## 6 Conclusion

In this paper, we report on various approaches used to reduce premature convergence in GA's. In particular we have categorized a number of approaches used in parallel architecture approaches and introduced a new architecture, the injection island GA. To examine the relative merits of these approaches, we compare them on a deceptive objective function that solves a graph partition problem. Although we applied a number of approaches that address premature convergence in sequential GA's, none of them could find the global optima in a fixed evaluation period. In the cgGA approach, all of the 25-subpopulation GA's find the optimal value 0, suggesting that search quality is positively correlated with number of subpopulations.

One of the most interesting results concerns migration schemes. Tanese [26] reported that the isolated island GA's found better solutions than island GA's with migration. The claim was that incompatible migration prematurely converged the subpopulations. We add more data to this debate. First, the isolated island GA's performed better than the non-elitist models, and nearly as well as some of the ring topologies, and so were at least partially effective. However, the distance-connected topologies, especially the positive-connection topology, further increased the system's performance. Thus we did not see a degradation in results due to migration, but saw instead an improvement in results using restricted migration, based on Hamming distance measures between subpopulations.

The results of our simple iiGA's appear quite promising. This heterogeneous approach takes full advantage of the flexibility of cgGA's. It clearly outperformed the positive-distance topology architecture (the best on the two 3-CCCs partition problem) on the more difficult four 3-CCCs partition problem. This approach demonstrates that the evolution of subpopulations under different selection mechanisms, crossover methods, mutation methods, and most especially encoding methods, as in the iiGA, holds some promise. We believe that the concept of a two-level search space, as proposed by De Jong [6], could be feasibly addressed using such heterogeneous cgGA's.

## References


[1] D. Ackley, *Stochastic Iterated Genetic Hillclimbing*, Carnegie Mellon University, 1987.

[2] J. Baker, "Adaptive Selection Methods for Genetic Algorithms," *Proc. First ICGA*, June 1985, pp. 101-111.

[3] J. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm," *Proc. Second ICGA*, Jul. 1987, pp. 14-21.

[4] R. Collins, "Selection in Massively Parallel Genetic Algorithms," *Proc. Fourth ICGA*, July 1991, pp. 249-256.Proc.

[5] J. Crow, *Basic Concepts in Population, Quantitative, and Evolutionary Genetic*, 1986.

[6] K. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, University of Michigan, 1975.

[7] K. De Jong, "Genetic Algorithms: A 10 Year Perspective," *Proc. First Int. Conf. on PPSN*, 1990, pp. 169-177.

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

[9] D. Goldberg and T. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization," *Proc. Second ICGA*, July 1987, pp. 41-49.

[10] D. Goldberg and P. Segrest, "Finite Markov Chain Analysis of Genetic Algorithms," *Proc. Second ICGA*, July 1987, pp. 1-8.

[11] D. Goldberg, "A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing," *Complex Systems*, 1990, pp. 445-460.

[12] F. Hoffmeister and T. Back, "Genetic Algorithms and Evolution Strategies: Similarities and Differences," *Proc. First Int. Conf. on PPSN*, 1990, pp. 447-461.

[13] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[14] G. Laszewski, "Intelligent Structural Operators for the k-way Graph Partitioning Problem," *Proc. Fourth ICGA*, July 1991, pp. 45-52.

[15] B. Manderick and P. Spiessens, "Fine-Grained Parallel Genetic Algorithms," *Proc. Third ICGA*, Jun. 1989, pp. 428-433.

[16] M. Mauldin, "Maintaining Diversity in Genetic Search," *Nat. Conf. on Artificial Intelligence*, 1984, pp. 247-250.

[17] H. Muhlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," *Proc. Third ICGA*, June 1989, pp. 416-421.

[18] C. Pettey, M. Leuze, and J. Grefenstette, "A Parallel Genetic Algorithm," *Proc. Second ICGA*, Jul. 1987, pp. 155-161.

[19] W. Punch, E. Goodman, M. Pei, L. Chai-Shun, P. Hovland and R. Enbody, "Further Research on Feature Selection and Classification Using Genetic Algorithms," *Proc. Fifth ICGA*, June 1993, pp. 557-564.

[20] J. Richardson, M. Palmer, G. Liepins, and M. Hilliard, "Some Guidelines for Genetic Algorithms with Penalty Functions," *Proc. Third ICGA*, June 1989, pp. 191-197.

[21] A. Smith and D. Tate, "Genetic Optimization Using A Penalty Function," *Proc. Fifth ICGA*, June 1993, pp. 499-503.

[22] C. G. Shaefer, "The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique," *Proc. Second ICGA*, July 1987, pp. 50-55.

[23] N. Schraudolph and J. Grefenstette, *A User's Guide to GAucsd 1.4*, July, 1992.

[24] N. Schraudolph and R. Belew, "Dynamic Parameter Encoding for Genetic Algorithms," *Machine Learning*, June 1992, pp. 9-21.

[25] T. Starkweather, D. Whitley and K. Mathias, "Optimization Using Distributed Genetic Algorithms," *PPSN*, 1991, pp. 176-185.

[26] R. Tanese, "Distributed Genetic Algorithms," *Proc. Third ICGA*, June 1989, pp. 434-440.

[27] D. Whitley, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials Is Best," *Proc. Third ICGA*, June 1989, pp. 116-121.

**TABLE 3.** Time needed to reach a global optimum and the speedup with different numbers of nodes

| Number of Nodes | 1 | 5 | 10 | 25 |
|---|---|---|---|---|
| Ave. Time (sec) | >408.8* | 45.6 | 21.8 | 8.0 |
| Speedup | 1 | >9.0 | >18.8 | >51.2 |

* 7 of 10 optima found, so average time to find all 10 is surely > 408.8.

