4-1991

An Evolutionary Approach to Load Balancing Parallel Computations

N. Mansouri
Syracuse University, Department of Engineering and Computer Science, namansou@ecs.syr.edu

Geoffrey C. Fox
Syracuse University

SU-CIS-91-13

An Evolutionary Approach to Load Balancing Parallel Computations

Nashat Mansour and Geoffrey C. Fox

April 1991

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

SU-CIS-91-13

An Evolutionary Approach to Load Balancing Parallel Computations

Nashat Mansour and Geoffrey C. Fox

April 1991

School of Computer and Information Science
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

(315) 443-2368

An Evolutionary Approach to Load Balancing Parallel Computations

Nashat Mansour
School of Computer Science
Center for Computational Science
Syracuse University
Syracuse New York 13244

Geoffrey C. Fox
School of Computer Science
Department of Physics
Center for Computational Science
Northeast Parallel Architectures Ctr.
Syracuse University

Abstract

We present a new approach to balancing the workload in a multicomputer when the problem is decomposed into subproblems mapped to the processors. It is based on a hybrid genetic algorithm. A number of design choices for genetic algorithms are combined in order to ameliorate the problem of premature convergence that is often encountered in the implementation of classical genetic algorithms. The algorithm is hybridized by including a hill climbing procedure which significantly improves the efficiency of the evolution. Moreover, it makes use of problem specific information to evade some computational costs and to reinforce favorable aspects of the genetic search at some appropriate points. The experimental results show that the hybrid genetic algorithm can find solutions within 3% of the optimum in a reasonable time. They also suggest that this approach is not biased towards particular problem structures.

1. INTRODUCTION

Equal distribution of workload in multiprocessors is central to achieving a high utilization of the computational resources. This is why appropriate load balancing methods are needed for decomposing problems and assigning subproblems to processors. In distributed memory multiprocessors, henceforth called multicomputers, load balancing aims for the minimization of the total execution time of a problem by balancing the calculations across the processors and minimizing the interprocessor communication. A static implementation of load balancing methods is referred to as domain decomposition. In this work, we concentrate on the domain decomposition problem which is based on partitioning the underlying data set constituting the problem domain.

The domain decomposition problem is an NP-complete resource allocation problem. Several heuristic methods have been proposed, such as greedy algorithms, mincut-based heuristics, orthogonal recursive bisection, scattered decomposition, neural networks, and simulated annealing [1, 5, 7, 9, 11, 12, 19, 21]. The deterministic methods have predictable and low execution time. However, they, naturally, either make restrictive assumptions or tend to be biased towards particular structures of the problem domain. The stochastic methods make no assumptions about the domain considered; but require considerably greater execution time. The theory of complex systems has been suggested as a framework within which concurrency issues such as load balancing can be studied [8, 12]. Moreover, physical computation has been advocated for describing, simulating and solving complex systems, especially intractable optimization problems [13]. It should be emphasized here that all the approaches mentioned above, as well as our approach, aim at producing good sub-optimal solutions, and not necessarily the optimal, in an acceptable time.

In this work, we present a hybrid genetic algorithm (HGADD) as an evolutionary, physical and stochastic, method for domain decomposition. HGADD enhances the classical genetic algorithm (GA) with a number of features in order to alleviate the problem of premature convergence and to improve the evolution efficiency. Hybridization is a result of the addition of a problem-specific hill climbing procedure performed by the individuals in the population. The results of testing HGADD on realistic problems are favorable and suggest that it

can be applied to various problem domains and does not have a particular bias.

This paper is organized as follows. Section 2 describes the domain decomposition problem and defines an objective function. Section 3 presents HGADD and explains its constituents. The experimental results are given in section 4 and are discussed in sections 4 and 5. Section 6 concludes the paper.

2. DOMAIN DECOMPOSITION PROBLEM

Domain decomposition consists of partitioning the problem domain into subdomains and assigning them to the processors of the multicomputer such that an objective function is minimized. An objective function associated with the total execution time required for solving a problem is given below. The computational model is explained first, then exact and approximate objective functions and their assumptions are presented. Some aspects of the problem which will be utilized by HGADD are also given.

The model of computation considered here is that of loose synchronicity [12] with all processors running the same code (algorithm) and data being divided into subdomains. In this model, processors repeat a calculate-communicate cycle, where each processor performs calculations on its subdomain and then communicates with other processors to exchange necessary boundary information. The total execution time is determined by the slowest processor. Loose synchronicity is applicable to many science and engineering problems [12].

