Multiple Cost Optimization for Task Assignment in Heterogeneous Computing Systems Using Learning Automata

Raju D. Venkataramana
Dept. of Computer Science and Engineering
University Of South Florida
Tampa, FL 33620, USA
venkatar@csee.usf.edu

N. Ranganathan
Dept. of Electrical and Computer Engineering
University of Texas at El Paso
El Paso, TX 79968, USA
ranganat@ece.utep.edu

Abstract

A framework for task assignment in heterogeneous computing systems is presented in this work. The framework is based on a learning automata model. The proposed model can be used for dynamic task assignment and scheduling and can adapt itself to changes in the hardware or network environment. The important feature of the scheme is that it can work on multiple cost criteria, optimizing each criterion individually. The cost criterion could be a general metric like minimizing the total execution time, or an application specific metric defined by the user. The application task is modeled as a task flow graph(TFG), and the network of machines as a processor graph(PG). The automata model is constructed by associating every task in the TFG with a variable structure learning automaton [1]. The actions of each automaton correspond to the nodes in the PG. The reinforcement scheme of the automaton considered here is a linear scheme. Different heursitic techniques that guide the automata model to the optimal solution are presented. These heuristics are evaluated with respect to different cost metrics.

Key words: *Task assignment, variable structure learning automata, multiple cost criteria, stochastic optimization, task flow graph and processor graph.*

1. Introduction

Heterogeneous computing(HC) [2, 8, 15], is the tuned use of diverse processing hardware to meet distinct computational needs. A HC environment consists of a heterogeneous suite of machines and high speed interconnections, which can be used effectively to execute different portions of an application. An application is usually represented as a task flow graph (TFG), which is a directed acyclic graph showing the data dependencies between the various tasks of the application. An important problem in this domain is the task assignment problem, which corresponds to the assignment of tasks to the machines in the HC suite such that a specified cost criterion is optimized. It is well known that the assignment problem in general is NP-complete [6].

In the literature, a variety of mathematical formulations have been developed for the task assignment problem. These formulations are collectively called *selection theory* and try to choose the appropriate machine for each subtask of the TFG [6, 7, 9, 10, 4]. Approaches to the problem based on graph theoretic techniques [5, 4], simulated annealing [14], exhaustive state space search [16], and genetic techniques [12, 14, 11] have been proposed. In this work, the authors propose a new learning automata model for the assignment problem, which is based on a variable-structure learning automaton [1]. In [3], a stochastic learning automaton model was proposed for scheduling in distributed systems. The systems were assumed to be homogeneous and had a large number of automaton states, 2^M for an M-node network. The model proposed here, associates an automaton with every task in the TFG, and hence there are only M states for an M-node network. The different reinforcement schemes of the automata define the behavior of the assignment algorithm. The key feature of the proposed model is its ability to optimize multiple cost metrics. In other works multimetric optimization is achieved by optimizing the weighted sum of the various metrics. In the proposed model, the cost metrics are optimized independent of each other subject to the weight associated with it. The cost criterion itself could be a general metric like minimizing the total execution time, or it could be defined specifically to suit the needs of an application.

The paper is organized as follows. Section 2 introduces the framework, followed by a description of the HC system model and the cost criteria in section 3. A construction of the automata model and the heuristic techniques proposed

for the model are presented in the next section. The subject of section 5 is the performance analysis of these techniques. The last section provides the conclusions and scope for future work.

2. Model of the Framework

This section presents a general model of the proposed framework for task assignment in the HC system. Figure 1 depicts a schematic representation of the framework. The environment consists of the heterogeneous suite of machines which will be used to execute the application. The scheduling system consists of the proposed learning automata model and a model of the application and HC system. These models are used by the scheduler to assign the subtasks to the different machines.

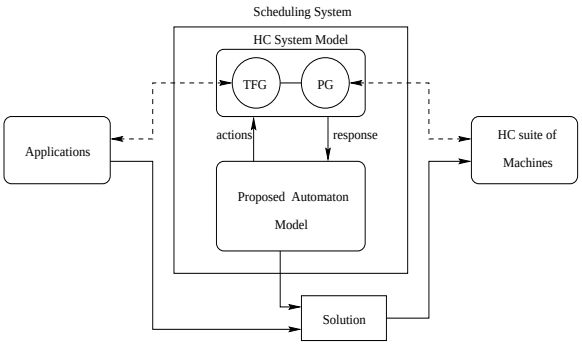

Figure 1. Model of the Proposed Framework

The HC system environment consists of a heterogeneous suite of machines interconnected in a specific topology. These machines could include sequential processors, MIMD systems, SIMD systems, or special purpose architectures and DSP processors. All these various systems are assumed to be interconnected by a high speed interconnection network. The application task is assumed to be represented as a task flow graph(TFG). The TFG is a directed acyclic graph which shows the dependencies of the subtasks. The interconnection of the machines in the HC system is represented similarily by means of a processor graph(PG). The objective of the scheduling system is to assign the subtasks of the TFG to the processors in the PG, so that the defined set of cost criteria are optimized. This is achieved by means of a automata model proposed in this paper. The model is built on a variable structure learning automaton and uses an iterative algorithm. The external environment for this automaton model is the system model formed by the TFG and the PG. Once the scheduling system determines on which processors the various tasks are to be executed, the actual execution on the HC system can take place.

