

**On-line Population Size Adjusting
Using Noise and Substructural Measurements**

**Tian-Li Yu
Kumara Sastry
David E. Goldberg**

IlliGAL Report No. 2005017
April, 2005

Illinois Genetic Algorithms Laboratory (IlligAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue, Urbana, IL 61801
<http://www-illigal.ge.uiuc.edu>

Online Population Size Adjusting Using Noise and Substructural Measurements

Tian-Li Yu, Kumara Sastry, and David E. Goldberg
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{tianliyu, kumara, deg}@illigal.ge.uiuc.edu

Abstract

This paper proposes an online population size adjustment scheme for genetic algorithms. It utilizes linkage-model-building techniques to calculate the parameters used in facetwise population-sizing models. The methodology is demonstrated using the dependency structure matrix genetic algorithm on a set of boundedly-difficult problems. Empirical results indicate that the proposed method is both efficient and robust. If the initial population size is too large, the proposed method automatically decreases the population size, and thereby yields significant savings in the number of function evaluations required to obtain high-quality solutions; if the initial population size is too small, the proposed scheme increases the population size on-the-fly and thereby avoiding premature convergence.

1 Introduction

One of the challenges faced by genetic and evolutionary algorithm (GEA) users is making appropriate choices of codings, operators, and parameter values. Several linkage-learning methods have since been developed to relieve GEA practitioners from having to develop problem-specific operators and encodings (Goldberg, 1999a). These linkage-learning methods automatically identify key sub-structures of the underlying search problem and solve them quickly, reliably, and accurately. Additionally, facetwise models have been developed to estimate the genetic-algorithm (GA) parameter values such as population size, run duration, crossover and mutation probabilities, and selection pressure (Goldberg, 2002). Subsequently, the results of the facetwise models have been used in parameterless GEAs have been developed (Harik & Lobo, 1999; Lobo, 2000; Pelikan & Lin, 2004) to alleviate the guess work out of parameter settings.

However, the parameterless GEAs do not completely use the population-sizing models and are restricted to having multiple populations with exponential increasing population sizes. While some of the factors in the population-sizing models are not known apriori, they can be estimated on-the-fly with linkage-learning GEAs in a straightforward manner. Therefore, the purpose of this paper is to develop an on-line population-sizing estimate which automatically identifies values of key components of the population-sizing model. We demonstrate the effectiveness of the proposed method using design structure matrix GA (Yu, Goldberg, Yassine, & Chen, 2003)—a linkage-learning GEA that is inspired by organization theory—on a class of boundedly difficult test problems. We demonstrate that the proposed method is not only robust, but also efficient in terms of number of function evaluations required to obtain high quality solutions.

This paper is organized as follows. Section 2 revisits some related work concerning adjusting population size on-the-fly. Then we give background for several facetwise population-sizing models

in Section 3. A brief introduction of dependency structure matrix genetic algorithm (DSMGA) is given in Section 4. The population size needed for DSMGA is empirically verified with the population-sizing model for linkage learning. Section 5 describes the propose method in detail, including how to calculate parameters needed by population-sizing models, and how to generate individuals for the next generation to maintain both efficiency and robustness. Section 6 proposes a scheme to improve the robustness when the GA starts from a small population. Finally, section 7 concludes this paper.

2 Related Work

Since the inception of GAs, numerous studies have been conducted on on-line and off-line parameter settings. A detailed survey of which is beyond the scope of this study and and the interested reader is referred elsewhere (Harik & Lobo, 1999; Lobo, 2000). Since we propose an on-line population resizing mechanisms for linkage-learning GEAs in this paper, we will briefly review related work on adaptive resizing of population sizes in the remainder of this section.

One of the early effort on resizing the population during the GA run can be backtracked to Smith (1993) and Smith and Smuda (1995). They dynamically resized the population to match a target selection loss L_t . They calculated the estimated selection loss \hat{L} for each mating pair. By adopting a sigmoid function, the fraction $\frac{\hat{L}}{L_t}$ is used to conduct a growth factor, which is then used to adjust the population size. One of the weakness of the methodology proposed by Smith (1993) and Smith and Smuda (1995) is that users need to determine three parameter values: the target selection loss, and two parameters (α and β) in the sigmoid function used to compute the growth factor.

