Profiling Communication in Distributed Genetic Algorithms

Jonathan Maresky, Yuval Davidor;[*] Daniel Gitler, Gad Aharoni and **Amnon Barak**
Institute of Computer Science,
The Hebrew University of Jerusalem,
Jerusalem 91904, Israel
yuvaKDschema.co.il

Abstract

To what extent is distribution beneficial to the search quality and computational resources used by a genetic algorithm execution? Most distributed genetic algorithms rely on communicating genetic information, in the form of individual solutions, between concurrently evolving populations.

Another way of effectively using the additional information generated by the parallel executions is the profiling approach to communication, where populations decide whether their own performance is satisfactory, relative to the global average improvement curve. Thus, communication between populations takes the form of improvement histories. This is shown to improve on the traditional communication approach, in terms of both solution quality and execution performance.

1 Introduction

Nature is essentially distributed. Distributed processing permits both nature and genetic algorithms to benefit from the parallel exploration of different solutions. This concept of separate populations developing in parallel and sharing information motivates distributed genetic algorithms. This approach is best implemented on a distributed system of computers, which provides parallel processing and data sharing. These two elements comprise the essential components of a distributed genetic algorithm (DGA): the core algorithm to be distributed, and the approach to sharing information between the processes.

This paper investigates the effect of distribution on a genetic algorithm, and, more specifically, studies the contribution of communication between concurrently executing processes to the distributed GA's performance. Most approaches to communication between concurrently evolving populations involves the migration of individual solutions from one process to another. The benefit of these approaches is highly dependent on the characteristics of the problem and its solution space. A new communication paradigm for DGAs is described: no genetic information is migrated between populations; instead, information about the improvement rate of neighbouring populations is used to measure a population's viability. This new approach introduces new parameters, which are described and analysed for robustness. The approach is tested on two different problems, and compared against a similar DGA without communication, and the basic sequential GA.

This paper is structured as follows: Section 2 discusses the construction of the DGA, and describes the sequential GA and the non-communicating DGA which are used for performance and behaviour comparisons with the new DGA. This section includes a short analysis of traditional DGA communication, which motivates the new communication approach. Section 3 introduces the profiling model of communication and describes the new parameters. Section 4 describes the problems used, the platforms tested on, and presents and analyses experimental results. Our conclusions are discussed in 5.

[*] Schema - Evolutionary Algorithms Ltd., 61a Hadar St., Herzlia

2 The Construction of a Distributed GA

To what extent is distribution beneficial to the search quality and computational resources of a GA execution? How well does the genetic algorithm paradigm lend itself to distribution? To what extent does the specific problem influence the effectiveness of the distribution?

The obvious questions of parallel search [Muhlenbein, 1990] are:

- Are N parallel searches of time t as efficient as a single search of time $N \times t$?
- Are N linked searches more efficient than N independent searches?
- How should the linkage be done?

Additionally, do the benefits of linkage justify its cost?

This section describes the construction of a distributed genetic algorithm (DGA) which provides the basis for answering these questions.

The well-known GA paradigm is described in Figure 1.

```
Produce an initial population
 of many individual solutions,
Evaluate all individuals,
While termination condition is false:
{
    Select fitter individuals
     for reproduction,
    Produce new individuals,
    Insert some of these individuals into
     the population, at the expense
     of other individuals,
    Re-evaluate all individuals,
}
Report on results.
```

Figure 1: *Basic description of the genetic algorithm paradigm*

2.1 The Restart Operator

Any analysis of a GA's performance depends on both the quality of solution and execution resources used, and may be implemented by holding constant one of these variables and observing the other. The inclusion of the *restart* operator [Maresky, 1994] is beneficial to this analysis. In order to continue a GA's search when the population has converged, the restart operator introduces new genetic material (typically, by completely reinitializing the population) and thus moves the GA into another solution region. This involves the obvious tradeoff between the probability of discovering sufficiently better individuals within the current area of the search space, and the use of extra resources in creating a new population and improving it to a similar level as the previous converged population. It has been shown in [Takana, *et al*, 1994] that a GA with optimal population size and the appropriate number of restarts, exhibits better performance than the same GA functioning without the restarts, and an appropriately larger population size.

