Combining Competent Crossover and Mutation Operators: a Probabilistic Model Building Approach

Cláudio F. Lima
DEEI-FCT
University of Algarve
Campus de Gambelas
8000-117 Faro, Portugal
clima@ualg.pt

Kumara Sastry
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
ksastry@uiuc.edu

David E. Goldberg
Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Ave, Urbana, IL 61801
deg@uiuc.edu

Fernando G. Lobo
DEEI-FCT
University of Algarve
Campus de Gambelas
8000-117 Faro, Portugal
flobo@ualg.pt

ABSTRACT

This paper presents an approach to combine competent crossover and mutation operators via probabilistic model building. Both operators are based on the probabilistic model building procedure of the extended compact genetic algorithm (eCGA). The model sampling procedure of eCGA, which mimics the behavior of an *idealized* recombination—where the building blocks (BBs) are exchanged without disruption—is used as the competent crossover operator. On the other hand, a recently proposed BB-wise mutation operator—which uses the BB partition information to perform local search in the BB space—is used as the competent mutation operator. The resulting algorithm, called hybrid extended compact genetic algorithm (heCGA), makes use of the problem decomposition information for (1) effective recombination of BBs and (2) effective local search in the BB neighborhood. The proposed approach is tested on different problems that combine the core of three well known problem difficulty dimensions: deception, scaling, and noise. The results show that, in the absence of domain knowledge, the hybrid approach is more robust than either single-operator-based approach.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search; I.2.6 [Artificial Intelligence]: Learning.

General Terms: Algorithms, Performance.

Keywords: Competent Genetic Algorithms, Probabilistic Model Building Genetic Algorithms, Hybridization, BB-wise Mutation.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

1. INTRODUCTION

Genetic Algorithms (GAs) that solve hard problems quickly, reliably, and accurately are known as *competent* GAs [4]. In contrast to traditional GAs, competent GAs use recombination operators that are able to capture and adapt themselves to the underlying problem structure. In this way, competent GAs successfully solve decomposable problems with bounded difficulties within a low-order polynomial number of function evaluations [4].

Basically, competent GAs take problems that were intractable to traditional GAs and renders them tractable requiring in certain cases only a subquadratic number of function evaluations. However, for large-scale problems, even a subquadratic number of fitness evaluations can be very demanding. This is especially true if the fitness evaluation requires a complex simulation or computation. So, while competence leads a problem intractable to tractable, efficiency enhancement takes it from tractable to practical. One of the efficiency-enhancement techniques for GAs is *hybridization.*

Hybridization typically involves combining the global-search capabilities of genetic algorithms with local-search methods that often includes domain- or problem-specific knowledge [9, 8, 18]. While hybridization is often used in applying GAs to solve real-world problems, systematic methods for hybridizing and designing competent global and local-search methods that automatically identify the problem decomposition and important problem substructures are however lacking.

Therefore, in this paper we present a hybridization of a competent recombination operator that effectively exchanges key substructures of the problem, and a competent mutation operator that efficiently searches for best substructures in the building blocks (BBs) partition. Specifically, we use the probabilistic model-building methodology of extended compact genetic algorithm (eCGA) [7] to determine the ef-

fective problem decomposition and the important substructures (or building blocks) of the underlying search problem. The probabilistic model, which automatically induces good neighborhoods, is subsequently used for two distinct purposes:

1. Effective recombination of BBs that provides rapid global-search capabilities.
2. Effective search in the BB neighborhood that locally provides high-quality solutions [15].

The key idea is to obtain the benefits from both approaches, recombination without disrupting the BBs, and mutation (local search) that rapidly searches for the best BBs in each partition.

The paper starts by reviewing some of the work done on the discussion "crossover versus mutation" and outlining the motivation for the present work. Section 3 introduces the extended compact GA, and its selectomutative counterpart is described in the subsequent section. Next, in Section 5, we describe the proposed hybrid extended compact genetic algorithm (heCGA) and outline other possible hybridization configurations. In Section 6, computational experiments are performed, in different problem difficulty dimensions, to evaluate the behavior of the proposed approach. Finally, we present a summary and conclusions.

2. CROSSOVER *VERSUS* MUTATION

Since the early days in the genetic and evolutionary computation (GEC) field, one of the hot topics of discussion has been the benefits of crossover versus mutation and viceversa. Crossover and mutation search the genotype space in different ways and with different resources. While crossover needs large populations to effectively combine the necessary information, mutation works best when applied to small populations during a large number of generations.

