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. It is based on a genetic algorithm that combines a number of design choices in order to ameliorate the problem of premature convergence. The genetic algorithm is further 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 computational costs and to reinforce favorable aspects of the genetic search. The experimental results show that the hybrid genetic algorithm can find solutions that are very close to the optimum.

1: Introduction

Equal distribution of workload in multicomputers is central to achieving a high utilization of the computational resources. 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 data set.

Domain decomposition is NP-complete. Several heuristic methods have been proposed, such as mincut-based heuristics, recursive bisection, scattered decomposition, neural networks, and simulated annealing [3, 4, 5, 6, 12, 14]. The deterministic methods have predictable and low execution time. However, they 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 greater execution time. Physical computation has been advocated for describing, simulating and solving complex systems, especially intractable optimization problems such as load balancing [7]. It should be emphasized here that all the approaches mentioned above, as well as our approach, aim at producing good suboptimal solutions, and not necessarily the optimal, in an acceptable time. In this paper, we present a hybrid genetic algorithm (HGADD) as an evolutionary, physical and stochastic, method for domain decomposition. HGADD enhances the classic genetic algorithm (GA) with a number of features in order to alleviate the problem of premature convergence and to improve the evolution efficiency.

This paper gives a summary of our approach and results. A clearer presentation is found in [10]. The next section describes the domain decomposition problem. Section 3 presents HGADD and its constituents. Some experimental results are given in section 4 and are discussed in sections 4 and 5. Section 6 concludes the paper.

2: The 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. The model of computation considered here is that of loose synchronicity [6], where processors repeat a calculate-communicate cycle. The total execution time, T, for a parallel program is then determined by the processor with the greatest combined load of calculation, W, and communication, C, that is,

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

Equation (1) represents the exact objective function to be minimized and 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 and a quadratic objective function has been proposed [6, 14] for avoiding excessive computations. It can be expressed as

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

where r is the amount of calculation per data element, N(p) is the number of elements allocated to p, (tcomm/tcalc) is a machine-dependent communication to calculation ratio, v is a scaling factor expressing the relative importance of communication with respect to calculation, and d(p,q) is the Hamming distance. The main advantage of this quadratic cost function is its 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.

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, whose maximum is *DOC(max)*. The second parame-

0-8186-2290-3/91/0000/0200/$01.00 © 1991 IEEE

ter is an estimate of the optimal objective function, *OBJ(opt)*, of a decomposition. These parameters are employed by HGADD for evading some computational costs and for reinforcing some aspects of the evolution.

3: Genetic algorithm

Genetic algorithms represent powerful weak methods for solving optimization problems [8]. 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 are conflicting and a compromise is usually required. In HGADD, a number of design choices have been combined for producing good quality solutions. Also, a hill-climbing procedure tailored to our application is added for improving the efficiency of the search, resulting in a hybrid GA. HGADD is outlined in Figure 1, and its constituents are briefly described in the remainder of this section.

Three stages of evolution: In the beginning of the evolution, the assignment of data elements to processors is random and, thus, the communication among processors is heavy, regardless of the distribution of the data elements. In the successive generations, clusters of elements are expected to be grown gradually and are assigned to processors such that the interprocessor communication is constantly reduced. Then, at some point in the search, the balancing of the calculational load becomes more significant for increasing the fitness. Therefore, two overlapped stages of evolution can be distinguished; a clustering stage and a calculation-balancing stage. A third stage can also be identified when the population is near convergence. In this advanced stage, the average DOC of the population approaches *DOC(max)*, 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, a fruitful search would concentrate on the adjustment of the boundaries of the clusters in the processors. This stage is henceforth referred to as the tuning stage. Boundary adjustment can be accomplished mainly by hill-climbing of individuals, aided by the probabilistic mutation of boundary elements. The responsibility of crossover becomes the propagation of high-performance building blocks.

Chromosomal representation: An instance of domain decomposition is encoded by a chromosome whose length is equal to the size of the data set. The value of an allele is an integer representing the processor to which a data element is allocated.

