A Case-Based Recommender for Task Assignment in Heterogeneous Computing Systems

S. Ghanbari[1] **M. R. Meybodi**[1] **K. Badie**[2]

[1]*Computer Engineering Department*
Amirkabir University
Tehran Iran

[2]*IT Research Faculty*
Iran Telecommunication Research Center
Tehran, Iran

Abstract

Case-based reasoning (CBR) is a knowledge-based problem-solving technique, which is based on reuse of previous experiences. In this paper we propose a new model for static task assignment in heterogeneous computing systems. The proposed model is a combination of the case based reasoning and the learning automata model. In this new model a learning automata model is used as adaptation mechanism which adapts previously experienced cases to the problem to be solved. The objective of the proposed model is to reduce the number of iterations required to find a semi-optimum solution. The application is modeled as a set of independent tasks and the heterogeneous computing system is modeled as a network of machines. Using computer simulation, it is shown that the combined model outperforms the model that only uses learning automata.

1. Introduction

Mixed-machine heterogeneous computing (HC) environments utilize a distributed suite of different high-performance machines, interconnected with high-speed links, to perform different computationally intensive applications that have diverse computational requirements. The matching of the set of tasks to machines and scheduling the execution order of these tasks is referred to as mapping. The general problem of optimally mapping tasks to machines in an HC suite has been known to be NP-complete[7].

Metatask is defined as a collection of independent tasks with no inter-task data dependencies. Metatasks occur in many situations. For example, all of the jobs submitted to a supercomputer center by different users would constitute a metatask. Another example of a metatask would be a group of image processing applications all operating on different images[5]. The mapping of the Metatasks is being performed statically (i.e., off-line, or in a predictive manner). The goal of this mapping is to minimize the total execution time of the metatask, which is referred to as makespan.

In the literature, varieties of mathematical formulations have been developed for the task assignment problem. Approaches to the problem based on graph theoretic techniques, simulated annealing[9], A* state space search[10], and genetic techniques[8] have been proposed. In [1], a learning automata model has been proposed, where the key feature of this model is its ability to optimize multiple cost metrics. The major problem with all the mentioned approaches above is that for each task assignment, they build the solution from the scratch and do not use their past experiences. It should be noted that approaches like GA and other evolutionary computing systems, despite of their ability in finding the optimum structures for problem solving, are not capable of handling directly the experience management issue.

Case-based reasoning is an approach to use the previous experiences of solving the problem in someway to solve the current problem. This can be achieved mainly through adapting the solutions of those previous experiences which are similar to the current problem.

This paper presents a novel model for task assignment over a set of idle processors in a heterogeneous computing environment by means of case-based reasoning. Adaptation of stored cases and searching for new solutions are done by using learning automata. Using case-based reasoning approach enables the model to incorporate its former cases to find a suitable mapping in a relatively short time. The proposed model is compared to the model reported in [1].

The rest of the paper is organized as follows. Section 2 introduces related works. Briefly reviewing the discussed HC system model and the learning automata concept, section 3 introduces the learning automata model used for adaptation. In section 4, the proposed case-based reasoning model is introduced. Section 5 discusses experiments and results. The last section provides conclusions.

2. Related Works

Case-based reasoning (CBR) is a knowledge-based problem-solving technique that is based on reuse of previous experience [4]. Unlike traditional knowledge-based techniques, which solve problems from scratch by reasoning with general knowledge, CBR focuses on specific problem-solving experience captured in cases that are collected in a case base. New problems are solved by retrieving cases that deal with previous similar problems. The solutions recorded in those similar cases are then adapted to become a solution for the new problem (this process is known as adaptation). Thus, the two basic hypotheses of CBR are similar problems have similar solutions, and reuse is more feasible than problem solving from scratch.

In recent years, case-based reasoning(CBR) which is a sort of analogical learning have shown high capabilities in different areas such as decision making, prediction, diagnosis, planning, quality/process control, decision support and information retrieval. However, almost no CBR applications are reported for task assignment and scheduling in distributed computing systems up to the composition of this paper. Other CBR applications in distributed computing are reported, for example in [16] a case-based expert system is reported which helps an organization assess its computing alternatives.