Harik and Lobo (1999) proposed a parameter-less GA, which is composed of two parts: eliminating selection pressure and crossover rate, and eliminating population size. The parameter-less GA eliminates population size by simulating a set of populations of different sizes; starting with a population of base size, the subsequent populations are twice the size of the previous one. Then, the GA operators are applied to each population in a special sequence. For every two GA generations for a populations of a particular size, a single GA generation is performed on a population of twice the size. Therefore, the number of function evaluations performed on each population are roughly the same. The parameter-less GA terminates a population when no improvement is expected from that population. The parameter-less GA stops when no improvement is expected from the larger populations as well. Suppose that a regular GA with an optimal population-size setting needs n_{fe}^* number of function evaluations for a particular problem. It is shown that the parameter-less GA uses no more than $O(n_{fe}^* \log(n_{fe}^*))$ (Pelikan & Lobo, 1999) number of function evaluations.

The parameter-less GA technique has been applied not only to simple GAs (Harik & Lobo, 1999), but also to linkage-learning GAs such as the extended compact GA (eCGA) (Lobo, 2000), and the hierarchical Bayesian optimization algorithm (hBOA) (Pelikan & Lin, 2004).

The method used in this paper to estimate the fitness variance of building blocks is similar to that in Smith (1993) and Smith and Smuda (1995) but no more parameter needed. Also, with the power of the linkage model, empirical results show that the population size estimation is more accurate, and the GA consumes roughly the same number of function evaluation with the GA with optimal population size setting. The method will be detailed in Section 5.

3 Population-Sizing Models

Facetwise and dimensional models have been very effective not only in the design of genetic algorithms, but also in understanding GA dynamics and mechanisms. Since our methodology depends on the facetwise models of population sizing, we briefly outline the models dictated by building block supply, decision making, and accurate linkage learning in the remainder of this section.

3.1 Building-Block Supply Model

The first step towards understanding population sizing is to tackle the issue of building-block (BB) supply, where the minimum population size required to ensure the presence of at least one copy of all raw schemata is modeled. Holland (1975) estimated the number of BBs that receive at least a specified number of trials using Poisson distribution. A later study (Goldberg, 1989) calculated the same quantity more exactly using binomial distribution and studied their effects on population sizing in serial and parallel computation. Reeves (1993) proposed a population sizing model for supply of alphabets with fixed cardinality. Recently, Goldberg, Sastry, and Latoza (2001) developed facetwise models for ensuring BB supply in the initial population for genetic algorithms. They considered a population of fixed-length strings consisting alphabets of arbitrary cardinality χ . Goldberg *et al* predicted that the population size required to ensure the presence of all competing building blocks with a tolerance of $\epsilon = 1/m$ is given by

$$n = \chi^k (k \log \chi + \log m), \quad (1)$$

where χ is the alphabet cardinality, k is BB size, and m is the number of BBs.

3.2 Gambler’s Ruin Population-Sizing Model

Goldberg, Deb, and Clark (1992) proposed population-sizing models for correctly deciding between competing BBs. They incorporated noise arising from other partitions into their model. However, they assumed that if wrong BBs were chosen in the first generation, the GAs would be unable to recover from the error. Harik, Cantú-Paz, Goldberg, and Miller (1999) refined the above model by incorporating cumulative effects of decision making over time rather than in first generation only. Harik et al. (1999) modeled the decision making between the best and second best BBs in a partition as a gambler’s ruin problem. Here we use an approximate form of the population-sizing model proposed by Harik et al. (1999):

$$n = \frac{\sqrt{\pi} \sigma_{BB}}{2} \frac{2^k}{d} \sqrt{m} \log m, \quad (2)$$

where k is the BB size, m is the number of BBs, d is the size signal between the competing BBs, and σ_{BB} is the fitness variance of a building block. building blocks. The above equation assumes a failure probability, $\alpha = 1/m$.

3.3 Model-Building+Decision Making Population Sizing

Facetwise modes for incorporating the effects of model building and BB-wise decision making on the population size have been analyzed for estimation of distribution algorithms (EDAs) in general, and Bayesian optimization algorithm and extended compact genetic algorithm in particular (Pelikan, Goldberg, & Cantú-Paz, 2000; Pelikan, Sastry, & Goldberg, 2003; Sastry & Goldberg, 2000; Sastry & Goldberg, 2004). The population-sizing model which incorporates the effect of model-building

and its accuracy on the population sizing of the GA, and predicts the population size required to solve a problem with m building blocks of size k with a failure rate of $\alpha = 1/m$, is given by

$$n = c_n \cdot 2^k \left(\frac{\sigma_{BB}}{d} \right)^2 m \log m, \quad (3)$$

where n is the population size, c_n is a problem-dependent constant, k is the BB length, α is the probability of failure.