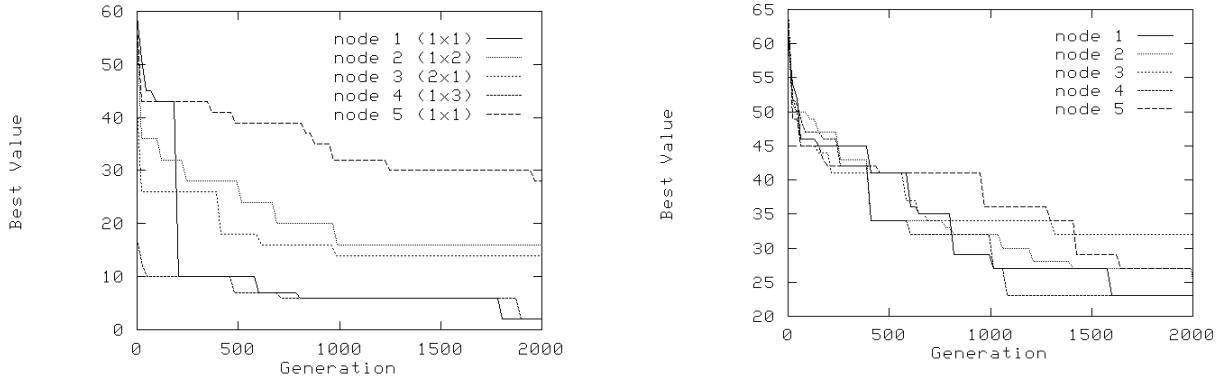
than other models because all nodes must complete 100 generations before migration, and thus the speed is limited by the slowest node.

The above experiments all used a fixed number of total evaluations in different numbers of nodes. The results don't show when the solution is reached. We are more interested in the speedup for a given search quality when running PGA's on different numbers of nodes. Since the single node experiments can't find the solution using the objective function we defined, *(PC, r) = (10, 1)*, we defined another setting *(PC, r) = (1, 1)* to make the function easier. We conducted the experiments in model 7, the positive-distance topology. With this setting, a single node can find a global optimum in 7 runs out of 10 runs and with nodes numbering greater than 1, can find a global optimum in each experiment. Table 3 shows the time needed to reach a global optimum and the speedup. Note that under "time-to-solution" we get a "superlinear" speedup. This is possible for some heuristic search algorithms like GA's because the total amount of work is different for different number of processors. The result shows that PGA's do improve the searching ability.

We can roughly rank the performance of these different models, but based on only 10 runs per model, cannot assign high confidence to the relative rankings. The distance topology approaches, both negative and positive, outperform the other models. The asynchronous island GA's with a ring topology and elitist strategy are ranked second most effective. The synchronous island GA's are third most effective. The isolated island GA's are fourth and the non-elitist GA's are last.

## 5.2 Four 3-CCCs Partition Problem and the iiGA Architecture

We demonstrate our new approach to PGA's, the injection island GA, on the more difficult four 3-CCCs partition problem. For this experiment we use the simple iiGA topology, which has each low-resolution node inject its result only into the highest resolution (1x1 blocksize) node. We use 5 nodes in this experiment, allocated as follows. We assign block sizes of 1x1, 1x2, 2x1, and 1x3 to nodes 1 through 4. We also allocate a fifth node, also with block size 1x1, as a control node. Only node 1 will receive solutions from other nodes. The subpopulation size for each node is 1,000 and the number of evaluations per node is 2,000,000. The interval of variation for f(x) is [0, 960]. The results are shown in Fig 6(A). We compare the iiGA to the best model from the two 3-CCCs partition problem, the positive-distance topology model, using the same parameters. The positive-distance results are shown in Fig 6(B). The best value found by the simple iiGA's is 2. The control node with no injection from other subpopulations, node 5, found a best of only 28. The best value found in the positive-distance model is 23. From the graph in Fig 6(A), node 1 gets a solution from node 4 around generation 200 and from there on, node 1's performance surpasses the control node, node 5. Since the search space of node 4 ($2^{32}$ for a 1x3 blocksize) is much smaller than that of node 1 ($2^{96}$ for a 1x1 blocksize), node 4 is likely to find a high-fitness individual more quickly. This may result in a high-fitness but incompatible solution being injected into node 1. Like the positive-distance topology architecture, it is our belief that a iiGA with a larger number of nodes on each level, exchanging individuals based on population similarity, won't have such a problem. We are pursuing the implementation to confirm this hypothesis.




**(A)** **(B)**

**FIGURE 6.**(A) Simple iiGA's and (B) Positive Distance Topology

**TABLE 2.** Experimental results for node number greater than 1. Model 1: Asynchronous ring, independent subpopulation. Model 2: Asynchronous ring, one uniform population. Model 3: Asynchronous ring, non-elitist, independent subpopulation. Model 4: Asynchronous ring, non-elitist, one uniform population. Model 5: Isolated island. Model6: Synchronous ring. Model 7: Positive-distance topology. Model 8: Negative-distance topology.

