Spectral Techniques for Graph Bisection in Genetic Algorithms

Jacob G. Martin
University of Georgia
Computer Science
Athens, GA, 30601, USA
martin@cs.uga.edu

ABSTRACT

Various applications of spectral techniques for enhancing graph bisection in genetic algorithms are investigated. Several enhancements to a genetic algorithm for graph bisection are introduced based on spectral decompositions of adjacency matrices of graphs and subpopulation matrices. First, the spectral decompositions give initial populations for the genetic algorithm to start with. Next, spectral techniques are used to engineer new individuals and reorder the schema to strategically group certain sets of vertices together on the chromosome. The operators and techniques are found to be beneficial when added to a plain genetic algorithm and when used in conjunction with other local optimization techniques for graph bisection. In addition, several world record minimum bisections have been obtained from the methods described in this study.

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition-Clustering[algorithms, similarity measures]

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic algorithm, singular value decomposition, graph bisection, graph partitioning, spectral bisection, genetic engineering, reduced rank approximation

1. INTRODUCTION

The technique of singular value decomposition (SVD) has proven itself valuable in several different problem domains: data compression [17], image recognition and classification [20], chemical reaction analysis [41], document comparison [14, 7], cryptanalysis [39, 46], and genetic algorithms [37, 36]. Although these domains are quite different in some aspects, each can be reduced to the problem of ascertaining or ranking relevance in data. Intuitively, the concept of relevance depends critically on the nature of the problem at hand. SVD provides a method for mathematically discovering correlations within data. The focus of this work is to investigate several possible methods of using SVD in a genetic algorithm to better solve the minimum graph bisection problem.

SVD is useful when bisecting certain types of graphs. To obtain a bisection of a graph, SVD is performed directly on the 0,1 adjacency matrix of the graph to be bisected. Next, an eigenvector is chosen and its components are partitioned based on the median of all of the components. Given that each component of an eigenvector represents a vertex of the graph, a partitioning of the graph is achieved. The process of using eigenvectors to bisection graphs is called *spectral bisection*. The technique's roots stem from the works of Fiedler [19], who studied the properties of the second smallest eigenvector of the Laplacian of a graph, and Donath and Hoffman [15], who proved a lower bound on the size of the minimum bisection of the graph.

In addition to applying SVD directly to graphs, it is also used in several ways to guide the search process of a Genetic Algorithm (GA). SVD helps guide the search process of the GA by identifying the most striking similarities between genes in the most highly fit individuals of the optimization history. The GA's mutation operator is then restricted to only modify the locus of the genes corresponding to these striking similarities. In addition, individuals are *engineered* out of the discovered similarities between genes across highly fit individuals. The genes are also reordered on a chromosome to group similar genes closer together on a chromosome. The heuristics show remarkable performance improvements. In addition, the performance achieved is magnified when the heuristics are combined with each other. As further evidence for the applicability of these new heuristics, several world record minimum bisections have been obtained from the genetic algorithm described in this paper.

The first section gives background information on the graph bisection problem, genetic algorithms, and SVD. The second section discusses the implementation details for the genetic algorithm. Section three describes the spectral heuristics that augment the standard GA. The fourth section gives experimental evidence for the applicability of the operators described. The last two sections provide future research ideas and a summary of the results.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

2. BACKGROUND

2.1 Minimum Graph Bisection

2.1.1 Problem Statement

A bisection of a graph $G = (V, E)$ with an even number of vertices is a pair of disjoint subsets $V_1, V_2 \subset V$ of equal size. The cost of a bisection is the number of edges $(a, b) \in E$ such that $a \in V_1$ and $b \in V_2$. The minimum graph bisection problem takes as input a graph G with an even number of vertices, and returns a bisection of minimum cost.

The minimum graph bisection problem arises in many important scientific problems. Several examples include the splitting of data structures between processors for parallel computation, the placement of circuit elements in engineering design, and the ordering of sparse matrix computations [11]. The minimum graph bisection problem has been shown to be NP–Complete [22], making it a prime candidate for research and study.

