On the Allocation of Documents in Multiprocessor Information Retrieval Systems

Ophir Frieder
Dept. of Computer Science
George Mason University
Fairfax, VA 22030

Hava Tova Siegelmann
Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08903

Abstract. Information retrieval is the 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 affect the total processing time. We describe and evaluate a data placement strategy for distributed memory, distributed I/O multicomputers. Initially, a formal description of the Multiprocessor Document Allocation Problem (MDAP) and a proof that MDAP is NP Complete are presented. A document allocation algorithm for MDAP based on Genetic Algorithms is developed. This algorithm assumes that the documents are clustered using any one of the many clustering algorithms. We define a cost function for the derived allocation and evaluate the performance of our algorithm using this function. As part of the experimental analysis, the effects of varying the number of documents and their distribution across the clusters as well the exploitation of various differing architectural interconnection topologies are studied. We also experiment with the several parameters common to Genetic Algorithms, e.g., the probability of mutation and the population size.

1.0 Introduction

An efficient multiprocessor information retrieval system must maintain a low system response time and require relatively little storage overhead. As the volume of stored data continues to increase daily, the multiprocessor engines must likewise scale to a large number of processors. This demand for system scalability necessitates a distributed memory architecture as a large number of processors is not currently possible in a shared-memory configuration. A distributed memory system, however, introduces the problem associated with the proper placement of data onto the given architecture. We refer to this problem as the Multiprocessor Document Allocation Problem (MDAP), a derivative of the Mapping Problem originally described by Bokhari [Bok81].

We assume a clustered document database. A clustered approach is taken since an index file organization can introduce vast storage overhead (up to roughly 300% according to Haskin [Has81]) and a full-text or signature analysis technique results in lengthy search times. In this context, a proper solution to MDAP is any mapping of the documents onto the processors such that the average cluster diameter is kept to a minimum while still providing for an even document distribution across the nodes.

To achieve a significant reduction in the total query processing time using parallelism, the allocation of data among the processors should be distributed as evenly as possible and the interprocessor communication among the nodes should be minimized. Achieving such an allocation is NP Complete. Thus, it is necessary to use heuristics to obtain satisfactory mappings, which may indeed be suboptimal.

Genetic Algorithms [DeJ89, Gol89, Gre85, Gre87, Hol87, Rag87] approximate optimal solutions to computationally intractable problems. We develop a genetic algorithm for MDAP and examine the effects of varying the communication cost matrix representing the interprocessor communication topology and the uniformity of the distribution of documents to the clusters.

1.1 Mapping Problem Approximations

As the Mapping Problem and some of its derivatives are NP complete, heuristic algorithms are commonly employed to approximate the optimal solutions. Some of these approaches [Bok81, Bol88, Lee87] deal, in some manner,

This work was partially supported by grants from DCS, Inc. under contract number 5-35071 and the Center for Innovative Technology under contract number 5-34042.

with the mapping of a communicating set of processes onto an architecture with a fixed interconnection topology. This problem is similar to MDAP in that both problems must map a set of tasks (items) onto a given architecture. However, the goals of the above efforts differ from MDAP as MDAP does not aim to maximize the amount of concurrent interprocess communication, but instead, aims at reducing the total communications diameter of its logical tasks (clusters).

Bokhari [Bok81] introduced a pairwise-exchange heuristic algorithm that accepted as input two adjacency matrices representing graphs G (set of communicating processes) and G' (target architecture). Using the cardinality of the number of communicating pairs that directly communicated with their neighbor as the objective function, Bokhari developed and evaluated an algorithm that mapped graph G onto graph G'. Lee and Aggarwal [Lee87] extended that effort by developing objective functions that more accurately quantified the communication overhead. Using a set of *objective functions* (parametric equations) that corresponded to the cost associated with the given mapping, Lee and Aggarwal precisely measured the optimality of the derived mapping. The main limitation in their approach was that it only employed a fixed path routing scheme for the network traffic.

