Enhancing the Efficiency of the ECGA

Thyago S.P.C. Duque, David E. Goldberg, and Kumara Sastry

Illinois Genetic Algorithms Laboratory,
University of Illinois at Urbana Champaign,
104 S. Mathews Ave, 117 Transportation Bldg,
Urbana, IL, USA
{thyago,deg}@illigal.ge.uiuc.edu,
kumara@kumarasastry.com
http://www.illigal.uiuc.edu/web/

Abstract. In this paper we show preliminary results of two efficiency enhancements proposed for Extended Compact Genetic Algorithm. First, a model building enhancement was used to reduce the complexity of the process from O(n^3) to O(n^2), speeding up the algorithm by 1000 times on a 4096 bits problem. Then, a local-search hybridization was used to reduce the population size by at least 32 times, reducing the memory and running time required by the algorithm. These results are the first steps toward a competent and efficient Genetic Algorithm.

Keywords: Estimation of Distribution Algorithms, ECGA, Model Building, Efficiency Enhancement.

1 Introduction

Evolutionary Algorithms (EA) [1] [2] have been successfully used in several different applications involving search, optimization and machine learning problems. Goldberg [3] presents a design-decomposition methodology for successfully designing scalable Genetic Algorithms (GAs). A GA that can solve hard problems accurately, efficiently and reliably is called a competent GA. These GAs can solve problems that are intractable for other algorithms in a tractable polynomial time.

However, to solve large scale problems it is oftentimes necessary to enhance the efficiency [4] of the algorithm. Some common approaches include parallelization [5], hybridization [6] [7] [8], time continuation [9], and evaluation relaxation [10].

The recent success of the Compact Genetic Algorithm (cGA) [11] on solving a billion bit noisy optimization problem [12] [13] has proven that GAs, if properly designed and optimized, can solve difficult large problems.

The objective of this work is to reproduce this success, creating an efficient and competent EA, capable of solving large scale difficult problems. Particularly, we are interested in creating an algorithm that can deal efficiently with problem sub-structures, solving a broader class of problems than the class the cGA can handle. This paper presents a proof of principle on the importance of efficiency enhancements for GAs. We present the first steps toward such efficient and competent EA and discuss future directions for the next steps.

2 The Extended Compact Genetic Algorithm

The Extended Compact Genetic Algorithm (ECGA) [14] is an evolutionary algorithm in the class of Estimation of Distribution Algorithms (EDAs) [15]. These algorithms substitute the selection-crossover-mutation process, common to GAs, by a selection-model-building-sampling process. Some examples of EDAs include the cGA [11], that uses probability vector as a model, and the Bayesian Optimization Algorithm (BOA) [16] that uses a Bayesian Network as a model, being able to represent which variables are linked together.

The ECGA uses the Marginal Product Model (MPM), which can be divided into two components, (I) a partition over the variables, defining which variables are independent and which variables are linked, and (II) a probability distribution over each partition.

The ECGA is based on the principle that the estimation of a good model for the population is equivalent to the linkage learning [2] [14] process. It searches the space of possible partitions to find an appropriate one and tunes the probability distribution to match the data. The ECGA uses the minimum description length (MDL) principle as learning bias, which means that the cost of representing the whole population under the compression induced by the model (Compressed Population Complexity - CPC), together with the cost of representing the model itself (Model Complexity - MC) should be minimal.

The ECGA chooses a partition that appropriately models the sub-problem structure by greedily optimizing the Combined Complexity Criterion (CCC) using Algorithm 1, which assumes that each of the variables are independent and them evaluates all possible pair wise merges and pick the best until no merging can improve the CCC. After the partition is determined, the probability distribution is estimated simply by counting the frequencies in the population. For detailed information about the ECGA and the CCC, please refer to [14].

The ECGA main loop can be summarized by the following steps. First, it generates a random population and then repeats the process of evaluation, selection, model-building, and sampling until any convergence criteria is satisfied.

