Concise Papers

Multiprocessor Document Allocation: A Genetic Algorithm Approach

Ophir Frieder, *Senior Member, IEEE*, and Hava T. Siegelmann

Abstract—We formally define the Multiprocessor Document Allocation Problem (MDAP) and prove it to be computationally intractable (NP Complete). Once it is shown that MDAP is NP Complete, we describe a document allocation algorithm based on genetic algorithms. This algorithm assumes that the documents are clustered using any one of the many clustering techniques. We later show that our allocation algorithm probabilistically converges to a good solution. For a behavioral evaluation, we present sample experimental results.

Index Terms—Genetic algorithms, information systems, information retrieval, multiprocessor, parallel processing, data placement, data allocation.

1 Introduction

INFORMATION retrieval is the selection of documents that are potentially relevant to a user's information needs. Given the vast volume of data stored in modern information retrieval systems, searching the document database requires vast computational resources. To meet these computational demands, various researchers have developed parallel information retrieval systems. As efficient exploitation of parallelism demands fast access to the documents, data organization and placement significantly affect the total processing time. We describe and evaluate an algorithm that derives an allocation that supports efficient access to a clustered document collection.

Formally, the Multiprocessor Document Allocation Problem (**MDAP**) is defined as follows:

GIVEN:

- A distributed memory architecture with:
  - — Nodes (PEs): $X = \{X_i \mid 0 \le i \le n-1\}$;
  - — Communication cost: M_{ij}, $(0 \le i, j \le n-1)$;
- A clustered document domain with:
  - — Documents: $D = \{D_i \mid 0 \le i \le d-1\}$;
  - — Clusters: $C = \{C_i \mid 0 \le i \le c-1, C_i \subseteq D\}$;
- A real value bound: B.

DERIVE:

An allocation mapping

$$\mathcal{A} : D \mapsto X$$

of the documents to the processors that satisfies the following conditions:

1) Let X_i be a node. Define the number of documents mapped onto this node by the allocation $\mathcal{A}$ as

$$\mu_{\mathcal{A}}(X_i) = |\{D_j \in D \mid \mathcal{A}(D_j) = X_i\}|,$$

where, for all i $(0 \le i \le n-1)$, $\mu_{\mathcal{A}}(X_i) \le \lceil \frac{d}{n} \rceil$.

2) Let C_j be a cluster of documents. Define the diameter of this cluster under a given allocation $\mathcal{A}$ as:

$$\text{diameter}_{\mathcal{A}}(C_j) = \max\{M_{\mathcal{A}(D_k),\mathcal{A}(D_l)} \mid D_k, D_l \in C_j\}.$$

Then,

$$\sum_{j=0}^{c-1} diameter_{\mathcal{A}}(C_j) \le B.$$

THEOREM 1. MDAP is NP-Complete.

To prove that MDAP is complete in NP, we reduce the NP-Complete problem, Binary Quadratic Assignment Problem [5] to MDAP in polynomial time. Details are found in [11].

Since MDAP is NP-Complete, obtaining an optimal allocation of documents onto the nodes is not computationally feasible. The heuristic algorithm proposed here is based on genetic algorithms [7]. Related document mapping algorithms and multiprocessor information retrieval efforts are found in, for example [1], [2], [3], [4], [6], [8], [9], [10], [12].

ALGORITHM:

Initialization Phase:

1) Create a permutation matrix, $P_{i,j}$ $(0 \le i \le p-1, 0 \le j \le d-1)$. Every row of P, P_i, $(0 \le i \le p-1)$ is a complete permutation of all documents D_j, $(0 \le j \le d-1)$. For example, if $p = 3$ and $d = 6$, a possible permutation matrix is P.

P =

	0	1	2	3	4	5
0	1	0	2	5	3	4
1	0	2	4	1	3	5
2	4	5	3	2	1	0

2) Define the document to node mapping function $\mathcal{A}_i : D \to X$ for any given row of P, P_i, $(0 \le i \le p-1)$ as $\mathcal{A}_i(D_k) = j \bmod n$, where j is the index in row P_i of document D_k, $(0 \le k \le d-1)$. If $n = 3$, row P_0 implies that documents 0 through 5 are mapped to nodes 1, 0, 2, 1, 2, 0, respectively.

