General Applicability of Genetic and Simulated Annealing Algorithms for Data Mapping

Jalal Kawash, Nashat Mansour
Lebanese American University
Computer Science Program
Beirut, Lebanon

Hassan Diab
American University of Beirut
Electrical Engineering Dept.
Beirut, Lebanon

Abstract

We experimentally analyze the general applicability of genetic algorithms (GA) and simulated annealing algorithms (SA) for mapping data to multicomputers. The results show that the GA and SA are insensitive to user parameters in wide ranges, completely fault tolerant, and unbiased towards particular multicomputer topologies. These properties of flexibility and general applicability, which are lacking in other heuristic algorithms, make the GA and SA attractive for automatic parallelization systems.

Key Words : *automatic parallelization, data mapping, load partitioning and balancing.*

1. Introduction

Let *ALGORITHM* be an algorithm intended to solve a problem with an underlying data set, *DATA*, on a multicomputer, *MCOMP*. We can construct a computation graph with *P* vertices representing the data points, and edges representing the data dependences specified by *ALGORITHM*. Data mapping refers to partitioning the computation graph and mapping the subgraphs to *N MCOMP* processors such that the total execution time of the parallel program associated with *ALGORITHM* is minimized. This problem is NP-complete, and several heuristics have been proposed to find sub-optimal solutions. [Berger and Bokhari 1987; Chrischoides et al 1991; Ercal 1988; Fox 1988; Simon 1991; Williams 1991]. Also simulated annealing (SA) and genetic algorithms (GA) have been adapted to the mapping problem [Mansour and Fox 1992, 1994a, 1994b]. The GA and SA are slower, but do not make apriori assumptions about the underlying problem. It has been claimed that these two algorithms are flexible and are applicable to various network topologies and incomplete architectures [Mansour and Fox 1992]. However, these claims have not been supported.

In this paper, we experimentally analyze the behavior of the GA and SA for data mapping. Our analysis has three dimensions: the degree of sensitivity of the GA and SA to their user-defined parameters, the capability to map to a partially faulty multicomputer, the applicability to various multicomputer topologies.

This paper is organized as follows. Sections 2 and 3 present the parameters used in the experimental analysis. Sections 4-6 present the experimental analysis and results. In section 7, we present our conclusions.

2. Objective Function Parameters Used in the Analysis

The execution time of the parallel *ALGORITHM* is determined by the slowest *MCOMP* processor and is typically represented by the following objective function

$$OF_{typ} = Max_n \left\{ W(n) + \sum_m C(n,m) \right\}$$

for loosely synchronous programs [Fox et al 1988]. *C(n,m)* is the cost of the communication between processors *n* and *m* [Bokhari 1990] and is given by

$C(n,m) = t_{flaot} . \left[\rho . B(n,m) + \sigma + \tau . H(n,m) \right]$ where ρ, σ, and τ are machine dependent parameters normalized with respect to t_{float}, the time for a floating point operation. ρ is the time needed to communicate one word, σ is the message start-up time, and τ represents the communication time per unit distance. *B(n,m)* is the number of vertices mapped to *n* and are boundary with *m*. *W(n)* is the computational load of processor *n* and is proportional to the size of the data subset mapped to *n* and to λ; λ is the number of computation operations per computation graph edge per iteration, which is *ALGORITHM*-dependent.

OF_{typ} is computationally expensive and is approximated by

$$OF_{appr} = \sum_n W^2(n) + \mu . \sum_n \sum_m C(n,m)$$

where μ is a scaling factor expressing the relative importance of the communication term with respect to the computation term. The quality of a mapping solution is defined as

$$\eta = \sum_n W(n) \Big/ N . OF_{typ} \ .$$

In this paper we study the sensitivity of the solution quality, given by η, and the time taken by the mapping algorithm, t_{map}, to

changes in the values of μ, λ, σ, ρ, and τ. Secondly, we explore the capability of GA and SA to map to an *MCOMP* with some faulty processors, which will be reflected in the evaluation of *W(n)* and *C(n,m)*. Thirdly, mapping to different *MCOMP* topologies is explored.

3. GA and SA parameters

The GA used is a distributed GA (DGA) [Mansour and Fox 1994b] which works with a population of candidate solutions over the nodes of a hypercube network and evolve over a number of iterations. Each iteration is composed of two phases: the drift phase where a local GA [Mansour and Fox 1994b] is applied for *DLEN* generations, and the migration phase where M% of deme's individuals migrate to other demes. DGA converges when there is no improvement in η for a number of generations, called convergence threshold (*CT*). Different migration schemes can be used leading to three DGA versions based on: one-way migration, two-way migration, and shifting balance scheme.

The SA mapping algorithm [Mansour and Fox 1992] starts with a randomly generated mapping configuration and uses the energy function OF_{appr}. It is based on perturbation snd cooling schemes that run over many iterations until convergence.

