs [1, 3, 4] or minimizing the program turnaround time [5, 6], the maximization of the system reliability [7-14]. Distributed computing system has attracted several researchers by posing several challenging problems. In a DCS, the execution of a program may be distributed among several computing elements to reduce the overall system cost by taking advantage of heterogeneous computational capabilities and other resources within the system.

A large number of techniques to task allocation in DCSs have been reported in [1, 2, 3], [8-14]. They can be broadly classified into three categories: graph theoretic technique [8, 9], integer programming technique [15-16] and heuristic technique [1, 2, 3], [17-18]. Graph theoretic and integer programming techniques yields an optimal solution at all the times. But these techniques are restricted to the small size problems. If the problem size is very large, it is necessary to use the heuristic technique to get near optimal solutions. The choice of a particular technique depends on the structure of the problem [10].

In this paper, we have developed a task allocation model and have proposed a heuristic algorithm for task allocation that will find an optimal solution to the problem. The proposed algorithm try to minimize the inter processor communication cost (IPCC) by assigning those task first, which has the heaviest communication link sum (CLS). Using this approach it has been seen that the system cost will minimize more than other heuristic.

The rest of the paper is organized as follows: section-2 formulates the task allocation problem for minimizing the overall system cost; section-3 discusses in detail the proposed allocation technique and algorithm; section-4 gives an implementation of the proposed algorithm. In the last section-5 concludes the paper.

Notations

T : the set of tasks of a parallel program that are to be executed.
P : the set of processors in DCS.
n : the number of processors.
m : the number of tasks.
t_i : i^{th} task of the given program.
P_k: k^{th} processor in P.
x_{ik}: the decision variable such that x_{ik} =1, if i^{th} task is allocated to k^{th} processor, x_{ik} = 0, otherwise.
ec_{ik} : incurred execution cost (EC), if i^{th} task is executed on k^{th} processor.
cc_{ij} : incurred inter task communication cost between task t_i and t_j, if they are executed on separate processors.
ECM(,) : execution cost matrix.
ITCCM(,) : inter task communication cost matrix.
$T_{ass}\{\}$: a linear array to hold assigned tasks.
$T_{non\text{-}ass}\{\}$: a linear array to hold non assigned tasks.
$T_{CLS}\{\}$: a linear array to hold the task according to their communication link sum.

Assumptions

To allocate the tasks of a parallel program to processors in DCS, we have been made the following assumptions:

- The processors involved in the DCS are heterogeneous and do not have any particular interconnection structure.
- The parallel program is assumed to be the collection of m- tasks that are free in general, which are to be executed on a set of n- processors having different processor attributes.
- Once the tasks are allocated to the processors they reside on those processors until the execution of the program is completed. Whenever a group of tasks is assigned to the processor, the inter task communication cost (ITCC) between them is zero.
- It is also assumed that the number of tasks to be allocated is more than the number of processors (m >> n) as in real life situation.

2. PROBLEM'S DESCRIPTION

In the past, different task allocation models and techniques for minimizing the overall system cost have been widely investigated in the literature. In this paper, we follow [1, 2, 3, 5, 19, 20, 21] to formulate the task allocation problem.

2.1 Problem Statement

The problem being addressed in this paper is concerned with an optimal allocation of the tasks of a parallel application on to the processors in DCS. An optimal allocation is one that minimizes the system cost function subject to the system constraints. In this paper, we have considered a distributed computing system made up by two sets, $P = \{P_1, P_2, \ldots\ldots, P_n\}$ of heterogeneous processors, interconnected by communication links and $T = \{t_1, t_2, \ldots\ldots, t_m\}$ of program tasks, which collectively form a common goal [1].

The execution costs of a task running on different processors are different and it is given in the form of a matrix of order $m \times n$, named as execution cost matrix ECM (,). Similarly, the inter task communication cost between two tasks is given in the form of a symmetric matrix named as inter task communication cost matrix ITCCM (,) of order $m \times m$.

Now, an allocation of tasks to processors can be defined by a function X as follows:

$X: T \rightarrow P$, such that $X(i) = k$; if i^{th} task is allocated to k^{th} processor.

The purpose of defining the above function is to allocate each of the m- tasks to one of the n-processors such that the overall system cost is minimized.

