CROSS ENTROPY BASED MODULE ALLOCATION FOR DISTRIBUTED SYSTEMS

Niklas Widell and Christian Nyberg
Department of Communication Systems
Lund University
Box 118
SE-221 00 Lund, Sweden
Phone: +46 46 2227195

{niklasw,cn}@telecom.lth.se

Keywords: Optimization algorithm, randomized algorithm, resource allocation, distributed computing

Abstract

In the module allocation problem a collection of software modules are to be assigned to physical processing nodes, subject to execution and communication cost. The cost of an allocation is a function of the execution costs and the communication costs for any pair of modules allocated to distinct processors. The module allocation problem has been well studied and is known to be NP-complete except for certain communication configurations. To solve the problem, several heuristics have been proposed. This paper discusses an alternative approach to solving the module allocation problem by applying a stochastic optimization method called the Cross Entropy (CE) Method. The CE Method is a state-of-the-art stochastic method for solving combinatorial and multi-extremal continuous optimization problems. The CE method uses a distribution with parameter $\mathbf{v}$ to generate sample allocation. The generated samples are then used to update $\mathbf{v}$ according to sample quality. This process is repeated until the distribution converges to a possibly optimal allocation. The results in this paper indicate that the CE method can successfully be applied to the module allocation problem and efficiently generate high quality solutions. Also, the CE method allows the use of non-standard objective functions that are used to find allocations that have multiple conflicting objectives.

1 Introduction

An important problem that arises in distributed computer systems is the allocation of software modules to physical processing nodes. The problem is relevant in architectures such as CORBA or Web Services that consist of software modules communicating via a common middle-ware layer, providing transparent distribution to the applications. Such systems are both flexible and dynamic. However, to fully realize the potential of a distributed system, mechanisms that make efficient use of resources must be introduced. Good allocations may lead to better throughput, reliability, processing times and better resource usage. Poor allocations often lead to poor system performance. Thus, finding good module allocations is an important way to improve performance and reduce resource consumption.

This paper investigates the use of a novel stochastic optimization method called the Cross Entropy (CE) method for solving the module allocation problem. The CE method for combinatorial optimization was first proposed by Rubinstein [1]. The CE method transforms the deterministic optimization problem into a stochastic one, and then uses rare event simulation techniques to solve the problem. The CE method applied to module allocation involves iteration over two phases:

1. Generate sample allocations according to a given distribution parameterized by a parameter $\mathbf{v}$.

2. Update $\mathbf{v}$ based on the relative performance of the samples generated in the previous phase, in order to generate "better" samples in the next iteration.

Two things are needed apply the CE method to a problem: a sample generation mechanism and update rules for the distribution used in the sample generation process. This paper presents such a mechanism with updating rules for module allocation problem.

Both theoretical and heuristic methods have previously been applied to the module allocation problem.

Theoretical methods, such as branch-and-bound and integer programming, will find the optimal solution. However, as shown by Fernández-Baca [2], the module allocation problem is NP-complete, except for certain special cases. Stone [3] used a network flow model to solve the problem with two nodes. Ma *et al* [4] presented a branch-and-bound based solution and Bastarrica *et al* [5] discussed solutions using integer programming. Choi and Wu [6] presented a multi-objective solution using the Niched Pareto Genetic-Algorithm.

Heuristic methods are fast and efficient, but find sub-optimal solutions. Heuristic methods commonly include