The operator operates *outside* the main loop of the GA (see Figure 2, which extends the inner loop of Figure 1) and is reliant on some definition of convergence.

The resultant GA is used as the sequential basis for a distributed genetic algorithm.

2.2 A Distributed Genetic Algorithm Without Communication

There are many different approaches to parallelizing a genetic algorithm [Fogarty and Huang, 1990]. Most algorithms differ on the topology of the distribution and the separate processes themselves: how they are controlled, how they communicate and what their tasks are. In particular, the island approach [Tanese, 1990] is least reliant on processor synchrony and coordination. Communication between processes is minimized by the controlling algorithm assigning separate populations to the processes, which develop in parallel and may exchange information. This exchange generally takes the form of copying individuals between the populations.

```
While termination condition false:
{
    Produce population,
     and evaluate individuals,
    While population has not converged,
     and termination condition is false:
    {
        Do an inner reproduction loop
         of Figure 1,
    }
    Update global result, if necessary,
}
Report on results.
```

Figure 2: *Outer loop structure of the genetic algorithm with restart operator*

This allows the construction of a DGA without communication. Each process is assigned a population, of the same size as that of the sequential GA. Communication only occurs after the termination criterion has been met, in order to summarize the global results of the execution. This model forms a basis for the determination of the specific effects of adding different forms of communication. The resulting algorithm is shown in Figure 3.

```
Parent process spawns other processes,
For each process i:
{
    While termination condition false,
     OR a signal is not received:
    {
        Execute sequential GA of Fig. 2
        If termination condition true,
         signal other processes to quit,
    }
    Return final results to parent,
    Quit.
}
Parent process collects, summarizes results.
```

Figure 3: *Distributed GA (good-enough approach) with simple deme communication*

Here, each processor's results are independent of its neighbours. As the probability distribution of solution quality and function executions used during an execution is not uniform, the *speedup* (improvement due to distribution) is expected to be sub-linear in the number of processes used.

2.3 Traditional Communication in Distributed GAs

Any improvement in the performance of the algorithm as a result of communication is beneficial, unless the cost of communication outweighs the benefits. Previous island DGAs communicate genetic information between the populations, mostly in the form of individual solutions, copied *(migrated)* either at intervals or continuously [Pettey, *et al*, 1987; Starkweather, *et al*, 1990; Tanese, 1990]. This approach is beneficial for higly separable problems, where the good solutions from different areas of the solution space may be successfully combined. However, the algorithm's performance is highly dependent on its parameters (migrant selection policy, rate of migration, migrant insertion policy) and is less beneficial for less separable problems.

Additionally, this form of communication suffers from *conqu est* [Maresky, 1994]. Populat ions evolving asynchronously in parallel are often at different levels of evolution. The arrival of highly evolved migrants from a strong population will result in their higher rate of selection than local, Jess evolved individuals. Thus, the sending population's solution is often imposed on that of the receiver. Conversely, migrants arriving from a less evolved population are not selected for reproduction and are wasted.

3 The Profiling Model

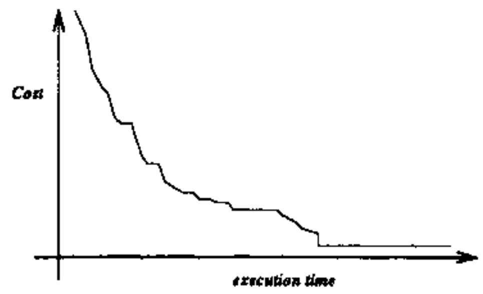
Another way to take advantage of the parallel processing and resultant increase of available information is through the dynamic cross-checking of a process's progress. This approach does not communicate genetic information, rather information about a process's progress. As such, the profiling approach departs from the popular analogy with biological systems.

