Evolutionary Algorithms for Vertex Cover

Isaac K. Evans
Department of Electrical and Computer Engineering
University of Iowa
Iowa City, IA 52242
ikevans@eng.uiowa.edu

Abstract. This paper reports work investigating various evolutionary approaches to vertex cover (VC), a well-known NP-Hard optimization problem. Central to each of the algorithms is a novel encoding scheme for VC and related problems that treats each chromosome as a binary decision diagram. As a result, the encoding allows only a (guaranteed optimal) subset of feasible solutions. The encoding also incorporates features of a powerful traditional heuristic for VC that allow initial EA populations to be seeded in known promising regions of the search space. The resulting evolutionary algorithms have displayed exceptionally strong empirical performance on various vertex cover, independent set, and maximum clique problem classes.

1 Introduction

Vertex cover (VC) is a well-known combinatorial optimization problem with practical applications in, for example, computer networking and scheduling. VC formulated as a decision problem is NP-Complete [9]. The NP-Hard optimization problem is simply to find a cover of minimum size in the given graph, i.e., given an undirected graph G consisting of nodes V and edges E, find a minimum size subset of nodes $V_C \subseteq V$ such that every edge in E is incident on at least one of the nodes in V_C. Excellent overviews of VC approximation algorithms can be found in [11] and [13].

Vertex cover is also closely related to the important problems of independent set (IS) and maximum clique (MC). In this paper EAs for VC will be directly compared against GAs for IS. As is always true, however, appropriate caution must be exercised when considering a transformation between problem types. Performance bounds may not remain useful after transformation [9]. Additionally, the conversion may not preserve the practical tractability of a particular problem instance or class. As a result, section 4 includes only limited comparisons against previously reported GAs for MC [5] and only in cases where independent set results have also been reported.

2 Vertex Cover Heuristics

Many traditional heuristics for vertex cover are known. Perhaps the most obvious is the greedy algorithm, in which the vertex of maximum remaining degree (incident on the maximum number of edges) is repeatedly removed from the graph and added to the cover until all edges are covered. While intuitive, the algorithm actually performs poorly on many classes of graphs and has no fixed performance bounds. An alternative algorithm is focused on finding a maximal matching in the graph: while edges remain, choose an arbitrary edge, add both endpoints to the cover, and remove both vertices from the graph. Because each selected edge must be covered by at least one of its endpoints, the algorithm has a fixed performance bound of 2, i.e., the resulting cover is at most twice the optimum cover. A number of other traditional approaches to VC, typically with performance bounds close to 2, have been reported and are discussed in [11] and [13]. A local-ratio approximation algorithm with best known bound of $2 - \log\log n / 2\log n$ relies on the repeated removal of sub-graphs, specifically small odd cycles or triangles [3].

Pramanick describes a novel stochastic optimization approach to VC [15] as a "practical method for computing vertex covers for large graphs" [14]. Parallel Dynamic Interaction (PDI) is an inherently parallel optimization methodology that exploits the non-deterministic behavior of shared-memory multiprocessors as the stochastic input to the algorithm. Individual processors search for covers in subsets of the complete graph. Global covers are formed by dynamic interaction between processors (reminiscent of a commodities trading process), in which the non-deterministic time of completion resolves competition between local solutions. PDI displayed very strong empirical performance on two problem classes (described later) when compared against traditional algorithms [15]. Previously reported neural network [16] and genetic algorithm [12] approaches to VC provide improved quality solutions on certain of the benchmark problems used in the PDI study, albeit with significantly increased computation times.

The work reported in this paper makes use of an alternate traditional heuristic for vertex cover, labeled OBIG [7]. The opportunistically-bounded inverse greedy algorithm is described by the following pseudo-code:

```
OBIG: V_C ←∅, B = 0
    while edges remain {
        while vertices of degree 1 remain {
            select d1 vertex; add adjacent vertex to V_C
            remove d1 and adjacent vertex from G
            match removed vertices;  B = B + 1
        }
        select arbitrary vertex of minimum degree ≥ 2
        add all adjacent vertices to V_C
        remove selected and all adjacent vertices from G
        opportunistically match removed vertices;  B = B + m    { m = # pairs,  m ≥ 1 }
    }
    Output V_C, relative performance bound | V_C | / B
```

