7

Efficiency Enhancement of Estimation of Distribution Algorithms

Kumara Sastry, Martin Pelikan, and David E. Goldberg

Summary. Efficiency-enhancement techniques speedup the search process of estimation of distribution algorithms (EDAs) and thereby enable EDAs to solve hard problems in *practical* time. This chapter provides a decomposition and an overview of different efficiency-enhancement techniques for estimation of distribution algorithms. Principled approaches for designing an evaluation-relaxation, and a time-continuation technique are discussed in detail.

Key words: Efficiency enhancement, evolutionary computation, estimation of distribution algorithms, parallelization, evaluation relaxation, hybridization, time continuation, speedup

7.1 Introduction

A key challenge in genetic and evolutionary algorithm research is the design of *competent* genetic algorithms (GAs) that can solve *hard problems* quickly, reliably, and accurately. Estimation of distribution algorithms (EDAs) are one such class of competent GAs. In essence, EDAs take problems that were intractable with first-generation GAs and render them tractable, oftentimes requiring only a polynomial (usually subquadratic) number of fitness evaluations. However, for large-scale problems, the task of computing even a subquadratic number of function evaluations can be daunting. This is especially the case if the fitness evaluation is a complex simulation, model, or computation. For example, if a search problem requires over a million evaluations, and if each evaluation takes about 10 s, EDAs would take over 120 days to successfully solve the problem. This places a premium on a variety of *efficiency enhancement techniques*. In essence, while competence leads us from *intractability* to *tractability*, efficiency enhancement takes us from *tractability* to *practicality*. In addition to function evaluations, in EDAs, the probabilistic model building process can also be computationally intensive, especially with

K. Sastry et al.: *Efficiency Enhancement of Estimation of Distribution Algorithms*, Studies in Computational Intelligence (SCI) **33**, 161–185 (2006)
www.springerlink.com

increasing problem sizes, making a variety of *model-efficiency-enhancement techniques* also a necessity.

A distinct advantage of EDAs over many other evolutionary algorithms is that the probabilistic models contain useful information about problem structure that can be exploited in the principled design of various efficiency-enhancement methods. Systematically incorporating problem knowledge *mined* through the model-building process of EDAs into the design of an efficiency-enhancement technique makes it adaptive and can potentially enhance the speed-up of the method. For example, when a simple surrogate (approximate fitness function) is used as an alternative to an expensive and accurate fitness evaluation, we obtain a moderate speed-up of about 1.3[75]. On the other hand, when the probabilistic model is used to design a surrogate, we obtain a speed-up of about 50 [64]. That is, by incorporating problem knowledge contained in the probabilistic model into the design of the surrogate, we obtain about *39-fold* increase in the speed-up.

In this chapter, we present an overview of different efficiency-enhancement techniques, used to speedup not only the search process, but also the model-building process. We will also illustrate systematic and principled ways of incorporating and integrating the knowledge gained through probabilistic models in the efficiency-enhancement methods – specifically, evaluation relaxation [72], and time continuation [24] – to yield maximum speedup. Additionally, subsequent chapters will discuss in detail some of the efficiency-enhancement methods outlined here.

This chapter is organized as follows. We start with a brief outline of fundamental tradeoffs exploited by different EDA efficiency-enhancement methods and discuss four broad classes of efficiency enhancement techniques (1) Parallelization, (2) hybridization, and (3) time continuation, and (4) evaluation relaxation. We then provide examples of two principled efficiency-enhancement techniques (1) An evaluation-relaxation scheme where we build an endogenous fitness-estimate model using the probabilistic models built by EDAs – specifically, the Bayesian optimization algorithm (BOA) [62, also see chapter by Pelikan et al] – in Sect. 7.3, and (2) a time-continuation scheme where we develop a scalable mutation operator in the extended compact GA (eCGA) [37, also see chapter by Harik et al] that searches locally in the substructural neighborhood in Sect. 7.4. Summary and key conclusions are given in Sect. 7.5.

7.2 Decomposition of Efficiency Enhancement Techniques

In practical optimization problems we are often faced with limited computation resources, which brings forth different tradeoffs involving (1) time, which is the product of population size, number of generations per epoch, and the number of convergence epochs, and (2) solution quality assessment. Note that the time includes both the function-evaluation time and the EDA time (time

for selection, model building, model sampling, and replacement). It should be noted that the EDA time – especially the model building, sampling, and replacement – can be very significant and sometimes comparable to – if not more than – the function-evaluation time.

One or more of the following tradeoffs are exploited by efficiency-enhancement techniques to speedup EDAs:

Quality-Duration Tradeoff: Usually, the longer we run an EDA (with a sufficient population size), the higher will be the solution quality. However, in real-world scenarios, the computational resources are often limited, which leads to a tradeoff between solution quality and the search duration. Therefore, efficiency can be gained by choosing a search procedure that maximizes solution quality given the computational resource requirements. For example, quality-duration tradeoff might result in deciding between running a single epoch of an EDA with large population, as opposed to multiple epochs of the EDA with small population.

In addition to the search process, building a high quality model in EDAs might require longer time. On the other hand, reasonably accurate models might be built in less amount of time. Therefore, efficiency in EDAs can be gained by correctly deciding model accuracies during the search process.

Accuracy-Cost Tradeoff: Oftentimes, many complex real-world optimization problems involve computationally expensive function evaluation. However, an array of cheaper fitness functions can be easily developed, but at the cost of accuracy of fitness estimation. That is, the approximate fitness functions (or surrogates) suffer from various levels of error, and typically, cheaper the fitness function, larger the error in it. This introduces a tradeoff between fitness functions that are computationally cheap, but less accurate and fitness functions that are accurate, but computationally expensive. Therefore, we have to decide on the level of solution-quality assessment accuracy required during the search process, such that high-quality solutions can be obtained at minimum computational cost.

Additionally, in EDAs, a similar tradeoff exists between the probabilistic model accuracy and the cost. Typically, high-quality models are more expensive than low-quality models and striking an appropriate balance between model accuracy and model cost can significantly improve the model-building efficiency of EDAs.

Time Budget and Resource Allocation Tradeoff: Given limited computational resource its allocation in terms of population size, run duration, and number of convergence epochs can significantly influence the efficiency of the search algorithm. Time budgeting tradeoffs are often faced when distributing the EDA process between multiple processors (to strike a balance between communication and computation times), dividing the overall search time between different variation operators of an EDA such as crossover and

mutation, or dividing the search time between different local and global search methods.

Additionally, in EDAs time budgeting tradeoffs can also be faced when dividing resources between model building and model usage (in terms of exploration via model sampling and evaluation of sampled individuals).

Efficiency-enhancement techniques that exploit one or more of the aforementioned tradeoffs can be broadly classified into four categories:

Parallelization: EDAs are run on multiple processors and the computations are distributed among these processors [14]. The use of parallelization – of both the search process and the model-building process – in EDAs is discussed in detail elsewhere in this book (see chapter by Ocenasek et al).

Hybridization: Domain-specific knowledge and other techniques are coupled with EDAs to create a search bias and to accelerate the search process [16, 26, 39, 44, 54, 80]. In addition to traditional hybridization methods, prior knowledge can be incorporated in the probabilistic models of EDAs, details of which are provided elsewhere in this book (see chapter by Baluja). Additionally, the effectiveness of hybridizing EDAs with local search methods is empirically demonstrated for the spin-glass problems elsewhere in this book (see chapter by Pelikan and Hartmann).

