International Journal of Physics and Mathematical Sciences ISSN: 2277-2111 (Online)
An Open Access, Online International Journal Available at http://www.cibtech.org/jpms.htm
2014 Vol. 4 (2) April-June, pp. 39-44/Kumar
Research Article

A SIMULATED ANNEALING APPROACH TO ASSIGNMENT PROBLEM OF PROGRAM TASKS TO PROCESSORS

***Surinder Kumar**
Department of Mathematics, S.G.G.S College, Sec-26, Chandigarh, India
**Author for Correspondence*

ABSTRACT
A simulated annealing approach to the assignment of program tasks to processors in a distributed computer system is presented. Tasks of a program require certain capacitated computer resources. They also communicate at a given rate. Processors are interconnected by a communication network constituted of various types of links: local area network (LAN), wide area network (WAN) and specialised links. The communication resources are also capacitated. The purpose is to find the assignment of tasks to processors such that a measure of performance is optimised, the requirements of each task are met and the capacities of the resources are not violated. Various versions of the problem are identified and formulated. The design of the simulated annealing algorithm to solve the most general version is then described. The results of computational experience are reported.

Keywords*: Computer Networks, Execution Cost, Communication Cost, Task Assignment, Combinatorial Optimization, Distributed Computer System, Simulated Annealing*

INTRODUCTION
Optimised assignment of program tasks to processors in a distributed system is an important problem, whose effective solution is useful during all phases of system development. During the design phase, it helps in determining the configuration that attains a certain performance level during the operational phase, it facilitates optimising the use of available resources; and during reconfiguration, it aids in reassigning tasks in response to system changes.

The problem, in various versions, has received considerable attention in the literature on parallel and multi-processor configurations; see for example Berman and Snyder (1987); Billionet *et al.*, (1992); Bokhari, (1987); Houstis and Aboelaze, (1991); Nicol and O'Halloran, (1991); and Norman and Thanisch, (1993). Existing algorithms fall into the following categories:

- Exact algorithms applied to simplified problems (Fernandez-Baca and Medepalli, 1993; Sinclair, 1987). Heuristic algorithms applied to the full problem or a simplified version (Fernandez-Baca and Medepalli, 1993; Hagin *et al.*, 1996; Lo, 1384-1397).
- Exact algorithms applied to the full problem (Hagin *et al.*, 1996).

However, since the problem is NP-complete, solving it exactly, for example by using branch and bound schemes (Hagin *et al.*, 1996), is possible only for small problem instances. Therefore, employing simulated annealing, which has proved quite effective in solving to near optimality many combinatorial optimisation problems (Reeves, 1993), is justified.

Problem Definition
The tasks of a program have a computation time requirement and a memory requirement. They also, typically, communicate data with other tasks at given rates. Since computer and communication resources are capacitated, the problem of task assignment arises: finding the assignment of tasks to processors such that a measure of performance is optimised, the requirements of each task are met and the capacities of the resources are not violated.

Communication requirements may be represented by a directed graph $G(V, E)$, called the communication graph (Figure 1), whose vertices V represent the modules and where the presence of an edge $(i, j) \in E$ signifies that communication occurs between tasks *i* and *j*. Each edge may be weighted by the channel capacity required for the communication to take place.

Depending on context, several versions of the problem may be formulated. To do so, define the following parameters:

m_i : memory requirements of task *i*.

p_i : processing requirements of task *i*.

M_n : memory capacity of processor *n*.

P_n : processing capacity of processor *n*.

c_{ijnl} : communication cost between task *i* and *j* if i is assigned to processor *n* and *j* is assigned to processor *l*.

c_{in} : cost of executing task *i* on processor *n*.

d_{ij} : communication load between tasks *i* and *j*.

D_{nl} : capacity of the virtual communication link between processor *n* and *l*.

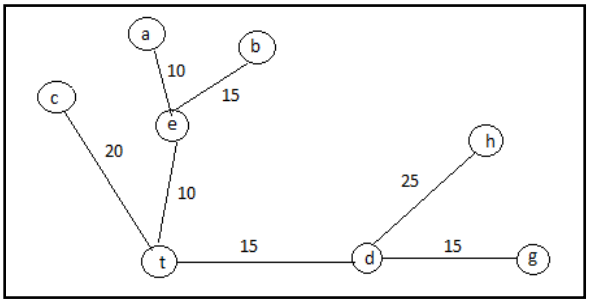

Figure 1: An example of the communiucation requirement graph

and the variable:

$$x_{in} = \begin{Bmatrix} 1 & if\ task\ i\ is\ assigned\ to\ processor\ n, \\ 0 & otherwise, \end{Bmatrix}$$

