Sub-Structural Niching in Estimation of Distribution Algorithms

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

Hussein A. Abbass
Artificial Life and Adaptive Robotics Laboratory
School of Information Tech. and Electrical Eng.
University of New South Wales
Australian Defense Force Academy
h.abbass@adfa.edu.au

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

D.D. Johnson
Department of Materials Sciences and Eng.
University of Illinois at Urbana-Champaign
Urbana, IL 61801
duanej@uiuc.edu

ABSTRACT

We propose a sub-structural niching method that fully exploits the problem decomposition capability of linkage-learning methods such as the estimation distribution algorithms and concentrate on maintaining diversity at the sub-structural level. The proposed method consists of three key components: (1) Problem decomposition and sub-structure identification, (2) sub-structure fitness estimation, and (3) sub-structural niche preservation. The sub-structural niching method is compared to restricted tournament selection (RTS)—a niching method used in hierarchical Bayesian optimization algorithm—with special emphasis on sustained preservation of multiple global solutions of a class of boundedly-difficult, additively-separable multimodal problems. The results show that sub-structural niching successfully maintains multiple global optima over large number of generations and does so with significantly less population than RTS. Additionally, the market share of each of the niche is much closer to the expected level in sub-structural niching when compared to RTS.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Genetic algorithms, estimation of distribution algorithms, multimodal, multiobjective, niching, probabilistic models, fitness estimation, fitness approximation, building blocks, eCGA

1. INTRODUCTION

One of the daunting challenges in the field of genetic and evolutionary computation is the systematic and principled design of scalable genetic algorithms (GAs) and significant progress has been made along these lines. A design decomposition theory has been proposed and several *competent* GAs—GAs that solve hard problems quickly, reliably, and accurately—have been developed [8]. One such class of competent GAs is the estimation distribution algorithms (EDAs) [29, 21]. EDAs replace the traditional variation operators of GAs with probabilistic model building of promising solutions that identifies key sub-structures (or building blocks) of the underlying search problem, and sampling the model to generate new candidate solutions. EDAs have successfully solved problems of bounded difficulty at a single level or at multiple hierarchical levels requiring only polynomial (oftentimes sub-quadratic) number of function evaluations [28, 27, 31].

One of the important components required by EDAs for successfully solving multimodal, hierarchical, dynamic, and multiobjective optimization problems is an efficient niching method. The niching mechanism has to stably maintain a diverse population throughout the search, thereby allowing EDAs to (1) identify multiple optima reliably when solving multimodal and multiobjective problems, (2) identify the global optimum by deciding successfully between sub-structures when all the hierarchical interactions are revealed, and (3) rapidly identify global solutions as and when changes occur in non-stationary problems. Such a niching method not only needs to adaptively identify and conform to all the niches and niche-distance distributions, but also need to maintain them effectively over the duration of the search.

Traditional niching methods usually maintain diversity at the level of individuals and are not often adaptive to the niche size and distribution. Additionally, they also do not exploit the underlying working mechanism of EDAs and other linkage-learning algorithms. That is, the traditional nichers often use distance information based on the entire individual and do not often directly respect or exploit problem decomposition. Therefore, in this paper we propose a niching method that respects problem decomposition, and utilizes the sub-structure identification capability of EDAs, and maintains diversity at sub-structure level in a stable manner. Such a niching method is not only advantageous for maintaining multiple niches, but also effective for hierarchical [27], and dynamic [3, 1] problem optimizations where sub-structure niche preservation is what actually required.

The proposed method consists of three components: (1) Sub-structure identification, where we use the probabilistic model built by EDAs, specifically, extended compact GA [16], (2) sub-structure fitness estimation, where we use the fitness-estimation procedure proposed by Sastry *et al* [34], and (3) sub-structure niche preservation, where different mechanisms can be envisioned, and suitability of each is based on the purpose and objective of niching. The key idea of the sub-structure niche preservation mechanism is to preserve highly-fit sub-structures in desired proportions in the population in a stable manner over the duration of the search.

The sub-structural niching mechanism is compared with restricted tournament selection [17]—a nicher used in hierarchical Bayesian optimization algorithm (hBOA)—on a class of boundedly-difficult additively-separable multimodal problems. Specifically, we compare (1) the stability of maintaining multiple niches over a large number of generations, (2) the capability of allocating market share to different niches at the desired level, and (3) the population size required to consistently maintain all the global optima.

The paper is organized as follows. The next section provides a brief introduction to the extended compact genetic algorithm, followed by a detailed description of the proposed sub-structural niching mechanism. The performance of the proposed method is compared to that of RTS in section 4, followed by key conclusions of the paper.