Time continuation/utilization: Capabilities of both mutation and recombination are utilized to the fullest extent, and time budgeting issues are addressed depending on the problem type [24, 47, 73, 74, 82, 83]. In this chapter, we will illustrate a principled manner of incorporating neighborhood information, contained in the probabilistic models, with time-continuation operators to yield maximum speedup.

Evaluation relaxation: Accurate, but expensive fitness functions are replaced by less accurate, but inexpensive fitness functions (or surrogates), and thereby the total number of costly fitness evaluations is reduced [2, 8, 31, 43, 53, 64, 72, 75, 76, 81]. In this chapter, we will illustrate a principled approach for using substructural knowledge provided by probabilistic models of EDAs to develop an endogenous surrogate that can be used instead of the expensive fitness function to obtain high-quality solutions and thus provide maximum speedup. In addition to relaxing the solution-quality assessment measures, we can also relax the model-quality assessment in EDAs [66].

The speedup obtained by employing an efficiency-enhancement technique (EET) is measured in terms of a ratio of the computation effort required by an EDA when the is not used to that required when the EET is used. That is, $\eta = T_{\text{base}}/T_{\text{efficiency-enhanced}}$. The speedup obtained by employing even a single EET can potentially be significant. Furthermore, assuming that the performance of one of the above methods does not affect the performance of others, if we employ more that one EET, the overall speedup is the product

of individual speedups. That is, if the speedups obtained by employing parallelization, hybridization, time continuation and evaluation relaxation be η_{p}, η_{h}, η_{t}, and η_{e} respectively, then the overall speedup obtained is

$$\eta_{\mathrm{total}} = \eta_{\mathrm{p}}\eta_{\mathrm{h}}\eta_{\mathrm{t}}\eta_{\mathrm{e}}.$$

Even if the speedup obtained by a single EET is modest, a combination of two or more EETs can yield a significant speedup. For example, if we use a parallel EDA that yields linear speedup with 100 processors, and each of the other three EETs makes EDAs 25% more efficient, then together they yield a speedup of $100 * 1.25^3 = 195.3$. That is evaluation relaxation, time continuation, and hybridization would give slightly more than 95 processors' worth of additional computation power.

Before we demonstrate principled methodologies for utilizing information from probabilistic models of EDAs for maximum efficiency enhancement, we present a brief outline of each of the four classes of efficiency-enhancement techniques.

7.2.1 Parallelization

In parallelization, EDAs are run on multiple processors and the computations are distributed among these processors [13, 14]. Evolutionary algorithms are by *nature* parallel, and many different parallelization approaches such as a simple master-slave [9, 30], coarse-grained [32, 67, 84], fine-grained [27, 28, 50, 70], or hierarchical [23, 29, 33, 48] architectures can be readily used. Regardless of how parallelization is done, the key idea is to distribute the computational load of EDAs on several processors thereby speeding-up the search process. A principled design theory exists for developing an efficient parallel GA and optimizing the key facts of parallel architecture, connectivity, and deme size [14], some of which are discussed in the next chapter. Apart from parallelizing the function evaluations, the probabilistic model building process can also be parallelized [56–58] which is also discussed in the chapter by Ocenasek et al.

7.2.2 Hybridization

In hybridization, domain-specific knowledge and other local-search techniques are coupled with evolutionary algorithms to obtain high-quality solutions in reasonable time [16, 26, 39, 44, 54, 80]. Most industrial-strength evolutionary algorithms employ some sort of local search for a number of reasons such as achieving faster convergence [12, 39, 80], repairing infeasible solutions into legal ones [42, 59], initializing the population [21, 68], and refining of solutions obtained by a GA [41]. In addition to traditional ways of hybridizing EDAs [34, 46, 60, 61, 71], prior knowledge of the search problem can be incorporated into the probabilistic models as discussed elsewhere in this book (see chapter by Baluja).

While evolutionary-algorithm practitioners have often understood that real-world or commercial applications require hybridization, there have been limited efforts in developing a principled design framework on answering critical issues such as the optimal division of labor between global and local searches (or the right mix of exploration and exploitation) [26, 79], the effect of local search on sampling [39, 40], and the optimal duration of local search [39, 45], and similar efforts are yet to be attempted for understanding and designing hybrid EDAs.

7.2.3 Time Continuation

In time continuation, capabilities of both mutation and recombination are optimally utilized to obtain a solution of as high quality as possible with a given limited computational resource [24, 47, 73, 74, 82, 83]. Time utilization (or continuation) exploits the tradeoff between the search for solutions with large population and a single convergence epoch and using a small population with multiple convergence epochs.

Early theoretical investigations indicate that when the subsolutions are of equal (or nearly equal) salience and both recombination and mutation operators have the linkage information, then a small population with multiple convergence epochs is more efficient. However, if the fitness function is noisy or has overlapping subsolutions, then a large population with single convergence epoch is more efficient [73, 74]. On the other hand, if the subsolutions of the problem are of nonuniform salience, which essentially requires serial processing, then a small population with multiple convergence epochs is more efficient [24]. While early efforts on developing adaptive continuation operators using probabilistic models of EDAs are promising [35, 47, 73], much work needs to be done to develop a principled design theory for efficiency enhancement via time continuation and to design adaptive continuation operators to reinitialize population between convergence epochs.

7.2.4 Evaluation relaxation

In evaluation relaxation, an accurate, but computationally expensive fitness evaluation is replaced with a less accurate, but computationally inexpensive fitness estimate. The low-cost, less-accurate fitness estimate can either be (1) *exogenous*, as in the case of approximate fitness functions [8, 43, 49], where external means can be used to develop the fitness estimate, or (2) *endogenous*, as in the case of *fitness inheritance* [81] where, some of the offspring fitness is estimate based on fitness of parental solutions.

Evaluation relaxation in GAs dates back to early, largely empirical work of Grefenstette and Fitzpatrick [31] in image registration [20] where significant speedups were obtained by reduced random sampling of the pixels of an image. Approximate models have since been used extensively to solve complex

optimization problems in many engineering applications such as aerospace and structural engineering [8, 11, 19].

While early evaluation-relaxation studies were largely empirical in nature, design theories have since been developed to understand the effect of approximate evaluations via surrogates on population sizing and convergence time and to optimize speedups in approximate fitness functions with known variance [51, 53], in integrated fitness functions [3, 4], in simple functions of known variance or known bias [72], and in fitness inheritance [75]. While exogenous surrogates can be readily used in EDAs, the probabilistic models of EDAs can be effectively used to develop endogenous surrogates that provide significant speedup [64, 76], details of which are provided in the next section. In addition to relaxing the solution-quality assessment measures, we can also relax the model-quality assessment in EDAs. For example, we can use *sporadic* model building, where the structure of the probabilistic model is built once every few generations and the probabilities are updated every generation [66].