The advantages of the proposed framework can be seen quite clearly. Since the actual task assignment and scheduling is based on a model of the application(TFG) and the machines of the HC system(PG), the framework can be used for dynamic task assignment and scheduling. The proposed learning automaton model optimizes the set of cost criteria based on a given PG. Hence, whenever there is a change in the machine or network configuration, the model will generate a solution that is optimal to the transformed system configuration. The important feature of the proposed framework, is the incorporation of multiple cost criteria, all of which are optimized simultaneously. The following sections explain the construction of the framework and highlight its advantages.

3. HC System Model

This section deals with the modeling of the application and the hardware configuration of the HC system. Certain assumptions that have been made are introduced first. The later part of the section discusses how the different cost metrics for the system can be defined. The task assignment algorithm will generate a solution that tries to optimize these cost criteria.

3.1. Assumptions

1. The application program is assumed to be decomposed into multiple tasks. The data dependencies between the tasks are given by a Task Flow Graph(TFG), which is an acyclic directed graph.
2. The HC system is assumed to consist of a set of heterogeneous machines, which communicate by means of an underlying interconnection network. This is represented by a Processor graph(PG).
3. The expected execution time of the tasks on each machine in the HC suite is known a priori. These execution times can be obtained by task profiling and analytical benchmarking techniques.
4. The cost of communicating a single unit of data between any two machines is also known a priori. If any pair of machines in the HC suite cannot communicate with each other, then the cost of communication between them is assumed to be ∞.

3.2. Cost Metric

The TFG contains a set of tasks S. s_i is the ith task in S. The set of machines in the HC environment is given by M, where m_j is the jth machine. C denotes the set of cost metrics defined for the application. The assignment problem then corresponds to a mapping π from the set S to the set M, such that the metrics in C are optimized.

$S = \{s_i, 0 \leq i < |S|\}$
$M = \{m_j, 0 \leq j < |M|\}$
$C = \{c_k, 0 \leq k < |C|\}$ where c_k is a specific cost metric
$\pi : S \rightarrow M$

A solution vector $X(n)$ of size $|S|$ x $|M|$, gives an instance of the mapping π at iteration n. Each element of the vector, $x_i(n)$ indicates the machine to which that particular task is assigned to at iteration n. The cost of executing a task on a machine, the communication time between the machines, the number of data units exchanged between tasks and the amount of power consumed by executing a task on a machine, are all given by the following matrices.

$X(n) \rightarrow$ Solution Vector of order $|S|$ x $|M|$
$EX \rightarrow$ Execution cost matrix of order $|S|$ x $|M|$
$CC \rightarrow$ Communication cost matrix of order $|M|$ x $|M|$
$DX \rightarrow$ Data exchange matrix of order $|S|$ x $|S|$
$PW \rightarrow$ Power cost matrix of order $|S|$ x $|M|$

The next part in formulating the problem involves the actual definition of the cost metrics. There are a number of such metrics that can be defined by the user based on the application. For instance, the user might want to minimize the total execution time, or minimize the maximum loaded processor, or minimize the amount of power consumed. There may also be other metrics that are specific to a particular application. These may be defined with the help of the matrices defined above. Each of these metrics is defined at an iteration n as $c_k(n)$. A couple of these metrics are defined here:

Minimize the load on the maximum loaded processor:

Let the total load on a particular machine say m_j at iteration n be represented as $l_j(n)$. Therefore

$$l_j(n) = \sum_{i=0,x_i(n)=j}^{|S|-1} ex_{ix_i(n)} + \sum_{i=0,x_i(n)=j}^{|S|-2} \sum_{k=i+1}^{|S|-1} dx_{ik} * cc_{x_i(n)x_k(n)}$$

In the equation for $l_j(n)$, the first part represents the total computation cost on the processor m_j and the second part corresponds to the total communication cost incurred by the processor at iteration n. The load on every processor needs to be computed, in order to identify the heaviest loaded processor. The optimal assignment for the problem would be that which results in minimizing the load on the heaviest loaded processor. Therefore, the cost metric $c_1(n)$ would be:

$$c_1(n) = Max(\ l_j(n)\ ) \qquad \text{for } j = 1 \text{ to } |M| - 1$$

Minimize the total execution time:

Let $cp(n)$ represent the total computation time for a particular assignment and $cc(n)$ represent the total communication time at a particular iteration. Hence:

$$cp(n) = \sum_{i=0}^{|S|-1} ex_{ix_i(n)}$$
$$cc(n) = \sum_{i=0}^{|S|-2} \sum_{k=i+1}^{|S|-1} dx_{ik} * cc_{x_i(n)x_k(n)}$$

The total execution time is the sum of the total computation and communication time for the task assignment at an iteration. Therefore for this cost metric, $c_2(n)$ would be:

$$c_2(n) = cp(n) + cc(n)$$

Minimize the power consumed:

The cost metric for this formulation, $c_3(n)$ can be represented as:

$$c_3(n) = \sum_{i=0}^{|S|-1} pw_{ix_i(n)}$$

The task assignment algorithm, would now try to minimize each of the $c_k(n)$ ($k = 1$ *to* $|C|$)) metrics over n iterations until the absolute minimum is reached.