2.2 Definitions

2.2.1 Execution cost (EC)

The execution cost ec_{ik} of a task t_i, running on a processor P_k is the amount of the total cost needed for the execution of t_i on that processor during the execution process. If a task is not executable on a particular processor, the corresponding execution cost is taken to be infinite (∞).

2.2.2 Communication cost (CC)

The communication cost (cc_{ij}); incurred due to the inter task communication is the amount of total cost needed for exchanging data between t_i & t_j residing at separate processor during the execution process. If two tasks executed on the same processor then $cc_{ij} = 0$.

2.2.3 Communication link Sum (CLS)

It is an important characteristic of ITCCM (,), denoted by CLS, which measures how communication intensive a task is. The CLS of a task t_i: $1 \leq i \leq m$, can be easily determined by finding the sum of communication costs of all the tasks which are interacting with t_i in ITCCM (,). Therefore, in inter task communication cost matrix, the communication link sum of a task t_i can be computed as:

$CLS(t_i) = \sum_{j=1}^{m} cc_{ij}$ for i = 1, 2, ……, m. (1)

2.3 Task allocation model for system cost

Here, we have developed a task allocation model to get an optimal system cost. We can achieve this objective by making task allocation properly. Therefore, an efficient task allocation of the program tasks to processor is imperative. However, obtaining an optimal allocation of tasks of a random program to any arbitrary number of processors interconnected with non-uniform links is a very difficult problem [2]. Henceforth, in order to allocate the tasks of such program to processors in DCS, we should know the information about the input such as tasks attributes [e.g. execution cost, inter task communication cost etc] and processor attributes [e.g. processor topology, inter processor distance etc] etc. since obtaining such information is beyond the scope of this paper therefore, a deterministic model that the required information is available before the execution of the program is assumed [22].

2.3.1 Processor Execution Cost (PEC)

For given a task allocation, X: T→P, X(i) = k, the execution cost ec_{ik} represent to execute task t_i on processor P_k and used to control the corresponding processor allocation. Therefore, under a task allocation X, the processor execution cost, needed to execute all the tasks assigned to k^{th} processor can be computed as:

$PEC\ (X)_k = \sum_{i=1}^{m} e\ c_{ik}x_{ik}$ (2)

2.3.2 Inter- Processor Communication Cost (IPCC)

Inter processor communication cost is incurred when the data is transmitted from task to task if they are residing on separate processors, due to the inter task communication. Therefore, inter processor communication cost (IPCC) is proportional to inter task communication cost cc_{ij} [18, 19].

Therefore, under a task allocation X, the inter processor communication cost for k^{th} processor can be computed as:

$IPCC(X)_k = \sum_{i=1}^{m}.\sum_{j>i}^{m}(cc_{ij})\ x_{ik}x_{jb}$ (3)

In this model, both the costs are application dependent and takes play an important role in task allocation. Now, the total cost on k^{th} processor is the sum of the processor execution cost (PEC) and IPCC for k^{th} processor, under a given task allocation X.

$T_{Cost}(x)_k = PEC(X)_k + IPCC(X)_k$ (4)

and the total cost of the system is computed by:

$S_{Cost}(X) = \sum_{k=1}^{n} T_{Cost}\ (X)_k$ (5)

2.3.3 System cost model

With system resources constraints taken into account, the task allocation model for system cost may be formulated as follows:

$\min.S_{Cost}(X)$

s.t. $\sum_{k=1}^{n} x_{ik} = 1$ $\forall$ i= 1, 2, 3, ……,m. (6)

$x_{ik} \in \{0,1\}$ $\forall$ i,k. (7)

In this model, constraint-6, states that each task should be assigned to exactly one processor. Constraint-7, guarantees that, x_{ik} is being decision variable. The above model defines an integer programming problem and is known to be NP- hard problem [1, 3, 7-14, 20, 21]. An optimal solution to this problem can be found by enumerating all possible allocations. But this technique requires $O(n^m)$ time computations. This is prohibitive even for small size problems. Hence, in this paper, we present a heuristic algorithm to find quickly the solution of high quality, by ordering the tasks according to their CLS and made allocation of these tasks to processors in that order. The proposed technique has been given in next section that will find near optimal solution to the mentioned problem at all the times.

3 Proposed Task Allocation Technique And Algorithm

The technique by which the tasks comprising the program are allocated to the available processors in DCS are essential, to minimize the system cost. To achieve this objective, the order in which the tasks in a program are considered for allocation is a critical factor affecting the optimality of the resulting allocations [23].