7.3 Evaluation Relaxation: Designing Adaptive Endogenous Surrogates

As mentioned earlier, a distinct advantage of EDAs over first-generation GAs is the availability of variable-interaction information in terms of the probabilistic models *mined* from a population of promising solutions. Therefore, we can use the probabilistic models to infer the structural form of the surrogate. This is in contrast to surrogates often used to speedup evolutionary algorithms, which are of fixed form and do not adapt to key variable interactions of the underlying search problem. In other words, with the help of probabilistic models built in EDAs, we can use the probabilistic models to decide on the form of the surrogate and use one of the system identification, estimation, or regression methods to estimate the coefficients of the surrogate.

For example, the probabilistic model of eCGA represents nonoverlapping partitions of variables. The resulting surrogate inferred from the model would then be a polynomial, whose order and terms are decided based on the substructures identified by the model, and the coefficients of the surrogate represent the partial contribution of the subsolutions to the overall fitness of the individual [76]. The surrogates designed with the information provided by the probabilistic models are quite accurate and yield substantial speedups. For example, on a class of boundedly difficult additively decomposable problems endogenous surrogates in BOA yields speedups of about 50 [64]. This is in contrast to a moderate speed-up of about 1.3 obtained by using a simple *fitness inheritance* method [75, 81].

In the remainder of this section, we illustrate the design of adaptive endogenous surrogates in the Bayesian optimization algorithm. We note that the design method can be extended to other EDAs and the key idea is to using the probabilistic model to infer the structural form of the surrogate and to

use system-identification and estimation tools for computing the coefficients of the surrogate.

7.3.1 Evaluation Relaxation: Endogenous Surrogates in BOA

The Bayesian optimization algorithm (BOA) uses Bayesian networks to model candidate solutions [62, also see chapter by Pelikan et al]. The structure of the Bayesian network is encoded by a directed acyclic graph with the nodes corresponding to the variables in the modeled data set and the edges corresponding to conditional dependencies. A Bayesian network encodes a joint probability distribution given by

$$p(X) = \prod_{i=1}^{n} p(X_i|\Pi_i), \tag{7.1}$$

where $X = (X_0, \ldots, X_{n-1})$ is a vector of all the variables in the problem; Π_i is the set of parents of X_i (the set of nodes from which there exists an edge to X_i); and $p(X_i|\Pi_i)$ is the conditional probability of X_i given its parents Π_i.

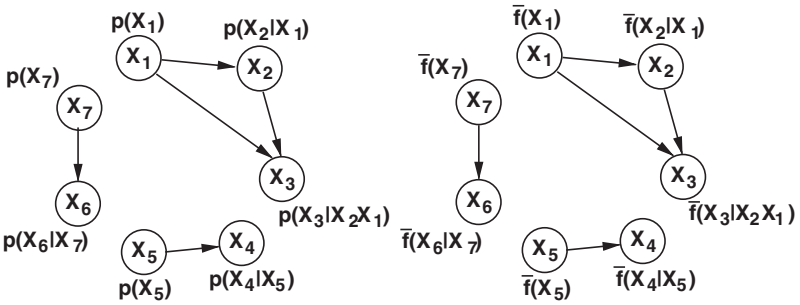
The parameters of the Bayesian networks are represented by a set of conditional probability tables (CPTs) specifying a conditional probability for each variable given any instance of the variables that the variable depends on. CPTs store conditional probabilities $p(X_i|\Pi_i)$ for each variable X_i. Local structures – in the form of decision trees or decision graphs – can also be used in place of full CPTs to enable more efficient representation of local conditional probability distributions in Bayesian networks. While we describe the design of endogenous surrogate in BOA with CPTs, similar methodology can be used for BOA with decision trees and graphs.

Given the probabilistic model (in form of a Bayesian network), we can infer the form of the surrogate as an acyclic tree whose nodes correspond to the variables and the edges correspond to the marginal fitness contributions of subsolutions (or the coefficients of the surrogate). That is, for every variable X_i and each possible value x_i of X_i, an estimate of the marginal fitness contribution of a subsolution with $X_i = x_i$ must be stored for each instance π_i of X_i's parents Π_i. In the binary case, each row in the CPT is thus extended by two additional entries. Figure 7.1 shows an example of the probability model and the substructural surrogate in BOA. The substructural fitness can be estimated as

$$f_{\text{est}}(X_1, X_2, \ldots, X_\ell) = \bar{f} + \sum_{i=1}^{\ell} \left(\bar{f}(X_i|\Pi_i) \right), \tag{7.2}$$

where $\bar{f}(X_i|\Pi_i)$ denotes the average fitness of solutions with X_i and Π_i. That is,

$$\bar{f}(X_i|\Pi_i) = \frac{1}{n_h} \sum_{\{j|y_j \supset x_i, \pi_i\}} f\left(y_j\right) - \bar{f}(\Pi_i), \tag{7.3}$$



(a) Probability model (b) Substructural surrogate

Fig. 7.1. Substructural fitness estimation model in Bayesian optimization algorithm. The estimated fitness for the model is given by $f_{\text{est}}(X_1, X_2, \ldots, X_7) = \bar{f} + \bar{f}(X_1) + \bar{f}(X_2|X_1) + \bar{f}(X_3|X_2X_1) + \bar{f}(X_5) + \bar{f}(X_4|X_5) + \bar{f}(X_7) + \bar{f}(X_6|X_7)$

where n_h is the total number of individuals that contain the schema π_i, y_j is the j^{th} individual and $f(y_j)$ is its fitness, and $\bar{f}(\Pi_i)$ is the average fitness of all solutions with Π_i.

Similar to earlier fitness-inheritance studies, we begin with fully evaluating the initial population, and thereafter evaluating an offspring with a probability $1-p_i$. In other words, we use the endogenous surrogate to estimate the fitness of an offspring with probability p_i. One question remains as to where to obtain information for computing the coefficients of the surrogate, which is addressed in Sect 7.3.2.

7.3.2 Where to Estimate the Marginal Fitnesses From?

In the proposed method, for each instance x_i of X_i and each instance π_i of X_i's parents Π_i, we must compute the average fitness of all solutions with $X_i = x_i$ and $\Pi_i = \pi_i$. In this section, we discuss two sources for computing the coefficients of the surrogate:

1. Selected parents that were evaluated using the actual fitness function
2. The offspring that were evaluated the actual fitness function

The reason for restricting computation of the coefficients of the surrogate to selected parents and offspring is that the probabilistic model used as the basis for selecting relevant statistics represents nonlinearities in the population of parents and the population of offspring. Since it is best to maximize learning data available, it seems natural to use both populations to compute the marginal fitness of the components of the surrogate. The reason for restricting input for computing these statistics to solutions that were evaluated using the actual fitness function is that the fitness of other solutions was estimated only and it involves errors that could mislead the surrogate and propagate through generations.

We have extensively tested the proposed evaluation-relaxation scheme on a class of boundedly difficult additively decomposable problems. Before presenting the key results, we now briefly introduce facetwise models to predict the scalability and speed-up of using endogenous surrogates as an alternative to expensive fitness evaluation.

7.3.3 Scalability and Speedup

Facetwise and dimensional models can be used to analyze the scalability of and the speedup provided by endogenous surrogates in EDAs. In this section, we present the key results of the analysis and the details are given elsewhere [76].