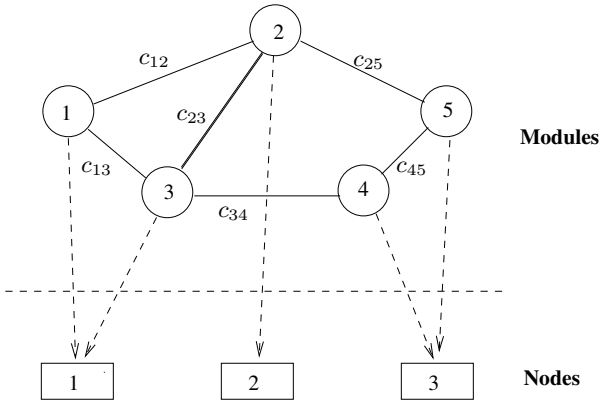


Figure 1. System model

ways for partitioning the communication graph. Efe [7], Lo [8] and Stoyenko *et al* [9] each presented algorithms for clustering modules that communicate much and then allocate clusters to nodes. Woodside and Monforton [10] described a several heuristics that speed up allocation using a bin-packing technique.

In comparison with the mentioned methods, our use of the CE method for module allocation is based on stochastic optimization theory. Our method requires neither special problem structure nor heuristic assumptions. The method is also largely independent of choice of objective function.

The main contribution of this paper is to show how the Cross Entropy Method can be used to solve the module allocation problem in an efficient way. The paper shows how to apply the algorithm together with a discussion of the parameters required to operate the algorithm.

The rest of the paper has the following structure: Section 2 presents a model of a distributed module based system. Section 3 presents the module allocation problem and how it can be approached. Section 4 describes the CE method and how it can be used to solve the module allocation problem. Section 5 presents two experiments to evaluate the CE method for module allocation as well as to compare it to other previous methods. Finally, section 6 discusses conclusions from the experiments.

2 Model

In this section we outline a general model of a distributed module based system, where communicating software modules are to be allocated to physical processing nodes in a network. In figure 1, the rectangles are nodes, circles are modules, lines are communication requirements between modules and dashed lines indicate an allocation of modules to nodes.

In the model we will use the term *cost* as a measure of what needs to be optimized. The cost is a relevant measurable parameter such as MIPS or load generated by the software modules.

The physical resources in the model are represented by a collection of N processing nodes. The nodes are heterogeneous, and a particular module incurs different costs when executed on different processors, due to varying computational facilities at each node. The nodes belong to a common domain with high network speed compared to node processing capacity. Thus, network latency is negligible, and the nodes can be assumed to be fully connected.

The software in the system is represented by by M modules. The execution cost for a module is given by an $N \times M$ matrix $\mathbf{E}$, where element e_{ij} is the execution cost for module j allocated to node i. $e_{ij} = \infty$ if module j may not be allocated to node i.

The communication model is similar to Bokhari [11], where the actual communication cost between two modules is zero if they are co-allocated on the same node. Otherwise, the communication cost is given by an $M \times M$ matrix $\mathbf{C}$, where element c_{jk} is the communication cost if modules j and k are allocated to different nodes. Note that $c_{jj} = 0$ and we assume that $c_{jk} = c_{kj}$. The communication cost can also be viewed as a graph of interconnected modules, with edges representing that two modules communicate. The communication cost models the protocol handling required to support communication between nodes, and is a cost for the nodes hosting the modules involved in communication. We do not consider the loading of the underlying network, since we have already assumed that it is of high enough capacity.

The allocation of modules to nodes is given by a allocation matrix α with binary elements. Element $\alpha_{ij} = 1$ if module j is allocated to node i, $\alpha_{ij} = 0$ otherwise. When several allocations are considered at the same time, let $\alpha_{k,ij}$ be α_{ij} for allocation k.

3 Module allocation problem

The module allocation problem is to allocate modules to physical nodes in a way that best satisfies a given set of objectives and constraints. In this paper, we use three different performance-related objectives: minimization of total cost, minimization of maximum node cost and minimization of allocation badness. First, let the the cost w_i for node i given allocation α be defined by:

$$w_i(\alpha) = \sum_{j=1}^{M}(e_j\alpha_{ij} + \sum_{k=1}^{M} c_{jk}\alpha_{ij}(1-\alpha_{ik})) \qquad (1)$$

The *total cost minimization* objective was formulated by Stone [3], and is known to be NP-complete, except for certain communication configurations configurations. In our notation, total cost minimization is equivalent to

$$\min_{\alpha} L_{com} = \sum_{i=1}^{N} w_i(\alpha) \qquad (2)$$

This objective makes no effort to balance the cost on the nodes. However, to prevent clustering, constraints limiting either the total cost (see Efe [7]) or other factors such as memory consumption of modules on nodes (see Roupin [12]) can be introduced. Note that in a homogeneous system, (2) is equivalent to minimizing total communication cost.