2. EXTENDED COMPACT GENETIC ALGORITHM

Extended compact genetic algorithm (eCGA) [16] is an EDA that replaces traditional variation operators of genetic and evolutionary algorithms by building a probabilistic model of promising solutions and sampling the model to generate new candidate solutions. The typical steps of eCGA can be outlined as follows:

1. *Initialization:* The population is usually initialized with random individuals. However, other initialization procedures can also be used in a straightforward manner.

2. *Evaluation:* The fitness or the quality-measure of the individuals are computed.

3. *Selection:* Like traditional genetic algorithms, EDAs are selectionist schemes, because only a subset of better individuals is permitted to influence the subsequent generation of candidate solutions. Different selection schemes used elsewhere in genetic and evolutionary algorithms—tournament selection, truncation selection, proportionate selection, etc.—may be adopted for this purpose, but a key idea is that a "survival-of-the-fittest" mechanism is used to *bias* the generation of new individuals.

4. *Probabilistic model estimation:* Unlike traditional GAs, however, EDAs assume a particular probabilistic model of the data, or a *class* of allowable models. A *class-selection metric* and a *class-search mechanism* is used to search for an optimum probabilistic model that represents the selected individuals.

   Model representation: The probability distribution used in eCGA is a class of probability models known as marginal product models (MPMs). MPMs partition genes into mutually independent groups and specifies marginal probabilities for each linkage group. For example, the following MPM, `[1,3][2][4]`, for a four-bit problem represents that the 1st and 3rd genes are linked and 2nd and 4th genes are independent. An MPM must also specify probabilities for each sub-structure. For the above example, the MPM consists of the marginal probabilities: $\{p(x_1 = 0, x_3 = 0)$, $p(x_1 = 0, x_3 = 1)$, $p(x_1 = 1, x_3 = 0)$, $p(x_1 = 1, x_3 = 1)$, $p(x_2 = 0)$, $p(x_2 = 1)$, $p(x_4 = 0)$, $p(x_4 = 1)\}$, where x_i is the value of the ith gene.

   Class-Selection metric: To distinguish between better model instances from worse ones, eCGA uses a minimum description length (MDL) metric [32]. The key concept behind MDL models is that all things being equal, simpler models are better than more complex ones. The MDL metric used in eCGA is a sum of two components:

   - **Model complexity** which quantifies the model representation size in terms of number of bits required to store all the marginal probabilities:

     $$C_m = \log_2(n) \sum_{i=1}^{m} \left(2^{k_i} - 1\right). \qquad (1)$$

     where n is the population size, m is the number of linkage groups, k_i is the size of the i^{th} group.

   - **Compressed population complexity**, which quantifies the data compression in terms of the entropy of the marginal distribution over all partitions.

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

     where p_{ij} is the frequency of the j^{th} gene sequence of the genes belonging to the i^{th} partition.

Class-Search method: In eCGA, both the structure and the parameters of the model are searched and optimized to best fit the data. While the probabilities are learnt based on the variable instantiations in the population of selected individuals, a greedy-search heuristic is used to find an optimal or near-optimal probabilistic model. The search method starts by treating each decision variable as independent. The probabilistic model in this case is a vector of probabilities, representing the proportion of individuals among the selected individuals having a value '1' (or alternatively '0') for each variable. The model-search method continues by merging two partitions that yields greatest improvement in the model-metric score. The subset merges are continued until no more improvement in the metric value is possible.

5. *Offspring creation:* In eCGA, new individuals are created by sampling the probabilistic model. The offspring population are generated by randomly generating subsets from the current individuals according to the probabilities of the subsets as calculated in the probabilistic model.

6. *Replacement:* Many replacement schemes generally used in genetic and evolutionary computation—generational replacement, elitist replacement, niching, etc.—can be used in EDAs, but the key idea is to replace some or all the parents with some or all the offspring.

7. Repeat steps 2–6 until one or more termination criteria are met.

3. SUB-STRUCTURAL NICHING

Traditional niching methods [4, 5, 14, 22, 35, 17, 24, 19, 25] achieve speciation by maintaining diversity at the level of individuals. Effectiveness of such methods are strongly dependent on the niche distribution. While some methods exist that can automatically adjust the niche radius [15], they still detect diversity on the individual level.