Reproduction Phase:

3) Given the mapping function $\mathcal{A}_i$ for a given row P_i, $(0 \le i \le p-1)$, determine the cluster diameter, $R_{i,j}$, $(0 \le j \le c-1)$ for each cluster association list array entry, C_j. $R_{ij} = Max\{M_{\mathcal{A}_i(D_k),\mathcal{A}_i(D_l)}$, where $0 \le k, l \le d-1$, and $D_k, D_l \in C_j\}$. If

- *O. Frieder is with the Faculties of Computer Science and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901. He is currently on leave from the Department of Computer Science of George Mason University. E-mail: ophir@cs.fit.edu.*
- *H.T. Siegelmann is with the Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel. E-mail: iehava@ie.technion.ac.il.*

Manuscript received 22 Feb. 1995.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number K97022.

M =

	0	**1**	**2**
0	0	2	4
1	2	0	1
2	4	1	0

C =

	0	**1**	**3**	**4**	**5**
1	0	2			

then R is

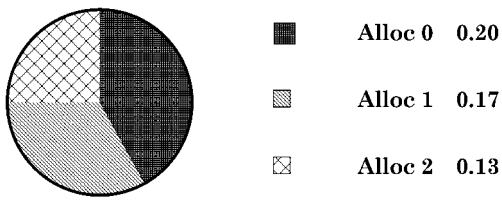
R =

	0	**1**
0	4	1
1	4	2
2	4	4

4) Define an evaluation function, E. This function measures the "goodness" of the allocation defined by a row P_i, $(0 \le i \le p-1)$, and the corresponding mapping function $\mathcal{A}_i$. In our case,

$$E(P_i) = \sum_{j=0}^{c-1} R_{ij} \; 0 \le i \le p-1.$$

5) Create a biased roulette. Compute the reciprocal of each $E(P_i)$, $(0 \le i \le p-1)$. Call them $E^{-1}(P_i)$. Bias the roulette proportionally to $E^{-1}(P_i)$. Assign each allocation an interval on the unit vector 0 to 1 based on the corresponding biased probability. In the above example, $E(P_0) = 5$, $E(P_1) = 6$, and $E(P_2) = 8$, resulting in the following roulette wheel.



Thus, permutations P_0, P_1, and P_2, are weighted at a probability of 0.40, 0.34, and 0.26, and are assigned the intervals [0.0, 0.40), [0.4, 0.74), [0.74, 1.00], respectively.

6) Replace the permutation matrix P. Randomly choose p numbers from within the interval [0.0, 1.0]. For each of the p random values obtained, copy the allocation permutation whose assigned interval corresponds to the random value generated into row P_i, $(0 \le i \le p-1)$. To insure the survival of successful document allocations (permutations), the lowest cost allocation is always kept. Therefore, if the permutation corresponding to the largest interval, say P_j, $(0 \le j \le p-1)$, is not selected within the first $p-1$ selections, P_j is assigned to row P_{p-1}. In the example, if 0.23, 0.92, and 0.36 were the random numbers obtained, then P would be

P =

	0	**1**	**2**	**3**	**4**	**5**
0	1	0	2	5	3	4
1	4	5	3	2	1	0
2	1	0	2	5	3	4

Crossover Phase:

7) While maintaining a copy of the lowest-cost permutation, say P_i', randomly pair up the rows in P. If p is odd, ignore the unpaired row. For each pair of rows in P, say A and B, randomly generate two integer values, i and j, such that $0 \le i \le j \le d-1$. Position-wise exchange A_i, A_{i+1}, A_{i+2}, ..., A_{j-1}, A_j, with B_i, B_{i+1}, B_{i+2}, ..., B_{j-1}, B_j, respectively within the two strings. Replace the highest cost permutation with P_i'. The replacement of the resulting highest cost permutation by P_i' guarantees the survival of the "most-fit" parents. For example, $A = P_1$, $B = P_2$, $i = 3$, $j = 4$, mapping string A to string B exchanges the 2 and 5 and the 1 and 3 in row B while mapping string B to string A swaps the 5 and 2 and 3 and 1 in row A. In this example, P_0 is the minimum-cost permutation. The resulting P is

P =

	0	**1**	**2**	**3**	**4**	**5**
0	1	0	2	5	3	4
1	4	2	1	5	3	0
2	3	0	5	2	1	4

Mutation Phase:

8) Mutate the permutation periodically to *prevent premature loss of important notions* [7]. Randomly choose a number from the interval [0, 1]. If the number falls outside the interval $[1-q, 1]$, where q is the probability of mutation, then terminate the mutation step. Otherwise, select a random number between 1 and r, that designates the number of mutations that occur in the given step. For each of the mutations, select three random integer values i, j, k, such that $0 \le i \le p-1$, $0 \le j, k \le d-1$, $j-k$, and position-wise exchange $P_{i,j}$ with $P_{i,k}$. Given $q = 0.01$ and $r = 1$, a randomly generated value of 0.006, $i = 0$, $j = 1$, and $k = 5$, then P would be

P =

	0	**1**	**2**	**3**	**4**	**5**
0	1	4	2	5	3	0
1	4	2	1	5	3	0
2	3	0	5	2	1	4

Control Structure:

9) Repeat steps 3 through 8. The precise number of iterations is dictated by an early termination condition (all allocations are identical) or by a maximum iteration count. Upon termination, evaluate the "goodness" of the allocation defined by a row P_i, $(0 \le i \le p-1)$, and the corresponding mapping function $\mathcal{A}_i$. Choose the best allocation.

As with any other heuristic algorithm, the above algorithm is not assured to yield an optimal solution. However, we can still characterize its behavior and prove its likelihood to convergence to a good allocation [11]. Convergence was demonstrated by generalizing the schema notation to include permutations.

As a limited comparison study of the algorithm, we experimented with a small dataset and noted the average observed values of five runs. The greedy algorithm is deterministic and hence only one run was needed. For fairness, the random algorithm was

executed for the same duration of time as was our genetic algorithm. The comparison results are shown in Table 1 for four different architectural structures. The values represented are in terms of message hops. The lower the total number of hops (value), the better is the allocation. For additional experimental results, see [11].

TABLE 1
BEHAVIORAL COMPARATIVE EVALUATION

	Genetic Algorithm	Random Algorithm	Greedy Algorithm
Hypercube	23	25	24
Mesh 16 - by - 1	29	76	56
Mesh 8 - by - 2	23	43	32
Mesh 4 - by - 4	19	33	32

ACKNOWLEDGMENT

This work is supported, in part, by the National Science Foundation NYI program, under Contract No. IRI 935-7785.

REFERENCES

[1] I.J. Aalbersberg and F. Sijstermans, "High-Quality and High Performance Full-Text Document Retrieval: The Parallel Infoguide System," *Proc. IEEE Conf. Parallel and Distributed Information Systems*, pp. 142–150, Dec. 1991.

[2] S.H. Bokhari, "On the Mapping Problem," *IEEE Trans. Computers*, vol. 30, no. 3, pp. 207–214, Mar. 1981.

[3] J.K. Cringean, R. England, G.A. Manson, and P. Willett, "Parallel Text Searching in Serial Files Using a Processor Farm," *Proc. ACM SIGIR*, pp. 413–428, Sept. 1990.

[4] P. Efraimidis, C. Glymidakis, B. Mamalis, P. Spirakis, and B. Tampakas, "Parallel Text Retrieval on a High Performance Super Computer Using the Vector Space Model," *Proc. ACM SIGIR*, July 1995.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[6] D. Grossman, O. Frieder, D. Holmes, and D. Roberts, "Integrating Structured Data and Text: A Relational Approach," *J. Am. Soc. Information Science*, Jan. 1997.

[7] D.E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*. New York: Addison-Wesley, 1989.

[8] D. Hawking, and P. Thistlewaite, "Searching for Meaning with the Help of PADRE," *Overview Third Text Retrieval Conf. (TREC 3)*, Apr. 1995.

[9] C.A. Pogue, E.M. Rasmussen, and P. Willett, "Searching and Clustering of Databases Using the icl Distributed Array Processor," *Parallel Computing*, vol. 8, pp. 399–407, Oct. 1988.

[10] G. Salton and C. Buckley, "Parallel Text Search Methods," *Comm. ACM*, vol. 31, no. 2, pp. 202–215, Feb. 1988.

[11] H.T. Siegelmann and O. Frieder, "Document Allocation in Multiprocessor Information Retrieval Systems," *Lecture Note Series in Computer Science: Advanced Database Concepts and Research Issues*, N.R. Adam and B. Bhargava, eds., Springer-Verlag, 1993.

[12] C. Stanfill, "Partitioned Posting Files: A Parallel Inverted File Structure for Information Retrieval," *Proc. ACM SIGIR*, pp. 413–428, Sept. 1990.