In genetic algorithms, significant attention has been paid to the design and understanding of recombination operators. Systematic methods of successfully designing competent selectorecombinative GAs have been developed based on decomposition principles [4]. Mutation, on the other hand, is usually a secondary search operator which performs a random walk locally around a solution and therefore has received far less attention. However, in evolutionary strategies (ESs) [14], where mutation is the primary search operator, significant attention has been paid to the development of mutation operators. Several mutation operators, including adaptive techniques, have been proposed [14, 17, 1, 2, 6]. The mutation operators used in ESs are powerful search operators (specially for continuous domains), however, the neighborhood information is still local around a single or few solutions. In fact, when solving boundedly difficult GA-hard problems, local neighborhood information is not sufficient, and a mutation operator which uses local neighborhood requires $\mathcal{O}(l^k \log l)$ function evaluations [10] (being l the problem size and k the BB size), which for moderate values of k, grows extremely fast and the search becomes inefficient compared to competent GAs.

Spears [19] did a comparative study between crossover and mutation operators, and showed that there were some important features of each operator that were not captured by the other. These results provide a theoretical justification for the fact that the role of crossover is the construction of high-order BBs from low-order ones. Clearly, mutation can not perform this role as well as crossover. However, in terms of disruption, mutation can provide higher levels of disruption and exploration, but at the expense of preserving alleles common to particular defining positions [19].

Sastry and Goldberg [16] analyzed the relative advantages between crossover and mutation on a class of deterministic and stochastic additively separable problems. For that study, the authors assumed that the crossover and mutation operators had perfect knowledge of the BBs partition and effectively exchanged or searched among competing BBs. They used facetwise models of convergence time and population sizing to determine the scalability of each operator-based algorithm. The analysis shows that for additively separable deterministic problems, the BB-wise mutation is more efficient than crossover, while for the same problems with additive Gaussian noise the crossover-based algorithm outperforms the mutation approach. The results show that the speed-up of using BB-wise mutation on deterministic problems is $\mathcal{O}(\sqrt{k}\log m)$, where k is the BB size and m is the number of BBs. In the same way, the speed-up of using crossover on stochastic problems with fixed noise variance is $\mathcal{O}(\sqrt{k}m/\log m)$.

Over the years, several researchers have identified that the robustness and strengths of GAs lies in using crossover *and* mutation. One possible approach to combine the best features of both operators is *hybridization*, which has often been used in GAs as an efficiency-enhancement technique [8]. Hybrid GAs combine the typical steps of GAs with local search methods that use some sort of domain or problem specific knowledge. Many applications of GAs in industry follow this approach in order to gain from the benefits of hybridization. Hybrid GAs are also referred as memetic algorithms [9] or genetic local search methods.

However, most hybridization methods are ad hoc and automatic methods of identifying and exploiting problem decomposition in both global search and local search methods are lacking. Therefore, in this paper, we investigate the efficiency-enhancement capabilities of combining competent recombination and mutation operators. Specifically, we focus on probabilistic-model-building-based operators to propose a *competent hybrid* GA. In probabilistic model building genetic algorithms (PMBGAs) the variation operators are replaced by building and sampling a probabilistic model of promising solutions. This procedure tries to mimic the behavior of an ideal crossover operator, where the BBs are mixed without disruption. One of the state-of-the-art PMBGAs is the extended compact genetic algorithm [7], that uses marginal product models (MPMs) to represent the problem decomposition. Based on this probabilistic model building procedure, Sastry and Goldberg [15] recently proposed a BB-wise mutation operator that performs local search in the building block space. In the same line of work, we propose a *competent hybrid* GA that combines a BB-wise crossover operator with a BB-wise mutation operator via probabilistic model building. We name this approach as probabilistic model building hybrid genetic algorithm (PMBHGA). Note that conceptually a PMBHGA is different from a typical hybrid PMBGA in the sense that the local search that is performed is based on the probabilistic model instead of using specific problem knowledge, which turn it into a more general applicable hybridization.

3. EXTENDED COMPACT GA

The extended compact genetic algorithm (eCGA) [7] is based on the idea that the choice of a good probability distribution for promising solutions is equivalent to linkage learning. The eCGA uses a product of marginal distributions on a partition of genes. This kind of probability distribution belongs to a class of probability models known as marginal product models (MPMs). For example, the following MPM, `[1,3][2][4]`, for a 4-bit problem represents that the 1^{st} and 3^{rd} genes are linked, and the 2^{nd} and 4^{th} genes are independent.