One of the key elements of Goldberg's design decomposition [10, 9]—which has been influential in the design and development of many competent GAs—suggests that one of the critical steps for GA success is problem decomposition, and identification and mixing of building blocks. Since the EDAs work by first decomposing the search problems into sub-structures and then creating new solutions by exchanging different sub-structures, it might be advantageous, sometimes even necessary, to maintain diversity at the building-block (sub-structural) level and not at individual level [33]. This is especially the case for dynamic optimization, hierarchical-problem optimization, and multi-objective optimization.

Sub-structural niching requires three key elements:

Sub-structure identification: To maintain diversity at the sub-structural level, we first need a mechanism to automatically identify all the important building blocks of the underlying search problem. In this study, we use the probabilistic models built by the eCGA. However, other linkage identification techniques [13, 12, 20, 26, 36, 18, 29, 21] can be used in a straightforward manner.

Sub-structure fitness estimation: Once key sub-structures are identified, we must decide on which sub-structures to preserve and in what proportion. While sometimes, we might require to retain all sub-structure alternatives, usually we only need to preserve the highly fit ones. In order to do so, we need a way to estimate the quality of substructures from the fitnesses of individuals that possess them.

In this study, we use the fitness-estimation method proposed by Sastry, Pelikan and Goldberg [34]. That is, after the probabilistic model is built and the linkage map is obtained, we estimate the fitness of sub-structures. In all, we estimate the fitness of a total of $\sum_{i=1}^{m} 2^{k_i}$ schemas, where k_i is the order of i^{th} schema. For example, for a four-bit problem, whose model is `[1,3][2][4]`, the schemas whose fitnesses are estimated are: {`0*0*`, `0*1*`, `1*0*`, `1*1*`, `*0**`, `*1**`, `***0`, `***1`}.

The fitness of a sub-structure, h, is defined as the difference between the average fitness of individuals that contain the schema and the average fitness of all the individuals. That is,

$$\hat{f}_s(h) = \frac{1}{n_h} \sum_{\{i|x_i \supset h\}} f(x_i) - \frac{1}{n'} \sum_{i=1}^{n'} f(x_i), \tag{3}$$

where n_h is the total number of individuals that contain the schema h, x_i is the i^{th} individual and $f(x_i)$ is its fitness, n' is the total number of individuals that were evaluated. If a particular schema is not present in the population, its fitness is arbitrarily set to zero. Furthermore, it should be noted that the above definition of schema fitness is not unique and other estimates can be used. The key point however is the use of the probabilistic model in determining the schema fitnesses. Further details regarding the estimation method are given elsewhere [34, 30].

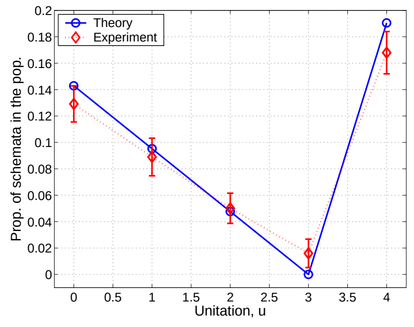
Sub-structure niche preservation: Having identified the sub-structures and estimated their quality is not enough, we still need to decide on a methodology for preserving the sub-structures. Different methods such as fitness-proportionate, ranking, and truncation can be used and no one method is better than the other. For example, we can opt to preserve the sub-structures in proportion to their estimated fitness (so called fitness-proportionate method). That is, we have to modify the sampling frequencies of each sub-structure, h_j:

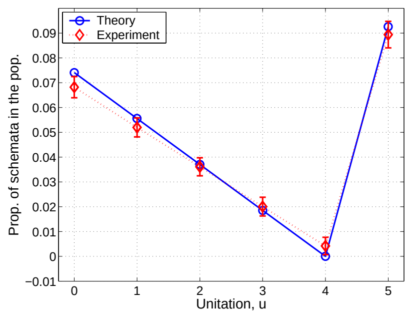
$$p_s(h_j) = \frac{\hat{f}_s(h_j)}{\sum_{i=1}^{2^k} \hat{f}_s(h_i)}, \tag{4}$$

where k is the order of sub-structure h_j and h_i are the sub-structures that compete with and belong to the same partition as h_j. We then use the above frequencies to sample the substructures to create new offspring.

Regardless of how the sub-structure preservation is done, the key idea is to preserve those sub-structures that are potentially highly fit and are a part of different global optima. The different sub-structure preservation methods usually requires modification of the sampling frequencies of sub-structures used in EDAs to generate new candidate solutions.