| Model | Number of Nodes | Ave. Online | Ave. Best | Best Value | Ave. Time(sec) | Processor Speedup |
|---|---|---|---|---|---|---|
| 1 | 5 | 70.65 | 8.6 | 8 | 976.38 | 5.46 |
| | 10 | 63.57 | 5.2 | 0 (2/10) | 480.56 | 11.08 |
| | 25 | 53.51 | 2.6 | 0 (6/10) | 282.28 | 18.87 |
| 2 | 5 | 70.12 | 7.4 | 6 | 948.66 | 5.61 |
| | 10 | 64.66 | 5.2 | 0 (3/10) | 485.20 | 10.98 |
| | 25 | 53.29 | 1.8 | 0 (7/10) | 294.88 | 18.06 |
| 3 | 5 | 74.24 | 12.7 | 10 | 963.67 | 5.51 |
| | 10 | 71.59 | 10.7 | 10 | 479.55 | 11.08 |
| | 25 | 66.41 | 8.8 | 8 | 194.09 | 27.38 |
| 4 | 5 | 73.69 | 12.4 | 8 | 957.73 | 5.55 |
| | 10 | 71.11 | 11.0 | 8 | 483.26 | 11.00 |
| | 25 | 66.25 | 8.4 | 8 | 192.94 | 27.55 |
| 5 | 5 | 68.52 | 7.8 | 6 | 962.90 | 5.53 |
| | 10 | 61.50 | 4.4 | 0 (3/10) | 480.20 | 11.09 |
| | 25 | 52.14 | 4.2 | 0 (3/10) | 223.84 | 23.80 |
| 6 | 5 | 70.77 | 8.2 | 6 | 1019.52 | 5.22 |
| | 10 | 64.33 | 6.7 | 0 (1/10) | 524.53 | 10.15 |
| | 25 | 54.24 | 3.0 | 0 (5/10) | 300.20 | 17.74 |
| 7 | 5 | 69.54 | 8.0 | 6 | 982.04 | 5.42 |
| | 10 | 64.18 | 4.2 | 0 (4/10) | 492.53 | 10.81 |
| | 25 | 54.51 | 0 | 0 (10/10) | 197.92 | 26.91 |
| 8 | 5 | 70.19 | 8.0 | 6 | 960.18 | 5.54 |
| | 10 | 65.86 | 5.4 | 0 (3/10) | 494.02 | 10.78 |
| | 25 | 56.30 | 1.2 | 0 (8/10) | 200.82 | 26.52 |

isolated island GA, which maintains roughly the same best for all subpopulation numbers.

Because our parallel environment consists of distributed workstations in campus computer labs, the machine load is usually heavier during the day than at night, and different machines have different loads. This makes runtime and speedup comparisons difficult. We measured the runtime by the average turn around time of each node. In Fig 5 we can still observe that increased processor number leads to approximately linear speed-up. That is, each node added to the system contributes a linear speed-up in the overall processing time if the number of nodes is smaller than population size. Note also that the performance at 25 processors in some models was degraded due to the communication overhead and the non-uniform load. The average running time of the synchronous island implementation is higher

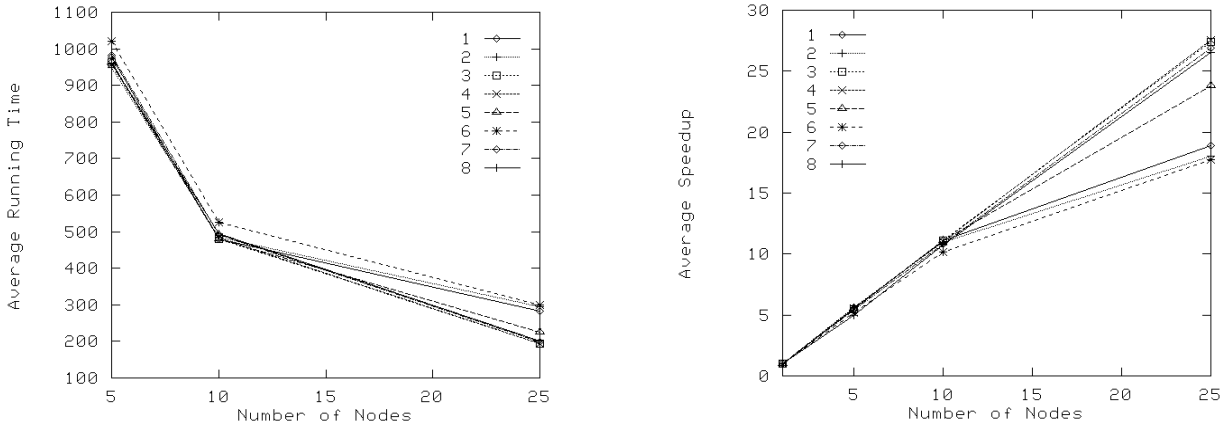



**FIGURE 5.**Average Running Time and Average Speedup

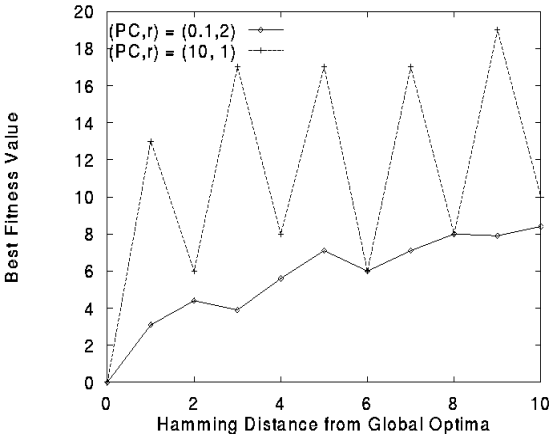



**FIGURE 2.** The Best Fitness Value vs. Hamming Distance from the Optimal Partition

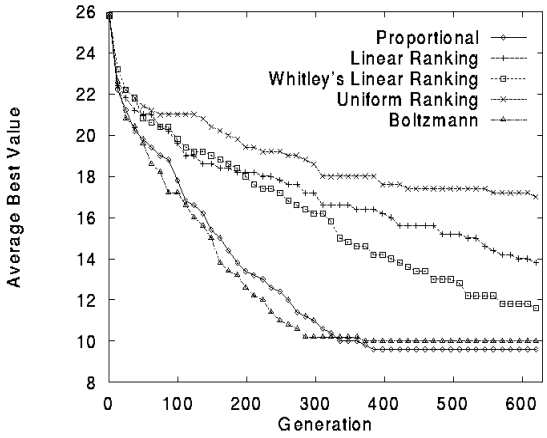



**FIGURE 3.** Sequential GA's with Different Selection Schemes

synchronous island GA's as discussed in section 3.3).