4. Sensitivity to User-Defined Parameters

Three test cases of different characteristics, which are shown in Figure 1, have been used in the experiments. For brevity, the results for DGA only are summarized in Table 1. The detailed results can be found in [Kawash 1994].

Figures 2 and 3 give examples for the parameters μ and *CT*. Note that the refernce values in Table 1are taken from [Mansour 1992].

5. Fault Tolerant Mapping

A mapping algorithm is said to be fault tolerant if it is capable of mapping *DATA* to an *MCOMP* with some missing (faulty) processors. Most of the data mapping heuristics are bisection-based methods [Chrisochoides et al. 1994], which are not fault tolerant. The GA and SA are fault tolerant, and our experimental results support this claim [Kawash 1994].

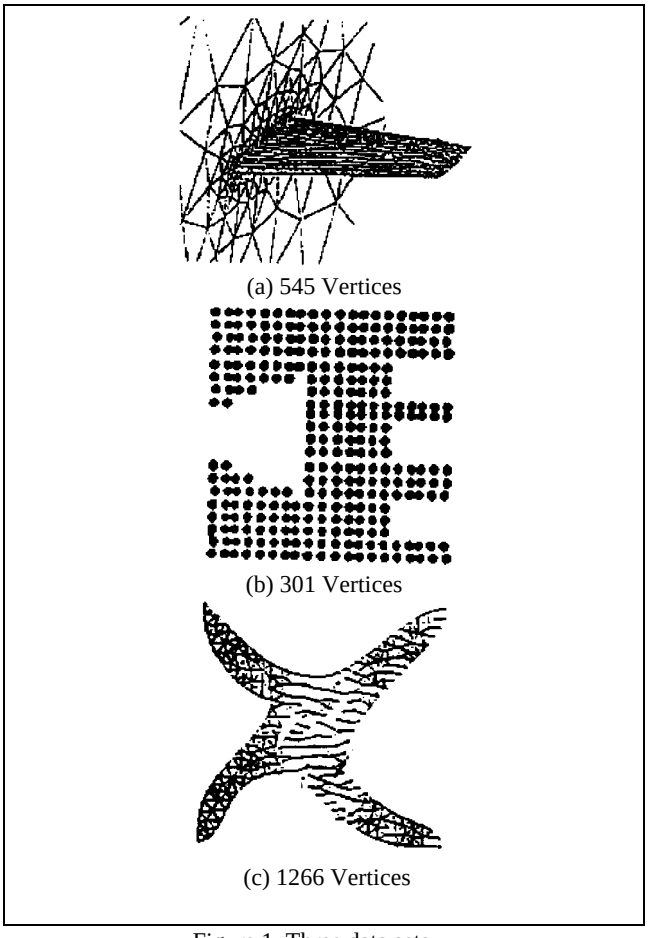
Examples are given in Figures 4 and 5 for the 545-vertx data and $|N| = 8$. Note that *F* denotes the set of faulty processors.

6. Mapping to Different Topologies

The applicability of DGA and SA to different multicomputer topologies is shown in this work by mapping to seven commonly used topologies with different properties: linear array, binary tree, ring, hypercube, 2-D mesh, 3-D mesh, and star graph [Hwang 1993; Misic and Jovanovic 1994; Day and Tripathi 1994]. The other heuristics do not enjoy this general applicability property. Figures 6-9 demonstrate that the DGA and SA have no bias towards particular multicomputer topologies.

7. Summary and Conclusion

A summary of our results is given in Table 2. These results show that the genetic and simulated annealing algorithms are flexible and of general applicability in mapping data to multicomputers. Hence, they should be considered for automatic parallel programming systems.



(a) 545 Vertices

(b) 301 Vertices

(c) 1266 Vertices

Figure 1. Three data sets.

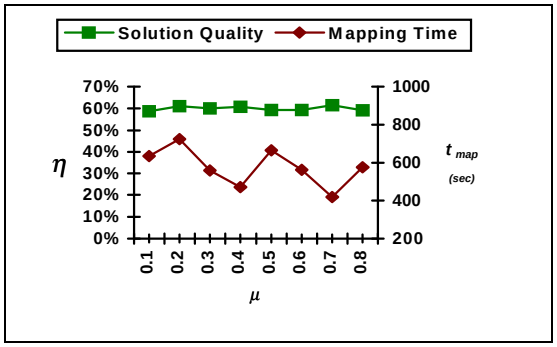


Figure 2. DGA results for different μ values.