In eCGA, both the structure and the parameters of the model are searched and optimized to best fit the data (promising solutions). The measure of a good MPM is quantified based on the minimum description length (MDL) principle, that penalizes both inaccurate and complex models, thereby leading to an near-optimal distribution. Formally, the MPM complexity is given by the sum of model complexity, C_m, and compressed population complexity, C_p. The model complexity, C_m, quantifies the model representation in terms of the number of bits required to store all the marginal probabilities. Let a given problem of size l with binary encoding, have m partitions with k_i genes in the i^{th} partition, such that $\sum_{i=1}^{m} k_i = l$. Then each partition i requires $2^{k_i} - 1$ independent frequencies to completely define its marginal distribution. Taking into account that each frequency is of size $\log_2(n+1)$, where n is the population size, the model complexity C_m is given by

$$C_m = \log_2(n+1) \sum_{i=1}^{m} (2^{k_i} - 1). \quad (1)$$

The compressed population complexity, C_p, quantifies the data compression in terms of the entropy of the marginal distribution over all partitions. Therefore, C_p is given by

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2(p_{ij}), \quad (2)$$

where p_{ij} is the frequency of the j^{th} gene sequence of the genes belonging to the i^{th} partition. In other words, $p_{ij} = N_{ij}/n$, where N_{ij} is the number of chromosomes in the population (after selection) possessing bit sequence $j \in [1, 2^{k_i}]$ for the i^{th} partition. Note that a BB of size k has 2^k possible bit sequences where the first is denoted by 00...0 and the last by 11...1.

As we can see in Figure 1, the extended compact GA is similar to a traditional GA, where the variation operators (crossover and mutation) are replaced by the probabilistic model building and sampling procedures. The offspring population is generated by randomly choosing subsets from the current individuals, according to the probabilities of the subsets stored in the MPM.

Analytical models have been developed for predicting the scalability of PMBGAs [11, 13]. In terms of number of fitness evaluations necessary to converge to the optimal solution, these models predict that for additively separable problems the eCGA scales subquadratically with the problem size: $\mathcal{O}(2^k \sqrt{k} m^{1.5} \log m)$. Sastry and Goldberg [15] empirically verified this scale-up behavior for the eCGA.

***Extended Compact Genetic Algorithm* (eCGA)**

(1) Create a random population of n individuals.

(2) Evaluate all individuals in the population.

(3) Apply s-wise tournament selection [5].

(4) Model the selected individuals using a greedy MPM search procedure.

(5) Generate a new population according to the MPM found in step 4.

(6) If stopping criteria is not satisfied, return to step 2.

Figure 1: Steps of the extended compact genetic algorithm (eCGA).

***Extended Compact Mutation Algorithm* (eCMA)**

(1) Create a random population of n individuals and evaluate their fitness.

(2) Apply s-wise tournament selection [5].

(3) Model the selected individuals using a greedy MPM search procedure.

(4) Choose the best individual of the population for BB-wise mutation.

(5) For each detected BB partition:

(5.1) Create $2^k - 1$ unique individuals with all possible schemata in the current BB partition. Note that the rest of the individual remains the same and equal to the best solution found so far.

(5.2) Evaluate all $2^k - 1$ individuals and retain the best for mutation in the other BB partitions.

Figure 2: Steps of the extended compact mutation algorithm (eCMA).

4. PROBABILISTIC MODEL BUILDING BB-WISE MUTATION ALGORITHM

The probabilistic model building BB-wise mutation algorithm (BBMA) [15] is a selectomutative algorithm that performs local search in the building block neighborhood. Instead of using a bit-wise mutation operator that scales polynomially with order k as the problem size increases, the BBMA uses a BB-wise mutation operator that scales subquadratically. For BB identification, the authors [15] used the probabilistic model building procedure of eCGA. However, other probabilistic model building techniques can be used with similar or better results. In this work, we restrict ourselves to the probabilistic model building procedure of eCGA, so from now on we refer to this instance of BBMA as the extended compact mutation algorithm (eCMA). Once the linkage groups are identified, an enumerative BB-wise mutation operator [16] is used to find the best schema for each detected partition. A description of the eCMA can be seen in Figure 2.

The performance of the BBMA can be slightly improved by using a greedy heuristic to search for the best among competing BBs in each partition. Even so, the scalability of BBMA is determined by the population size required to accurately identify the BB partitions. Therefore, the number of function evaluations scales as $\mathcal{O}(2^k m^{1.05}) \leq n_{fe} \leq \mathcal{O}(2^k m^{2.1})$ [15, 13].

It should be also noted that on BBMA the linkage identification is only done at the initial stage. This kind of offline linkage identification works well on problems of nearly equal salience, however, for problems with non-uniformly scaled BBs, the linkage information needs to be updated at regular intervals. This limitation will be empirically shown in our experimental results.

5. PROBABILISTIC MODEL BUILDING HYBRID GENETIC ALGORITHM

In Sections 3 and 4, we presented two *competent* operators for solving additively decomposable hard problems, based on the probabilistic model building procedure of eCGA. Therein, we use the same procedure to build the probabilistic model and combine both operators in the same algorithm. Similar to eCGA, our hybrid extended compact genetic algorithm (heCGA) models promising solutions in order to be able to effectively recombine the BBs and perform effective local search in their space.

