International Journal of Computer Mathematics
Vol. 87, No. 3, March 2010, 476–490

Taylor & Francis
Taylor & Francis Group

Homogeneous genetic algorithms

Alexander Stanoyevitch*

Mathematics Department, California State University, Carson, CA, USA

(*Received 05 June 2007; revised version received 31 October 2007; accepted 20 December 2007*)

A new type of genetic algorithm (GA) is developed to mitigate one or both of the following two major difficulties that traditional GAs may suffer: (1) when the number of 'active genes' needs to be held constant or kept within some prescribed range, and (2) when the set of genes is much larger than the set of active genes of feasible solutions under consideration. These homogeneous GAs use (unordered) sets to represent 'active genes' in chromosomes rather than strings, and a correspondingly natural crossover operator is introduced. 'Homogeneous' refers to the fact that, in contrast to traditional GAs where pairs of genes that are 'close' have better chances of being preserved under crossover, there is no notion of proximity between pairs of genes. Examples are provided that will demonstrate superior performance of these new GAs for some typical problems in which these difficulties arise.

Keywords: evolutionary computation; genetic algorithms; homogeneous genetic algorithms; covariance matrix adaptation-evolutionary strategies; the p-centre problem

2000 AMS Subject Classification: 68W20; 68T20; 90C15; 90C59; 90C27
CCS Category: G.3

1. Introduction

Evolutionary computation is a burgeoning field of artificial intelligence that models computation on natural biological models. Genetic algorithms (GAs) are prototypical evolutionary algorithms that were introduced in the 1970s by Holland [12] and are based on the ('survival of the fittest') theory of Charles Darwin. GAs are stochastic heuristic algorithms that can be applied to a great variety of discrete optimization problems. Elements of the feasible solution space are represented by strings in some alphabet (traditionally binary). A fitness function is introduced that reflects qualities of the optimal solution so that optimal solutions optimize the fitness function, and 'more optimal' feasible solutions have fitness values at least as good as those of 'less optimal' feasible solutions. Such heuristics have proven very useful for numerous applications where it is not possible to obtain (in real time) an optimal solution, and, in particular, where the space of feasible solutions is far too large to allow a brute-force solution.

In this paper we will motivate and develop a new type of GA that is designed to overcome some of the difficulties that can arise with traditional GAs. For simplicity, we will base our comparisons and

*Email: astanoyevitch@csudh.edu

ISSN 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/00207160801968770
http://www.informaworld.com

developments on binary encoded GAs; the generalization to larger alphabets can be done in several natural ways. The main difference between (traditional) GAs and the homogeneous GAs that we introduce here is in the encoding process and the naturally corresponding crossover operation. In each, the solution space is put into a one-to-one correspondence with binary sequences. With traditional GAs, the entire binary sequences make up the chromosomes of the populations that are to be mated and mutated in future generations. The ordering of the genes (slots of the binary string) of the chromosomes is crucial on its effects of the mating process, and in many applications the overall size of the chromosomes can be quite large in relation to what we call the 'active genes,' which are the genes corresponding to an allele (binary value) of 1, of typical desirable solution forms. Moreover, it is often desirable to only look at chromosomes for which the number of active genes stays within a prescribed range. Such properties are difficult to control with traditional GAs, and penalty functions are usually brought in to compensate, but this technique has had mixed results. Homogeneous GAs represent solution elements by their unordered sets of active genes, and the mating and mutation operators are defined using set-theoretic constructs rather than string operations.

Homogeneous GAs are developed in Section 2 and are compared with existing modifications of GAs that have appeared in the literature. In Section 3 we give some general guidelines on problems for which homogenous GAs can outperform traditional GAs. We also provide some examples of problems where traditional GAs are more appropriate than homogeneous GAs. In Section 4 we will provide experimental evidence of the claims made in Section 3 using instances of the three sample problems that were introduced therein. In order to make fair performance comparisons with traditional GAs, we perform detailed sensitivity analyses in the determination of the parameters used for each algorithm, and for each separate problem considered. The final Section 5 provides some concluding observations. Some good general references on evolutionary computation include the recent books by De Jong [5] and by Fogel [7]. For GAs in particular, Goldberg's book [9] has become a standard reference; the book by Mitchell [15] includes many interesting applications and references. We mention also the book by Michalewicz [13], which provides a nice survey of GAs along with some general developments of alternatives to traditional GAs.

2. Homogeneous genetic algorithms

For a given discrete optimization problem, in a *homogeneous GA*, elements of the feasible solution set are represented by their corresponding (unordered) set of *active genes*. The solution space must first be represented by binary sequences in some predetermined way. For example, in a homogeneous GA, the chromosome that represents the following binary string:

$$0001000000100000000000000000000000000000000000000000$$

would be {4, 11} since only the 4th and 11th genes are active in this binary string.

We will define a crossover operator that will be intrinsically tied with this structure of chromosomes. Mating selection and mutation operators have a great variety for traditional GAs, and most all can be easily modified to work for homogeneous GAs. The key difference will be that the crossover operation will naturally preserve any desired inequalities in the number of active genes in the populations. In a traditional GA, genes are ordered in an encoding process. The resulting ordering gives rise to a metric of distances between pairs of genes in a chromosome. These distances have a substantial effect on whether allele values of pairs of genes are preserved under crossover, but in numerous applications, such facts are not taken into consideration in the encoding process. The adjective 'homogeneous' refers to the fact that there is no notion of distance

between genes in our set-theoretic (order free) encoding. Thus, relationships between alleles of pairs of genes are more likely to be preserved by crossover solely if such pairs are prevalent in chromosomes with higher fitness levels. Apart from these key differences, in our comparisons with traditional GAs, we will be using the same fitness functions, analogous mating and mutation operators, and perform separate sensitivity analyses to determine the parameters that will be employed. The aim is to produce experiments that result in fair comparisons. To specify any GA (homogeneous or traditional) after establishing the encoding scheme and an appropriate fitness function, we need to specify the three main components that are used (in order) to go from one generation to the next: (1) The mating selection scheme, (2) The crossover scheme, and (3) The mutation scheme. We now explain each of these three components and how they will be implemented in this paper.

