11-14-1994

Multiprocessor Document Allocation: a Neural Network Approach

Abdulaziz Sultan Al-Sehibani
Syracuse University

Kishan Mehrotra
Syracuse University, mehrotra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Sanjay Ranka
Syracuse University

Multiprocessor Document Allocation: a Neural Network Approach

Abdulaziz Al-Sehibani
Kishan Mehrotra, Chilukuri Mohan, Sanjay Ranka

School of Computer and Information Science
4-116 Center for Science and Technology
Syracuse University
Syracuse, NY 13244-4100

email: asalsehi/kishan/mohan/ranka@top.cis.syr.edu
tel: (315) 443-2368

November 14, 1994

Abstract

We consider the problem of distributing the documents to a given set of processors so that the load on each processor is as equal as possible and the amount of communication is as small as possible. This is an NP-Complete problem. We apply continuous as well as discrete Hopfield neural networks to obtain suboptimal solutions for the problem. These networks perform better than a genetic algorithm for this task proposed by Frieder et al. [4]; in particular, the continuous Hopfield network performs extremely well.

Keywords: Document Allocation, Hopfield Network, Multiprocessor, Information Retrieval

1 Introduction

Multiprocessor systems with distributed memory are powerful tools for information retrieval. These systems can support multiple queries simultaneously, with various documents distributed among the processors. Efficient exploitation of parallelism in such systems requires fast access to documents. In this paper, we address the task of allocating documents onto processors to achieve better performance and lower operating costs. A poor allocation can cause, for example, heavy traffic on the communication network degrading the performance drastically. On the other hand, a clever allocation can reduce the unnecessary communication, causing the network to have a faster response time.

Our objective is to minimize the communication cost among processors while maintaining an even load of documents on processors and maximizing the throughput of the retrieval system. This is done by minimizing the "average cluster distance", which is the average distance between different processors containing documents which belong to the same cluster, and by making the processors evenly loaded. Therefore, the multiprocessor document allocation problem is a constrained nonlinear optimization problem, where the cost (energy) function to be minimized is the average cluster distance, subject to the conditions that the processors should be evenly loaded, and that every document should appear exactly once on only one processor.

Frieder and Siegelmann [4] have proved that the multiprocessor document allocation problem is NP-complete. Therefore, the determination of an optimal allocation of documents is not computationally feasible, and heuristic approaches should be applied. Frieder and Siegelmann used a genetic algorithm to find a good suboptimal allocations for small sets of documents. In this paper, we show that excellent suboptimal solutions can be obtained for the multiprocessor document allocation problem using discrete and continuous Hopfield neural networks. The Hopfield networks give equally good or better solutions using much less computational resources.

The rest of this paper is organized as follows. Section 2 describes the neural network model for solving the multiprocessor document allocation problem using the discrete Hopfield network, while Section 3 describes the continuous Hopfield model. Performance of these two models is evaluated in Section 4. Conclusions are presented in Section 5.

2 The Discrete Hopfield Network Model

We assume that a set of d documents, $D = \{d_i : 1 \leq i \leq d\}$, is partitioned into C clusters, with $m(q)$ being the number of documents in the qth cluster, $q = 1, 2, \ldots .C$. A multiprocessor system with P nodes is given. The communication cost of sending a unit data from processor i to processor j is given by c_{ij} for $i, j = 1, \ldots, P$. We consider a neural network approach to solve this problem.

The first neural network model is based on the discrete Hopfield network. Let $x_{i,j,q}$ be a decision variable that takes the value 1 if the document j of the qth cluster is assigned to the ith processor, and 0 otherwise. Then the neural network for the multiprocessor document allocation problem can be represented as a three-dimensional array of neurons. The first dimension represents the processors, the second dimension represents the documents, and the third dimension represents the clusters. The network has $P \times C \times d$ nodes.

Tagliarini et al. [12] have used a similar architecture to solve the weapon-to-target assign-

ment problem.

