

# Graph Based Evolutionary Algorithms

Kenneth M. Bryden  
Mechanical Engineering Department  
Iowa State University  
Ames, Iowa 50011  
kmbryden@iastate.edu

Steven Corns  
Mechanical Engineering Department  
Iowa State University  
Ames, Iowa 50011  
scorns@iastate.edu

Daniel A. Ashlock  
Mathematics Department  
Iowa State University  
Ames, Iowa 50011  
danwell@iastate.edu

Stephen J. Willson  
Mathematics Department  
Iowa State University  
Ames, Iowa 50011  
swillson@iastate.edu

## Abstract

Evolutionary algorithms use crossover to blend pairs of solutions to a problem in hopes of creating novel solutions. The best of these are retained by selection. Hopefully, crossover takes distinct good features from each of the two structures involved. This creates a conflict: progress results from crossing over structures with different features. Crossover, however, is part of a process that produces new structures that are like their parents and so reduces the diversity on which successful crossover depends. As evolution continues from generation to generation, the part of the search space currently occupied becomes limited. Mutation can help maintain diversity but is not a panacea for diversity loss. In this paper we describe and test evolutionary algorithms that use a combinatorial graph to limit the choice of crossover partners. These graphs act to limit the speed and manner in which information can spread, giving competing solutions time to mature. This use of graphs forms a computationally inexpensive method of picking a global level of trade-off between exploration and exploitation. The results of testing 26 graphs with a diverse collection of graphical properties are presented. The test problems used are one-max, DeJong's functions, the Griewangk function in 3-7 dimensions, a function based on self avoiding random walks in 9, 12, 16, 20, 25, 30, and 36 dimensions, the plus-one-recall-store problem with  $n=15, 16,$  and 17, location of length 6 1-error-correcting DNA barcodes, and the semi-symbolic solution of a simple differential equation. The choice of combinatorial graph has a significant effect on the time to solution. In the cases studied the optimum choice of graph impacted solution time as much as 63-fold with typical impact being in the range of 15% to 100% variation in time to solution. The graph yielding superior performance is found to be problem dependent. In general the optimum graph diameter increases

and the optimum average degree decreases with the complexity and difficulty of the fitness landscape. The use of diverse graphs as population structures for a collection of problems also permits a classification of the problems. A phylogenetic analysis of the problems is performed. This analysis uses normalized time to solution on each graph as the characters for phylogenetic classification. The resulting classification tree groups the numerical problems as a clade together with one-max. Self avoiding walks form a clade with the single example of semi-symbolic differential equation solution, albeit a looser one than the numerical problems. The PORS and Bar Code problems form a super-clade with the numerical problems but are substantially distinct from them. This novel form of analysis has potential to improve the choice of problems within a test suite.

## 1 Introduction

In nature, constraints such as geography, mutual infertility, or partner selection mechanisms are imposed on a individual's ability to reproduce sexually with other individuals. In the Simple Genetic Algorithm (SGA) [11] the only constraint on reproduction is that more fit individuals have a higher probability of being selected to participate. In nature individuals who are separated by great distances, no matter what their respective fitnesses, have a very low probability of reproducing with each other. Within many species one also finds cultural or behavioral constraints on the probability of two individuals reproducing. Birds have complex mating dances that help to identify good partners, frogs use distinctive calls for the same purpose, insects employ pheromones, and human partner selection techniques are complex and variable. Any widespread biological phenomenon that appears over and over in populations subject to natural selection must convey a selective advantage. Limiting mate choice is intuitively a good thing in an evolutionary algorithm because it can prevent what is called premature convergence. In this paper we impose a class of geography-like constraints on evolutionary algorithms for a selection of test problems and assess the results.

One of the standard issues in population genetics is explaining why there are not greater problems with loss of diversity in natural populations even though simple mathematical models show that diversity should vanish rapidly. Sewall Wright has his theory of isolation by distance [34]. Kimura and Crow [15] examine the rate at which different graphical population structures lose their genetic diversity under simple reproduction without selection. Analogously, one of the fundamental problems in evolutionary algorithms is maintaining useful diversity in the population as the algorithm progresses. When reproduction happens, individuals in the population are replaced by individuals with parts copied from a stochastically restricted subset of the population, and so diversity loss is acute if not carefully managed. Currently, the primary tool for such management is setting the rate of application of mutation operators. Imposing geography on the algorithm is another management tool for diversity loss. Implemented properly such geography can have a very low runtime cost.

Various approaches to managing diversity loss appear in the literature. These include using a high mutation rate, reducing the fitness of organisms in proportion to the number of organisms representing similar solutions (niche specialization), directly rejecting duplicate solutions (e.g. taboo search). These methods suffer to some degree from requiring the

ability to compute whether creatures are similar. With a plain string representation in which each character has meaning, this is easy. More complex representations such as finite state machines [10], Genetic Programming (with its potential for bloat) [6], or GP-Automata [2], all of which permit a single solution to have multiple encodings, render this kind of distance computation challenging. An example of this type of distance computation appears in [28], in which the authors make diversity part of a multi-objective optimizer.

Another approach is to impose a geography upon the population. In [1] a population is placed on each processor of a multiprocessor machine with occasional migration, adopting the connection topology native to the machine. In [20] a version of Charles Darwin’s ideas about the origin of diversity on islands and its later winnowing on continents appears. One of a large series of investigations by Darryl Whitley[32] explores island model algorithms. Distinct populations are placed on islands and migration rates and populations sizes are tuned with resulting performance enhancement at least partially attributable to the geographic preservation of diversity.

Except possibly for raising the mutation rate, imposing a geography is the least intensive of the extant diversity preservation techniques. The geography is selected as part of the algorithm design and need not impact runtime significantly. This study explores this last notion by imposing various geographical structures coded as combinatorial graphs on a type of evolutionary algorithm. A small, initial study in this area appears in [3]. An application of graph based evolutionary algorithms to the problem of optimizing the design of a wood burning stove appears in [8]. In Section 2 we give the background mathematical definitions. In Section 3 we define graph based genetic algorithms and describe the twenty-three test problems used. Section 4 gives the precise design of the experiments. Section 5 describes the outcomes of the experiments and discusses the results. Section 6 provides the taxonomic analysis of the results and discusses their significance. Section 7 draws overall conclusions for the study. Section 8 discusses what directions might be valuable for additional study.

## 2 Mathematical background

We assume some familiarity with graph theory [30]. A *combinatorial graph* or *graph*,  $G$ , is a collection  $V(G)$  of vertices and  $E(G)$  of edges where  $E(G)$  is a set of unordered pairs from  $V(G)$ . Two distinct vertices of the graph are *neighbors* if they are members of the same edge. The number of edges containing a vertex is the *degree* of that vertex. If all vertices in a graph have the same degree, the graph is said to be *regular*. If the common degree of a regular graph is  $k$ , then the graph is said to be  $k$ -regular. A graph is *connected* if one can go from any vertex to any other vertex by traversing a sequence of vertices and edges. The *diameter* of a graph is the largest number of edges in a shortest path between any two of the vertices. The diameter is, in some sense, the shortest path across the graph.

In this paper a graph used to constrain mating in a population will be called the *population structure*. The general strategy is to use the graph to specify the geography on which a population lives, permitting mating only between neighbors, and finding graphs that preserve diversity without hindering progress due to heterogeneous crossover.

This paper utilizes a nonstandard operation on graphs called *simplexification*. Simplexification at a vertex  $v$  replaces  $v$  with a cluster of vertices, one for each neighbor of  $v$  so that

all the new vertices are neighbors of one another and each is a neighbor of exactly one of  $v$ 's former neighbors. Simplexification of a vertex with four neighbors is shown in Figure 1. The effect of simplexification is to create small groups of vertices that are closely coupled to one another but less closely coupled to the rest of the graph. This creates an analog of a biological refuge in the graphical connection topology. By *simplexification* of a graph, we mean simultaneous simplexification of all the graph's vertices.

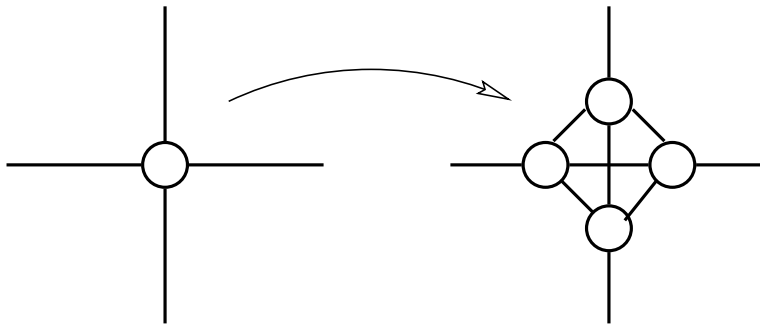


Figure 1: Simplexification of a vertex with four neighbors

## 2.1 List of graphs

In this section we give the list of combinatorial graphs used in this study, as well as defining graphs required to describe the graphs used.

**Definition 1** *The complete graph on  $n$  vertices, denoted  $K_n$ , has  $n$  vertices and all possible edges. An example of a complete graph is shown in Figure 2.*

**Definition 2** *The complete bipartite graph with  $n$  and  $m$  vertices, denoted  $K_{n,m}$ , has vertices divided into disjoint sets of  $n$  and  $m$  vertices and all possible edges that have one end in each of the two disjoint sets. The 3 – pre – Z graph shown in Figure 2 is the complete bipartite graph  $K_{4,4}$ .*

**Definition 3** *The  $n$ -cycle, denoted  $C_n$ , has vertex set  $\mathbb{Z}_n$ . Edges are pairs of vertices that differ by  $1 \pmod{n}$  so that the vertices form a ring with each vertex having two neighbors.*

**Definition 4** *The  $n$ -hypercube, denoted  $H_n$ , has the set of all  $n$  character binary strings as its set of vertices. Edges consist of pairs of strings that differ in exactly one position. A 4-hypercube is shown in Figure 2.*

**Definition 5** *The  $n \times m$ -torus, denoted  $T_{n,m}$ , has vertex set  $\mathbb{Z}_n \times \mathbb{Z}_m$ . Edges are pairs of vertices that differ either by  $1 \pmod{n}$  in their first coordinate or by  $1 \pmod{m}$  in their second coordinate but not both. These graphs are  $n \times m$  grids that wrap (as tori) at the edges. A  $12 \times 6$ -torus is shown in Figure 2.*

**Definition 6** The generalized Petersen graph with parameters  $n, k$ , with  $k$  relatively prime to  $n$ , is denoted  $P_{n,k}$  and has vertex set  $0, 1, \dots, 2n - 1$ . The vertices  $0 \dots n - 1$  are connected in a standard  $n$ -cycle. The vertices  $n \dots 2n - 1$  are also connected in an  $n$  cycle but adjacent vertices in this cycle are of the form  $i, (i + k); (\text{mod } n)$ . Finally, pairs of vertices  $i, n + i$  are also connected. The graph  $P_{32,5}$  is shown in Figure 2.

**Definition 7** A tree is a connected graph with no cycles. Degree zero or one vertices are termed leaves of the tree. A balanced regular tree of degree  $k$  is a tree constructed in the following manner. Begin with a single vertex. Attach  $k$  neighbors to that vertex and place these neighbors in a queue. Processing the queue in order, add  $k - 1$  neighbors to the vertex most recently removed from the queue and add these neighbors to the end of the queue. Continue in this fashion until the tree has the desired number of vertices. The resulting graph is a tree in which all non-leaves have degree  $k$  and which has, constructively, the smallest possible diameter among trees with all non-leaves having degree  $k$ . We denote these graphs  $RBT(n, k)$  where  $n$  is the number of vertices. Notice that not all  $n$  are possible for a given  $k$ .

**Definition 8** The graph  $Z$  is created by starting with  $K_{4,4}$  and then simplifying the entire graph three times. Two of the steps leading to the graph  $Z$  are shown in Figure 2.