2.1. *Mating selection*

Since mating selection schemes depend only on fitness values of members of a given population generation, all such schemes can be used interchangeably with homogeneous and traditional GAs. In our schemes, we will use fitness proportionate selection with fitness scaling (see [9, p. 76ff]). Moreover, in passing from each generation to the next, we clone a member of the population of highest fitness value, and another of second highest fitness value to automatically appear in the next generation (or lowest values for a minimization problem) without being subjected to the mutation operator. This cloning is sometimes known as *elitism*, and will guarantee that the maximum fitness levels of successive generations cannot decrease.

2.2. *Crossover operator for mating*

In a homogeneous GA, once the mating pool is formed, parents are randomly paired off to mate using the following crossover scheme that we call *random mixing crossover*: Suppose that a mating pair of parents is represented by the following sets of active genes: $x = \{a_1, a_2, \ldots, a_n\}$ and $x' = \{a'_1, a'_2, \ldots, a'_{n'}\}$, where a_i, a'_j are the active genes. We let $\text{Int} = x \cap x'$ denote the set-theoretic intersection of the active genes of the parents, and we let t denote the number of elements of Int. If either parent coincides with this intersection (i.e., if either $n = t$ or $n' = t$), then the two offspring are identical with the parents (i.e., the parents are cloned into the next generation, but subject to the mutation operator). In all other cases, we randomly choose an integer $j \in \{1, 2, \ldots, \min(n, n') - t\}$, and then randomly choose two subsets of size j from the two parent chromosomes that lie outside of their common intersection: $z \subset x \sim \text{Int}$ and $z' \subset x' \sim \text{Int}$. (We are using the tilde '$\sim$' to denote relative complements of sets.) The two offspring are then defined by swapping the genes of these two subsets in the two parents. In set-theoretic notation, the offspring can thus be expressed as $y = (x \sim z) \cup z'$, and $y' = (x' \sim z') \cup z$. See Figure 1. For traditional GAs, the corresponding natural (and widely used) single-point crossover scheme is employed, see [9,13,15] for details.

2.3. *Mutation operator*

For homogeneous GAs, throughout this paper we will be using a natural mutation scheme that we refer to as *random pool mutation*. The chromosomes of the new offspring (with the two that were cloned being excepted) are first randomly selected to undergo mutations with a specified probability p_{select}. For a selected chromosome z, a random number of genes j to be mutated is selected from the range $\{1, 2, \ldots, \min(\text{size}(z), \text{size}(A \sim z))\}$. Here, A is simply the set of all available genes (available for all feasible solutions to the optimization problem) and we use

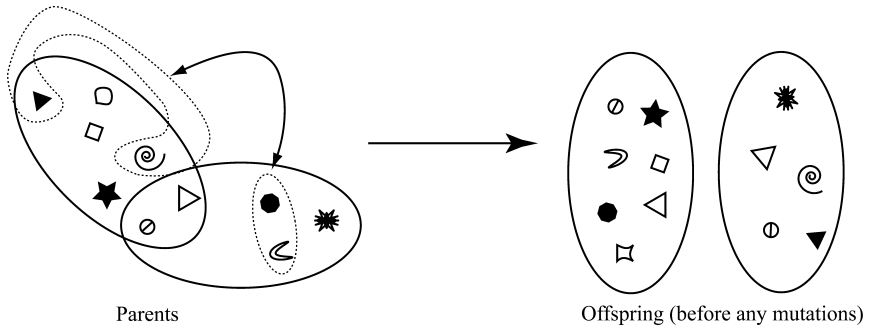

Figure 1. Illustration of the random mixing crossover operator for homogeneous GAs.

'size(S)' to mean the size/cardinality of a set S. Then a random set of j genes is selected from z and is exchanged with a randomly selected set of j genes from $A \sim z$.

Recall that the random mixing crossover operator always produces a pair of offspring whose chromosome sets have the same sizes (numbers of active genes) as those of the parents. Observe that the mutation operator naturally preserves the number of active genes in a chromosome. These properties, are in stark contrast with corresponding operators for traditional GAs, and are useful for an assortment of applications. For traditional GAs, sets of genes (of the same expected size as for homogenous GAs) are selected and flipped, i.e., alleles of selected genes of a selected chromosome whose values were 0 get changed to 1, and vice-a-versa.

When we make performance comparisons with traditional GAs, we will aim to use the same parameters to the greatest extent possible. Moreover, in two of the three applications that we give where both types of GAs will be compared (each dealing with an NP hard problem), we will use fitness functions that were designed and used in the literature for traditional GAs on the corresponding problems.

We point out that the traditional GAs and homogeneous GAs that we use here will both satisfy the following general convergence theorem for evolutionary algorithms (see [21, Theorem 2.1]) that include selection, crossover, and mutation procedures:

THEOREM *Suppose that in a* (*stochastic*) *evolutionary algorithm the following properties hold*:

(i) *The probabilities that any member of a certain generation population gets selected for mating or appears in the premutated offspring are both fixed positive numbers.*
(ii) *For any two feasible solutions x and y, there is a path of feasible solutions $x_1 = x, x_2, \ldots, x_n = y$, such that the probability that x_i can be mutated to $x_{i+1} (1 \leq i < n)$ is greater than a fixed positive number.*

Then the evolutionary algorithm will eventually converge to an optimal solution, with probability 1.

This theorem, by itself, is important, but not very practical (a random search algorithm would have the same conclusion), the key of any successful heuristic is to speed up this convergence to feasible solutions of decent quality well beyond that of a random (unintelligent) search.

2.4. *Comparisons with other modifications of genetic algorithms in the literature*

Since they first appeared in the 1970s, several variations of traditional GAs have been developed in the literature to mitigate an assortment of shortcomings that traditional GAs have encountered

when applied certain types of problems. We will briefly summarize and contrast some of these in this subsection that share some common features of homogenous GA. The underlying theme in many such variants is to move the GA closer to the problem class(es) at hand. For problems in continuous domains, floating point (real number) representations have been used in GAs in place of binary (or n-ary) encoding. There are numerous different crossover operators that have been used for *floating point GAs*, and the encoding scheme naturally allows 'close' solution components to be encoded by close chromosomes (more likely to be preserved, if of high fitness level). As with our traditional GAs, such floating-point representations can also provide great economy on the sizes needed to represent many continuous problems with many variables; the data structures and operations, however, are not so conveniently described in the language of discrete structures. For more details see [9; 13, Chapter 5].