4 An Introduction to DSMGA

This section gives a brief introduction to the DSMGA. Readers who are interested in DSM clustering are referred to Yu, Yassine, and Goldberg (2003). For more details about DSMGA, please refer to Yu, Goldberg, Yassine, and Chen (2003).

4.1 DSM and DSM Clustering

A DSM is a matrix where each entry d_{ij} represents the dependency between node i and node j (Steward, 1981; Yassine, Falkenburg, & Chelst, 1999). Entries d_{ij} can be real numbers or integers. The larger the d_{ij} is, the higher the dependency is between node i and node j . The diagonal entries (d_{ii}) have no significance and are usually set to zero or blacked-out. Figure 1 gives an example of DSM.

	A	B	C	D	E	F	G	H
A	■		×			×		×
B		■			×			×
C	×		■			×		×
D		×		■				×
E	×		×		■			×
F						■		
G		×		×			■	
H	×		×		×			■

Figure 1: A DSM. “x” means that dependency exists; the blank squares means no dependency. This figure illustrates, for example, that A and B are independent, and that A and C are dependent. Clustering is not so obviously at the first glance.

The goal of DSM clustering is to find subsets of DSM elements (i.e., clusters) so that nodes within a cluster are maximally dependent and clusters are minimally interacting. DSM clustering is a sophisticated task which requires expertise (Sharman, Yassine, & Carlile, 2002). For example, it is not intuitive how to cluster the DSM in Figure 1. However, after we reorder the nodes (Figure 2), it is easily seen that the DSM can be cleanly clustered into three parts: $\{B, D, G\}$, $\{A, C, E, H\}$, and $\{F\}$.

Yu, Yassine, and Goldberg (2003) proposed the following objective function by using the minimal description length principle.

$$f_{DSM}(M) = (n_c \log(n_c) + \log(n_n) \sum_{i=1}^{n_c} cl_i) + (|S|(2 \log(n_n) + 1)), \quad (4)$$

where f measures the description length that a model M needs to describe a given DSM; n_c is the number of clusters in M ; n_n is the number of nodes of the DSM; cl_i is the size of the i -th cluster in M ; S is the set of mis-described data of M . The above objective function has shown capable to cluster a given DSM, and the clustering results competes with human experts.

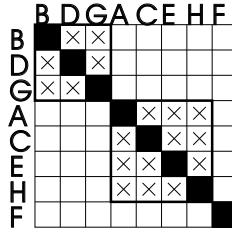


Figure 2: The same DSM in Figure 1 but after reordered. The DSM can be cleanly clustered as ((B,D,G)(A,C,E,H),(F)).

4.2 Utilizing DSM Clustering to Identify BBs: The DSMGA

The DSM clustering algorithm can be thought as an linkage-identification algorithm which turns pair-wise linkage information into high-order linkage information, and DSMGA (Yu, Goldberg, Yassine, & Chen, 2003) is motivated based on this thought.

DSMGA utilizes statistical analysis to estimate the fitness of order-2 schemata. Based on a nonlinearity-detection method similar to LINC (Munetomo & Goldberg, 1999), a DSM where each entry d_{ij} represents the linkage between gene i and gene j is created. By applying the DSM clustering algorithm, the pair-wise linkage information is turned into BB information, which is then utilized to achieve the BB-wise crossover (Thierens & Goldberg, 1994). The DSMGA has shown capable to correctly identify BBs and efficiently solve problems with uniformly-scaled BBs.

4.3 Population Sizing for DSMGA

Unlike EDAs, the linkage model in DSMGA is not exactly a probability density function. Nevertheless, empirical finding shows that the population-sizing model (Equation 3) for EDAs predicts well the population size needed by DSMGA (Figure 3). The test function is an m - k trap (Deb & Goldberg, 1993), where $m = 10$ and $k = 3$. The 3-bit trap function is defined as follows:

$$trap_3(u) = \begin{cases} 0.8, & u = 0 \\ 0.4, & u = 1 \\ 0.0, & u = 2 \\ 1.0, & u = 3 \end{cases} \quad (5)$$

where u is unitary (the number of 1's among the 3 bits). By taking average, the constant c_n is empirically found to be roughly 1.2 for this 10-3 trap. This test function is also used for the experiments in later sections.