Using the above representation for document allocation, the average cluster distance is proportional to

$$F = \frac{1}{2}\sum_{q=1}^{C}\sum_{i=1}^{P}\sum_{i'=1}^{P}\sum_{j=1}^{m(q)}\sum_{j'=1}^{m(q)} x_{i,j,q}x_{i',j',q}c_{i,i'}.$$

This is minimized subject to the constraint that all processors receive equal load, i.e.,

$$\sum_{q=1}^{C}\sum_{j=1}^{m(q)} x_{i,j,q} = \frac{d}{P}, \text{ for } i = 1,\ldots,P,$$

and each document is assigned to one and only one processor, i.e.,

$$\sum_{i=1}^{P} x_{i,j,q} = 1, \text{ for all } j \text{ and } q.$$

Therefore, the function to be minimized can be written as

$$E = F + k_1\sum_{i=1}^{P}(\frac{d}{P} - \sum_{q=1}^{C}\sum_{j=1}^{m(q)} x_{i,j,q})^2 + k_2\sum_{q=1}^{C}\sum_{j=1}^{m(q)}(1 - \sum_{i=1}^{P} x_{i,j,q})^2$$

where k_1, k_2 are positive penalty factors that are used to penalize violations of the constraints. The above energy function can be rearranged in terms of its quadratic, linear, and constant components to obtain the connection weights between nodes.

Assume asynchronous update rule for the nodes of the network, and let node $x_{i_1j_1q_1}$ be the node selected for updating at time t. Then the change of energy $\Delta E(t) = E(t+1) - E(t)$ is given by

$$\begin{aligned}\Delta E(t) \;=\; &\Bigg[\frac{1}{2}(c_{i_1i_1} + 2\sum_{\substack{i'=1\\ i'\neq i_1}}^{P}\sum_{\substack{j'=1\\ j'\neq j_1}}^{m(q1)} x_{i',j',q_1}c_{i_1i'}) + k_1(1 + 2\sum_{\substack{q'=1\\ q'\neq q_1}}^{C}\sum_{\substack{j'=1\\ j'\neq j_1}}^{m(q')} x_{i_1,j',q'}) + \\ &k_2(1 + 2\sum_{\substack{i'=1\\ i'\neq i_1}}^{P} x_{i',j_1,q_1}) - 2k_1\frac{d}{P} - 2k_2\Bigg]\Delta x_{i_1,j_1,q_1}\end{aligned}$$

For convenience, we will write the above equation as $\Delta E = (net_{i_1j_1q_1})\Delta x_{i_1j_1q_1}$. The selected node x_{i_1,j_1,q_1} will change state if and only if $\Delta E < 0$. Assuming that $x_{i_1,j_1,q_1}(t) = 0$ and $x_{i_1,j_1,q_1}(t+1) = 1$, then the node will change its state if and only if $(net_{i_1j_1q_1}) < 0$. Likewise, if $x_{i_1,j_1,q_1}(t) = 1$, and $x_{i_1,j_1,q_1}(t+1) = 0$ then $\Delta x_{i_1j_1q_1} = -1$, and the node will change its state if and only if $(net_{i_1,j_1,q_1}) > 0$.

The nodes on the same cluster plane (the nodes with the same cluster index q) are connected by symmetric connections with weight $w_{x_{i,j,q},x_{i'j'q}} = c_{ii'}$, the nodes on the same processor plane (nodes with the same processor index i) are connected by symmetric connections with weight $w_{x_{i,j,q},x_{i,j',q'}} = 2k_1$, and the nodes with the same cluster and document indices, q and j respectively, are connected by symmetric connections with weight $w_{x_{i,j,q},x_{i',j,q}} = 2k_2$. Every node has an external input $I_{ijq} = -2k_1\frac{d}{P} + k_1 - k_2$.

The following algorithm simulates computation using the above described neural network:

```
REPEAT
  initialize all nodes randomly;
  while there is a change in state do
        for each node do
            if (ΔE < 0) then change node state;
        end for;
  end while;
  if the solution is feasible
        then calculate average communication distance
UNTIL the number of trials = MAX-NUM;
Compute the average cost over all the trials.
```

3 The Continuous Hopfield Neural Network Model

The previous approach has the disadvantage of requiring a very large network because the number of neurons increases linearly with the number of documents. An alternative approach is to make the number of nodes proportional to the number of clusters. For this approach, we reformulate the neural network as a two dimensional array of $C \times P$ neurons.The (ij)th node of this array represents the fraction of the number of documents of the cluster i allocated to the jth processor. This fraction is denoted by x_{ij}. This approach results in a continuous Hopfield model, because x_{ij}'s take values in the continuous range from 0 to 1; $0 \leq x_{ij} \leq 1$; $\forall i, j$.