Grouping GAs, developed by Falkenauer in the early 1990s (see [6]), are an interesting variant that are particularly suitable for problems that require the sorting of a collection of objects into groups in a certain way as to minimize some quantity. The prototypical example of such a problem is the *bin packing problem*, where a given collection of objects of various integer weights is to be placed in a minimum number of bins, each having the same (integer) capacity. Grouping GAs encode solutions with augmented strings: the first part corresponds to the objects, and the latter part to the groups. Unlike our homogeneous GAs, the order of the genes matters in a chromosome, but what is crucial is that only the ordering of the portion that encodes the groups is relevant. Thus, this algorithm focuses on the groups rather than on the objects themselves. The genetic operators apply only to the group portion of the chromosome, the object portion is used simply to look up which group an object is associated with. Another difference with homogeneous GAs is that in grouping GAs the length of a chromosome can vary (with the number of groups represented). Grouping GAs have been shown to produce very impressive results for an assortment of grouping problems. For an engaging development of grouping GAs along with some nice applications, see [6]; see also [3] for a more recent application involving cell formation.

As a final example, we mention a recent breed of GAs that has been developed to serve as effective heuristics for time series segmentation problems. They were developed by Chung *et al.* [4] with the motivation of the following time series problem: given a time series presented as a real-valued function from an interval of positive integers, say $D = \{1, 2, 3, \ldots, 100\}$ (this is the given time series), and a desired segment length K, say $K = 8$, the object is to find an increasing subsequence in D of length 8, so that the piecewise linear time (subtime) series corresponding to the selected length K subsequence of the given function 'best represents' it, among all subtime series having length K. The 'best representation' is quantified using a least squares type metric. The authors use exactly the size K subsequences to be the chromosomes. Thus, this representation also shares the economy that is enjoyed with homogeneous GAs. However, in this time series representation, the order of the genes is important (so it is not a homogeneous representation); moreover, the crossover operation that they use is a natural analogue of the single point crossover (whereby a single integer in D is randomly selected as the splice location). The authors have obtained some encouraging results with this approach when applied to time series for certain Asian stock prices.

3. When should homogeneous genetic algorithms be used?

3.1. *Problems where the number of active genes needs to be held constant or in some specified range*

In traditional GAs, crossover and mutation operators typically leave no control on the numbers of active genes in a chromosome. Rather than try to 'repair' these defective chromosomes, what is

usually done in practice is to modify the fitness function to include a *penalty function* to detract fitness value for such defects. The penalty should be graded so as to assess a greater penalty to chromosomes that are further away from being admissible candidates for the problem. These penalty function methods have had mixed results. See, for example [14,19,22] for more details on this method. In a homogenous GA, the natural crossover and mutation operators automatically preserve the numbers of active genes, so for problems where this is important, homogeneous GAs are a natural choice. The examples that we introduce in this section rely on the terminology of graph theory. For background and definitions we refer the reader to the book by West [25].

A prototypical problem where maintaining the size of active genes is important is the so called p-centre problem:

The p-centre problem. Suppose that we are given a connected graph with E weighted edges and n weighted vertices V, and a positive integer $p < n$. For any subset $C \subset V$, and vertex $v \in V$, we define the *distance* from v to C, $\mathrm{dist}(v, C)$, to be the minimum sum of the weights of the edges over all paths that join v to a vertex C. The problem is to find a subset C of V of size p for which the maximum weighted distance $\max_{v \in V} \mathrm{dist}(v, C) \cdot \mathrm{wgt}(v) \equiv \rho(C)$, the so-called *p-radius*, is as small as possible.

This problem is important for applications, for example, in facility location problems. It has been shown to be NP-complete (see [8, pp. 219–220]). For the 200 node network problem with $p = 30$ that we will use for our experiments, a direct approach would be of complexity $\binom{200}{30} \cdot 200^2 \approx 10^{40}$. There does not yet exist a very efficient general algorithm for the p-centre problem. Asymptotically, the fastest exact algorithm for this problem seems to be the one found by Tamir [23], which has complexity $O(E^p n^p \log^2 n)$. But for our 200 node network problem this would work out to an astounding complexity estimate that exceeds 10^{160}, making a direct (brute-force) approach more efficient. In the natural binary encoding for this problem, the number of available genes will simply be the set of n vertices. The corresponding chromosomes in a traditional GA will be binary strings of length n, while in a homogeneous GA, the chromosomes will be the subsets of the vertices (subsets of $\{1, 2, \ldots, n\}$, if the vertices are labelled in the natural way) of size p. The experimental evidence of the next section will show that homogeneous GAs perform significantly better than their traditional counterparts on this problem, even using a fitness function that was designed for the latter (from the literature).

3.2. *Problems where the number of active genes is much smaller than the number of available genes*

In many search problems amounting to the choice of a subset of a predetermined size (or range of sizes) from a significantly larger set, the chromosomes of a traditional GA would take up an inordinate amount of storage, sometimes to the point of making the algorithm unusable for the problem. In a homogenous GA, the chromosomes simply consist of the set of selected elements, and these much smaller storage requirements (that are amplified by the population size of the GA) can result in a much more efficient algorithm. The advantage of Section 3.1 can also come into play here.

The example that we give here is a problem in network logistics. Although it is of polynomial complexity, the exponent grows fast enough (with one of the main problem parameters) to make exact algorithms unfeasible for the experiments that we will perform in the following section.

A network logistics problem. For an existing network consisting of a (strongly) connected directed graph, the *diameter* of the resulting network is the largest number of edges that we

would actually need to use to travel from any node to any other node. The problem asks to add a certain number, k, of new edges to the network so as to reduce the diameter as much as possible.