Algorithm 1. Greedy search for an appropriated model in the ECGA

```
1. Start assigning each variable to an independent partition
2. Repeat:
3.   For each pair of partitions:
4.     Merge that pair
5.     Evaluate the CCC of the model
6.     Undo the merging
7.   Merge the pair that induced the smaller CCC if any
8.   End the search if no improvement is possible
```

3 Efficiency Enhancements for the ECGA

The methodology proposed by Goldberg [3] allows us to design competent GAs, which can solve hard problems in polynomial time. The same methodology can be used to design efficiency enhancements for GAs [4], which can be divided in four main categories: Parallelization, hybridization, time continuation and evaluation relaxation.

In a very superficial view, parallelization deals with the division of the GA in several processors. Hybridization deals with the integration of GAs with other search procedures. Time continuation deals with the tradeoff among using a larger population for short time or smaller population for long time and evaluation relaxation deals with the tradeoff among having a noisy and cheap evaluation against an accurate and expensive one.

This paper proposes two extensions to improve the performance of the ECGA. One of them follows the cited methodology; it is the hybridization of the ECGA with a local search procedure that will work as a preprocessing mechanism. The other extension addresses the model building process, which according to running time profiling information consumes more than 90% of the computational time. The profiling information was generated using the GNU gprof utility for UNIX systems.

All results presented in this section use the mk-trap problem [9] with trap size (k) of 4 as benchmark problem. The mk-trap is an additively separable problem with m sub-problems, each of them being a trap function of k bits. The tournament size was fixed to 16 for all experiments. The population size and number of sub-problems (m) varies on the experiments.

3.1 Improving the Model Building Process

The ECGA is a competent genetic algorithm, but it is computationally expensive. The first step to improve its performance was to profile the code and to determine which are the most time consuming steps of the algorithm. The result of such profiling showed that the model building process took more than 90% of the computational time. This concerning aspect lead us into a search for model building alternatives to reduce this time.

The first possible approach to that problem would be the use of a cache structure, already used on [17], which sacrifices memory to get runtime improvements. However, this improvement is not sufficient for large problems and memory may also become a limiting resource.

Reviewing Algorithm 1, we point that line 2 introduces a loop that depends on the size of the chromosome (l). Line 3 introduces iteration over pairs of variables in the string, which can be re-written as: "For each variable; For each other variable", introducing a order two iteration over l. The overall complexity is O(l^3).

One possible way to reduce the runtime would be to finish the loop in line 3 before it evaluate all pairs. Instead of searching for the best merging among all pairs, it might be enough to accept any merge that improves the CCC and stop the loop. This approach works for tight coded problems [2], but fails for loose coded ones when the population size is close to the minimum necessary for the ECGA, since spurious relations among actually independent variables are expected show up by just chance. Using a larger population is a way to bypass this problem but this price overcomes the benefits.

A successful approach would have to reduce the overall complexity of the algorithm. This is achieved using the Algorithm 2. Fundamentally, instead of evaluating the CCC over all pairs of partitions, an O(l^2) step, Algorithm 2 selects one partition (a) and evaluates the CCC only over the pairs that include this partition, reducing the complexity of this step to O(l).

Algorithm 2. Improved greedy model building for the ECGA

```
1. Start assigning each variable to an independent partition
2. Repeat:
3.   Choose a partition (a)
4.     For each other partition (b):
5.       Merge a and b
6.       Evaluate the CCC of the model
7.       Undo the merging
8.    Merge the pair that induced the smaller CCC if any
9.    With some small probability pb, break any partition
10.   Cool down the breaking probability
11.   End the search if no improvement is possible
```

As mentioned, spurious relations are expected to show up just by chance. To overcome that problem, we used a random breaking mechanism that chooses a random partition and divides it. However, the signal of spurious relations decreases as the real relations are discovered. This fact motivated the adoption of a cooling down mechanism to decrease the perturbation rate over time. This local search/random perturbation mechanism was successfully applied on several problems and is the basis of algorithms like Simulated Annealing.

Two items need further explanation. The selection of the partition (a) to be merged (Line 3) and the setting of the breaking probability pb. The best results were achieved using a round-robin policy for choosing a and a initial breaking probability $pb = 0.01$, although for noisy problems a greater pb values would produce more accurate results. To cool down, we assign $pb = 0.9pb$.