When this neural network reaches its stable state, the outputs of the neurons are interpreted as proportions of documents of every cluster to be allocated onto the different processors.

We formulate the problem as an energy function to be minimized based on the average cluster distance, subject to the conditions that each processor receives equal load and each document is assigned to one and only one processor. From the energy function so obtained, we will derive the connection weights and external inputs of the neural network.

3.1 The Energy Function

The jth processor receives $\sum_{i=1}^{C} x_{ij} m(i)$ documents and the kth processor receives $\sum_{i=1}^{C} x_{ik} m(i)$ documents. Therefore, the average cluster distance is proportional to

$$F = \frac{1}{2} \sum_{i=1}^{C} \sum_{j=1}^{P} \sum_{k=1}^{P} x_{ij} x_{ik} m^2(i) c_{jk}.$$

This is the function that we wish to minimize, subject to the constraint that processors have roughly equal load, i.e., minimizing

$$\left(\sum_{i=1}^{C} x_{ij} m(i) - \frac{d}{P} \right)^2,$$

for every processor, and that each document in each cluster is assigned to some processor, i.e.,

$$\left(\sum_{j=1}^{P} x_{ij} - 1\right)^2$$

is minimized for every cluster.

The total energy function to be minimized can hence be written as :

$$E = F + k_1 \sum_{j=1}^{P} (\frac{d}{P} - \sum_{i=1}^{C} x_{ij} m(i))^2 + k_2 \sum_{i=1}^{C} (1 - \sum_{j=1}^{P} x_{ij})^2$$

Separating the quadratic, linear, and constant components,

$$E = F + k_1 \sum_{j=1}^{P} ((\sum_{i=1}^{C} x_{ij} m(i))^2 - 2 \sum_{i=1}^{C} x_{ij} m(i) + (\frac{d}{P})^2) + k_2 \sum_{i=1}^{C} ((\sum_{j=1}^{P} x_{ij})^2 - 2 \sum_{j=1}^{P} x_{ij} + 1). \quad (1)$$

After some mathematical manipulation, equation (1) can be expanded as

$$E = F + k_1 \sum_{j=1}^{P} \sum_{i=1}^{C} \sum_{l=1}^{C} x_{ij} x_{lj} m(i) m(l) - 2k_1 \frac{d}{P} \sum_{j=1}^{P} \sum_{i=1}^{C} x_{ij} m(i) + k_1 \sum_{j=1}^{P} (\frac{d}{P})^2 + k_2 \sum_{i=1}^{C} \sum_{j=1}^{P} \sum_{k=1}^{P} x_{ij} x_{ik} - 2k_2 \sum_{i=1}^{C} \sum_{j=1}^{P} x_{ij} + k_2 \sum_{i=1}^{C} 1. \quad (2)$$

We use a sigmoidal neuron activation function $x_{ij} = g(u_{ij})$ such that

$$g(u_{ij}) = 0.5(1 + \tanh(\lambda u_{ij}))$$

where u_{ij} denotes the net input (activation level) of neuron x_{ij}, and λ is the gain parameter which controls the gradient magnitude of the transfer function.

The network will have no inherent loss terms, and its dynamics are governed by the following equation

$$\frac{\partial u_{ij}}{\partial t} = -\frac{\partial E}{\partial x_{ij}} = -\left(\sum_{k=1}^{P} x_{ik} m(i)^2 C_{jk} + 2k_1 \sum_{l=1}^{C} x_{lj} m(i) m(l) + 2k_2 \sum_{k=1}^{P} x_{ik} - 2k_1 \frac{d}{P} m(i) - 2k_2\right) \quad (3)$$

Additionally,

$$\frac{\partial u_{ij}}{\partial t} = \sum_{l=1}^{C} \sum_{k=1}^{P} T_{ij,lk} x_{lk} + I_{ij} \quad (4)$$

where Equation 3 is used to obtain the connection weights $T_{ij,lk}$ between neurons x_{ij}, x_{lk}, and the external input I_{ij} to x_{ij} as :

$$T_{ij,lk} = -m(i)^2 C_{jk} \delta_{il} - 2k_1 m(i) m(l) \delta_{jk} - 2k_2 \delta_{il}$$

$$I_{ij} = 2k_1 \frac{d}{P} m(i) + 2k_2$$

where $\delta_{ij} = 1$ if and only if $i = j$, and $\delta_{ij} = 0$ otherwise.