For concreteness, we view the network of the problem as the available flights that a shipping company has between different cities. If there are N cities, this network can be represented by an $N \times N$ *incidence matrix* A, each of whose entries is 0 or 1: $a_{ij} = 1$ if, and only if there is a flight (directed edge in the network) from city $\#i$ to city $\#j$. The relevant (and interesting) fact about the incidence matrix is the following: if m is a positive integer, and $B = A^m$, then b_{ij} is the number of different routes that start at city $\#i$, end at city $\#j$, and use exactly m flights. (This fact is not too difficult to prove using the definition of matrix multiplication.) We can extend this interpretation to $m = 0 (A^0 = I)$, since we can only go from a city to itself using 0 flights. From these facts, it in turn follows that the diameter of the network (recall, this is the most number of flights we would need to get from any city to any other city in the network) is simply the smallest exponent d for which the matrix sum $A^0 + A^1 + A^2 + \cdots + A^d$ has no zero entries. Using this fact to check each of the $\binom{N^2 - N - E}{k}$ different ways of adding k new flights to the network (corresponding to changing k nondiagonal 0 entries of the incidence matrix to 1s), leads to an algorithm of complexity $O(N^{4+2k})$, provided that both k and E are small compared with N^2. The problem on which we will work on in the next section will have $N = 200$, $E = 200$, and $k = 4$. The above exact algorithm would thus have an unfeasibly large complexity of 4.1×10^{27}. Since there are $N^2 - N - E = 39,600$ genes to choose from, in a traditional GA with a population size of 100, we would need a matrix with close to four million entries to store each generation of the population. With a homogenous GA, each chromosome would only consist of four numbers (that represent the flight locations in the matrix) so the population matrix would only require 400 entries of storage. In addition to this huge storage handicap, the homogenous GA would also suffer the problems of Section 3.1 (since the number of active genes should be $k = 4$ here). Taken together, these problems would cause the traditional GA to run unacceptably slow in comparison to both the homogeneous GA and random search and perform no better than a random search.

3.3. *Interactive usages for general optimization problems*

For many problems, the number of active genes in optimal or near optimal solutions may not be known *a priori*, but after a few experiments or runs of (either type of) GA, as more information is gained, constraints on the number of active genes can be added in future trials of a homogeneous GA, leading to improved results. We will demonstrate such methods on the following problem:

The maximum independent set problem. A subset S of the vertex set V of a graph G is called *independent* if no two vertices belonging to S are adjacent, i.e., joined by an edge in G. The problem is to find an independent set S inside V of maximum size.

This is a central problem in network applications and is NP complete (see [8, pp. 194–195]). The first exact algorithm to significantly outperform the brute-force approach was found by Tarjan and Trojanowski [24] in 1977 and had a complexity of $O(\text{size}(V)^{n/3})$. In 1999, Beigel [2] developed an improved algorithm with complexity $O(\text{size}(V)^{0.29n})$.

Initially, the first run of a homogeneous GA might use 1, and size (V) for the minimum and maximum sizes of active genes in the search for a maximum independent set, respectively. After viewing the results of subsequent runs, these limits can be adjusted to improve results. We will demonstrate this in the next section.

4. Experimental results

4.1. *The p-centre problem*

4.1.1. *The problem instance and the genetic algorithms*

We randomly generated a 200×200 incidence matrix A that had roughly 3.23% of its entries being nonzero and these entries were randomly generated positive integers from the set $\{1, 2, \ldots, 100\}$. (The diameter of the resulting network was 6). The 200 vertices were randomly assigned weights from the set $\{8, 9, \ldots, 13\}$. The problem we aim to solve is how to best locate $p = 30$ facilities among the $N = 200$ nodes of this network. These data, as well as the best feasible solution that we obtained can be downloaded from the following URL: `http://www.csudh.edu/math/astanoyevitch/research/HGA.html`

For the fitness function (in both the homogeneous and the traditional GA) we used $f(C) = 1/(1 + \rho(C))$ (where $\rho(C)$ is as in Section 3.1) if the size of the subset C (identified with the chromosome) is p, otherwise $f(C) = 0$. In the homogeneous GA, this latter possibility will never occur. This is the fitness function used by Nayeem and Pal [18]. Throughout this section, all fitness values are linearly scaled using Goldberg's method [9, p. 76ff] for the mating selection process. The initial population for the traditional GA was randomly generated using a binomial distribution with mean p. For the homogeneous GA, random subsets of size p were chosen from the N nodes. An experimental design procedure (outlined in the next subsection) was used to separately determine the parameters that were used in both GAs. The crossover and mutation operators for both of the GAs are as described in Section 2.

4.1.2. *A sensitivity analysis*

The determination of optimal, or at least good choices of parameters to use in an evolutionary algorithm, or more in a general stochastic optimization algorithm is an important issue. A basic task in this direction is to determine the relative effects that changing certain parameters has on the (expected) performance of the stochastic algorithm. Cost also needs to be taken into consideration. Some parameters, such as a fitness scaling parameter, have negligible effects on the cost of running an evolutionary algorithm, but others, such as the population size or the number of generations to run usually have significant effects – especially since fitness evaluations are often the most expensive parts of running a stochastic optimization algorithm. Since stochastic algorithms have only well-defined expected outcomes, whereas the results of individual runs can be clouded by noise, statistical methods falling under the general categories of *design of experiments*, or *response surface methodology* can be adapted to the setting of evolutionary algorithms. Good general references in this area include the book by Montgomery [16] and the book by Myers and Montgomery [17]. A nice outline of a design of experiment programme applied to an evolutionary algorithm can be found in the recent paper by Ridge and Kudenko [20]. One important reality here is that any decent procedure in this area will take a great deal of computer time to carry through. This is particularly true because of the noisy nature of stochastic optimization algorithms, and thus large samples are needed to acquire sufficiently representative data sets.

Initially, we create three response surface graphs both for the homogeneous GA and the traditional GA, using each of the following settings for the population size s, and the number of generations G: (one of these settings involves $sG = 20{,}000$ objective function evaluations, while the other two involve $sG = 40{,}000$ evaluations) (a) $s = 100$; $G = 200$, (b) $s = 100$; $G = 400$, and (c) $s = 200$; $G = 200$.