Bollinger and Midkiff [Bol88] used a two-phase simulated annealing algorithm to map a logical system onto a physical architecture. The first phase, process annealing, assigned the processes onto the physical nodes. The connection annealing phase mapped the logical connections onto the network data links so as to minimize communication link conflicts. This effort improved upon Lee and Aggarwal [Lee87] in that it utilized the information concerning the actual routing rules.

Du and Maryanski [Du88] attacked a variation of the mapping problem. This variation concerned the allocation of data in a dynamically reconfigurable environment. The allocation algorithm used a set of "benefit" functions and a greedy search algorithm. The underlying execution architecture was based on a client/server model (a heterogeneous system). Although their problem more closely resembles MDAP, as the underlying architectural model significantly differs from the MDAP execution environment, their assumptions are not relevant to MDAP.

1.2 Related Multiprocessor Systems

Distributed-memory information retrieval systems have been investigated as a mean of providing short response times to users' requests. Some of these systems include various efforts on the Connection Machine [Aso90, Sta86, Sta89, Sta90], on the Distributed Array Processor (DAP) [Pog87, Pog88], on a network of Transputers [Cri90], and on hypercube systems [Sha89]. Both the commentary on and the extensions of the Connection Machine efforts [Aso90, Sal88, Sta89, Sta90, Sto87] as well as the hypercube [Sha89], DAP [Pog87, Pog88] and Transputer [Cri90] efforts have all addressed the notion of data organization in the search and retrieval scheme employed.

Stone [Sto87] demonstrated analytically that, by indexing keywords, a uniprocessor system with comparable memory to that of the Connection Machine employed in the Stanfill and Kahle effort [Sta86] can achieve similar user retrieval response times to those times reported in [Sta86]. Via keyword indexing, the volume of data that had to be searched was reduced, significantly reducing the total I/O processing time. A parallel index-based retrieval effort on the Connection Machine was later reported in Stanfill, Thau, and Waltz [Sta89] and in Asokan, Ranka, and Frieder [Aso90]. Additional parallel text retrieval search methods are described in Salton and Buckley [Sal88].

Various descriptions of efforts that focus on the organization of data for the DAP system, appear in the literature. In Pogue and Willett [Pog87], an approach using text signatures is proposed and evaluated using three document databases comprising of roughly 11000, 17000, and 27000 documents. Pogue, Rasmussen, and Willett [Pog88], describe several clustering techniques using the DAP. Both reports clearly demonstrate that if a proper document mapping onto the individual Processing Elements (PEs) is established, the DAP system readily achieves a high search rate. However, an improper mapping results in a poor search rate stemming from the inability of the DAP system to access the documents.

Cringean, et. al., [Cri90] describe early efforts aimed at developing a processor-pool based multicomputer system for information retrieval. The physical testbed hardware consists of an Inmos Transputer network. To reduce the volume of data accessed and hence the total query processing times, a two phase retrieval algorithm is proposed. The initial phase acts as a filter to retrieve all potentially relevant documents. By using text signatures, the majority of the non-relevant documents are eliminated from further processing. This filtering of documents vastly

reduces the volume of data processed in the compute intensive second phase. (Some non-relevant documents are selected as a consequence of false-drops. False-drops are common to all text signature analysis approaches.) In the second phase, full text search is performed. The two phase algorithm is yet another example that emphasizes the need for intelligence in the organization and retrieval of documents.

Finally, Sharma [Sha89] describes a hypercube-based information retrieval effort. The results presented are based on the timing equations provided. To reduce the volume of data read, Sharma relies on the fact that the documents are initially partitioned into clusters, and only documents that belong to "relevant" clusters are retrieved. As in our approach, Sharma does not address nor is dependent on any particular clustering technique. Sharma does require, however, that the cluster scheme employed yield non-hierachical clusters, whereas we do not impose such a restriction. Thus, all the clustering schemes, including the numerous schemes described in Willett [Wil88] can also be employed in our scheme. Sharma partitions the clusters across the individual nodes according to an architectural topology-independent, best-fit heuristic. No evaluation of the document distribution algorithm is provided.