Rather than choosing the vertex of maximum remaining degree to include in a cover, the inverse greedy approach to VC chooses the vertex of minimum remaining degree to exclude from the cover. Each excluded vertex forces adjacent vertices into the cover to maintain feasibility of the solution. This inverse greedy technique is analogous to the greedy algorithm for independent set (the "inverse" problem), attributed to Paul Erdös (see [11]). OBIG also relies on the observation that a vertex of degree 1 need never be included in an optimal cover, because the adjacent vertex may always be included instead without loss of optimality. (OBIG also implicitly ignores vertices of degree 0). This methodology effectively reduces the size of the search space - only a subset of feasible solutions (guaranteed to include optimal solutions) need be considered. It is important to note that this feature embodies more than a simple preprocessing step on the input graph, because degree 1 vertices are removed automatically at each iteration of the algorithm whenever uncovered in the residual graph. A simple greedy algorithm for independent set also optimally handles vertices of degree 0 and 1 [10]; OBIG's explicit removal of degree 1 vertices lends itself to the BDD encoding described in the next section.

While not critical to the cover formulation, the OBIG algorithm also provides a constructive bound by opportunistically matching vertices as they are removed from the graph over the course of the algorithm. Any optimum cover must include at least one vertex for each matched pair. Assuming appropriate data structures, OBIG is an $O(|V|+|E|)$ algorithm, or linear with graph size. The key insight supporting this conclusion involves the sort implied by minimum degree vertex selection, which can be performed in linear time because the degree of any vertex is integral with maximum $(|V|-1)$. OBIG shares some features with a previously reported search procedure (GRASP) for IS [8]. GRASP orders vertices by degree but incorporates local search to select a vertex of "low" degree to add to the independent set, as well as in a preprocessing phase. This exponential local search is arbitrarily limited in scope.

3 EAs for VC and IS

The importance of the encoding of the underlying problem is well-known for EA optimization approaches. In the case of vertex cover and related problems, the most obvious approach is a direct encoding in which each bit of a binary chromosome of length $|V|$ defines the presence or absence of the corresponding vertex in the cover. Most previously reported GAs for VC, IS, MC and related problems have used this direct encoding style, e.g. in [1],[2],[5] and [12]. An obvious drawback of the approach is that the direct encoding allows infeasible solutions. The bitstring of all zeros, for example, corresponds to an invalid candidate cover with no vertices. More importantly, the encoding allows infeasible solutions to be created from existing feasible solutions using typical mutation and recombination operators found in EAs. Previously reported work has typically attacked the problem with penalty functions, in which the fitness of solutions that violate constraints is reduced, or with validation procedures, in which infeasible solutions are corrected to some "nearby" valid solution. Kommu compared two validation techniques with three different penalty methods [12]. After lengthy empirical investigation on the VC problem sets of [15], he concluded that the validation procedure based GAs performed somewhat better than the various penalty methods as well as providing significantly better solutions than traditional or PDI heuristics. Bäck and Khuri used a direct encoding with a graded penalty function in their GA for IS [2]. They tested their algorithm with randomly constructed graphs, as well as a class of scalable regular graphs. Aggarwal and co-authors use a direct encoding along with a domain-specific "optimized crossover" operator in their GA for IS [1]. Their crossover operation incorporates a local search (NP-Hard in general) along with a validation procedure to correct infeasible child solutions.

The approach taken in this paper uses a fitness evaluation skeleton based on the traditional OBIG heuristic with an embedded binary decision diagram (BDD) encoding. The BDD encoding and fitness evaluation can be described as follows:

```
BDD-IG: V_C ← ∅, i = 0
    while edges remain {
        while vertices of degree 1 remain {
            select d1 vertex; add adjacent vertex to V_C
            remove d1 and adjacent vertex from G
        }
        select next  vertex v of minimum degree ≥ 2
        if chromosome bit i = 0 { • exclude v •
            add all adjacent vertices to V_C ; remove v and all adjacent vertices from G
        }
        else if chromosome bit i = 1 { • include v •
            add v  to V_C ; remove v  from G
        }
        i = i + 1
    }
    Output fitness = |V_C|
```