7. Same as model 2 except that the connection topology is dynamic, based on a positive-distance topology. An individual is passed to another node only if the Hamming distance between the best individuals of the two subpopulations is less than 24.
8. Same as model 2 except that the connection scheme is a negative-distance topology. An individual is passed to another node only if the Hamming distance between the best individuals of the two nodes is greater than 24.

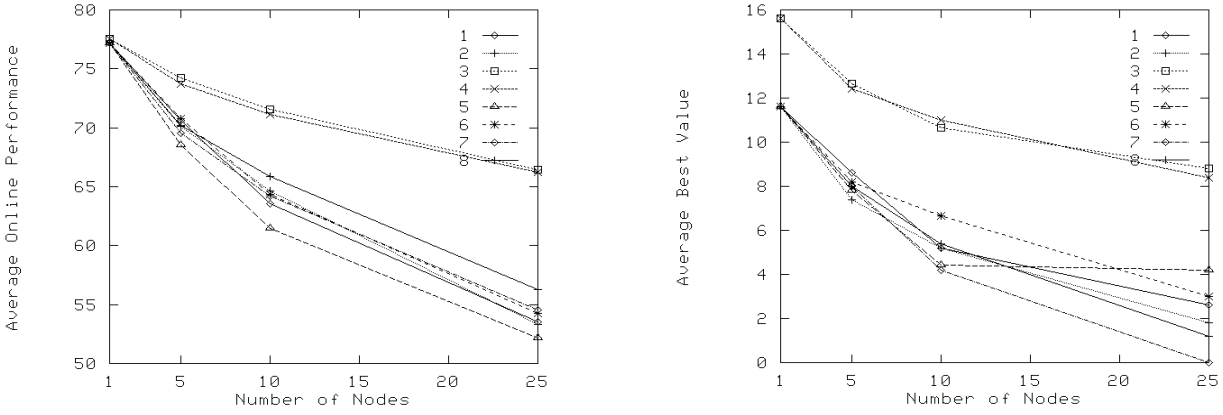
**TABLE 1.** One node experimental results

| Elitist Strategy | Ave. Online | Ave. Best | Best Value | Ave. Time (sec) |
|---|---|---|---|---|
| Yes | 77.19 | 11.6 | 8 | 5326.00 |
| No | 77.58 | 15.6 | 12 | 5314.80 |

One-node experiments are the same in different connection schemes. For them we only conducted the elitist model (1, 2, 5, 6, 7, and 8) and the non-elitist model (3 and 4). The results (Table 1) of other experiments are shown in Table 2. The interval of variation for f(x) is [0, 480].

From these results, it appears that the elitist strategy is important since none of the non-elitist experiments (model 3 and 4) ever found the global optima, even with parallelization. Comparing those models using the elitist strategy, the results indicate that the probability of finding a global optimum increases as the number of subpopulations increases. Remember that in our experiments the total number of evaluations is the same, therefore any search improvement is due to the indicated parameter settings.

In terms of network topology, the positive-distance topology model found a global optimum in each experiment with 25 subpopulations and in nearly half of the experiments with 10 subpopulations. Further note that in some 25-node ring topologies, different high fitness individuals are maintained in different nodes, indicating that with more nodes or evaluations, an optimum might be found. Figure 4(A) shows that the isolated island GA has the best online performance and that the two non-elitist models have the worst online performance. In Figure 4(B), one can see that all models' average best solutions increase as the number of subpopulations increase except for the




(A) (B)

**FIGURE 4.** (A) Average Online Performance and (B) Average Best Value

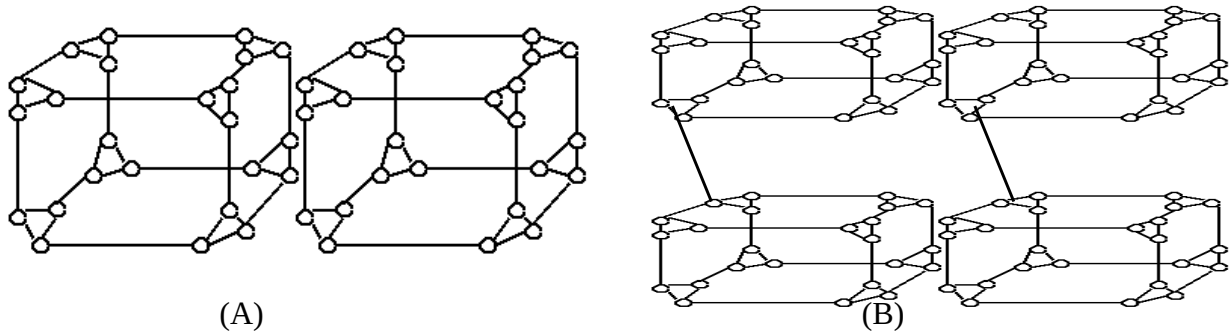



**FIGURE 1.** (A) A Two-Partition 3-CCC and (B) A Four-Partition 3-CCC with Two Additional Links.

has shown that a *PC* of 0.1 and an *r* of 2 allow a traditional GA to find the optimal partition (we conducted ten experiments with a traditional GA under these conditions and it found the optimum every time). We therefore needed to find a more difficult problem which a traditional GA would find difficult, so as to examine the characteristics of our cgGA's. We therefore set *PC* to 10 and *r* to 1. Fig. 2 gives a comparison of the objective functions under these two conditions. The deep valleys found in *(PC, r) = (10, 1)* makes the local optima more difficult to overcome than *(PC, r) = (0.1, 2).*

For some experiments we expanded the problem to four 3-CCCs (Fig 1B). The number of nodes here is 96, for a search space of $2^{96}$. Two "extra" links from node 1 to node 25 and node 49 to node 73 result in two optimal partitions, $0_1 \ldots 0_{48}\ 1_{49} \ldots 1_{96}$ and $1_1 \ldots 1_{48}\ 0_{49} \ldots 0_{96}$.