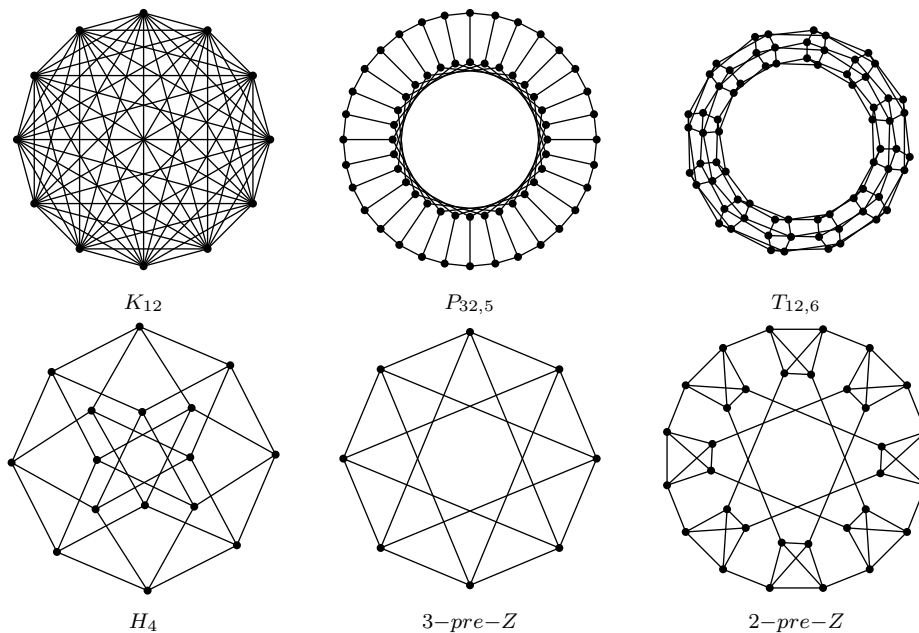


Figure 2: Examples of complete, Petersen, Torus, and hypercube graphs, and some of the steps leading to the  $Z$  graph. These examples are all smaller than the graphs actually used but are members of the same family of graphs.

In addition, four classes of *random* graphs are examined. A random graph is specified by a randomized algorithm which corresponds to a type of random graph (a probability distribution on some set of graphs). We use three instances of each class of random graph used.

**Definition 9** An edge move is performed as follows. Two edges  $\{a, b\}$  and  $\{c, d\}$  are found that have the property that none of  $\{a, c\}$ ,  $\{a, d\}$ ,  $\{b, c\}$ , or  $\{b, d\}$  are themselves edges. The edges  $\{a, b\}$  and  $\{c, d\}$  are deleted from the graph, and the edges  $\{a, c\}$  and  $\{b, d\}$  are added. Notice that edge moves preserve the regularity of a graph if it is regular.

**Definition 10** We generate random regular graphs by the following algorithm. Start with a regular graph and then repeatedly perform 3,000 edge moves on vertices selected uniformly at random from those that are valid for edge moves. For 3-regular random graphs use  $P_{256,1}$  as the starting point. For 4-regular random graphs use  $T_{16,32}$  as the starting point. For 9-regular random graphs use  $H_9$  as the starting point. These graphs are denoted  $R(n, k, i)$  where  $n$  is the number of vertices,  $k$  is the regular degree, and  $i = 1, 2, 3$  is the instance of the graph in this study.

**Definition 11** We generate random toroidal graphs as follows. A set of 512 points are randomly placed onto the unit torus (unit square wrapped at the edges), and edges are created between those at distance 0.07 or less from one another. This distance was chosen to give an average degree of about 6. After generation the graph was checked to see if it was connected. Graphs that were not connected were rejected so that the graphs used were chosen at random from the connected random toroidal graphs. These graphs are denoted  $R(r, i)$ , where  $r = 0.07$  is the radius for edge creation, and  $i = 1, 2, 3$  is the instance of the graph in this study.

It should be noted that all graphs used in this study, including the random graphs but excluding  $RBT(510, 5)$ , have 512 vertices so as to control for population size. The one off size graph has 510 vertices as a 5-regular balanced tree cannot have 512 vertices. Exploration of the trade-offs involved in varying the number of vertices more than a tiny amount is a topic for future research. The complete graph  $K_{512}$  is included as a baseline. Graph based evolutionary algorithms become equivalent to a standard type of evolutionary algorithm when the graph used is  $K_n$ .

### 3 Graph based evolutionary algorithms

This section defines a graph based evolutionary algorithm (GBEA) as it is used in this study. Clearly many other methods of incorporating graphs into evolutionary algorithms are possible. Assume that you have chosen a graph  $G$  with vertex set  $V(G)$  and edge set  $E(G)$  to use as a population structure. One individual is placed on each vertex of  $G$ . Then utilize a steady state evolutionary algorithm [22, 27, 33] where evolution proceeds one mating event at a time. A mating event is performed as follows. Pick a vertex  $v \in V(G)$  uniformly at random. A neighbor of  $v$  is then chosen for mating. The variation operators, crossover and mutation, are used to produce a single new individual that may or may not be used to replace the individual on vertex  $v$ . The details of how the neighbor is picked for mating and how to decide if the new individual replaces the individual on  $v$  are together called the *local mating rule* of the GBEA. In this research the local mating rule will pick neighbors in direct proportion to their fitness (local roulette selection) and let the new individual replace the old either automatically or only if it is at least as fit as the individual it replaces. We call these local mating rules *local roulette mating* and *local elite roulette mating* respectively.

Graph	Index Name	Regularity	Diameter	Mean Degree	Random
$C_{512}$	<b>C512</b>	<b>2</b>	<b>256</b>	<b>2</b>	<b>No</b>
$H_9$	<b>H9</b>	<b>9</b>	<b>9</b>	<b>9</b>	<b>No</b>
$K_{512}$	<b>Complete</b>	<b>511</b>	<b>1</b>	<b>511</b>	<b>No</b>
$P_{256,1}$	<b>P256_1</b>	<b>3</b>	<b>129</b>	<b>3</b>	<b>No</b>
$P_{256,17}$	<b>P256_17</b>	<b>3</b>	<b>18</b>	<b>3</b>	<b>No</b>
$P_{256,3}$	<b>P256_3</b>	<b>3</b>	<b>46</b>	<b>3</b>	<b>No</b>
$P_{256,7}$	<b>P257_7</b>	<b>3</b>	<b>22</b>	<b>3</b>	<b>No</b>
$R(512, 3, 1)$	<b>RAND3_1</b>	<b>3</b>	<b>11</b>	<b>3</b>	<b>Yes</b>
$R(512, 3, 2)$	<b>RAND3_2</b>	<b>3</b>	<b>10</b>	<b>3</b>	<b>Yes</b>
$R(512, 3, 3)$	<b>RAND3_3</b>	<b>3</b>	<b>10</b>	<b>3</b>	<b>Yes</b>
$R(512, 4, 1)$	<b>RAND4_1</b>	<b>4</b>	<b>8</b>	<b>4</b>	<b>Yes</b>
$R(512, 4, 2)$	<b>RAND4_2</b>	<b>4</b>	<b>7</b>	<b>4</b>	<b>Yes</b>
$R(512, 4, 3)$	<b>RAND4_3</b>	<b>4</b>	<b>7</b>	<b>4</b>	<b>Yes</b>
$R(512, 9, 1)$	<b>RAND9_1</b>	<b>9</b>	<b>4</b>	<b>9</b>	<b>Yes</b>
$R(512, 9, 2)$	<b>RAND9_2</b>	<b>9</b>	<b>4</b>	<b>9</b>	<b>Yes</b>
$R(512, 9, 3)$	<b>RAND9_3</b>	<b>9</b>	<b>4</b>	<b>9</b>	<b>Yes</b>
$R(0.07, 1)$	<b>RTor07_1</b>	<b>No</b>	<b>19</b>	<b>7.445</b>	<b>Yes</b>
$R(0.07, 2)$	<b>RTor07_2</b>	<b>No</b>	<b>20</b>	<b>7.805</b>	<b>Yes</b>
$R(0.07, 3)$	<b>RTor07_3</b>	<b>No</b>	<b>16</b>	<b>7.520</b>	<b>Yes</b>
$T_{16,32}$	<b>T16_32</b>	<b>4</b>	<b>24</b>	<b>4</b>	<b>No</b>
$T_{4,128}$	<b>T4_128</b>	<b>4</b>	<b>66</b>	<b>4</b>	<b>No</b>
$T_{8,64}$	<b>T8_64</b>	<b>4</b>	<b>36</b>	<b>4</b>	<b>No</b>
$Z$	<b>Z</b>	<b>4</b>	<b>19</b>	<b>4</b>	<b>No</b>
$RBT(512, 3)$	<b>RT1n512d3</b>	<b>3,1</b>	<b>16</b>	<b>1.996</b>	<b>No</b>
$RBT(512, 4)$	<b>RT1n512d4</b>	<b>4,1</b>	<b>11</b>	<b>1.996</b>	<b>No</b>
$RBT(510, 5)$	<b>RT1n510d5</b>	<b>5,1</b>	<b>9</b>	<b>1.996</b>	<b>No</b>

Table 1: Graphs used and their index names. Index names are used to index the graphs in figures.

Section 4, Experimental Design, will specify which local mating rule is used for each test problem.

## 4 Experimental design

Simulations were performed for 23 test problems on each of the 26 graphs given in Table 1. For 22 of the problems, 5000 independent evolutionary dsimulations were performed while for one problem (differential equation solution) 10,000 simulations were performed to obtain better resolution. The number of mating events required to find a correct solution to the problem was saved for each of these 3,120,000 simulations. If more than 1,000,000 mating events were required, the simulation was recorded as having failed to find an answer. For each graph and problem the mean and standard deviation of the number of mating events to solution were used to construct 95% confidence intervals for the mean time to solution.

These are displayed in Figures 5, 6, 7, 8, 9, 10, 11, and 12. The test problems, the local mating rule used with each test problem, and the exact character and rate of the variation operators used are described in the following sections.

## 4.1 One-max

The one-max problem uses a string of bits for a chromosome. In this study we used 20 bit strings. The fitness of a string is its weight (the number of 1's in it). For the one-max study we used local roulette mating. The crossover operator was two point crossover. The mutation operator flipped one bit selected uniformly at random. The choice not to use elite replacement on the one-max problem reflected the essentially trivial character of the problem. The search problem is harder without elite replacement and so more likely to yield information about the relative merit of graphs.

## 4.2 De Jong's Functions

The DeJong's functions  $F_1 - F_5$  are described in detail in [13]. The function  $F_1$  is a three dimensional bowl. De Jong's function  $F_2$  is a fourth degree bivariate polynomial surface featuring a broad suboptimal peak. Function  $F_3$  is a sum of integer parts of five independent variables, creating a function that is flat where it is not discontinuous, a kind of six dimensional ziggurat. Function  $F_4$  is a fourth-order paraboloid in thirty dimensions, with distinct diameters in different numbers of dimensions, made more complex by adding Gaussian noise added to it. Function  $F_5$  is the so-called "foxhole" function with many narrow local optima placed on a grid. These functions are traditional test problems in function optimization but do not serve as a complete test suite. See the review article by Whitley [31] for incisive comments.

## 4.3 The Griewangk Function

The Griewangk function is a sum of quadratic bowls with cosine terms added to them, translated to be a positive function. The function has one such bowl summed per dimension. It provides a test case with a plethora of local optima and is a natural member of a test suite. As the dimension of the Griewangk increases the function becomes smoother and hence less challenging [31]. For this reason we include this function in five cases of relatively low dimension,  $N = 3, 4, \dots, 7$ .

