The Allocation of Documents in Multiprocessor Information Retrieval Systems: *An Application of Genetic Algorithms*

Hava Tova Siegelmann
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
siegelma@paul.rutgers.edu

Ophir Frieder
Department of Computer Science
George Mason University
Fairfax, VA 22030
ophir@gmuvax2.gmu.edu

Abstract - Information Retrieval is the organization, manipulation, and selection of documents that are potentially relevant to a user's information need. To support the computational demands of information retrieval systems, various researchers have focussed on efforts that exploit parallelism. As efficient exploitation of parallelism demands fast access to the documents, data organization and placement significantly affects the total processing time. The determination of an optimal document allocation within a distributed memory, distributed I/O multicomputer, the Multiprocessor Document Allocation Problem (MDAP), is an NP-Complete problem. Obtaining an optimal document allocation, therefore, is computationally intractable, and hence, heuristic approaches are required. We describe our genetic-algorithm based approach to MDAP. We provide a proof of convergence for our algorithm and present a brief review of our experimental findings.

I. INTRODUCTION

Information retrieval is the organization, manipulation, and selection of documents that are potentially relevant to a user's information need. 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 affects the total processing time.

The determination of an optimal document allocation within a distributed memory, distributed I/O multicomputer, the Multiprocessor Document Allocation Problem (MDAP), is an NP-Complete problem. (A formal proof that MDAP is NP-Complete is provided in Frieder and Siegelmann [Fri91].) Obtaining an optimal document allocation, therefore, is computationally intractable, and hence, heuristic approaches are required.

We review and formally evaluate our heuristic approach to solve MDAP. The approach, initially introduced in [Fri91], is based on Genetic Algorithms [DeJ89, Gol89, Hol87]. In [Fri91], results from a simulation study of the algorithm are presented. Here, we highlight some of these results and provide a proof that, with a high probability, the allocations derived by our algorithm converge to an near optimal mapping.

Due to space limitations, we forgo our discussion on prior heuristic approaches to mapping problems that are similar to MDAP and refer the reader to [Bok81, Bol88, Du88, Lee87]. Some prior efforts related to multiprocessor information retrieval are described in [Aso90, Cri90, Fri89, Lee86, Mak91, Pog87, Pog88, Sha89, Sta86, Sta89, Sta90, Sto87].

The remainder of this paper is organized as follows. In Section II, we briefly review our genetic algorithm based approach to MDAP. In Section III, we provide the proof of convergence for our algorithm. We present a brief review of our experimental findings and conclude in Section IV.

II. A GENETIC ALGORITHM BASED APPROACH TO MDAP

Given a document collection D, consisting of documents D_i, $(0 \le i \le d - 1)$, and partitioned into clusters C_j, $(0 \le j \le c - 1)$, and given an n-node multicomputer architecture with an internode communication matrix M, an instance of MDAP requires that each document D_i, be allocated to a node X_l, $(0 \le l \le n - 1)$, such that the average cluster diameter is at a minimum. The diameter of a

Manuscript received July 29, 1991. This work was partially supported by grants from George Mason University, under the Faculty Summer Research Funding Program, DCS, Inc., under contract number 5-35071, and the Center for Innovative Technology under contract number 5-34042.

ISSN# 0-7803-0233-8/91 $1.00©1991 IEEE

cluster is the maximum distance between any pair of nodes that contain documents belonging to the given cluster. Each entry $M_{ij} \in M$, $(0 \le i, j \le n-1)$, designates the cost of sending a single packet from node i to node j.

Our genetic algorithm based approach to MDAP assumes that the documents are clustered using any one of the many clustering algorithms. For a review of clustering algorithms, see [Rag87, Sal83, Wil88]. The cost function to be minimized is the average cluster diameter.

The algorithm comprises of four steps (Initialization, Reproduction, Crossover, and Mutation) and is presented below. A detailed example is provided in [Fri91].

ALGORITHM:

Initialization Phase:

1. Create a permutation matrix, P_{ij} $(0 \le i \le p-1, 0 \le j \le d-1)$. Every row P_i, $(0 \le i \le p-1)$, is a complete permutation of all documents D_j, $(0 \le j \le d-1)$.