2.1.2 Literature Review

Many heuristics have been developed for this problem. Frieze and McDiarmid provide an analysis of the performance of algorithms on random graphs [21]. Perhaps the best known heuristic is the Kernighan–Lin heuristic [32, 10]. The Kernighan–Lin heuristic has a time complexity of $O(n^3)$ and is P–Complete [45, 27]. Fiduccia and Mattheyses gave a simplification of the Kernighan–Lin heuristic that has time complexity $\Theta(E)$ [18]. The efficiency is gained by sorting data using a method called the bucket sort. A simulated annealing approach is used by Johnson *et al.* [29]. Spectral techniques for graph bisection are motivated by the work of Fiedler [19]. Indeed, spectral techniques are often used to enhance graph algorithms [1, 43, 2, 5]. Donath and Hoffman are among the first to suggest using spectral techniques for graph partitioning [15]. Alpert and Yao showed that more eigenvectors may help improve results [3]. Their main result showed that when all eigenvectors are used, the min–cut graph partitioning and max-sum vector partitioning problems objectives are identical. Graph partitioning with genetic algorithms has been studied extensively [35, 12, 33, 48, 48]. Most GA methods incorporate other algorithms and heuristics, such as spectral partitioning or Kernighan–Lin. Singular value decomposition has also proved to be a useful tool when clustering graphs [16, 31]. However, this paper contains one of the first attempts to combine these results, providing strategies for using singular value decomposition in a genetic algorithm for the minimum graph bisection problem.

2.2 Genetic Algorithms

2.2.1 Background and Terminology

Genetic Algorithms (GAs) are search and optimization methods that mimic natural selection and biological evolution to solve optimization and decision problems. The book by David Goldberg [24] provides a thorough introduction to the field of Genetic Algorithms. A brief overview of genetic algorithms and some definitions of terminology follow.

A *chromosome* is a sequence of gene values. In this paper, each gene will usually have a value of either a zero or one. A potential solution to a problem is represented by a chromosome. For graph problems, the number of vertices is the size of the chromosome. A *schema* is a pattern of genes consisting of a subset of genes at certain gene positions. If n is the size of a chromosome, a *schema* is an n–tuple $\{s_1, s_2, \cdots, s_n\}$ where $\forall i, s_i \in \{0, 1, \star\}$. Positions in the schema that have a $\star$ symbol correspond to don't–care positions. The non-$\star$ symbols are called *specific symbols*, and represent the defining values of a schema. The number of specific symbols in a schema is called the *order*, and the length between the first and last specific symbols in a schema is called the *defining length* of the schema. The schema theorem implies that the smaller the order of a schema, the more copies it will have in the next generation.

Although genetic algorithms do not specifically work with schemata themselves, schemata are a fundamental concept when analyzing the exploratory process of a genetic algorithm. According to the *building block hypothesis* [24, 28], GAs implicitly favor low–order, high–quality schemata. Furthermore, as evolution progresses, the GA creates higher order, high–quality schemata out of low–order schemata. This is partially due to the nature of the crossover operator. The repercussions of this behavior are impetus for the schema reordering algorithms presented in Section 3.4.2.

2.3 Singular Value Decomposition

THEOREM 1. *Let A be an $m \times n$ real matrix with rank r. Then there exists an $m \times n$ diagonal matrix*

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \tag{1}$$

where the diagonal entries of D are the first r singular values of A, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, and there exist an $m \times m$ orthogonal matrix U and an $n \times n$ orthogonal matrix V such that

$$A = U \Sigma V^T \tag{2}$$

The existence and theory of SVD is established by several mathematicians: Beltrami, Jordan, Sylvester, and Schmidt[47]. Stewart provides an excellent survey of the history of discoveries that lead to the theory of the SVD. [49].

2.3.1 Summary

As Theorem 1 states, SVD expresses an $m \times n$ matrix A as the product of three matrices, U, Σ, and V^T. The matrix U is an m x m matrix whose first r columns, u_i $(1 \leq i \leq r)$, are the orthonormal eigenvectors that span the space corresponding to the row auto-correlation matrix AA^T. The last $m - r$ columns of U form an orthonormal basis for the left nullspace of A. Likewise, V is an n x n matrix whose first r columns, v_i $(1 \leq i \leq r)$, are the orthonormal eigenvectors that span the space corresponding to the column auto-correlation matrix A^TA. The last $n - r$ columns of V form an orthonormal basis for the nullspace of A. The middle matrix, Σ, is an m x n diagonal matrix with $\Sigma_{ij} = 0$ for $i \neq j$ and $\Sigma_{ii} = \sigma_i \geq 0$ for $\forall i$. The σ_i's are called the singular values and are arranged in descending order with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. The singular values are defined as the square roots of the eigenvalues of AA^T and A^TA. The SVD can equivalently be expressed as a sum of rank one matrices

$$A = \sum_{i=1}^{r=rank(A)} \sigma_i u_i v_i^T \tag{3}$$

The u_i's and v_i's are the columns of U and V respectively. Using the Golub-Reinsch algorithm [25, 23], U, Σ, and V can be calculated for an m by n matrix in time $O(m^2n + mn^2 + n^3)$.