## 4.4 Self Avoiding Walks

The self-avoiding walk (SAW) problem uses a string as its chromosome. The string is over the alphabet  $\{up, down, left, right\}$  with the letters corresponding to moves on a grid. The cases of the SAW problem on grids of size  $3 \times 3, 3 \times 4, 4 \times 4, 4 \times 5, 5 \times 5, 5 \times 6$ , and  $6 \times 6$  are used. The length of a SAW chromosome is equal to the number of cells in the grid minus one. Fitness is evaluated by starting in the lower left corner of the grid and then making the moves specified by the chromosome. The sequence of moves made is referred to as the *walk*. If a move is made that would cause the walk to leave the grid then that move is ignored.

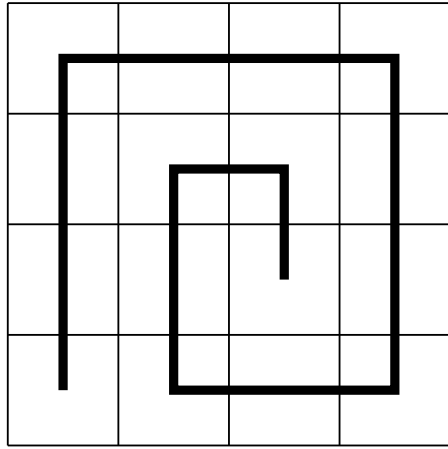
The walk can also revisit cells of the grid. When the walk is completed fitness is set equal to the number of squares visited. The problem is called the *self-avoiding* walk problem because optimal solutions may not revisit squares; they are self avoiding walks. Examples of SAW chromosomes and their fitness evaluations are given in Figure 3.

The self avoiding walk functions fill a role similar to those of NK-landscapes [14]. Both types of problem provide scalable problems that can exhibit a large degree of epistasis and both possess many global and local optima. The fourth example given in Figure 3, with fitness fifteen, has no near neighbors (in the Hamming metric) with fitness 16. It serves as an example of a local optimum. The SAW problem has several differences from the NK-landscape problems. Every instance of the SAW problem has a known best fitness; it is possible to know when you have succeeded. This makes the collection of statistics on algorithm behavior easier. The walk for a given SAW chromosome yields a simple and intuitive visualization that can be used to help in the analysis of the behavior of evolutionary algorithms.

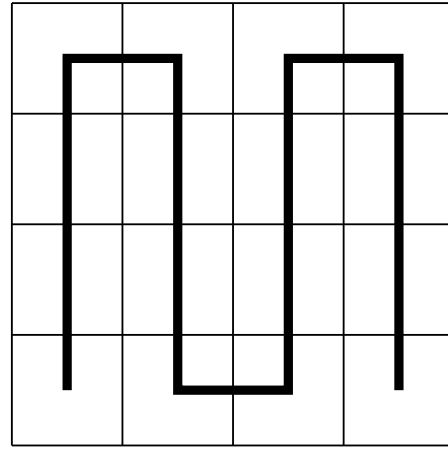
As SAW problems are a new type of test problem they should be checked against the list of criteria for good test suite problems given in [31]. Criterion one: SAW problems are quite resistant to hill climbing. Testing with a single mutation hill climber showed that the ratio of local to global optima located explodes combinatorially as the problem size increases. This is the mutation operator used in the tests presented here. Criterion 2: the SAW problem is constructively non-linear, nonseparable, and non-symmetric. If we permute the order of moves made, the fitness of a given chromosome jumps all over the place. Since sequences of moves are good only from a particular starting position the SAW problem is quite non-separable. Loci near the beginning of the chromosome have fitness independent of later loci, but the fitness of later loci deeply depends on the values of earlier loci; the fitness is thus not even close to additive and the problem is non-linear. Criterion 3: scalability. The SAW problems contain an infinite number of cases  $N \times M$  that can be scaled from trivial to to hard. Criterion 4: scalable evaluation cost. This is the sole criterion that the SAW fails to satisfy. The evaluation cost of a SAW problem is small when its size is such that there is any hope of solving it. Criterion 5: canonical representation. The SAW problem uses a string over a four letter alphabet. This is a canonical representation. The SAW problems thus can satisfy four of the five criteria needed for members of a good test suite and so, if paired with a problem that has scalable cost, yield an acceptable test suite. Work on using GBEAs with a high evaluation cost problem appear in [29, 8].

## 4.5 Plus-one-recall-store

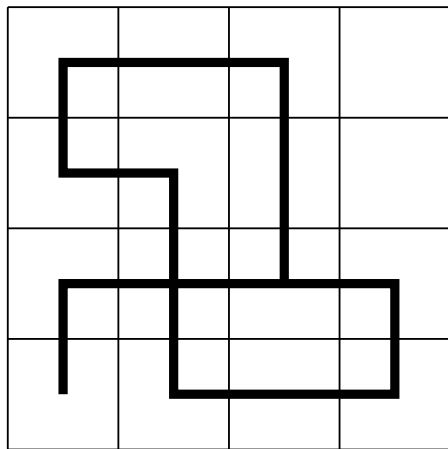
The plus-one-recall-store (PORS) problem is described in detail in [5]. It is a type of maximum problem within the domain of genetic programming [17, 16, 6] with a small operation set and a calculator-style memory. The goal of the test problem, called the PORS efficient node use problem, is to find parse trees that, when executed, generate the largest integer result possible given a fixed maximum number of parse tree nodes. The language has two operations, integer addition and a store operation that places its argument in an external memory location. The language also has two terminals, the integer 1 and recall from an external memory. The difficulty of the PORS efficient node use problem varies strongly according to the congruence class (*mod* 3) of the number of nodes permitted. We ran ex-



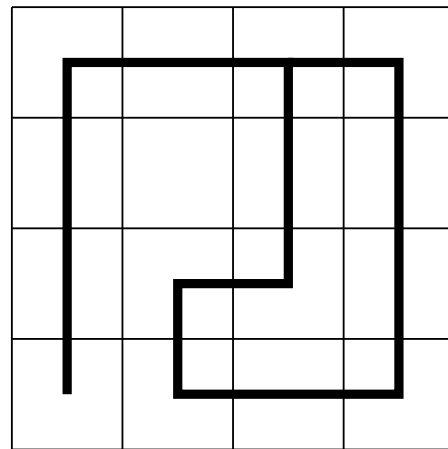
UUURRRDDDLLUURD



UUURDDDRUUURDDD



URDRRULLULURRDD



UUURRDDLDRRUUUL

Figure 3: Examples of two optimal and two suboptimal walks for the  $4 \times 4$  instance of the SAW problem. The fitnesses of the examples are 16,16,14, and 15, corresponding to the number of squares visited.

periments on  $n = 15$ ,  $n = 16$ , and  $n = 17$  nodes representing all three classes. The hardest case is  $n = 15$ , the easiest is  $n = 16$ . An example of a solution located for  $n = 16$  is given in Figure 4. Fitness for a given parse tree was the size of the number it produced when evaluated. In this set of experiments, the initial population was composed of randomly generated trees with exactly  $n$  nodes. A successful individual was defined to be a tree that produces the largest possible number (these numbers are computed in [5]). Crossover was performed by the usual subtree exchange [17]. If this produced a tree with more than  $n$  nodes, then a subtree of the root node iteratively replaced the root node until the tree had fewer than  $n$  nodes. This operation is called *chopping*. Mutation was performed by replacing a subtree picked uniformly at random with a new random subtree of the same size for each new tree produced. For the PORS experiments we used local elite roulette mating.

(+ (Sto (+ (+ (Sto (+ (Sto (+ (+ 1 1) 1)) Rcl)) Rcl) Rcl)) Rcl)

Figure 4: An optimal PORS tree located by a GBEA with graph  $C_{512}$  for  $n = 16$  nodes shown in LISP-like notation.

## 4.6 DNA Barcodes

DNA barcodes [4], are error correcting codes [19] over the DNA alphabet  $\{C, G, A, T\}$ , able to correct errors relative to the *edit metric*[12]. They are used as embedded markers in genetic constructs to permit retention of source information when sequencing pooled genetic libraries. An example of their successful use to retrieve sequence source information appears in [21].

Unlike binary error correcting codes over the Hamming metric, edit metric codes lack a beautiful algebraic theory. Those used were located with a *greedy closure evolutionary algorithm* [4]. This type of evolutionary algorithm uses a representation consisting of a partial structure. The fitness of an individual partial structure is the quality (in this case size) of its completion by a greedy algorithm. When searching for DNA barcodes the partial structure is a choice of three random DNA codewords and the greedy algorithm is Conway’s lexicode algorithm[4]. Fitness is simply the size of the code located by Conway’s algorithm. The DNA barcode search problem exhibits a high degree of epistasis and work thus far suggests it has an exceedingly rugged fitness landscape.

In this study we search for six letter DNA words that are at mutual distance at least three. These are the parameters used for the wet lab testing of the technique in [21]. Barcodes of this size and distance can correct one sequencing (edit) error.

## 4.7 Differential equation solution

Solving differential equations is a standard genetic programming problem. In addition to solving a differential equation within a GBEA, we also modify the usual technique by extracting the derivatives needed to compute fitness symbolically. Code for performing differential equation solution with symbolic derivatives is available by contacting the second author.

We solve the differential equation

$$y'' - 5y' + 6y = 0 \tag{1}$$

a simple homogeneous equation with a two dimensional solution space,

$$y = Ae^{2x} + Be^{3x} \tag{2}$$

for any constants  $A, B$ .

The parse tree language used has operations and terminals given in Table 2. Trees were initialized to have six total operations and terminals. Fitness for a parse tree coding a function  $f(x)$  was computed as the sum over 100 equally spaced sample points in the range  $-2 \leq x \leq 2$  of the error function  $E(x) = (f''(x) - 5f'(x) + 6f(x))^2$ . This is essentially the squared deviation from agreement with the differential equation. This function is to be minimized and the algorithm continues until 1,000,000 mating events have taken place (this did not happen in practice) or until the fitness function summed over all 100 sample points drops below 0.001. A filter was included to prevent trivial solutions, e.g.  $f(x) = 0$ , and trivial solutions were assigned a fitness of  $1 \times 10^{12}$  when they were detected.

Crossover and chopping were performed as in the PORS experiments, except that trees were chopped if they had in excess of 22 total operations and terminals. In addition to subtree mutation of the sort used in the PORS experiments, a *constant mutation* was applied to each new parse tree. Constant mutation has no effect on parse trees that do not contain ephemeral real constants. For a tree that does contain such constants, either as a terminal or as a part of the unary scaling operation, one of the constants is selected with a uniform distribution and then a number uniformly distributed in the region  $[-0.1, 0.1]$  is added to the constant. Ephemeral constants are initialized in the range  $[-1, 1]$  but may be taken outside of this range by constant mutation. For the differential equation solution experiments we used local elite roulette mating. Each new tree produced results from a subtree crossover and is subjected to both a subtree mutation and a constant mutation.

Equations 3-5 are examples of solutions found by a GBEA on the graph  $Z$ . All of these are in fact exact solutions to the equation as were the majority of solutions located.

$$(e^x)^2 \tag{3}$$

$$(-0.42 * (0.024 + e^{0.71})e^x)^2 \tag{4}$$

$$e^{x+x+x} \tag{5}$$

## 5 Results

The test problems were chosen because they represented different classes of problems that were well studied in many cases and all had known solutions. As targets for evolutionary algorithms the one-max is a standard test problem. De Jong's problems are well known and will permit comparison with other work using those function, although they do not meet the criteria given in [31] to be a test suite. The lower dimensional cases of the Griewangk function are difficult functions for optimization. PORS is a test problem with an exceptionally well-characterized fitness landscape for genetic programming. The  $n = 15$  case is a deceptive

Terminals	
x	the independent variable.
r	ephemeral real constants.
Unary operations	
~	Negation
sin	trigonometric sine function
cos	trigonometric cosine function
atan	trigonometric arctangent function
exp	exponential function
log	natural log function
sqr	square function
Xr	scaling by an ephemeral real constant
Binary operations	
+	binary addition
-	binary subtraction
*	binary multiplication
/	binary division

Table 2: Operations and terminals for the differential equation parse tree language. The symbol r denotes a real number.