As we can see in Figure 3, the heCGA starts like the regular eCGA (steps 1-4), but after the model is built the linkage information is used to perform BB-wise mutation in the best individual of the population. After that, heCGA updates the BB frequencies of the model (found on step 4) based on the BB instances of the mutated solution. This is done by increasing the frequency of each BB instance of the new best individual (the one that was mutated) by s and decreasing each BB instance of the previous best solution by s, where s is the tournament selection size. Note that we can also replace the copies of the best individual by the mutated one, with a similar overall effect. Finally, we generate a new population according to the updated model, and repeat these steps until some stopping criteria is satisfied.

It should be noted that eCGA, and consequently eCMA and heCGA, can only build linkage groups with non-overlapping genes. However, the BB-wise mutation operator and this BB-wise hybrid GA can be extended to other linkage identification techniques that can handle overlapping BBs such as the Bayesian optimization algorithm (BOA) [12] or the dependency structure matrix driven genetic algorithm (DSMDGA) [20].

As mentioned earlier, the performance of the BB-wise mutation operator can be slightly improved using a greedy procedure to search for the best among competing BBs. This can be particularly useful if we consider other ways to integrate BB-wise mutation with BB-wise crossover. An alternative way to combine these operators would be to apply a stochastic BB-wise mutation to all individuals in the population. This way, instead of having the traditional bit-wise mutation with a certain probability to be applied to each bit, we would have a BB-wise mutation with a certain probability to be applied to each BB partition in each individual. In this kind of scheme it is important to spend less than $2^k - 1$ function evaluations when searching for each optimal BB schema, specially if we use high probabilities of applying BB-wise mutation.

***Hybrid Extended Compact Genetic Algorithm* (heCGA)**

(1) Create a random population of n individuals.

(2) Evaluate all individuals in the population.

(3) Apply s-wise tournament selection [5].

(4) Model the selected individuals using a greedy MPM search procedure.

(5) Apply BB-wise mutation to the best individual.

(6) Update the frequencies of the MPM found on step 4 according to the BBs instances present on the mutated individual:

  (6.1) Increase the BB instances frequencies of the mutated individual by s.

  (6.2) Decrease the BB instances frequencies of the previous best individual by s.

(7) Generate a new population according to the updated MPM.

(8) If stopping criteria is not satisfied, return to step 2.

Figure 3: Steps of the hybrid extended compact genetic algorithm (heCGA).

Another approach is to heuristically choose which individuals will be BB-mutated, and instead of mutating all BBs just mutate one or some randomly (or again heuristically) chosen. For example, a clustering criteria can be used where only the best individual from each cluster is mutated. In this paper, we limit our study to the first proposed hybrid scheme using deterministic BB search and leave the other possibilities as future work.

6. EXPERIMENTS

In this section we perform computational experiments in various problems of bounded difficulty. Following a design approach to problem difficulty [4], we test the described algorithms on a set of problems that combine the core of three well known problem difficulty dimensions:

1. Intra-BB difficulty: Deception;
2. Inter-BB difficulty: Scaling;
3. Extra-BB difficulty: Noise.

For that, we assume that the problem at hand is additively decomposable and separable, such that

$$f(X) = \sum_{i=0}^{m-1} f_i(x_{I_i}), \tag{3}$$

where I_i is the index set of the variables belonging to the i^{th} subfunction. As each subfunction is separable from the rest, each index set I_i is a disjoint tuple of variable indexes.

For each algorithm, we empirically determine the minimal number of function evaluations to obtain a solution with at least $m-1$ building blocks solved, that is, the optimal solution with an error of $\alpha = 1/m$. For eCGA and eCMA, we use a bisection method over the population size to search for the minimal sufficient population size to achieve a target solution. However, for heCGA an interval halving method is more appropriate given the algorithm behavior as the population increases, as will be shown later (Figure 5). The results for the minimal sufficient population size are averaged over 30 bisection runs. In each bisection run, the number of BBs solved with a given population size is averaged over another 30 runs. Thus, the results for the number of function evaluations and the number of generations spent are averaged over 900 (30x30) independent runs. For all experiments, tournament selection without replacement is used with size $s = 8$.

6.1 Problem 1: Deception

As the core of intra-BB difficulty, deceptive functions are among the most challenging problems for competent GA candidates. This kind of functions normally have one or more deceptive optima that are far away (in the genotype space) from the global optimum and which misleads the search in the sense that the attraction area of the deceptive optima is much greater than the one of the optimal solution. A well known deceptive function is the k-trap function [3] defined as follows:

$$f_{trap}(u) = \begin{cases} 1 & \text{if } u = k \\ 1 - d - u * \frac{1-d}{k-1} & \text{otherwise} \end{cases} \quad (4)$$

where u is the number of 1s in the string, k is the size of the trap function, and d is the fitness signal between the global optimum and the deceptive optimum. In our experiments we use $d = 1/k$. Considering m copies of this trap function, the global boundedly deceptive function is given by

$$f_d(X) = \sum_{i=0}^{m-1} f_{trap}(x_{ki}, x_{ki+1}, \ldots, x_{ki+k-1}). \quad (5)$$