5 Methodology

This section describes how to calculate parameters needed in population-sizing models and how to generate individuals for the next generation so that both efficiency and robustness are maintained.

5.1 Population Size Estimation: Calculating Parameters in Population-Sizing Models

The BB information given by DSMGA tells us explicitly which genes form a BB. Therefore, calculating the number of BBs (m) and the size of BBs (k) is straightforward given the DSM clustering

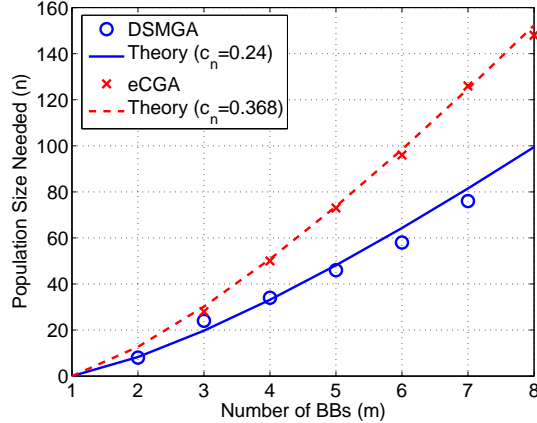


Figure 3: The population-sizing model for EDAs predicts well the population size needed by DSMGA.

arrangement in DSMGA. For example, suppose that DSMGA tells us that $BB_1 = \{x_2, x_4\}$ and $BB_2 = \{x_1, x_3, x_5\}$ (where x 's are genes), we know that $k_1 = 2$, $k_2 = 3$, and $m = 2$. For problems with the same size of BBs, we can simply take the average and then the estimated parameters would be $\hat{k} = 2.5$ and $\hat{m} = 2$.

The calculations of the fitness variance of BBs (σ_{BB}) and the signal size between competing BBs (d) are not any more difficult. First, define $H_{BB,i}$ as the schemata (Holland, 1975) which have allele values (0 or 1) for those genes in that BB and wild cards (*) for any other positions; i is just an enumerated number. In the previous example, $BB_1 = \{x_2, x_4\}$, we have $H_{BB_1,1} = *0*0*$, $H_{BB_1,2} = *0*1*$, $H_{BB_1,3} = *1*0*$, $H_{BB_1,4} = *1*1*$. Note that there are 2^k such schemata for a BB of size k . Define $f(H)$ as the fitness value for schema H . We can then express σ_{BB} and d as follows:

$$\sigma_{BB}^2 = \text{var}_i[f(H_{BB,i})], \quad (6)$$

$$d = \text{max}_i[f(H_{BB,i})] - \text{second_max}_i[f(H_{BB,i})], \quad (7)$$

where var is the variance function, and here we assume a maximization problem. The fitness of a schema can be estimated from the current population P by examining each individual x_i : $\hat{f}(H) = \text{mean}_{i, x_i \in H}[f(x_i)]$, where mean is the arithmetic averaging function.

5.2 Population Size Adjustment: First Attempt and Empirical Results

After estimating the population size for the next generation, the most straightforward way to generation the next population is to simply apply GA operators to generate offspring of the desired size.

We test the proposed method on a m - k deceptive trap problem, where $m = 10$ and $k = 3$. We compare the performance of a DSMGA with on-line population-size adjustmentADJ and a DSMGA with fixed population sizes (FIX). In particular, we focus on the efficiency and robustness of each of the two GA methodologies. By efficiency we mean the number of function evaluations that the GAs need to converge to a solution of predefined quality. By robustness we mean that the rate that the GAs fail to converge. The termination condition is that on average $(m - 1)/m = 90\%$

percent of BBs converge correctly. The failure rate used in the population size estimation is set to $1/m = 10\%$. All results are averaged over 30 independent runs.

Figure 4 shows the number of function evaluations that the GAs need to converge solutions with 90% correct BBs by varying the initial population size (n_0). Given the empirical finding that $c_n \simeq 0.24$ for DSMGA on this test function and the failure rate 0.1, the population size needed by FIX is roughly 137. As shown in Figure 4, FIX consumes the least number of function evaluation (n_{fe}) when the initial population size n_0 is set close to $n^* = 137$ (we do not count the case for $n_0 = 100$, since FIX does not usually converge for that fixed population size). If the population size in FIX is less than the required size (even 30% less), the GA converges to a local optimum and therefore does not yield high quality solutions. On the other hand, if the population size is greater than the required size, the number of function evaluations grows linearly with the initial population size n_0 , thereby wasting significant amount of computation resources unnecessarily. In contrast to FIX, ADJ is able to properly shrink the population size if n_0 is large, and hence consumes roughly a constant number of function evaluations when n_0 is slight larger than n^* .