Before we use the proposed method for sustained maintenance of multiple global optima, we need to verify whether the sub-structure fitness estimate is accurate and if the method is capable of preserving different substructures over time. For the verification, we use fitness-proportionate method; that is, we maintain different substructures in proportion to their estimated fitness. We first investigate the accuracy of relative fitness estimates of different sub-structures in a given partition. The results for two different additively decomposable problems, m-k deceptive trap [2, 7, 6] and m-k bipolar function [11] are shown in figure 1. To demonstrate that the niching method can maintain all


(a) Trap: m = 10, k = 4


(b) Trap: m = 5, k = 5

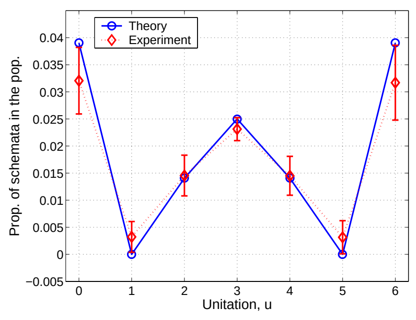

(c) Bipolar: m = 5, k = 6

Figure 1: Comparison of the ideal and experimental sub-structure frequencies for different additively separable problems.

substructures in a sustained manner over time, we plot the market share of each of the 16 schemata for the 10-4 deceptive trap functions in figure 2. The results show that the substructure-fitness estimation is quite accurate and that it can preserve the substructures at their desired proportions over time.

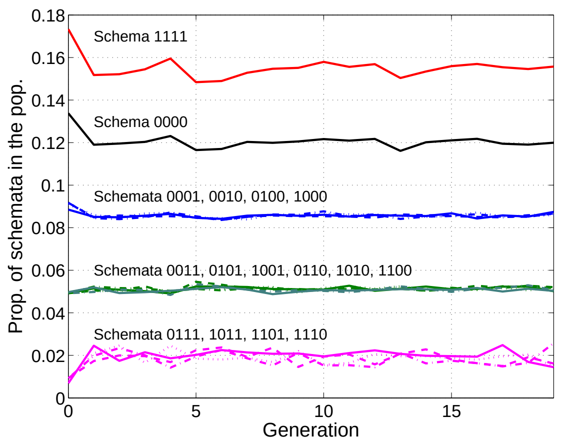


Figure 2: Illustration of sub-structure preservation via fitness-proportionate method for a 10-4 deceptive trap function

4. RESULTS AND DISCUSSION

In this section, we investigate the effectiveness of the substructural niching in stably maintaining all the global optima over a large number of generations, and the population size required to do so, as a function of the number of optima. We note that in all the results presented in this paper, we consider fitness-proportionate sub-structural niche preservation mechanism for proof-in-principle and, as mentioned earlier, other possible mechanisms can be more beneficial depending on the goal for using the niching mechanism. Additionally, the window size for RTS was set to the problem size, as suggested elsewhere [27]. Before presenting the results we first give a brief description of the test problem considered in the experiments.

Our approach in verifying the performance of substructural niching is to consider bounding *adversarial problems* that exploit one or more dimensions of problem difficulty [9]. Particularly, we are interested in problems where building-block identification is critical for the GA success. Additionally, the problem solver (eCGA) should not have any knowledge of the building-block structure of the test problem, but should be known to researchers for verification purposes.

One such class of problems is the m-k deceptive *trap* problem, which consists of additively separable *deceptive* functions [2, 7, 6]. Deceptive functions are designed to thwart the very mechanism of selectorecombinative search by punishing any localized hillclimbing and requiring mixing of whole building blocks at or above the order of deception. Using such *adversarially* designed functions is a stiff test—in some sense the stiffest test—of algorithm performance. The idea is that if an algorithm can beat an adversarially designed test function, it can solve other problems that are equally hard or easier than the adversarial function.

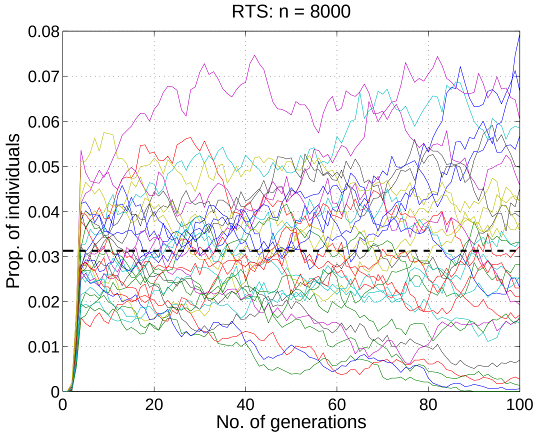
In this study, we use a modified m-4 deceptive trap problems where both 0000 and 1111 have equal fitness. Therefore there are 2^m global optima with an identical fitnesses. That is, each k-bit trap is defined as follows:

$$trap_k(u) = \begin{cases} 1 & \text{if } u = k \\ 1 & \text{if } u = 0 \\ 0.75\left[1 - \frac{u}{k-1}\right] & \text{otherwise} \end{cases}, \qquad (5)$$