2.3.2 Reduced Rank Approximations

The magnitudes of the singular values indicate the weight of a dimension. To obtain an approximation of A, all but the $k < r$ largest singular values in the decomposition are set to zero. This results in the formation of a new low-dimensional matrix A_k, of rank k, corresponding to the k most influential dimensions.

$$A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{4}$$

Here, U_k and V_k are the matrices formed by keeping only the eigenvectors in U and V corresponding to the k largest singular values. Eckart and Young's paper is a rediscovery of this property, first proved by Schmidt [47].

3. IMPLEMENTATION DETAILS

Individuals are represented in binary in the following manner. If the i^{th} component of an individual is one, then the i^{th} vertex is placed in the set V_1. Otherwise, if the i^{th} component of an individual is zero, then the i^{th} vertex is put in the set V_2. Notice that individuals are symmetrical in this representation. That is, flipping every bit in one solution gives the exact same bisection. Before the GA starts, the ordering of the vertices is permuted to prevent the results from containing any possible bias on the ordering of the input. Tests are performed using a custom GA, implemented entirely in Java^{TM}. The SVD is computed using LAPACK routines and the Matrix Toolkits for Java^{TM} (MTJ).

3.1 The Genetic Algorithm

An approach similar to the $(\mu + \lambda)$ evolution strategy is used, with populations of size 100 generating 100 candidate individuals. Reinsertion is achieved by picking the best 100 individuals out of the 200 total parents and children. The results correspond to the average of the best individual at each generation, over 100 different random initial populations. Let $f(x)$ be the value of the function that is being optimized when applied to an individual x. The log fitness of an individual is defined as

$$logfitness(x) = \ln \frac{1}{1 + |f(x) - target|} \leq 0 \tag{5}$$

In this fitness function, the function value $f(x)$ approaches its target (for example the minimum) as the fitness function approaches zero. Individuals with higher fitness represent better solutions than those with lower fitness. An individual with a fitness equal to zero is an exact solution because only then will $f(x) = target$.

A pseudocode listing of the genetic algorithm appears in Figure 1. An explanation of each individual function appears in Sections 3.2, 3.3, and 3.4.

3.2 Local Improvements

Hybrid GAs are those that incorporate a local search procedure during each generation on the new offspring. Local searches are almost always problem specific. Their goal is to improve a candidate solution to a problem by exploring locally around the solution's values. Hybrid GAs are a hybridization of a genetic algorithm with a local search heuristic that is tailored specifically for solving a certain problem. Generally, the performance of the local improvement heuristic is compromised to give a lower time complexity when creating a hybrid GA. This ensures that the local improvement heuristic does not overwhelm the overall running time of the GA.

```
spectral_injection;
do {
  reorder_schema;
  restrict_space;
  for i from 1 to 100 do
      choose parent1 and parent2 from population;
      child = crossover(parent1, parent2);
      mutate(child);
      modified_kernighan_lin(child);
      children.add(child);
  end for;
  replace(population, children);
  engineered = engineer(population);
  replace(population, engineered);
} until(stopping condition)
```

Figure 1: The Hybrid Genetic Algorithm

The implemented GA uses a trimmed down variant of the Kernighan–Lin [32] optimization algorithm. The traditional Kernighan–Lin heuristic has a time complexity of $O(n^3)$ and is not guaranteed to provide the minimum bisection. The algorithm's time complexity is trimmed down in the exact way that is described in Bui and Moon's paper on graph partitioning with a GA [12].

Additionally, the data structures and implementation of the algorithm are done in constant time by using the methods of Fiduccia and Mattheyses [18]. Fiduccia and Mattheyses gave a simplification of the Kernighan–Lin heuristic that has time complexity $\Theta(E)$ [18]. The efficiency is gained by sorting vertex gains using a method called the bucket sort. The addition of the Fiduccia–Mattheyses technique grants the ability to perform a limited, low cost, local search when solving various graph bisection problems.

3.3 Genetic Operators

The mutation rate is set at 12%. A modified mutation method of switching two random genes is implemented to keep the number of ones and zeroes in an individual equal. In the case of subproblem evolution, a gene from the subproblem area is flipped and an opposite gene from the non–subproblem area is also flipped. In plain GAs, the mutation operator simply exchanges the values of two opposite genes. The crossover operator is duplicated from an earlier paper about graph bisection with genetic algorithms by Bui and Moon [12]. The crossover operator is a modified 5-point crossover that considers the symmetric nature of chromosomes in the graph bisection problem. After mutation and crossover, repair operators are utilized to repair the resulting partitions that are not perfectly balanced. The repair process is also implemented in the same manner as Bui and Moon [12].