In addition to (2), the second most common objective in literature (for instance, see Woodside *et al*[10] is *min-max* node cost. In our notation,

$$\min_{\alpha}(L_{min-max} = \max w_i(\alpha)) \tag{3}$$

By minimizing the maximum node cost the optimal allocation found will have both balanced cost and low communication.

The CE method for module allocation is not restricted to using the previously mentioned functions. We use this advantage to introduce the notion of *allocation badness*. The allocation badness for node i is:

$$b_i(\alpha) = e^{(w_i(\alpha)-w_{i,ref})/w_{i,ref}} \tag{4}$$

where $w_{i,ref}$ is equal to the average execution cost of all modules. Note that $b_i = 1$ for balanced allocations with no communication. The objective is to minimize allocation badness:

$$\min_{\alpha} L_{badness} = \sum_{i=1}^{N} b_i(\alpha) \tag{5}$$

By using (5), communication minimization and load balancing is achieved.

4 The Cross Entropy Method

This section outlines the Cross Entropy Method (The CE method), and how it can be applied to module allocation problems.

The CE method was introduced by Rubinstein [1] as a stochastic method for combinatorial optimization problems. Rubinstein saw a link between importance sampling estimation of rare events and combinatorial optimization problems, such as the Traveling Salesman Problem, Shortest Path problems and so on.

The main idea behind the CE method is to transform the original deterministic optimization problem into an associated stochastic problem, and then tackle the stochastic problem using an adaptive sampling algorithm. In the process, a sequence of random solutions is constructed that converges probabilistically to the optimal or a near-optimal solution.

In each iteration, importance sampling is used to alter the allocation distribution matrix P_t, amplifying the probability for finding better allocations. Cross Entropy minimization is used to efficiently update parameters. In addition, a performance function further weights the allocation qualities to give more influence to better allocations in the alteration of the allocation matrix. The objective is to let the allocation matrix converge to a binary matrix giving the optimal allocation. Cross Entropy is a measure of distance between two distributions and it is defined in [1].

The CE method is different from other stochastic optimization methods such as Simulated Annealing (SA) or Genetic Algorithms (GA). The CE method does not do local search like SA or GA, instead it performs a global search in the entire solution space.

In this paper we use the Boltzmann version of the two-stage Cross Entropy Algorithm in [1]. This version weighs the allocation according to the performance of the allocation. According to [1], the Boltzmann version is superior to the originally suggested threshold version of the CE method.

4.1 Sample generation and initialization of P_t

The samples used in the algorithm are generated using a $N \times M$ probability matrix P. Let P_t be the probability matrix used in iteration t. Element $P_{t,ij}$ is then the probability that module j is allocated to node i in iteration t.

The initial value for P_t is arbitrary for a unconstrained allocation, as long as there is a high probability that a module j is allocated to a node j in the first iteration of the algorithm. For example, $P_{t=0,ij} = 1/N$. The column sum of P_t is equal to 1.

As mentioned in section 2, security and reliability requirements that restrict certain modules to a subset of available nodes must be taken into account. This is done by adjusting P_t accordingly by setting $P_{t=0,ij} = 0$ for unavailable i:s and scaling up the remaining probabilities.