We can estimate the trend of the behavior for these two algorithms. Let n_{fe}^* be the number of function evaluations needed for FIX when $n_0 = n^*$. Empirically, $n_{fe}^* \simeq 2700$. Following the facetwise modeling approach (Goldberg, 2002), we assume the running duration is nearly independent of the population size. For $n_0 = 600$, we can compute that the number of generations that FIX runs before convergence $g \simeq 14.8$. Therefore, we can estimate the number of function evaluations need for FIX for a fixed population size n_0 is roughly (the **FIX-trendline** in Figure 4):

$$n_{fe-FIX} = g \cdot n_0. \quad (8)$$

In ADJ, for a large initial population size n_0 , we can assume that linkage model is quite accurate because n_0 is large. Hence, ADJ should estimate the population size accurate (close to n^*) for the next generation. Hereafter, we then assume ADJ performs in a similar behavior of FIX at an optimal population size n^* . Therefore, we model the total number of function evaluations needed for ADJ is the number of function evaluations in the first generation (n_0) plus the number of function evaluations needed for the optimal population size setting (n_{fe}^*) is roughly (the **ADJ-trendline** in Figure 4):

$$n_{fe-ADJ} = n_{fe}^* + n_0. \quad (9)$$

Compare Equations 8 and 9, we can see a dramatic reduction in the number of function evaluations when the on-line population size adjustment scheme is applied, especially for a large initial population size. Now we examine the algorithms from another direction: robustness for small initial population size. Figure 5 shows the rate that FIX and ADJ fail to converge for different small n_0 . For small n_0 , FIX rarely converges. According to the population-sizing theory, FIX does not have enough BB supply and cannot decide well when $n_0 < n_*$. When the initial population size is very small, $n_0 \ll n^*$, both FIX and ADJ fail to yield high-quality solutions. However, when the initial population is slightly increased (but still $n_0 < n^*$), the failure rate of ADJ decreases rapidly, while FIX consistently fails to yield high-quality solutions. The robustness of ADJ can be further enhanced by ensuring sufficient BB supply and maintaining diversity, especially in the early stage of the GA run (the method will be described in the next section). To understand the reason for ADJ to fail on a very small initial population size, imagine that $n^* = 1000$, and ADJ starts from $n_0 = 10$. Even if the linkage model is luckily accurate and estimates the the population size for the next generation should be 1000, ADJ generates the 1000 offspring chromosomes from only the 10 parent chromosomes. As can be anticipated, the newly generated population lacks of diversity and ADJ converges prematurely.

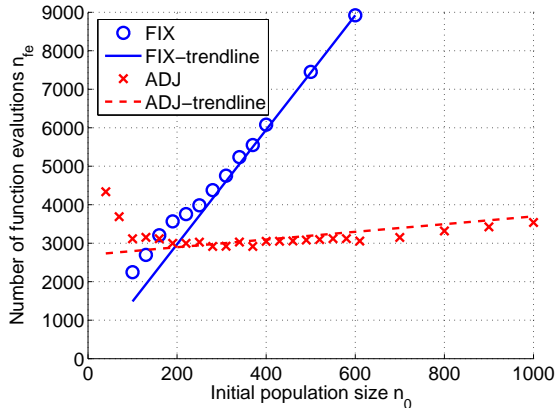


Figure 4: Number of function evaluations consumed by DSMGAs with a fixed population (FIX) and with the population adjustment scheme (ADJ). ADJ needs slightly more number of function evaluations than FIX when a near-optimal population size (n^*) is used. For large initial population size, however, ADJ constantly outperforms FIX. For a too small initial population size, both FIX and ADJ do not converge.