Figure 4 presents the results obtained for the boundedly deceptive function. The number of BBs (or subfunctions) is varied between 2 and 20, for $k = 4$. As we can see, eCGA needs smaller populations than eCMA and heCGA to solve the problem, however, takes more function evaluations than both algorithms. This happens because in eCGA (1) the BBs are discovered in a progressive way and (2) more generations are required to exchange the right BBs. Although increasing the population size for eCGA accelerates the BB identification process, additional generations are still needed to mix the correct BBs into a single individual. Since eCGA (like every selectorecombinative GA) always have to spend this mixing time, relaxing the BB identification process (using smaller populations, thus saving function evaluations) to a certain point seems to be the best way to tune eCGA performance.

The scalability difference between eCGA and eCMA is not surprising and was verified before [15, 16]. The similarity between eCMA and heCGA performances leads us to conclude that the best way to use heCGA on deterministic and uniformly scaled boundedly deceptive functions, and the problems that are bounded by this one, is to set a large

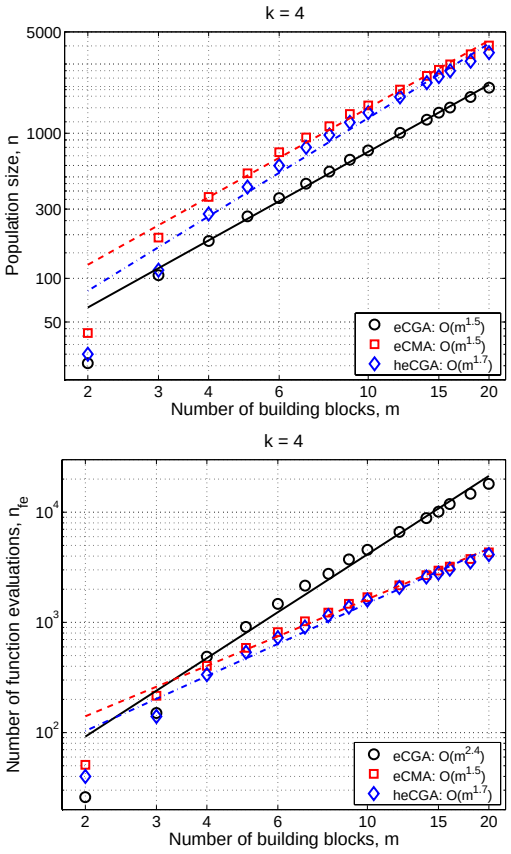

Figure 4: Population size and number of function evaluations required by eCGA, eCMA, and heCGA to successfully solve $m-1$ BBs for the boundedly deceptive function with $k = 4$ and $m = [2, 20]$. The results for population size are averaged over 30 runs, while the number of function evaluations is averaged over 900 independent runs.

enough population size to get the problem structure in the first generation, and then perform BB local search to achieve the global optimum.

These results suggest that there is no direct gain of heCGA over eCMA for this problem, however, there is another observation that can be made. From a practitioner point of view, heCGA is a more flexible search algorithm since it gets the optimal solution within a bigger range of population size values. In Figure 5, the number of function evaluations for heCGA to get the target solution (for $k = 4$ and $m = 10$), as the population size increases, is shown. Only population sizes that solve $m-1$ BBs on average (over 30 runs) are shown in the plot. The plotted points form four increasing lines. In each line, as the population increases the number of function evaluations also increases until it falls down into a lower line and then keeps increasing again. This behavior repeats itself until the population size is enough to discover all[1] correct BB partitions in the first generation, being the problem solved by the enumerative BB local search procedure of heCGA in the initial generation. Each discon-

[1]In our specific case, $m-1$ partitions due to the stopping criteria used.

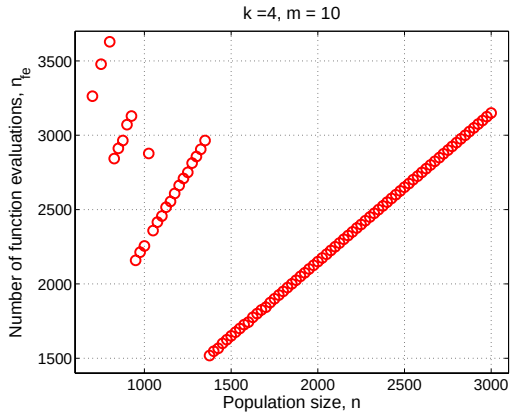

Figure 5: Number of function evaluations required by heCGA to successfully solve $m-1$ BBs for the boundedly deceptive function with $k=4$ and $m=10$. The results are averaged over 30 runs. Only population sizes that solve $m-1$ BBs on average are shown.

tinuity between lines represents a decrease in the number of generations necessary for heCGA to successfully solve the problem. This happens because, as the population size is increased, the model building procedure can capture more and more correct BB partitions, improving the ability of BB local search to quickly solve the problem.