We have selected all the tasks for allocation according to their CLS. The CLS of each task can be computed by using

$CLS\ (t_i) = \sum_{j=1}^{m} cc_{ij}$ for i = 1, 2, ……, m.

Now, all the tasks are sorted in the monotonically decreasing order of their CLS in a linear array $T_{CLS}\{\}$ and they are considered for allocation in that order. Tie breaking is done randomly i.e. one of the tasks with equal CLS is selected randomly. If the CLS of a task t_i is very high than other task i.e. the inter task communication of t_i becomes more intensive as compared to other tasks. In this case, system cost derived could be lower due to involvement of more communication links. Therefore, first we have to allocate such tasks to the processors, to minimize the IPC [2, 20]. Thus, the result will decrease in system cost.

Initially, we assume that the linear array $T_{ass}\{\} \leftarrow \Phi$ and $T_{CLS}\{\} \leftarrow T_{non\_ass}\{\}$. Now, for an optimal allocation of tasks to processors in DCS, we have defined two rules.

Rule-1

This rule is incorporated to the selection of suitable processors, whose capabilities is most appropriate for the task. We apply this rule as:

Pick up the task t_i from $T_{non\_ass}\{\}$ and assign t_i to processor P_k {k = 1, 2, 3,……., n} for which ec_{ik} is minimum. Suppose that processor is P_r. Therefore, we have assigned t_i to the r^{th} processor.

Rule-2

It is another rule incorporated to IPC caused by the inter task communication (ITC). Task t_i, which has been assigned to the r^{th} processor (say) using rule-1 in DCS, rule-2 is used to add the effect of communication cost of the executing task $t_i \rightarrow P_r$, with other tasks residing on $P_1, P_2, P_3,……., P_{n-1}$ except the r^{th} processor as:

Select the i^{th} column of ITCCM (,) and add this column to all the columns of ECM (,) except the r^{th} column. Now, we have modified both the matrices ECM (,) and ITCTM (,) by deleting the i^{th} row and column. Thus, we store the task t_i in a linear array $T_{ass}\{\}$ and the linear array $T_{non\_ass}\{\}$ is modified by deleting i^{th} task from $T_{non\_ass}\{\}$.

Hence, in the above manner, both the rules will be repeated for each task of $T_{non\_ass}\{\}$, until and unless $T_{non\_ass}\{\} \leftarrow \Phi$ and $T_{CLS}\{\} \leftarrow \{t_1, t_2,…….,t_n\} = T_{ass}\{\}$. The detailed process of allocating the tasks to the processors is given below in the form of algorithm.

3.1 Proposed algorithm

The proposed algorithm consists of the following steps.

Step-0: input: m, n, ECM (,), ITCCM (,).

Step-1: compute the communication link sum (CLS) of each task using

$CLS(t_i) = \sum_{j=1}^{m} CC_{ij}$ for i = 1, 2, ……,m.

Step-2: sort all the tasks in $T_{CLS}\{\}$ according to decreasing order of their CLS.

Step-3: initialize: $T_{CLS}\{\} \leftarrow T_{non\_ass}\{\}$

$T_{ass}\{\} \leftarrow \Phi$

Step-4: pick up the i^{th} task (say t_i) from $T_{non\_ass}\{\}$ and then

4.1: assign t_i to the appropriate processor by using Rule-1 and Rule-2.

4.2: $T_{non\_ass}\{\} \leftarrow T_{non\_ass}\{\} / \{t_i\}$

$T_{ass}\{\} \leftarrow \Phi \cup \{t_i\} = \{t_i\}$.

Step-5: for all tasks from $T_{non\_ass}\{\}$, repeat step-4 until and unless we get

$T_{non\_ass}\{\} \leftarrow \Phi$ and $T_{CLS}\{\} \leftarrow T_{ass}\{\}$

Step-6: compute:

$T_{cost}(X)_k = PEC(X)_k + IPCC(X)_k$

and $S_{cost}(X) = \sum_{k=1}^{n} T_{Cost}(X)_k$

Step-7: End.

3.2 Algorithm complexity

Using the method suggested by H. Ellis *et al.* [24], the run time complexity of the proposed algorithm can be analyzed as follows: step-1, executed in O(m) time operations. Step-2, has a worst case time complexity of O(m log m). In step-4, a single task requires O(1(n)) time operations. Therefore, for m- tasks step-5 requires O(m(n)) time operations. Thus, the overall time complexity of the proposed algorithm is O(m + m log m + mn). Since m >> n, therefore, the run time complexity of the proposed algorithm is O(mn).