We also rely on clustering, but use a genetic algorithm approach that uses information about the underlying architecture to map the documents onto the nodes. As the actual dataset used in the [Sha89] evaluation is not described, it is not possible to directly compare the results of our algorithm to the algorithm described in [Sha89]. For tutorials on clustering and other information retrieval related topics, the reader is referred to [Bra90, Sal83, Wil88].

The remainder of the paper is organized as follows. A proof that MDAP is NP-Complete is sketched in Section 2. Section 3 comprises of a description of a proposed Genetic Algorithm for MDAP. Results from an experimental evaluation of our approach are illustrated in Section 4. We conclude in Section 5.

2.0 MDAP is NP-Complete

An instance of MDAP consists of a homogeneous distributed memory architecture with n nodes, X_i, $0 \le i \le n - 1$, and partitions of the documents D_i, $0 \le i \le d - 1$, called clusters, C. Each cluster C_i, $0 \le i \le c - 1$, represents a list of all the documents associated with it. Typically, $\sqrt{d} \le c \le d/\text{constant}$ [Sha89]. The distance between a pair of nodes is defined as the cost of sending a packet from node i to node j and is represented by the internode communication cost matrix, M_{ij}, $0 \le i, j \le n - 1$. The diameter of a cluster is the maximum distance between any pair of nodes that contain documents belonging to the given cluster. MDAP requires the documents to be evenly distributed across the nodes, such that the sum over all cluster diameters is minimal. As the sum of the cluster diameters is reduced, the total communication traffic is minimized.

To prove that MDAP is NP-Complete a polynomial-time reduction of the Quadratic Assignment Problem [Gar79] to the decision form (yes/no problem) of MDAP is provided. MDAP is formally defined as:

Instance:

A distributed memory architecture with:

Nodes (PEs): $X = \{X_i \mid 0 \le i \le n - 1\}$;
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: $C = \{C_i \mid 0 \le i \le c - 1, C_i \in D\}$;

A real value bound: B.

Question:

Is there an allocation of the documents to the processors such that:

1. The number of documents at each node X_i is $\frac{d}{n}$, $(0 \le i \le n - 1)$;

2. $\sum_{\text{all clusters } C_j,\ 0 \le j \le c - 1} \text{diameter}(C_j) \le B.$

   where diameter (C_j) = $\max(M_{rt})$, and documents $D_k, D_l \in C_j$, $0 \le k, l \le d - 1$, $0 \le j \le c - 1$, reside at nodes r and t, respectively.

Theorem 1. MDAP is NP-Complete.
Proof:

1. Given an instance of MDAP, the defined allocation can be checked to satisfy the stated equality and bound requirements (statement 1 and 2) using a polynomial time algorithm. Therefore, MDAP is in NP.

2. To show the NP-Completeness of MDAP, we use the NP-Complete, Binary Quadratic Assignment Problem [BQAP] defined in Garey and Johnson [Gar79].

Binary Quadratic Assignment Problem [BQAP]:

Instance:
Non-negative costs:

$b_{ij} \in \{0,1\}$, $b_{ij} = b_{ij}$; $1 \le i, j \le g$;
Distances: m_{kl}, $1 \le k, l \le h$;
Bound: $Z \in Z^+$.

Question:
Is there a one to one function $f:\{1, 2, 3, ..., g\} \rightarrow \{1, 2, 3, ..., h\}$ such that

$$\sum_{i \neq j} b_{ij} * m_{f(i)f(j)} \le Z \ ?$$

Given an instance of the BQAP, define the instance of MDAP as follows:
An architecture with h processors;
Cost matrix $M_{kl} = m_{kl}$, $1 \le k, l \le h$;
A set of g documents;

A mapping of the documents to clusters such that each cluster consists of two documents.
$\{D_i, D_j\}$ is a cluster iff $b_{ij} = b_{ij} = 1$;
A bound $B = \frac{Z}{2}$.