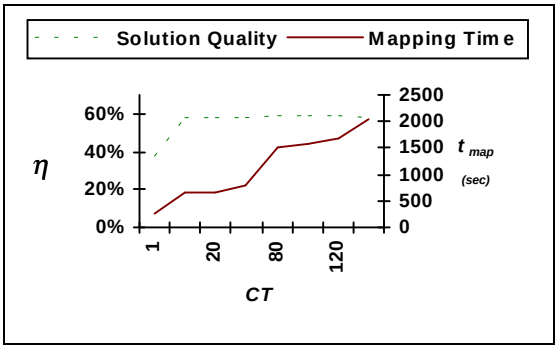


Figure 3. DGA results for different *CT*

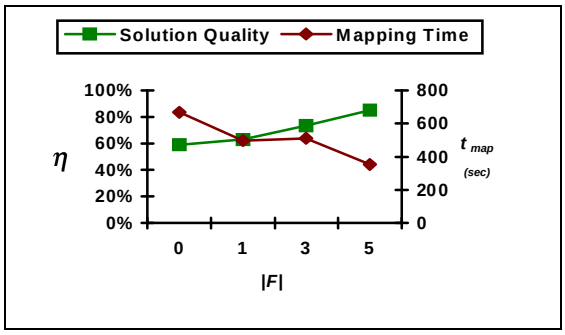


Figure 4. DGA results when mapping to incomplete multicomputers.

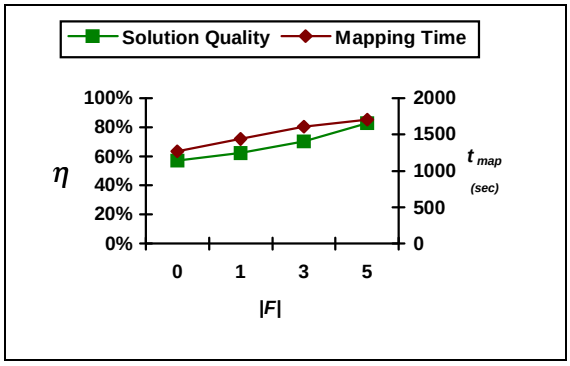


Figure 5. SA results when mapping to incomplete multicomputers.

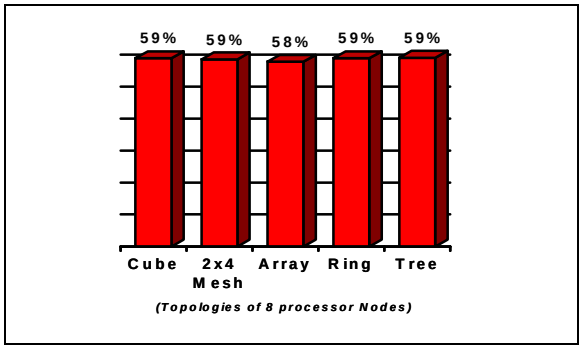


Figure 6. DGA mapping to different multicomputer topologies ($|N|= 8$).

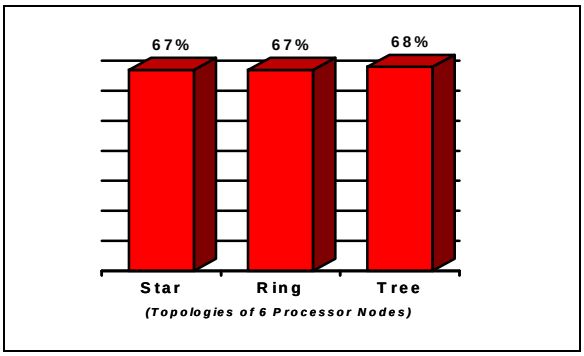


Figure 7. DGA mapping to different multicomputer topologies ($|N| = 6$).

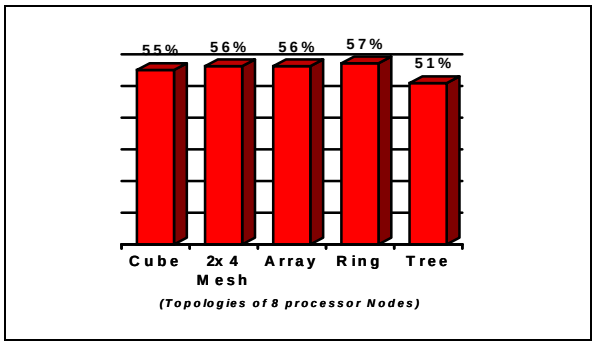


Figure 8. SA mapping to different multicomputer topologies ($|N| = 8$).

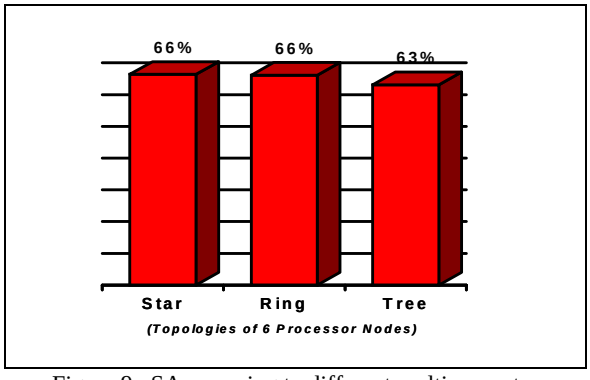


Figure 9. SA mapping to different multicomputer topologies ($|N| = 6$).