2. Define the document to node mapping function f_i: D -> X for any given row of P_i, $(0 \le i \le p-1)$ as $f_i(D_k) = j \bmod n$, where j is the index in row P_i of document D_k, $(0 \le k \le d-1)$.

Reproduction Phase:

3. Given the mapping function f_i for a given row P_i, $(0 \le i \le p-1)$, determine the cluster radii, R_{ij}, $(0 \le j \le c-1)$ for each cluster association list array entry, C_j.

$$R_{ij} = \text{Max}\{M_{f_i(D_k), f_i(D_l)} \mid 0 \le k, l \le d-1 \text{ and } D_k, D_l \in C_j\}.$$

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 f_i. In our case,

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

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.

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}.

Crossover Phase:

7. While maintaining a copy of the lowest-cost permutation, say P'_t, randomly pair up the rows in P. If p is odd, ignore the unpaired row. Randomly generate two integer values, i and j, such that $0 \le i \le j \le d-1$. For each pair of rows in P, say A and B, 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'_t. The replacement of the resulting highest cost permutation by P'_t guarantees the survival of the "most-fit" parents.

Mutation Phase:

8. Mutate the permutations periodically to *prevent premature loss of important notions* [Gol89]. Randomly choose a number from the interval [0, 1]. If the number falls outside the interval [0, q], where q is the probability of mutation, then terminate the mutation step. Otherwise, select a random number, t, $1 \le t \le T$, that designates the number of mutations that occur in the given step. For each of t iterations, select three random integer values i, j, k, such that $0 \le i \le p-1$, $0 \le j, k \le d-1$, $j \ne k$, and position-wise exchange P_{ij} with P_{ik}.

III. THEORETICAL FOUNDATION

Define Q as an equivalent, unique representation of the permutation matrix P where permutation Q_i, $(0 \le i \le p-1)$, is an allocation of the documents onto the processors, such that document D_j, $(0 \le j \le d-1)$, is allocated on processor X_k, $(0 \le k \le n-1)$,

if and only if ($Q_i[j] \bmod n) = k$. That is, position j in the permutation represents document j. The corresponding value modulus n is the processor where the document is stored. For example, if P is:

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

then the equivalent Q is:

Q =

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

Let P_i, $(0 \le i \le p - 1)$, be an arbitrary permutation in P. The equivalent representation of P_i in Q is Q_i, $(0 \le i \le p - 1)$. Replacing each value $Q_{ij} \in Q_i$ with (Q_{ij} modulus n) results in a vector of length d with the values 0 to n-1. We refer to this vector as the *allocation vector s* of the permutation P_i and Matrix S as the allocation matrix of P. Given n =3, the equivalent allocation matrix for P, S, is:

S =

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

Vector s defines an allocation where each document D_j is allocated to processor X_{sj}. Every allocation is represented by a unique vector s. Notice that the function AL: p -> s is not one to one. On the contrary, $\left(\left(\frac{d}{n}\right)!\right)^n$ different permutations map to the same allocation. If the population size (the number of rows in the matrix P) is k, than the probability that the initialization phase yields two or more permutations that define the same allocation is:

$$1 - \prod_{i=0}^{k-1} \frac{d! - ix}{d!}, \text{ where } x = \left(\left(\frac{d}{n}\right)!\right)^n .$$

Define the *s-schema* of an allocation vector s as the vector itself, however, in some of the positions in s, instead of the actual value appearing, a special symbol & is present. This symbol designates any of the valid orderings of the values 0 to n - 1. For example, given the allocation vectors s_0 and s_1, s_0 = "1 0 2 1 2 0", s_1 = "0 0 1 1 2 2", s-schema H_0 may be "& 0 & 1 2 &". H_0 is an s-schema of both s_0 and s_1. Schema H_1 = "1 0 & 1 2 &", however, is an s-schema of s_0 but not of s_1. Note that each value i, out of the possible set of n values, must appear $\frac{d}{n}$ times in each vector s. Let #(i) designate the number of the fixed positions that the value i already appears in the schema. Then, $\frac{d}{n}$ - #(i) of the appearances of the special symbol &, designate the value i.