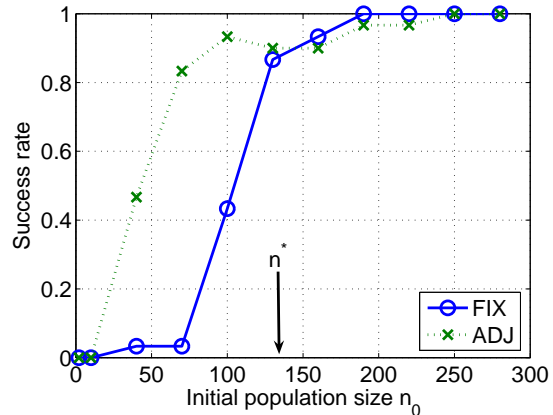


Figure 5: The failure rates for FIX and ADJ for different n_0 . FIX rarely converges for $n_0 < n^*$ as the population size predicts. ADJ is slightly more robust than FIX, but it still suffers from insufficient BB supply and the lack of diversity for small n_0 .

6 A More Robust Scheme for Small Initial Population Size

The online population size adjustment scheme described in the previous section allows users to start with a large population which guarantees good solution quality without spending too many extra function evaluations. While the on-line population adjustment scheme in the previous section (ADJ) is significantly more robust and efficient than a GA with a fixed population size, the performance—both in terms of efficiency and robustness—can be further enhanced when the GA practitioner sets the initial population size to be much smaller than the required size. This section proposes an new-individual-injection scheme to improve the robustness of the population size adjustment scheme.

6.1 New Individual Injection Scheme

As mentioned in the previous section, in ADJ the new population is generated via the variation operators—crossover and mutation—of DSMGA. While this is a straightforward method, it does not alleviate the problem of insufficient BB supply and the lack of diversity in the population. To circumvent these problems, new individuals need to be introduced into the population at the right time. One of the most intrinsic way is to introduce new individuals whenever the estimated population size is larger than the current population size.

In the previous example, suppose that $n^* = 1000$, $n_0 = 10$, and the GA estimates the population size for the next generation should be 1000. Then the GA generates 10 offspring chromosomes by recombining the 10 parent chromosomes and the generate the other 990 offspring chromosome by

random initialization. However, the injection of randomly generated solutions into the population, especially during the later stages of the GA run has one sever drawback. Due to random initialization, the average solution quality goes down and the fitness variance goes up causing an elongation of the run duration (Goldberg, 1999b). As shown in Figure 6, whenever randomly initialized solutions are injected into the population, the proportion of correct BBs in the population is reduced causing a delay in the convergence.

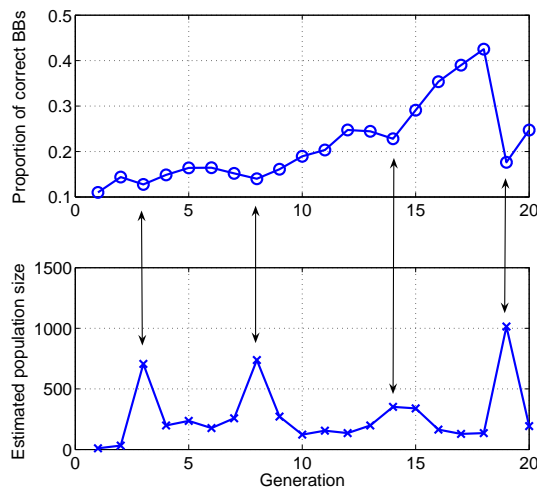


Figure 6: One run of the GA with the introducing-new-individual scheme switched on all the time. The GA suffers from slow convergence because new individuals keep joining. The arrows indicates that whenever new individuals joins the population, the number of correct BBs goes down.

To alleviate the elongation in the run duration and also to ensure adequate supply of BBs, we should inject randomly initialized solutions into the population during the initial stages of the GA run and then create the new solutions only through recombination in the later stages. Using recombination during the later stages of the GA run not only circumvents the delay in convergence, but also significantly improves the exchange of building blocks and thereby allowing rapid creation of good quality solutions. One choice for the switching time can be the generation when the estimated population size for the next generation is smaller than the current population size. Intuitively this reduction in the population size indicates that the GA has sufficient supply of raw BBs and good decision making can be ensured. Therefore, after the first time that the estimated population size is smaller than the current one, we do not inject any new individual into the population.

6.2 Empirical Results

The test function is the same m - k trap with $m = 10$ and $k = 3$. Two different GAs are compared: (1)DSMGA + population-size-adjustment, and (2) DSMGA + population-size-adjustment + new-individual-injection. For simplicity, we call the first GA **ADJ** and the second **ADJ+INJECT**. The parameter settings are the same as what used in the previous section. And again, all results are averaged out of 30 independent runs.