It has been shown that under certain circumstances, the quality of a GA execution is correlated to the steepness, or speed, of its improvement [Davidor and Ben-Kiki, 1992; Manderick, *et al*, 1991]. That is, the more rapid the improvement, the more chance of the GA execution reaching a good result, and vice versa. The GA and, more specifically, the process, has no a-priori knowledge of the solution space of the problem, or what constitutes "good" progress. A process receiving information regarding the progress of other processes checks its own relative progress. If this is inferior, the process restarts in similar fashion to the normal restart operator.

A restart in this manner does not mean that a better solution would not have been found in the old search space. Rather, as in the case of the restart operator (see Section 2.1), it is less likely that the old search space would yield a better solution. The use of resources in restarting the population and bringing it to a comparable level, is *probably* more beneficial in producing better solutions.

This approach implies regular communication, unlike the traditional communication model where genetic information is migrated at any time. This communication is of *profiles:* strings of *costs* (unnormallized fitness values) forming the history of a process's progress (see Figure 4). In this example, the first cost value is stored after 200 evaluations; additional values are stored at intervals of 200 evaluations.

Profiles are sent to other random processes at regular intervals; at the same intervals, profiles from other processes are received and the local population's progress is checked against the global average. The curve formed by the average process profiles exhibits properties of "optimization graphs" and "learning curves", used in other frameworks.



time	200	400	600	–	2400	2600	...	5600	...	10400	10600
cost	85.62	54.11	28.91	...	4.67	4.11	...	0.21	...	0.032	0.032

Figure 4: *A profile of the execution improvement against time*

Initially, an *off-line* model was constructed: each process's progress is checked against a static, predetermined, global progress profile, without any communication or dynamic recalculations of the curve. This off-line curve is obviously different from that of the *on-line* model, where a process's execution may be terminated because of sub-average performance, changing the curve considerably.

It was decided to have a process's profile values carry the average improvement since the beginning of the execution, instead of since the last restart. In other words, a process's profile is not reinitialized on restart; this allows for more information to be utilized.

```
While termination condition is false:
{
  Every interval evaluations:
  {
    Get, store profiles from other processes
    Update, send my profile to a process
    Check my progress against the global
     average:  if unsatisfactory, restart
  }
  Do an inner reproduction loop
   of Figure 1,
```

Figure 5: *DGA with profile communication*

The initial approach observed the local process's best-of-generation cost against the "sleeve" defined by the offline cost curve and its flanking standard deviation curves. However, this means each process communicating the standard deviations at each point as well as the profiles. It was thus decided to restart the process if its cost at any point was more than some constant percentage worse than that of the offline curve; the resulting algorithm is shown in Figure 5, and replaces the inner loop of Figure 3.

This requires four new parameters:

- *When to begin* profiling: the first few evaluations may not provide a sufficiently good indication of the future potential of the execution.
- *How often* to profile: while higher granularity of profiling provides better information, this involves a tradeoff with lower communication rates. This parameter also determines the amount of profiles to store.
- *How often* can the process stray outside the sleeve: once may be too easily misled by the stochastic behaviour of the execution; more may limit the effectiveness of the approach.
- *How wide* is the sleeve determined by the cost curve and the upper bound percentage: a small percentage (and narrow sleeve) disadvantages some functions; similarly high percentages and wider sleeves.

4 Results

4.1 The MOSIX-486 Platform

While most experimental executions were run on Sun SPARCstations, all timing results were obtained on a 4-node Intel 486-PC configuration running under the MOSIX operating system [Barak, *et al,* 1993]. MOSIX is a UNIX based multicomputer operating system that incorporates the CPU resources of a network of workstations by supporting dynamic process migration and load-balancing, which are transparent to the application level, for efficient utilization of the network-wide resources and balanced workload distribution.

4.2 Problems

Two problems are selected for testing. One, *bal-lj,* is a quadratic assignment problem, and *J Ml* is a dynamic control problem.

The highly inseparable quadratic assignment problem [Davidor, 1993] involves 14 dimensional quadratic minimization, which has many applications in dynamics (though the dimensionality may vary).