There are 2^d schemata in a given allocation s. If the population size is k, then there are between 2^d to $k * 2^d$ different schemata that are implied by the population. Similar to the definitions in Goldberg [Gol89], let

o(H) be the schema order. That is, o(H) is the number of fixed positions in the schema (positions not marked with a &).

δ(H) be the difference between the first and the last location of the fixed part of the schema H.

r(H) be the difference of d and the index of the right most position of the fixed part of the schema.

Consider the effects of reproduction, crossover and mutation phases on the schemata implied by the population permutations. The effects of reproduction precisely mimic the effects of reproduction on binary strings. For complete details of the effect of reproduction of binary strings, see Goldberg [Gol89]. Briefly, suppose that at time t there are m(H,t) copies of a particular schema H in the population A(t). Let f(H) be the average fitness of the permutations representing schema H at time t.

If $\bar{f}$ is the average fitness over the population, then after a reproduction step roughly

$$m(H,t+1) = m(H,t) * \left(\frac{f(H)}{\bar{f}}\right)$$

copies of schema H will exist. If for all time t,

$f(H) \geq (1+c) * \bar{f}$, where c is a constant,

then

$m(H,t) = m(H,0) * (c+1)^d$.

Thus, popular schemata grow exponentially.

In the crossover phase, two random numbers are chosen as the boundaries. A schema can be destroyed if the boundaries bound some of its fixed values. Therefore, the probability of the survival of a schema H is

$$\left(\frac{r(H)}{d}\right)^2 + \left(\frac{d - r(H) - \delta(H)}{d}\right)^2.$$

This probability results in an increase of the number of copies of short schemata.

The mutation phase modifies schema H of a permutation P_i, $(0 \leq i \leq p - 1)$, if and only if

1. At least one of the two positions chosen are of the fixed part of the schema; (This occurs with a probability of $\frac{2o(H)}{d} - \left(\frac{o(H)}{d}\right)^2$)
2. The positions do not designate the same processor. (This occurs with a probability of $\frac{d - n}{d - 1}$)

During each mutation phase, i single mutations result, $1 \leq i \leq T$. The probability of surviving a mutation is:

$$(1-q) + \frac{q}{T}\sum_{i=1}^{T}\left(1 - \frac{1}{p}\left(\frac{d-n}{d-1}\right)\left(\frac{2o(H)}{d} - \left(\frac{o(H)}{d}\right)^2\right)\right)^i$$

where q is the probability that a mutation will occur. A schema with a small number of fixed positions is more likely to survive.

We summarize the combined influence of the three operations is the next formula (ignoring low order numbers):

$$m(H,t+1) \geq m(H,t) * \left(\frac{f(H)}{\bar{f}}\right) * \left(1 - \frac{2\delta(H)}{d} - q\left(1 - \left(1 - \frac{1}{p}\left(\frac{d-n}{d-1}\right)\left(\frac{2o(H)}{d}\right)\right)^T\right)\right)$$

This equation demonstrates that a schema grows exponentially according to the ratio of its fitness and the average fitness in the current population, and inversely to the number of fixed positions and the distance between the first and last fixed position within the schema.

In summary, the reproduction phase exponentially converges the population towards a minimal cluster diameter. The crossover and mutation phases guarantee that a wide search space will be investigated. Therefore, given an initial random population representing an arbitrary assignment of documents onto nodes, with a high probability, a near optimal document to node allocation will be derived.

IV. EXPERIMENTAL EVALUATION

A simulation was developed to evaluate the described algorithm. Two document distributions and various 16-node multicomputer configurations, namely, a 4-dimensional hypercube, and a 4-by-4, 8-by-2, and 16-by-1 mesh topologies, were used in the study. Each document distribution comprised of 64 items partitioned into 8 clusters, with the distribution of documents varied across the clusters. That is, in the first distribution, each cluster comprised of 8 documents, while in the second, 25 percent of the clusters contained 50 percent of the number of documents. Using the notation prescribed in [Fri91], (64, 8, 25, 50) refers to the latter document distribution, while the even document partitioning is represented by (64, 8, x, x), for all values of x, $0 < x \leq 100$.