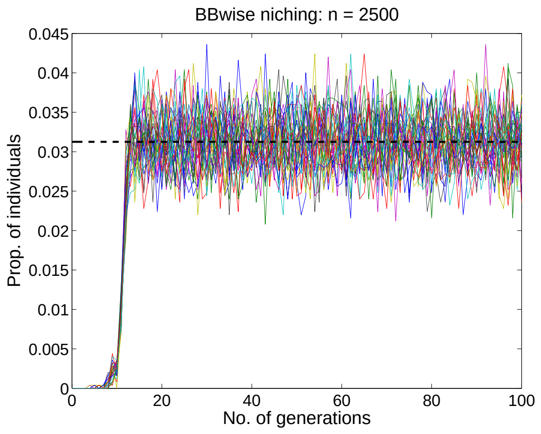
where u is the number of 1s in the input string of k bits.

First, we compare the ability of RTS and sub-structural niching in maintaining all the global optima over time in a stable manner (see figure 3). We start by comparing the single GA run behavior of both niching methods in figures 3(a) and 3(b), where we show the proportion of individuals in each of the 32 global optima of a 5-4 trap function as a function of time. The results clearly show that in contrast to RTS the niche maintenance of sub-structural niching is highly stable and the allocated market share for each optima is in agreement with the desired proportion of $1/32 = 0.03125$[1]. We then consider the average behavior of both RTS and sub-structural niching, where we plot the average market share of each of the global optima over time in figures 3(c) and 3(d). The lines above the bars in figures 3(c) and 3(d) depict the standard deviation and optimal solution

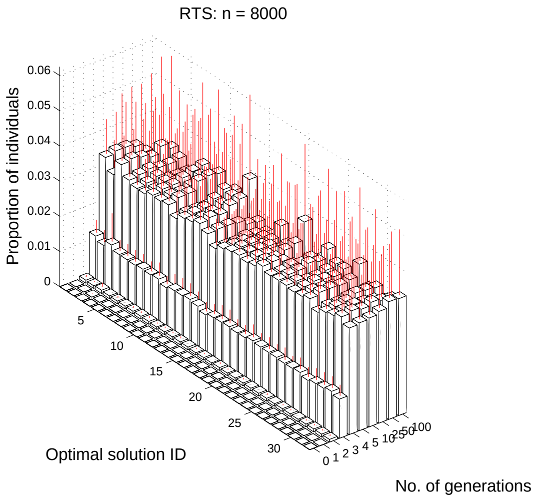
[1]Since all the global optima have identical fitness, we expect that the market share of each optima is $1/32 = 0.03125$.