In a system with homogeneous processing times the initial P_0 can be chosen in a way that the number of generated permutations is reduced. First, place module 1 on node 1 ($P_{0,11}$=1.0), place module 2 on *either* node 1 or 2 with equal probabilities ($P_{0,12} = P_{0,22} = 0.5$), continuing until the first i modules have been allocated. With this method, P_0 will contain zeros below the diagonal of the first i columns. Every column will still sum up to 1.0, however.

4.2 Cross Entropy Algorithm for Module Allocation

0. Initialize $P_{t=0}$.

1. Generate a set A of K sample allocations α_k using P_t. Each sample α_k represents an allocation of modules to nodes. Let $L(\alpha_k)$ be the cost of allocation α_k, where L is one of the objective functions defined in section 3. Calculate the minimum Boltzmann temperature γ_t to fulfill the average allocation performance:

$$\max \gamma_t \tag{6}$$

$$\text{s.t.} h(P_t, \gamma_t) = \frac{1}{K}\sum_{k=1}^{K} H(\alpha_k, \gamma_t) > \rho \tag{7}$$

where $H(\alpha_k, \gamma_t) = e^{-L(\alpha_k)/\gamma_t}$ is the Boltzmann performance function for allocation α_k, ρ is a search focus parameter, typically in the range of $10^{-2} \leq \rho \leq 10^{-6}$. A small value of ρ improves convergence speed, but is more likely to get caught in local minimizes.

2. Using γ_t together with generated set of samples A from step 1 and $H(\alpha_k, \gamma_t)$, update P_t by minimizing the distance to the optimal matrix. This is done by solving the following problem

$$\max_{P_{t+1}} \frac{1}{K} \sum_{k=1}^{K} H(\alpha_k, \gamma_t) \sum_{ij \in \alpha_k} \ln P_{t,ij} \qquad (8)$$

The solution to this problem is shown in [1] to be

$$P_{t+1,ij} = C_t \sum_{k=1}^{K} \alpha_{k,ij} H(\alpha_k, \gamma_t) \qquad (9)$$

where $C_t = 1/(\sum_{l=1}^{K} H(\alpha_l, \gamma_t))$ is a normalization constant for iteration t. Equation 9 will minimize the Cross Entropy between P_t and P_{t+1} and optimally improve allocation probabilities given the calculated γ_t and the performance function.

3. If convergence then stop iteration, else $t = t+1$ and go to step 1. A criterion for convergence is that $\gamma_{t-k} = \gamma_{t-k+1} = ... = \gamma_t$ for a given k, say $k = 5$.

Say that the algorithms terminates at $t = \tau$ the algorithm returns two (possibly equivalent) allocations $\alpha^{iteropt}$ and $\alpha^{globopt}$. $\alpha^{iteropt}$ is the optimal allocation as given by the converged probability matrix P_τ while $\alpha^{globopt}$ is the best allocation found during the entire run of the algorithm. Note that $L(\alpha^{globopt}) \leq L(\alpha^{iteropt})$.

4.3 Algorithm Parameters

The sample size K should be adjusted to the values for N and M. A large K will generate more statistically stable results requiring fewer iterations. However, the positive effect of increasing K is decreased if K grows too large. If a small K is used the results will fluctuate more. The effect of small K can be helped by the introduction of a smoothing function, to prevent that $P_{t,ij}$:s may be reduced to zero prematurely:

$$P_{t,ij} = \hat{P}_{t,ij} * \beta + P_{t-1,ij} * (1 - \beta)$$

where $\hat{P}_{t,ij}$ is the value calculated in the algorithm at iteration t and β is a smoothing parameter. While the objective of the algorithm is to converge P_t to zeros and ones, doing so prematurely decreases the efficiency of the algorithm.

However, it should be noted that the presented algorithm is quite insensitive to exact choice of parameters. As long as ρ is not to small, the smoothing parameter β is < 1 and K is large enough, the results of the algorithm are robust.