3.2 Minimization of the Energy Function

Using the standard argument (see, for example, Takefuji [13]) we show that any change in the network will lead to a decrease in the energy of the system. In view of Equation 4, it is easy to see that

$$\begin{aligned}\frac{\partial E}{\partial t} &= \sum_{i=1}^{C}\sum_{j=1}^{P}\frac{\partial E}{\partial x_{ij}}\frac{\partial u_{ij}}{\partial t} \\ &= -\sum_{i=1}^{C}\sum_{j=1}^{P} g\prime(u_{ij})\frac{\partial u_{ij}}{\partial t}\frac{\partial u_{ij}}{\partial t} \\ &= -\sum_{i=1}^{C}\sum_{j=1}^{P} g\prime(u_{ij})(\frac{\partial u_{ij}}{\partial t})^2\end{aligned}$$

since $g(u_{ij})$ is a monotonic function, hence $\frac{\partial E}{\partial t}$ is either negative or zero. In other words, the energy is non-increasing with any change in the network.

3.3 Network Parameters

The continuous Hopfield neural network described above is simulated for several values of the network parameters, e.g., the slope of the transfer function, penalty factors, k_1 and k_2 and document sets with different numbers of documents and different cluster sizes. Equation 3, which plays an important role in insuring that the dynamics of the network is satisfied, is solved numerically.

3.3.1 Effect of the Gradient Magnitude

When the gradient magnitude of the transfer function is high, the convergence is very slow, and in some experiments it did not converge within a prespecified time. On the other hand, if the gradient magnitude is low, the convergence is fast but the quality of results is not good. So, the gradient magnitude of the transfer function is treated as a time-varying parameter.

So we start with a transfer function of a high gradient magnitude and during the simulation we decrease the gradient magnitude as the number of iterations increases. This gave good results with fast convergence.

3.3.2 Effect of the Penalty Factors

We performed many experiments using different penalty factors, k_1 and k_2. Some of these parameters gave infeasible allocations, i.e., a large fraction of a cluster is not allocated onto any processor or some clusters are duplicated redundantly among many processors. The penalty factor, k_2, associated with feasibility was small in the experiments which gave infeasible results. So, to get feasible allocations, k_2 should be selected large enough to guarantee feasibility, but at the same time it should not be too large to minimize the negative effect of increasing the average distance or load balancing.

To get a feasible solution, the sum of the fractions of each cluster on the different processors should be 1. But trying to make the sum *exactly* 1 may slow down the convergence and

substantially increase the average distance. For this reason, if

$$|\sum_{j=1}^{P} x_{ij} - 1| \leq 0.1$$

for any cluster i, then we consider the solution to be feasible, otherwise it is considered as an infeasible solution.

3.4 Interpretation of the Final Outputs of Neurons

In acceptable solutions, we normalize the node outputs by dividing each x_{ij} by $\sum_j x_{ij}$. Multiplying this fraction by the number of documents in cluster i and rounding the result to the nearest integer, yields m_{ij}, the number of documents of cluster i that is allocated onto processor j. We add $m(i) - \sum_j m_{ij}$ documents to the processor that contains the largest fraction of cluster i. This addition has a negligible effect on load balancing.

4 Performance Evaluation

The performance measures used are:

- *load imbalance*: expressed as the standard deviation of variance of load among various processors.
- *average distance*: the average distance between documents of a certain cluster which are allocated onto different processors.

Performance depends on the network parameters, the initial states of the neurons and the underlying data set. For this reason, we performed many experiments using different penalty factors, different network initializations and different data sets.

We performed many iterations in each case. Some of these iterations may not converge to feasible results, e.g., one or more documents are not allocated onto any processor. So, for every one of these experiments, we have measured the *valid-rate*, which is the proportion of iterations that gave feasible results.

Since an optimal allocation is computationally intractable (NP-Complete problem), we have compared our results with those of the genetic algorithm proposed in [4] as well as with the results of random allocation of documents.

The two models (the discrete and continuous Hopfield models) were tested using two multiprocessor architectures, a 16-1 mesh and a 16-node hypercube. The communication cost between any two adjacent nodes (any two nodes connected by a direct link) in these architectures is assumed to be 1 unit, where the unit can be considered as transmission time for 1 data packet.