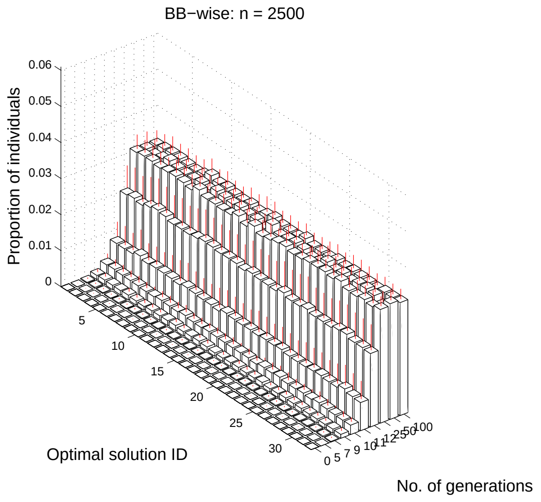
(a) RTS



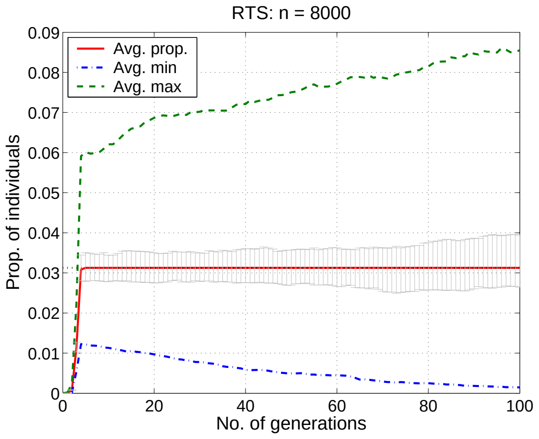
(b) Sub-structural niching



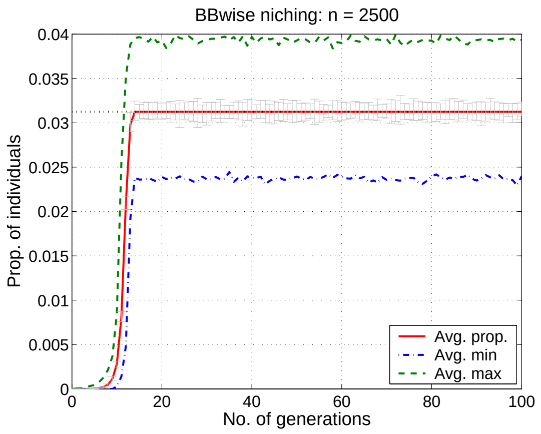
(c) RTS



(d) Sub-structural niching

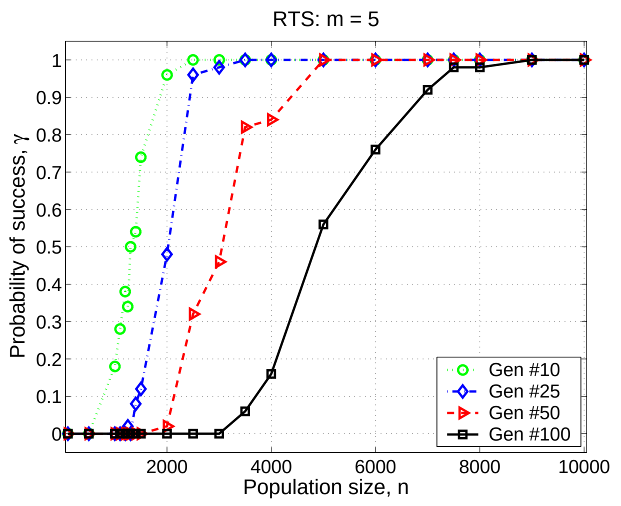


(e) RTS

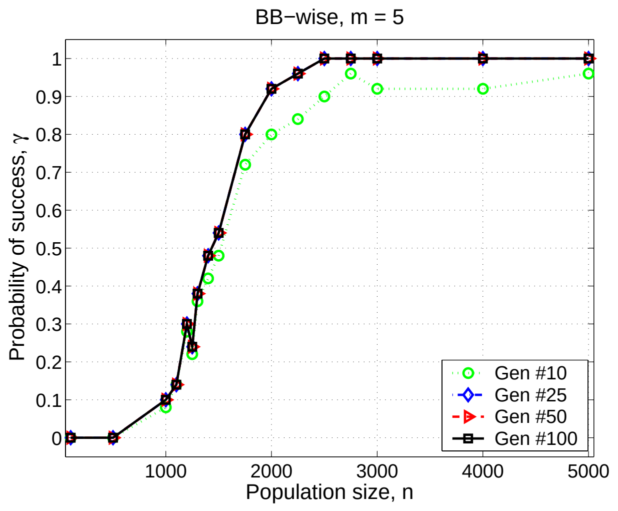


(f) Sub-structural niching

Figure 3: Comparison of the performance of RTS and sub-structural niching in stably maintaining all global optima for a concatenated 5-4 deceptive trap problem: (a) & (b) Single GA run behavior, (c) & (d) Average behavior, and (e) & (f) Average, average minimum and average maximum proportion allocated to an optima. The maintenance of all the optima by RTS is very noisy and unstable, while, sub-structural niching maintains all the niches stably over large number of generations. Additionally, the market share of each optima in sub-structural niching is close to the expected proportion of 0.03125. Results in (c)-(f) are averaged over 50 independent GA runs.



(a) RTS



(b) Sub-structural niching

Figure 4: The probability of maintaining at least one copy of all the global optima, γ, over different number of generations as a function of population size for RTS and sub-structural niching. RTS requires significantly larger population size to maintain all the global optima than the sub-structural niching. The results are averaged over 50 independent GA runs

ID refers to an arbitrary (but unique) number for each of the 32 global optima. For simplicity, we also plot the average, average minimum, and average maximum market share of an optima in figures 3(e) and 3(f). Figures 3(c)–3(f) clearly show that RTS cannot stably maintain the global optima, even at a larger population size, when compared to sub-structural niching.