Figure 7 shows the number of function evaluations that the GAs need to converge by varying the initial population size (n_0). As shown in the figure, the number of function evaluations needed for ADJ and ADJ+INJECT are roughly the same for a wide range of initial population size n_0 .

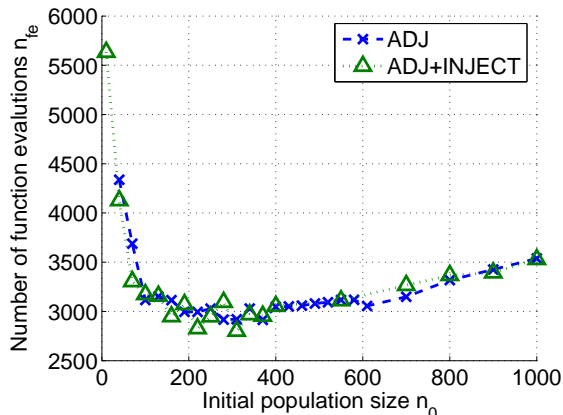


Figure 7: Number of function evaluations consumed by DSMGAs with the population adjustment scheme (ADJ) and with population adjustment scheme + the injection scheme (ADJ+INJECT). The number of function evaluations needed for both algorithms are roughly the same.

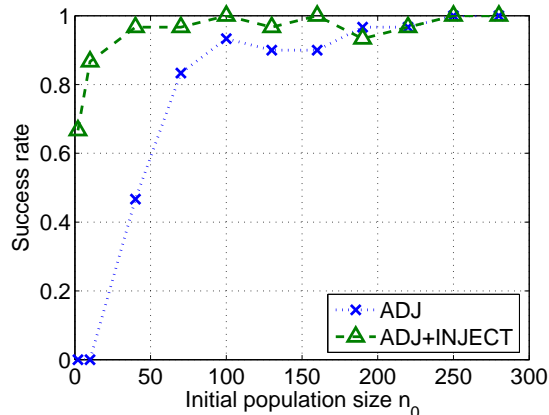


Figure 8: The failure rates for ADJ and ADJ+INJECT for different n_0 . ADJ+INJECT is significantly more robust than ADJ. ADJ+INJECT successfully converges for a wide range of initial population size n_0 .

Figure 8 shows the failure rate of ADJ and ADJ+INJECT. As can be seen in the figure, ADJ+INJECT is significantly more robust than ADJ. Even for $n_0 = 2$ (which is ridiculously small), ADJ+INJECT still converges to the global optimum 66.7% of the time.

Finally, Figure 9 illustrates the population estimation behavior of ADJ+INJECT for different initial population size n_0 . Except the the too small initial population size $n_0 = 10$, the behaviors are similar for different n_0 . At the beginning, the linkage model indicates that more individuals are needed. Towards the end of GA run, the population loses the diversity (fitness variance of BB in the current population is small), and hence not many individuals are needed. The behavior well match the up-to-date GA theory, and moreover, the similar of population size estimation behavior provides a stable number of function evaluations for varying initial population sizes.

7 Conclusion

This paper presents an online population size adjustment scheme based on facetwise population-sizing models. Given the linkage model, we calculate the parameters required by population-sizing models and estimate the population size for the next generation according to the linkage-learning population-sizing model (Equation 3). The individuals of the next generation are generated by a new-individual-injection scheme. The scheme provides sufficient BB supply at the beginning of the GA run even for a small initial population size, and concentrate on mixing to ensure good solution quality toward the end of the GA run. The proposed method is shown to have both efficiency and robustness. With the proposed online population size adjustment scheme, the number of function evaluations needed grows slowly with large initial population size, and the GA has a higher probability to converges even with a small initial population size.

As for future work, we would like to apply this scheme to other linkage-learning GEAs (*e.g.*

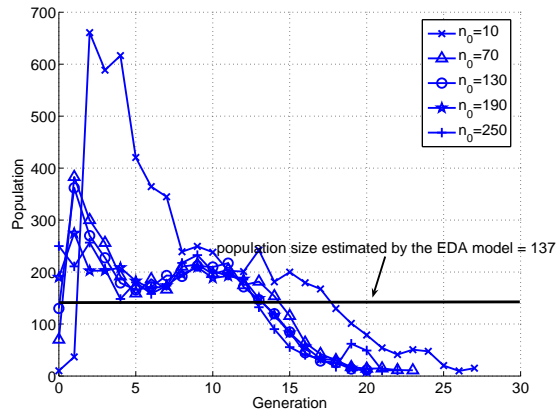


Figure 9: The population estimation behavior of ADJ+INJECT for different initial population size n_0 .