4.4 Problem constraints

In the presented algorithm, there is no provision to ensure that generated samples are feasible. Since all costs are positive, infeasibility typically occurs when more cost is allocated to a node than it can hold. There are three ways to handle infeasible samples: (1) Leave them as they are, (2) Artificially give infeasible samples very low performance (i.e. a very high $L(\alpha_k)$), and (3) draw new samples until all are feasible. Depending on the problem, feasibility is not a problem if most generated samples are feasible.

4.5 Global and iteration optimal solutions

Even if the CE method typically is fast to converge for a problem, it has an additional property that is very useful: By saving the best sample from all iterations so far, $\alpha^{globopt}$, at least one high quality solution is known at all times.

5 Experiments

In this section we describe two experiments using the the CE method for allocation of modules. The first experiment applies the CE method to a sample problem previously by Efe [7]. The second experiment compares the accuracy of the CE method using the objectives presented in section 3.

5.1 An illustrative example

As an illustration of how the CE method works, we will now show how the method works for a small example previously investigated by Efe [7], Williams [13] and Woodside and Monforton [10]. Efe used a clustering algorithm, which forms K clusters of the M modules so as to minimize inter-cluster communication, and then a module reassignment algorithm that did pairwise module interchange based on node loading. Williams ordered tasks by communication factors and then allocated them in that order, using the execution costs for placement. Woodside and Monforton used an heuristic extension of the MULTIFIT algorithm (which in turn was initially investigated by Coffman [14]), that used an iterative bin-packing technique, where costs where increased according to communication cost.

The example uses three homogeneous nodes and ten modules. The module execution costs and communication costs are found in the module graph is found in figure 2. The objective used was the same as in Woodside, min-max cost balancing (3). The CE method parameters used were: search focus parameter $\rho = 10^{-2}$, sample size $K = 5 \cdot N \cdot M = 150$ samples and convergence parameter (in step 3) $k = 5$.

The CE method algorithm was run 50 times. The algorithm converged on the optimal solution in 80% of the cases. However, in all cases, the optimal solution $\alpha^{globopt}$ was found during the sampling process already after a few iterations. Using 100 samples per iteration, the mean number of iterations before convergence was 41.

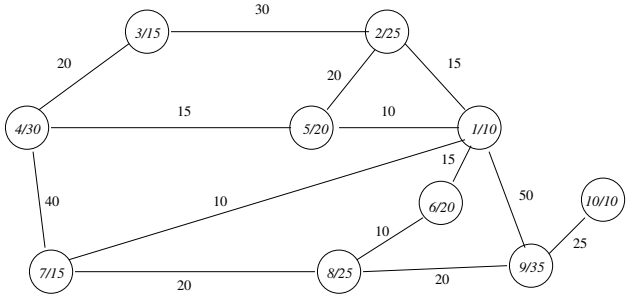


Figure 2. Module graph for Efe's example. Labels on edges are communication costs and labels on vertices are module index followed by execution costs. Note that nodes are homogeneous in the example.

Table 1 contains results from the CE method compared to other methods. The values in the table are the optimal cost L for a node as well as the modules allocated in the optimal case.

N	*Widell L/modules*	*Woodside L/modules*	*Efe L/modules*	*Williams L/modules*
1	120 (2,3,5)	120 (2,3,5)	120 (2,3,5)	115 (2,5)
2	145 (4,7,8)	145 (4,7,8)	145 (4,7,8)	180 (3,4,6,7)
3	140 (1,6,9,10)	140 (1,6,9,10)	140 (1,6,9,10)	160 (1,8,9,10)

Table 1: Cost and allocated modules per node, for optimal solution given Efe's example.