The error introduced by the surrogate can be modeled as additive Gaussian noise with zero mean and variance $p_{\mathrm{i}}\sigma^2_{\mathrm{f,t}}$, where p_{i} is the probability of an individual receiving estimated fitness, and $\sigma^2_{\mathrm{f,t}}$ is the true fitness variance. However, this approximation is not valid for very high p_{i} values as the substructural fitness is estimated from very few individuals, which increases the error in the estimate significantly. Empirically, we observed that the noise variance becomes significantly higher than the above approximation when $p_{\mathrm{i}} \geq 0.85$. Error due to variance (as in additive Gaussian noise) increases both the population size and run duration required for EDA success [25, 38, 52, 72].

The increase in the required population size due to the use of the substructural surrogate is given by

$$n_{\mathrm{r}} = \frac{n}{n_o} = (1 + p_{\mathrm{i}})\,. \tag{7.4}$$

where n_o is the minimum population size required to obtain a solution of quality $(m-1)/m$ when the endogenous surrogate is not used. Here, m is the number of key substructures of the search problem.

The increase in the run duration due to the use of the surrogate is given by

$$t_{\mathrm{c,r}} = \frac{t_{\mathrm{c}}}{t_{\mathrm{c,o}}} = \sqrt{1 + p_{\mathrm{i}}}. \tag{7.5}$$

where $t_{\mathrm{c,o}}$ is the run duration – in other words, the number of generations – taken by the EDA to successfully solve the search problem when the surrogate is not used.

Using (7.4) and (7.5) and, after further simplifications and approximations, we can estimate the increase in the total number of function evaluations required to obtain a solution with at least $m-1$ out of m substructures at their optimal values as

$$n_{\mathrm{fe,r}} \approx (1 + p_i)^{1.5}\,(1 - p_i)\,. \tag{7.6}$$

Therefore, the speedup provided by using endogenous fitness-estimation model is given by the inverse of the function-evaluation ratio:

$$\eta_{\text{endogenous fitness model}} = \frac{1}{(1+p_{\text{i}})^{1.5}\,(1-p_{\text{i}})}. \tag{7.7}$$

Equation (7.6) indicates that the number of function evaluations initially increases with p_{i}, reaching a maximum at $p_{\text{i}} = 0.2$. The function-evaluation-ratio model indicates that the number of function evaluations decreases with p_{i} for $p_{\text{i}} > 0.2$ and reaches a minimum at $p_{\text{i}} = 1$. In other words, the speedup decreases initially ($p_{\text{i}} < 0.2$) and then increases reaching a maximum at $p_{\text{i}} = 1$. However, as mentioned earlier, the facetwise models for the population sizing and the convergence time are not valid at very high values of p_{i}. Nevertheless, the models are suggestive and as shown in the results, we obtain maximum speedups when p_{i} is close to 1.0.

It should be noted that in our scalability and speedup analysis, we only considered the cost of actual fitness evaluation. In other words, we ignored the time complexity of selection, fitness model construction, generation of new candidate solutions, and fitness estimation. Combining these factors with the complexity estimate for the actual fitness evaluation can be used to compute the optimal proportion of candidate solutions whose fitnesses can be estimated using the endogenous surrogate. We reiterate that the proposed evaluation-relaxation scheme is beneficial when the actual fitness evaluation is expensive, in which case the above costs are indeed negligible and the models developed in this section valid.

7.3.4 Results and Discussion

We use two test functions for verifying and validating the use of the endogenous surrogate instead of costly, but accurate function-evaluation method. Our approach in verifying the models and observing if the proposed evaluation-relaxation scheme yields speedup is to consider bounding *adversarial problems* that exploit one or more dimensions of problem difficulty [25]. Particularly, we are interested in problems where substructure identification and exchange is critical for the EDA success. Specifically, we use OneMax – where the fitness function is the number of ones in the binary string – and $m-k$ deceptive trap problem [1, 17, 18, 22].

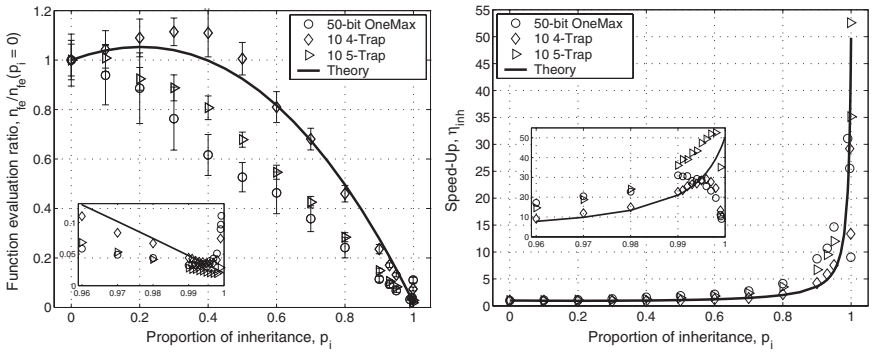
While the optimization of the OneMax problem is easy, the probabilistic models built by EDAs such as eCGA and BOA, however, are known to be only partially correct and include spurious linkages [63]. Therefore, the speed-up results on the OneMax problem will indicate if the effect of using partially correct linkage mapping on the endogenous surrogate is significant. For an ideal surrogate developed for the OneMax problem, the average fitness of a 1 in any leaf should be approximately 0.5, whereas the average fitness of a 0 in any leaf should be approximately -0.5.

Unlike, the OneMax problem, $m-k$ deceptive problems are boundedly difficult and the accurate identification and exchange of key substructures are critical to EDA success. For the $m-k$ trap problem, $\bar{f}(X_{\text{i}} = 0)$ and

$\bar{f}(X_\mathrm{i} = 1)$ depend on the state of the search because the distribution of contexts of each bit changes over time and bits in a trap are not independent. The context of each leaf also determines whether $\bar{f}(X_\mathrm{i} = 0) < \bar{f}(X_\mathrm{i} = 1)$ or $\bar{f}(X_\mathrm{i} = 0) > \bar{f}(X_\mathrm{i} = 1)$ in that particular leaf.

Figure 7.2(a) and 7.2(b) present the scalability and speedup results of the evaluation-relaxation scheme for BOA on a 50-bit OneMax, 10-4 and 10-5 deceptive trap functions. We considered a binary ($s = 2$) tournament selection without replacement. For each test problem, the following proportions of using the surrogate, p_i, were considered: 0–0.9 with step 0.1, 0.91–0.99 with step 0.01, and 0.991–0.999 with step 0.001. For each test problem and p_i value, 30 independent experiments were performed. Each experiment consisted of 10 independent runs with the minimum population size to ensure convergence to a solution within 10% of the optimum (i.e., with at least 90% correct bits) in all 10 runs. For each experiment, bisection method was used to determine the minimum population size, and the number of evaluations (excluding the evaluations done using the model of fitness) was recorded. The average of 10 runs in all experiments was then computed and displayed as a function of the proportion of candidate solutions for which fitness was estimated using the surrogate. Therefore, each point in Figs. 7.2(a) and 7.2(b) represents an average of 300 BOA runs that found a solution that is at most 10% from the optimum.