6.2 Problem 2: Deception + Scaling

In this problem, the inter-BB difficulty is explored together with the intra-BB difficulty. Here, we use the boundedly deceptive function used above, but now each subfunction fitness contribution to the overall fitness is exponentially scaled. The weight of each BB fitness contribution is given by powers of 2, being our exponentially scaled boundedly deceptive function defined as

$$f_{ds}(X) = \sum_{i=0}^{m-1} 2^i f_{trap}(x_{ki}, x_{ki+1}, \ldots, x_{ki+k-1}) \qquad (6)$$

This function has the interesting property that a high scaled subfunction gives more fitness contribution than the sum of all subfunctions below it. When solving this problem with a GA in the initial generations, the signal that comes from the low-salient BBs is negligible when faced with the decision making that is being done between the high-salient BBs. Whenever the higher BBs are solved, the next higher scaled BBs will have their time of attention by the GA, and so on. Given this property, the correct BB partitions can only be discovered in a sequential way, which contrast with the uniformly scaled case where the problem structure can be captured in the first generation with a sufficient large population size. Therefore, eCMA is not able to solve exponentially scaled problems with reasonable population sizes, as was point out before [15]. The model built based on the selected initial random individuals will only be able to get the high-salient BB partitions, failing the rest. As Sastry and Goldberg [15] proposed for future work, the model of eCMA has to be updated at a regular schedule to be able to capture the BBs structure in a sequential manner.

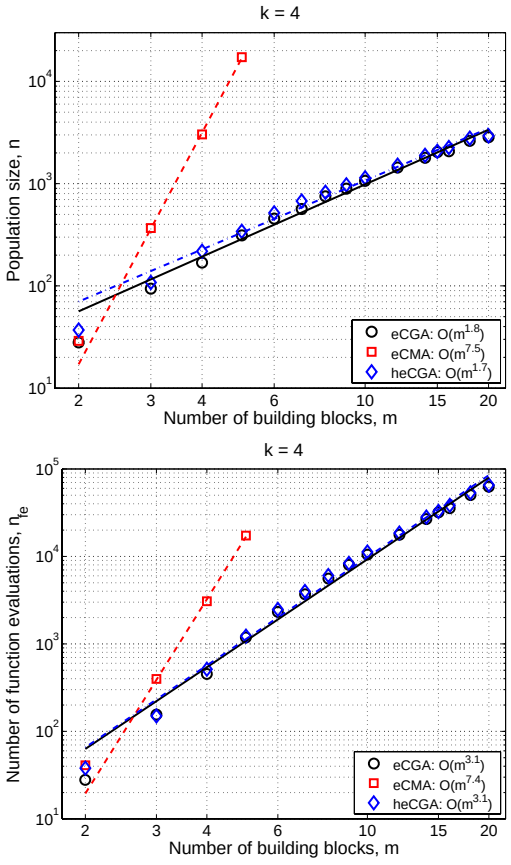

Figure 6: Population size and number of function evaluations required by eCGA, eCMA, and heCGA to successfully solve $m-1$ BBs for the exponentially scaled boundedly deceptive function with $k=4$ and $m=[2,20]$. The results for population size are averaged over 30 runs, while the number of function evaluations is averaged over 900 independent runs.

Figure 6 empirically shows that eCMA needs exponential population sizes to achieve the target solution. In heCGA the model is updated every generation and the BB-wise mutation can benefit from that. Nevertheless, heCGA spends approximately the same number of function evaluations to solve the problem than the regular eCGA. In this case, heCGA behaves similarly to eCGA, preferring a reasonable population size, enough to get the most relevant BBs and then keep going sequentially to the remaining ones.

Figure 7 shows the number of function evaluations that heCGA needs to solve this problem as the population size increases. Here, we can see that the number of function evaluations grows almost linearly with the population size. Since increasing the population size won't reveal much more correct BB partitions, the effect on the overall search process is minor.