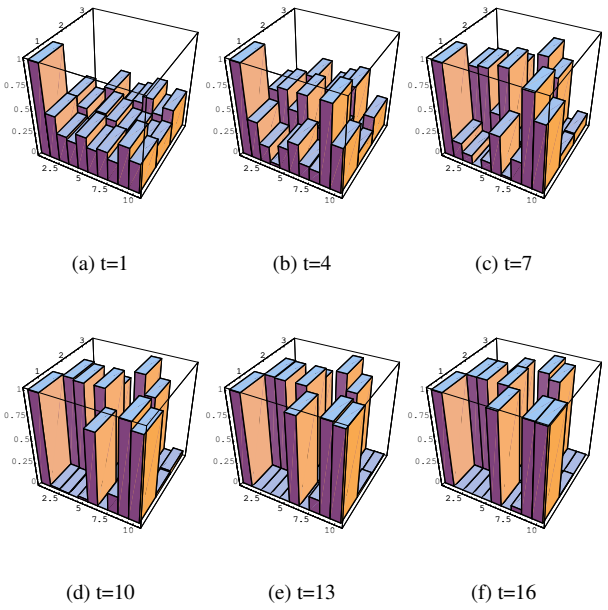
To further show the operation of the CE method, figure 3 shows the matrix P_t is at different iterations t of the algorithm. In this particular instance, the algorithm converged after 33 iterations. The height of the bars represent the probability that a module will be allocated to that node in the next iteration. Note that since the system is homogeneous, the algorithms converge to a permutation of the optimal allocation. Note also that the permutation-reduction trick was used for the initial distribution.

In figure 3 we see that initial convergence is quite fast, but the algorithm takes time to settle. One reason for this is that we are using the min-max objective function. The CE method performs better with additive functions. A less strict convergence criteria will also decrease convergence time.

Despite these observations, the results from this experiment indicate that the CE method can be useful in finding good allocations, without any regard to problem structure.

5.2 A large symmetric system

We have also investigated the performance of the CE method on larger systems. Since the allocation problem is very hard, we have used artificially constructed problems, with known solutions, to compare our converged solutions with.



(a) t=1 (b) t=4 (c) t=7

(d) t=10 (e) t=13 (f) t=16

Figure 3. Illustration of convergence for Efe's example, with convergence at $t = 33$ (last iterations not shown, since they are not visibly different)

As an example, we used homogeneous systems with the following characteristics. Module execution cost was same on all nodes ($e_{ij} = 10$ for $i = 1..N$ and $j = 1..M$). We let the number of modules be $M = G \cdot N$. Modules were clustered in groups of size G, with all modules within a group communicating only within the group. The resulting communication cost matrix was (for a example with $N = 3$ and $G = 3$):

$$\mathbf{C} = \begin{pmatrix} 0 & 7 & 7 & 0 & 0 & \dots \\ 7 & 0 & 7 & 0 & 0 & \dots \\ 7 & 7 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 7 & \dots \\ 0 & 0 & 0 & 7 & 0 & \dots \\ & & \dots & & & \ddots \end{pmatrix}$$

For this example it is easy to see that the optimal allocations for all three objectives (2), (3) and (5) are those that allocate all modules in a group to the same node. In addition for the latter objective, only one group is allocated to one node.

We ran the allocation algorithm 10 times for each objective and cases. Each iteration used $N \cdot M$ samples. The results are summarized in table 2. The table shows algorithm accuracy (*Acc.*) for finding the optimal allocation for each case and objective, and $\bar{t}$ is the average number of iterations iterations to do so. The accuracy is the number out of 10 times the algorithm converged on the optimal value. For the cases that did not find an optimal solution, the convergence in all occasions was to allocations with at most

Objective	*Comm. cost. (2)*		*Minimax (3)*		*Badness (5)*	
Case	*Acc.*	t	*Acc.*	t	*Acc.*	t
$N = 6, M = 24$	10	53	8	74	9	31
$N = 8, M = 32$	9	78	6	104	8	63
$N = 10, M = 40$	8	111	4	143	7	105

Table 2: Accuracy (*Acc*) and average number of iterations for the symmetric problem.

two modules that were not allocated to the same node as the rest of their group.