On the other hand, research on task assignment for scientific computations on homogeneous as well as heterogeneous systems has been extensively investigated in the literature [12][13]. There are variety of works done in this field such as: *OLB*(Opportunistic Load Balancing), *MET* (*Minimum Execution Time*)[14], *MCT* (*Minimum Completion Time*)[14], *Min_min*[14], *Max_min* [14],GA[15],A*[10], Learning automata[1].
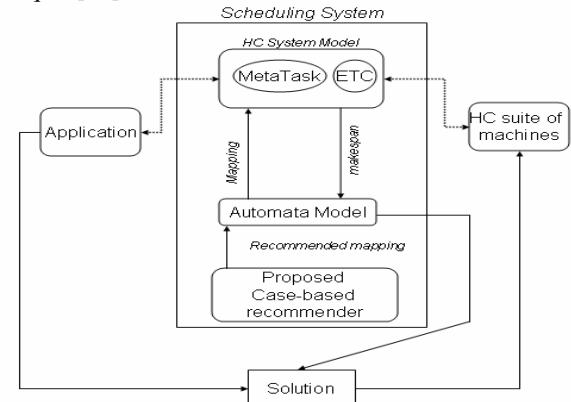
3. Background

3.1. HC System Model

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 that are used to execute the application. The scheduling system consists of the proposed case-based recommender, HC system model and the learning automata model. These are used by the scheduler to assign the subtasks to different machines.

In the proposed model, some assumptions are made. Firstly, the metatask is assumed to be decomposed into multiple independent tasks. Secondly, the HC system is

assumed to consist of a set of heterogeneous machines, which communicate by means of an underlying interconnection network. Thirdly, 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[11].

Figure 1. Model of the proposed framework

The estimate of the expected execution time for each task on each machine is known prior to execution and contained within a $\tau \times \mu$ *ETC* (Expected Time to Compute) *matrix*. One row of the ETC matrix contains the estimated execution times for a given task on each machine. Similarly, one column of the *ETC* matrix consists of the estimated execution times of a given machine for each task in the metatask. Thus, for an arbitrary task s_i and an arbitrary machine m_j , $ETC(s_i m_j)$ is the estimated execution time of s_i on m_j. The $ETC(s_i, m_j)$ entry could be assumed to include the time to move the executables and data associated with task s_i from their known source to machine m_j.

3.2. The Learning Automata Model

This section deals with the concept of learning automata first and then proceeds with a description of the learning automata model used as the adaptation mechanism.
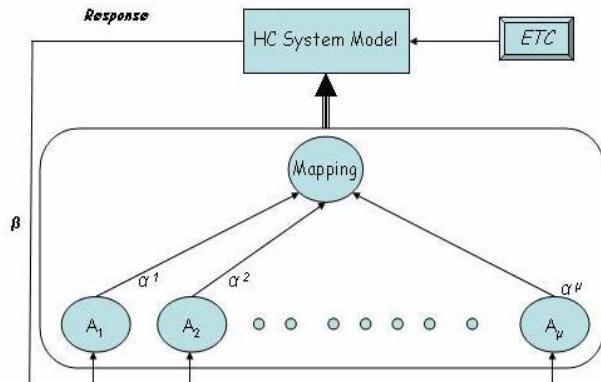
Learning Automata are adaptive decision-making devices operating on unknown random environments. A Learning Automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in

selection of the optimal action. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA) [3]. In the following, the variable structure learning automata is described.

A VSLA is a quintuple $< \alpha, \beta, p, T(\alpha,\beta,p) >$, where α, β, p are an action set with s actions, an environment response set and the probability set p containing s probabilities, each being the probability of performing every action in the current internal automaton state, respectively. The function of T is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. Assume $\beta=\{0,1\}$. A general linear schema for updating action probabilities can be represented as follows. Let action i be performed then

If $\beta(n)=0$,

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$

$$p_j(n+1) = (1-a)p_j(n) \qquad \forall j \quad j \neq i$$

If $\beta(n)=1$,

$$p_i(n+1) = (1-b)p_i(n)$$

$$p_j(n+1) = (b/s-1) + (1-b)p_j(n) \quad \forall j \quad j \neq i$$

where a and b are reward and penalty parameters. When $a=b$, the automaton is called L_{RP}. If $0<a<<b<1$ the automaton is called $L_{P\varepsilon R}$, and if $0<b<<a<1$, the automaton is called $L_{R\varepsilon P}$. For more Information about learning automata the reader may refer to [3].