To formulate an objective function representing the cost of a decomposition, both the problem domain and the multicomputer are considered to be graphs. The vertices of the problem graph are the data elements and the edges refer to the calculation dependency. The vertices of the multicomputer graph are the processors and the edges are given by the interconnections. Domain decomposition becomes a mapping of subsets of the vertices of the problem graph to vertices in the multicomputer graph. Let w(i) denote the calculation time for a data element i, c(i,j) denote the amount of information to be exchanged between elements i and j, tc(p,q) denote the time taken for a unit of information to be communicated from processor q to p. The amount of calculation W(p) and the amount of communication C (p) for a processor p are given by

$$W(p) = \sum_i w(i)\,\delta(i,p), \text{ and}$$

$$C(p) = \sum_q tc(p,q) \sum_{i,j} c(i,j)\,\delta(i,p)\,\delta(j,q)$$

respectively, where $\delta(i,p)$ equals 1 if element i is mapped to processor p and equals 0 otherwise. The expression for C(p) assumes that messages are so large that the set-up time can be ignored. The total execution time, T, for a parallel program is determined by the processor with the greatest load of calculation and communication, that is

$$T = \max_p \{W(p) + C(p)\} \quad \ldots\ldots\ldots (1)$$

Equation (1) represents the exact objective function to be minimized subject to the constraint that the sum of elements allocated to all processors is equal to the total number of elements in the data set. This equation is the basis for evaluating the results of HGADD. The performance measure will be the efficiency of the decomposition; defined as the ratio of the sequential execution time to the product of T and the number of processors in the multicomputer. However, the use of this minimax criterion is computationally expensive mainly because the calculation of a new T caused by any change in the mapping of elements to processors may require the calculation of the loads of all processors. To avoid such excessive calculations, a quadratic objective function has been proposed [7, 10, 21] to approximate the cost of a decomposition. The optimal decomposition approximately corresponds to the minimum of

$$r^2 \sum_p N^2(p) + v\left(\frac{tcomm}{tcalc}\right) \sum_{p,q} d(p,q) \quad \ldots.(2)$$

where r is the amount of calculation per data element (a characteristic of the algorithm), N(p) is the number of elements allocated to processor p,

(tcomm/tcalc) is the ratio of the time needed to communicate a unit of information one unit distance to the time required for one calculation operation (a characteristic of the machine), v is a constant scaling factor expressing the relative importance of communication with respect to calculation, and d(p,q) is the Hamming distance between processors p and q. The objective function in expression (2) does not take into account the concurrency in performing communication among processors, but it still leads to a good approximation to the cost of a decomposition. Clearly, the first term is minimal when the calculational load is as evenly distributed among processors as possible, and a minimum of the second term means that the sum of all interprocessor communication is minimized. The main advantage of using this quadratic cost function is that it enjoys the locality property. Locality means that a change in the cost due to a change in the assignment of elements to processors is determined by the reassigned elements only. Since HGADD incorporates a hill-climbing procedure based on incremental reassignment of elements, the locality property becomes very important for keeping hill-climbing as fast as possible. Another important consideration in using the objective function in (2) is the choice of the weight v. In this work, values for v are chosen in harmony with the behavior of HGADD for the purpose of generating better quality solutions. This is elaborated in the next section within the HGADD context.

Two parameters derived from the objective function are utilized by HGADD. The first is the degree of clustering (DOC) of the data elements in a domain decomposition instance. DOC is the inverse of the number of units of information that are exchanged by the processors. Thus, it is inversely proportional to the sum of distances term in (2) with every distance equal unity. A smaller value of the average number of the units of communicated information implies a smaller value of the communication term in (2), a better decomposition, and a higher DOC. The maximum DOC corresponds to optimal decompositions of the data set; provided that only nearest-neighbor communication occurs. For irregular domains, a rough estimate for the maximum degree of clustering can be shown to be

$$DOC(max) = (1/(4\alpha\sqrt{Ne/P})) \ldots (3)$$

where Ne is the problem size, P is the multicomputer size, and $0 < \alpha < 1$ is a weighting factor which increases with larger granularity (Ne/P) and decreases with domain irregularity. The second parameter for HGADD is a near-optimal value for the objective function of a decomposition. Using *DOC(max)* in the communication term in expression (2) and (Ne/P) in the calculation term, a rough approximation for this parameter can be written as

$$r^2 P (Ne/P)^2 + 4v\left(\frac{tcomm}{tcalc}\right) P\alpha\sqrt{Ne/P} \ldots (4)$$

DOC and expression (4) are employed by HGADD, as explained in the next section, for evading some computational costs and reinforcing some aspects of the evolution.