4. Proposed Learning Automata Model

In the proposed framework, a learning automata model is used to determine an optimal task assignment for the HC system. This model is based on a variable structure learning automaton explained in [1]. The section begins with an introduction to this automaton model. The mathematical formulation and advantages of using the model are discussed. The construciton of such an automata model for the purpose of task assignment in the HC system forms the later part of this section. The section details how the model for the system is constructed. The HC system model, which is reflective of the actual HC system, serves as the external environment for the automata model.

4.1. Variable Structure Learning Automata

The variable structure learning automata is represented by the quintuple $\{\phi, \alpha, \beta, A, G\}$, where

1. The state of the automaton at any iteration n, denoted by $\phi(n)$ is an element of the finite set
$\phi = \{\phi_1, \phi_2, ..., \phi_s\}$
2. The output of the automaton at the iteration n, denoted by $\alpha(n)$, is an element of the set
$\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$.
3. The input to the automaton at the iteration n, denoted by $\beta(n)$, is an element of the set
$\beta = \{\beta_1, \beta_2, ..., \beta_m\}$.
4. A, is the updating algorithm or the reinforcement scheme.
5. The output function $G(.)$, determines the output of the automaton at any iteration n in terms of the state at that iteration.
$\alpha(n) = G[\phi(n)]$.

If each state of the quintuple corresponds to a unique action and the number of states are finite (as will be the case with our HC system), then $r = s < \infty$ and hence G is an identity mapping. Therefore, the automata can be represented as the triple $\{\alpha, \beta, A\}$.

An action probability $p_j(n)$ is associated with every action or state of the automaton at any iteration n. $p_j(n)$ represents the probability that the automata will be in state j at an iteration n, amongst the r possible states of the automaton. The updating algorithm or the reinforcement scheme, A, attempts to update the action probabilities at every iteration n and brings the automaton to an optimal solution state. The precise manner in which the action α_i performed at an iteration n and the response $\beta(n)$ of the environment change the probability $p_j(n)$, completely defines the reinforcement scheme. If $p_j(n+1)$ is a linear function of $p_j(n)$, then the reinforcement scheme is said to be linear, otherwise it is termed non-linear.

Reinforcement Scheme

Consider a variable-structure automaton with r actions to be operating in a stationary environment with $\beta = \{0,1\}$. In other words, the automaton is operating in a P-model environment, where the response is binary valued. Let N be the set of nonnegative integers and let $n \epsilon N$. A general scheme to update the action probabilities can be represented as:

If $\alpha(n) = \alpha_i \quad (i = 1, 2, ..., r)$

$p_j(n+1) = p_j(n) - g_j[p(n)]$ when $\beta(n) = 0$

$p_j(n+1) = p_j(n) + h_j[p(n)]$ when $\beta(n) = 1$

for all $j \neq i$.

Now, we have $\sum_{j=1}^{r} p_j(n) = 1$, in order to preserve the probability measure. Therefore

$p_i(n+1) = p_i(n) + \sum_{j=1, j\neq i}^{r} g_j(p(n))$ when $\beta(n) = 0$.

$p_i(n+1) = p_i(n) - \sum_{j=1, j\neq i}^{r} h_j(p(n))$ when $\beta(n) = 1$.

Consider the variable structure automaton operating in the S-model environment, where the responses can take continuous values over an interval. By normalization, it can be assumed that the interval is a unit interval [0,1]. Hence, β for this model is a continuous variable where $\beta \in [0,1]$. The equations for the S-model can be written as:

$$p_i(n+1) = p_i(n) - (1-\beta(n)) * g_i(p(n)) + \beta(n) * h_i(p(n)$$
$$\text{if } \alpha(n) \neq \alpha_i$$
$$p_i(n+1) = p_i(n) + (1-\beta(n)) * \sum_{j\neq i} g_j(p(n)) - \beta(n) * \sum_{j\neq i} h_j(p(n)$$
$$\text{if } \alpha(n) = \alpha_i$$

The following assumptions are made with regard to the function g_j and $h_j (j = 1, 2..., r)$.

$Assumption\ 1$: g_j and h_j are continuous functions.

$Assumption\ 2$: g_j and h_j are non-negative functions.

$Assumption\ 3$: $0 < g_j(p) < p_j$

$$0 < \sum_{j=1, j\neq i}^{r} [p_j + h_j(p)] < 1$$

for all $i = 1, 2, ...r$ and all p whose elements are all in the open interval (0,1).

Different reinforcement schemes are represented by how the functions $g(.)$ and $h(.)$ are characterized. For instance if the functions are linear representations of the action probabilities, then scheme is linear, otherwise they are non-linear. It's also possible to have hybrid scheme's. Essentially, the nature of the reinforcement scheme affects the behavior of the model.

4.2. Construction of the Model

Figure 2 shows the schematic of the learning automata model. The model is constructed by associating every task s_i in the TFG with a variable structure automaton. Each of the automata are represented as a 3-tuple, $(\alpha^{s_i}, \beta^{s_i}, A^{s_i})$. Since the tasks can be assigned to any of the $|M|$ machines, the action set of the automata are identical. It is assumed that the environment is a S-model environment and hence the domain of the input set to the automata is in the interval $[0,1]$.