## 5 Experimental Results

### 5.1 The Two 3-CCCs Partition Problem

We examined 5 selection schemes discussed in section 2 of sequential GA's and 8 different models of cgGA's using the two 3-CCCs partition problem. Each model was run ten times using the same parameters, to allow for comparisons. The following conditions held for all models. The total population size was 5,000. We used ten random seeds to generate the ten initial populations. The crossover rate was 0.8 and the mutation rate was 0.005. The number of total evaluations was 5,000,000. The sequential GA experiment was done as a comparison point for the two general methods of dealing with premature convergence -- selection tuned for slow convergence, and parallelization.

For the sequential GA's with various selection schemes, the results shown in Fig 3 indicate that no sequential GA's found a global optimum.

For each of the 8 different cgGA models, 4 different subpopulation allocations were also evaluated. The subpopulation size is the total population size (5,000) divided by the number of nodes (1, 5, 10, or 25) used. The number of evaluations per node is the total evaluation number (5,000,000) divided by the number of nodes. The migration interval is 100 generations for each model except the isolation island GA's. At each migration point, the best individual is passed to its neighbor. The cgGA models use proportional selection with stochastic universal sampling [3] and two-point crossover. Each reported result is the average of ten runs. Each model's result table reports: the number of nodes/subpopulations in the experiment, the average on-line performance (an average of all function evaluations up to and including the current trial), the average best solution, the best solution (if a case finds a global optimum, the number of times it found the optimum out of the 10 runs is reported), the average time in seconds for the 5,000,000 evaluations, and the average speedup (the ratio of time between the experiment's runtime to 5,000,000 evaluations and the one-population run time).

1. This model is a ring topology using asynchronous communication. A node that achieves 100 generations triggers a migration event. In each migration, the node tries to receive individuals from its previous neighbor (if no individuals are passed from the previous neighbor, the node does not wait, but continues its GA search) and sends its best individual to its next neighbor. The initial subpopulation is randomly generated *independently* on each node. It uses the elitist strategy, which ensures that the best individual from each generation survives to the next.
2. Same as model 1 except in the way the initial subpopulations are generated. This model generates a super-uniform random population of 5,000. Then individuals are randomly selected to form equal-sized subpopulations. This initializes subpopulations to different parts of the search space.
3. Same as model 1, without elitism.
4. Same as model 2, without elitism.
5. Same as model 2 with *no migration* between nodes (the isolated island GA's as discussed in section 3.3).
6. Same as model 2, but using synchronous communication; that is, *all* nodes must wait to exchange solutions before the slowest node complete 100 generations (the

### 3.4.2 iiGA migration rules

An iiGA may have a number of different block sizes being used in its subpopulations. To allow interchange of individuals, we only allow a one-way exchange of information, where the direction is from a low resolution to a high resolution node. Solution exchange from one node type to another requires translation to the appropriate block size, which is done without loss of information from low to high resolution. One bit in an *n'*-long representation is translated into *r* bits with the same value in an *n*-long representation. Thus all nodes *inject* their best individual into a higher resolution node for "fine-grained" modification. This allows search to occur in multiple encodings, each focusing on different areas of the search space.

More formally, we note that node *x* with block size $p_1$x$q_1$ can pass individuals to node *y* with block size $p_2$x$q_2$ if and only if

$p_1 = j \times p_2$ , $q_1 = k \times q_2$

where j, k are integers, $j, k \geq 1$ .

This establishes a hierarchy of exchange, where node *x* (lower resolution) is the parent of node *y* (higher resolution) and node *y* is the child of node *x*. The direct child is the child with the largest block size; others are "heirs" of the lower-resolution "ancestors." Based on this general migration rule, we have designed the four following static topologies:

- **Simple iiGA's**

  Each node passes individuals to only one node, the node with the highest resolution (a block size of 1x1).
- **Complete iiGA's**

  Each node passes individuals to all of its heirs (of higher resolution) in the hierarchy.
- **Strict iiGA's**

  Each node passes individuals only to its direct children.
- **Loose iiGA's**

  Each node passes individuals to both its direct children and the node with the highest resolution (block size of 1x1).

### 3.4.3 iiGA Advantages

iiGA's have the following advantages over other PGA's.

1. Building blocks of lower resolution can be directly found by search at that resolution. After receiving lower resolution solutions from its parent node(s), a node of higher resolution can "fine-tune" these solutions.
2. The search space in nodes with lower resolution is proportionally smaller. This results in finding "fit" solutions more quickly, which are injected into higher resolution nodes for refinement.
3. Nodes connected in the hierarchy (nodes with a parent-child relationship) share portions of the same search space, since the search space of parent is contained in the search space of child. Fast search at low resolution by the parent can potentially help the child find fitter individuals.
4. iiGA's embody a divide-and-conquer and partitioning strategy which has been successfully applied to many problems. Homogeneous PGA's cannot guarantee such a division since crossover and mutation may produce individuals that belong to many subspaces --i.e., the divisions cannot be maintained. In iiGA's, the search space is fundamentally divided into hierarchical levels with well defined overlap (the search space of the parent is contained in the search space of the child). A node with block size $r = p'$ x $q'$ only searches for individuals separated by Hamming distance *r.*
5. In iiGA's, nodes with smaller block size can find the solutions with higher resolution. Although DPE [24] and ARGOT [22] also deal with the resolution problem using zoom or inverse zoom operators, they are different from iiGA's. First, they are working on the phenotype level and only for real-valued parameters. iiGA's divide the string into small blocks regardless of the meaning of each bit. Second, the sampling error can fool them into prematurely converging on sub-optimal regions. Unlike PDE and ARGOT, iiGA's search different resolution levels in parallel and reduce the risk of searching the wrong target interval.