3. GENETIC ALGORITHM

3.1 BACKGROUND

Genetic algorithms are search techniques based on natural evolution, where species search for adaptations to a changing environment. Adaptation occurs over successive, often discontinuous, generations. Each generation consists of a population of individuals (chromosomes), which are candidate solutions. The initial generation is generated randomly. The next generation is always created by the individuals climbing adaptive peaks in parallel. Firstly, individuals reproduce according to their fitness. Then, mates are selected and genetic operators are employed to create offsprings, which replace their parents. In this process, high-performance building blocks are expected to be propagated and combined to find better structures, i.e. solutions. Eventually, optimal or near-optimal solutions are expected to evolve.

3.2 HGADD

Genetic algorithms represent powerful weak methods for solving optimization problems, such as domain decomposition, by providing search strategies

with a reasonable balance between exploration of the search space and exploitation of the better solutions generated. For a number of reasons, however, the implementation of GA's often encounters the problem of premature convergence to local optima, otherwise a long time may be required for the evolution to reach an optimal or near-optimal solution. Methods for overcoming the two problems of premature convergence and inefficiency would be conflicting and a compromise is usually required. To alleviate premature convergence, a number of techniques have been suggested, dealing with the selection schemes, and the genetic operators and their rates [2, 3, 4, 14, 15]. The advantages of these techniques have been demonstrated by comparing the resulting performance with that of the classical GA [16]. Often, the performance verification is carried out for DeJong's testbed of functions or for other specific applications, such as the traveling salesperson problem. In this work, a number of techniques dealing with selection and genetic operators have been combined for producing good quality solutions for the domain decomposition problem. Also, a hill-climbing procedure tailored to our application is added for improving the efficiency of the search, resulting in a hybrid GA. The techniques and the procedure comprise HGADD which is outlined in Figure 1. In the remainder of this section, the constituents of HGADD are explained. An illustration of the stages of the HGADD search is given in the beginning as a prelude to the description of the design choices that aim for enhancing appropriate aspects of the genetic search.

```
Read (problem graph and multicomputer graph);
Random Generation of initial population P(0) of size POP;
Evaluate fitness of individuals in P(0);
For (gen = 1 to maxgen) OR until convergence do
   Set (v, operator rates);
   Rank individuals in P(gen-1), and
        allocate reproduction trials stored in MATES[];
   /* produce new generation P(gen) */
   For (i = 1 to POP step 2) do
      Randomly select 2 parents from MATES [];
      Apply genetic operators;
      Hill-climbing by new individuals;
   endfor
   Evaluate fitness of individuals in P(gen);
   Retain the better of {fittest(gen) , fittest(gen-1)};
endfor
Solution = Fittest.
```

Fig. 1 An Outline of HGADD.

(i) Three Stages of Evolution

In the beginning of the evolution, the assignment of data elements to processors is almost random and, thus, the communication among processors would be heavy and very far from optimal regardless of the distribution of the number of elements. In the successive generations, clusters of elements are expected to be gradually grown and assigned to processors such that the interprocessor communication is constantly reduced, at least in the fitter individuals in the population. Then, at some point in the search, the balancing of the calculational load becomes more significant for increasing the fitness. Therefore, two stages of evolution can be distinguished. The first stage is the clustering stage which lays down the foundations of the basic pattern of the interprocessor communication. The second stage will be referred to as the calculation-balancing stage. Obviously, the two successive stages overlap.

A third stage in the evolution can also be identified when the population is near convergence. In this advanced stage, the average DOC of the population approaches *DOC(max)*, defined in equation (3), and the clusters of elements crystallize. If these clusters are broken, the fitness of the respective individual would drop significantly and its survival becomes less likely. At this point, crossover becomes less useful for introducing new building blocks, mutation of elements in the middle of the clusters is useless and a fruitful search is that which concentrates on the adjustment of the boundaries of the clusters in the processors. This stage will henceforth be referred to as the tuning stage. Boundary adjustment can be accomplished mainly by the hill-climbing of individuals, which is explained below, aided by the probabilistic mutation of the boundary elements. The main responsibility of crossover becomes the propagation and the inheritance of high-performance building blocks and the maintenance of the

drive towards convergence for the sake of efficiency. For hill-climbing and mutation to take on their roles in this stage, it is necessary to increase the relative weight of the calculation term in the fitness function. This is elaborated below with the description of hill-climbing.