We tested the discrete model using small data sets (small sets of documents) which consists of 64, 128, and 256 documents partitioned into 8 clusters. We observed that as we increase the value of k_2 (the penalty factor associated with feasibility) we get higher values of valid-rate, i.e., more iterations giving feasible allocations. Also, as we increase the value of k_1 (the penalty factor associated with load balancing of documents), the standard deviation of the distribution of documents on processors gets smaller. By increasing k_1 and k_2, the average cluster distance

Table 1: Average diameters and average distances of all experiments.

	Average Diameter						Average Distance					
	Equal Clustering			Uneq. Clustering			Equal Clustering			Uneq. Clustering		
No. of Documents	64	128	256	64	128	256	64	128	256	64	128	256
Hop-Disc-Mesh	3.70	6.68	8.35	4.8	9.28	12.0	1.56	2.26	3.47	1.94	3.70	4.17
Hop-Disc-Hyper	2.16	2.125	2.66	3.35	3.78	4.0	1.09	0.99	1.17	1.88	1.83	1.94
Hop-Cont-Mesh	1.0	1.0	0.94	1.0	1.0	1.0	0.46	0.44	0.39	0.50	0.47	0.45
Hop-Cont-Hyper	1.0	1.25	1.25	1.0	1.0	1.0	0.58	0.57	0.56	0.58	0.53	0.52
Genetic-Mesh	5.38	12.25	13.625	8.25	11.38	13.38	2.58	4.93	5.20	3.69	4.60	4.97
Genetic-Hyper	3.0	3.88	4.0	3.0	3.63	3.88	1.90	2.0	2.05	1.80	1.93	2.0

increases, due to the reason that more weight is given to the constraints (feasibility and load balancing) and less emphasis to the main objective function. These observations are consistent with the expected behavior of the system.

We tested the continuous model with the the same document sets as well as with other larger data sets of 2560 and 25600 documents, partitioned into 8 clusters. In these experiments, we observed that for small values of k_2 some experiments did not converge to feasible solutions indicating that k_2 (the penalty factor associated with feasibility) should be large. On the other hand, when we increase k_1 (the penalty factor associated with load balancing) while maintaining the feasibility condition, the average distance of clusters increases, which means that there is a trade-off between load balancing and minimizing the average distance. Therefore, to keep the average distance small, k_1 should not be set to large values. As expected the results of both Hopfield networks are much better than random allocations (results not presented in this paper). In the next section, we focus on comparison with the genetic algorithms.

4.1 The Genetic Algorithm vs. Hopfield Models

We implemented the genetic algorithm proposed in [4] in order to compare its results to those we obtained using the two Hopfield models. As suggested in [4], we used two document distributions. In the first distribution, clusters contain equal numbers of documents, while in the second, 25 percent of the clusters contain 50 percent of the documents. We refer to the former case as "equal-distribution" and the latter as "non-equal-distribution".

We compared the performance of the genetic algorithm and Hopfield networks in terms of the average diameter as well as the average distance of clusters, and the CPU times needed to run these algorithms.

In the genetic algorithm, the mutation rate varied from 0.1 to 0.5 and the population size was proportional to the number of documents, i.e., it was set to 50, 100, and 200, for the document sets 64, 128, and 256, respectively.

In Figure 1 and Table 1, we see that the two Hopfield models are superior to the genetic algorithm in terms of both the average diameter and the average distance of clusters, especially for the 16-1 mesh. The continuous Hopfield model is superior to the discrete model with respect to both measures. The genetic algorithm was slightly better than the discrete Hopfield model with respect to the average diameter in the case of "non-equal-distribution" for the hypercube topology, but the difference between the performance of the two algorithms was very small.

Table 2: CPU Time of all experiments.

	CPU Time					
	Equal Clustering			Uneq. Clustering		
No. of Documents	64	128	256	64	128	256
Hop-Disc-Mesh	4.40	11.70	54.30	4.49	16.0	35.0
Hop-Disc-Hyper	4.80	12.89	85.04	4.10	12.89	34.37
Hop-Cont-Mesh	275.18	134.53	49.50	325.47	176.39	97.04
Hop-Cont-Hyper	178.56	131.05	49.13	324.89	206.69	96.31
Genetic-Mesh	139.30	215.95	1882.5	191.9	232.80	1680.20
Genetic-Hyper	33.90	209.35	1534.68	73.99	238.50	1819.02

On a SUN-4.1.3 Sparc workstation, the Hopfield models needed much less time to execute as it is presented in Table 2 and Figure 1((e) and (f)). The continuous Hopfield model becomes faster than the discrete as the number of documents gets larger.