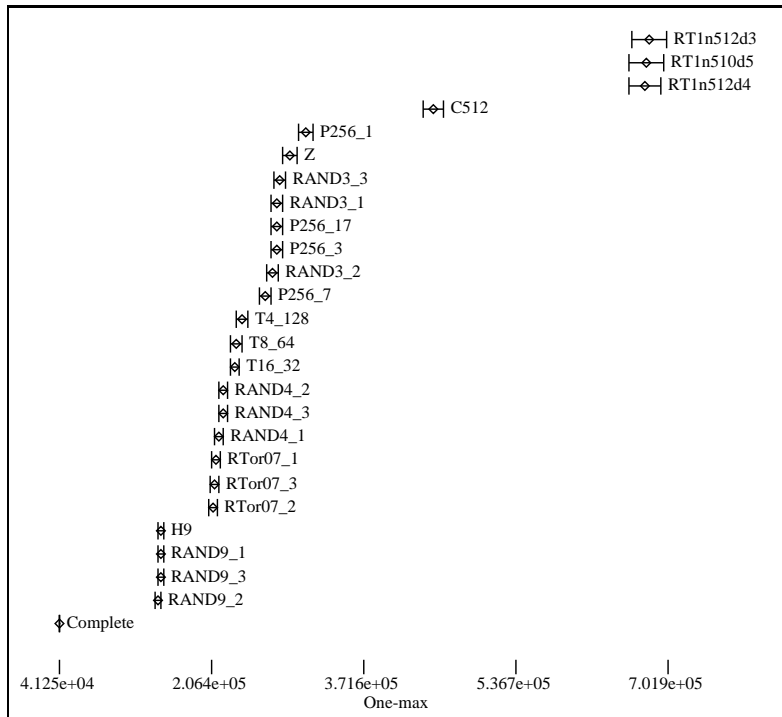


Figure 5: Mean mating events to solution with 95% confidence intervals for the one-max problem.

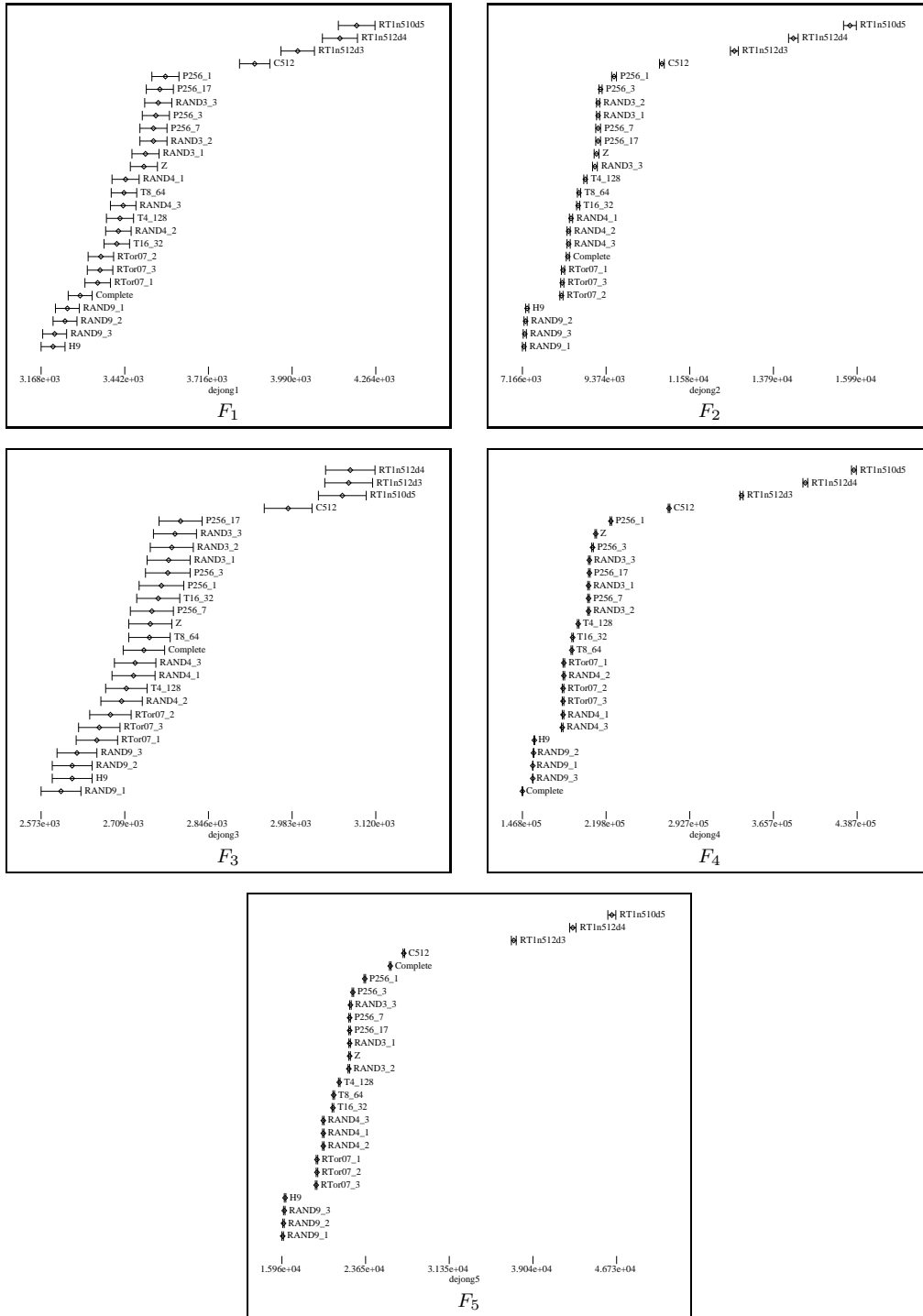


Figure 6: Mean mating events to solution with 95% confidence intervals for the DeJong test suite, functions  $F_1 - F_5$ .

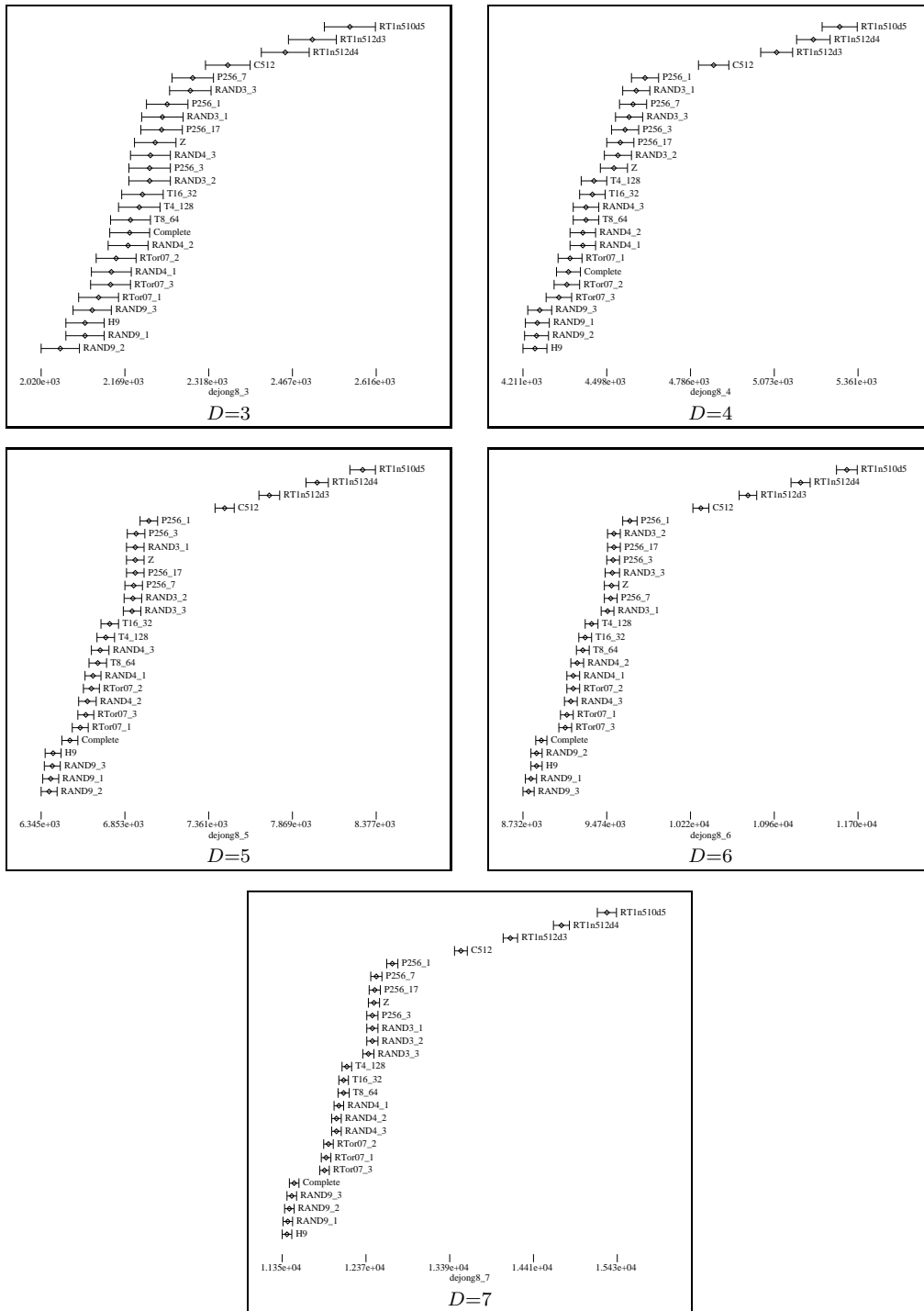


Figure 7: Mean mating events to solution with 95% confidence intervals for the Griewangk function in 3-7 dimensions.