In Figures 1 and 2, a point on any curve represents an iteration in which a better allocation was derived. All runs terminated at either a point in which the entire population (document allocations), in this case 30, were identical or after 1000 iterations (premature termination), which ever came first. A point at 1000 indicates that premature termination occurred. As expected, the higher the communication diameter of the architecture, the

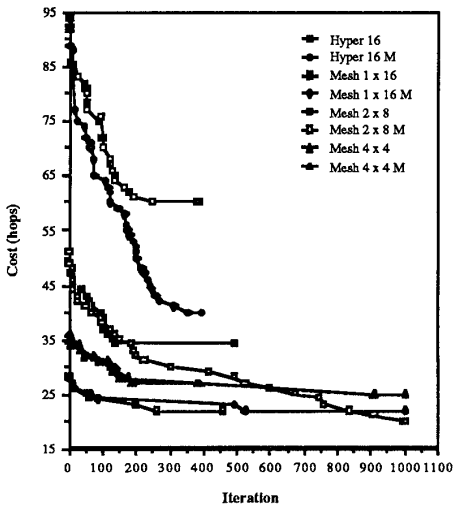

Fig. 1. An even document distribution across the cluster

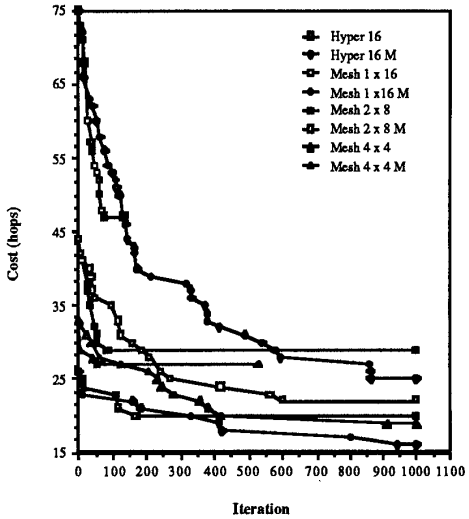

Fig. 2. A (64, 8, 25, 50) document distribution across the clusters

more significant was the improvement in the derived allocation from the initial random document distribution.

For each architecture studied, runs that incorporated and those that did not incorporate the mutation step as part of the genetic algorithm were analyzed. As seen, all runs incorporating mutation resulted in at least as good of an allocation as those were the mutation step was not included. The removal of the mutation step resulted in the algorithm converging on a local minima; therefore a global minima was not obtained.

V. SUMMARY

Exploiting parallelism in the information retrieval domain requires the allocation of documents onto nodes such that an efficient access to the documents is supported. Determining an optimal allocation based on access patterns and the relation of documents to each other is not computationally feasible. (The problem is NP-Complete.) Therefore, heuristics are developed to derive a near optimal allocation.We described one such heuristic approach based on genetic algorithms. A proof of convergence of the derived allocation to an optimal distribution was outlined. Some experimental results from a simulation study of the approach were provided. The results presented illustrated the potential of our genetic-algorithm based approach as a means of tackling MDAP.

Future efforts will consist of additional simulation studies that will focus on the effects of varying the population size and the probability and maximum degree of mutation. Also, a hypercube information retrieval system is currently being developed. Once available, we will use the hypercube engine as our experimental test-bed to evaluate our derived mappings as compared to the more traditional round-robin, hashed, and/or attribute-based partitioning schemes.

ACKNOWLEDGMENT

We would like to thank the support and comments of William Steiger during the early part of this effort.

REFERENCES

[Aso90] Asokan, N., S. Ranka, and O. Frieder, "A Parallel Free-text Search System with Indexing," *Proceedings of the IEEE International Conference on Parallel Architectures and Databases,* pp 519-521, March 1990.

[Bok81] Bokhari, S. H., "On the Mapping Problem," *IEEE Transactions on Computers,* 30(3), pp 207-214, March 1981.