Apart from these two parameters, our GAs had two others: C_S, the fitness scaling parameter in Goldberg's linear scaling method, and p_{select}, the mutation selection probability. For each pair of

settings:

$$C_S \in \{1.20, 1.25, 1.30, 1.35, \ldots, 1.95, 2.0\}, \quad \text{and}$$
$$p_{\text{select}} \in \{0.025, 0.050, 0.075, 0.100, \ldots, 0.950, 0.975, 1.000\},$$

we ran each GA for 40 trials. We then plotted the resulting mean response surfaces. The results are shown in Figure 2 (for the homogeneous GA) and Figure 3 (for the traditional GA), except that we restricted the range of p_{select} to be in [0, 0.5], since it became obvious (after viewing the complete surfaces) that larger values of p_{select} worked poorly for both GAs. These response

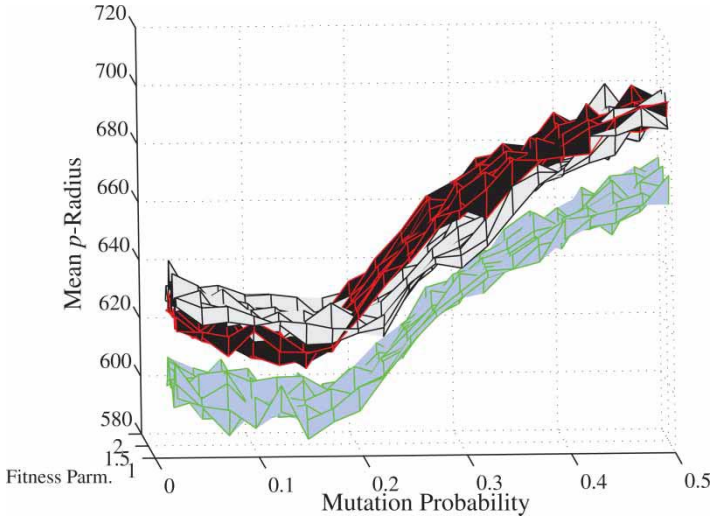

Figure 2. Mean response surfaces for the homogeneous GA to our 200 city p-centre problem instance resulting from 40 trials with the following settings: (a) $s = 100$; $G = 200$ (unshaded), (b) $s = 100$; $G = 400$ (medium shaded), and (c) $s = 200$; $G = 200$ (darkly shaded).

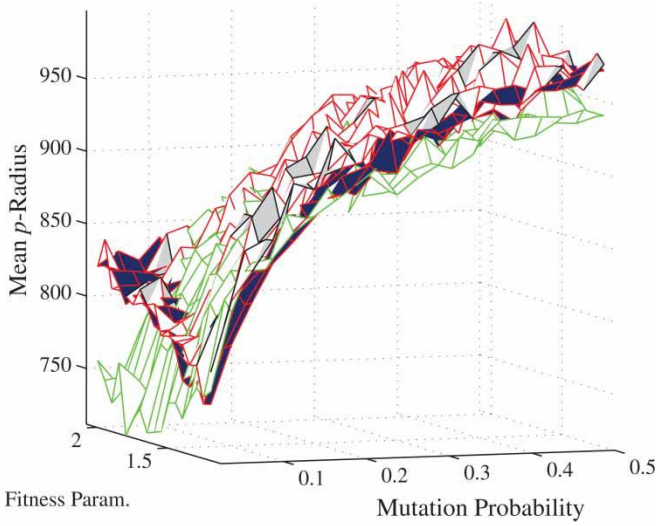

Figure 3. Mean response surfaces for the traditional GA to our 200 city p-centre problem instance resulting from 40 trials with the following settings: (a) $s = 100$; $G = 200$ (unshaded), (b) $s = 100$; $G = 400$ (medium shaded), and (c) $s = 200$; $G = 200$ (darkly shaded). Among other differences, the reader should notice the higher fitness values (p-radii) here compared with those of Figure 2.

surfaces are sufficiently smooth and quite informative. (We could have made the surfaces even less noisy by increasing the number of trials for each pair of parameter values, but this would be very expensive and the additional benefits would be minimal.) Notice in Figure 2 that the medium shaded response surface corresponding to the parameter settings (b) $s = 100$; $G = 400$ clearly lies below the other two. This shows that (for the problem at hand) the homogeneous GA is much more sensitive to a doubling of G than to a doubling of s. It is also interesting to observe that the darkly shaded surface (corresponding to (c) $s = 200$; $G = 200$) actually crosses above the unshaded surface (with a smaller value of sG); this is perhaps due to the fact that a high mutation probability is detrimental to the algorithm's performance, the benefits of a larger population are overshadowed by the increased number of suboptimal mutant chromosomes that get introduced.

Figure 3 shows that the mean response surfaces for the traditional GAs, apart from having significantly higher p-radius values than for the homogeneous GAs, all lie much closer together, but the unshaded one, corresponding to the parameter settings (c) $s = 200$; $G = 200$, clearly dips to the lowest values when the mutation probability is close to zero. We also notice that for both types of GA, the response surfaces are much less sensitive to the parameter C_S, than to p_{select}, thus it is more crucial (and simple) to find a good value for p_{select}. The fitness landscape in the direction of C_S is a lot less variable, and our surfaces had a lot more noise in this dimension.

With the values of s, and G determined using the above sensitivity analysis, we next applied the *covariance matrix adaptation-evolutionary strategies* (CMA-ES) method to search for good values of the remaining two parameters. This is a general stochastic optimization method for unconstrained optimization problems where the objective function does not have an easily obtained Hessian matrix (of second partial derivatives). The method has been extensively developed by Nikolaus Hansen, who maintains a website (`http://www.bionik.tu-berlin.de/user/niko/cmaesintro.html`) containing a tutorial, downloadable state-of-the-art programmes, as well as useful papers and references. Very roughly, the CMA-ES method proceeds iteratively like a Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method. Assuming that we do not have the required partial derivatives at our disposal to guide our search, the method performs a statistical sampling of points at each iteration. The objective values of these points are evaluated, and this information is used to update the estimate for the covariance matrix that will determine the Gaussian distribution from which to generate the next generation of points. For further details see either Hansen's above mentioned tutorial, or see [10,11].