4. IMPLEMENTATION OF THE MODEL

In this section, we give three numerical examples to illustrate the formulation and solution procedure of the proposed task allocation model. To show the performance of our allocation technique for better allocation, we have tested the proposed algorithm on these examples.

4.1 Example-1

The efficacy of the proposed algorithm has been shown by solving the same running example as in [16]. In this example, we have consider a DCS consists of three processors P = {P_1, P_2, P_3} and a typical program made up by 4- executable tasks T = {t_1, t_2, t_3, t_4}. Table- 1 and table- 2 shows, the execution cost of each task on processors and inter task communication cost between the tasks in the form of matrices ECM and ITCC respectively.
We have applied the proposed algorithm on this example in the following manner:

Step-0: Input: m = 4, n = 3, ECM (,), ITCCM (,).
Using these inputs, the proposed algorithm traces the following output.
Step-1: first of all, we have to calculate the communication link sum (CLS) of each task using
$CLS(t_i) = \sum_{j=1}^{m} cc_{ij}$ for i = 1, 2, ……,m.
Thus, we get CLS(t_i) = 11, 3, 14, 14 corresponding to i = 1, 2, 3, 4.

Step-2 and 3: sort all the tasks in $T_{CLS}\{\}$, according to decreasing order of their $CLS(t_i)$
$T_{CLS}\{\} = \{t_3, t_4, t_1, t_2\}$ and initially we assume,
$T_{CLS}\{\} \leftarrow T_{non\_ass}\{\} = \{t_3, t_4, t_1, t_2\}$
$T_{ass}\{\} \leftarrow \Phi$

Step-4**:** Now, pick up the first task t_3 from $T_{non\_ass}\{\}$. Apply Rule-1 for t_3. Since execution cost for t_3 is minimum on processor P_3. Therefore, assign $t_3 \rightarrow P_3$ and add the effect of communication to processor P_3 by using Rule-2. Thereafter, modify ECM (,) and ITCCM (,) by removing t_3 from ECM (,) and ITCCM (,). Thus, modified ECM (,) and ITCCM (,) have been given in table- 3 and table- 4, respectively. Thus,

$T_{non\_ass}\{\} \leftarrow T_{non\_ass}\{\} / \{t_3\}$
$= \{t_4, t_1, t_2\}$
$T_{ass}\{\} \leftarrow \Phi \cup \{t_3\}$

Step-5: For all tasks of $T_{non\_ass}\{\}$, repeat step- 4, until and unless we get
$T_{non\_ass}\{\} \leftarrow \Phi$.
$T_{CLS}\{\} \leftarrow T_{ass}\{\} = \{t_3, t_4, t_1, t_2\}$

Step-6**:** Processor wise total costs are:

$T_{COST}(X)_1 = 6$ {i.e. Total cost of processor P_1 }

$T_{COST}(X)_2 = 0$ {i.e. Total cost of processor P_2 }

$T_{COST}(X)_3 = 18$ {i.e. Total cost of processor P_3 }

and total system cost

i.e.

$S_{COST}(X) = T_{COST}(X)_1 + T_{COST}(X)_2 + T_{COST}(X)_3 = 6 + 0 + 18 = 24.$

Step-7: End.

Table-5 shows an optimal allocation of tasks to processors in DCS, for the present task allocation model. For this example,

$t_2 \rightarrow P_1$; Nil $\rightarrow P_2$ and $t_3, t_4, t_1 \rightarrow P_3$.

Thus, the optimal processor cost of P_1, P_2 and P_3 are 6, 0 and 18 respectively and the optimal value of system cost 24 with the proposed algorithm.

The results obtained with the proposed algorithm and the algorithm presented in [16], for this example has been given below in table-6.

In table- 6, our algorithm shows that the proposed algorithm tries to minimize IPCC much more than the algorithm of H. Kumar *et al.,* [16]. Thus, the proposed algorithm produces lower system cost in comparison to the algorithm presented in [16]. As we can observe from table- 6, the system cost is minimized by 11.11% than that of [16] for this example. Hence, the proposed algorithm produces near optimal allocation at all the times

4.2 Example-2

We consider a typical program made up by 9- executable tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ to be executed on the DCS having three processors $\{P_1, P_2, P_3\}$. We have taken the execution cost of each task on different processors and inter task communication cost (ITCC) between the tasks in the form of matrices ECM (,) and ITCCM (,) respectively. Both the matrices have been given in Table- 7 and Table- 8 respectively.