It is important to notice that decreasing the model building accuracy may have undesired side effects on the ECGA. On the performed tests, the model building accuracy for Algorithm 1 and for Algorithm 2 was the same. This is a general property for trap functions but further experiments are necessary to confirm the hypothesis that both methods have the same accuracy. Initial experiments show that, for problems where proper mixing is fundamental for success (deceptive problems), Algorithm 2 does not suffer from accuracy problems. Challenging problems like noisy problems and exponentially scaled problems might offer more difficulty and further refinements on Algorithm 2 might be necessary.

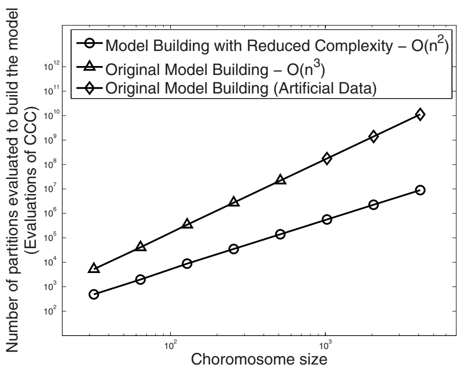

Fig. 1. A comparison of the scalability of the Model Building with reduced complexity and the Original Model Building shows that the proposed approach successfully reduces the number of different partition to be evaluated in one order of complexity, resulting in a speedup greater than 1000 times for individuals of size 4096 bits. The slope of the lines indicates a reduction from cubic to quadratic order.

Figure 1 shows the scalability of the methods on a LogLog scale. On the x axis we have the chromosome size and on the y axis we have the number of evaluations of the CCC necessary to build the correct model. Both the old (original) and the new method are straight lines, what means that they are governed by a power law. The slope of the line for the new method is close to 2 and for the old method close to 3, confirming the scalability hypothesis.

The improvement in the total runtime of the new method when compared to the old one is of order of 10 times for a small problem instance (32 bits) and of more than 1000 times for a 4096 bit problem.

3.2 Improving ECGA through Local Search

GAs are often able to operate over large and multi modal search spaces, presenting a good global search nature. However, local search methods are, in general, more efficient in tuning a solution and reaching the closest local optimum. These two characteristics are desirable and several hybrid approaches have been proposed. There are several ways to hybridize a GA with a local search method and each of them is more suitable to a different class of problems. In this work we propose to use the local search method as a pre-processing mechanism to the ECGA.

The reason for that decision is that the ECGA is very efficient at combining optimal sub-structures, but not so efficient at finding them. To bypass this inefficiency, large population sizes are required to ensure the initial BB supply [3]. Once that initial supply is provided, the ECGA can select promising solutions and learn the appropriate decomposition, building a model that allows it to combine the substructures properly.

The model-building process can become easier if the entropy of the partitions is low, i.e., if we need the smallest number of bits possible to represent that partition. The lowest entropy happens when all non-optimal instances of a BB have probability 0 and only one optimal solution is present. In this case, the entropy is 0.

However, sub-problems may be difficult to solve even if a proper decomposition is known. For instance, a trap function is a difficult function itself. We argue that if it is impossible or too expensive to find the optimal solution to a sub-problem, removing all non-locally-optimal solutions from the population still reduces the entropy significantly[1] and helps the model building process. A local search pre-processing can easily remove such non-locally-optimal solutions.

A local hill climber that processes each variable separately like the one in Algorithm 3, when applied to all individuals in the population, will have the effect of taking every instance of a substructure located on a particular hill on the search space to the peak of that hill. As an effect, all non-locally-optimal solutions will be replaced by a particular local-optimum, reducing the entropy and easing the model building process. Moreover, since fewer different instances of one sub-structure need to be recombined, the overall performance of the ECGA as a mixer of sub-structures will be improved (the mixing time is reduced [18]).

Algorithm 3. A hill climber to be used as a local search for the ECGA

```
1. Given an individual s of length l and fitness f:
2. For each position p in the chromosome:
3.   Flip the bit at p to its complementary value
4.   Evaluate the local change effect and calculate new fitness f'
5.   If f' is not better than f:
6.     Flip the bit p back to the original value
7.   else
8.     f = f'
9. Return the resulting individual
```

This reasoning proved to be true. The application of local search as a pre-processing to the ECGA reduces the population size required by the method, consequently reducing the memory and runtime of the algorithm. Figure 2 compares the minimal population size (determined according to the Bisection Method) required by the ECGA and by the Hybrid to solve problems of growing size (the x axis represents the problem size l). The population size in ECGA scales as $O(l \cdot log(l))$ [9] for problems with constant BB size. Figure 2 shows that the hybrid uses smaller population and scales no worse than the original method.