Therefore for any task s_i, $0 \leq i < |S|$:

$\alpha^{s_i} = m_1, m_2, ...m_{|M|-1}$

$\beta^{s_i} \in [0,1]$

where β^{s_i} *closer* to 0, indicates that the action taken by the automaton of task s_i was favorable to the system, and *closer* to 1 indicates an unfavorable response.

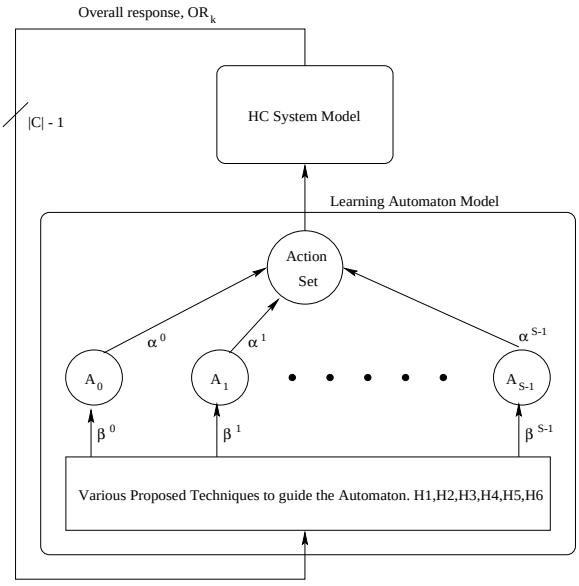

Figure 2. Learning Automata Model

The output response of the HC system model corresponding to each of the cost metrics in the set C, is assumed to be binary. In other words, if the value of a particular cost metric $c_k \in C$ at iteration n, is more optimal than its value at iteration $n-1$, then the output is favorable and if not the response is unfavorable.

If $c_k(n)$ is more optimal then $c_k(n-1)$

then $OR_k = 0$, else $OR_k = 1$ where $1 \leq k \leq |C|$

where OR_k is the overall output response of the system with

respect to $c_k(n)$.

The next step in developing the model is to translate OR_k into the inputs $\beta^{s_i}(n)$ for each of the automata. This involves two steps. First, OR_k would have to be translated into the input to each of the s_i automata. Let $\mu_k^{s_i}$ represent this value. Hence, $\mu_k^{s_i}(n)$ corresponds to the input to the automaton ($\alpha^{s_i}, \beta^{s_i}, A^{s_i}$), with respect to cost metric c_k at iteration n. Assume that λ_k indicates the weightage associated with the cost metric $c_k(n)$. Now the second step of this two step process can be represented as:

$\beta^{s_i}(n) = \sum_{k=1}^{|C|} \lambda_k * \mu_k^{s_i}$ for $i = 0$ *to* $|S| - 1$

where

$\sum_{k=1}^{|C|} \lambda_k = 1$ and $\lambda_k \geq 0$ for $k = 1$ *to* $|C|$

There are different ways of accomplishing the first step. The simplest and easiest way would be to make OR_k as the input to each of the automata in the model.

$H1 : \mu_k^{s_i}(n) = OR_k$ for all $s_i \in S$, and $c_k \in C$

$H1$ is simple to implement and completely ignores the structure of the TFG, while guiding the learning of the automata in the model. To put it differently, using $H1$ it is possible that an unfavorable output response from the system may be falsely interpreted as an unfavorable input to a particular automaton.

Ideally, the model should generate an assignment that results in a consistent favorable output response from the system. It is difficult to achieve this objective based on the structure of the TFG. To overcome this problem, the authors propose techniques based on the history of the actions generated. With every action of an automaton, two additional fields are associated. The first one indicates the number of favorable responses, and the second field indicates the number of unfavorable responses, corresponding to a particular action. Consequently:

FAV $\rightarrow$ Matrix of order $|S|$ x $|M|$ x $|C|$, the elements $fav[i, j, k]$ of which indicate the number of times a favorable response resulted when task s_i was assigned to machine m_j for cost metric c_k.

$UNFAV$ $\rightarrow$ Matrix of order $|S|$ x $|M|$ x $|C|$, the elements $unfav[i, j, k]$ of which indicate the number of times an unfavorable response resulted when task s_i was assigned to machine m_j for cost metric c_k.

Based on this history information, different heuristic techniques can be developed that will help guide the automata to the optimal assignment. Five general heuristic techniques that the authors experimented with are presented here. Knowledge about the nature of an application can be exploited to develop heuritics specific to that application. But in this work, no assumptions are made with respect to the applications and consequently the heuristics can be used in any environment. The results of these techniques are compared with each other, and with the technique H1 that is not dependent on the structure of the TFG.

$H2$: If $fav[i, x_i(n), k] > unfav[i, x_i(n), k]$
then $\mu_k^{s_i}(n) = 0$, else $\mu_k^{s_i}(n) = 1$
for all $s_i \in S$, and $c_k \in C$

Since the objective is to maximize the number of favorable responses and minimize the unfavorable ones, this heuristic assigns an input to the automaton as favorable if for that particular action the number of favorable responses is greater than the unfavorable ones.

$H3$: If $(fav[i, x_i(n), k] - unfav[i, x_i(n)], k)$ is the maximum for all actions of $s_i \in S$, and $c_k \in C$
then $\mu_k^{s_i}(n) = 0$, else $\mu_k^{s_i}(n) = 1$

Here the difference in favorable and unfavorable responses is compared against all other actions. If the chosen action gives the maximum value, the automaton input is termed favorable.