5 Conclusions

In this paper, we have implemented two neural network approaches to solve the multiprocessor document allocation problem. The first approach is based on the discrete Hopfield model of neural networks where the number of neurons is proportional to the number of documents, the number of clusters, and the number of processors. The other approach is based on the continuous Hopfield model. In this model, we allocate fractions of clusters (and not individual documents) onto the processors. In other words, the size of the neural network is proportional to the number of clusters of documents (instead of the number of documents) and the number of processors. The final outputs of the neurons represent the fractions of clusters that are allocated onto processors, and should be interpreted accurately to know the number of documents of each cluster that are allocated onto different processors.

Performance of both approaches is superior to that of the genetic algorithm proposed by Frieder and Siegelmann, as well as to that of random allocations of documents. Another major advantage, especially for the continuous model, is fast convergence to a feasible solution for large sets of documents, provided that network parameters are selected properly.

References

[1] Chakraborty K., Mehrotra K., Mohan C., and Ranka S., "An Optimization Network for Solving a Set of Simultaneous Linear Equations," IEEE/INNS Intl. Joint Conf. on Neural Networks, Baltimore, June, 1992.

[2] Cichockl A. and Unbehauen R., "Neural Networks for Optimization and Signal Processing," John Wiley and Sons, 1993.

[3] Cornell D. and Yu P., "On Optimal Site Assignment for Relations in the Distributed Database Environment," IEEE Trans. on Software Engineering, Vol. 15, No. 8, Aug. 1989.

[4] Frieder O. and Siegelmann H., "The Allocation of Documents in Multiprocessor Information Retrieval Systems: An Application of Genetic Algorithms," Proc. of the 1991 IEEE Intl. Conf. on Systems, Man. and Cybernetics.

[5] Hecht-Nielsen R.,"Neurocomputing," Addison Wesley, 1990.

[6] Hertz J., Krogh A. and Palmer R., "Introduction to the Theory of Neural Computation," Addison Wesley, 1991.

[7] Hopfield J., "Neurons with graded response have collective computational properties like those of two-state neurons," Proc. of the National Academy of Sciences, Vol. 81, pp.3088-3092, May 1984.

[8] Kamoun F. and Ali M.,"Neural Networks for Shortest Path Computation and Routing in Computer Networks," IEEE Trans. on Neural Networks, Vol. 4, No. 6, Nov. 1993.

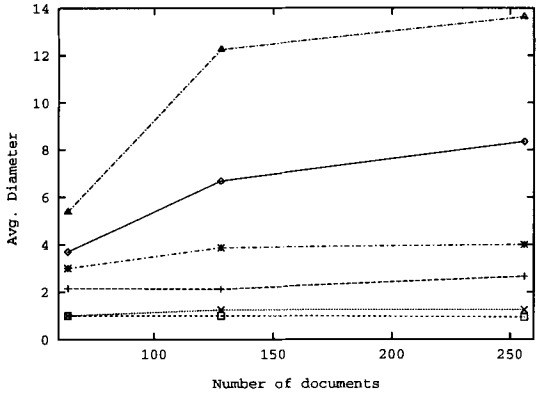
[9] Lu Y. and Thomborson C., "Gate Array Global Routing Using A Neural Network," Artificial Neural Networks in Engineering, Dagli, Kumar and Shin, editors, pp.985-990, Nov. 1991.

[10] Protzel W., Palumbo D. and Arras M., "Performance and Fault Tolerance of Neural Networks for Optimization," IEEE Trans. on Neural Networks, Vol. 4, No. 4, July 1993.

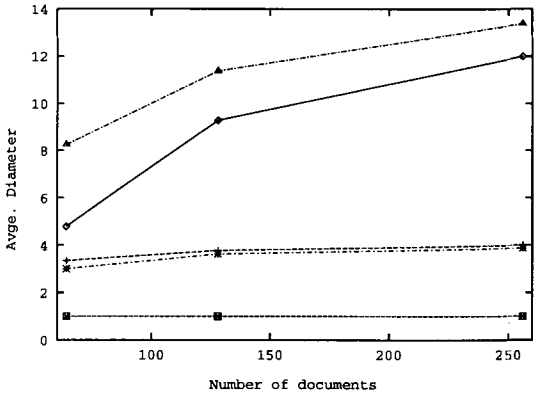
[11] Rotem D., Schloss A. and Segev A.," Data Allocation for Multidisk Databases," IEEE Trans. on Knowledge and Data Engineering., Vol. 5, No. 5, Oct. 1993.