Fitness evaluation: The fitness of an individual is the inverse of expression (2). The choice of v is of particular interest. It should be made 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

```
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.

stage. In the later stages, the value of v should allow the emphasis to shift to the calculation term in the fitness. Such a value can be determined from the ratio of the communication and calculation terms in *OBJ(opt)*.

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 [1], the individuals are allocated a number of copies according to a predetermined scale. In HGADD, the scale bounds are 1.2 and 0.8, 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.

Genetic operators: Two-point ring-like crossover is used because it offers less positional bias than the standard crossover. Standard mutation is employed in the first two stages of evolution. In the tuning stage, mutation is restricted to boundary elements. The inversion operator is included, where a contiguous section of the chromosome is inverted.

Operator rates: The operator rates are varied for maintaining diversity in the population and, hence, for alleviating the premature convergence problem [2]. DOC is used to guide the changes in the rates instead of expensive diversity detection metrics.

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. In HGADD, individuals carry out a simple hill-climbing procedure that can increase their fitness in every generation.

Hill-climbing for an individual is performed by considering only the boundary data elements allocated to processors. An element e is transferred from processor p1 to p2 if and only if the objective function drops or stays the same. It can be shown that the Change in Objective Function, COF, due to the transfer is given by

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

where $N(x)$ is the current number of elements in x, R is

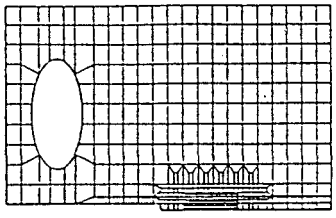


Fig. 2 551-element Grid1.

Fig. 3 301-element Grid2.

(tcomm/tcalc), 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 simple formulation of COF is a direct result of the locality property of the approximate objective function mentioned in section 2.

Hill-climbing plays a distinct role in the tuning stage of the search, where it adjusts the boundaries of the clusters assigned to the processors. In this stage, the basic pattern of interprocessor communication can not be significantly changed and the search ceases to offer significant gains. For these reasons, 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 fitness function.

4: Experimental results

The results described here illustrate typical solutions that can be obtained by HGADD. Two irregular problems with realistic sizes are considered as test cases. These are shown in Figures 2 and 3 and are henceforth referred to as Grid1 and Grid2, respectively. The performance measures are the exact efficiency of the decomposition (from equation 1) and the average fitness of the population (from expression 2). For clarity, the results are normalized with respect to the geometrically optimal values. It should be understood, however, 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 first result only refers to Grid1. All the other results 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 normalized efficiency of the decomposition is 0.93, and the normalized fitness is 0.99. This solution is obtained after 280 generations. Each generation takes about 30 seconds on a SPARC 1 workstation.

(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 ratio of 0.97 and a fitness ratio of 0.99. Each generation takes about 12 seconds. It can be seen from Figure 5 that HGADD emphasizes the balancing of the combined calcula-

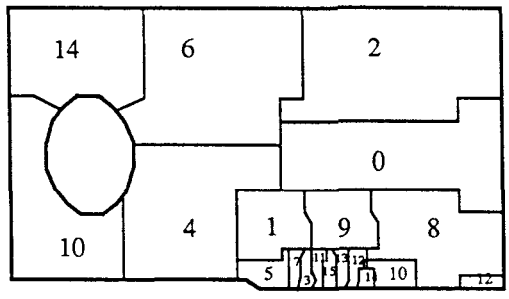
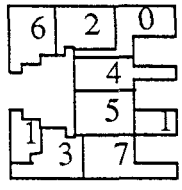


Fig. 4 Decomposition of Grid1 for 4-cube.



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.

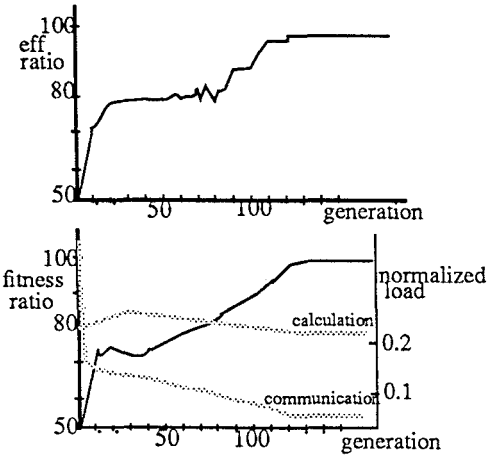


Fig. 6 Efficiency and fitness ratios for HGADD.

tion 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 necessarily bad in our model of computation. In fact, for many highly-irregular long-perimeter grids, an optimal decomposition can not be contiguous. Also, 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. Decreasing v in the tuning stage allows an increase in efficiency by about 8%.

(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%.

(iv) The effect of increasing the selection pressure has been explored by increasing the maximum rank from 1.2 to 2.0 in the selection scheme. This has led to early convergence to a surprisingly good solution (96% efficiency) in only 66 generations. However, the large percentage of indi-