$H4$:If($fav[i, x_i(n), k]$ - $min_{0 \leq p < |S|, 0 \leq q < |M|}$ $fav[p, q, k]$ $+max_{0 \leq p < |S|, 0 \leq q < |M|} unfav[p, q, k]$- $unfav[i, x_i(n), k]$) $>$ ($max_{0 \leq p < |S|, 0 \leq q < |M|}$ $fav[p, q, k]$ - $fav[i, x_i(n), k]$ + $unfav[i, x_i(n), k]$ - $min_{0 \leq p < |S|, 0 \leq q < |M|}$ $unfav[p, q, k]$)
then $\mu_k^{s_i}(n) = 0$, else $\mu_k^{s_i}(n) = 1$,
for all $s_i \in S$, and $c_k \in C$

In this heuristic, if the chosen action has a high value for favorable responses and a low value for unfavorable responses, the input is favorable.

$H5$: If $fav[i, x_i(n), k] = max_{0 \leq p < |S|, 0 \leq q < |M|}$ $fav[p, q, k]$,
then $\mu_k^{s_i}(n) = 0$, else $\mu_k^{s_i}(n) = 1$,
for all $s_i \in S$, and $c_k \in C$

In H5, if the chosen action has the maximum favorable responses, the automaton input is favorable else it is unfavorable.

$H6$: If $unfav[i, x_i(n), k] = min_{0 \leq p < |S|, 0 \leq q < |M|}$ $unfav[p, q, k]$,
then $\mu_k^{s_i}(n) = 0$, else $\mu_k^{s_i}(n) = 1$,
for all $s_i \in S$, and $c_k \in C$

Similarily, in H6 if the chosen action has the least unfavorable responses the input is favorable.

In order to complete the model, action probabilities need to assigned for the actions of the automata. The action set α^{s_i}, of an automaton is equal to the set $|M|$ as discussed earlier. Hence, the action probability would be the probability of assigning a task s_i to one of the machines. Let this probility be $p_{ij}(n)$. Therefore:

$p_{ij}(n)$ $\rightarrow$ the probability of assigning task s_i to machine m_j.

To preserve the probability measure, the sum of all the action probabilities for a particular task should equal one. Therefore:

$\sum_{j=0}^{|M|-1} p_{ij}(n) = 1 \quad$ for any $s_i \in S$

Reinforcement Scheme

The general reinforcement scheme for this model can now be formulated in terms of the action probabilities. Following the construction of the scheme from [1], consider the automaton for a task s_i:

$$p_{ij}(n+1) = p_{ij}(n) - ((1 - \beta^{s_i}(n)) * g(p_{ij}(n))) + (\beta^{s_i}(n) * h(p_{ij}(n))) \quad \text{for all } j \neq x_i(n)$$

$$p_{ix_i(n)}(n+1) = p_{ix_i(n)}(n) + ((1 - \beta^{s_i}(n)) * \sum_{j=0, j\neq x_i(n)}^{|M|-1} g(p_{ij}(n))) - (\beta^{s_i}(n) * \sum_{j=0, j\neq x_i(n)}^{|M|-1} h(p_{ij}(n)))$$

The nature of the functions $g(.)$ and $h(.)$ determine the nature of the reinforcement scheme and hence the performance of the scheduling system. These functions could be linear, non-linear or hybrid. For our study here, it is assumed that the functions are linear and are defined as:

$g(p_{ij}(n)) = a * (p_{ij}(n))$

and

$h(p_{ij}(n)) = b \,/\, (|M| - 1) - (b * p_{ij}(n))$

Parameters a and b are known as reward and penalty parameters respectively, and help guide the automaton to the optimal solution. The choice of these parameters affects the behavior of each of the proposed techniques. The next section details the performance analysis of these techniques.

5. Results and Discussion

In this section, the different heuristics that are proposed are analysed with respect to specific cost metrics. It begins with an introduction to the simulation environment and subsequently the results are presented and analysed.

The performance of the different techniques was evaluated by using randomly generated task graphs and processor graphs. The execution cost, communication time, data exchange and the power cost matrices were randomly generated over some predefined ranges with uniform probability. A specific set of cost metrics was then chosen and the performance evaluated for varying values of the reward and penalty parameters, a and b. The automata model was iterated until the probability of a chosen action in each automaton exceeded 0.99, or the number of iterations reached a maximum limit. If the former condition stopped the automata, then the model was said to converge. The latter condition meant the model was non-convergent. Table 1 shows the predefined ranges used to generate the random graphs. The efficiency of an algorithm is determined by the quality of the solution generated and the fastness of the algorithm.

Number of tasks	10
Number of machines	7
Number of edges	5,10,15,20 and 25
Execution matrix data range	1000
Communication matrix data range	4
Data exchange matrix data range	500
Power matrix data range	25

Table 1. Parameters for TFG and PG

Therefore, in order to study the performance of our model, the number of iterations required for convergence and the solution cost generated are studied as a function of the communication complexity.