Figure 3 clearly indicates overall the effectiveness of sub-structural niching in stably maintaining all the global optima at the desired proportion over large number of generations. We note that the time to detect the global optima is faster in RTS than in sub-structural niching. This is to be expected as sub-structural niching maintains diversity in all sub-structures proportional to their fitness, and it takes longer for mixing to hone in on to the global optima. However, once the optima are found, sub-structural niching preserves much more stably than RTS.

We also studied the effect of population size, n, on the success probability of maintaining at least one copy of all the global optima, γ, the results of which are shown for 5-4 deceptive trap function in figure 4. The figure plots the probability of maintaining all the global optima for different number of generations as a function of population size for both RTS and sub-structural niching. As shown in the figure, RTS, requires larger population sizes to maintain the global optima for longer time. This is well understood phenomena of traditional nichers and has been analyzed by Mahfoud for fitness sharing [23]. However, in sub-structural niching, the population size required to achieve a certain success probability, γ, is independent of the number of generations we would like to maintain the niches. Additionally, RTS requires significantly larger population size than sub-structural niching to achieve the same level of success probability.

Finally, we use the n versus γ results to determine the population size required to maintain successfully all the global optima with high probability. Specifically, we plot the minimum population size required by sub-structural niching and RTS for maintaining at least one copy of $n_{opt} - 1$ or more global optima in the population for different number of generations as a function of number of optima, n_{opt}, is figure 5. The lines plotted for the RTS results are from the population-sizing model of Mahfoud [23]:

$$n \propto \frac{\log\left[\left(1-\gamma^{1/t}\right)/n_{opt}\right]}{\log\left[\left(n_{opt}-1\right)/n_{opt}\right]}, \tag{6}$$

where t is the number of generations we need to maintain all the niches. Figure 5 clearly shows that sub-structural niching requires significantly less population size than RTS to maintain all the global solutions with a high probability. We note that the population size for sub-structural niching eventually will grow linearly (in accordance with the population-sizing model) with n_{opt} as we need at least one individual for each of the optima. However, this might not be the case for hierarchical and dynamic optimization problems, where diversity is required only at the sub-structural level and not at the solution level. Nevertheless, sub-structural niching requires orders of magnitude smaller populations than RTS and stably maintains niches with a high probability even for a problem with about a thousand global optima.

5. SUMMARY AND CONCLUSIONS

In this paper we proposed a sub-structural niching mechanism, which, in contrast to traditional niching mechanisms, exploits the problem decomposition capability of estimation distribution algorithms and stably maintains diversity at the sub-structure, or building-block level rather than at individual level. The sub-structural niching mechanism consists of three components: (1) Sub-structure identification, where we use the probabilistic model built by EDAs, specifically, extended compact GA [16], (2) sub-structure fitness estimation, where we use the fitness-estimation procedure proposed

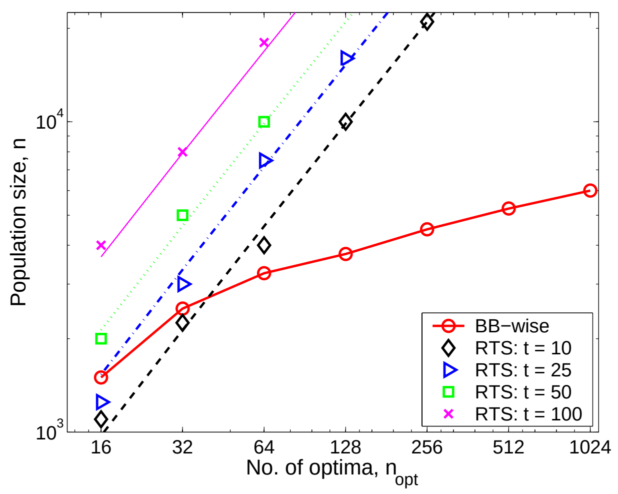


Figure 5: Minimum population sizing required for maintaining at least one copy of $n_{opt} - 1$ optima ($\gamma = (n_{opt} - 1)/n_{opt} = (2^m - 1)/2^m$) over different number of generations as a function of number of optima for RTS and sub-structural niching. The results for RTS conform to the population-sizing model of Mahfoud [23] (equation 6). The results are averaged over 50 independent GA runs.

by Sastry *et al* [34], and (3) sub-structure niche preservation, where different mechanisms are can be envisioned, each suitable based on the purpose and objective of using the niching method. Regardless of how it is done, the key idea of the sub-structure niche preservation mechanism is to preserve highly-fit sub-structures in desired proportions in the population in a stable manner over the duration of the search.