[12] Tagliarini G., Christ F., and Page E., "Optimization Using Neural Networks," IEEE Trans. on Computers, Vol. 40, No. 12, Dec. 1991.

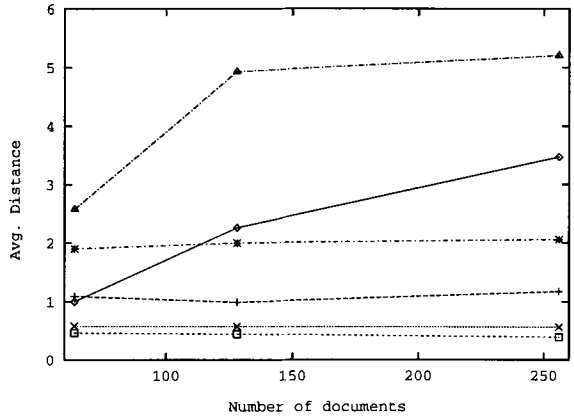
[13] Takefuji Y., "Neural Network Parallel Computer,"Kluwer Academic Publisher, Boston, 1992.



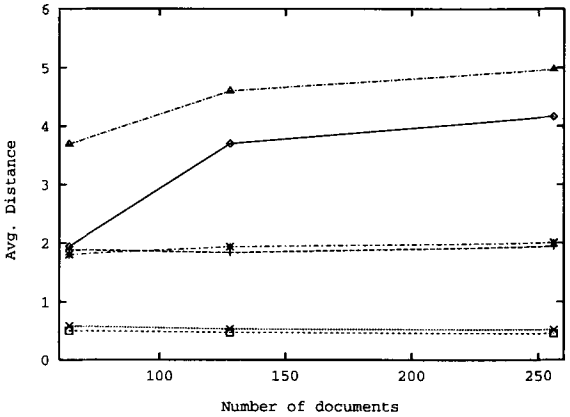
(a) Average Diameter (equal-distribution)



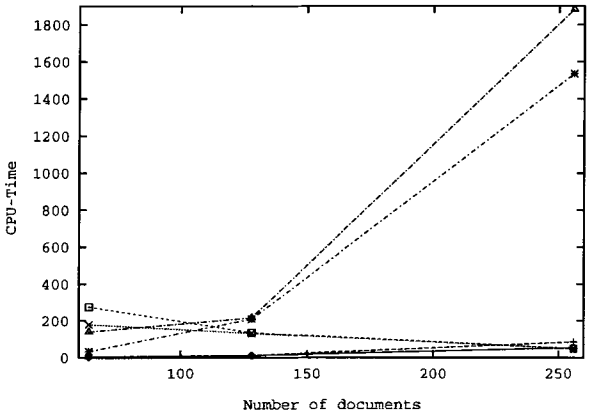
(b) Average Diameter (unequal-distribution)



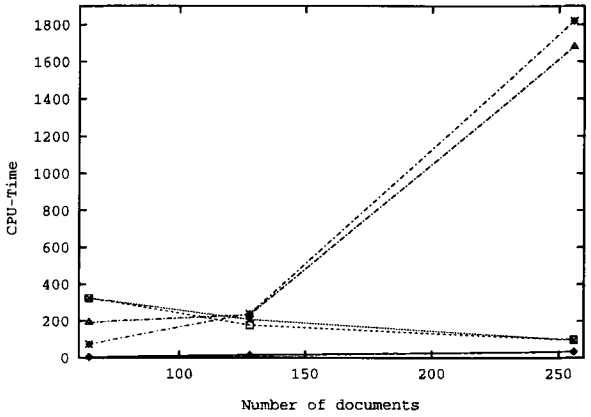
(c) Average Distance (equal-distribution)



(d) Average Distance (unequal-distribution)



(e) CPU-Time (equal-distribution)



(f) CPU-Time (unequal-distribution)

Figure 1. Performance of Hopfield models and the genetic algorithm.