## 4 Experimental Comparisons of Various Coarse-Grain PGA Architectures

We implemented a number of PGA's, using various characteristics discussed in Section 3, and compared them on a graph-partitioning problem. The graphs we used are two separate 3-CCCs (Cube-Connected Cycles) with a total of 48 nodes (Fig 1A). Node 1 to node 24 are in CCC partition 0 and node 25 to node 48 are in CCC partition 1. We encode a partition solution as a 48-bit string in which the value of the *n*th bit specifies the partition of the *n*th node. There are two global optima with value 0. The optimal partitions are $0_1 \ldots 0_{24}\ 1_{25} \ldots 1_{48}$ and $1_1 \ldots 1_{24}\ 0_{25} \ldots 0_{48}$. This is a function minimization problem with a search space of $2^{48}$ and minimum value of 0. The objective function is defined as

$$f(x) = cutsize + PC \times |1(x) - O(x)|^r$$

where x is the 48-bit string, cutsize is the number of edges with one endpoint in one partition and the other in another partition, 1(x) is the number of 1's in x, and *0*(x) is the number of 0's. The terms *PC* and *r* are penalty values in the objective function which play an important role.

The behavior of this objective function changes significantly under different penalty values[19, 20]. Ackley [1]

### 3.3.2 Connection Schemes

The connectivity of the processing nodes, in terms of the degree of connectivity and the topology of connection, affect the performance of a cgGA. More interestingly, another question is how such connections might change over time.

- **Static Connection Scheme**

The connections between nodes are established at the beginning of the run and are not modified, keeping the network topology static during execution. Topologies can be of various types, including: lines, rings, meshes, n-cubes, etc., but the topology determines which neighbors can exchange information, and that topology is static.

- **Dynamic Connection Scheme**

The topology of node connection is mutable during runtime. There are two basic reasons to allow modification of the topology during runtime. First, such modifications make distributed workstation parallelism practical in a real-world environment. If a node's GA process is stopped, then reconfiguration of the network occurs so as to continue processing. cgGA's can withstand a number of such events before losing evolutionary progress. Second, reconfiguration could occur based on changes that occur in the evolution of the subpopulations. One of the drawbacks of coarse-grain GA's is that the insertion of a new individual from another subpopulation may not be effective. The new individual may be grossly incompatible with that subpopulation, and therefore either be ignored or dominate the subpopulation. To avoid this, the subpopulations could start processing without any neighbors, and when a migration occurs, the node's neighbors could be determined by similarity (or dissimilarity) with other populations. For example, the best individuals of each population can be compared, and those with the closest Hamming distance could be established as neighbors. This establishes a *distance connection* topology that changes over time, and maintains interchange between only similar (or dissimilar) subpopulations.

### 3.3.3 Node Homogeneity

Node homogeneity is a measure of how similar the GA processes are on different processing nodes.

- **Homogeneous Island GA's**

The GA on each node uses the same parameters (population size, crossover rate, mutation rate, migration interval, etc.), genetic operators, objective functions, and encoding methods. Most research on PGA's has focused on homogeneous PGA's. One advantage of homogeneous island GA's is that they are relatively easily implemented.

- **Heterogeneous Island GA's**

Heterogeneous Island GA's allow the evolution of subpopulations via GA's with different parameters, genetic operators, objective functions, and encoding methods. In general, it is difficult to find the best initial parameter settings for a GA. This is especially true in the case of multiple objective optimization problems where various parameter settings can cause a GA to focus search in different portions of the search space. More importantly, many problems can be encoded in a GA using different methods, each with particular advantages and disadvantages. Such variations in parameter settings and encoding methods pose problems for interchange between populations, but these problems can be addressed.

## 3.4 Injection Island GA's (iiGA's)

We have begun work on a new PGA architecture called injection island GA's (iiGA's). iiGA's are a class of asynchronous, static- *or* dynamic-topology, heterogeneous GA's. We run them on a distributed network. The two most interesting aspects of an iiGA are its migration rules and the heterogenous nature of its nodes.

### 3.4.1 iiGA Heterogeneity

GA problems are typically encoded as an *n*-bit string which represents a complete solution to the problem. However, for many problems, the resolution of that bit string can be allowed to vary. That is, we can represent those *n* bits in *n'* bits, $n' < n$, by allowing one bit in the *n'*-long representation to represent *r* bits, $r>1$, of the *n*-long bit representation. In such a translation, all *r* bits take the same value as the one bit from the *n'*-long representation and vice-versa. Thus the *n'*-long representation is an abstraction of the *n*-long representation. More formally, let

$$n = p \times q$$

where *p* and *q* are integers, $p, q \geq 1$.

Once *p* and *q* are determined, we can re-encode a *block* of bits *p'* x *q'* as 1 bit if and only if

$$p = l \times p', \; q = m \times q'$$

where *l* and *m* are integers, $l, m \geq 1$.

Such an encoding has the following basic properties,

1. The smallest block size is 1*x*1. The search space is $2^n$.
2. The largest block size is *p*x*q*. The search space is $2^1=2$.
3. The search space with a block size *p'*x*q'* is $2^{p/p'} \text{x } 2^{q/q'}$.