Parameter	Range	η	t_{map}	Reference Value
μ	[0.2, 0.8]	I	NU	0.5
λ	[-85%, 328%]	I	-	7
τ	[-90%, 200%]	I	-	100
σ	[-50%, 50%]	I	-	325
ρ	[-85%, 230%]	I	-	15
Population size	-	I	S	100
CT	≥ 15	I	S	20
Migration Scheme	-	I	I	1wDGA
M	10 - 50%	I	NU	30%
DLEN	[1, demesize]	I	NU	demesize/3

Table 1. Summary of DGA sensitivity results (I : Insensitive within ± 5%, S : Sensitive, NU : No Uniform behavior).

	DGA	SA
Sensitive to objective function parameter (μ)	NO	NO
Sensitive to architecture-dependent parameters (τ, σ, and ρ)	NO	NO
Sensitive to ALGORITHM-dependent parameter (λ).	NO	NO
Sensitive to its own parameters : - Population Size *- CT*	YES for t_{map} YES for t_{map}	- YES for t_{map}
Fault tolerant	YES	YES
Map to different multicomputer topologies	YES	YES

Table 2. Summary of results.

References

Berger M. and Bokhari S. 1987. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE trans. on Parallel and Distributed Systems*, 1 (1) , 91-106.

Bokhari S. 1990. Communication overhead on the INTEL iPSC-860 hypercube, ICASE Report no. 90-10.

Bultan T. and Aykanat C. 1992. A new mapping heuristic based on mean field annealing. *Journal of Parallel and Distributed Computing*, 16, 292-305.

Chrischoides N., Houstis E. N., and Houstis C. E. 1991. Geometry based mapping strategies for PDE computations. *Int. Conf. on Supercomputing*, Athens, 115-127.

Chrisochoides N., Mansour N., and Fox G.C. 1994. Performance evaluation of load balancing algorithms for parallel single-phase iterative PDE solvers. *Scalable High-Performance Computing Conference*, Knoxville, Tennessee.

Day K. and Tripathi A. 1994. A comparative study of topological properties of hypercubes and star graphs. *IEEE trans. on Parallel and Distributed Systems*, 5 (1), 31-38.

Ercal F. 1988. *Heuristic Approaches to Task Allocation for Paralell Computing*. Ohio State University, Ph.D.Thesis.

Fox G.C. 1988. A review of automatic load balancing and decomposition methods for the hypercube. In M. Schultz (ed.) *Numerical Algorithms for Modern Parallel Computers*, Berlin: Springer-Verlag, 63-76.

Fox G.C., Johnson M., Lyzenga G., Otto S., Salmon J., and Walker D. 1988. *Solving Problems on Concurrent Processors*, Englewood Cliffs, N.J: Printice Hall.

Fox G.C. and Furmanski W. 1988. Load balancing loosely synchronous problems with neural networks. *3rd Conf. Hypercube Concurrent Computers, and Applications*, 241-278.

Hwang K. 1993. *Advanced Computer Architecture: Parallelism, Scalability, Programability*, Singapore: McGraw-Hill.

Kawash J. 1994. *Sensitivity to Parameters and General Applicability of Genetic Algorithms and Simulated Annealing Algorithms for Mapping Data to Multicomputers*. M.Sc. Thesis, Lebanese American University.

Mansour N. 1992. *Physical optimization algorithms for mapping data to distributed-memory multiprocessors*. Syracuse, N.Y., Ph.D. diss., Comp. Sc.

Mansour N. and Fox G.C. 1992. Allocating data to multicomputer nodes by physical optimization algorithms for loosely synchronous computations. *Concurrency: Practice and Experience*, 4 (7), 557-574.

Mansour N., and Fox G.C. 1994a. Parallel Physical Optimization Algorithms for Allocating Data to Multicomputer Nodes. *The Journal of Supercomputing*, 8, 53-80.

Mansour N., and Fox G.C. 1994b. Allocating data to distributed-memory multiprocessor by genetic algorithms. *Concurrency: Practice and Experience*, 6 (6), 485-504.

Misic J., and Jovanovic Z. 1994. Communication Aspects of the star graph interconnection network. *IEEE trans. on Parallel and Distributed Systems*, 5 (7), 678-687.

Simon H. 1991. Partitioning of unstructured mesh problems for parallel processing. *Conf. Parallel Methods on Large Scale Structural Analysis and Physics Applications*, Permagon Press.

Williams R.D. 1991. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*, 3 (5), 457-481.