Each bit of the chromosome corresponds not to a particular vertex, but rather to the next decision to be made during synthesis of a feasible cover. Because only feasible solutions are considered (actually only a subset of feasible solutions guaranteed to contain optimal solutions), no penalty functions or validation procedures are required in evolutionary algorithms based on this encoding. The search is naturally limited to (a subset of) the feasible region, which is dramatically smaller than the entire search space for many classes of graphs (although still exponential). These results are summarized by the following theorems [7]:

THEOREM 1. *BDD-IG encodes only feasible covers.*

PROOF. Constructive: each step of the algorithm maintains feasibility of the partial solution. All edges are eventually removed from the graph, and no edge is removed unless at least one of its endpoints has been included in the cover.

THEOREM 2. *BDD-IG cannot encode all feasible covers.*

PROOF. Consider a graph with at least one degree one vertex. Because the degree one vertex will be automatically excluded from the cover, BDD-IG cannot encode the feasible cover consisting of all vertices in the graph.

THEOREM 3. *BDD-IG can encode covers of each optimal (and optimum) size.*

PROOF. Constructive: each step of the algorithm maintains potential optimality of the current partial solution under construction. Consider a selected vertex of degree 1. This vertex is automatically excluded from the cover, but because it is incident on only a single edge, the adjacent vertex may always be included instead without increasing the size of the cover. Consider a selected vertex v. Any optimal cover must either include v, or exclude v. Since both possibilities can be encoded, the potential optimality of the cover under construction is never reduced.

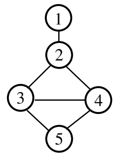

Fig. 1. Example-Th3

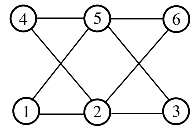

Fig. 2. Misp6

One of the most important features of the BDD-IG (and OBIG) algorithms is their compression of the underlying search space. In fact, the encoding in a given graph is effectively variable length with values between 0 and $|V|$-2 easily demonstrated. Any star graph, for example, requires a chromosome of length 0 because the central vertex is automatically included in the cover as the first arbitrary degree 1 vertex is excluded. A cover for a fully connected graph can be encoded by a string of $|V|$-2 ones (although in this case the cover could also be encoded by a single zero at any locus). In practice, the actual length of a particular encoding is strongly dependent on the average degree of vertices encountered during the cover construction and on the number of zeros in the encoding, i.e., on the number of vertices excluded from the cover, each of which forces multiple vertices to be included in the cover without necessitating additional bits in the chromosome. The actual BDD-IG encoding length also varies with graph instance and class, of course, but is typically much less than the $|V|$ bits required by a direct encoding. One graph examined later, for example, is keller6, which contains 3361 vertices, but requires only ~40 to ~60 bits to encode "good" solutions with populations seeded in known promising regions of the search space. This can be (roughly) viewed as a dramatic reduction in the search space by a factor of $2^{(3361-60)}$ or 2^{3301}. While theorem 3 guarantees that BDD-IG encodes optimal covers within the reduced search space, BDD-IG does not necessarily encode every cover of a given optimal size. The example graph of figure 1 has four different optimum covers of size 3. BDD-IG will automatically exclude vertex 1 and subsequently return one of three optimum covers that include vertex 2. Because vertex 1 is excluded, however, BDD-IG cannot encode the optimum cover {1,3,4}.

Figure 2 displays another illustrative example. Graph misp6 is an example of a class of regular, structured graphs introduced to test a graded penalty function based GA for IS [2]. It is evident from the figure that misp6 has an optimum cover of size 2 (vertices {2,5}) with an equivalent maximum IS of size 4. Bäck and Khuri empirically tested similar graphs using 102 and 202 vertices. In 100 runs each (20K-40K fitness evaluations per run), the penalty function based GA failed to find the optimum for either graph, although it found solutions close to optimal. On the same graphs, OBIG deterministically finds the globally optimum VC or IS, and also guarantees the optimality of the solution with a relative bound of 1.00. (Consider arbitrarily selecting minimum degree vertex 4 first in figure 2. Vertices 2 and 5 are forced into a cover of size 2. The optimum cover is guaranteed to have minimum size 2 because two pairs of vertices can be opportunistically matched, e.g. {4,5} and {1,2}). Similar bounding results can be obtained from an expanded BDD-IG definition that includes opportunistic matching, but it is often more appropriate to simply run OBIG or another algorithm with better bounding performance [7] as an EA post-processing step, from which an effective relative bound can be computed.