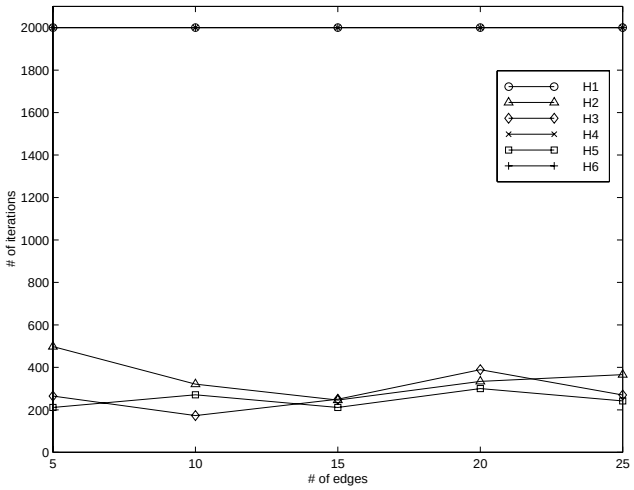

Figure 3. # of iterations vs Edges: a = b = 0.1

Initially, to verify the working of the model, a single cost metric c_1(refer section 3) was chosen. Two sets of studies were performed. For the first one, the reward and penalty parameters a and b were made equal, $a = b = 0.1$, and the maximum limit for the number of iterations was set at 2000. For the second set, $a = 0.1$ and $b = 0.009$, and the limit for the number of iterations was set at 10000. Figures 3 and 4, show the results for the first set of data. From the graphs, it can be concluded that when $a = b$, techniques $H1, H4$ and $H6$ are non-convergent and heuristic $H2$ provides the best results for all communication complexities. Figures 5 and 6, show the results for the second set of experiments. Here, techniques $H4$ and $H6$ continue to remain non-convergent but $H1$ converges though it requires considerably higher number of iterations. The cost graph indicates $H1$ generates better results than $H2$.

The first set of experimental results validate the working of the model. The best solutions generated by the proposed techniques were close to the optimal solution. The results

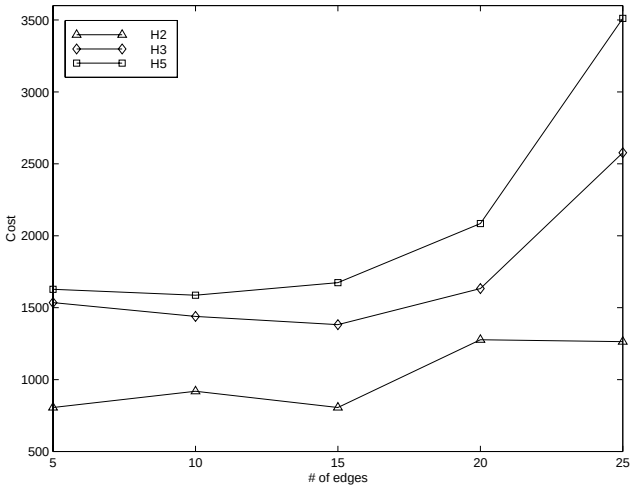

Figure 4. Cost vs Edges: a = b = 0.1

indicate that the choice of the reward and penalty parameters affect the solution cost and the choice of the heuristics.

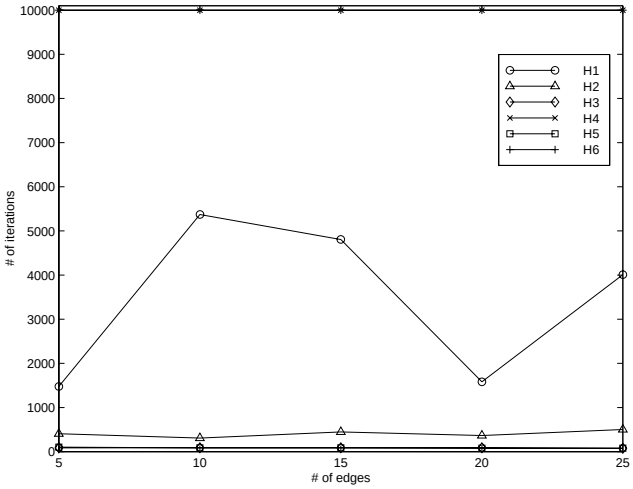

Figure 5. # of iterations vs Edges: a = 0.1,b = 0.009

In order to study the effectiveness of the model for multiple cost metric optimization, two cost metrics were chosen, c_1 and c_3(refer section 3). Experiments were performed by varying the weights associated with the metrics, λ_{c_1} and λ_{c_3} such that $\lambda_{c_1} + \lambda_{c_3} = 1$. The maximum limit for the number of iterations was set at 2000. The values for the reward and penalty parameters were chosen to get the best results. The results that were obtained have been tabulated in Tables 2,3,4 and 5. Techniques H4 and H6 were once again non-convergent. The effect of the weights associated with cost metrics can clearly be seen in all the tables. The first observation that can be made from the tables is that, as the weight associated with a particular cost metric is decreased the optimality of its solution also decreased. Essentially therefore, the automata model achieves independent optimization of the cost metrics subject to its weight. This is different from other works in that, the weighted sum of the metrics are optimized instead of independent optimization. With regards to the heuristic techniques, it can be seen that technique H1 produces the best results. For each of the graphs that were generated the optimal cost of metric c_1 was 985 and that of c_2 was 17. From the table it can be seen clearly that solution generated by H1 is close to the optimal. Technique H2 produced the next best results. Both H1 and H2 converge in almost the same number of iterations. This is in sharp contrast to techniques H3 and H5 which converge very fast. But the solution space generated by these techniques are far from the optimal. This is very pronounced as the communication complexity increases. But H1 on the other hand continued to produce optimal solutions even with increased communication complexity.