The results obtained with the proposed algorithm for this example has been given below in table-9.

4.3 Example-3

We consider a typical program made up by 9- executable tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ to be executed on the DCS having three processors $\{P_1, P_2, P_3\}$. We have taken the execution cost of each task on different processors and inter task communication cost (ITCC) between the tasks in the form of matrices ECM (,) and ITCCM (,) respectively. Both the matrices have been given in Table- 10 and Table- 11 respectively.

In this example, we have considered the inter task communication cost in the form of 0 or 1; i.e. either the communication between the tasks are exist or there is no communication.

The results obtained with the proposed algorithm for this example has been given below in table-12.

5. CONCLUSION

In this paper, we have looked at the problem of task allocation in DCS. But, task allocation problem is known to be NP- hard problem in complexity, when we required an optimal solution to this problem. Therefore, we have proposed an efficient algorithm, which finds near optimal system cost for the DCS, having arbitrary structure of processors. We have used static task allocation policy to achieve this objective. One of the best options to minimize the system cost, is the minimization of IPC. Therefore, the proposed algorithm tries to allocate the tasks to the processors on the basis of CLS and found that, it is a good heuristic to minimize the system cost. The performance of the proposed algorithm is compared with [16]. Also, the run time complexity of the proposed algorithm is O (mn), which is very time saving as compared to the complexities of the algorithms presented in [16, 21, 25]. Whose complexities are O (n^m), O (n^m) and O ($m^2 + mn$), respectively. For several sets of input data (m, n), a comparison between the complexities of the proposed algorithm and the complexities of the algorithms presented in [16, 21, 25], has been given in table-13 and figure-1 and it is found that the proposed algorithm is suitable for a DCS having arbitrary inter connection of processors with random program structure and workable in all the cases.

Table 1: Execution Cost Matrix

Processors→ / Tasks ↓	P_1	P_2	P_3
t_1	9	2	6
t_2	3	8	7
t_3	7	10	3
t_4	3	4	9

Table 2: The Inter Task Communication Cost Matrix

Tasks	t_1	t_2	t_3	t_4
t_1	0	1	4	6
t_2	1	0	2	0
t_3	4	2	0	8
t_4	6	0	8	0

Table 3: Modified Execution Cost Matrix

Processors→ / Tasks↓	P_1	P_2	P_3
t_1	13	6	6
t_2	5	10	7
t_4	11	12	9

Table 4: Modified Inter Task Communication Cost Matrix

↓Tasks→	t_1	t_2	t_4
t_1	0	1	6
t_2	1	0	0
t_4	6	0	0

Table 5: An Optimal Allocation of Tasks

Optimal allocation		Total optimal processor's cost	Total Optimal system cost
tasks	processors		
t_2	P_1	6	24
Nil	P_2	---	
t_3,t_4,t_1	P_3	18	

Table 6: Comparison between the proposed algorithm and the algorithm of H. Kumar *et al.* [16]

Proposed Algorithm			H. Kumar *et al.* Algorithm [16]		
Tasks	Processers	Optimal System Cost	Tasks	Processers	Optimal System Cost
t_2	P_1	24	t_2	P_1	27
Nil	P_2		t_1, t_4	P_2	
t_3,t_4,t_1	P_3		t_3	P_3	

Table 7: Execution Cost Matrix

Processors→ / Tasks↓	P_1	P_2	P_3
t_1	174	176	110

t_2	95	15	134
t_3	196	79	156
t_4	148	215	143
t_5	44	234	122
t_6	241	225	27
t_7	12	28	192
t_8	215	13	122
t_9	211	11	208

Table 8: The Inter Task Communication Cost Matrix

↓Tasks→	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
t_1	0	8	10	4	0	3	4	0	0
t_2	8	0	7	0	0	0	0	3	0
t_3	10	7	0	1	0	0	0	0	0
t_4	4	0	1	0	6	0	0	8	0
t_5	0	0	0	6	0	0	0	12	0
t_6	3	0	0	0	0	0	0	0	12
t_7	4	0	0	0	0	0	0	3	10
t_8	0	3	0	8	12	0	3	0	5
t_9	0	0	0	0	0	12	10	5	0