eCGA (Harik, 1999)) and test the scalability. We would also like to add an update rule to the population-sizing scheme. For example, if the estimated population size is \hat{n} and the current population size is n_t , the next population size is decided by $n_{t+1} = \alpha n_t + (1 - \alpha)\hat{n}$. Although the update rule introduce a new parameter α , it might add some values of stability.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). The US 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 Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

References

- Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2*, 93–108.
- Goldberg, D. E. (1989). Sizing populations for serial and parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, 70–79.
- Goldberg, D. E. (1999a). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In Bentley, P. (Ed.), *Evolutionary Design by Computers* (Chapter 4, pp. 105–118). San Mateo, CA: Morgan Kaufmann.
- Goldberg, D. E. (1999b). Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. *Proceedings of the Genetic and Evolutionary Computation Conference*, 212–219.
- Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Academic Publishers.

- Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6, 333–362. (Also IlliGAL Report No. 91010).
- Goldberg, D. E., Sastry, K., & Latoza, T. (2001). On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference*, 336–342.
- Harik, G. (1999, February). *Linkage Learning via Probabilistic Modeling in the ECGA* (IlliGAL Report No. 99010). Urbana, IL: University of Illinois at Urbana-Champaign.
- Harik, G., Cantú-Paz, E., Goldberg, D. E., & 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).
- Harik, G., & Lobo, F. (1999). A parameter-less genetic algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference*, 258–265.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Lobo, F. (2000, July). *The parameter-less genetic algorithm: Rational and automated parameter selection for simplified genetic algorithm operation*. phdthesis, University of Lisbon, Portugal.
- Munetomo, M., & Goldberg, D. E. (1999). Identifying linkage groups by nonlinearity/non-monotonicity detection. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1*, 433–440.
- Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). Bayesian optimization algorithm, population sizing, and time to convergence. *Proceedings of the Genetic and Evolutionary Computation Conference*, 275–282. (Also IlliGAL Report No. 2000001).
- Pelikan, M., & Lin, T.-K. (2004). Parameter-less hierarchical boA. *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO-2004)*, 24–35.
- Pelikan, M., & Lobo, F. (1999). *Parameter-less genetic algorithm: A worst-case time and space complexity analysis* (IlliGAL Report No. 99014). Urbana, IL: Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.
- Pelikan, M., Sastry, K., & Goldberg, D. E. (2003). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3), 221–258. (Also IlliGAL Report No. 2002024).
- Reeves, C. (1993). Using genetic algorithms with small populations. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 92–99.
- Sastry, K., & Goldberg, D. E. (2000). On extended compact genetic algorithm. *Late-Breaking Paper at the Genetic and Evolutionary Computation Conference*, 352–359. (Also IlliGAL Report No. 2000026).
- Sastry, K., & Goldberg, D. E. (2004). 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.
- Sharman, D., Yassine, A., & Carlile, P. (2002, Sept.). Characterizing modular architectures. *ASME 14th International Conference, DTM-34024*.
- Smith, R. E. (1993). Adaptively resizing populations: An algorithm and analysis. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 653.
- Smith, R. E., & Smuda, E. (1995). Adaptively resizing populations: Algorithm, analysis, and first results. *Complex Systems*, 1(9), 47–72.
- Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28, 77–74.
- Thierens, D., & Goldberg, D. E. (1994). Convergence models of genetic algorithm selection schemes. In *Parallel Problem Solving from Nature, PPSN III* (pp. 119–129).
- Yassine, A., Falkenburg, D. R., & Chelst, K. (1999). Engineering design management: An informatoin structure approach. *International Journal of production research*, 37(13), 2957–2975.

- Yu, T.-L., Goldberg, D. E., Yassine, A., & Chen, Y.-p. (2003). Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. *Proceedings of Artificial Neural Networks in Engineering 2003 (ANNIE 2003)*, 327–332. (Also IlliGAL Report No. 2003007).
- Yu, T.-L., Yassine, A., & Goldberg, D. E. (2003). A genetic algorithm for developing modular product architectures. *Proceedings of the ASME 2003 International Design Engineering Technical Conferences, 15th International Conference on Design Theory and Methodology (DETC 2003)*, DTM-48657. (Also IlliGAL Report No. 2003024).