(ii) Chromosomal Representation

An instance of domain decomposition is encoded by a chromosome whose length is equal to the number of data elements (vertices) in the problem graph. The value of an allele is an integer representing the processor identification number to which a data element is allocated. The element is, therefore, the index (locus) of the processor (gene) to which it is assigned.

(iii) Fitness Evaluation

The fitness of an individual in any generation is evaluated as the inverse of the objective function in expression (2). The goal of HGADD is to find an individual with maximal fitness. As pointed out in section 2, the choice of v is of particular interest. Its value should be chosen in accordance with the properties of the evolution in different stages. That is, v is chosen to favor the fitness of the individuals whose structure involves nearest-neighbor interprocessor communication in the clustering stage. In the later stages, the value of v should allow the emphasis to shift to the calculation term in the fitness taking into account the basic interprocessor communication pattern that has already been laid out. A value for v which satisfies these requirements can be determined from the approximate form of the optimal objective function given in expression (4) by considering the ratio of its communication and calculation terms. In subsection 3.2(vii), it will be argued that v has to be decreased in the tuning stage.

(iv) Reproduction Scheme

The reproduction scheme adopted in HGADD is elitist ranking followed by random selection of mates from the list of reproduction trials, or copies, allocated to the ranked individuals. In ranking [2], the individuals are sorted by their fitness values and are allocated a number of copies according to a predetermined scale of equidistant values for the population, and not according to their relative fitness. In HGADD, the ranks assigned to the fittest and the least fit individuals are 1.2 and 0.8, respectively, resulting in a survival percentage of 92% to 98%. This scheme offers a suitable way for controlling the selective pressure and, hence, the convergence of the population.

Elitism in the reproduction scheme refers to the preservation of the fittest individual. In HGADD, the preceding fittest individual is passed unscathed to the new generation, but it is forced to compete with the new fittest and only the better of the two is retained. The purpose of elitism and its current implementation is ensuring that good candidate solutions are saved if the search is to be truncated at any point, and preventing the complete loss of good building blocks. To patch up a part of the loophole created by the use of the approximate objective function, the criterion for choosing between the current fittest and the preceding fittest individuals is changed in the tuning stage. The exact expression for fitness is used and has been found beneficial.

(v) Genetic Operators

The Genetic operators employed in HGADD are crossover, mutation and inversion. The two-point ring-like crossover is used because it offers less positional bias than the one-point standard crossover without introducing any distributional bias [6]. Other more complex and presumably higher-performance crossover operators have not been used in this work in order to avoid significant additions to the computational complexity.

The standard mutation operator is employed in the first two stages of evolution. In the tuning stage, for the reason explained in subsection 3.2(i), mutation is restricted to elements at the boundaries of the clusters.

Inversion is used in the standard biological way, where a contiguous section of the chromosome is inverted. In HGADD, the chromosome is considered as a ring. Inversion at a low frequency helps in introducing new building blocks into the population for an application such as domain decomposition.

(vi) Operator Rates

Variable operator rates are useful for maintaining diversity in the population and, hence, for alleviating the premature convergence problem [3,4]. Rates are varied in the direction that counteracts the drop in diversity. Several Measures have been suggested for the detection of diversity, but their evaluation invariably requires considerable computations [2, 4, 14]. In HGADD, this cost is not incurred. Instead, the degree of clustering (DOC) is used to guide the variation of the rates of the genetic operators since the DOC approximately follows diversity. This design decision is based upon the observation that diversity is reduced in the population as the clustering of elements increases.

(vii) Hill-Climbing

Since genetic algorithms are blind, the addition of problem-specific information helps direct the search to more profitable adaptive peaks in the landscape [15]. In HGADD, individuals carry out a simple hill-climbing procedure that can increase their fitness. The procedure is greedy and allows the transfer of data elements from overloaded processors to underloaded ones. Its inclusion improves the efficiency of the search significantly.

Hill-climbing for an individual is performed by considering only the boundary data elements allocated to processors, one element at a time. A boundary element e is an element that is allocated to a processor p1 and has at least one neighboring element (in the problem graph) allocated to a different processor p2. Such an element is transferred from p1 to p2 if and only if the transfer causes the objective function to drop or stay the same. It can be shown that the Change in Objective Function, COF, due to the transfer of element e is given by

$$2r^2[1+N(p2)-N(p1)]+2vR(CCD)$$

where N(x) is the number of elements allocated to processor x before the transfer, R is the (tcomm/tcalc) ratio, and CCD is the change in communication cost (sum of distances) for element e. From this expression, it can easily be seen that a transfer of an element can only take place from overloaded processors to underloaded processors. It should be emphasized here that the formulation of COF, which leads to a simple implementation of hill-climbing, is a direct result of the locality property of the approximate objective function mentioned in section 2.