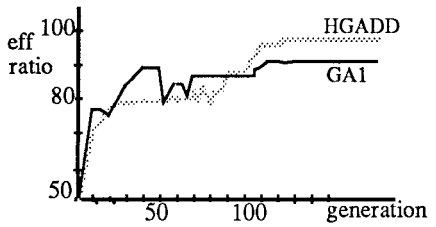


Fig. 7 Comparison of HGADD and GA1.

viduals (up to 20%) that die every generation, makes a maximum rank of 2.0 too high to be generally reliable for producing good solutions.

5: Discussion

Since HGADD makes no assumptions about the structure of the problem or the multicomputer, the good quality of the results described in section 4 can also be expected for general structures. In all these results, the fitness of the population converges to the global (approximate) optimum. However, an important reason for not finding an 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 in section 4 compare favorably with those obtained by other faster techniques. For example, recursive bisection [4] produces a decomposition for Grid2 whose efficiency is 87% of the optimum. Scattered decomposition [12] with a patch size of 4 yields an efficiency of 61%. Naive rectangular decomposition gives a 74% efficiency ratio.

HGADD is not restricted to the model of computation described in section 2. Other models can easily fit into HGADD by modifying the objective function module. Moreover, the main constituents of HGADD can be utilized for solving related problems such as Occam configuration [13], mapping unstructured finite element meshes [14], and mapping partitioned program modules [9].

It is worthwhile emphasizing the trade-off between the solution quality and the evolution time. The range of values of 1.2 to 2.0 for the maximum rank in the selection scheme allows several choices for a compromise between the two performance measures. On the other hand, larger problems require the aggregation of data elements into sectors before applying HGADD in order to make the evolution time practical. However, this makes load balancing more difficult.

6: Conclusions and further work

The evolutionary approach and the design constituents of HGADD have led to good suboptimal static load balancing. The results also suggest that HGADD has no bias and is applicable to general problem structures and interconnection networks.

The performance of HGADD can be further improved. A better crossover operator can be used to enable the search to concentrate on useful work [2]. The evolution efficiency can be increased and better solutions might be generated by adding a reverse pass for scanning the boundary elements in hill-climbing, and then accepting the better result of the two passes. A preliminary step, which groups data elements into sectors, can be added for significantly reducing the evolution time. Further work is also required for the optimization of the population size and the maximum rank in selection as a function of problem size, solution quality, and evolution 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 [11].

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] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms," *Int. Conf. Genetic Algorithms 85*, 101-111.

[2] L. Booker, "Improving Search in Genetic Algorithms," in *Genetic Algorithms and Simulated Annealing*, ed. L. Davis, Morgan Kaufmann Publishers, 1987, 61-73.

[3] F. Ercal, Heuristic Approaches to Task Allocation For Parallel Computing, Ohio State University, *Diss.*, 1988.

[4] G. C. Fox, "A Graphical Approach to Load Balancing and Sparse Matrix Vector Multiplication on the Hypercube," *Caltech C3P*-327b, 1986.

[5] G. C. Fox and W. Furmanski, "Load Balancing Loosely Synchronous Problems with a Neural Network," *Conf. Hypercube Conc. Comp., and Applic.*, 1988, 241-278.

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

[7] G. C. Fox, "Physical Computation," *Int. Conf. Parallel Computing: Achievements, Problems and Prospects*, Italy, June 1990.

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

[9] K. Hwang and J. Xu, "Mapping Partitioned Program Modules onto Multicomputer Nodes Using Simulated Annealing," *Int. Conf. Par. Proc.* 1990, Vol. II, 292-293.

[10] N. Mansour and G. C. Fox, "An Evolutionary Approach to Load Balancing Parallel Computations," *Technical Report* SU-CIS-91-13 and SCCS-25, Syracuse University.

[11] N. Mansour and G. C. Fox, "Parallel Genetic Algorithms with Application to Load Balancing," *in preparation.*

[12] R. Morison and S. Otto, "The Scattered Decomposition for Finite Elements," *Caltech C3P*-286, 1985.

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

[14] R. D. Williams, "Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations," Submitted to *Concurrency Practice and Experience*, 1990.