Looking at the behavior of heCGA on both uniformly and exponentially scaled problems, we can observe distinct dynamics for each problem. In the uniformly scaled case, heCGA has a similar behavior to eCMA, which is the algorithm that performs better. For the exponentially scaled problem, heCGA changes completely its dynamics behav-

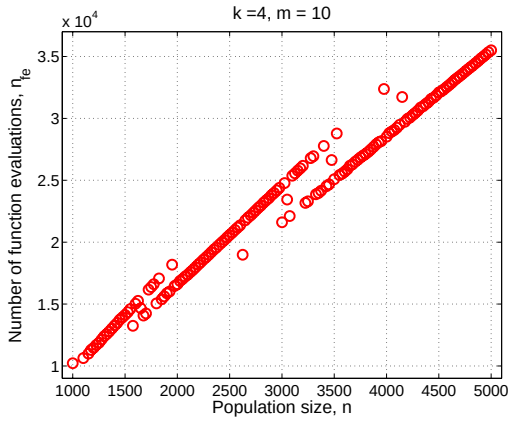

Figure 7: Number of function evaluations required by heCGA to successfully solve $m-1$ BBs for the exponentially scaled boundedly deceptive function with $k=4$ and $m=10$. The results are averaged over 30 runs. Only population sizes that solve $m-1$ BBs on average are shown.

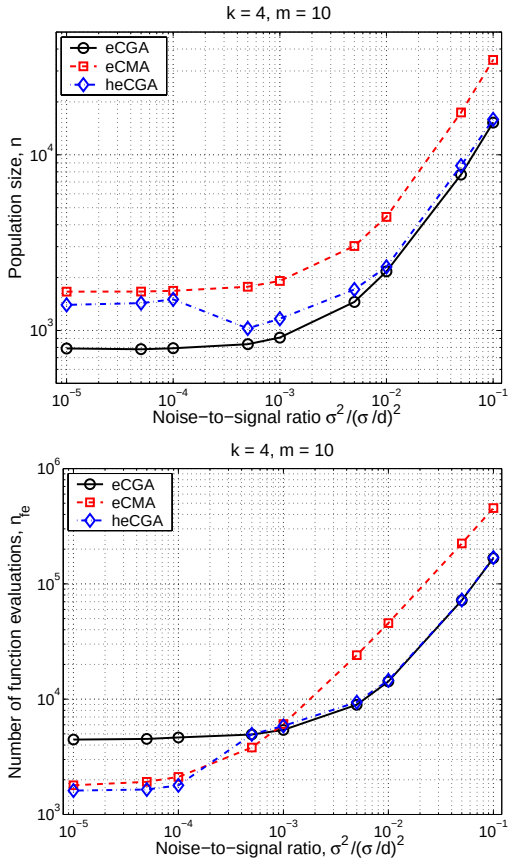

Figure 8: Population size and number of function evaluations required by eCGA, eCMA, and heCGA to successfully solve $m-1$ BBs for the noisy boundedly deceptive function with $k=4$ and $m=10$. The results for population size are averaged over 30 runs, while the number of function evaluations is averaged over 900 independent runs.

ing like eCGA, that is known to perform much better than eCMA. Also in this case, no direct gain is achieved by heCGA over the best algorithm. Nevertheless, we can observe what seems to be the greatest advantage of the proposed approach: robustness. For both problems, heCGA obtains the same performance as the one obtained by the best algorithm for each domain.

Additionally, the same experiments were performed for $k=5$ and similar qualitative results were obtained, confirming the observed robustness for moderate increases on subfunction size k. To get a better insight on these observations, we perform additional experiments with a problem with additive exogenous noise, which is considered to be the core of the extra-BB difficulty dimension.

6.3 Problem 3: Deception + Noise

Noise is a common factor in many real-world optimization problems. Sources of noise can include physical measurement limitations, incomplete sampling of large spaces, stochastic simulation models, human-computer interaction, among others. Furthermore, evaluation-relaxation techniques are commonly used in genetic and evolutionary algorithms (GEAs) for performance enhancement, bringing an additional source of noise to the original optimization problem. Thus, analyzing the heCGA performance in noisy environments is important to strengthen the robustness claims verified for the first two problems.

For our experiments, we assume that the exogenous noise follows a Gaussian distribution with mean 0 and variance σ_N^2. To make the problem even more challenging, we try to optimize a noisy version of the uniformly scaled boundedly deceptive function used before. This function is defined as follows

$$f_{dn}(X) = f_d(X) + \mathrm{G}(0, \sigma_N^2) \tag{7}$$

To overcome the noise with eCMA, each function evaluation in the BB local search phase needs to be performed over an average of function evaluations. The number of times that each individual needs to be evaluated, to allow correct decision making between competing BBs, depends on the noise variance. Therefore, to obtain the optimal results for eCMA in noisy conditions we need to run 2 bisections methods, one over the initial population size and the other over the number of samples that is necessary to correctly evaluate an individual. First, we run a bisection method to get the minimal population size that generates a model with at least $m-1$ correct BB partitions. Then, for each population that captures the target dependencies, a second bisection method is performed over the number of fitness samples to obtain the minimal number of times that an individual needs to be evaluated, in order to achieve a final solution with the BBs detected by the model optimally solved.