INPUT = **Adjacency Matrix** A
OUTPUT = **Partition List** P

1. *Compute* **all** of *the eigenvectors of the input matrix* A.
2. **For each** *eigenvector, compute the median of its components and place vertex* i *in the first partition if the* i*'th component of the eigenvector is less than or equal to the median. Otherwise, place vertex* i *in the second partition.*
3. *If necessary, repair the partition to make the number of vertices equal by moving vertices from the bigger partition to the smaller partition until the number of nodes in each partition is equal. Start with nodes that are closer to the other partition in terms of their corresponding eigenvector's component.*
4. *Add the resulting partition to the list of all partitions to return,* P.

Figure 2: Algorithm for Spectral Bisection

3.4 SVD Incorporation

The goal is to discover the genes that are used *similarly* across the best individuals. The ideas to be presented next can be generalized to other methods of determining similarly used genes. However, SVD yields accurate identification of subproblems in optimization problems whose solutions have a block representation [42]. The SVD of a matrix containing the best few (5) individuals in the entire optimization history is computed. Instead of aiming for the sole fittest individual, the GA used SVD to decompose the best few fittest individuals and therefore directed the search towards a *combination* of the best individuals. The computational complexity of computing the SVD may outweigh the complexity of the problem being solved. However, problems with a computationally expensive fitness function may benefit from the methods to be described. In particular, if complex problems can be decomposed into smaller and simpler subproblems, then the benefit will outweigh the cost of computing the SVD. Several time optimizations can also be made to decrease the amount of time used computing the SVD. For example, existing SVDs can be updated using special algorithms for adding or removing rows and columns [6]. Also, random projections are a fast alternative to SVD [42].

3.4.1 Spectral Injection

The technique of spectral bisection provides initial population seedings for the genetic algorithm. Initially, the SVD of the adjacency matrix of the graph to be bisected is computed. All bisections are created using the algorithm in Figure 2. The best spectrally found bisections are initially injected into the population to influence the GA towards good bisections. Experiments with this method show that spectral injection gives the GA a tremendous head start in comparison to not using it at all. The motivation for using spectral partitioning is that the eigenvalues and eigenvectors of many types of adjacency matrices have been shown to have many relationships to properties of graphs. Moreover, every eigenvalue and eigenvector of a matrix can be computed efficiently in polynomial time. Therefore, eigenvalues and eigenvectors are prime candidates for constructing efficient algorithms for solving various intractable graph problems.

The relationships between the spectrum of a graph (which are the eigenvalues of its adjacency matrix) and the properties of the graph itself have been popular topics for research and discovery in the last fifty years [13]. The spectrum has been used to help solve the problem of graph isomorphism [50]. Certain eigenvectors of adjacency matrices in several representations sometimes tend to partition its corresponding graph into two halves such that the conductance of the parts is high, but the conductance between parts is low. Eigenvectors have been used to find good minimum cut partitions and to find good colorings for graphs [8, 5, 4]. However, most studies usually only focus on one eigenvector of one representation type for the adjacency matrix. This eigenvector is called the Fiedler vector, and corresponds to the second smallest eigenvalue of the Laplacian [19]. In the context of this paper, spectral bisection is performed on every eigenvector of the adjacency matrix of the graph. The best 100 resulting partitions are then used to seed the genetic algorithm's first population.

3.4.2 Schema Reordering

Due to the nature of the problems addressed, good schema are apt to be destroyed during crossover if the locations forming the schema are scattered apart on the chromosome. To combat the disruptive nature of crossover, chromosomes are reordered to group the similar genes closer together on a chromosome. This helps to create higher–quality schemata with shorter defining lengths. SVD defines the reordering at every generation during optimization. The reordering groups similar genes together, allowing the GA to benefit from the building block hypothesis. This is in contrast to a strategy that only performs an initial schema preprocessing once before the GA for the minimum graph bisection problem starts [12]. It should be noted that this schema reordering technique affects the defining length, but not the order of the schema.

As the building block hypothesis suggests, the computational power of genetic algorithms largely comes from manipulating the solutions of subproblems, i.e., building blocks. Hence, identifying subproblems has been a center of many subfields within genetic and evolutionary computation. Three examples of related fields that should be studied to better connect the use of SVD to current GA research are Linkage Learning [26], Probabilistic Model Building Genetic Algorithms [44], and Learnable Evolution Models [38].

3.4.3 Restricted Mutation

The mutation operator is restricted to a strategically chosen subset of the genes. This isolates the search process to the genes in highly fit solutions, facilitating the determination of the local optimum. The subset of genes is chosen by using a SVD process described in a previous paper by Martin[36]. The subset of genes is chosen randomly from the set of all sets of highly correlated genes identified by SVD. The restriction only happened every other generation. This allows the mutation operators to have full access to the entire space of possible chromosomes.