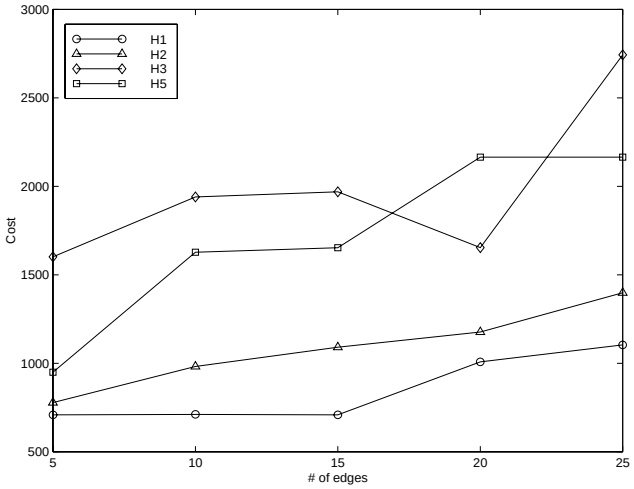

Figure 6. Cost vs Edges: a = 0.1,b = 0.009

6. Conclusions and Future Work

In conclusion, this work presented a framework for task assignment in heterogeneous computing systems. The framework was based on a learning automaton model and optimizes multiple cost metrics. The proposed model can be used for dynamic task assignment and scheduling. The model can also adapt itself to changing hardware environments. The cost metrics could be application specific or

# of edges	λ_{c_1}	λ_{c_3}	c_1	c_3	Iterations
5	1.00	0.00	985	51	420
	0.75	0.25	985	30	529
	0.50	0.50	1039	17	504
	0.25	0.75	1125	17	476
	0.00	1.00	1168	17	366
10	1.00	0.00	985	48	430
	0.75	0.25	985	34	460
	0.50	0.50	1103	17	549
	0.25	0.75	1192	17	383
	0.00	1.00	1265	17	342
15	1.00	0.00	985	53	453
	0.75	0.25	985	24	960
	0.50	0.50	1154	18	564
	0.25	0.75	1342	17	516
	0.00	1.00	1431	17	266
20	1.00	0.00	985	62	327
	0.75	0.25	985	27	1021
	0.50	0.50	1561	19	476
	0.25	0.75	1776	17	572
	0.00	1.00	1776	17	246
25	1.00	0.00	985	55	907
	0.75	0.25	1119	38	1390
	0.50	0.50	1460	20	460
	0.25	0.75	1911	17	456
	0.00	1.00	1776	17	246

Table 2. Cost values for Technique H1

# of edges	λ_{c_1}	λ_{c_3}	c_1	c_3	Iterations
5	1.00	0.00	1185	57	255
	0.75	0.25	1168	33	295
	0.50	0.50	1263	23	427
	0.25	0.75	1165	23	388
	0.00	1.00	1201	19	457
10	1.00	0.00	1156	57	317
	0.75	0.25	1168	37	341
	0.50	0.50	1192	29	266
	0.25	0.75	1265	19	340
	0.00	1.00	1215	19	457
15	1.00	0.00	1137	54	304
	0.75	0.25	1357	39	292
	0.50	0.50	1392	19	373
	0.25	0.75	1342	28	241
	0.00	1.00	1489	19	369
20	1.00	0.00	1692	59	306
	0.75	0.25	1634	58	389
	0.50	0.50	1561	30	336
	0.25	0.75	2061	19	226
	0.00	1.00	1909	19	443
25	1.00	0.00	1647	67	304
	0.75	0.25	1685	58	354
	0.50	0.50	1718	24	280
	0.25	0.75	1776	17	460
	0.00	1.00	1909	19	443

Table 3. Cost values for Technique H2

a general metric. The model was constructed by associating every task in the TFG with a learning automaton. The various automata work in unison to produce the optimal assignment of tasks. The paper presented various techniques to guide the learning of the automata. These techniques are not specific to any application and are applicable to any environment. The performance analysis of these techniques revealed that technique $H1$ results in the most optimal solution when both a single cost metric and multiple cost metrics are considered. It can therefore be concluded that for the proposed framework, the structure of the TFG does not adversely affect the task assignment procedure. However, techniques specific to a class of applications can be developed that will produce better results. Techniques H3 and H5 provide fast results but they are sub-optimal. As future work, it would be interesting to study how these techniques perform for application specific cost metrics. Also, a thorough analysis of the effect of the reward and penalty parameters in guiding the model to the optimal solution would be essential.