In all experiments, the number of actual fitness evaluations decreases with p_i. Furthermore, the surrogates built in BOA are applicable at high p_i values, even as high as 0.99. That is, by evaluating less than 1% of candidate solutions and estimating the fitness for the rest using the endogenous surrogate, we obtain speedup of 31 (for OneMax) to 53 (for m k-Trap). In other words, by developing and using the endogenous surrogate to estimate the fitness of



(a) Function-evaluation-ratio (b) Speedup in BOA

Fig. 7.2. The effect of using the endogenous surrogate on the total number of function evaluations required for BOA success, and the speedup obtained by using the evaluation relaxation-scheme in BOA. The empirical results are obtained for a 50-bit OneMax, 10 4-Trap and 10 5-trap problems

99% of the individuals, we can reduce the number of actual fitness evaluation required to obtain high quality solutions by a factor of up to 53.

Overall, the results suggest that significant efficiency enhancement can be achieved through an endogenous surrogate that incorporates knowledge of important subsolutions of a problem and their partial fitnesses. The results clearly indicate that using the surrogate in EDAs can reduce the number of solutions that must be evaluated using the actual fitness function by a factor of 31–53. Consequently, if fitness evaluation is a bottleneck, there is a lot of room for improvement using endogenous surrogates in EDAs in general, and BOA, in particular. For real-world problems, the actual savings may depend on the problem being considered. However, it can be expected that developing and using the fitness-estimate model enables significant reduction in the number of fitness evaluations on many problems because deceptive problems of bounded difficulty bound a large class of important nearly decomposable problems.

The probabilistic models are not only useful for the design of surrogates, but can be exploited in other facets of efficiency enhancement as well. In the following section, we illustrate the use of probabilistic models in the principled design of time continuation operators.

7.4 Time Continuation: Mutation in EDAs

In time continuation, we investigate and decide between the fundamental tradeoff between using an evolutionary algorithm with a large population for a single convergence epoch or with a small population for multiple convergence epochs, as illustrated in Fig. 7.3. A *continuation operator* is required when using a small-population, multiple-epoch evolutionary algorithm for maintain diversity in population between *epochs*. In the ideal case, the continuation operator perturbs only bad building blocks (BBs) at the end of each convergence epoch. However, in practice, the continuation operator not only perturbs bad building blocks, but also some good ones, and a regeneration cost – or cost of reduction in solution quality between two epochs – is incurred. Since the continuation operator modifies only a few individuals to seed the population in subsequent epochs, it is assumed to be some form of mutation or local search method. Therefore, the decision making involved in time continuation can also be posed as choosing between two key genetic operators – recombination and mutation.

Goldberg [24] developed an analytical framework to optimally solve the *time continuation* problem. Early studies considered the effect of the salience structure and observed that while a large population run is preferable for problems with near-uniform salience structure, a small population run is advantageous for problems with exponential-salience structure [24, 82, 83]. More recent studies considered the effectiveness of incorporating the building-block structure into both global and local evolutionary algorithm operators and the effect of noise, salience structure and the crosstalk between different building

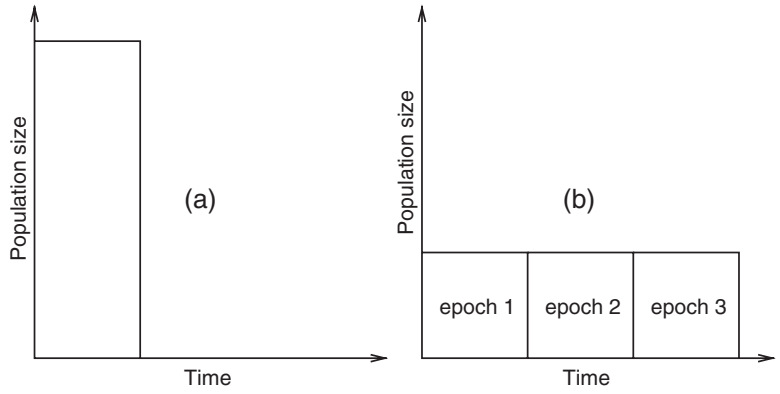


Fig. 7.3. Two scenarios of resource utilization: **(a)** Large population, single convergence epoch, and **(b)** Small population, multiple convergence epochs

blocks on time continuation [74, 77]. When both global and local operators are given the substructural information (or good neighborhood information), a small population run is beneficial for deterministic problems with near-uniform salience structure. On the other hand, for noisy problems, a large population is preferred.

One of the key challenges in time continuation is the design of effective continuation operators that searches in the correct neighborhoods. Existing mutation (or continuation) operators usually search in the local neighborhood of an individual, without taking into account the *global* neighborhood information. In genetic algorithms, mutation is usually a secondary search operator which performs random walk *locally* around a solution. On the other hand, in evolution strategies, while powerful mutation operators are used [6, 10, 36, 69, 78], the neighborhood information is still *local* around a single or few solutions. In local-search literature, while the importance of using a good neighborhood operator is often highlighted [5, 7, 15, 85, 86], no systematic methods for designing neighborhood operators that can solve a broad class of bounding problems have been developed.

However, for solving boundedly difficult problems, local neighborhood information is not sufficient, and a mutation operator which uses local neighborhoods requires $\mathcal{O}(m^k \log m)$ number of evaluations [55]. Therefore, we utilize the probabilistic models built in eCGA for automatically building *global* neighborhood (or linkage) information into the mutation operator. Unlike, adaptive mutation techniques in evolution strategies, which usually have local neighborhood information adapted over time, our method leads to a more global induction of the neighborhood.

In Sect. 7.4.1, we illustrate the design of an efficient operator that utilizes the *global* neighborhood information mined by the probabilistic models of the

extended compact genetic algorithm (eCGA) [37, also see chapter by Harik et al] to search among competing subsolutions.

7.4.1 Mutation in eCGA: Searching in Substructural Neighborhood

As described in the chapter by Harik et al, eCGA builds marginal product models that yields a direct mapping of linkage groups among successful individuals. Therefore, we use the model-building procedure of eCGA to identify the key substructures of a problem. Once the linkage-groups are identified, we use an *enumerative building-block-wise mutation* operator [74] to search for the best among competing subsolutions. For example, if the model builder identifies m BBs of size k each, the eCGA continuation operator will select the best subsolution out of 2^k possible ones in each of the m partition.

That is, from a sample of randomly generated candidate solution the top solutions (as determined by the selection mechanism) are used to build probabilistic model in eCGA. The best solution in the population is used for substructural mutation: Consider the first nonmutated substructure, where the substructures are arbitrarily chosen from left-to-right, however, different schemes can be – or may required to be – chosen to decide the order of choosing substructures to mutate. For example, substructural partitions that contain most *active* variables might be mutated before those that contain less active variables. For the substructure in consideration, create $2^k - 1$ unique individuals with all possible subsolutions in the chosen partition, where k is order of the substructure. The subsolutions in other partitions are not modified. Evaluate all $2^k - 1$ individuals and retain the best for mutation of other substructures. Thus at each convergence epoch the best subsolution in each partition is chosen and the search ends after m convergence epochs, where m is the number of substructures in the problem.

Note that the performance of the above mutation can be slightly improved by using a greedy heuristic to search for the best among competing BBs, however, as shown later, the scalability of the mutation operator is determined by the population-size required to accurately identify the building blocks. Furthermore, we perform linkage identification only once in the initial generation. This offline linkage identification works well on problems with BBs of nearly equal salience. However, for problems with BBs of nonuniform salience, we would have to perform linkage identification and update BB information in regular intervals. The key idea in designing the mutation operator in other EDAs such as BOA is that the operator should effectively use the neighborhood information contained in the probabilistic models.