We see in the table 2 that the CE method is most accurate for the two additive objective functions (minimize communication cost (2) and minimize badness (5)). This is as predicted in Rubinstein [1]. The explanation is that the additive functions carries more information about solution quality than the minimax function. This also explains why the two additive functions converge much faster, as more information from the generated samples are carried over to the new distribution. Note that convergence is quickest for the minimize badness, as this function puts a stronger penalty to poor allocations than the other two.

6 Conclusions

This paper presents how to apply the Cross Entropy (CE) method to the module allocation problem in distributed systems. The CE method is a stochastic method for combinatorial optimization. Each iteration of the CE method involves two steps: first generation of sample allocations using a parameterized distribution and then updating the distribution's parameters according to the qualities of the generated samples. The distribution converges after a number of iterations, producing a solution. In addition to the converged solution, the sampling process of the algorithm also finds high quality solutions, so that even if the algorithm is terminated before full convergence, the best solution so far is available. For module allocation, the CE method allows for module allocation without requiring special structure of the system, which is required by other methods. Experiments described in the paper show that the CE method finds the optimal in a large number of cases, and performs at least as well as previous heuristics described in literature. In addition, the CE method handles more complex objective functions than is usually used in described heuristics. However, we also see that the efficiency of the CE method is decreased if non-additive objective functions are used.

7 Acknowledgments

This work has been partially sponsored by the Swedish Research Council under contract no. 621-2001-3053.

Many thanks to Professor Bjarne Helvik, Institutt for Telematikk NTNU, Trondheim, Norway, for his invaluable help on the basics of the Cross Entropy Method.

References

[1] R. Rubinstein. The cross-entropy method for combinatorial and continous optimization. *Methodology and Computing in Applied Probability*, Vol 1: 127-190, 1999.

[2] D. Fernández-Baca. Allocating modules to processors in a distributed system. *IEEE Transactions on Software Engineering*, Vol 15, No 11, 1989.

[3] H. S. Stone. Multiprocessor sceduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering*, Vol 3, No 1, 1977.

[4] P. Y. R Ma, E. Y. S. Lee, and M. Tsuchiya. A task allocation model for distributed computing systems. *IEEE Transactions on Computers*, Vol C-31, No 1, 1982.

[5] M. C. Bastarrica, S. A. Demurjian Sr, and A. A. Shvartsman. Software architectural specification for optimal object distribution. In *Proc. of XVIII International Conference of the Chilean Society of Computer Science, SCCC'98, Antofagasta, Chile, IEEE Press*, 1998.

[6] S. Choi and W. Chisu. Partitioning and allocation of objects in heterogeneous distributed environments using niched Pareto genetic-algorithm. In *Asia Pacific Software Engineering Conference*, 1998.

[7] K. Efe. Heuristic models of task assignment scheduling in distributed systems. *Computer*, June, 1982.

[8] V. M. Lo. Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers*, Vol 37, No 1, 1988.

[9] A. Stoyenko, J. Bosch, M. Aksit, and T. Marlowe. Load balanced mapping of distributed objects to minimize network communication. *Journal of Parallel and Distributed Computing*, Vol 34: 48-66, 1996.

[10] C. Murray Woodside and Gerald G. Monforton. Fast allocation of processes in distributed and parallel systems. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):164–174, 1993.

[11] S. H. Bokhari. Partitioning problems in parallel pipelined and distributed computing. *IEEE Transactions on Computers*, Vol 37, No 1, 1988.

[12] F. Roupin. On approximating the memory-constrained module allocation problem. *Information Processing Letters*, No 61, 1997.

[13] E. A. Williams. *Design, Analysis and Implementation of Distributed Systems from a Perormance Persptective*. PhD thesis, University of Texas at Austin, 1983.

[14] E. G. Coffman. *Computer and Job Shop Scheduling*. John Wiley and Sons Inc., 1976.