We also tested performance of the sub-structural niching mechanism on a class of boundedly-difficult additively separable multimodal problems and compared it with those of restricted tournament selection (RTS)—a niching method used in hierarchical Bayesian optimization algorithm. The results show that not only is the sub-structural niching mechanism able to stably preserve multiple global optima over large number of generations, but does so with a high probability requiring significantly less population size than RTS. The results indicate that sub-structural niching can be particularly effective with hierarchical and dynamic problem optimization.

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 ITR grant DMR-01-21695 at CPSD, and the Dept. of Energy under grant DEFG02-91ER45439 at Fredrick Seitz Materials Research Lab. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

6. REFERENCES

[1] H. A. Abbass, K. Sastry, and D. E. Goldberg. Oiling the wheels of change:The role of adaptive automatic problem decomposition in non-stationary environments. IlliGAL Report No. 2004027, University of Illinois at Urbana-Champaign, Urbana, IL, January 2004.

[2] D. H. Ackley. *A Connectionist Machine for Genetic Hill Climbing*. Kluwer Academic Publishers, 1987.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Boston, MA, 2001.

[4] D. J. Cavicchio. *Adaptive search using simulated evolution*. PhD thesis, Unversity of Michigan, Ann Arbor, MI, 1970.

[5] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975. (University Microfilms No. 76-9381).

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

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

[8] D. E. Goldberg. The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In P. Bentley, editor, *Evolutionary Design by Computers*, chapter 4, pages 105–118. Morgan Kaufmann, San Mateo, CA, 1999.

[9] D. E. Goldberg. *Design Of Innovation: Lessons from and for competent genetic algorithms*. Kluwer Acadamic Publishers, Boston, MA, 2002.

[10] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992. (Also IlliGAL Report No. 91010).

[11] D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. *Parallel Problem Solving from Nature*, 2:37–46, 1992. (Also IlliGAL Report No. 92007).

[12] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms*, pages 56–64, 1993. (Also IlliGAL Report No. 93004).

[13] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989. (Also IlliGAL Report No. 89003).

[14] D. E. Goldberg and J. J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[15] D. E. Goldberg and L. Wang. Adaptive niching via coevolutionary sharing. In D. Quagliarella, J. Periaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms in Engineering and Computer Science*, pages 21–38. John Wiley and Sons, Ltd., Chichester, NY, 1998. (Also IlliGAL Report No. 97007).

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

[17] G. R. Harik. Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, 1995. (Also IlliGAL Report No. 94002).

[18] G. R. Harik. *Learning linkage to eficiently solve problems of bounded difficulty using genetic algorithms*. PhD thesis, University of Michigan, Ann Arbor, MI, 1997. (Also IlliGAL Report No. 97005).

[19] J. Horn. *The nature of niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1997. (Also IlliGAL Report No. 97008).

[20] H. Kargupta. The gene expression messy genetic algorithm. *Proceedings of the International Conference on Evolutionary Computation*, pages 814–819, 1996.

[21] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[22] S. W. Mahfoud. Crowding and preselection revisited. *Parallel Problem Solving from Nature*, 2:27–36, 1992. (Also IlliGAL Report No. 92004).

[23] S. W. Mahfoud. Population size and genetic drift in fitness sharing. *Foundations of Genetic Algorithms*, 3:185–224, 1994. (Also IlliGAL Report No. 94005).

[24] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1995. (Also IlliGAL Report No. 95001).

[25] O. J. Mengshoel and D. E. Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 409–416, 1999. (Also IlliGAL Report No. 99004).

[26] M. Munetomo and D. E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4):377–398, 1999.

[27] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 511–518, 2001. (Also IlliGAL Report No. 2000020).

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

[29] M. Pelikan, F. Lobo, and D. E. Goldberg. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21:5–20, 2002. (Also IlliGAL Report No. 99018).

[30] M. Pelikan and K. Sastry. Fitness inheritance in the bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:48–59, 2004.

[31] M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2003. (Also IlliGAL Report No. 2002024).

[32] J. J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[33] K. Sastry, H. A. Abbass, and D. E. Goldberg. Sub-structural niching in non-stationary environments. *Proceedings of the Australian Conference on Artificial Intelligence*, pages 873–885, 2004.

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

[35] X. Yin and N. Germay. Improving genetic algorithms with sharing through cluster analysis. *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 100, 1993.

[36] 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. *Artificial Neural Networks in Engineering*, pages 327–332, 2003. (Also IlliGAL Report No. 2003007).