Proceedings of the 1992 Scalable High Performance Computing Conference, SHPCC '92, Williamsburg, VA, April 26-29, 1992.

HYPERGEN - A Distributed Genetic Algorithm on a Hypercube*

Leslie R. Knight
Roger L. Wainwright

Department of Mathematical and Computer Sciences
The University of Tulsa
rogerw@tusun2.mcs.utulsa.edu

Abstract

The genetic algorithm is a robust search and optimization technique based on the principles of natural genetics and survival of the fittest. Genetic algorithms (GA) are a promising new approach to global optimization problems, and are applicable to a wide variety of problems. HYPERGEN was developed as a research tool for investigating parallel genetic algorithms applied to combinatorial optimization problems. It provides the user with a wide variety of options to test the particular problem at hand. In addition, HYPERGEN is modular enough for a user to insert routines of his own for special needs, or for doing further research studies on parallel GAs. HYPERGEN was used successfully to find new "best" tours on three "standard" TSP problems, and out performed our parallel simulated annealing algorithm on various Package Placement Problems. We found it fairly easy to fine tune the parameters that drive a parallel GA for near optimal performance (population size, migration rate, and migration interval).

1: Introduction

The genetic algorithm (GA) designed by Holland [9] is a robust search and optimization technique based on the principles of natural genetics and survival of the fittest. The GA has the ability to create an initial population of feasible solutions, and then recombine them in a way to guide its search to only the most promising areas of the state space. GAs are applicable to a wide variety of problems. In particular, genetic algorithms are a promising new approach to global optimization problems. GAs have been used successfully for a wide range of applications areas including scheduling, design, control, machine meaning, robotics, combinatorial optimization, signal processing, image processing, economic forecasting, and medical applications.

Genetic algorithms differ from traditional algorithms in several ways [6]. The genetic algorithm works with a coding of the parameter rather than the actual parameter. The GA works from a population of strings instead of a single point. The genetic algorithm uses probabilistic transition rules, not deterministic rules, and the applications of the genetic operators causes information from the previous generation to be carried over to the next generation. In addition, genetic algorithms produce "close" to optimal results in a "reasonable amount of time", and they make no assumptions about the problems space. Finally, genetic algorithms are fairly simple to develop and they are suitable for parallel processing. Genetic algorithm packages for a single processor have been available for only a few years, GENITOR [20] and GENESIS [8] are two popular packages. Several researchers have investigated the benefits of solving combinatorial optimization problems using genetic algorithms [1,3,4,5,19].

2: A Distributed Genetic Algorithm

Several researchers have investigated distributed genetic algorithms on various architectures [2,7,11,14,15,17, 18]. These investigations have shown that parallel genetic algorithms work very well. Furthermore, parallel genetic algorithms have proved very successful in solving some NP-complete problems [5,12,19]. Empirical studies of parallel genetic algorithms far out number theoretical analysis [13].

HYPERGEN is a distributed genetic algorithm for a hypercube developed at The University of Tulsa. HYPERGEN distributes the initial population evenly among the processors. Each processor (island) executes a

* Research partially supported by OCAST Grant AR0-038 and Sun Microsystems, Inc.

sequential GA on its subpopulation performing crossover and mutation. HYPERGEN uses a one-at-a-time reproduction scheme, similar to GENITOR, where only a few members of the population are removed at each iteration. That is, the reproduction cycle consists of selecting two "fit" chromosomes from the population pool, performing crossover and perhaps mutation to yield a single offspring. The parents remain in the population pool. The offspring is placed into the population pool according to its fitness function, and the weakest chromosome is removed. The population pool size remains constant. After a prescribed number of reproductions, (called the *migration interval*) the "fittest" chromosomes in each processor are exchanged among other processors introducing new genetic material into each island. The amount of genetic material to exchange is called the *migration rate*. This process continues until the entire population stabilizes. Tanese [17] describes these issues in more detail.

HYPERGEN was developed as a research tool for investigating parallel genetic algorithms applied to combinatorial optimization problems. HYPERGEN is a modular collection of routines for generating the initial population, evaluation function, selection (based on a bias function), reproduction, mutation, migration interval, migration rate and summary statistics. The sequential GA on each processor and the periodic migration of genetic material between processors is performed automatically for the user. One of the design requirements for HYPERGEN is that the user can use the package without concerning himself with the hypercube topology, message passing, or other details of parallel processing.

To drive the HYPERGEN system, the user specifies the following parameters: Most of the parameters have defaults, if not specified. Of course, the user must provide the evaluation function, which defines the problem to be solved.