Hill-climbing plays a distinctive role in the tuning stage of the search. In this stage, hill-climbing fine-tunes the structures by adjusting the boundaries of the clusters assigned to the processors. Since the basic pattern of interprocessor communication can not be significantly changed in this advanced phase and since the search ceases to offer significant gains at this point, the emphasis upon balancing the calculational load should be artificially increased for the purpose of facilitating the boundary adjustment. This is achieved by decreasing the value of the weight v in the objective function gradually from the fixed value used throughout the search to a small suitable value determined by the COF expression. The smallest useful value for v is that which makes COF negative or zero when the following conditions coexist. The first condition is that an overloaded processor has two elements more than the underloaded processor. The second condition is that the transfer of an element e does not increase the sum of communication distances of e by more than one.

4. EXPERIMENTAL RESULTS

The results described here illustrate typical solutions that can be obtained by HGADD. They also compare some of the design parameters of HGADD with those used in classical GA's.

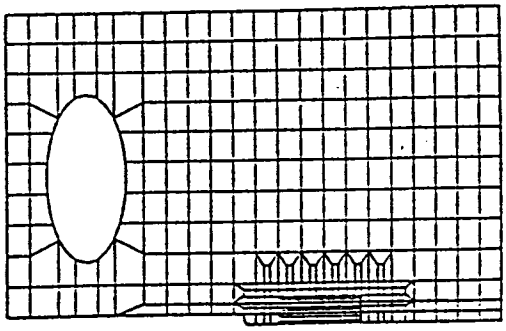


Fig. 2 551-element Grid1.

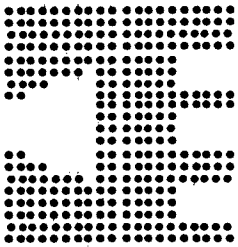


Fig. 3 301-element Grid2.

Several test cases have been employed. For small and regular problems, HGADD has always found optimal decompositions efficiently. These results are not be presented here. However, two irregular problems with realistic sizes are considered. These are shown in Figures 2 and 3 and are henceforth referred to as Grid1 and Grid2, respectively. In all experiments, a solution refers to the decomposition corresponding to the fittest individual. The results given here are the averages of three runs. The performance measures are the efficiency of the decomposition of the fittest individual and the average fitness of the population. Both measures are plotted below with respect to the number of generations, which, in its turn, is used to assess the efficiency of the search. The efficiency is based on the exact objective function (equation (1)). For clarity, the results are given as ratios, where efficiency is normalized with respect to the (exact) optimum and fitness is normalized with respect to the (approximate) optimal fitness (from expression (2)). It should be understood that the use of exact efficiency and approximate fitness for expressing the quality of the solutions will obviously exhibit a discrepancy in the results for the two measures.

The following parameters are used for HGADD. The maximum rank for the ranking-based selection scheme is 1.2. The population size is 500 for Grid1 experiments and 300 for Grid2. These values have been empirically chosen to be approximately equal to the length of the chromosome. Operator rates vary in a stepwise fashion as follows. Crossover rate increases from 0.5 to 1.0, mutation rate increases form 0.002 to 0.004, and inversion rate decreases from 0.03 to 0.0.

4.1 Results

The first experiment only refers to Grid1. All the following experiments refer to the decomposition of Grid2 for an 8-node hypercube.

(i) The decomposition of Grid1 for a 16-node hypercube by HGADD is depicted in Figure 4. The efficiency of the decomposition is about 0.93 of the optimum. Its fitness is about 0.998 of the optimum. This solution is obtained after 280 generations. Each generation takes about 30 seconds on a SPARC 1 workstation. For Grid1 and a 4 by 4 mesh multicomputer, HGADD finds a solution with an efficiency ratio of 0.95 after 282 generations.