We now present the scalability of the enumerative BB-wise mutation operator and followed by an analysis of the efficiency enhancement provided by time continuation in eCGA.

7.4.2 Scalability of Building-Block-Wise Mutation

The scalability of the BB-wise mutation operator depends on two factors (1) the population size required to build accurate probabilistic models of the linkage groups, and (2) the total number of evaluations required by the BB-wise mutation operator to find optimal subsolutions in all the partitions. Pelikan, Sastry, and Goldberg [65] developed facetwise models for predicting the critical and maximum population-size required to correctly identify good interactions among variables. They showed that the minimum population size scales as

$$n_{\min} = \Theta\left(2^k m^{1.05}\right), \tag{7.8}$$

and the maximum population size which avoids discovery of false dependencies between independent variables is given by

$$n_{\max} = \Theta\left(2^k m^{2.1}\right). \tag{7.9}$$

In other words, to avoid incorrect identification of BBs, the population size should be less than $n_{\max}$. Since we require that *all* the BBs be correctly identified in the first generation itself, the population size required should be greater than $n_{\min}$, but less than $n_{\max}$. That is,

$$\Theta\left(2^k m^{1.05}\right) \leq n \leq \Theta\left(2^k m^{2.1}\right). \tag{7.10}$$

Since the model building is performed only once, the total number of function evaluations scales as the population size. That is,

$$\Theta\left(2^k m^{1.05}\right) \leq n_{\text{fe},1} \leq \Theta\left(2^k m^{2.1}\right). \tag{7.11}$$

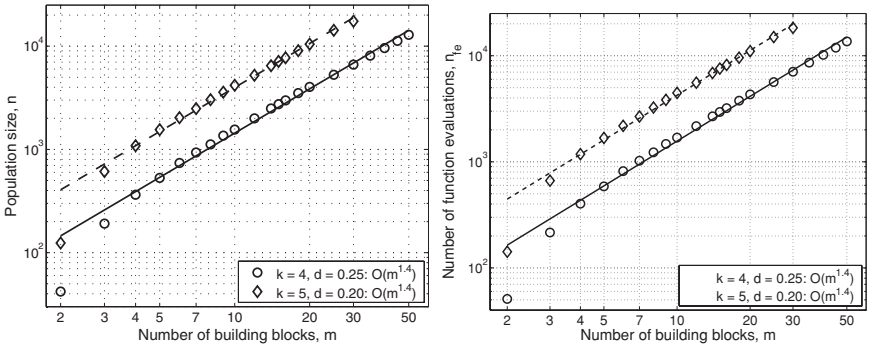
During BB-wise mutation, we evaluate $2^k - 1$ individuals for determining the best BBs in each of the m partitions. Therefore, the total number of function evaluations used during BB-wise mutation is

$$n_{\text{fe},2} = \left(2^k - 1\right) m = \mathcal{O}\left(2^k m\right). \tag{7.12}$$

From Equations 7.11 and 7.12, the total number of function evaluations scales as

$$\Theta\left(2^k m^{1.05}\right) \leq n_{\text{fe}} \leq \Theta\left(2^k m^{2.1}\right). \tag{7.13}$$

We now empirically verify the scale-up of the population size and the number of function evaluations required for successfully solving the $m-k$ deceptive trap problem in Figs. 7.4(a) and 7.4(b), respectively. For the empirical runs, we use tournament selection without replacement with a tournament size of 8. The average number of subsolutions correctly converged are computed over 30 independent runs. The minimum population size required such that $m-1$ subsolutions converge to the correct value is determined by a bisection method [72]. The results of population-size is averaged over 30 such bisection



(a) Population size (b) Number of function evaluations

Fig. 7.4. Population size (7.10) and the number of function evaluations (7.13) required by BB-wise mutation for solving $m - k$ Trap function. The results are averaged over 900 runs for the number of function evaluations and 30 bisection runs for the population size. The relative deviation for the empirical results is less than 0.2%. The population size and the number of function evaluations both scale as $\Theta(2^k m^{1.5})$

runs, while the results for the function-evaluation ratio is averaged over 900 independent runs.

In contrast to fixed mutation operators which require $\mathcal{O}(m^k \log m)$ number of function evaluations to solve additively separable GA-hard problems [55], the proposed eCGA-based BB-wise mutation operator that automatically identifies the linkage groups requires only $\mathcal{O}(2^k m^{1.5})$ (polynomial) number of evaluations.

7.4.3 Crossover vs. Mutation in eCGA

We know that eCGA scales as $\Theta\left(2^k \sqrt{k} m^{1.5} \log m\right)$ [73, also see chapter by Harik et al], and from (7.13), we know that the BB-wise mutation scales as $\Theta\left(2^k m^{1.5}\right)$ for additively separable problem of bounded difficulty. Therefore, the BB-wise mutation operator in eCGA is $\Theta\left(\sqrt{k} \log m\right)$ faster than eCGA in solving boundedly difficult additively separable problems. That is, the speedup – which is defined as the ratio of number of function evaluations required by eCGA to that required by the selectomutative GA – is given by

$$\eta = \frac{n_{\text{fe}}(\text{eCGA})}{n_{\text{fe}}(\text{BBwise Mutation})} = \Theta\left(\sqrt{k} \log m\right), \tag{7.14}$$

which is empirically verified in Fig. 7.5. The results clearly indicate the efficiency enhancement provided by the time continuation operator that automatically and adaptively searches for the subsolution neighborhood as identified by eCGA.

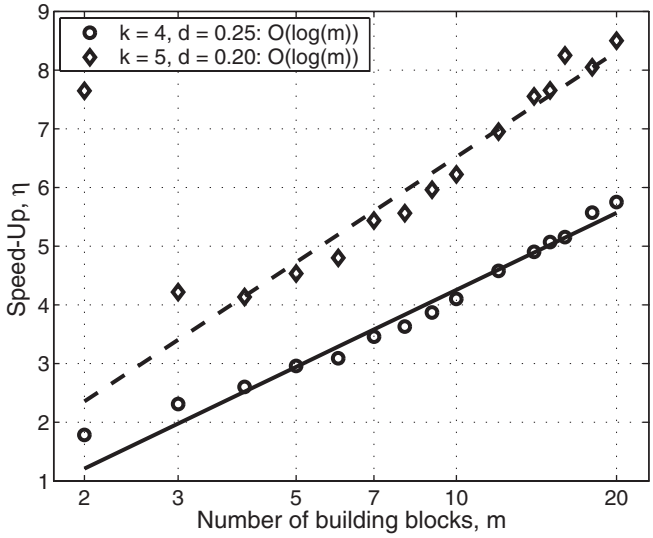


Fig. 7.5. Empirical verification of the speedup (7.14) obtained by using the probabilistic model building BB-wise mutation over eCGA for the $m - k$ Trap function. The results show that the speedup scales as $\Theta(\sqrt{k} \log m)$

Time-continuation scenarios have also been studied when dealing with noisy problems and problems with overlapping building blocks, where a competent crossover is often more efficient than a competent mutation, and therefore a large-population, single convergence epoch eCGA run is preferred [74, 77]. One of the important efforts directly motivated by this study, which is currently underway, is the design and development of *adaptive time continuation operators* that utilize the substructural models built by eCGA not only in mutation and recombination operators, but also to automatically decide between using a large population with single convergence epoch or a small population eCGA with multiple convergence epochs [47]. Simply stated, the model building is used to identify the appropriate population size regime and whether local or global operators are used. For example, for noisy problems an adaptive time continuation operator should implicitly switch from local to global search operator as the problem becomes more noisy. The decision making depends upon the type of the problem being solved, and results in significant savings even for modestly sized problems.