Figure 8 depicts the results obtained for a uniform scaled boundedly deceptive function with additive noise for $k=4$ and $m=10$. As the noise-to-signal ratio $\sigma_N^2/(\sigma_f/d)^2$ increases, two different scenarios can be identified. For small values of noise, as $\sigma_N^2/(\sigma_f/d)^2 \rightarrow 0$, the picture painted here is somewhat similar to the deterministic case, where eCMA and heCGA perform better than eCGA. When the noise increases, eCGA starts to perform better than eCMA,

which is expected given that crossover is likely to be more useful than mutation in noisy environments [16]. However, heCGA, which was behaving like eCMA (using bigger population sizes to solve the problem in the first generation) to small noise values, starts performing similarly to eCGA, that is known to be a best approach than eCMA to moderate-to-high noise values. This change in heCGA behavior can be better observed in the population size plot.

Note that in heCGA the BB local search phase doesn't use the averaging technique used in eCMA, since we want to test heCGA in various difficulty dimensions as a black-box method. Based on this results, the robust behavior of heCGA still stands for noisy conditions, confirming the observations made in Sections 6.1 and 6.2.

7. SUMMARY & CONCLUSIONS

In this paper, we have proposed a probabilistic model building hybrid GA based on the mechanics of eCGA. The proposed algorithm—the hybrid extended compact GA—combines the BB-wise crossover operator from eCGA with a recently proposed BB-wise mutation operator that is also based on the probabilistic model of eCGA [15]. Basically, heCGA makes use of the BBs partition information to perform (1) effective recombination of BBs and (2) effective local search in the space of BBs. We performed experiments on three different test functions that combine important difficulty dimensions: deception, scaling, and noise. Our results showed that, independently from the faced difficulty dimension(s), the hybrid extended compact GA obtained the best performance, imitating the behavior of the best approach (crossover-based or mutation-based) for each problem.

The results presented in this work indicate the robustness of using both search operators—crossover and mutation—in the context of PMBGAs, as it is known to be advantageous for traditional GEAs. Given the observed robustness of heCGA, one can think of applying it in a "black-box system" basis, where the solver is expected to perform well on problems bounded by the ones of our test suite.

Acknowledgments

C. Lima and F. Lobo were supported by the Portuguese Foundation for Science and Technology (FCT/MCES) under grants POSI/SRI/42065/2001 and SFRH/BD/16980/2004.

This work was also sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163 and F49620-03-1-0129, the National Science Foundation under grant DMI-9908252, ITR grant DMR-99-76550 (at Materials Computation Center), and ITR grant DMR-0121695 (at CPSD), and the Dept. of Energy through the Fredrick Seitz MRL (grant DEFG02-91ER45439) at 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.

Finally, the authors would like to thank the anonymous reviewers for helpful comments and suggestions.

8. REFERENCES

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[2] H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1996.

[3] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2*, pages 93–108, 1993.

[4] D. E. Goldberg. *The Design of Innovation - Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA, 2002.

[5] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[6] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[7] G. R. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[8] W. E. Hart. *Adaptive global optimization with local search*. PhD thesis, University of California, San Diego, San Diego, CA, 1994.

[9] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, CA, 1989.

[10] H. Mühlenbein. How genetic algorithms really work: I.Mutation and Hillclimbing. In R. Männer and et al., editors, *Parallel Problem Solving from Nature 2*, pages 15–25. Elsevier Science, 1992.

[11] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Bayesian optimization algorithm, population sizing, and time to convergence. In D. Whitley and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 275–282. Morgan Kaufmann, 2000.

[12] M. Pelikan, D. E. Goldberg, and E. Cant-Paz. BOA: The Bayesian Optimization Algorithm. In W. Banzhaf and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pages 525–532. Morgan Kaufmann, 1999.

[13] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2003.

[14] I. Rechenberg. *Evolutionsstrategie Optimierung technischer systeme nach prinzipien der biologischen evolution*. Friedrich Frommann Verlag, Stuttgart-Bad Cannstatt, 1973.

[15] K. Sastry and D. E. Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. In K. Deb and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004), Part II, LNCS 3103*, pages 114–125. Springer, 2004.

[16] K. Sastry and D. E. Goldberg. Let's get ready to rumble: Crossover versus mutation head to head. In K. Deb and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004), Part II, LNCS 3103*, pages 126–137. Springer, 2004.

[17] H.-P. Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. In *Interdisziplinäre Systemforschung*, chapter 1, pages 5–8. Birkhäuser Verlag, Basel and Stuttgart, 1977.

[18] A. Sinha. A survey of hybrid genetic and evolutionary algorithms. IlliGAL Report No. 2003004, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2003.

[19] W. M. Spears. Crossover or mutation? In D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 221–237. Morgan Kaufmann, 1993.

[20] T.-L. Yu, D. E. Goldberg, A. Yassine, and Y.-P. Chen. A genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In *Proceedings of the Artificial Neural Networks in Engineering 2003 (ANNIE 2003)*, pages 327–332, 2003.