3.4.4 Genetic Engineering

A genetic engineering approach is tested at every generation. First, the rank–2 SVD of 5 to 10 proportionally selected individuals is computed. The number of individuals is chosen uniformly at random. Then, using a process similar to that described in a previous paper [36], a new graph of correlated genes is generated. Specifically, the magnitude of the (i, j) entry in unit scaled matrix $A_2 A_2^T$ determines if an edge appears between vertex i and vertex j in the new graph. If the entry is bigger than 0.9, an edge is created. The vertices in the new graph represent represent the original graph's vertices but are instead connected to those vertices that the top 5 to 10 best individuals collectively believe should be clustered into the same side of the bisection. Ironically, a minimum bisection of the new graph gives a good approximation of the combination of the best individual minimum bisections in the original graph. To keep the problem from becoming self referentially intractable, an approximate minimum bisection of the new graph is discovered by running only one iteration of full Kernighan–Lin on a randomly generated individual. If better, the newly generated individual replaces the worst individual in the current population.

3.4.5 Low Rank Approximations

Two forms of the SVD are tested. The first is the full rank version of the SVD. The second is based on the reduced rank version, where all but the first k largest singular values are set to zero, giving A_k. As expected, the reduced rank strategies generally discover the subproblems more efficiently than the full rank versions. This is due in part to the theoretical results mentioned in the probabilistic analysis of reduced rank spectral clustering in a well known paper by Papadimitriou *et al.* [42]. The performance may also have improved because, in the application domains tested, the GA is only seeking one block in the solution space. Reduction to a lower rank correctly directs the search towards the correct block because a lower value of k in A_k increases the cosines of the angles between vectors of similar types [9]. Another reason may be that in comparison with higher rank reductions, lower rank reductions are less restrictive and will identify larger subsets of related genes as the rank is reduced. Therefore, lower rank reductions allow the restrictive mutation and crossover operators to have more freedom during exploration. However, lowering the rank too much may not always increase the performance because all genes will be seen as similar to all other genes.

4. EMPIRICAL RESULTS

Intuitively, the number of generations it takes to find a solution is the greatest factor in proving a genetic algorithm's performance. It is also illuminating to compare the average best individual at every generation. This allows one to discover the convergence properties of a particular configuration of the GA. The results are based on average of the best or average individual fitness at each generation over 100 independent runs of the GA.

The GA is compared with various combinations of genetic operators, local search functions, and techniques used for solving the minimum graph bisection problem. To assess the amount of benefit achieved using the SVD heuristics, comparisons are made to a plain genetic algorithm that does not use the SVD heuristics. The plain GA serves as a strawman for the SVD methods. Unless otherwise indicated, the plain GA is augmented with local search. In some cases, the spectral injection heuristics discussed earlier are also included in the plain GA.

4.1 Minimum Graph Bisection

4.1.1 Graph Types

Geometric and caterpillar graphs are studied and used as the basis of experiment. A description of the notation and construction details of each type of graph follows. Experiments on other types of graphs (random, grid, path, and highly clustered) give similar results but are not included in this paper due to space restrictions.

1. **Random Geometric Graphs** — $U_{n.d}$: A graph on n vertices created by associating n vertices with different locations on the unit square. The unit square is located in the first quadrant of the Cartesian Plane. Therefore, each vertex's location is represented by a pair $(x, y) \in \Re$ for some $0 \leq x, y \leq 1$. An edge is created between two vertices if and only if the Euclidean distance between the two is d or less. These graphs are defined and tested in the simulated annealing study by Johnson *et al.* [29].

2. **Caterpillar Graphs** — CAT_n : A caterpillar graph on n vertices. Two of the vertices are the head and tail of the caterpillar. Next, $\lfloor \frac{(n-2)}{7} \rfloor$ vertices are chosen to represent the discs in the spine of the caterpillar. To each of these vertices is then attached 6 legs from the remaining $(n-2) - \lfloor \frac{(n-2)}{7} \rfloor$ vertices. The caterpillar graphs considered here have an even number of discs in their spine. This implies that the only possible caterpillars have an even number of vertices with

$$\begin{aligned} n &\in \{(i * 6 + i) + 2 : \forall i \geq 2, i \bmod 2 = 0\} \\ &= \{16, 32, 44, \cdots, 352, \cdots\} \end{aligned}$$

   Here, i represents the total number of discs on the spine. Caterpillar graphs have been shown to be very difficult for standard graph bisection algorithms such as Kernighan–Lin [30, 12]. In addition, the minimum bandwidth problem for caterpillars with hair length 3 is NP–Complete [40].