The transformation takes not more than polynomial time in the input's size and results in an even distribution of documents if and only if there exists a one-to-one function when $g \le h$. The sum of the clusters' diameters is not more than B if and only if the sum of $b_{ij} * m_{f(i)f(j)}$ is not more than twice B, namely Z.
Thus, MDAP is NP-Complete. ❑

3.0 A Genetic Algorithm for MDAP

As 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 [Gol89]. In our representation, the set of documents are represented by *document vectors* which are a sequence of integers 0 to d - 1. A permutation of this sequence defines an allocation of the documents onto the nodes where a document D_i found at position j is stored at node j modulus n. This representation scheme results in all nodes containing an equal number of documents, with the possible difference of a single document. Each allocation is evaluated as the sum of the cluster diameters it defines. The optimization function used is the inverse of the sum of the cluster diameters. The lower the sum, the better is the allocation.

As with most genetic algorithms, the proposed algorithm comprises of three major phases (initialization, reproduction, and crossover) and an additional secondary phase (mutation). In the initialization phase, a set of random permutations of the document vectors are generated. Each random permutation represents a possible allocation of the documents onto the nodes. By repetitively modifying the permutations, a near optimal allocation is generated. The number of simultaneous permutations (p) is an experimental parameter that is evaluated in Section 4.

The reproduction phase replaces permutations that represent *poor* mappings with those permutations that are viewed as *good*. Using the sum of the cluster diameters as an objective function, the merit of each permutation is evaluated. A biased roulette wheel favoring the better permutations (allocations) is created. A randomly sampled value is obtained. Using the biased roulette wheel and the sampled value, a corresponding allocation is determined. Each selection corresponds to the *birth* of a new allocation. The permutation that is replaced by this new birth is deemed as *deceased*. Upon the completion of each reproduction phase, with a high probability, the *poor* allocations are *killed*, while additional copies of the *good* allocations are reborn.

The crossover phase represents the *cross-fertilization* of permutations, similar to the composition of genes from both parents in a birth, and consists of a position-wise exchange of values between each randomly paired permutations. Two random numbers are chosen and serve as the bounds for the position-wise exchange. Each document of the first permutation that falls within the determined bounds is swapped with the corresponding document of the second permutation, and likewise the second permutation with the first.

Finally, to lower the probability of convergence of the allocation to local values that are not a global minima, a mutation phase is incorporated into the algorithm. Periodically, with a low probability, a permutation is randomly modified.

ALGORITHM:

Initialization Phase:

1. Create a permutation matrix, P_{ij} $(0 \le i \le p - 1, 0 \le j \le d - 1)$. 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 f_i: D -> X for any given row of P, 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, $(0 \le k \le d - 1)$. Continuing with the above example, 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 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. $R_{ij} = \text{Max}\{M_{f_i(D_k)}, f_i(D_l) \mid 0 \le k, l \le d - 1$ and $D_k, D_l \in C_j\}$. Then, if

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 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. 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 0.4, 0.34, and 0.26, and are assigned the intervals [0.0, 0.4), [0.4, 0.74), [0.74, 1.0], 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'_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. For example, if i = 3, j = 4, A = P_1, and B = P_2, 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 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, 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}. Given q = 0.01 and t = 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. Throughout the experimentation presented here, the maximum limit was set at 1000. In the future, an appropriate limit, possibly a percentage of the population size and/or the number of documents, will be determined experimentally. A proof of convergence of the derived allocation to an optimal mapping is provided in Siegelmann and Frieder [Sie91]. 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 f_i. Choose the best allocation.

4.0 Simulation Study

To evaluate the described algorithm, a simulation was developed. Given a particular multicomputer architecture (the number of nodes and a cost matrix specifying the internode communication topology) and a set of documents partitioned into clusters, the simulation derived a document allocation using the proposed genetic algorithm. The cost of the derived allocations over a magnitude of architectures and document distributions were used to evaluate the merit of the algorithm.