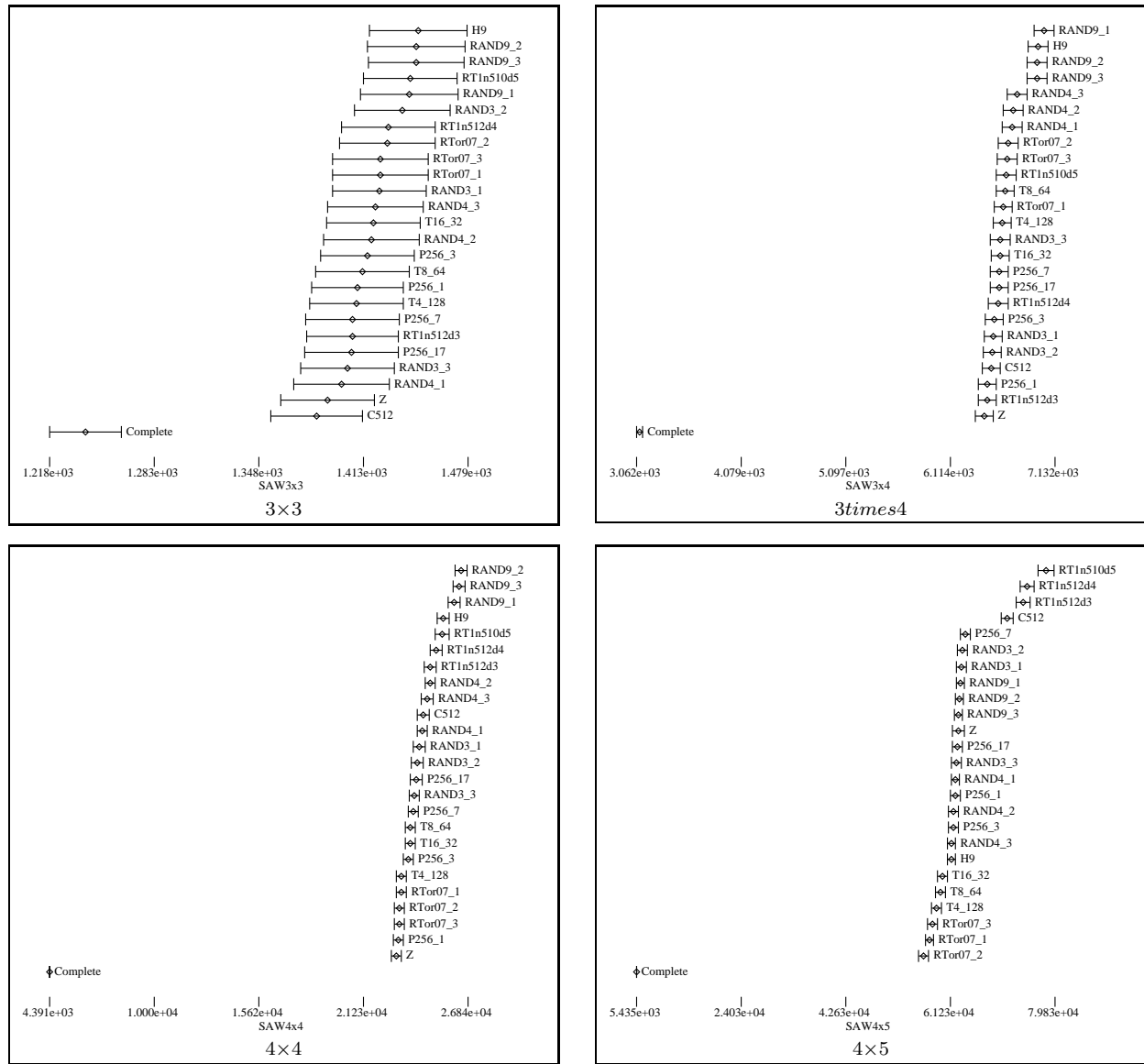


Figure 8: Mean mating events to solution with 95% confidence intervals for self avoiding walks of size  $3 \times 3$ ,  $3 \times 4$ ,  $4 \times 4$ , and  $4 \times 5$ .

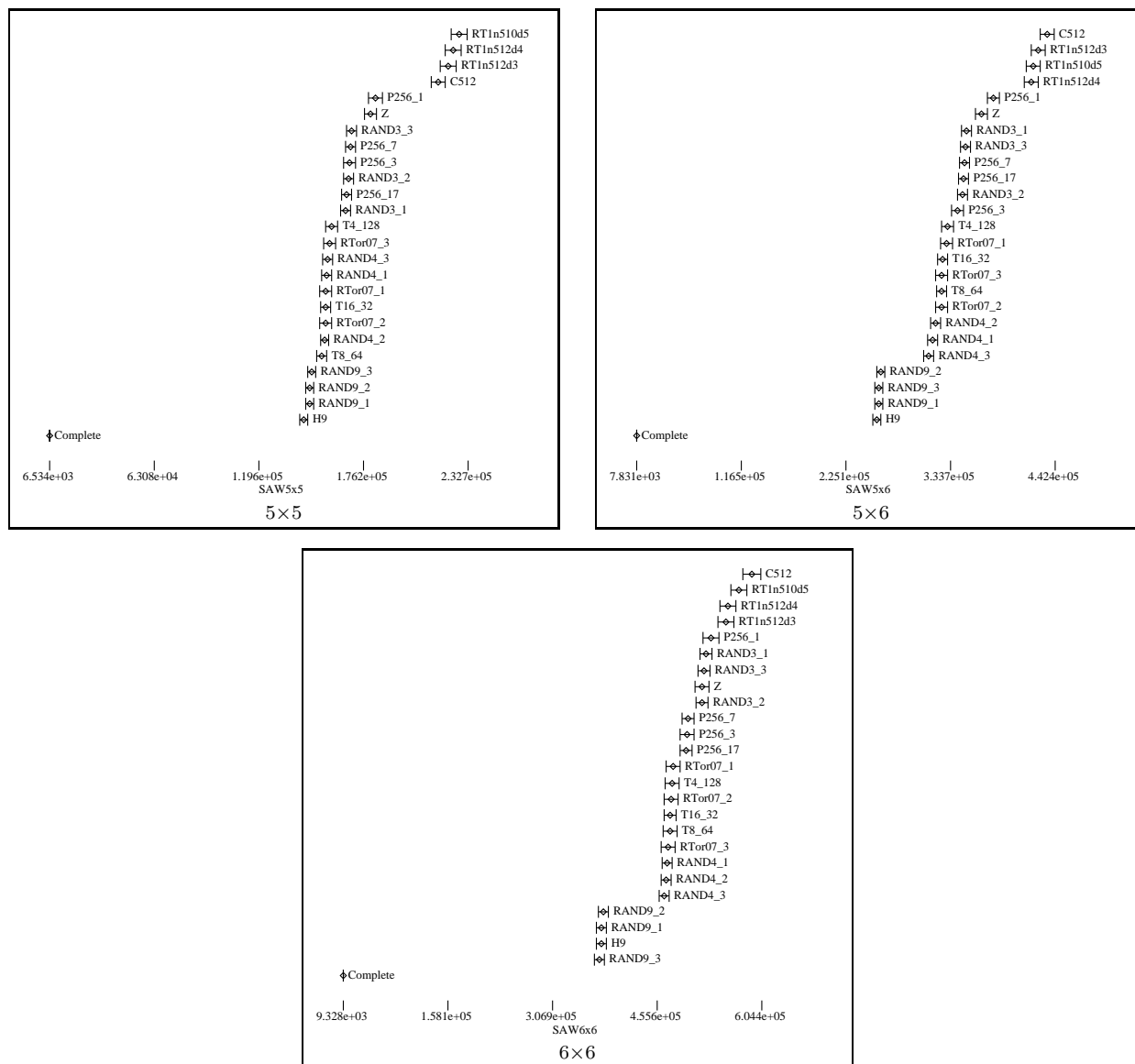


Figure 9: Mean mating events to solution with 95% confidence intervals for self avoiding walks of size  $5 \times 5$ ,  $5 \times 6$ , and  $6 \times 6$ .

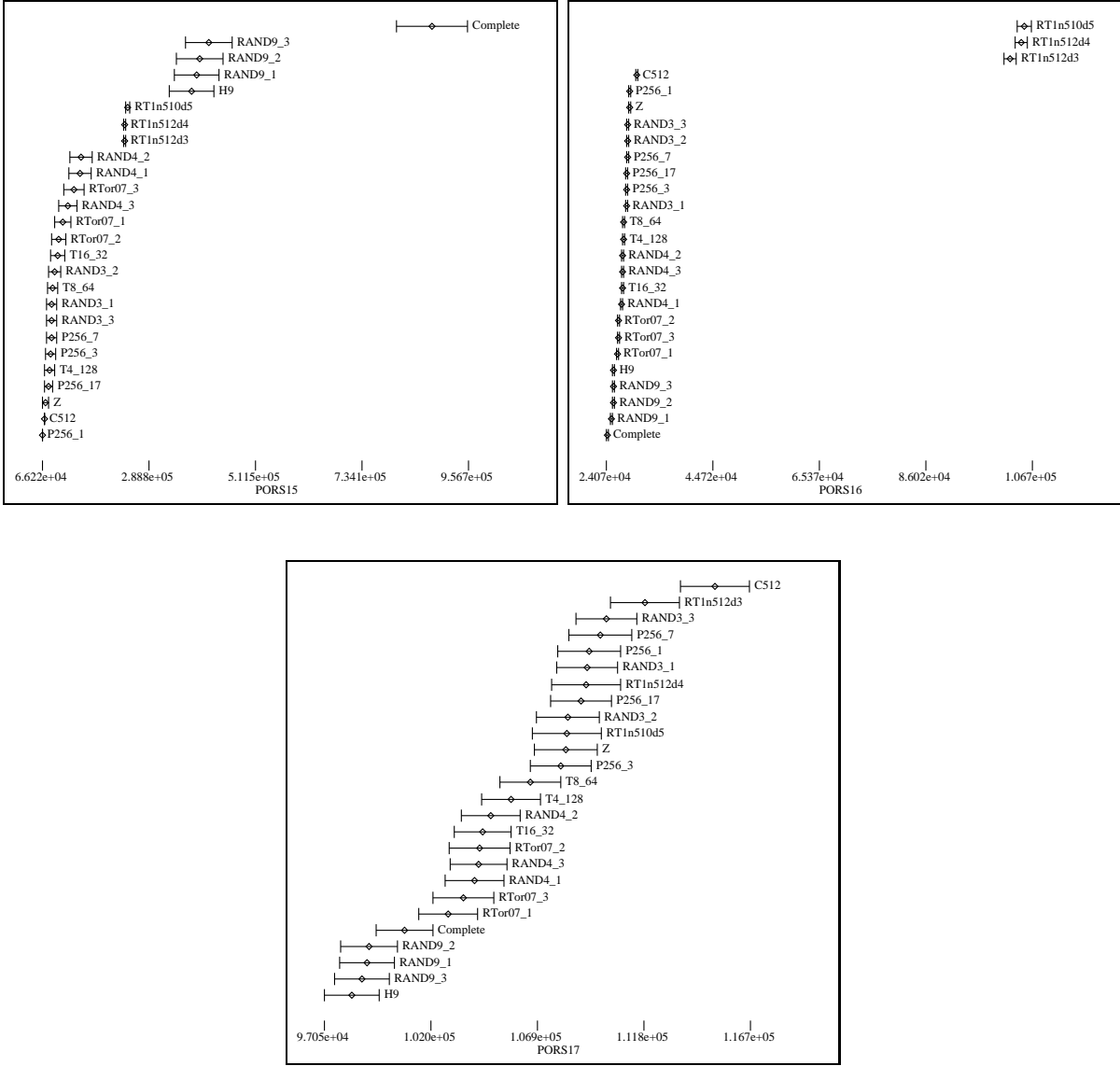


Figure 10: Mean mating events to solution with 95% confidence intervals for the PORS problem with  $n = 15$  (top left),  $n = 16$  (top right), and  $n = 17$  (bottom).

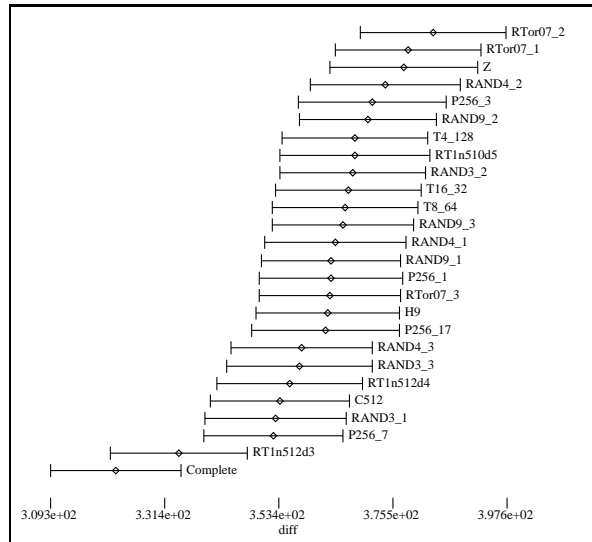


Figure 11: Mean mating events to solution with 95% confidence intervals for the differential equation solution problem.