7.5 Summary and Conclusions

Like any industrial-strength search algorithm, practical deployment of EDAs strongly rely on one or more efficiency-enhancement techniques such as parallelization, hybridization, time continuation, and evaluation relaxation. While EDAs take problems that were intractable by first generation genetic

algorithms, and render them tractable, principled efficiency-enhancement techniques take us from tractability to practicality. In this chapter, we presented an overview of various efficiency-enhancement techniques for speeding-up EDAs. We also provided two examples of principled efficiency-enhancement techniques, both of which utilize the probabilistic models built by the EDAs. The first example was an evaluation-relaxation method, where we build an endogenous substructural surrogate to estimate fitness of majority of the population, while actual fitness is computed for only a small portion of the population. The second example developed a competent mutation (or time continuation) operator in the extended compact genetic algorithm, which uses the probabilistic models and searches locally in the subsolution neighborhood.

The two examples clearly demonstrate that by systematically incorporating problem knowledge gained through the probabilistic models built in EDAs into the efficiency-enhancement technique, the speedup can be significantly enhanced. Furthermore, the overall efficiency of combining such nearly independent efficiency-enhancement techniques is multiplicative. For example, if we use a parallel EDA that yields linear speedup with 100 processors, and each of the other three EETs makes EDAs 25% more efficient, then together they yield a speedup of $100 * 1.25^3 = 195.3$. That is, evaluation relaxation, time continuation, and hybridization would give slightly more than 95 processors' worth of additional computation power.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-03-1-0129, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center, and the Research Award and the Research Board at the University of Missouri. 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] Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hill Climbing.* Kluwer, Boston, MA

[2] Albert, L. A. (2001). Efficient genetic algorithms using discretization scheduling. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL

[3] Albert, L. A. and Goldberg, D. E. (2001). Efficient evaluation relaxation under integrated fitness functions. *Intelligent Engineering Systems*

Through Artificial Neural Networks, 11:165–170. (Also IlliGAL Report No. 2001024)

[4] Albert, L. A. and Goldberg, D. E. (2002). Efficient discretization scheduling in multiple dimensions. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 271–278. (Also IlliGAL Report No. 2002006)

[5] Armstrong, D. E. and Jacobson, S. H. (2005). Data independent neighborhood functions and strict local optima. *Discrete Applied Mathematics*, 146(3):233–243

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

[7] Barnes, J. W., Dimova, B., and Dokov, S. P. (2003). The theory of elementary landscapes. *Applied Mathematical Letters*, 16:337–343

[8] Barthelemy, J.-F. M. and Haftka, R. T. (1993). Approximation concepts for optimum structural design—a review. *Structural Optimization*, 5:129–144

[9] Bethke, A. D. (1976). Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Efficiency of use of processing capacity. Tech. Rep. No. 197, Univeristy of Michigan, Logic of Computers Group, Ann Arbor, MI

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

[11] Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1998). A rigorous framework for optimization of expensive functions by surrogates. Technical report, National Aeronautics and Space Administration (NASA), Hampton, VA. ICASE Report No. 98-47

[12] Bosworth, J. L., Foo, N., and Zeigler, B. P. (1972). Comparison of genetic algorithms with conjugate gradient methods. ORA Tech. Report No. 00312-1-T, University of Michigan, Department of Computer and Communication Sciences, Ann Arbor, MI

[13] Cantú-Paz, E. (1997). A summary of research on parallel genetic algorithms. IlliGAL Report No. 97003, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL

[14] Cantú-Paz, E. (2000). *Efficient and accurate parallel genetic algorithms*. Kluwer, Boston, MA

[15] Colletti, B. W. and Barnes, J. W. (2004). Using group theory to construct and characterize metaheuristic search neighborhoods. In Rego, C. and Alidaee, B., editors, *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, pages 303–329. Kluwer , Boston, MA

[16] Davis, L., editor (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, NY

[17] Deb, K. and Goldberg, D. E. (1992). Analyzing deception in trap functions. *Foundations of Genetic Algorithms*, 2:93–108. (Also IlliGAL Report No. 91009)

[18] Deb, K. and Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408. (Also IlliGAL Report No. 92001)

[19] Dennis, J. E. and Torczon, V. (1997). Managing approximation models in optimization. In Alexandrov, N. M. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 330–347, Philadelphia, PA. SIAM

[20] Fitzpatrick, J. M., Grefenstette, J. J., and Van Gucht, D. (1984). Image registration by genetic search. In *Proceedings of IEEE Southeast Conference*, pages 460–464, Piscataway, NJ. IEEE press

[21] Fleurent, C. and Ferland, J. (1994). Genetic hybrids for the quadratic assignment problem. In *DIMACS Series in Mathematics and Theoretical Computer Science*, volume 16, pages 190–206

[22] Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In Davis, L., editor, *Genetic algorithms and simulated annealing*, chapter 6, pages 74–88. Morgan Kaufmann, Los Altos, CA

[23] Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading, MA

[24] Goldberg, D. E. (1999). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 212–219. (Also IlliGAL Report No. 99002)

[25] Goldberg, D. E. (2002). *Design of innovation: Lessons from and for competent genetic algorithms*. Kluwer, Boston, MA

[26] Goldberg, D. E. and Voessner, S. (1999). Optimizing global-local search hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 220–228. (Also IlliGAL Report No. 99001)

[27] Gorges-Schleuter, M. (1989). ASPARAGOS: A population genetics approach to genetic algorithms. *Evolution and Optimization '89*, pages 86–94

[28] Gorges-Schleuter, M. (1989). ASPARAGOS: An asynchronous parallel genetic optimization strategy. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–428

[29] Gorges-Schleuter, M. (1997). ASPARAGOS96 and the traveling salesman problem. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 171–174

[30] Grefenstette, J. J. (1981). Parallel adaptive algorithms for function optimization. Tech. Rep. No. CS-81-19, Vanderbilt Univeristy, Computer Science Department, Nashville, TN

[31] Grefenstette, J. J. and Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 112–120

[32] Grosso, P. B. (1985). *Computer simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model*. PhD thesis, University of Michigan, Ann Arbor, MI. (University microfilms no. 8520908)

[33] Gruau, F. (1994). *Neural network synthesis using cellular encoding and the genetic algorithm.* PhD thesis, L'Universite Claude Bernard-Lyon I

[34] Handa, H. (2003). Hybridization of estimation of distribution algorithms with a repair method for solving constraint satisfaction problems. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 991–1002

[35] Handa, H. (2005). The effectiveness of mutation operation in the case of estimation of distribution algorithms. *Journal of Biosystems.* (To appear)

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

[37] Harik, G. (1999). Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Urbana, IL

[38] Harik, G., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253. (Also IlliGAL Report No. 96004)

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

[40] Hart, W. E. and Belew, R. K. (1996). Optimization with genetic algorithm hybrids using local search. In Belew, R. K. and Mitchell, M., editors, *Adaptive Individuals in Evolving Populations*, pages 483–494. Addison-Wesley, Reading, MA