It is important to remark that the local search procedure is itself time consuming. We can use the locality and independence of the substructures to reduce the cost of the function evaluations required by the hill climber, introducing the concept of a local change evaluation, which evaluation calculates the fitness variation induced by a single bit change. In the mk-trap problem the cost of a local change evaluation is $1/m$ of the cost of a complete function evaluation.

[1] Given that the sub-problem is boundedly multimodal.

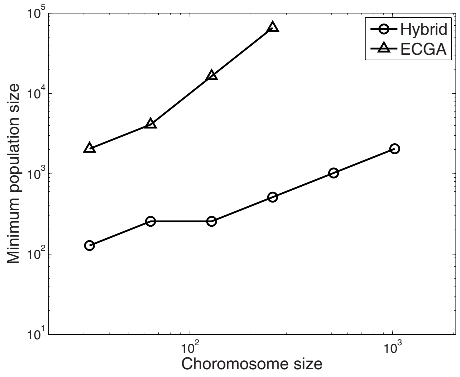

Fig. 2. The Hybridization of the ECGA with a local search procedure reduces the minimum population size required by the algorithm (determined using the bisection method [10]), reducing the memory requirements and potentially the running time

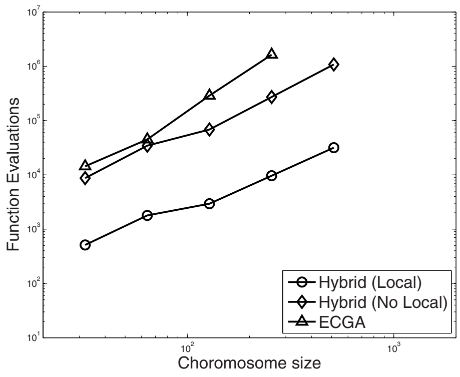

Fig. 3. The Hybridization of the ECGA also affects the number of function evaluations. Considering the cost of a local evaluation as $1/m$ of the cost a complete evaluation we have a significant improvement on the number of function evaluations required. Even when local change evaluations are not available, the number of function evaluations for the hybrid is no worse than the number of function evaluations for the original method. The hybrid still reduces the population size and, consequently, memory and running time.

Figure 3 show the scalability of the number of function evaluations required by the ECGA (ECGA) and the Hybrid. For the latter, two lines where presented, one of them (Hybrid - No Local) assumes that no local change evaluation is available, counting each evaluation as a complete function evaluation. The other line (Hybrid - Local) assumes the availability of local evaluations, counting m of them as one function evaluation. The x axis represents the problem size l.

Figure 3 shows that even in the worst case, the hybrid is no worse than the ECGA. When no local evaluation is available, the hybrid behaves very similar to the ECGA, but requires a smaller population, generating a smaller overhead for model building, selection and other GA related processes, reducing both memory and running time.

4 Future Work

This paper presents enhancements to the ECGA and successfully improves its performance. However, in order to create an effective EA several other enhancements are still necessary. In this section we point and describe some of the next steps toward such effective EA. The proposed enhancements are useful steps, but future work should not be restricted only to these options.

The proposed relaxed model builder shows no accuracy lost for trap functions. Noisy or exponentially scaled problems might be more challenging, requiring further enhancements to the algorithm.

Parallelization: Since the Hill Climber processes each individual independently it is easy to distribute the population over several processors, achieving a speedup near the number of processors for the pre-processing step. Parallelizing the model building deserves special care, since this process centers the information distributed in the population.

Model building improvements: It is possible to further speed-up the algorithm by avoiding unnecessary full model building steps. This can be achieved by three different ways: (I) Sporadic model building [19], which builds the model only after some generations or some important event, instead of every generation; (II) Once and forever model building, which build the model in the first generation and reuses it through the GA run and; (III) Incremental model building, which changes the model using small incremental steps.