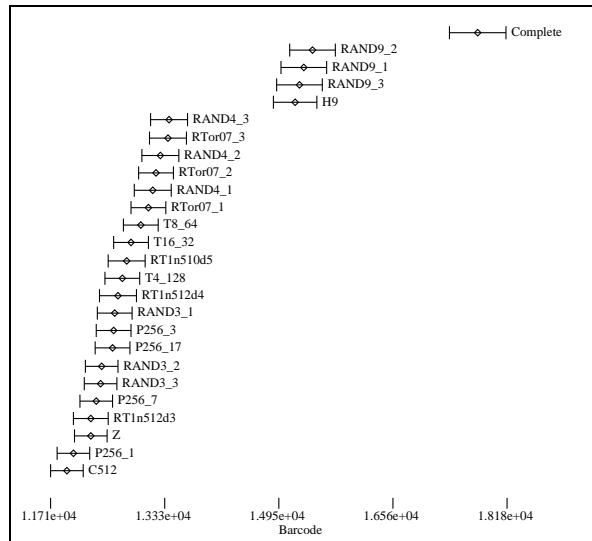


Figure 12: Mean mating events to solution with 95% confidence intervals for the DNA bar-code problem.

problem, containing a unique and narrow global optimum and many broader local optima, while the  $n = 16$  and  $n = 17$  cases are not. The DNA barcode problem is a new one, included to have at least one applied problem, but the parameters used in this study are the most studied. The ordinary differential equations solution is a precursor to many applied problems, including heat transfer, fluid flow and combustion [9, 23, 7]. Our primary interest is in determining the potential impact of population graphs on solution speed and documenting which graphs yield superior performance for a specific problem. The major results found are that choice of graph substantially impacts solution time and that the correct choice of graph varies from problem to problem.

For each graph and test problem, 5000 tests were completed, except in the case of the differential equation where 10,000 tests were completed for each graph. In each case, time-to-solution numbers were saved with time measured in mating events. For the one-max, SAW, and PORS problems, the solution consisted of the appearance of the first instance of the known correct solution. In the function optimization problems a simulation was said to have found the solution when it obtained a value within 0.001 of the known optimal value. For DNA barcodes we evolved until we achieved the size of the current best-known solution. For the differential equation problem, the correct solution was taken to be a total squared error over all 100 sample points of at most 0.001. We display the relative performance of each graph as scatter plots with 95% confidence intervals and the graphs sorted in increasing order of time to solution in Figures 5-11

As used in this discussion, “performance” refers to the number of mating events required to find an acceptable solution to the problem. The top-to-bottom impact that the choice of graph has on problems is shown in Table 3. An initial examination of the confidence intervals shows that performance varies from graph to graph, often significantly. Also, the degree to which performance varies is problem dependent. This indicates that graph based evolutionary algorithms have the potential to significantly reduce convergence time for many classes of challenging engineering problems. It is important to remember that the one-max problem as presented here uses a non-elitist algorithm, increasing its difficulty as a search problem. The other twenty sets of simulations use elitist algorithms. The test problems can be divided into the following groups.

- (1) Problems with a simple fitness landscape. **One-max,  $F_1, F_3$ , and  $F_4$**  – The fitness landscape for these problems is a single hill that fills the entire landscape, adjusted in the case of  $F_4$  by noise. One-max and  $F_3$  are very similar, discontinuous pyramids, save that  $F_3$  is encoded as real numbers and so has a different representation. These are relatively straightforward optimization problems.
- (2) Problems in which the fitness landscape has many local optima and several global optima. PORS  $n=16,17$ , Differential Equation, SAW – Although the PORS  $n = 16$  problem has multiple optimal hills in its fitness landscape, each of these hills has the same fitness value. Additionally, many of the sub-optimal solutions for the PORS  $n = 16$  problem contain “building blocks” that are tree fragments that create the numbers 2 or 3 or multiply a single argument by those numbers. The optimal answer requires two fragments creating or multiplying by 2 and two fragments creating or multiplying by 3. The effect of this is that the majority of the local optima in the

Problem	Mean Time		Ratio	Benefit
	Minimum	Maximum		
One-max	42282.10	682860.70	16.15	0%
PORS15	67324.34	882285.11	13.10	1210%
PORS16	24320.66	105289.54	4.33	0%
PORS17	98339.50	115117.00	1.17	2%
Diff. Equation	322.10	383.30	1.19	0%
De Jong $F_1$	3209.19	4201.28	1.31	3%
De Jong $F_2$	7217.55	15823.72	2.19	16%
De Jong $F_3$	2605.59	3079.35	1.18	5%
De Jong $F_4$	147398.13	435944.96	2.96	0%
De Jong $F_5$	16054.04	46313.54	2.88	62%
Griewangk $D = 3$	2054.43	2570.03	1.25	6%
Griewangk $D = 4$	4253.44	5300.13	1.25	3%
Griewangk $D = 5$	6395.64	8296.51	1.30	2%
Griewangk $D = 6$	8787.67	11603.99	1.32	1%
Griewangk $D = 7$	11409.45	15317.12	1.34	1%
SAW $3 \times 3$	1240.80	1448.10	1.17	0%
SAW $3 \times 4$	3096.87	7028.53	2.27	0%
SAW $4 \times 4$	4434.07	26506.25	5.98	0%
SAW $4 \times 5$	5483.04	78328.98	14.29	0%
SAW $5 \times 5$	6586.71	228207.52	34.65	0%
SAW $5 \times 6$	7888.20	435027.35	55.15	0%
SAW $6 \times 6$	9389.10	591590.50	63.01	0%
DNA Barcode	11945.65	17772.00	1.49	49%

Table 3: Impact of choice of graph on solution time. For each problem the minimum and maximum mean time to solution for any of the 26 graphs used is given together with their ratio. The benefit column gives the improvement over the baseline standard evolutionary algorithm.

search space contain the tree fragments needed in each of the 24 optimal answers. These optimal answers differ only in the details of how they use the building blocks. See [5] for details.

The fitness landscape for a differential equation problem is the most intricate of the test problems. It is far larger and weirder than landscapes for the other test problems. As a search problem, it is dense with small correct answers (e.g. Equations 3 and 5), so much of the space is not involved in most searches. Unlike the PORS16 problem the majority of these solutions cannot be built from fragments of each other.

- (3) Mildly deceptive or difficult landscapes with a global optima hidden by a larger local optima, e.g. DeJong  $F_2$  and some of the lower dimensional Griewangk functions.
- (4) Problems with very difficult, possibly deceptive landscapes. **PORS15**,  $F_5$ , **DNA**

**barcodes** – The PORS problem for  $n = 15$  is the hardest search problem among the test problems we examined. Following the reasoning for  $n = 16$ , the difficulty arises because the correct solution is a tree that computes  $32, 2^5$ . This means that the correct solution is unique, making the solution difficult; additionally, trees that generate 3's are local optima that use large (5 node) tree fragments that are of no use at all in an optimal solution. See [5] for details. The foxhole function,  $F_5$  offers a large number of traps, somewhat similar to the suboptimal solutions in the PORS15 problem.

From Table 3 it is clear that use of graphs has a substantial impact on the difficult or deceptive problems in the test set. Even for the three hard problems ( $F_5$ , PORS15, DNA barcodes) the impact of graphs was very different. To compare  $F_5$  and PORS15 examine Figure 13. The baseline evolutionary algorithm, the GBEA with the complete graph  $K_{512}$ , is the lowest diameter graph in both plots, with  $\text{Log}_2(\text{Diameter}) = 0$ . For the foxhole function  $F_5$  the complete graph is an outlier whereas in the PORS15 the complete graph is part of a smooth inverse correlation between diameter and time to solution.

## 5.1 Performance of the complete graph

The complete graph, a GBEA configured to run as a standard evolutionary algorithm, yielded the best results for the following problems: one-max, the noisy unimodal function  $F_4$ , the simplest of the three PORS problems, all cases of the SAW problem, and the differential equation problem. This last had the lowest time to solution on average of any of the problems checked. These problems include both unimodal and the most highly polymodal problems in the test set. They do not include the difficult problems, PORS15, DNA barcodes, and the foxhole function  $F_5$ .

## 5.2 Performance of degree 9 graphs

The hypercube or one of the three random graphs derived from it were the best graph for  $F_1, F_2, F_3, F_4, F_5$ , and all instances of the Griewangk function. In most of these cases the hypercube and its random analogs outperformed the complete graph by a modest margin. For the deceptive function  $F_5$  the difference was quite large. If one graph must be chosen then the suite of problems used in this study suggest the hypercube is a good compromise choice. It did however perform poorly on both PORS15 and DNA barcodes.

## 5.3 Performance on the hardest problem

If we rate problem difficulty by time to solution, the PORS15 problem is the most difficult problem in the test suite. The worst graph for this problem is the complete graph. The second worst are the four degree 9 graphs. Thus the two types of graph that, between them, are the best for all other problems in the test suite are the worst for the PORS15. The best graphs from this problem are  $C_{512}, P_{256,1}$ , and  $Z$ .

The graphs  $C_{512}, P_{256,1}$  are the two highest diameter graphs. The graph  $Z$  is the only one specifically designed for GBEAs. Its essentially fractal character, with closely coupled smaller groups organized into less closely coupled larger groups across three levels of scale,

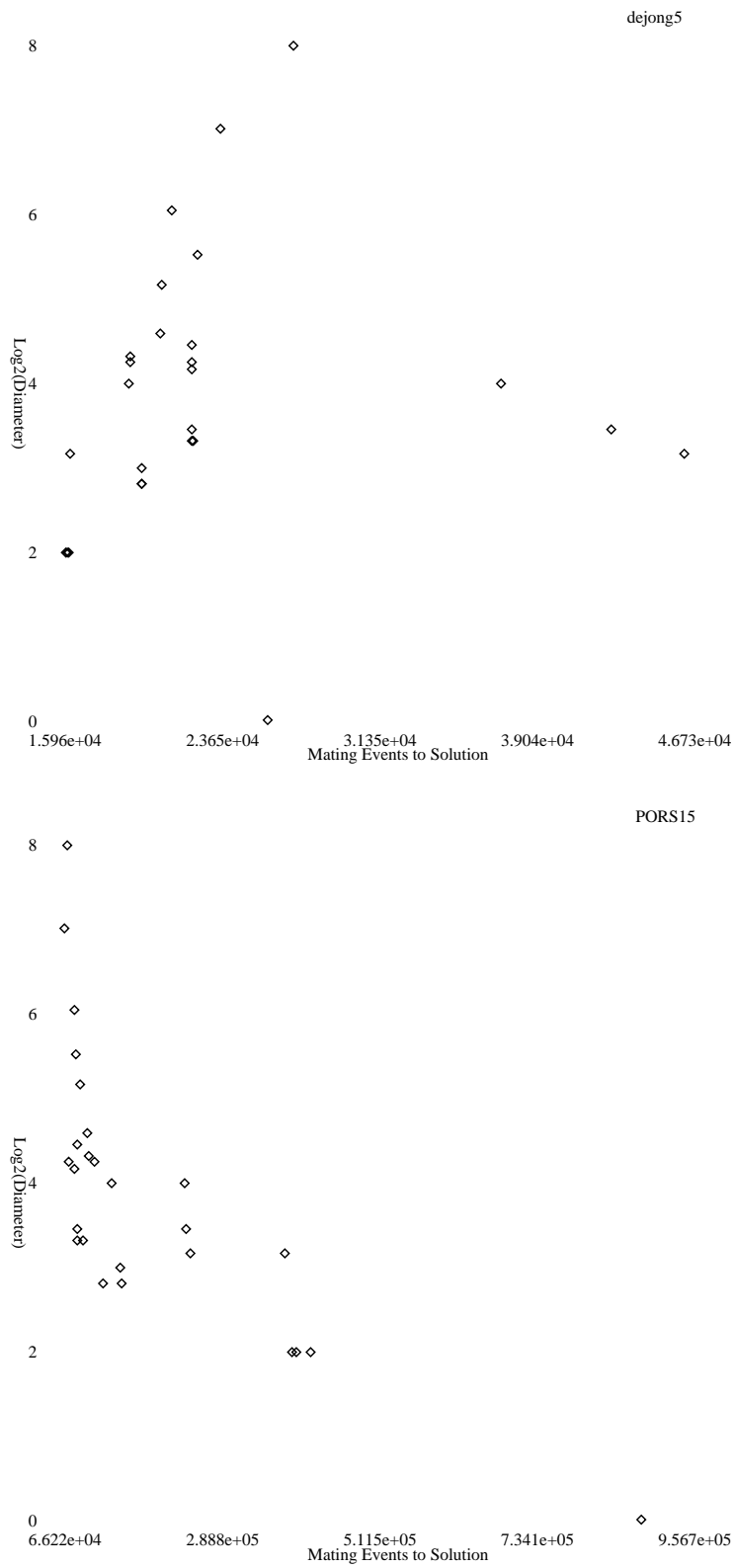


Figure 13: A plot of graph diameter versus time to solution for  $F_5$  function and PORS15.

is intended to create local demes. Having a high diameter is another method of creating disparate demes through isolation by distance.