4.1.2 Discussion

The SVD engineering technique is similar in function to a voting scheme. Evidence for this is provided in Figure 3. The voting technique takes the top 5 to 10 proportionally selected individuals and calculates a vote for which side of the bisection each vertex should belong. Before the vote is counted, every bit in an individual's representation is flipped if and only if the first bit is zero. This helps account for the symmetrical nature of candidate solutions for the graph bisection problem. Next, the vote is taken and a new individual is engineered from the resulting votes. It can be seen from Figure 3 that the cut solution qualities of the generated individuals are similar, but that the SVD engineering performs better. In addition, the average generated cut size increases rapidly in the first 10 generations, and then rapidly decreases. This indicates that the GA needs some time to discover good basis individuals for engineering. Finally, the hypothesis that the SVD engineering technique acts as a

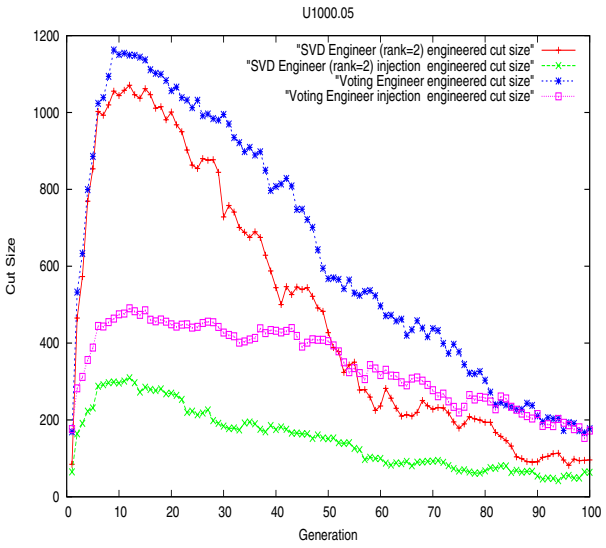

Figure 3: Cut size comparisons between voting and SVD engineering approaches with *no* local improvements for Bui's $U_{1000.05}$

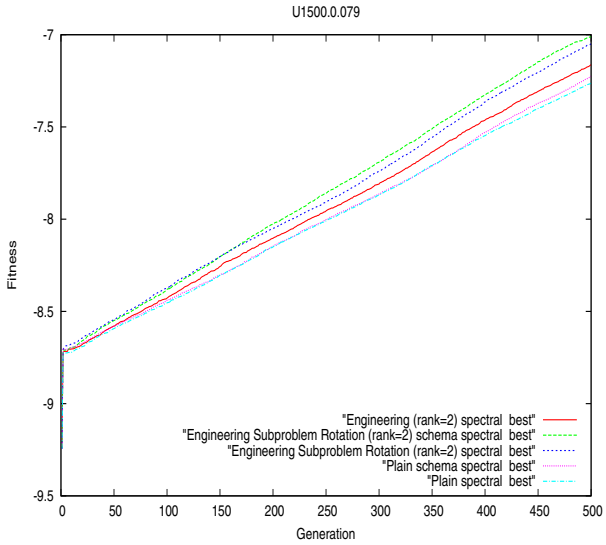

Figure 4: Average best fitness per generation when using spectral injection and the modified KL approach on $U_{1500.0797788}$

shared approximate vote is validated by the similarities between the optimization curves for voting and SVD.

Figure 4 depicts the results from an experiment that compares most of the described heuristics. In addition, local searches are performed at each generation. Spectral injection, subproblem restriction and rotation[36], engineering, and schema reordering are all verified to positively influence the performance of the genetic algorithm separately for this graph. Figure 5 shows that the performance increase is much more dramatic when the the local search operator is not performed. However, Figure 6 shows that when the Kernighan–Lin local improvement is used with graphs for which KL does not perform well (caterpillars), the SVD techniques outperform the plain GA by a more significant margin. This indicates that SVD may be a viable alternative to KL and that it can be successfully paired with KL to provide additional performance.

In addition to the previous experiments, several record size minimum bisections for real world graphs are found using the techniques described in this paper. The three graphs for which record bisections are achieved are named data, add20 (a 20 bit adder), and bcsstk33 (a statics module of a pin boss). These results are listed in Chris Walshaw's graph partitioning archive located at `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/` [48].