At low dimensions, the problem may be solved using the Simplex method which guarantees optimality. However, numerical problems are encountered at high dimensions which cause convergence to a local optimum and a GA approach becomes attractive.

Each individual is coded as containing a chromosome of 14 real numbers, a total mass, moment, cost and a fitness value. The solution space size is of the order of 10^{287}.

The dynamic control problem is from [Janikow and Michalewicz, 1991]:

$$\min\left(x_N^2 + \sum_{k=0}^{N-1}(x_k^2 + u_k^2)\right)$$

where

$$x_{k+1} = x_k + u_k, k = 0, 1, \ldots, N-1.$$

where *xo* is the initial state, $x_k \in \mathcal{R}$ is a state, and *u* is the solution vector.

We call the following problem *JMV.* a fixed domain of (-200,200) is assumed for each u_i, with x_0 = 100 and *N* = 45. In similar fashion to the *bal~14* problem, each individual is represented by a 45-ary vector of real numbers, a cost and a fitness value. This problem is highly separable and responds very well to traditional methods of distributed communication.

4.3 Results and Discussion

In order to compare the performance of the DGA to both the sequential GA and the DGA without communication described in 2.2, the performance of the algorithm is measured in reaching some acceptably good solution. The approximate comparison is on the basis of measuring function evaluations, while the actual comparison refers to the actual time taken for execution on the MOSIX-486 described in 4.1. All results are taken over 250 executions.

It is clear that the more difficult the problem or the goal cost of the good-enough approach, the longer the execution and the more advantage to the DGA communicating profiles. A short execution does not enable the processes to build up sufficient knowledge about the average cost improvement curve. A short execution also implies fewer process restarts as a result of this greater knowledge due to the communication of profiles.

Additionally, the execution until the first restart is hardly affected by the communication of profiles, for two reasons. Firstly, the information (or knowledge) gained is not statistically useful as not enough data has been collected: each process can have at most n cost values for each profile position, for *n* processes. Secondly, the faster processors do not receive any useful information at all until after they restart for the first time.

Problem	(1)	(2)	(3)	(4)	(5)
bal-14	0.03	50	9504(6900)	16%	3.65(3.41)
bal-14	0.01	50	29439(26555)	34%	4.82(4.49)
JM	30000	70	15583(7742)	-9%	1.52(1.41)

1. **Acceptably good solution cost**
2. **Optimal communication interval - no. of evaluations**
3. **DGA with profiling: # evaluations (std. deviation)**
4. **Approximate % improvement over the non-communicating DGA**
5. **Approximate speedup over the sequential GA (actual speedup)**

Figure 6: *Profiling results for 4 processes (250 executions)*

Some results are shown in Figure 6. For each problem, the performance of the profiling DGA is measured against the core sequential GA, and against the DGA without communication. The comparison is on the basis of number of function evaluations needed to reach a specific solution quality (specified in the second column). The fourth column shows the contribution of the new communication to the core DGA (without communication), and the last column presents approximate (in terms of function evaluations) and actual (in terms of time elapsed) speedup results over the core sequential GA. Note the close correlation between the function evaluation speedup and the actual time speedup.

The quadratic assignment problems are most strongly responsive to the new approach (see Figure 6). While reasonably good results are obtained for a cost value of 0.03, a closer observation reveals that the problem is not difficult enough to provide advantage to the communication: on average, less than two restarts are needed to find a good solution. Setting a harder cost, namely 0.01, results in significant speedups of over 4.4 at optimal parameter settings.

The new parameters introduced are particularly robust. A brief analysis of the effects of changes in the parameters on the results of the DGA reveals that an early start to profiling does not overly hinder the process, while starting late (for example, after 2,000 evaluations) is less effective. For *bal-14,* acceptable results are obtained with a profiling frequency of up to 200 evaluations - equivalent to a communication probability of 0.005. A lower rate of communication does not provide sufficient information, while much higher rates are too expensive in terms of communication overhead. The parameter of two or three consecutive intermediate results outside of the sleeve is optimal here - one "hit" is too random, and four or more hits implies too slow a response. In terms of the width of the sleeve, a value of 40% is most effective - wider sleeves result in fewer restarts.