The method has been tested on a great variety of problems, both pure and applied, and has performed remarkably well. It does not yet seem to have been applied to optimize parameters in GAs, but this seems like a very natural and opportune application. Initially, for this first application, we applied it (using Hansen's MATLAB program) to optimize only the two remaining parameters, and we found that it produced good results that checked well with the corresponding response surface graphs. In particular, for the above two problems, it produced values of p_{select} equalling 0.1137 (for the homogeneous GA with $s = 100$; $G = 400$), and 0.0589 (for the traditional GA with $s = 200$; $G = 200$), which fit well with the preceding response surface graphs. We will see in the next subsection that when we use these parameter settings (along with the less important values for C_S that were produced by CMA-ES), the performance of both GAs is quite impressive. In Section 4.3, we will use it in a more sophisticated fashion to optimize a set of five parameters. Again, we stress that although the results that get produced here are impressive, the word 'optimize' is not quite appropriate since CMA-ES, like any evolutionary algorithm, is a heuristic.

4.1.3. *Computational results*

Using the parameters determined by the (separate) sensitivity analyses for each GA, we ran the above GAs 100 times each and compared the results with completely random searches (fixing the correct size of the centre to be p) with the same number of fitness evaluations. The statistical

Table 1. Summary of p-centre experiments for 100 trials.

	Homogeneous GA	Traditional GA	Random search
Minimum	504	638	696
Mean	576.1	730.8	754.7
Standard deviation	27.8	46.2	19.6

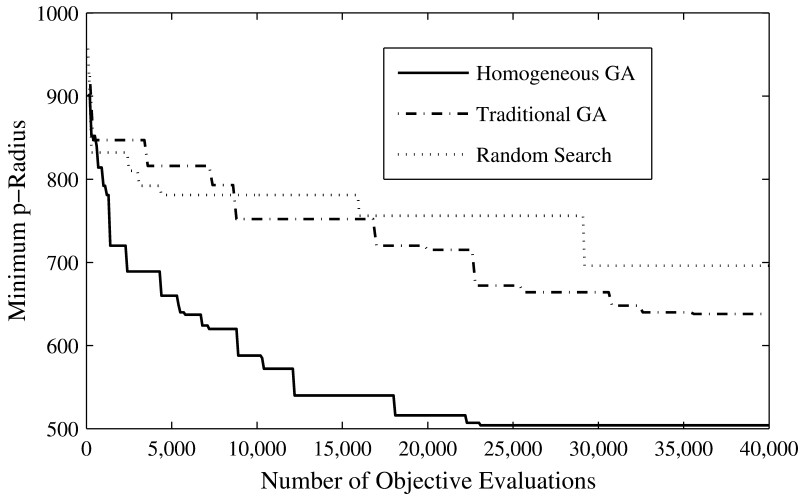

Figure 4. Comparison of the generational progression for the $N = 200$ node network 30-centre problem for the best of 100 trial runs. The significantly improved performance of the homogeneous GA is typical for such problems.

results are shown in Table 1, and Figure 4 shows the generational progression of the best of the 100 trials for each of the three algorithms.

From Table 1, we observe that over the 100 trials, the averages of the best solutions found by the traditional GA were 26.9% greater than those found by the homogenous GA, and actually only 3.1% less than the corresponding averages for the completely random search. In the best trial over the 100, the traditional GA outperformed the random search by (a modest) 8.3%, but the former's best value was 26.6% greater than the best overall solution found by the homogeneous GA.

4.2. *Network logistics problem*

Our illustrative example will concern a rather large but quite simple cycle network with $N = 200$ nodes labelled from 1 to 200, and 200 directed edges: $[i \quad i+1](1 \leq i < 200)$, and the edge [200 1]. As explained in Section 3.2, traditional GAs are impractical for this problem, so we compare only the more suitable homogeneous GA with a random search. For the fitness function we simply take the resulting reduction in diameter for the new network (with $k = 4$ flights added) from the diameter of the original network, which is clearly 199. We let the homogeneous GA run for 20 generations, and use a population of size 100. Since we will not be comparing with a traditional GA, and since fitness evaluations are so costly for this problem, we will forgo a detailed experimental design for parameter selection of the homogeneous GA, and just use some reasonable (after some small experiments), albeit somewhat arbitrary parameters: $p_{\text{select}} = 0.03$, and $C_S = 1.6$. For this problem the fitness evaluations constitute, by far, the most expensive part of the algorithm. The results of a typical run of both programmes are shown in Figure 5.

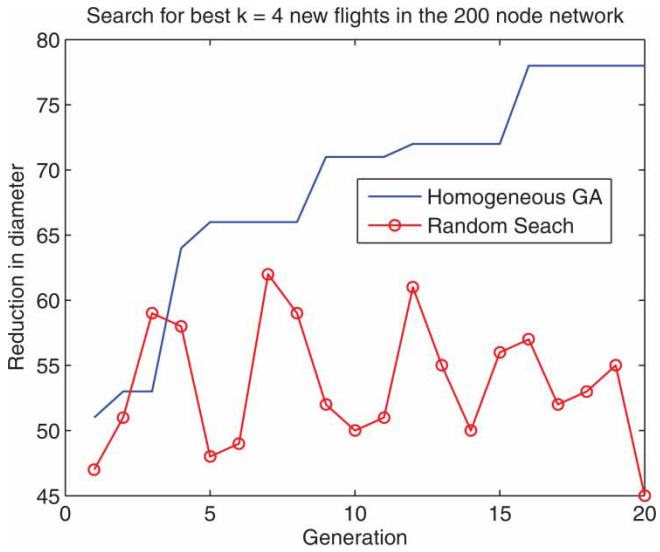

Figure 5. Comparison for typical runs of the homogeneous GA and the random search's best feasible solution (100 fitness evaluation per generation) for the 200 city network routing problem.

4.3. *The maximum independent set problem*

For our experiments, we will use the scalable graph shown in Figure 6. We will work with this graph when $k = 50$ (so it has 150 vertices). Note that maximal independent sets (of size 50) are not unique, since there are $3 \cdot 2^{49} \approx 2 \cdot 10^{15}$ (*Proof*: Each of the 50 complete 3-graphs making up T_{150} must contain exactly one vertex of a maximum independent set, once one is specified from the outer triangle, there will be only two choices for each subsequent triangle as we progress inward.) The probability of randomly choosing a maximum independent set (even if we knew to look for size 50 sets of which there are $\binom{150}{50} \approx 2 \cdot 10^{40}$) would only be about 10^{-25}.