Various partitioning schemes of the documents into clusters were considered. Sharma [Sha89] stated that d-document collections form from $\sqrt{d}$ to d/constant clusters and assumed such a cluster organization in his evaluation. However, he did not mention what assumptions were made regarding the number of documents per cluster. In this study, we assumed $\sqrt{d}$ clusters and varied the number of documents per cluster. That is, the number of documents per cluster varied from a uniform distribution of documents to clusters to a partitioning in which 25 percent of the clusters contained 50 percent of the documents. The behavior of the proposed algorithm was observed in terms of these varied allocations.

Several multicomputer architectures were considered. These include a 16-node hypercube engine and three mesh configurations (1 by 16, 2 by 8, 4 by 4). Future studies will include various additional interconnection topologies and the varying of the number of nodes.

The effects of varying several parameters common to many genetic algorithms were studied.

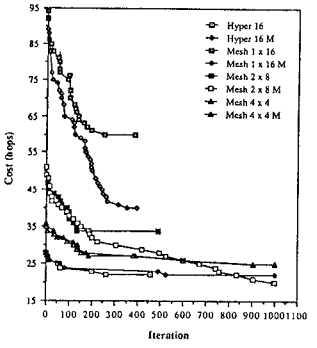

Fig. 1A. All architectures with an even document distribution

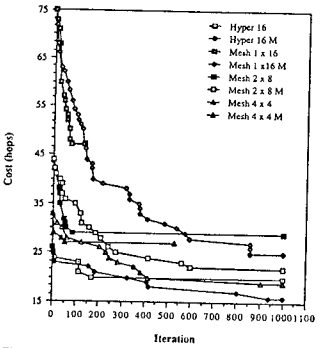

Fig. 1B. All architectures with a (64, 8, 25, 50) distribution

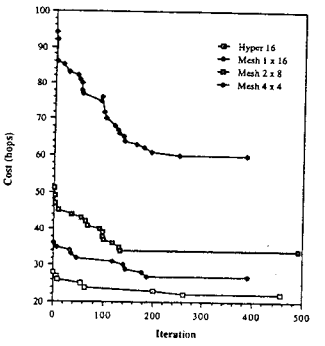

Fig. 2A. Without mutation on an even distribution

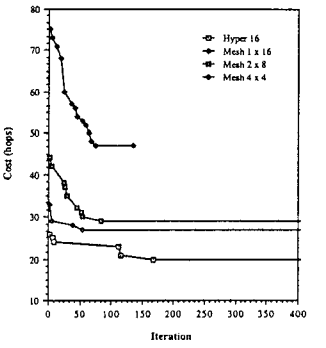

Fig. 2B. Without mutation on a (64, 8, 25, 50) distribution

These parameters include the size of the population (number of permutations) and the probability of mutation. Five population sizes ranging from 10 to 50 permutations in increments of ten permutations were examined. The population sizes investigated were kept small to coincide with the size of the database modelled (64 documents). Future studies will involve models comprising of more realistically-sized databases and larger population samples. The effects of both incorporating and not incorporating the mutation phase were also investigated.

Figures 1 through 5 illustrate results for a 64 document database distributed over 16 node systems of varying interconnection topologies. The results for two different document to cluster partitioning are presented. Both document partitions employ 8 clusters but the distribution of documents to clusters is varied. That is, in the first distribution (figures 1(a), 2(a), 3(a), 4(a) and 5(a)), all clusters contain an equal (8) number of documents. In the second distribution, (figures 1(b), 2(b), 3(b), 4(b) and 5(b)), 25 percent of the clusters contain 50 percent of the number of documents. For notational convenience, we describe a document partitioning by a four-tuple (D, C, x, y), where D is the number documents, C is the number of clusters, and x and y represent the x percentage of clusters containing y percent of the documents. Therefore, (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, $0 < x \leq 100$.