Figure 2. Learning automata model

```
while(true)
Begin
    LAᵢ selects action for all 1≤ i ≤ τ
    Evaluate the makespan
    If  Tμ (n)<Tμ (n-1) then LAᵢ.Signal(0) for all 1≤ i
    ≤ τ
    Else LAᵢ.Signal(1) for 1≤ i ≤ τ
If no change in makespan occurs for 150 consecutive
iterations or 10000 iterations is over then exit
End
```

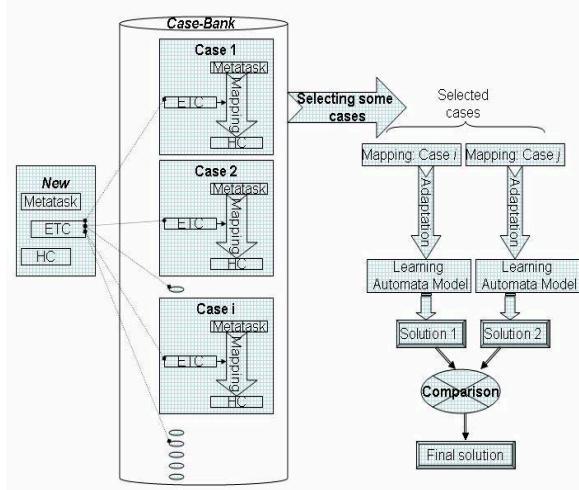**Figure 3. General procedure for learning automata model**

As mentioned before, learning automata is used as the case adaptation mechanism in the proposed task assignment problem. Figure 3 shows the schematic of the learning automata model. The model is constructed by associating every task s_i, $1 \leq i \leq \tau$, in the metatask with a variable structure learning automaton $(\alpha(i),\beta(i),A(i))$.

Since the tasks can be assigned to any of the μ machines, the action set of all learning automata are identical, Therefore, for any task s_i, $1 \leq i \leq \tau$, $\alpha(i)=m_1,m_2,...,m_\mu$. It is assumed that the environment is a P-model so the input set for each learning automaton $A(i)$ is $\{0,1\}$, where $\beta(i)$ *equal* to 0 indicates a favorable, and *equal* to 1 indicates an unfavorable response. If the value of makespan at iteration n is less than the value of makespan at iteration $n-1$, then the input to each automaton is favorable; otherwise it is unfavorable. The general procedure for learning automata model is shown in figure 4.

4. The Case-Based Reasoning Model

In CBR, a set of cases stored in a case base is the primary source of knowledge. Cases represent specific experience in a problem-solving domain, rather than general rules. The main activities when solving problems with cases are described in the case-based reasoning cycle[4]. This cycle proposes the four steps: retrieve, reuse, revise, and retain. First, the new problem to be solved must be formally described as a case (new case). A case consists of three parts: Problem situation including the initial conditions and the goal, solution and the performance. Then, a case that is similar to the current problem is retrieved from the case base. In the next step, the solution contained in this retrieved case is reused to solve the new problem; i.e., the solution is adapted in order to come to a solution of the current problem. Thereby, a new solution is obtained and presented to the user, who can verify and possibly revise the solution. The revised case (or the experience gained during the case-based problem solving process) is then retained for future problem solving; e.g., the case can be stored in the case base. The last step realizes the learning phase of a CBR application. It should be noted that CBR does not offer a definite solution but proposes hypotheses and ideas to go through the solution space. There are various approaches for solution adaptation where transformation and combination are among well-known techniques[2]. In the proposed model, the adaptation, which is performed by means of learning automata, is a hybrid combinational transformational adaptation.

Figure 5 shows the schematic of the proposed case based recommender. The three parts of a case are *ETC*, Mapping and makespan; which is denoted by a 3-tuple (*ETC*,π,*C*). Similarity criterion is defined as the similarity between two matrixes: *ETC* of the newly arrived metatask and *ETC*s of the stored cases, which is evaluated as the Euclidian distance of the two matrixes. In general, calculating Euclidian distance of two matrixes of size $\tau \times \mu$ has the time complexity of order $O(\tau \times \mu)$. To reduce computation cost of seeking in the case base, we define similarity criterion as the Euclidian distance of the first columns of two matrixes. Due to the fact that tasks in a metatask are arbitrary, without loss of generality, it is assumed that an *ETC* matrix is sorted on the first column on a descending bases, i.e *ETC[1][i]≥ETC[1][j]* for all *1≤ i < j ≤ τ.*, in order to make the Euclidian distance meaningful as a similarity criterion.

Figure 4. The proposed case-based recommender model

After retrieval of a set of cases, adaptation phase starts. Adaptation is the process of transforming the map of retrieved cases so as to find a mapping for the newly arrived metatask. This process is done by means of the learning automata model described in previous section.