Table 9: An optimal allocation of tasks

Optimal Allocation		Total Optimal Processor's Cost	Total Optimal System Cost
Tasks	Processors		
t_5,t_7	P_1	91	528
t_8,t_9,t_2,t_3	P_2	137	
t_1,t_4,t_6	P_3	300	

Table 10: Execution Cost Matrix

Processors→ / Tasks↓	P_1	P_2	P_3
t_1	174	176	110
t_2	95	15	134

t_3	196	79	156
t_4	148	215	143
t_5	44	234	122
t_6	241	225	27
t_7	12	28	192
t_8	215	13	122
t_9	211	11	208

Table 11: The Inter Task Communication Cost Matrix

↓Tasks→	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
t_1	0	1	1	1	0	1	1	0	0
t_2	1	0	1	0	0	0	0	1	0
t_3	1	1	0	1	0	0	0	0	0
t_4	1	0	1	0	1	0	0	1	0
t_5	0	0	0	1	0	0	0	1	0
t_6	1	0	0	0	0	0	0	0	1
t_7	1	0	0	0	0	0	0	1	1
t_8	0	1	0	1	1	0	1	0	1
t_9	0	0	0	0	0	1	1	1	0

Table 12: An optimal allocation of tasks

Optimal Allocation		Total Optimal Processor's Cost	Total Optimal System Cost
Tasks	Processors		
t_5,t_7	P_1	60	464
t_2,t_3,t_8,t_9	P_2	122	
t_1,t_4,t_6	P_3	282	

Table 13: Results of run time complexity of the algorithms

S.NO.	Input Combination (Tasks, Processors)	Run time complexity of the algorithms		
		Richard *et al.* [21], Peng *et al.*[25] $O(n^m)$	H.Kumar *et al.*[16] $O(m^2+mn)$	Proposed Algorithm O(mn)
1	(4,3)	81	28	12
2	(5,3)	243	40	15
3	(6,4)	4096	60	24
4	(7,4)	16384	77	28
5	(8,5)	390625	104	40
6	(9,5)	1953125	126	45
7	(10,6)	60466176	160	60
8	(11,6)	362797056	187	66
9	(12,7)	13841287201	228	84
10	(13,7)	96889010407	260	91

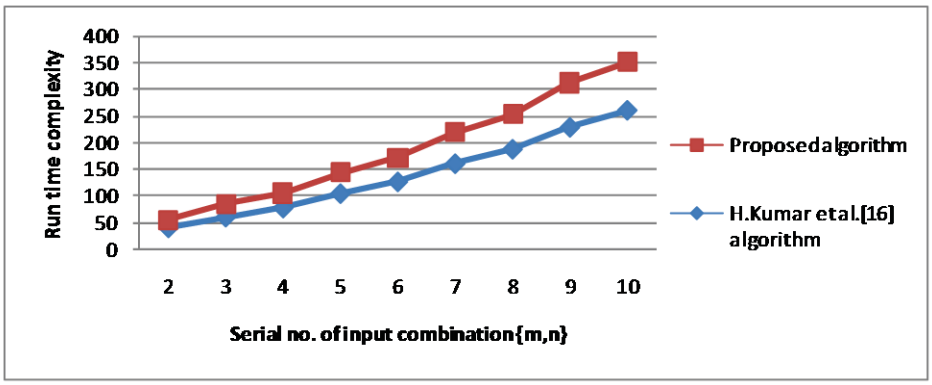

Fig.1-Comparison between the complexities of the algorithms

References

[1] Ajith Tom P. and C. Siva Ram Murthy, An Improved Algorithm For Module Allocation in Distributed Computing Systems, *Journal of Parallel and Distributed Computing Systems*," 42 (1997), pp. 82-90.

[2] G. Sagar and A.K. Sarje, Task Allocation Model for Distributed System, *Int. J. Systems Sci*. Vol. 22, 9(1991). pp. 1671- 1678.

[3] M. Kafil and I. Ahmad, Optimal Task Assignment in Heterogeneous Computing Systems, 0-8186- 7879- 8/97$ 10.00 ©1997 *IEEE* .

[4] Cheol-Hoon Lee, Dongmyun Lee and Myunghwan Kim, "Optimal Task Assignment in Linear Array Networks, *IEEE Transactions on Computers*, Vol.41, 7(1997).