These parameter settings are not universal. Different settings produced optimal results for the other problem, J*MI.* This showed very little improvement over the non-corrimunicating DGA. The problem is monotonic and does not readily benefit from having executions prematurely terminated. It conforms less to the discussed relation between speed of improvement and quality of final result. Additionally, one of the main reasons behind the profiling DGA's effectiveness (for the *hal-14* problem) is due to the shape of the probability distribution defined by the DGA without communication. Here, a strong, right-sloping tail is evidence of the noncommunicating DGA's tendency towards long executions which search unprofitable regios before eventually terminating. The probability distribution of the noncommunicating DGA for the *JM1* problem has no such tail, that is, almost all executions terminate within some bound. Numerous attempts at parameter setting were tried, without success.

In response to the questions of Section 2, the results from the DGA communicating profiles (in Figure 7) for the *bal-14* problem, are similar to those results already presented. The sequential GA is executed, using 400, 000 evaluations; the DGA with profiling is executed for 4 processes, each using 100,000 evaluations. Thus, after using the same total number of function evaluations, there is an improvement (in the best value found) over the sequential GA. This is analogous to the achieving of a superlinear speedup in the previous analyses. The improvement is slightly less than expected - this can be explained by the design of the profiling approach to communication: the additional information about the average improvement curve is present fairly early in the execution. The execution continues for 100, 000 evaluations, which lessens the advantage of distribution and imposes the partial information (gathered early in the execution) on the later stages of the execution. This also results in very varied expected costs.

Problem	(1)	(2)	(3)
hal-14	0.00299 (0.00326)	**0.00238 (0.00326)**	Communication every 100 evals, starting after 100 evals

(1) Sequential GA mean result (std. deviation) over 400,000 evaluations

(2) DGA mean result (std. deviation) over 100,000 evaluations x 4 processes

(3) Communication parameters

Figure 7: *Evaluational comparison of the SGA and the DGA communicating profiles (250 executions)*

Figure 8 shows the effect of varying the number of processes on the performance of the DGA communicating profiles, for the *bal-14* problem, for different good-enough costs. The results are approximate, and do not take physical timing into consideration.

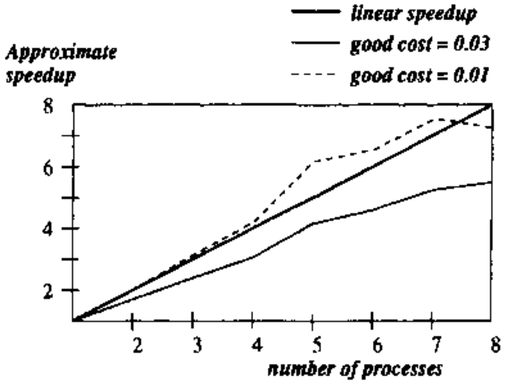


Figure 8: *Scaling effects for the DGA communicating profiles (250 executions)*

The influence of a harder good-enough cost is clearly seen: the easier problem does not benefit greatly from the profiling. A DGA communicating profiles on this problem returns different results depending on the good-enough cost. Note also the consistent superlinear speedup on the harder problem, which improves until a specific level, where increasing the number of processes does not improve the quality of global convergence information.

An attempt was made to combine the profiling communication with the traditional communication discussed in Section 2.3. The results, for the *bal-14* problem, improved on those of the DGA with traditional communication, but were worse than those presented above. This suggests that the DGA communicating genetic information benefits from restarting populations which are not providing fast improvement, while the communication of emissaries is detrimental to the DGA communicating profiles. No fine-tuning of profile or emissary communication was attempted, because of the relatively poor results.Note that this hybrid approach would imply greater communication overhead than the previous approaches.

The profiling approach is novel and inspires interesting modifications and extensions to distributed GAs. For example, profiling plays a similar role to restart: both operators renew the GA search in a new area of the problem space. In both operators, the probabilistic decision is based on perceived less-than-optimal performance and is easily fooled by erratic or volatile improvement curves. It is necessary to investigate the distributed profiling approach as an alternative to the restart function, using the average profile generated by parallel executions to determine the worth of continuing an execution instead of restarting. It is also necessary to reduce the dependence on the additional parameters introduced.