An iiGA has multiple subpopulations that encode the same problem using different block sizes. Each generates its own "best" individual separately. (Note: This is not the same as the dynamic parameter encoding (DPE) feature of GAucsd 1.4 [23])

ing sections.

### 3.1 Micro-Grain GA's

Micro-grain GA's (mgGA's)[19] differ from other parallel approaches in that only a *single* population is maintained. The parallelism comes from the use of multiple processing nodes for evaluating individual fitness. At its best, a mgGA allocates a processor for each solution in the population, making the evaluation time for the entire population equivalent to the time required to evaluate the most costly individual. If fewer processors than solutions are available, then each node is responsible for processing a subset of the population, making the population's evaluation time equivalent to the evaluation time of the most costly subset. All other genetic operations (crossover, mutation, inversion etc.) are typically conducted sequentially by a single "master" node which controls the system. This scheme is especially useful when the evaluation function is computationally expensive relative to other GA operations. In [19], the authors show that mgGA's display "linear speedup" i.e., every doubling of processors(up to the population size) essentially halves processing time. Note however that mgGA's do *not* address the premature convergence problem. Their primary advantage is speed compared to sequential GA's.

### 3.2 Fine-Grain GA's

In fine-grain GA's (fgGA's), exactly one population solution is assigned to *each* processor [15, 17], which it processes in parallel with all others. Each solution is part of multiple subpopulations, with membership determined by the network connection topology of the processors. Thus the entire population can be viewed as numerous small overlapping subpopulations. In fgGA's, the topology of the network determines the degree of isolation, and therefore diversity, of the individuals in the population. The main problem to solve in a fgGA is the network topology. High connectivity between neighbors increases the spread of high-fitness individuals, making subpopulations susceptible to domination and perhaps premature convergence. Limiting the interactions will partially solve this problem, but essentially reduces the size of subpopulations, and can slow search dramatically.

### 3.3 Coarse-Grain GA's

Coarse-grain GA's (cgGA's) [18, 25, 26] are similar to fgGA's in that they have independent subpopulations that exchange solutions. However, in cgGA's each subpopulation contains a large number of individuals, not just a single individual. As a result, the frequency of migration between subpopulations is typically much smaller than found in fgGA's, and the effect of interaction between subpopulations is also smaller due to the large subpopulations. While not an exhaustive description, we here categorize cgGA's along three dimensions: migration method, connection scheme and processor node homogeneity.

#### 3.3.1 Migration Method

The migration method determines how often, and under what timing constraints, individuals are exchanged between populations.

- **Isolated Island GA's**

  There is no migration between subpopulations. This is the simplest model of cgGA's.

- **Synchronous Island GA's**

  The migration between two subpopulations is synchronized. By synchronizing migration, the populations evolve at roughly the same "rate" before exchanges occur. This rate measure (number of generations, convergence percentage etc.) determines the synchronicity of the subpopulations. Dedicated parallel hardware can directly support such synchronization, but synchronization in a distributed workstation environment can cause uneven work loads. Different machine speeds and different loads can cause some nodes to "wait", slowing the speed of evolution to that of the slowest node.

- **Asynchronous Island GA's**

  Asynchronous GA's allow migration based on a single event that does *not* relate to the state of evolution across all of the system's subpopulations. Such asynchronous behavior is the kind of migration typically found in nature, since different environments are responsible for differences in evolution speed. Our recent work has focused on asynchronous GA's, since we implement cgGA's on distributed workstations in a shared-use (campus computing) environment. In this setting, each GA process competes for computing resources with whatever users/processes are on the same machine. The different loads and diverse machine architectures cause different evolution speeds in each node, making asynchronous migration more appropriate. This is especially true when the number of processing nodes is in the hundreds, and the machine architecture types diverse, as occurs in MSU's cgGA runs. Such an approach, however, does pose some questions. When a relatively high-fitness individual from a fast-evolution node is inserted into a low-fitness population on a slow-evolution node, is that low-fitness subpopulation quickly dominated by the high-fitness individual? Does that insertion help the low-fitness subpopulation find the global optimum? We will attempt to answer some of these questions in the sections on experimental results.

tion, selection is likely to keep it there and prevent further adaptation within any practical timeframe. This is the premature convergence problem. Previous research has focused on two general approaches to avoid premature convergence. The first approach is to lower the convergence speed so the GA can do a more through search before converging, increasing the chance of finding the global optimum. These schemes affect the selection phase. The second approach focuses on keeping the diversity of the population high by modifying traditional replacement and mating operators. (Of course, a parallel GA may be seen as a particularly natural and efficient version of such an approach.)

### 2.2 Variations on Selection

The literature reports a number of approaches that modify traditional, proportional selection:

- **Linear ranking**

Here, selection probability is based on the solution's relative fitness ordering:

$$P_s(i) = \frac{1}{N}\left(\eta_{max} - (\eta_{max} - \eta_{min})\frac{i-1}{N-1}\right)$$

where $\eta_{max}$ and $\eta_{min}$ are the expected numbers of individuals with rank 1 and rank $N$ respectively. According to Baker [2], the relation $1 \le \eta_{max} \le 2$ and $\eta_{min} = 2.0 - \eta_{max}$ must be satisfied.

- **Whitley's linear ranking**

This selection probability [27] is identical to Baker's linear ranking, but it has the advantage of directly expressing the index calculation of the individual that should be selected:

$$P_i(\chi, a) = \frac{N}{2(a-1)}(a - \sqrt{a^2 - 4(a-1)\chi})$$

where $\chi$ is a random number uniformly distributed on the interval [0,1]. $a$ is similar to $\eta_{max}$ in linear ranking, but provides a wider variety of selection pressure (the expected number allocated to the top ranked individual).