[5] Imtiaz Ahmad Muhammad K. Dhodhi and Arif Ghafoor, Task Assignment in Distributed Computing Systems, *IEEE Concurrency* (1995), pp. 49- 53.

[6] Elsadek A.A. Elsadek & B.E. Wells, A heuristic model for task allocation in heterogeneous distributed computing systems, *International Journal of Computers and their Applications*, Vol.6, No.1, 1999. Pp. 0-35.

[7] Sol M. Shatz, Jia-Ping Wang and Masanori Goto, Task Allocation For Maximizing Reliability of Distributed Computer Systems, *IEEE Transactions on Computers*, Vol.41.9 (1992).

[8] S. Kartik and C. Siva Ram Murthy, Task Allocation Algorithms for Maximizing Reliability of Distributed Computing System, *IEEE Transactions on computers*, Vol.46, 6 (1997).

[9] D.J. Chen *et al.,* A Heuristic Algorithm for the Reliability- Oriented File Assignment in a Distributed Computing System," *Computers Math. Applic.* Vol. 29.10(1995), pp. 85-104,

[10] Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, Yi-Te Wang, Task Allocation for maximizing Reliability of a Distributed System using Hybrid Particle Swarm Optimization. *The Journal of Systems and Software,* 80(2007). Pp. 724-735.

[11] Santhanam Srinivasan and Niraj K. Jha, Safety and Reliability Drivan Task Allocation in Distributed Systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol.10. 3(1999).

[12] D.P. Vidayarthi and A.K. Tripathi, Maximizing reliability of Distributed Computing System with Task allocation using Simple Genetic Algorithm*, Journal of System Architecture*, 47(2001), pp.549-559.

[13] Pradeep Kumar Yadav, M.P. Singh and Kuldeep Sharma, Task Allocation Model for Reliability and Cost optimization in Distributed Computing System, *International Journal of modeling, simulation and scientific computations*, vol-2, 2(2011), pp. 1-19.

[14] Qin-Ma Kng, Hong He, Hui- Min Song, Rong Deng, Task allocation for maximizing Reliability of distributed Computing System using Honey bee mating Optimization, *The Journal of Systems and software,* Vol.83, 2010.

[15] C.C. Shen and W.H. Tasi, A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minimax Criterion, *IEEE Transactions on Computers*, Vol. C- 34, 3(1985).

[16] H. Kumar *et al.,* A Task Allocation Model for Distributed Data Network, *Journal of Mathematical Sciences*, Vol.1, 4(2006),pp.379-392.

[17] Lo V.M., Heuristic Algorithms for Task assignment in distributed systems, *IEEE Transactions on computers,* Vol.37. No. 11, pp. 1384- 1397, November 1988.

[18] K. Kfe, "Heuristic Models of Task Assignment Scheduling in Distributed Systems," Computer, Vol. 15, pp. 50- 56, June 1982.

[19] W.W. Chu, Leslie J. Holloway, Min-Tsung Lan and Kemal Kfe, Task Allocation in Distributed Data Processing, *IEEE Concurrency,* November 1980.pp.57-69.

[20] A.K. Sarje and G. Sagar, "Heuristic Model for Task Allocation in Distributed Computer Systems," *IEE Proceedings -E*, Vol.138, 5(1991).

[21] P-Y. Richard MA, Edward Y.S. Lee and Masahiro Tsuchiya, "A Task Allocation Model for Distributed Computing Systems,'' *IEEE Transactions on Computers*, Vol.C-31, 1(1982), pp.41- 46.

[22] Sung- Ho Woo, Sung- Bang Yang, Shen- Dug Kim and Tack- Don Han, "Task scheduling in distributed computing systems with a genetic algorithm," 0- 8186- 7901- 8/97 $ 10.000© 1997 *IEEE,* pp. 301-305.

[23] Hongjun Lu, "load Balanced Task Allocation in locally Distributed Computer Sciences," Technical report # 633, feb- 1996.

[24] H. Ellis, Sahni S and S. Rajsekaram, "Fundamentals of computers algorithm," *Galgotiya publication Pvt Ltd.*, (2005).

[25] Peng, Dar- Tezen, Shin, K.G. and Abdel, Zoher, T.F., "Assignment Scheduling Communication periodic Tasks in Distributed Real Time System," *IEEE Transactions on Software Engineering,* SE-13, (1997), pp. 745- 757.