1. Input Dataset Describing the Problem
2. Output Dataset (optional)
3. Seed the Initial Population (optional)
4. Dimension of the hypercube
5. Random Number Seed
6. Population Size (per node)
7. Bias Parameter for Selection
8. The Migration Interval
9. The Migration Rate
10. How often to Collect and Report Statistics
11. Maximum Migrations Allowed
12. Select Crossover Function
13. Select Mutation Function

HYPERGEN allows the user to input a dataset describing the problem. For example, a sequence of coordinates describing locations of cities for the TSP problem. Results can be sent to a file of default to the screen. The user also has the option to seed the initial population or allow for a random initial population to be generated. The bias parameter (range from 1 to 2) is used to indicate how much more likely is the best chromosome to be selected over the median chromosome. This is similar to GENITOR. The migration interval, default of 20, specifies how many local reproductions are to occur in each node between migrations. The migration rate, default 20%, is the percent of the population in each node that is exchanged with another node. We are currently developing a feature that will allow for an adaptive migration interval and for an adaptive migration rate. Adaptive migration means the migration rate and interval are monitored and altered depending on the conditions of the population. For example different interval and rates may work better as the population matures. Genetic material is exchanged during each successive migration along the dimensions of the hypercube (hyperswap). For example in a five dimensional cube, node 0 on each successive exchange will swap genetic material with nodes 1,2,4,8,16 and so forth.

The user has the ability to select one of several system supplied crossover and mutation functions. The user also has the option to write his own crossover and/or mutation function and place it into HYPERGEN. An example of adaptive mutation is given by Whitley and Starkweather [18]. The hamming distance between two parents is monitored, and the more similar the parents, the more likely mutation is applied. We are presently studying the benefits of adaptive crossover. That is, some crossover schemes may work well on a rather young population, but not on a mature population.

3: Order-Based Genetic Algorithms

At the present time, HYPERGEN is implemented for order-based genetic algorithms. An order-based GA is where all chromosomes are a permutation of a list. Order-based GAs can be applied to a wide range of combinatorial optimization problems such as the follow-

ing classic problems: TSP, Bin Packing, Package Placement, Job Scheduling, Network Routing, Vehicle Routing, and various layout problems, etc. The HYPERGEN system supplied crossover functions for order-based GAs include Edge Recombination, Order Crossover #1, Order Crossover #2, PMX, Cycle Crossover, Position Crossover. These functions are described in Whitney *et al.* [18] and Starkweather *et al.* [16]. The Mutation functions available in HYPERGEN include swapping two elements, moving one element to another location, and reordering a sublist.

We tested HYPERGEN on two classic combinatorial optimization problems, Traveling Salesman Problem (TSP), and the One Dimensional Package Placement Problem (PPP). Results from this research can be applied to many other combinatorial optimization problems. We tested several "standard" TSP tours, that have been studied and analyzed before, The 30, 50, 75, and 105 city problems [18,19]. Table I compares each of the four TSP problems against the best known solutions to date for each. Since there has been some confusion how tour lengths should be calculated, we used all three of the techniques reported in previous research. In Table I, the first column of values under "Best Known" and "HYPERGEN" reports the Euclidian distance in real numbers. The second column of values rounds the edge distances as it sums them, and the third column rounds once after all the edges have been summed. HYPERGEN obtained the same best known results in the 105 city tour. However, better results were obtained in the case of the 30, 50 and 75 city tours. Table II indicates the specific parameters used in HYPERGEN for each of the tours. In each case the cube dimension was five, the Bias Rate was 1.2, the Edge Recombination crossover function was used, and the Mutation Rate was set to zero. We found that larger populations need to interact less often that smaller populations.

Table III shows the results of our experiments to date for the Package Placement Problem using 32, 64, and 128 packages. We tested these problems using our parallel simulated annealing algorithm [10], parallel genetic algorithm (HYPERGEN), and a sequential genetic algorithm developed using the GENITOR package. For comparison purposes the optimum and worst package arrangements are included. For the Package Placement Problem we tested several crossover functions: PMX, position, cycle, and order2, which are described by Whitley *et al.* [18], and an edge recombination crossover specially adapted for the Package Placement Problem. We tested each of these functions on the 32, 64, and 128 package problems. In each case the edge recombination crossover function yielded better results than any of the others. The parallel genetic algorithm performed consistently better than the parallel simulated annealing algorithm and the sequential genetic algorithm except in the 128 case. In this case GENITOR took over 10 hours to complete, while HYPERGEN took only 2 hours and on a slower processor. Given more time and fine tuning of the parameters, we are convinced HYPERGEN would obtain the same or better results than GENITOR. The sequential genetic algorithm given enough CPU time out performed the parallel simulated annealing algorithm in the package placement examples.