- **Uniform ranking**

Uniform ranking [12] corresponds to Baker's ranking with $\eta_{max} = 1$ and is similar to $(\mu, \lambda)$-selection as used in Evolution Strategies. This is identical to a random walk.

- **Boltzmann tournament selection**

Boltzmann tournament selection is an attempt to transfer the acceptance criterion found in Simulated Annealing to GA selection. The transition method is described in [11].

### 2.3 Variations on Mating

De Jong introduced the concept of a crowding scheme [6]. In a crowding scheme, an offspring replaces an existing individual according to its similarity with other individuals in a randomly drawn subpopulation of size CF (crowding factor). The similarity measure is based on the Hamming distance between solutions in the subpopulation. Similar individuals compete in the same subpopulation, reducing the speed at which one individual can dominate the whole population.

Mauldin introduced a uniqueness operator to maintain high genotype diversity [16]. The insertion of an offspring into the population is possible only if the offspring is genotypically different from all individuals of the population (specified as a given Hamming distance).

Goldberg defined a sharing function [9] that permits the formation of stable subpopulations centered on different individuals, thereby permitting the parallel investigation of many peaks in the search space. This scheme determines the selection probability according to the average fitness of similar individuals in the population. The similarity criterion can be the distance between either encoded genotypes or decoded phenotypes.

Crow used a phenotypic comparison function to enforce restricted mating [5]. Only similar individuals are allowed to mate. This scheme avoids the production of low fitness offspring which could be produced from the mating of incompatible individuals.

### 2.4 Evaluation

Although both of the above schemes partly solve the convergence problem, many are not appropriate for some real-world problems. First, a similarity evaluation between all individuals is computationally expensive and therefore degrades performance. Furthermore, these evaluations are sequential in nature and hard to parallelize. Second, a phenotypic similarity evaluation is domain specific, requiring special encoding for each new problem. These encodings are necessarily domain-dependent and not easily applied to other problems.

## 3 Parallel Genetic Algorithms

An alternative approach to deal with premature convergence is parallelization of GA's (PGA's), which produces a more realistic model of nature than a single large population. PGA's both decrease processing time and better explore the search space. Unlike sequential GA's which pay a high computational cost for maintaining subpopulations based on similarity comparisons, PGA's (except micro-grain PGA's, as described below) maintain multiple, separate subpopulations which may be allowed to evolve independently. This allows each subpopulation to explore different parts of the search space, each maintaining its own high-fitness individuals and controlling how mixing occurs with other subpopulations, if at all. Some PGA approaches from the literature are described in the follow-

# Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach

Shyh-Chang Lin W.F. Punch III E.D. Goodman

Genetic Algorithm Research and Applications Group
Case Center and Department of Computer Science
Michigan State University

*Accepted for Publication, Parallel & Distributed Processing, Dallas TX, Oct 1994*

**Abstract**

*This paper describes a number of different coarse-grain GA's, including various migration strategies and connectivity schemes to address the premature convergence problem. These approaches are evaluated on a graph partitioning problem. Our experiments showed, first, that the sequential GA's used are not as effective as parallel GA's for this graph partition problem. Second, for coarse-grain GA's, the results indicate that using a large number of nodes and exchanging individuals asynchronously among them is very effective. Third, GA's that exchange solutions based on population similarity instead of a fixed connection topology get better results without any degradation in speed. Finally, we propose a new coarse-grained GA architecture, the Injection Island GA (iiGA). The preliminary results of iiGA's show them to be a promising new approach to coarse-grain GA's.*

## 1 Introduction

In 1975, Holland [13] described a methodology for studying natural adaptive systems and designing artificial adaptive systems. It is now frequently used as an optimization method, based on analogy to the process of evolution. References [13, 8] contain a theoretical analysis of a class of adaptive systems in which the space of structural modifications is represented by sequences (strings) of symbols chosen from some (usually binary) alphabet. The searching of this representation space is performed using so-called "Genetic Algorithms" (GA's). The genetic algorithm is now widely recognized as an effective search paradigm in artificial intelligence, image processing, VLSI circuit layout, optimization of bridge structures, solving of non-linear equations, correlation of test data with functional groupings, and many other areas.

In this paper, we detail a particular kind of parallel GA, called coarse-grain GA's, which aid in global search and retard premature convergence. Coarse-grain GA's maintain multiple, independent populations with occasional interchange of solutions between these populations. We will study coarse-grain GA's using a graph partitioning problem[4, 14]. Our experimental results present some coarse-grain GA schemes which can solve graph partition problems which are difficult for sequential GA's.

## 2 Variations on Traditional GA's

The traditional GA proposed by Holland [13] used proportional selection, one-point crossover, and standard mutation. In proportional selection the probability $P_s$ that individual $i$ with fitness $f_i$ is selected to become a parent in the next generation is based on the fitness proportion. That is,

$$P_s(i) = f_i / \left( \sum_{j=1}^{N} f_j \right)$$

One-point crossover exchanges substrings between individuals starting at a position chosen randomly on the strings. The crossover rate is the proportion of the population that will undergo crossover each generation. Mutation operators change a string by randomly changing one bit. The probability of each bit undergoing mutation is the bitwise mutation rate. One common problem in this traditional GA approach is premature convergence --i.e., finding a local instead of global optimum. Many variations on traditional GA's have been devised to address this problem. In this section, we discuss some of them.

### 2.1 The Premature Convergence Problem

To understand premature convergence, we first examine the concept of *genetic drift*. Genetic drift is the modification of a population resulting from random events. Under no selection pressure (a random walk), a population will be dominated by genetic drift and converge [10]. So whether selection or drift dominates, convergence is inherent in traditional GA's and is the reason that traditional GA's can't maintain different high fitness individuals in one population. Once a suboptimal individual dominates the popula-