[Bol88] Bollinger, S. W. and S. F. Midkiff, "Processor and Link Assignment in Multicomputers Using Simulated Annealing," *Proceedings of the 1988 International Conference on Parallel Processing,* pp 1-7, August 1988.

[Cri90] Cringean, J. K., R. England, G. A. Manson, and P. Willett, "Parallel Text Searching in Serial Files Using a Processor Farm," *Proceedings of the 1990 ACM SIGIR,* pp 413-428, September 1990.

[DeJ89] De Jong, K. A. and W. M. Spears, "Using Genetic Algorithms to Solve NP-Complete Problems," *Proceedings of the Third International Conference on Genetic Algorithms,* pp 124-132, June 1989.

[Du88] Du, X. and F. J. Maryanski, "Data Allocation in a Dynamically Reconfigurable Environment," *Proceeding of the IEEE Fourth International Conference on Data Engineering,* pp 74-81, February 1988.

[Fri89] Frieder, O., K. C. Lee, and V. Mak, "JAS: A Parallel VLSI Architecture for Text Processing," *IEEE Database Engineering,* 8(1), March 1989.

[Fri91] Frieder, O. and H. T. Siegelmann, "On the Allocation of Documents in Multiprocessor Information Retrieval Systems," *Proceedings of the 1991 ACM SIGIR,* October 1991.

[Gol89] Goldberg, D. E. Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley, New York, 1989.

[Hol87] Holland, J. H., "Genetic Algorithms and Classifier Systems: Foundations and Future Directions," *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications,* pp 82-89, June 1987.

[Lee86] Lee, D., "Altep - A Cellular Processor for High-Speed Pattern Matching," *New Generation Computing,* vol. 4, pp 225-244, September 1986.

[Lee87] Lee, S.-Y. and J. K. Aggarwal, "A Mapping Strategy for Parallel Processing," *IEEE Transactions on Computers,* 36(4), pp 433-442, April 87.

[Mak91] Mak, V., K. C. Lee, and O. Frieder, "Exploiting Parallelism in Pattern Matching: An Information Retrieval Application," *to appear in ACM Transactions on Information Systems.*

[Pog87] Pogue, C. A. and P. Willett, "Use of Text Signatures for Document Retrieval in a Highly Parallel Environment," *Parallel Computing,* Elsevier (North-Holland), vol. 4, pp 259-268, June 1987.

[Pog88] Pogue, C. A., E. M. Rasmussen, and P. Willett, "Searching and Clustering of Databases Using the ICL Distributed Array Processor," *Parallel Computing,* Elsevier (North-Holland), vol. 8, pp 399-407, October 1988.

[Rag87] Raghavan, V. V. and B. Agarwal, "Optimal Determination of User-Oriented Clusters: An Application for the Reproductive Plan," *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications,* pp 241-246, June 1987.

[Sal83] Salton, G. and M. J. McGill, Introduction to Modern Information Retrieval, McGraw Hill, New York, 1983.

[Sha89] Sharma, R., "A Generic Machine for Parallel Information Retrieval," *Information Processing and Management,* Pergamon Press, 25(3), pp 223-235, 1989.

[Sta86] Stanfill, C. and B. Kahle, "Parallel Free-text Search on the Connection Machine System," *Communications of the ACM,* 29(12), pp 1229-1239, December 1986.

[Sta90] Stanfill, C., "Partitioned Posting Files: A Parallel Inverted File Structure for Information Retrieval," *Proceedings of the 1990 ACM SIGIR,* pp 413-428, September 1990.

[Sta89] Stanfill, C., R. Thau, and D. Waltz, "A Parallel Indexed Algorithm for Information Retrieval," *Proceedings of the 1989 ACM SIGIR,* pp 88-97, June 1989.

[Sto87] Stone, H. S., "Parallel Querying of Large Databases: A Case Study," *IEEE Computer,* 20 (10), pp 11-21, October 1987.

[Wil88] Willett, P., "Recent Trends in Hierarchic Document Clustering: A Critical Review," *Information Processing and Management,* Pergamon Press, 24(5), pp 577-597, 1988.