(ii) The decomposition of Grid2 for 3-cube by HGADD is shown in Figure 5. The evolution of the efficiency and the fitness is plotted in Figure 6. The relative average loads of calculation and communication are also shown. After generation 118, the search converges to a solution with an efficiency 0.97 of the optimal and a fitness ratio of 0.998. Each generation takes about 12 seconds. It can be seen from Figure 5 that HGADD does not strictly insist on assigning equal number of elements to processors. Instead, it emphasizes the balancing of the combined calculation and communication load, as required by the computational model. Another feature of the solution in Figure 5 is that processor 1 is allocated discontiguous subdomains. This is not

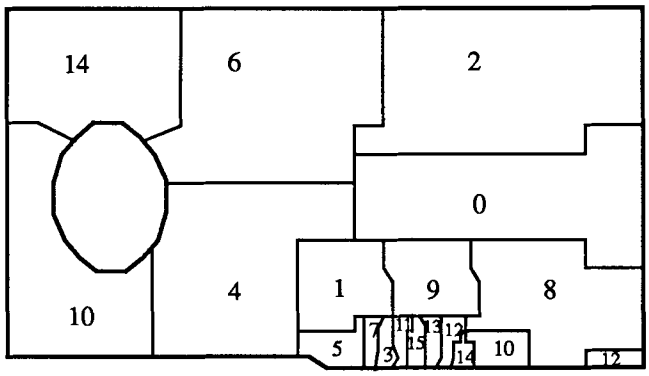
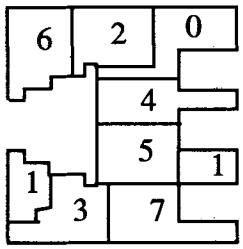


Fig. 4 Decomposition of Grid1 for 4-cube.



Proc	Calc	Comm
0	37	17
1	36	11
2	36	16
3	37	15
4	38	14
5	35	18
6	42	6
7	40	11

Fig. 5 Decomposition of Grid2 for 3-cube by HGADD, and processor loads.

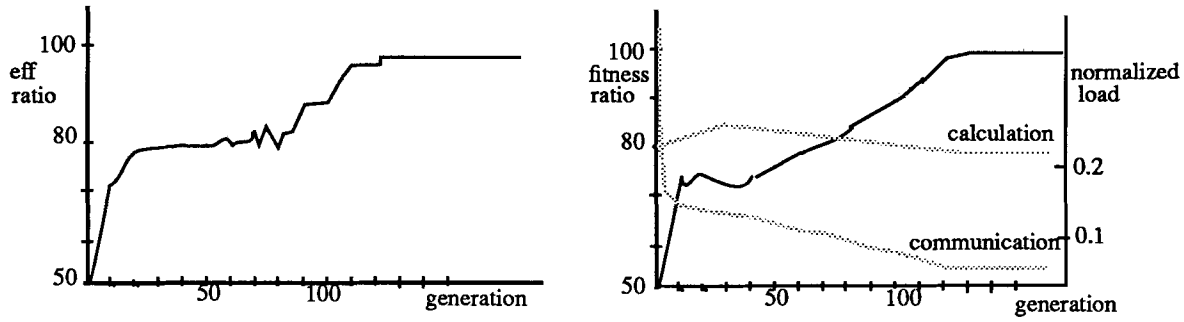


Fig. 6 Efficiency ratio and fitness ratio for HGADD.

necessarily bad in our model of computation. In fact, for many highly-irregular long-perimeter grids, an optimal decomposition can not be contiguous.

The three stages of the search can be identified in the fitness and workload curves in Figure 6. Roughly, their overlapping points are generations 50 and 100. It can be seen that in the first stage, the communication load drops steadily regardless of the calculation load which happens to increase. In the second stage, both loads decrease and the fitness rises. Decreasing v in the tuning stage enhances HGADD's tendency to reduce the calculation load. If v had not been decreased at this advanced stage, the efficiency would have been trapped at 89%.

(iii) HGADD is compared with a classical GA in Figure 7. GA1 uses 1-point crossover, normal mutation in all stages, no inversion, and fixed operator frequencies. However, it still employs ranking selection and is also hybridized. GA1 converges more rapidly to a lower quality solution with efficiency ratio of 91%. Clearly, the combined effect of 2-point crossover, boundary mutation in the tuning stage, inversion and variable rates is beneficial.

(iv) The effect of increasing the selection pressure is explored by increasing the maximum rank value to 2.0; as in HGADD2. This results in an early convergence as shown in Figure 8. HGADD2 finds a good solution (96% efficiency) in only 66 generations, which is 60% of the time required by HGADD to find a solution of the same quality. However, the large percentage of individuals (up to 20%) that die every generation, makes a maximum rank of 2.0 too high to be generally reliable for producing good solutions. This highlights the trade-off that exists between the solution quality and the search efficiency.