Let Θ be the set of selected cases. To build up the adaptation model for each case $\varphi_k, 1 \le k \le |\Theta|$, every task s_i is associated with a variable structure learning automata $A^{\varphi_k}(i)$ and it is set biased to the mapped machine m_j where $m_j = \pi^{\varphi_k}(i)$. It is done by initializing the automaton action probability vector near one for the action corresponding to m_j. For a biased automaton, the rate of penalizing is set to a value much greater than the rate of rewarding. It helps a wrongly

biased automaton correct itself rapidly. On the other hand, if $\pi^{\varphi_k}(i)$ is not available (i.e. in the retrieved case no corresponding machine is defined for s_i), action probabilities of the associated automaton are set equal (i.e. left unbiased). Learning automata models for each φ_k start iterating in the way explained in the previous section until one of the termination conditions is fulfilled.

The mappings adapted by each learning automata model are used for the next step, proposing a solution. Mappings derived from Θ are compared and the mapping with the minimum makespan is selected as the final solution. If the final solution has a major difference with its original case, it is stored as a new case.

Due to starting from a point near the solution in the solution space, the proposed model reduces number of iterations. In the next section, our assertion is investigated through some experiments.

5. Experimental Results

5.1. Simulation Environment

For the simulation studies, characteristics of the *ETC* matrices were varied in an attempt to represent a range of possible HC environments. The *ETC* matrices used were generated using the method introduced in [6].

To generate different mapping scenarios, the characteristics of the ETC matrix were varied based on several different methods from[5]. The amount of variance among the execution times of tasks in the metatask for a given machine is defined as *task heterogeneity*. *Machine heterogeneity* represents the variation that is possible among the execution times for a given task across all the machines.

To further vary the characteristics of the *ETC* matrices in an attempt to capture more aspects of realistic mapping situations, different *ETC* matrix consistencies were used. An *ETC* matrix is said to be *consistent* if whenever a machine m_j executes any task s_i faster than machine m_k, then machine m_j executes all tasks faster than machine m_k [6]. In contrast, *inconsistent* matrices characterize the situation where machine m_j may be faster than machine m_k for some tasks and slower for others.

Eight different cases for ETC matrix characteristics are used in this study: high or low task heterogeneity, high or low machine heterogeneity, and one type of consistencies; consistent or inconsistent. The experiments are done over 16 machines and 512 tasks.

5.2. Experiment No.1

In this experiment the impact of combination of case based reasoning and learning automata over iteration and makespan is examined. The iteration needed to resolve a mapping is compared with the model that uses learning automata only, where the learning automata model uses reinforcement algorithm $L_{R\varepsilon P}$ with $a=0.1$ and $b=0.01$.

The reinforcement scheme of the learning automata model used for adaptation is $L_{P\varepsilon R}$ with $a=0.05$ and $b=0.1$. For conducting the experiment, the case bank is initially filled with 100 cases, and each experiment is done for 20 runs.

As shown in table 1, in the worst case i.e. experiment No.7, at least 10 percent reduction in the number of iterations is gained and it peaks to 17 percent in experiment No.2.

Table 1. Iteration enhancement for each model (Case-base of size 100)

Experiment	Machine Heterogeneity	Task Heterogeneity	Consistent	Makespan Increase	Iteration Enhancement
1	Low	Low	Yes	4.8%	12.7%
2	Low	High	Yes	3.8%	16.8%
3	High	Low	Yes	3.6%	13.1%
4	High	High	Yes	4.2%	12.6%
5	Low	Low	No	5%	12.5%
6	Low	High	No	3.5%	11.7%
7	High	Low	No	4.03%	10.35%
8	High	High	No	4.01%	11.82%

5.3. Experiment No.2

As mentioned in the previous section, to adapt a case to the problem, a learning automaton is assigned to each task, and set biased to the mapping recorded in the case. This may raise a question: setting LA biased to an action, can it escape from it? The answer is shown in table 2. The experiment is made for 10 cases with low task and machine heterogeneity. It is seen that 95 out of 512 assignments changed after adaptation. This shows that wrongly biased automata can correct themselves.
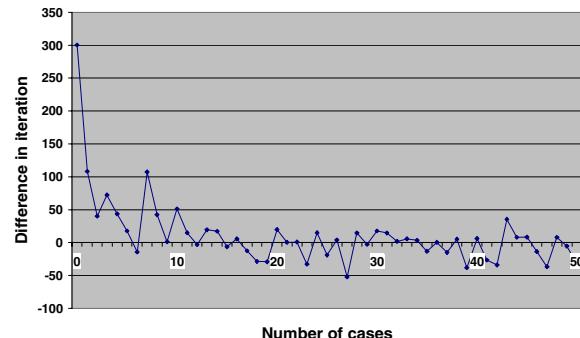
Table 2. Assignments change after adaptation