## 5.4 The impact of randomization

For degree 3, 4, and 9 we included three randomized graphs of that degree in the set of graphs tested. The experiments demonstrate very little impact of this randomization. In most experiments the randomized graphs group with the non-random graphs of the same degree. Even when a statistically significant separation appeared, e.g. degree 4 graphs for PORS  $n = 15$ , the graphs were in the middle of the distribution of performances. There are combinatorially huge numbers of randomized versions of a given type of regular graph. Some of these may in fact exhibit significantly superior performance - nevertheless randomly sampled graphs within a degree family do not exhibit useful levels of enhanced performance.

## 5.5 The very worst type of graph

The regular trees were extraordinary in having only problem where any of them performed well, the DNA barcode problem. For the SAW problems they were in the middle of the pack. The SAW problems were the ones best solved with a standard algorithm, i.e. a GBEA using  $K_{512}$ . For PORS  $n = 15$  they were in the bottom half but beat the complete and degree 9 (hypercube) families. In all other problems they were the worst, often by a large margin. The current test suite of problems gives no reason to think these graphs should ever be used.

## 5.6 The deceptive functions

The DeJong function  $F_5$  and the PORS  $n = 15$  problems are the deceptive problems in the test suite. Figure 13, which displays time to solution versus the log of graph diameter, shows that the behavior of the graphs on these problems are very different. For  $F_5$  there is a rough correlation of log diameter and time to solution with the complete graph and the regular trees behaving as outliers. For PORS15 there is a fairly strict inverse correlation of log diameter with time to solution.

The behavior of  $C_{512}$  and  $K_{512}$  on  $F_5$  demonstrate that diameter and degree do not tell the whole story. If we dismiss the behavior of the regular trees as pathological, the two graphs with extreme degree and diameter have almost the same average time to solution on  $F_5$ . The way that  $H_9$  beats  $K_{512}$  on many problems is additional evidence that graph structure beyond degree and diameter impacts performance.

The PORS15 problem has a simpler behavior than  $F_5$ . We hypothesize that high diameter graphs act like island models. The water between the islands are made of majority low fitness members of the initial population. The islands form around distinct higher fitness individuals. Each island is a chance not to fall into one of the local optima of the fitness landscape. In order to check this hypothesis a set of runs were performed with 128 disjoint copies of  $K_4$ . The time to solution was comparable to that of  $C_{512}$ .

## 6 Phylogenetic analysis

The data available after performing the 26 graph by 23 problem comparison permit a novel sort of analysis of the problems used. A *taxonomy* is a hierarchical classification of a set. Linnaeus established the first definite hierarchy used to classify living organisms. Each organism was assigned a kingdom, phylum, class, order, family, genus, and species. This hierarchy gave a tree structure to the taxonomy for all living creatures. Modern taxonomy has nineteen levels of classification, extending Linnaeus' original seven. A *cladogram* is a tree diagram showing the evolutionary relationship among various taxonomic groups. The reader should see [18] for details on modern taxonomic procedures for living organisms. The data gathered in the GBEA study are used to create a taxonomy of the test problems used. Making such a cladogram requires that we extract *taxonomic characters* from the collection of problems. A taxonomic character is simply a measurable or computable quantity such as number of legs or maximum number of teeth in a healthy adult. With taxonomic characters in hand, hierarchical clustering is then used to produce a cladogram that classifies the problems as more and less similar. Hierarchical clustering starts with the members of a set, thought of as singleton subsets. It then joins the closest pair and replaces them with their union or average.

The choice of taxonomic characters used for clustering is critical. They must avoid bias, they must vary across the set of problems, and they must avoid arbitrary judgments to the greatest degree possible. Using color in a numerical tree building algorithm, for instance, requires numbers be assigned to colors in a fashion that arbitrarily ranks some colors as closer to one another than others. The preceding brief discussion gives only a taste of the difficulty of choosing good taxonomic characters. Readers familiar with automatic classification, decision trees, and related branches of machine learning will recognize that those choosing decision variables face similar issues. Any taxonomic character or decision variable must be relevant to the decision being made, vary across the set of objects being classified, and be cleanly computable for all members of the set of objects being classified.

GBEAs provide a source of taxonomic characters that are computable for any evolutionary computation problem that has a detectable solution or end point. These characters are purely numerical. The characters can be defend these characters as objective, in the sense that they do not favor any particular choice of representation or parameter setting. These characters are computed in the following fashion. When using GBEAs, the time-to-solution for a problem varies in a complex manner with the choice of graphical connection topology. This complexity is itself the genesis of our taxonomic characters. The taxonomic characters used to describe a problem are the normalized mean solution times for the problem on each graphs. This gives each of our 23 problems a set of 26 taxonomic characters. The resulting taxonomy is given in Figure 14.

### 6.1 Details of the taxonomic technique

For each of the 23 problems  $P$  a real vector  $m(P)$  with 26 entries corresponding to the normalized mean solution time in each of the 26 graphs was obtained. The entry  $m(P, G)$  of  $m(P)$  corresponding to graph  $G$  was the normalized mean number of mating events required to solve problem  $P$  on graph  $G$ . The linear normalization was set so that the solution of

UPGMA

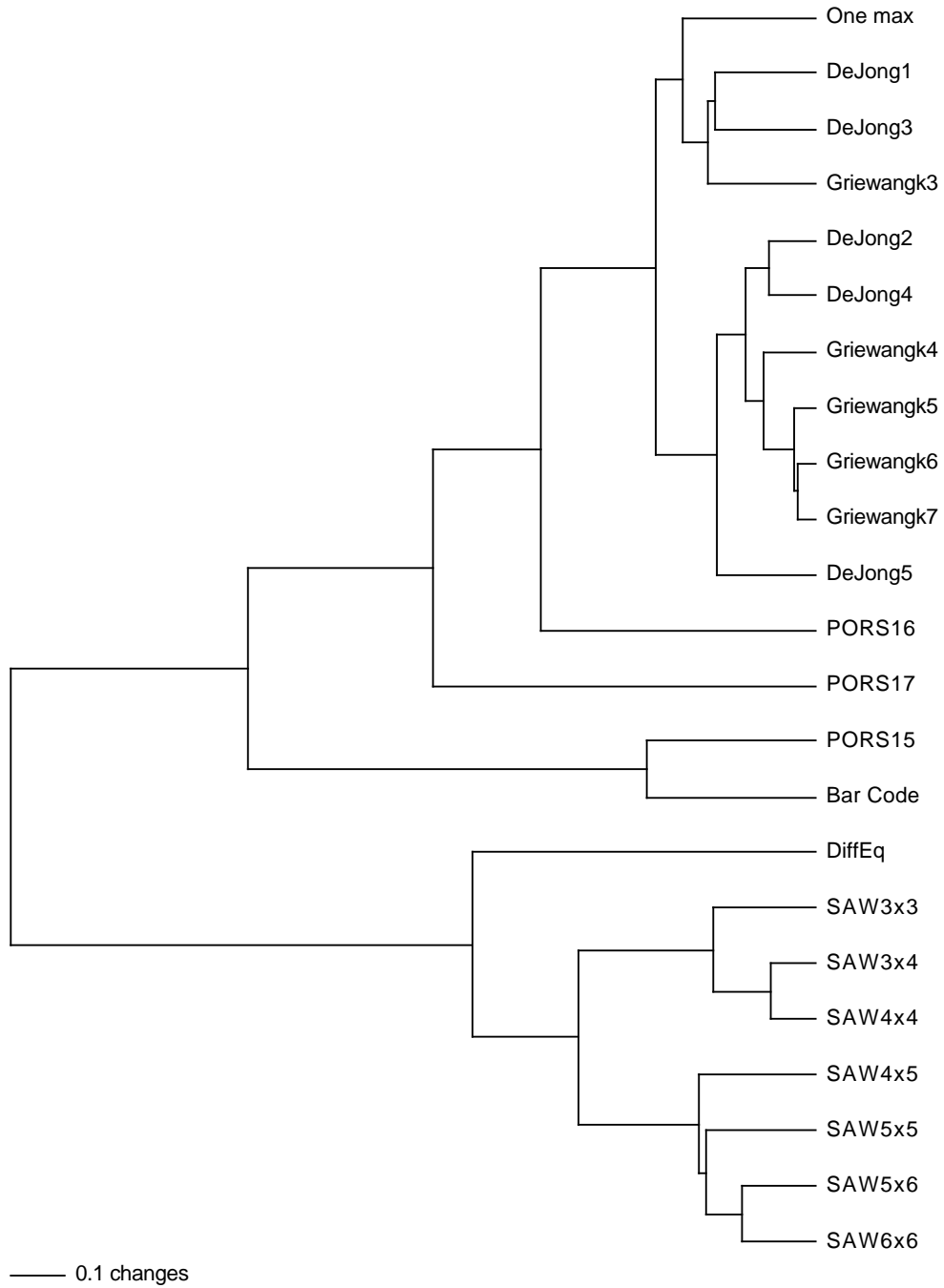


Figure 14: Results of Taxonomic Analysis of the test problems.

$P$  on the graph  $G$  which required the largest mean number of mating events among the 26 different graphs received the score  $m(P, G) = 1$ , while the graph  $G$  which required the smallest mean number of mating events received the score  $m(P, G) = 0$ . For each pair of problems  $P$  and  $Q$ , the Euclidean distance  $d(P, Q)$  between the vectors  $m(P)$  and  $m(Q)$  was then computed by the formula

$$d(P, Q) = \sqrt{\sum_{i=1}^{26} [m(P, G_i) - m(Q, G_i)]^2}.$$

We interpreted  $d(P, Q)$  as the distance between the problems  $P$  and  $Q$ . In order to describe the taxonomic relationships among the 23 problems, we then found the “UPGMA” tree.

UPGMA is a clustering method commonly used to transform distance data into a tree. It received attention in [24], and a good recent description may be found in [26]. It is especially reliable if the distances have a uniform meaning. Our normalization of the numbers  $m(P, G)$  had the effect of making the widely different rates of convergence comparable, so that we think that the inferred distances are appropriate for analysis by UPGMA.

UPGMA is an acronym for “Unweighted Pair Group Method with Arithmetic mean.” Given a collection of taxa and distance  $d_{ij}$  between taxa  $i$  and  $j$ , the method first links the two taxa  $x$  and  $y$  that are least distant. The taxa  $x$  and  $y$  are merged into a new unit  $z$ . For all taxa  $i$  other than  $x$  and  $y$ , a new distance  $d_{iz}$  is computed as the average of  $d_{ix}$  and  $d_{iy}$ , and it is noted that the new taxon  $z$  really represents the average of 2 original taxa. Henceforth,  $x$  and  $y$  are ignored, and the procedure is repeated to find the next pair of taxa that are least distant. When two taxa  $u$  and  $v$  are combined into a new taxon  $w$ , the new distance  $d_{iw}$  is the average of  $d_{iu}$  and  $d_{iv}$ , weighted according to the number of original taxa in  $u$  and  $v$  respectively. It is noted that  $w$  contains all the original taxa in both  $u$  and  $v$ . The procedure ends when the last two taxa are merged.