Constrained module allocation: This problem version was motivated by system where a host computer with a large memory shares its load with a smaller, more specialised processor with limited memory capacity (Rao *et al.*, 1979). Thus the problem is to minimise total system cost, subject to a resource constraint on the smaller processor, for convenience assumed to be processor 1. It can be formulated as:

$$\min \quad \sum_i \sum_n c_{in} x_{in} + \sum_{(i,j)\in E} \sum_n \sum_l c_{ijnl} x_{in} x_{jl} \qquad (1)$$

subject to

$$\sum_n x_{in} = 1 \qquad \forall i, \qquad (2)$$

$$\sum_i m_i x_{i1} \le M \qquad (3)$$

Balanced module allocation: Here the objective is to minimise the maximum processor load, as a way of obtaining a balanced assignment (Chu and Lan, 1987). The total load on a processor for a given assignment is the total execution cost of the tasks assigned to it plus the sum of the communication costs from that task to all other tasks. It is worth noting that if two communicating tasks are assigned to

different processors then the communication cost contributes to the load on both processors. The problem can be formulated as:

$$\min \max_{n} \left\{ \sum_{i} c_{in} x_{in} + \sum_{(i,j)\in E} \sum_{l} c_{ijnl} x_{in} x_{jl} \right\} \quad (4)$$

subject to

$$\sum_{n} x_{in} = 1 \qquad \forall i, \quad (5)$$

$$\sum_{i} m_{i} x_{in} \le M_{n} \qquad \forall n \quad (6)$$

General constrained task allocation. The objective here is to minimise total cost subject to both memory and processing capacity constraints for each processor. The problem can be formulated as (1) Subject to

$$\sum_{n} x_{in} = 1 \qquad \forall i, \quad (7)$$

$$\sum_{i} m_{i} x_{in} \le M_{n} \qquad \forall n, \quad (8)$$

$$\sum_{i} p_{i} x_{in} \le P_{n} \qquad \forall n, \quad (9)$$

General constrained task allocation with network constraints. In this problem, it is required to take account of the limited capacities of the resources that constitute the communication network (LANs, WANs and direct links) as shown in Figure 2.

One approach is to define a virtual link between each pair of processors and assign to it an appropriate capacity. Thereafter, it is assumed that if a pair of tasks is assigned to the pair of processors

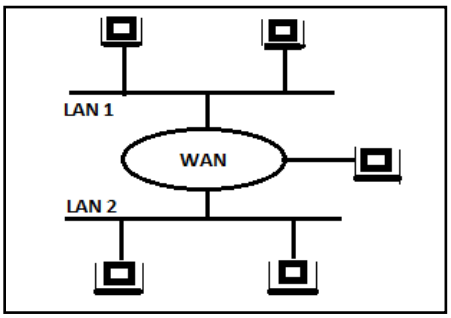

Figure 2: An example of a communication network

under consideration, then the tasks impose a known communication load on the virtual link between the processors. The problem can then be formulated as (1) subject to (7)-(9) and

$$\sum_{i} \sum_{j} d_{ij} x_{in} x_{jl} \le D_{nl} \qquad \forall n, l. \quad (10)$$

A more flexible approach is to identify the possible communication paths between each pair of processors and the communication resources on which each path is incident. The problem then becomes that of assigning tasks to processors, as well as choosing the communication path between each pair of processors such that the capacity of each communication resource is not violated. The cost of

communication between each specific two tasks assigned to each two specific processor could also be made dependent on the communication path chosen. By introducing binary variables to indicate the choice of each path or otherwise, the problem can be cast in a formulation similar to the one just cited above. However, such a formulation is cumbersome and is left out for that reason.
It is this last version of the problem that will be addressed in the sequel. However, it will become clear that the approach presented is applicable equally well to the less general versions.

$$\delta_{ijnl} \le x_{in}, \tag{11}$$

$$\delta_{ijnl} \le x_{jl} \tag{12}$$

$$\delta_{ijnl} \ge x_{in} + x_{jl} - 1, \tag{13}$$

The above formulations are all binary, nonlinear programming problems. It is possible to transform them into binary linear programming problems using standard device. Thus let δ_{ijnl} be a binary variable such that then each occurrence of $x_{in}x_{jl}$ can be substituted by δ_{ijnl} ,while adding the appropriate constraints (11)-(13). However, it is clear that this is at the expense of a significant increase in the number of both variables and constraints.
Whatever the version and whether a non-linear or a linear formulation is adopted, clearly, attempting to solve the task allocation problem to optimality is bound to fail for all bar insignificantly small problem instances.

The Simulated Annealing Algorithm

Since the basic outline of the simulated annealing algorithm is well known, we will confine ourselves here to presenting our development of its basic characteristic component; namely, the coding of solutions, the neighbourhood structure and the cooling schedule, for the problem under consideration.
Coding of solutions. Solutions are coded by assigning a program task to a processor and choosing a communication path between each pair of processors. Let the number of processors be *N*, the number of paths between processors *i* and *j* be P_{ij} and the number of tasks be *M.* Two array variables, $y(M \times 1)$ and $z(N \times N)$, are then defined such that:

- $y_m \in (1,2,...,N)$; Where y_m is the number of the processor to which task *m* is assigned,
- $z_{ij} \in (1,2,...,P_{ij})$, $\forall i, j \in (1,2,...,N)$; where z_{ij} is the number of the path used for communication between processors *i* and *j*.