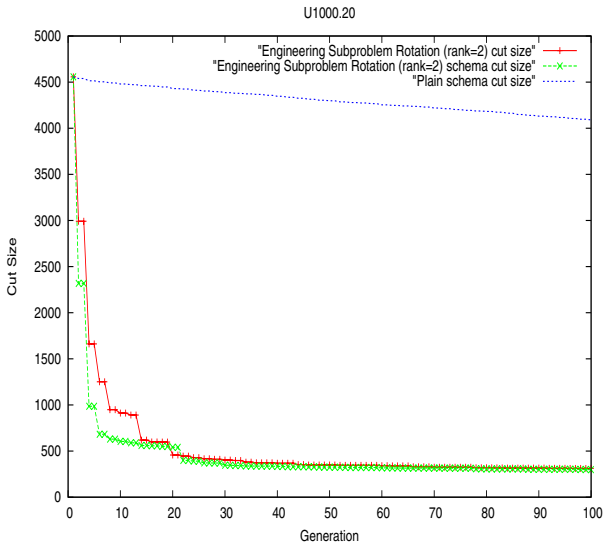

Figure 5: Cut size results corresponding to no spectral injection and no local improvements for Bui's $U_{1000.20}$

5. FUTURE WORK

The positive benefits of adding SVD to KL based algorithms have been explored in this paper. Analysis of variance (ANOVA) tests should be conducted to better prove that the presented methods work well in combination with each other. ANOVA tests should also be used to better isolate the benefits of each operator.

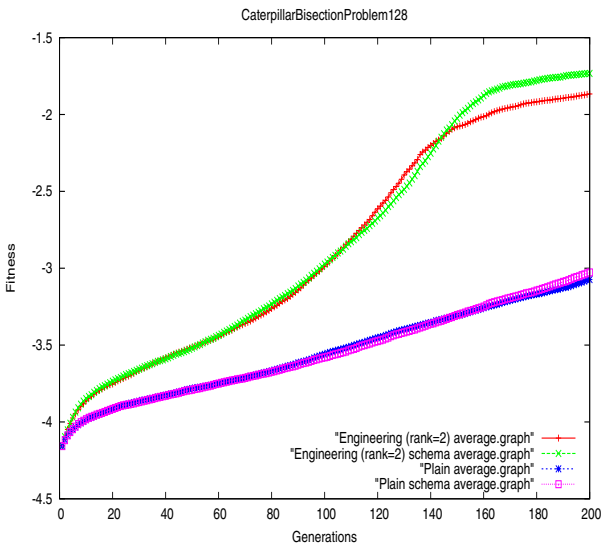

Figure 6: Average population fitness per generation for using a modified KL local improvement on CAT_{128}

A graph bisection technique called Lock-Gain (LG) partitioning was recently introduced by Kim and Moon [34]. LG partitioning extends KL by using a new tie-breaking strategy that intelligently selects the best highest gain vertex to exchange during one pass of KL. In addition, the gain of a vertex is calculated in a manner that takes into account vertices that have already been moved. The various combinations of the SVD operators and techniques described herein should be investigated in conjunction with the lock gain partitioning method and other metaheuristics for minimum graph bisection.

Additional operators and procedures based on spectral information should be considered. For example, a spectral crossover operator can be used to give a linkage probability to chromosomes that is related to the information provided by the spectral decomposition of the adjacency matrix of the graph to be bisected. This type of operator is justified because many of the eigenvectors of several adjacency matrix representations tend to group vertices together that should be placed in the same partition. The distance between the valuations for the vertices in the eigenvectors can be used to determine the probability that two genes travel together during crossover. Another example is the possibility of using spectral information to enhance tie–breaking strategies in LG and KL. Finally, the possible benefits of starting with a spectral population should be examined in detail.

6. CONCLUSION

This paper presents several methods for enhancing the performance of a genetic algorithm to better solve the minimum graph bisection problem. First, spectral techniques are employed to seed the initial population with good solutions. SVD is also used to engineer, restrict the locus of mutation, and to define schema reorderings based on approximations of highly fit individuals. The new operators and techniques are investigated with respect to their consequences on performance in conjunction with a hybridized genetic algorithm employing the Kernighan–Lin local search operator and other operators described in previous research papers [12, 36]. All of the introduced techniques are shown to be beneficial to the genetic algorithm. Empirical results obtained from the combination and application of these new heuristics are encouraging.

7. REFERENCES

[1] N. Alon. Spectral techniques in graph algorithms (invited paper). In *LATIN'98: theoretical informatics (Campinas, 1998)*, volume 1380 of *Lecture Notes in Comput. Sci.*, pages 206–215. Springer, Berlin, 1998.

[2] C. J. Alpert, A. B. Kahng, and S.-Z. Yao. Spectral partitioning with multiple eigenvectors. *Discrete Appl. Math.*, 90(1-3):3–26, 1999.

[3] C. J. Alpert and S.-Z. Yao. Spectral partitioning: The more eigenvectors, the better. In *DAC*, pages 195–200, 1995.