5 Conclusions

The profiling approach to communication between the concurrently evolving populations of a DGA is an alternative attempt to utilize the additional information generated by these parallel processes. This approach is motivated by the relationship between the speed of execution improvement and the execution's resulting solution quality. New parameters are introduced, many of which are robust, in the sense that varying their values does not affect the DGA's performance.

The populations communicate their improvement history, and a process decides on its own viability according to the global average improvement curve. This learning curve is also useful in determining the optimal number of processes to use, and in setting parameters.

Problems for which a DGA with traditional communication is less effective (and consistently returns sublinear speedups) are shown to produce superlinear results using this approach. This benefit is highly dependent on the correlation between the speed of improvement and the quality of an execution. This dependence, which is influenced by the problem, the GA and its parameter settings, can be measured and may constitute an important factor in determining *a priori* the success of the profiling effect on a specific problem.

The profiling approach also provides a positive answer to the question of parallel versus sequential search quality, given a fixed execution time.

References

[Barak, *et al,* 1993] A. Barak, S. Guday and R. G. Wheeler. The MOSIX Distributed Operating System. ***Lecture Notes in Computer Science*** **No. 672,** Springer Verlag, 1993.

[Davidor and Ben-Kiki, 1992] Y. Davidor and 0. Ben-Kiki. The Interplay Among the Genetic Algorithm Operators: Information Theory Tools Used in a **Holistic Way. In *Parallel Problem Solving from Nature*** *2,* 1992.

[Davidor, 1993] Y. Davidor. An ECOlogical Model for Evolutionary Computing. In *The Japanese Journal* ***for Systems, Control and Information,*** **Vol. 37, No.** 8, 1993.

[Fogarty and Huang, 1990] T. C. Fogarty and R. Huang. Implementing the genetic algorithm on transputer based parallel processing systems. In *Parallel Problem* ***Solving from Nature,*** **1990.**

[Gordon and Whitley, 1993] V. S. Gordon and D. Whftley Serial and parallel genetic algorithms as function **optimizers. In *Proc. of the 5th Intl. Conf. on Genetic Algorithms,* 1989.**

[Janikow and Michalewicz, 1991] C.Z. Janikow and Z. Michalewicz. An Experimental Comparison of Binary and Floating Point Representation in Genetic **Algorithms. In *Proc. 4th Int. Conf. on Genetic Algorithms,*** 1991.

[Maresky, 1994] J. Maresky. On Efficient Communication in Distributed Genetic Algorithms. M.S. dissertation, Institute of Computer Science, The Hebrew University of Jerusalem, 1994.

[Manderick, *et al,* 1991] B. Manderick, M. de Weger and P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In *Proc. 4th Int. Conf.* ***on Genetic Algorithms,*** **1991.**

[Muhlenbein, 1990] H. Muhlenbein. Evolution in Time and Space - the Parallel Genetic Algorithm. In *Foundations* ***of Genetic Algorithms*** **(Edited by Gregory** J.E. Rawlins), 1991.

[Takana, *et al,* 1994] R. Takana, Y. Davidor and T. Yamada. Optimal population size under constant computation **cost. In *Parallel Problem Solving from Nature*** *3,* Springer-Verlag, 1994.

[Pettey, *et al,* 1987] C. B. Pettey, M. R. Leuze and J. J. Grefenstette. A Parallel Genetic Algorithm. In *Proc.* ***2nd Intl. Conf. on Genetic Algorithms,*** **1987.**

[Starkweather, *et al,* 1990] T. Starkweather, D. Whitley and K. Mathias. Optimization using Distributed Genetic **Algorithms. In *Parallel Problem Solving from Nature,* 1990.**

[Tanese, 1989] R. Tanese. Distributed genetic algorithms. **In *Proc. 3rd Intl. Conf. on Genetic Algorithms,*** 1989.