Figures 1(a) and (b) present the results for all the architectures considered. A point on any curve represents an iteration in which a better allocation was derived. As shown, the number of points varies with the architecture considered. 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

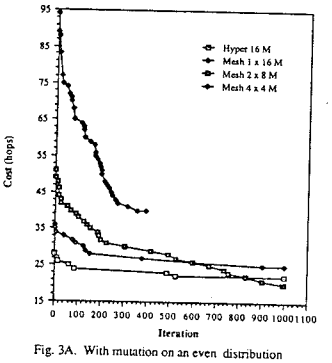

Fig. 3A. With mutation on an even distribution

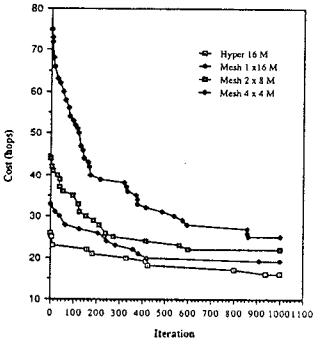

Fig. 3B. With mutation on a (64, 8, 25, 50) distribution

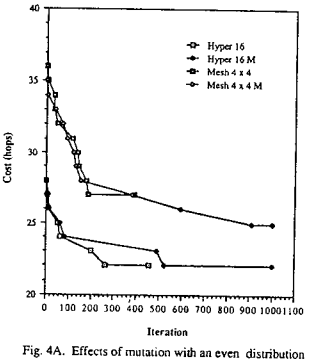

Fig. 4A. Effects of mutation with an even distribution

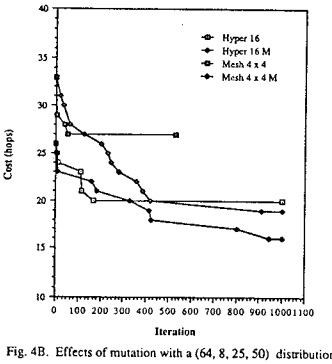

Fig. 4B. Effects of mutation with a (64, 8, 25, 50) distribution

communication diameter of the architecture, the more significant was the improvement in the derived allocation from the initial random document distribution.

Figures 2(a) and (b) and 3(a) and (b) more clearly illustrate the behavior of the proposed algorithm in the case where no genetic mutations are possible and when a 0.5 probability of mutation exists, respectively. A 0.5 mutation implies that with a probability of 0.5, a random number of pairs ranging from 1 to 10, will be exchanged. That is on average, 2.75 pairs will be exchanged per iteration. Figures 4(a) and (b) illustrate the effects of mutation on the allocations derived for a hypercube and a 4-by-4 mesh systems. As seen, and in all runs performed, mutation results in better allocations. The better allocations result from the prevention of local minima interference.

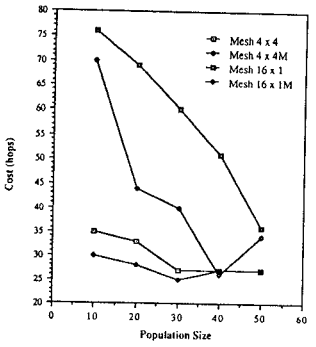

Fig. 5A. Effects of population size using an even distribution

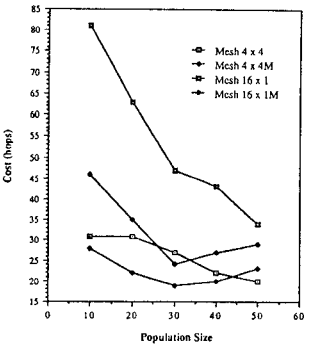

Fig. 5B. Effects of population size using a (64, 8, 25, 50) distribution

	(64, 8, 25, 50)	(64, 8, x, x)
Hypercube	23	25
Mesh 16-by-1	66	76
Mesh 8-by-2	37	43
Mesh 4-by-4	28	33

Fig. 6. Cost of psuedo-optimal distributions

Finally, figures 5(a) and (b) demonstrate the effects on varying the population size from 10 to 50 allocations in increments of 10. When mutations are not possible, the performance consistently improves with the increase in the population size. The improved allocations result from the greater number of possibilities explored during each iteration. In the case where mutation is possible however, initially the performance is improved and then eventually deteriorates. The improvement, as in the case where mutations are not possible, is caused by the increase in the number of possibilities explored. Since the maximum number of mutations per iteration is kept constant throughout, increasing the population size reduces the effects of mutation. Thus, the benefit derived from the mutation phase is diminished. Diminishing the effects of the mutation phase results in a derived allocation that more closely resembles the case in which no mutation is possible. Hence, the performance degrades. This result is a motivation for further study aimed at determining the precise ratio between the maximum allowable mutation and the size of the population.