Case No.	Iterations reduced	changed	unchanged
1	53	96	416
2	168	92	420
3	-32	101	411
4	136	100	412
5	189	93	419
6	151	110	402
7	9	97	415
8	214	85	427
9	-23	92	420
10	26	85	427

5.4. Experiment No.3

This experiment shows that by gradually filling the case base with new cases, the system improves its performance over time. The experiment starts with a case base with only one case and continues until the case base contains 50 cases. This experiment is performed for low task and machine heterogeneity and results reported are the average of 20 runs. Figure 6 shows the difference between the number of iterations made for the retrieved case and the number of iterations made for the newly adapted case. A rapid decline in the difference indicates that the proposed model enhances its performance overtime.

Figure 5. Iteration reduction over time

6. Conclusion

In this paper, a case-based reasoning model for task assignment in heterogeneous computing system based on learning automata is proposed. Previously stored cases speed up the process of finding a semi-optimized mapping from metatask to HC. The main idea is to use mappings of experienced cases to find a solution for newly introduced cases. By conducting some experiments we showed that the proposed model which is a combination of case based reasoning and a learning automata model outperforms the model that only uses learning automata. Experiments showed 10% reduction

in the number of iterations needed to find the solution, which is significantly cost-effective.

7. References

[1]. R. D. Venkataramana and N. Ranganathan, "Multiple Cost Optimization for Task Assignment in Heterogeneous Computing Systems Using Learning Automata," *IEEE 8th Heterogeneous Computing Workshop*, 1999, pp.137.

[2]. W. Wilk, R. Bergmann, "Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving," *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Vol. 1416, 1998, pp. 497-506.

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

[4]. R. Bergman, "Engineering Applications of Case-Based Reasoning," *Journal of Engineering Applications of Artificial Intelligence*, Vol. 12, 1999, pp.805.

[5]. T. D. Braun, H. J. Siegel, and N. Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," *Journal of Parallel and Distributed Computing*, Vol.61, 2001, pp. 810-837.

[6]. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel Distributed Computing*, Vol.59, 1999, pp.107-121.

[7]. D. Fernandez-Baca, "Allocating Modules to Processors in a Distributed System," *IEEE Transaction on Software Engineering*, Vol.15, 1989, pp.1427-1436.

[8]. H. Singh and A. Youssef, "Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms," *5th IEEE Heterogeneous Computing Workshop*, 1996, pp.86-97.

[9]. M. Coli and P. Palazzari, "Real Time Pipelined System Design through Simulated Annealing," *Journal of Systems Architecture*, Vol.42, 1996, pp.465-475.

[10]. K. Chow and B. Liu, "On Mapping Signal Processing Algorithms to a Heterogeneous Multiprocessor System," *International Conference on Acoustics, Speech, and Signal Processing*, Vol.3, 1991, pp.1585-1588.

[11]. A. A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. L. Wang, "Heterogeneous Computing: Challenges and Opportunities," *IEEE Computing*, Vol.26, 1993, pp.18-27.

[12]. T. Cavasant and J. Kuh, "A Taxonomy of Scheduling in General-Purpose Distributed Computing System," *IEEE Transactions on software Engineering*, Vol.12, 1996, pp.662-675.

[13]. D. Gupta and P. Bepari, "Load Sharing in Distributed Systems," National Workshop on Distributed Computing, 1999.

[14]. R. Armstrong, D. Hensgen, and T. Kidd, "The Relative Performance of Various Mapping Algorithms is independent of sizable variances in run-time predictions," *7th IEEE Heterogeneous Computing Workshop*, 1998, pp. 79-87.

[15]. L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-based Approach," *Journal of Parallel Distributed Computing*, Vol. 47, 1997, pp.1-15.

[16]. R. S. Freeman and R. DiGiorgio, "Assessing Alternative Technologies for the Cost-Effective Computation of Derivatives," *Applied Artificial Intelligence*, 1997, pp.491-503.