[41] Hartmann, A. K. and Rieger, H. (2001). *Optimization algorithms in physics*, chapter 9, pages 192–203. Wiley-VCH, Berlin

[42] Ibaraki, T. (1997). Combinations with other optimization methods. In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, pages D3:1–D3:2. Institute of Physics Publishing and Oxford University Press, Bristol and New York

[43] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1):3–12

[44] Krasnogor, N. (2002). *Studies on the Theory and Design Space of Memetic Algorithms.* PhD thesis, University of the West of England, Bristol, England

[45] Land, M. (1998). *Evolutionary algorithms with local search for combinatorial optimization.* PhD thesis, University of California at San Diego, San Diego, CA

[46] Lima, C. F. and Lobo, F. G. (2004). Parameter-less optimization with the extended compact genetic algorithm and iterated local search. *Proceedings of the Genetic and Evolutionary Computation Conference*, 1:1328–1339. (Also technical report cs.NE/0402047 at arXiv.org)

[47] Lima, C. F., Sastry, K., Goldberg, D. E., and Lobo, F. G. (2005). Combining competent crossover and mutation operators: A probabilistic model

building approach. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 735–742. (Also IlliGAL Report No. 2005002)

[48] Lin, S.-C., Goodman, E. D., and Punch, W. F. (1997). Investigating parallel genetic algorithms on job shop scheduling problem. *Sixth International Conference on Evolutionary Programming*, pages 383–393

[49] Llorà, X., Sastry, K., Goldberg, D. E., Gupta, A., and Lakshmi, L. (2005). Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1363–1370. (Also IlliGAL Report No. 2005009)

[50] Manderick, B. and Spiessens, P. (1989). Fine-grained parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 428–433

[51] Miller, B. L. (1997). *Noise, Sampling, and Efficient Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL. (Also IlliGAL Report No. 97001)

[52] Miller, B. L. and Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212. (Also IlliGAL Report No. 95006)

[53] Miller, B. L. and Goldberg, D. E. (1996). Optimal sampling for genetic algorithms. *Intelligent Engineering Systems through Artificial Neural Networks*, 6:291–297

[54] Moscato, P. (1989). 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

[55] Mühlenbein, H. (1992). How genetic algorithms really work: Mutation and hillclimbing. *Parallel Problem Solving from Nature II*, pages 15–26

[56] Munetomo, M., Murao, N., and Akama, K. (2005). Empirical studies on parallel network construction of Bayesian optimization algorithms. *Proceedings of the Congress of Evolutionary Computation*, 1-2-3: 1234–1238

[57] Ocenasek, J. and Pelikan, M. (2003). Parallel spin glass solving in hierarchical Bayeisan optimization algorithm. *Proceedings of the 9th International Conference on Soft Computing, Mendel 2003*, pages 120–125

[58] Ocenasek, J., Schwarz, J., and Pelikan, M. (2003). Design of multithreaded estimation of distribution algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1247–1258

[59] Orvosh, D. and Davis, L. (1993). Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints. *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 650

[60] Pelikan, M. (2005). *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithm*. Springer, Berlin Heidelberg New York

[61] Pelikan, M. and Goldberg, D. E. (2003). Hierarchical BOA solves Ising spin glasses and MAXSAT. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1271–1282. (Also IlliGAL Report No. 2003001)

[62] Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (2000). Linkage learning, estimation distribution, and Bayesian networks. *Evolutionary Computation*, 8(3):314–341. (Also IlliGAL Report No. 98013)

[63] Pelikan, M., Goldberg, D. E., and Sastry, K. (2001). Bayesian optimization algorithm, decision graphs, and Occam's razor. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 519–526. (Also IlliGAL Report No. 2000020)

[64] Pelikan, M. and Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:48–59. (Also IlliGAL Report No. 2004009)

[65] Pelikan, M., Sastry, K., and Goldberg, D. E. (2003). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258. (Also IlliGAL Report No. 2001029)

[66] Pelikan, M., Sastry, K., and Goldberg, D. E. (2005). Sporadic model building for efficiency enhancement of hBOA. IlliGAL Report No. 2005026, University of Illinois at Urbana-Champaign, Urbana, IL

[67] Pettey, C. C., Leuze, M. R., and Grefenstette, J. J. (1987). A parallel genetic algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 155–161

[68] Ramsey, C. L. and Grefenstette, J. J. (1993). Case-Based initialization of genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 84–91

[69] Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog, Stuttgart

[70] Robertson, G. G. (1987). Parallel implementation of genetic algorithms in a classifier system. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 140–147

[71] Sastry, K. (2001a). Efficient cluster optimization using a hybrid extended compact genetic algorithm with a seeded population. IlliGAL Report No. 2001018, University of Illinois at Urbana-Champaign, Urbana, IL

[72] Sastry, K. (2001b). Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL. (Also IlliGAL Report No. 2002004)

[73] Sastry, K. and Goldberg, D. E. (2004a). Designing competent mutation operators via probabilistic model building of neighborhoods. *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, 2:114–125. Also IlliGAL Report No. 2004006

[74] Sastry, K. and Goldberg, D. E. (2004b). Let's get ready to rumble: Crossover versus mutation head to head. *Proceedings of the 2004 Genetic*

and Evolutionary Computation Conference, 2:126–137. Also IlliGAL Report No. 2004005

[75] Sastry, K., Goldberg, D. E., and Pelikan, M. (2001). Don't evaluate, inherit. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 551–558. (Also IlliGAL Report No. 2001013)

[76] Sastry, K., Pelikan, M., and Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms building-block-wise fitness estimation. *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 720–727

[77] Sastry, K., Winward, P., Goldberg, D. E., and Lima, C. F. (2005). Fluctuating crosstalk as a source of deterministic noise and its effects on ga scalability. IlliGAL Report No. 2005025, University of Illinois at Urbana-Champaign, Urbana, IL

[78] Schwefel, H.-P. (1977). Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisciplinary Systems Research*, 26

[79] Sinha, A. (2003a). Designing efficient genetic and evolutionary hybrids. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL. Also IlliGAL Report No. 2003020

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

[81] Smith, R., Dike, B., and Stegmann, S. (1995). Fitness inheritance in genetic algorithms. In *Proceedings of the ACM Symposium on Applied Computing*, pages 345–350, New York, NY, USA. ACM

[82] Srivastava, R. (2002). Time continutation in genetic algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL

[83] Srivastava, R. and Goldberg, D. E. (2001). Verification of the theory of genetic and evolutionary continuation. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 551–558. (Also IlliGAL Report No. 2001007)

[84] Tanese, R. (1989). *Distributed genetic algorithms for function optimization*. PhD thesis, University of Michigan, Ann Arbor, MI. (University microfilms no. 8520908)

[85] Vaughan, D., Jacobson, S., and Armstrong, D. (2000). A new neighborhood function for discrete manufacturing process design optimization using generalized hill climbing algorithms. *ASME Journal of Mechanical Design*, 122(2):164–171

[86] Watson, J.-P. (2003). *Empirical Modeling and Analysis of Local Search Algorithms For The Job-Shop Scheduling Problem*. PhD thesis, Colorado State University, Fort Collins, CO