(v) The advantages of ranking based selection and hill-climbing have been noted by comparing HGADD with GA2 (roulette wheel fitness propor-

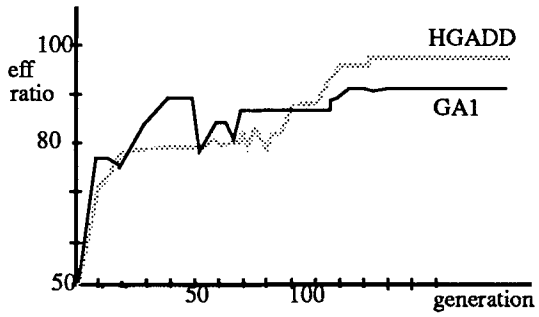


Fig. 7 Comparison of HGADD and GA1.

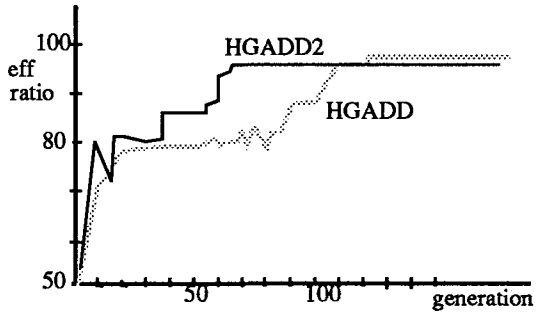


Fig. 8 Comparison of HGADD and HGADD2.

tionate selection with prescaling) and GA3 (without hill-climbing). GA2 loses population diversity much earlier and converges to a less favorable solution (94% efficiency) after 120 generations. GA3 is more than a hundred times slower than HGADD.

5. DISCUSSION

Some aspects of the experimental results are highlighted and discussed in this section. Also, remarks on some features of HGADD and its search efficiency are included.

The solutions obtained by HGADD are good suboptimal solutions. Since HGADD makes no assumptions about the structure of the problem or the interconnection network of the multicomputer, it is not biased towards any particular structures. Therefore, the good quality of the results described in section 4 can also be expected for any problem and any network. In all these results, the fitness of the population converges to the global (approximate) optimum. However, an important reason for not finding the optimal decomposition is the discrepancy between the approximate objective function guiding the adaptation of the individuals in the population and the exact objective function determining the actual solution quality. Nevertheless, the results obtained for Grid1 and Grid2 compare favorably with results obtained by other faster domain decomposition techniques. For example, recursive bisection [9] produces a decomposition for Grid2 whose efficiency is 87% of the optimum. Scattered decomposition [19] with a patch size of 4 yields an efficiency 61% of the optimum. Naive rectangular decomposition gives a 74% efficiency ratio. A qualitative comparison of HGADD solutions with those of simulated annealing and neural networks for Grid1 [7, 11] has also enhanced our confidence in the evolutionary approach.

HGADD is not restricted to the particular model of computation described in section 2. For example, data elements need not be of equal calculation requirements and the multicomputer need not be homogeneous. Other models of computation can easily be implemented in HGADD by modifying or replacing the objective function module. Moreover, the main constituents of HGADD can also be utilized for solving related problems such as mapping for production systems [22], Occam configuration [20], unstructured finite element meshes [21], and partitioned program modules [17].

It is worthwhile emphasizing some of the issues involved in determining the solution quality, the computational cost, and the trade-off between them. For example, it takes HGADD 118 generations to evolve a decomposition of 97% efficiency for Grid2. The evolution can be made faster by resorting to measures such as allowing the selection pressure to increase as in HGADD2 which yields a 96% efficiency in only 66 generations. This speed is accompanied with a clear rapid loss of diversity in the population. Therefore, the range of values of 1.2 to 2.0 for the maximum rank in the selection scheme allows the user to choose the desired compromise between solution quality and execution time. On the other hand, there does not seem to be a simple relation between the evolution time and the size of

the problem. Generally, it seems that doubling the problem size necessitates a similar increase in the population size. But, the amount of computation per individual will be almost doubled. Thus, the evolution time may become impractical for larger problem sizes. However, for large problems, sectors of data elements can easily and rapidly be formed before applying HGADD. Elements can be aggregated into sectors such that the total number of sectors is a multiple, K, of the number of processors in the multicomputer. K should be small enough to reduce the evolution time, but not too small otherwise it may become difficult to balance the load across the processors.

6. CONCLUSIONS & FURTHER WORK

The evolutionary approach of HGADD has led to good suboptimal solutions for static load balancing in parallel computing. The advantages of the design constituents of HGADD and the incorporation of application specific knowledge have been demonstrated for precluding premature convergence, improving the efficiency of the evolution, and avoiding excessive computations. Moreover, the results suggest that HGADD has no bias and is applicable to any problem structure and any interconnection network.