[4] B. Aspvall and J. R. Gilbert. Graph coloring using eigenvalue decomposition. *SIAM J. Algebraic Discrete Methods*, 5(4):526–538, 1984.

[5] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM J. Algebraic Discrete Methods*, 3(4):541–550, 1982.

[6] M. W. Berry, Z. Drmač, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, 41(2):335–362 (electronic), 1999.

[7] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Rev.*, 37(4):573–595, 1995.

[8] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis (extended abstract). In *FOCS*, pages 280–285, 1987.

[9] M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In *International Workshop On Artificial Intelligence and Statistics*, Jan. 2003. International Workshop On Artificial Intelligence and Statistics, January 2003.

[10] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[11] T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *PPSC*, pages 445–452, 1993.

[12] T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. Comput.*, 45(7):841–855, 1996.

[13] D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs*, volume 87 of *Pure and Applied Mathematics*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1980.

[14] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[15] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*,

17:420–425, 1973.

[16] P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1–3):9–33, 2004.

[17] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1972.

[18] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *DAC '82: Proceedings of the 19th conference on Design automation*, pages 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[19] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973.

[20] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, Sept. 1995.

[21] A. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Structures Algorithms*, 10(1–2):5–42, 1997.

[22] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP–complete graph problems. *Theoret. Comput. Sci.*, 1(3):237–267, 1976.

[23] G.Golub and C. Reinsch. *Handbook for Matrix Computation II, Linear Algebra*. Springer–Verlag, 1971.

[24] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison–Wesley, Reading, Mass., 1989.

[25] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1996.

[26] G. R. Harik and D. E. Goldberg. Learning linkage. In *Foundations of Genetic Algorithms*, pages 247–262, 1996.

[27] B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, 1995.

[28] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, Mich., 1975.

[29] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. part i, graph partitioning. *Oper. Res.*, 37(6):865–892, 1989.

[30] C. A. Jones. *Vertex and edge partitions of graphs*. PhD thesis, Pennsylvania State University, University Park, PA, USA, 1992.

[31] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *J. ACM*, 51(3):497–515 (electronic), 2004.

[32] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell Systems Journal*, 49:291–307, 1972.

[33] J.-P. Kim and B.-R. Moon. A hybrid genetic search for multi-way graph partitioning based on direct partitioning. In L. S. *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO–2001)*, pages 408–415, San Francisco, California, USA, 7–11 July 2001. Morgan Kaufmann.

[34] Y.-H. Kim and B. R. Moon. Lock-gain based graph partitioning. *J. Heuristics*, 10(1):37–57, 2004.

[35] H. S. Maini, K. G. Mehrotra, M. Mohan, and S. Ranka. Genetic algorithms for graph partitioning and incremental graph partitioning. Technical Report CRPC–TR94504, Center for Research on Parallel Computation, Rice University, Houston, TX, 1994.

[36] J. G. Martin. Subproblem optimization by gene correlation with singular value decomposition. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1507–1514, New York, NY, USA, 2005. ACM Press.

[37] J. G. Martin and K. Rasheed. Using singular value decomposition to improve a genetic algorithm's performance. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1612–1617, Canberra, 8-12 Dec. 2003. IEEE Press.

[38] R. S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Mach. Learn.*, 38(1–2):9–40, 2000.

[39] C. Moler and D. Morrison. Singular value analysis of cryptograms. *Amer. Math. Monthly*, 90(2):78–87, 1983.

[40] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP–complete. *SIAM J. Algebraic Discrete Methods*, 7(4):505–512, 1986.

[41] B. Noble and J. W. Daniel. *Applied Linear Algebra*. Prentice–Hall, Englewood Cliffs, NJ, USA, third edition, 1988.

[42] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: a probabilistic analysis. *J. Comput. System Sci.*, 61(2):217–235, 2000.

[43] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[44] K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practise*, chapter 13, pages 205–220. Kluwer, 2003.

[45] J. E. Savage and M. G. Wloka. Parallelism in graph–partitioning. *J. Parallel Distrib. Comput.*, 13(3):257–272, 1991.

[46] B. R. Schatz. Automated analysis of cryptogram cipher equipment. *CRYPTOLOGIA*, 1(2):116–142, Apr. 1977.

[47] E. Schmidt. Zur Theorie der linearen und nichtlinearen Integralgleichungen. *Math. Ann.*, 63(4):433–476, 1907.

[48] A. J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph–partitioning. *J. Global Optim.*, 29(2):225–241, 2004.

[49] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Rev.*, 35(4):551–566, 1993.

[50] E. R. van Dam and W. H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra Appl.*, 373:241–272, 2003.