For this problem, we use the same fitness function that was used by Bäck and Khuri [1] in their (traditional) GA investigation of this problem: $f(x) = \sum_{i=1}^{N} \left(x_i - N \cdot x_i \cdot \sum_{j=i}^{N} x_j e_{ij} \right)$, where x is the binary representation of the chromosome, N is the number of genes (so $N = 150$ here), and $e_{ij} = 1$ if and only if the graph contains an edge connecting vertex i and vertex j, otherwise $e_{ij} = 0$. This is simply the cardinality of the subset of vertices represented by x less a penalty function that takes away N units for each edge connecting two vertices in the set. Of course, for the homogenous GA, the notation for this function would be different. The initial populations were generated by independently selecting each gene with probability $1/2$ for the traditional GA. For the homogeneous GA, random subsets were generated from the vertex set with sizes uniformly distributed (two different ranges were used, see below). In addition to the crossover and mutation operators described in Section 2, we also invoked a secondary mutation operator in both GAs, that works as follows: an offspring chromosome (after having passed through the original mutation operator) gets selected with probability p_{add} (a new parameter). Selected chromosomes will have one new randomly selected gene turned on (i.e., a new vertex will be added). Thus, both GAs now have the following five parameters with the indicated ranges: $s > 0$, $G > 0$(integers), $0 \leq p_{\text{select}}$, $p_{\text{add}} \leq 1$, $1.2 \leq C_S \leq 2.0$. The number of fitness evaluations with these parameters is simply sG, and this is proportional to the cost of running the GA (either type). To determine these parameters, we used the CMA-ES programme described in Section 4.1. We used the above constraints, along with s having to be even, and placed the additional cost constraint: $sG = 10,000$.

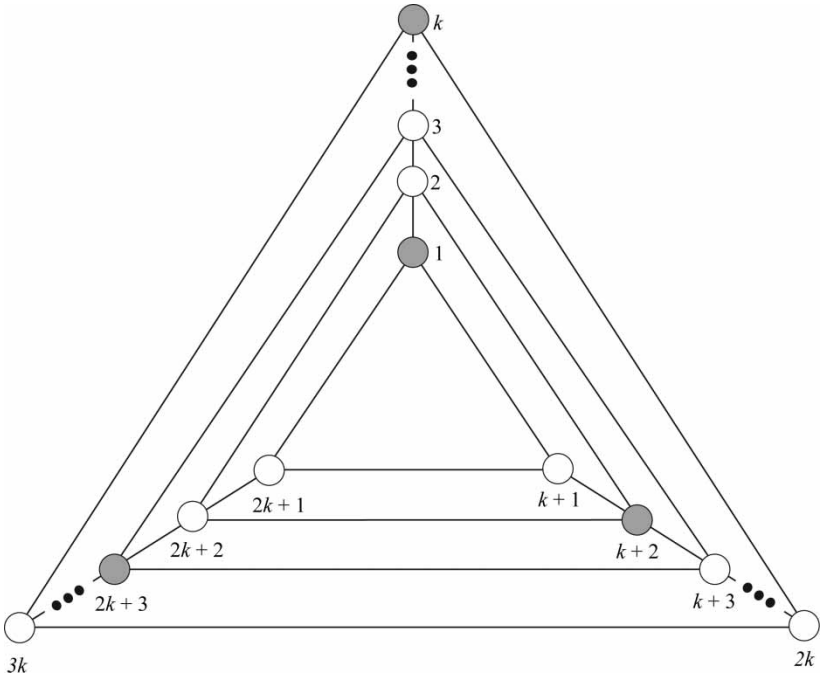

Figure 6. The scalable graph T_{3k} having $3k$ vertices and a nonunique maximal independent set of size k (shaded vertices).

We used the equality constraint since separately increasing either s or G can only (statistically) improve performance. The parameter settings found by the CMA-ES programme were quite enlightening: all findings tended to significantly favour a large number of generations with a small population and a high level of experimentation (through mutations). The parameters that were found were as follows: for (i) the traditional GA: $s = 14$, (and so $G = \text{round}(10,000/s) = 714$), $p_{\text{select}} = 0.226$, $p_{\text{add}} = 0.053$, and $C_S = 1.339$; (ii) the homogeneous GA with default values the minimum and maximum chromosome size: 1, and 150, respectively: $s = 14$, (so $G = 714$), $p_{\text{select}} = 0.0022$, $p_{\text{add}} = 0.460$, and $C_S = 1.87$; and (iii) the homogeneous GA with values of the minimum and maximum chromosome size being set to 10 and 60: $s = 10$, (so $G = 1000$), $p_{\text{select}} = 0.463$, $p_{\text{add}} = 0.672$, and $C_S = 1.320$. Hansen's programme also provides much useful statistical feedback that allowed us to see CMA-ES nicely converging to its solutions. As in Section 4.1, the GAs were less sensitive to the parameter C_S. We point out that the CMA-ES programme took between 900 and 3200 runs for each GA to reach its results. Multiple runs have shown close agreement with the outputted parameter sets. Comparing this with our response surfaces of Section 4.1, each surface took 13,600 runs to generate, and there we were working only in a two variable setting compared with four variables here. Also, in the response surfaces, we only sampled from a fixed discrete set of parameter values, so even if we greatly increased the sample size for each grid point, the precision would be limited to the grid size. In summary, an intelligent direct search such as CMA-ES can lead to a much faster and more accurate parameter determination.

Using the above parameter sets, we ran 100 trials each of (i) the traditional GA, (ii) the homogeneous GA with default values the minimum and maximum chromosome size: 1, and 150, respectively, (iii) the homogeneous GA with values of the minimum and maximum chromosome size being reset to 10 and 60, and (iv) a random search with the same number (10,000) of fitness evaluations. The distribution of the optimal values found with each of these methods over the 100 runs is shown in Table 2.

Table 2. Distribution of sizes of largest independent sets found in 100 trials with four different algorithms.