Hybridization with competent mutation operator: as discussed in [9], the mutation is useful on deterministic problems, while crossover is useful on noisy problems. A mutation-crossover hybrid can take advantage of both strengths, as showed by [20].

Chromosome compression: compressing the chromosome as in [21] can also improve the performance of the algorithm by reducing the chromosome length and search space.

5 Conclusion

This work presents a proof of principle on the importance of efficiency enhancements for practical GAs. We present two enhancements to the ECGA, a widely used competent genetic algorithm. These enhancements successfully improved performance of the algorithm, with speedups of more than 1000 times for large problems.

The first enhancement, the change on the model building process, was able to reduce significantly the time needed for the model building process. This step represents more than 90% of the original algorithm's runtime. The extension was able to reduce one order of complexity in the model building process, inducing a speedup around 10 times faster for small instance (32 bits) and more than 1000 times faster for a 4096 bit problem.

The second enhancement, the hybridization with a local search preprocessing, successfully reduced the population size required by the algorithm, reducing the memory and runtime requirements. Using the hybrid we were able to solve the same problem with a population at least 32 times smaller. This result holds for small strings (such as 32 bits) and for the larger ones.

Although the results achieved by these enhancements are relevant, this work is only the first step toward a competent and efficient EA, capable of solving difficult large scale problems in practical time. We also pointed some of the next steps toward such EA.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0370, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center and under grant ISS-02-09199 at the National Center for Supercomputing Applications, UIUC. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
3. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2002)
4. Sastry, K., Goldberg, D., Pelikan, M.: Efficiency enhancment of probabilistic model building genetic algorithm. Technical report, Illinois Genetic Algorithms Laboratory, Univeristy of Illinois at Urbana Champaign, Urbana, IL (2004)
5. Cantu-Paz, E.: Designing Efficient and Accurate Parallel Genetic Algorithms. PhD thesis, University of Illinois at Urbana-Champaign, Illigal Report No 99017 (1999)
6. Goldberg, D.E., Voessner, S.: Optimizing Global-Local Search Hybrids. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 220–228. Morgan Kaufmann, San Francisco (1999)
7. Sinha, A., Goldberg, D.: A survey of hybrid genetic and evolutionary algorithms. Technical report, University of Illinois at Urbana Chapaign, Urbana, IL (1999) IlliGal Report No. 2003004
8. Sinha, A.: Designing efficient genetic and evolutionary algorithm hybrids, Master Thesis, University of Illinois at Urbana Champaign (2003) (IlliGal Report No. 2003020)

9. Sastry, K., Goldberg, D.: Let's Get Ready to Rumble: Crossover Versus Mutation Head to Head. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103. Springer, Heidelberg (2004)
10. Sastry, K.: Evaluation-relaxation Schemes for Genetic and Evolutionary Algorithms. PhD thesis, University of Illinois at Urbana-Champaign (2001)
11. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. In: Proceedings of IEEE Iternational Conference on Evolutionary Computation (1998), pp. 523–528 (1998)
12. Goldberg, D., Sastry, K., Llorà, X.: Toward routine billion-variable optimization using genetic algorithms: Short Communication. Complexity 12(3), 27–29 (2007)
13. Sastry, K., Goldberg, D., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584 (2007)
14. Harik, G.: Linkage Learning via probabilistic modeling in the ECGA. Technical report, University of Illinois at Urbana Chapaign, Urbana, IL (1999)
15. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
16. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proceedings of the Genetic And Evolutionary Computation Conference, pp. 524–532 (1999)
17. de la Ossa, L., Sastry, K., Lobo, F.: χ–ary Extended Compact Genetic Algorithm in C++. Technical report, Illigal Report 2006013, Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign (2006)
18. Thierens, D., Goldberg, D.: Mixing in Genetic Algorithms. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 38–47 (1993)
19. Pelikan, M., Sastry, K., Goldberg, D.: Sporadic model building for efficiency enhancement of hierarchical BOA. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 405–412. ACM Press, New York (2006)
20. Lima, C., Sastry, K., Goldberg, D., Lobo, F.: Combining competent crossover and mutation operators: a probabilistic model building approach. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 735–742 (2005)
21. Yu, T., Goldberg, D.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1385–1392 (2006)