# of edges	λ_{c_1}	λ_{c_3}	c_1	c_3	Iterations
5	1.00	0.00	1541	61	81
	0.75	0.25	1119	79	85
	0.50	0.50	1257	68	86
	0.25	0.75	1489	34	72
	0.00	1.00	2446	32	81
10	1.00	0.00	1909	79	76
	0.75	0.25	1851	75	84
	0.50	0.50	1489	44	89
	0.25	0.75	2150	46	77
	0.00	1.00	2446	32	79
15	1.00	0.00	1793	95	80
	0.75	0.25	1975	105	80
	0.50	0.50	2075	59	75
	0.25	0.75	2470	44	79
	0.00	1.00	2472	32	78
20	1.00	0.00	3287	121	77
	0.75	0.25	2044	94	82
	0.50	0.50	2689	49	80
	0.25	0.75	3521	25	90
	0.00	1.00	2847	32	78
25	1.00	0.00	3287	121	77
	0.75	0.25	3001	184	81
	0.50	0.50	2689	42	76
	0.25	0.75	2061	46	86
	0.00	1.00	2847	32	79

Table 4. Cost values for Technique H3

References

[1] K. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

# of edges	λ_{c_1}	λ_{c_3}	c_1	c_3	Iterations
5	1.00	0.00	1431	012	84
	0.75	0.25	1591	88	80
	0.50	0.50	1621	85	78
	0.25	0.75	1591	50	81
	0.00	1.00	1591	50	81
10	1.00	0.00	1621	97	75
	0.75	0.25	1591	88	85
	0.50	0.50	1633	85	78
	0.25	0.75	2280	50	81
	0.00	1.00	2280	50	81
15	1.00	0.00	1985	97	76
	0.75	0.25	1665	97	78
	0.50	0.50	2266	72	75
	0.25	0.75	2334	50	81
	0.00	1.00	2334	50	81
20	1.00	0.00	1993	92	80
	0.75	0.25	2949	83	79
	0.50	0.50	2266	85	79
	0.25	0.75	2255	56	76
	0.00	1.00	3948	50	81
25	1.00	0.00	2949	102	81
	0.75	0.25	2949	96	77
	0.50	0.50	2266	85	79
	0.25	0.75	2255	56	77
	0.00	1.00	3948	50	81

Table 5. Cost values for Technique H5

[2] M. M. Eshaghian. *Heterogeneous Computing,* Artech House, Norwood, MA, 1996.

[3] R. Mirchandaney and J. A. Stankovic. Using Stochastic Learning Automata for Job Scheduling in Distributed Processing Systems. *Journal of Parallel and Distributed Computing*, 3:527–552, 1986.

[4] B. Narahari et al. Matching and Scheduling in a Generalized Optimal Selection Theory. *Proceedings of Heterogeneous Computing Workshop*, 3–8, April 1994.

[5] H. S. Stone. Multiprocessor Scheduling with the aid of Network Flow Algorithms. *IEEE Trans. Software Eng.*, 3(1):85–93, January 1977.

[6] R. F. Freund. Optimal Selection Theory for Superconcurrency. *Proceedings Supercomputing '89*, 699–703, November 1989.

[7] R. F. Freund. SuperC or Distributed Heterogeneous HPC. *Computing Systems in Engineering*, 2(9):349–355, 1991.

[8] R. F. Freund and H. J. Seigel. Heterogeneous Processing. *IEEE Computer*, 26(6):13–17, June 1993.

[9] M. Wang et al. Augmenting the Optimal Selection Theory for Superconcurrency. *Proceedings of the Workshop on Heterogeneous Processing*, 13–22, 1992.

[10] S.Chen et al. Selection Theory for Methodology for Heterogeneous Supercomputing. *Proceedings of Heterogeneous Computing Workshop*, 5–22, April 1993.

[11] L. Wang et al. Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach. *Journal of Parallel and Distributed Computing*, 47:8–22, November 1997.

[12] H. Singh and A. Youssef. Mapping and Scheduling Heterogeneous Task Graphs using Genetic Algorithms. *Proceedings of Heterogeneous Computing Workshop*, 86–97, April 1996.

[13] C. Hui and S. T. Chanson. Allocating Task Interaction Graphs to Processors in Heterogeneous Networks. *IEEE Trans. on Parallel and Distributed Systems*, 8(9):908–923, September 1997.

[14] P. Shroff et al. Genetic Simulated Annealing for Scheduling Data-dependent tasks in Heterogeneous Environments. *Proceedings of Heterogeneous Computing Workshop*, 98–117, April 1996.

[15] H. J. Seigel et al. Heterogeneous Computing. *Parallel and Distributed Computing Handbook*, A.Y. Zomaya, McGraw-Hill, New York, 725–761, 1996.

[16] C. C. Shen and W. H. Tsai. A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minimax Criterion. *IEEE Trans. on Computer*, 34(3):197–203, March 1985.

Raju D. Venkataramana is currently pursuing a PhD in the Dept. of Computer Science and Engineering at the University of South Florida, Tampa. He received his B.E. in Computer Science and Engineering from Sri Venkateswara College of Engineering, University of Madras, India and Masters in Computer Science from the University of South Florida,Tampa in 1995 and 1997 respectively. His research interests include heterogeneous computing, parallel processing, interconnection networks and VLSI design.

N. Ranganathan is currently a Professor of Electrical and Computer Engineering at The Univ of Texas at El Paso, El Paso. He was previously on the faculty of the Center for Microelectronics Research and the Dept. of Computer Science and Engineering at the University of South Florida, Tampa (1988- Aug 1998), where he continues to hold a courtesy appointment as professor. His research interests include VLSI design and hardware algorithms, computer architecture and parallel processing.