	Algorithm used to search for maximal independent set (optimal size: 50)			
	Traditional GA	Homogeneous GA Ver1	Homogeneous GA Ver2	Random search
Maximum	49	50	50	25
Mean	43.9	47.4	49.6	19.2
Standard deviation	1.8	1.4	0.9	2.1
Number $> 0^{\dagger}$	47	56	50	100

The number of function evaluations for each algorithm was the same: 10,000.
†For each version, this entry represents the total number of trials (out of 100) which resulted in positive fitness value (i.e., for which the algorithm found an independent set). The mean in the table is calculated using the data from these trials.

We see that the basic version of the homogenous GA outperforms the traditional GA for this problem, and that by adjusting the admissible ranges of the numbers of active genes, significant further improvements were realized (indeed, finding a true optimal solution) by the homogeneous GA.

5. Concluding remarks

Binary strings are a very natural data structure on most all computing platforms, and the string operations such as crossover and bitwise mutation that are at the core of traditional GAs are relatively simple and inexpensive operations to perform. The set-theoretic operations, such as intersections, and set differences that are employed in homogeneous GAs come a greater expense, so with all other things being equal this additional overhead can potentially result in slower runtimes of homogeneous GAs compared with traditional GAs. In many applications, however, the evaluations of the fitness functions are (by far) the most expensive part of running (either type of) a GA, so this set-theoretic overhead becomes a moot issue. More importantly, we have presented two general problem characteristics and a general strategy for search problems in which homogenous GAs can significantly outperform their traditional GAs counterparts. Our goal in these experiments was to accurately represent such advantages. Careful experimental designs were used to fairly determine parameters for each type of GA in each problem instance, and in fact we used fitness functions that were developed for traditional GAs. Due to their simplicity, added control of chromosome size ranges, and improved performance over traditional GAs in several important classes of problems shown above, homogeneous GAs appear to have potential for future applications in evolutionary computation.

Acknowledgements

The author would like to thank the two anonymous referees for their careful readings that have led to their helpful comments and suggestions, as well as some interesting references. He would also like to thank Thomas Bartz-Beielstein for introducing him to the CMA-ES method of stochastic optimization, and its inventor, Nikolaus Hansen for some helpful communications he had with the author when the latter was employing this method to assist in finding decent values for the parameters for the GAs used in this paper.

References

[1] T. Bäck and S. Khuri, *An evolutionary heuristic for the maximum independent set problem*, Proceedings of the IEEE World Congress on Computational Intelligence, vol. 2, IEEE Service Center, Piscataway, NJ, 1994, pp. 531–535.
[2] R. Beigel, *Finding maximum independent sets in sparse and general graphs*, Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, January 1999, SIAM Publishing, Philadelphia, PA, 1999.

[3] E.C. Brown and R.T. Sumichrast, *CF-GGA: a grouping genetic algorithm for the cell formation problem*, Int. J. Prod. Res. 39(16) (2001), pp. 3651–3669.
[4] F.L. Chung, T.-C. Fu, V. Ng, and R.W.P. Luk, *An evolutionary approach to pattern-based time series segmentation*, IEEE Trans. Evol. Comput. 8 (2004), pp. 471–489.
[5] K.A. De Jong, *Evolutionary Algorithms, A Unified Approach*, MIT Press, Cambridge, MA, 2006.
[6] E. Falkenauer, *The grouping genetic algorithms: widening the scope of the GAs*, Belg. J. Oper. Res. Stat. Compu. Sci. 33 (1992), pp. 79–102.
[7] D.B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 3rd Ed.*, John Wiley & Sons, IEEE Press, Hoboken, NJ, 2005.
[8] M.R. Garey and D.S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, 1975.
[9] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA, 1989.
[10] N. Hansen and A. Ostermeier, *Completely derandomized self-adaptation in evolution strategies*, Evol. Comput. 9(2) (2001), pp. 159–195.
[11] N. Hansen, S.D. Müller, and P. Koumoutsakos, *Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)*, Evol. Comput. 11(1) (2003), pp. 1–18.
[12] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.
[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed., Springer-Verlag, New York, 1994.
[14] K. Miettinen, M.M. Mäkelä, and J. Toivanen, *Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms*, J. Glob. Optim. 27 (2003), pp. 427–446.
[15] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.
[16] D.C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons Inc., New York, 2005.
[17] R.H. Myers and D.C. Montgomery, *Response Surface Methodology, Process and Product Optimization Using Designed Experiments*, 2nd ed., John Wiley & Sons Inc., New York, 2002.
[18] S.M.A. Nayeem and M. Pal, *A genetic algorithm to solve the p-center and p-radius problem on a network*, Int. J. Comput. Math. 82(5) (2005), pp. 541–550.
[19] J.M. Richardson, M.R. Palmer, G.E. Liepins, and M.R. Hilliard, *Some guidelines for genetic algorithms with penalty functions*, Proceedings of the Third International Conference on Genetic Algorithms, J.D. Schaffer, ed., Morgan Kaufmann Publishers, San Mateo, CA, 1989.
[20] E. Ridge and D. Kudenko, *Analyzing heuristic performance with response surface models: prediction, optimization, and robustness*, Proceedings of the 2007 Genetic and Evolutionary Computation Conference held in London, UK, Association of Computing Machinery (ACM), New York, 2007, pp. 150–157.
[21] G. Rudolf, *Finite Markov chain results in evolutionary computation, a tour d'horizon*, Fund. Inform. 35 (1988), pp. 67–89.
[22] A.E. Smith and D.M. Tate, *Genetic optimization using a penalty function*, Proceedings of the 5th International Conference on Genetic Algorithms, S. Forrest, ed., Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[23] A. Tamir, *Improved complexity bounds for the center location problems on networks by using dynamic data structures*, SIAM J. Discrete Math. 1 (1988), pp. 377–396.
[24] R.E. Tarjan and A.E. Trojanowski, *Finding a maximum independent set*, SIAM J. Comput. 6 (1977), pp. 537–546.
[25] D.B. West, *Introduction to Graph Theory*, 2nd ed., Prentice-Hall, Upper Saddle River, NJ, 2001.