Neighbourhood structure: A neighbouring solution is obtained by choosing at random one of two moves. The first is effected by moving a by choosing at random a task m from one processor to another (thus changing the value of y_{ij}), while the second is carried out by substituting at random the communication path between two processors thus randomly changing the value of the chosen variable z_{ij} .The choice of type move is done with equal probability.
*Calculating the cost***.** Tables of costs and resource utilisation coefficients are assembled a priori, so that the cost of moves can be calculated incrementally, independently of problem size. Two matrices containing the sub-costs of communication and assignment are kept from one iteration to another and the change is applied to modified variables only. Constraint violations are handled by adding appropriate penalties.
Cooling schedule. A geometric cooling schedule is used. The temperature is reduced in so that $T_{k+1} = \alpha \times T_k$, where α is a constant less than 1. The chain at each temperature is updated in a similar manner, $L_{k+1} = \beta \times L_k$, where β is a constant greater than1.

The initial temperature is set after executing a sufficiently large number of random moves, such that the worst move would be allowed.
Given

- m_1 and m_2 ,m the numbers corresponding cost reduction cost increase, respectively.
- δ° : the average cost increase value of the m_2 trials.
- $T_\circ$: the initial temperature,
- $a_\circ$: the desired initial acceptance value.

The following relation may then be written:

$$a_\circ = \frac{m_1 + m_2 e^{-\delta^\circ / T_\circ}}{m_1 + m_2},$$

which gives

$$T_\circ = -\frac{\delta^\circ}{\log\left(\frac{m_1}{m_2}(a_\circ - 1) + a_\circ\right)}$$

The final temperature is chosen for to give a low acceptance value.

Table 1: Computation times for five various size case studies

Case	No. of modules	No. of processors	No. of paths b/w two processors	Total no. of variables	Computation time (min)
1	7	5	3	110	14
2	9	6	3	162	24
3	10	7	3	217	37
4	12	8	3	288	54
5	14	10	3	440	103

RESULTS AND CONCLUSIONS

The algorithm has been coded in Matlab, with the computation time optimized by using the parallel computation options. Several real-life problem instances have been solved satisfactorily. The computation times proved to be rather long. For example, a problem instance involving 10 processors, 7 sub networks, 14 program tasks and 14 information flow edges required 103 minutes on an HP 712/60 workstation (for other cases refer to Table 1). However, computation time could be reduced drastically if the algorithm is coded and run in a more efficient environment; for example in C.
Although the solutions obtained appear excellent, there are no benchmark problems with known solutions available. For this reason, among others, we are currently developing a branch and bound algorithm with a linear evaluation procedure, in the hope of solving at least medium size problem instances for comparison.

REFERENCES

Berman F and Snyder L (1987). On mapping parallel algorithms in parallel architecture. *Journal of Parallel and Distributed Computing* **4**(5) 439-458.
Billionet A, Costa MC and Sutter A (1992). An efficient algorithm for a task allocation problem. *Journal of ACM* **39** 502-518.
Bokhari S (1987). *Assignment Problems in Parallel and Distributed Computing* (Kluwer, Boston).
Chu WW and Lan LM (1987). Task allocation and precedence relations for distributed real-time systems. *IEEE Transactions on Computers* **36** 667-669.

Fernandez-Baca D and Medepalli A (1993). Parametric module allocation on partial k-trees. *IEEE Transactions on Computers* **42** 738-742.
Hagin A, Dermler G and Rotherml K (1996). Problem formulation, model and algorithms for mapping distributed multimedia applications to distributed computer systems, Technical Report, University at Stuttgart, Fakultat Informatik.
Houstis CE and Aboelaze M (1991). A comparative performance analysis of mapping applications to parallel multi-processor systems: A case study. *Journal of Parallel and Distributed Computing* 17-29.
Lo VM (1988). Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers* 1384-1397.
Nicol DM and O'Halloran DR (1991). Improved algorithms for mapping pipelined and parallel computations. *IEEE Transactions on Computers* **40** 295-306.
Norman MG and Thanisch P (1993). Models of machines and computation for mapping in multi-computers. *ACM Computing Surveys* **3** 163-302.
Rao GS, Stone HS and Hu TC (1979). Assignment of tasks in a distributed processing system with limited memory. *IEEE Transactions on Computers* **28** 291-299.
Reeves CR (1993). *Modern Heuristic Techniques for Combinatorial Problems* (Blackwell Scientific Publications, Oxford).
Sinclair JB (1987). Efficient computation of optimal assignments for distributed tasks. *Journal of Parallel and Distributed Computing* **4** 342-362.