4: Summary and Future Enhancements

Several future enhancements are planned for HYPERGEN. Currently HYPERGEN picks the migrants skewed towards the more fit. Future strategies include (a) a uniform selection, and (b) the generation of "an over population", only during the migration step allowing the exchange of "extra" genetic material so nothing is lost from the sending population. The new arrivals are analyzed and placed into the population before it is reduced to the original size again. We will be investigating a special adaptive crossover and mutation strategy that is different on each processor depending on the maturity of the local population compared to the population as a whole.

In summary, HYPERGEN was developed as a research tool for investigating parallel genetic algorithms applied to combinatorial optimization problems. It provides the user with a wide variety of options to test the particular problem at hand. In addition, HYPERGEN is modular enough for a user to insert routines of his own for special needs, or for doing further research studies on parallel GAs. HYPERGEN was used successfully to find new "best" tours on three "standard" TSP problems, and out performed our parallel SA algorithm on various Package Placement Problems. We found it fairly easy to fine tune the parameters that drive a parallel GA for near optimal performance (population size, migration rate, and migration interval). Currently, HYPERGEN is being used as a tool in several research projects and has proved invaluable.

	Best Known			HYPERGEN		
30 City	423.912	420	424	423.741	420	424
50 City	430.857	427	431	427.855	426	428
75 City	545.493	538	545	542.039	535	542
105 City	14383	14379	14383	14383	14379	14383

Table I: Comparison of HYPERGEN TSP Results Verses the Best Known Results

Number of Cities	30	50	75	105
Pop. Size Per Node	32	50	200	225
Migration Interval	15%	30%	2%	1%
Migration Rate	17%	20%	10%	10%
Max. Migrations	100	5000	3000	3000

Table II: HYPERGEN Parameters

Algorithm	Number of Packages		
	32	64	128
Parallel SA	5884	50,462	397,940
Parallel GA	5712	44,704	375,556
GENITOR	5712	44,704	355,072
The Optimum	5712	44,704	353,600
The Worst	10912	87,360	699,008

Table III: Results of the Package Placement Problem

References

[1] D.E. Brown, C.L. Huntley and A.R. Spillane, "A Parallel Genetic Heuristic for the Quadratic Assignment Problem", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[2] J. Cohoon, S. Hegde, W. Martin, and D. Richards, "Punctuated equilibria: a Parallel Genetic Algorithm, *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, 1987.

[3] L. Davis, ed., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[4] L. Davis, ed., *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publisher, 1987.

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

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

[7] M. Gorges-Scheuter "ASPARAGOS: A Asynchronous Parallel Genetic Optimization Strategy", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[8] J. Grefenstette, GENESIS, Navy Center for Applied Research in Artificial Intelligence, Navy research Lab., Wash. D.C. 20375-5000.

[9] J.H. Holland "Adaptation in Natural and Artificial Systems", Ann Arbor: The University of Michigan Press, 1975.

[10] D.R. Mallampati, P.P. Mutalik and R.L. Wainwright, "A Parallel Multi-Stage Implementation of Simulated Annealing for the Traveling Salesman Problem", *Proceedings of the Sixth Distributed Memory Computing Conference*, April 28 - May 2, 1991.

[11] B. Manderick and P. Spiessens, "Fine-Grained Parallel Genetic Algorithms", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[12] H. Muehlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[13] C.C. Pettey and M.R. Leuze, "A Theoretical Investigation of a Parallel Genetic Algorithm", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[14] P. Spiessens and B. Manderick, "A Massively Parallel Genetic Algorithm Implementation and First Analysis", *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991.

[15] T. Starkweather, D. Whitley, and K. Mathias, "Optimization Using Distributed Genetic Algorithms," in *Parallel Problem Solving from Nature*, ed. H. Schwefel and R. Maenner, Springer Verlag, Berlin, Germany, 1991.

[16] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley and C. Whitley "A Comparison of Genetic Sequencing Operators", *Proceedings of the Fourth International Conference on Genetic Algorithms*, June, 1991.

[17] R. Tanese, "Distributed Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, ed. J.D. Schaffer, Morgan Kaufmann, 1989, pp. 434-439.

[18] D. Whitley and T. Starkweather, "GENITOR II: A Distributed Genetic Algorithm, *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1990) 189-214.

[19] D. Whitney, T. Starkweather, and D. Fuquat, "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator", *Proceedings of the Third International Conference on Genetic Algorithms*, June, 1989.

[20] D. Whitney and J. Kauth, GENITOR: A Different Genetic Algorithm, *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, Co., 1988, pp. 118-130.