Ideally, the cost of the derived mappings should be compared against the cost of an optimal allocation. Since determining an optimal mapping, in the general case, is not computationally feasible (the problem was shown to be NP-Complete in section 2), such a comparison is not possible. Instead, to compare the efficiency of the derived mapping, 100,000 permutations were randomly drawn. Given the architectural specification and the document distribution, the associated cost of each random permutation was computed. The lowest cost permutation was used to approximate the optimal solution.

Figure 6 presents the cost of the pseudo-optimal allocations for the respective architecture and document partitioning schemes. When mutations are possible, for all combinations of architectures, document partitioning schemes, and population sizes evaluated, the proposed genetic algorithm obtained better results than the pseudo-optimal solution. When mutations were not possible, the combination of a 16-by-1 Mesh, a (64, 8, 25, 50) document database, and a population size of 10, resulted in the pseudo-optimal algorithm deriving a better allocation than the genetic algorithm. This demonstrated the danger of employing too small of a population. In all other cases, the proposed algorithm derived better allocations than those derived by the pseudo-optimal algorithm. The results obtained indicate that the proposed genetic document allocation algorithm does indeed derive *good* document allocations using a computationally tractable, in contrast to using an exponential algorithm, approach.

6.0 Conclusions

The performance of multiprocessor information retrieval systems depend not only on the underlying parallel technology employed but, at least as significantly, on the organization of the data to be retrieved. Poor data allocations result in minimal performance gains on a parallel engine as compared to a uniprocessor system. The problem addressing the derivation of document allocations that support efficient retrieval of documents from a distributed-memory multiprocessor is called MDAP. As an optimal solution to MDAP is not computationally feasible (MDAP is NP-Complete), we proposed a genetic algorithm for MDAP. Via simulation, the derived document allocations were analyzed. The results obtained compared favorably with pseudo-optimal document allocations and demonstrated the success of the proposed algorithm.

We are presently continuing to evaluate our proposed algorithm. Future experimentation will consist of models representing larger document databases, documents with non-uniform access frequency, and correspondingly larger population spaces. A detailed study of the effects of mutation on the derived allocations will be conducted.

We are also presently developing a hypercube-based multiprocessor information retrieval system. Once developed, we will evaluate the actual response time difference resulting from various standard document allocation schemes, e.g., round-robin, hashed, etc., and those allocations derived by the algorithm described in this paper.

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.

[Bra90] Brajnik, G., G. Guida, and C. Tasso, "User Modeling in Expert Man - Machine Interfaces: A Case Study in Intelligent Information Retrieval," *IEEE Transactions on Systems, Man, and Cybernetics,* 20(1), pp 166-185, January/February 1990.

[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.

[Gar79] Garey, M. R. and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and Co., New York, 1979.

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

[Gre85] Grefenstette, J., Proceedings of an International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates Hinsdale, NJ, 1985.

[Gre87] Grefenstette, J., Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum Associates Hinsdale, NJ, 1987.

[Has81] Haskin, R. L., "Special-Purpose Processors for Text Retrieval," *Database Engineering 4,* pp 16-29, September 1981.

[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.

[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.

[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.

[Sal88] Salton, G. and C. Buckley, "Parallel Text Search Methods," *Communications of the ACM,* 31(2), pp 202-215, February 1988.

[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.

[Sie91] Siegelmann, H. and O. Frieder, "The Allocation of Documents in Multiprocessor Information Retrieval Systems: An Application of Genetic Algorithms," *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics,* Charlottesville, Virginia, October 1991.

[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.