The performance of HGADD can be further improved. The linear variation of the rates of the genetic operators can be replaced by an adaptive variation related to population diversity. A more fruitful crossover operator, such as the reduced surrogate operator [3], can be used to enable the search to concentrate on useful work. However it should be clear that additional computational costs will be incurred for both suggestions. The evolution efficiency can be increased and better solutions might be generated by adding another pass to hill-climbing. The second pass would scan the boundary elements in the reverse order and the better result of the two passes would then be accepted. The reverse pass can fulfill its objective without adding significant computational costs if it is applied to selected individuals only, the fittest for example, in the later generations. Further work is also required for the optimization of the population size and the maximum rank in selection as a function of problem sizes, levels of solution quality, and execution time.

In comparison with other load balancing techniques, GA's are highly parallelizable. Significant speed-ups and increased robustness can be obtained by parallel algorithms based on HGADD [18].

Acknowledgement

This work was supported by the Joint Tactical Fusion Program Office and the National Science Foundation under Cooperative Agreement No. CCR-8809165.

REFERENCES

1. F. Andre, J-L. Pazat, and T. Priol, Experiments with Mapping Algorithms on a Hypercube, 4th Conf. Hypercube Conc. Comp. Applic., 1989, 39-46.

2. J. E. Baker, Adaptive Selection Methods for Genetic Algorithms, ICGA'85, 101-111.

3. L. Booker, Improving Search in Genetic Algorithms, in Genetic Algorithms and Simulated Annealing, ed. L. Davis, Morgan Kaufmann Publishers, 1987, 61-73.

4. L. Davis, Adapting Operator Probabilities on Genetic Algorithms, ICGA'89, 61-69.

5. F. Ercal, Heuristic Approaches to Task Allocation For Parallel Computing, Ohio State University, Ph.D Dissertation, 1988.

6. L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, Biases in the Crossover Landscape, ICGA'89, 10-19.

7. J. Flower, S. Otto, and M. Salama, Optimal Mapping of Irregular Finite Element Domains to Parallel Processors, Caltech Concurrent Computation Program #292b, 1987.

8. G. C. Fox, A Review of Automatic Load Balancing and Decomposition Methods for the Hypercube, in M. Shultz, ed., Numerical Algorithms for Modern Parallel Computer Architectures, Springer-Verlag, 1988, 63-76.

9. G. C. Fox, A Graphical Approach to Load Balancing and Sparse Matrix Vector Multiplication on the Hypercube, Caltech Concurrent Computation Program #327b, 1986.

10. G. C. Fox, A. Kolawa, and R. Williams, The Implementation of a Dynamic Load Balancer, Proc. 2nd Conf. Hypercube Multiprocessors, ed. Heath, 1987, 114-121.

11. G. C. Fox and W. Furmanski, Load Balancing Loosely Synchronous Problems with a Neural Network, Proc. 3rd Conf. Hypercube Concurrent Computers, and Applications, 1988, 241-278.

12. G. C. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, Solving Problems on Concurrent Processors, Prentice Hall, 1988.

13. G. C. Fox, Physical Computation, Int. Conf. Parallel Computing: Achievements, Problems and Prospects, Italy, June 1990.

14. D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Co., 1989.

15. J. J. Grefenstette, Incorporating Problem Specific Knowledge into Genetic Algorithms, in Genetic Algorithms and Simulated Annealing, ed. L. Davis, Morgan Kaufmann, 1987, 42-60.

16. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

17. K. Hwang and J. Xu, Mapping Partitioned Program Modules onto Multicomputer Nodes Using Simulated Annealing, ICPP 1990, Vol. II, 292-293.

18. N. Mansour and G. C. Fox, Parallel Genetic Algorithms with Application to Load Balancing, in preparation.

19. R. Morison and S. Otto, The Scattered Decomposition for Finite Elements, Caltech Concurrent Computation Program #286, 1985.

20. H. Motteler, Occam Configuration as a Task Assignment Problem, Transputer Res. Applic. 4, D. L. Fielding, Editor, 244-250, IOS Press, 1990.

21. R. D. Williams, Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations, Submitted to Concurrency Practice and Experience, 1990.

22. J. Xu and K. Hwang, Simulated Annealing Method for Mapping Production Systems onto Multicomputers, Proc. IEEE Conf. AI Applic., 350-356, 1990.