The UPGMA tree  $T$  was computed using the standard software package PAUP\*[25].  $T$  is shown in Figure 14. Horizontal distances are proportional to the edge lengths, while vertical distances are arbitrary and selected for legibility. Problems separated by a small horizontal distance (such as Griewangk5, 6, and 7) should be regarded as very similar. The separation of widely separated problems should be regarded as significant.

## 6.2 Discussion of the taxonomic results

The tree given in Figure 14 has several striking features. (1) All the numerical problems are grouped into a single clade with the one-max problem. (2) All the SAW problems are grouped into a single clade. Moreover, the SAW problems break into two subclades (one of which is  $3 \times 4$ ,  $3 \times 4$ , and  $4 \times 4$ ) of comparable horizontal extent as the numerical problems and hence of comparable diversity of problem type with the numerical problems. (3) The three PORS functions appear substantially different both from each other and from the numerical and SAW clades, as indicated both by their large horizontal extent and their placement so as not to form a clade.

The utility of the taxonomy is demonstrated by the two-member clade containing the PORS15 and DNA barcode problems, which are the most difficult problems tested here.

Suppose that PORS15 were part of a standard test suite of problems, and the DNA bar code problem were regarded as a new practical problem, not part of the test suite. Taxonomic analysis could place DNA barcode with PORS15, which would then suggest that the graphs which worked well on PORS15 would be most likely to work well on DNA barcode as well. In fact, this expectation is realized in this case. Examining Figures 10 and 12, we see that these two problems perform best on the same three graphs ( $P_{256,1}$ ,  $C_{512}$ , and  $Z$ ) and also perform worst on the same five graphs ( $R(512, 9, 1)$ ,  $R(512, 9, 2)$ ,  $R(512, 9, 3)$ ,  $H_9$ , and complete). The good performance is on comparatively sparse graphs, and the poor performance is on graphs of high regular degree. This suggests that future searches for better DNA barcodes should probably utilize GBEAs with such sparse graphs (and especially avoid graphs of high regular degree). This information is of substantial worth in an ongoing project to create DNA barcode sets for new sequencing projects.

The SAW and PORS problems demonstrate their worth as test suite problems by exhibiting substantial diversity in problem character (horizontal extent in  $T$ ). These results confirm the mathematical analysis in [5] that suggests that the three PORS problems have substantially different characters. The placement of PORS17 between PORS15 and PORS16 confirms that PORS17 is of an intermediate nature compared to the other two problems. The SAW problem set generates substantial diversity by simply varying its parameter. By contrast, the numerical problems generate less diversity and an effective test suite might omit some of the problems as being redundant.

It is important to note that the taxonomy reflects relative performance on different graphs and not problem difficulty. The normalization of mean times into the range  $[0,1]$  before use in comparing the problems eliminates all information about the amount of time required to solve the problem. This explains why the semi-symbolic differential equation problem ended up as a sister group to the SAW clade even though it is enormously easier than most of the SAW problems. This comparative simplicity is shown by the small number of mating events required for solution in Figure 11 compared with Figure 9.

Overall, the taxonomy in Figure 14 is plausible and agrees with what the authors know of the test problems. The technique shows promise for helping to decide which problems are similar. It may also help to winnow large test problem suites by picking representatives from groups of similar problems (such as selecting only a few representative numerical problems rather than including all of them).

## 7 Conclusions

Graph based evolutionary algorithms can improve performance on some problems. Among the problems used in this study, performance gain is the greatest on the hardest problems. On many of the problems used a standard evolutionary algorithm exhibits the best performance. The choice of correct graph for a GBEA is clearly problem dependent. The taxonomy given in Figure 14 gives some guidance as to which problems are similar, at least in the sense of having similar choices of good and bad graphs. As a rule of thumb, difficult and deceptive problems want sparser graphs. The maximum improvement in performance as over 1200% but roughly half of all test problems showed no improvement from using a GBEA.

The run-time cost using a GBEA is very low. If the correct graph for a problem can be

located there is the potential for substantial benefit at very low cost. For the Griewangk and SAW functions the performance ordering of the graphs was robust as the dimension of the problem was changed. This suggests that locating a “correct” graph for a problem could be done on lower dimensional or smaller problem cases and then scaled. This notion requires additional study.

The behavior of a problem on a suite of graphs forms an interesting description of the problem itself. By looking at which graphs are good and bad for a given problem we can characterize the problem. This gives us an objective taxonomic tool which may be quite useful for classifying problems. It is worth noting that the taxonomy, as presented here, is an essentially exploratory technique for data analysis.

## 8 Future Directions

GBEAs are already being used in a project to design efficient wood burning stoves for the third world [29]. In this case fitness evaluation involves the use of CFD software with run-times of several minutes per fitness evaluation. The diversity preservation conjectured for GBEAs is critical in the small population size practical in this project, and GBEAs do improve performance in the search for wood stove designs.

The role of local connectivity in the performance of a graph needs to be understood. It is likely that local topology can be constructed that optimizes diversity while permitting some exchange of information. The graph  $Z$  was an attempt at designing such a graph. It showed average performance on most problems, though it shone on the hardest problem, PORS15. Such graph design may yet result in faster convergence for some classes of problems.

Several additional steps in this investigation are needed. First, examining the impact of differing population sizes; second, examining the impact of graph topology on performance within a family of graphs; third, examining a broader range of test problems. To the extent that they exist, taxonomies of test problems will be incorporated into our future experiments.

These results need to be extended to problems of practical interest, in addition to the wood burning stove problem. The authors are specifically working to extend these results to problems involving the design and optimization of energy systems, heat transfer, and fluid mechanics. Particularly intriguing are those problems in which micro-scale phenomena significantly change a macro-scale outcome of interest. One example of this is the problem of optimizing the performance of utility furnace and boiler. The macro-scale outcomes of efficiency and emissions are directly affected by the micro-scale phenomena with the combustion.

In the matter of the taxonomic analysis technique there are at least two threads of future work. One is winnowing; the other is application. In many cases several graphs gave essentially the same taxonomic information. The random graphs derived from the same regular graph seem to yield essentially the same performance as their progenitor and hence provide no additional taxonomic information. Reducing the set of graphs will proportionally reduce the amount of effort required to place a new problem into the classification.

The second thread is that of application. The current collection of test problems is large relative to the limitations of authors’ time and journal space but small relative to the task of assessing the utility of the taxonomy in guiding evolutionary algorithm design. A program

of predicting which graphs are good for a problem given the behavior of its smaller cases or the problems known characteristics will be followed.

Issues we have not treated at all in this study that merit attention are the impact of population size, of representation, and of smart initialization. Preliminary results indicate a substantial interaction of all three of these with graphs.

## 9 Acknowledgments

This research was supported in part by a grant from the National Energy Technology Laboratory of the U. S. Department of Energy. We would like to thank the members of the Iowa State Complex Adaptive Systems program for helpful comments and discussions.

## References

- [1] D. L. Ackley and M. L. Littman. A case for distributed Lamarckian evolution. Working Paper, Cognitive Science Research Group, Bellcore, New Jersey, 1992.
- [2] D. Ashlock. GP-Automata for dividing the dollar. In *Proceedings of the 1997 Congress on Evolutionary Computation*, pages 18–26, San Francisco, 1997. Morgan Kaufmann.
- [3] D. Ashlock, J. Walker, and M. Smucker. Graph based genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1362–1368, San Francisco, 1999. Morgan Kaufmann.
- [4] Dan Ashlock, Ling Guo, and Fang Qiu. Greedy closure genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1296–1301, Piscataway, NJ, 2002. IEEE Press.
- [5] Dan Ashlock and James I. Lathrop. A fully characterized test suite for genetic programming. In *Evolutionary Programming VII*, pages 537–546, New York, 1998. Springer-Verlag.
- [6] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming : An Introduction*. Morgan Kaufmann, San Francisco, 1998.
- [7] J. Bebernes and D. Eberly. *Mathematical Problems from Combustion Theory*. Springer-Verlag, New York, 1989.
- [8] Kenneth M. Bryden, Daniel A. Ashlock, Douglas S. McCorkle, and Gregory L. Urban. Optimization of heat transfer utilizing graph based evolutionary algorithms. *International Journal of Heat and Fluid Flow*, 24(2):267–277, 2003.
- [9] H.S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids, 2nd edition*. Oxford University Press, London, 1959.
- [10] L. J. Fogel, A. J. Owens, and M. J. Walsh. Intelligent decision making through a simulation of evolution. *Behavioral Science*, 11(4):253–272, 1965.

- [11] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.
- [12] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, Mass, 1997.
- [13] Kenneth A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
- [14] SA Kauffman. *The Origins of Order*. Oxford University Press, New York, 1993.
- [15] Motoo Kimura and James Crow. On the maximum avoidance of inbreeding. *Genetical Research*, 4:399–415, 1963.
- [16] Kenneth Kinneer. *Advances in Genetic Programming*. The MIT Press, Cambridge, MA, 1994.
- [17] John R. Koza. *Genetic Programming*. The MIT Press, Cambridge, MA, 1992.
- [18] Ernst Mayr and Peter D. Ashlock. *Principles of Systematic Zoology*. McGraw-Hill, New York, 1991.
- [19] Robert McEliece. *The Theory of Information and Coding*. Addison-Wesley, Reading, Mass, 1977.
- [20] Heinz Mühlenbein. Darwin’s continent cycle theory and its simulation by the prisoner’s dilemma. *Complex Systems*, 5:459–478, 1991.
- [21] F. Qiu, L. Guo, T.J. Wen, D.A. Ashlock, and P.S. Schnable. Dna sequence-based bar-codes for tracking the origins of ests from a maize cdna library constructed using multiple mrna sources. *Plant Physiology*, 133:475–481, 2003.
- [22] Craig Reynolds. An evolved, vision-based behavioral model of coordinated group motion. In Jean-Arcady Meyer, Herbert L. Roiblat, and Stewart Wilson, editors, *From Animals to Animats 2*, pages 384–392. MIT Press, 1992.
- [23] H. Schlichting. *Boundary Layer Theory, 7th edition*. McGraw-Hill, New York, 1979.
- [24] P.H. A. Sneath and R. R. Sokal. *Numerical Taxonomy; the Principles and Practice of Numerical Classification*. W.H. Freeman, San Francisco, 1973.
- [25] D. L. Swofford. PAUP\*. phylogenetic analysis using parsimony (\*and other methods). version 4. Sinauer Associates, Sunderland, Massachusetts., 2002.
- [26] D.L. Swofford, G.J Olsen, P.J. Waddell, and D.M.Hillis. Phylogenetic inference. In D. Hillis, C. Moritz, and B. Mable, editors, *Molecular Systematics, second edition*. Sinauer, Sunderland, MA., 1996.
- [27] Gilbert Syswerda. A study of reproduction in generational and steady state genetic algorithms. In *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

- [28] Andrea Toffolo and Ernesto Benini. Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation*, 11(2):151–167, 2004.
- [29] G.L. Urban, K. M. Bryden, and D. Ashlock. Engineering optimization of an improved plancha stove. *Energy for Sustainable Development*, 6(2):5–15, 2002.
- [30] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ 07458, 1996.
- [31] D. Whitley, K. Mathias, and Rana J. Dzubera. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276, 1996.
- [32] D. Whitley, S. Rana, and R. Heckendorn. Island model genetic algorithms and linearly separable problems. In D. Corne and J. Shapiro, editors, *Proceedings of AISB Workshop on Evolutionary Computation*, pages 109–125, New York, 1997. Springer-Verlag.
- [33] Darrel Whitley. The genitor algorithm and selection pressure: why rank based allocation of reproductive trials is best. In *Proceedings of the 3rd ICGA*, pages 116–121. Morgan Kaufmann, 1989.
- [34] Sewall Wright. *Evolution*. University of Chicago Press, 1986. Edited and with introductory Materials by William B. Provine.