Another feature of the BDD-IG encoding critical to the strong experimental results reported later is the ability to seed initial EA populations into known promising regions of the search space. Consider a BDD-IG encoded chromosome of all zeros. At each decision point in the algorithm, the zero bit of the chromosome forces the current selected vertex of minimum degree to be excluded from the cover, and forces all adjacent vertices to be included. This is exactly equivalent to the inverse greedy approach of OBIG. Thus BDD-IG reduces to OBIG when an all-zero chromosome is encountered[1]. Because OBIG is an intuitive heuristic that also displays strong standalone empirical performance, a reasonable a priori conclusion is that BDD-IG encoded chromosomes close to all-zeros should have "good" fitness. Actual results, of course, will be dependent on characteristics of the particular graph or graph class.

As suggested earlier, very few examples of encodings for VC/IS/MC other than direct encodings exist in the literature. Aggarwal provides one recent example of a non-direct encoding applied to IS [1], in which each cover is encoded as a string of |V| random integer keys, upon which normal mutation and crossover operators are applied. Each fitness evaluation requires running a greedy algorithm on the sorted set of keys. Like BDD-IG, Aggarwal's R-Key GA ensures feasibility of candidate solutions after crossover. In empirical testing on some of the DIMACS maximum clique problems, their R-Key GA provided poorer quality solutions than Aggarwal's own optimized crossover GA (OCH). The R-Key GA also required an order of magnitude longer computational times [1], perhaps in part due to the sorting required by each fitness evaluation. Various authors have also suggested biasing initial populations. Kommu, for example, used initial populations for vertex cover that were guaranteed feasible and at least partially minimal [12], while Aggarwal seeds the initial population of OCH with maximal independent sets determined from a greedy algorithm run on randomly ordered vertices of the graph [1].

4 Experimental Results

Both OBIG and BDD-IG encoded evolutionary algorithms have been empirically tested using the VC benchmarks described in [15]. Pramanick investigated two graph classes using PDI: a class of random graphs (irand##) and a class of graphs (sg##) derived from the ISCAS-89 VLSI circuit benchmark suite [4]. The graphs vary in size but include instances with up to 2083 vertices (denoted by ## in the label) and as many as 28799 edges. Pramanick compared PDI algorithm performance against traditional heuristics for VC, including greedy (GH), maximal matching (MM), and two versions of a local-ratio algorithm (LRC/T). She reported very strong PDI performance with PDI dominating the traditional heuristics in solution quality. The basic OBIG algorithm described in section 2 was compared against PDI and the traditional heuristics on the same graph instances. Results for GH, MM, LRC/T and PDI are extracted from [14] and [15]. Each of the traditional algorithms was run "several" times, except the local ratio algorithm, which was run once for each graph.

[1] BDD-IG actually reduces to a single arbitrary instance of OBIG. In OBIG, selection ties are broken arbitrarily (randomly), while in BDD-IG selection ties are broken in a fixed order (arbitrarily chosen for each EA run) to allow an encoded chromosome to effectively determine the total ordering of vertices considered by the algorithm - thus ensuring that the encoding deterministically maps to a particular cover.

Table 1. OBIG performance for large graphs

	$\lvert V_C \rvert$			runtime (seconds)	
graph	GH	PDI	OBIG	PDI	OBIG
sg269	190	177-189	173-178	10.2-23.9	0.0237
irand500	402	369-304	353-360		
sg698	494	465-492	450-452	196.0-238.4	0.142
sg821	465	433-465	392-394	50.5-90.3	0.0591
irand1000a	688	627-701	591-599	70.1-139.6	0.108
irand1000b	795	723-800	698-708	112.9-251.1	0.13
irand1300	1110	1045-1048	1014-1023		
sg1742	1206	1124-1202	1108-1111	557.2-673.8	0.207
sg1770	1114	1049-1129	1063-1077	880.5-929.4	0.232
sg2083	1246	1157-1291	1149-1155	302.1-349.2	0.355

PDI results are based on three or four runs for each of six PDI heuristic/strategy combinations [15]. OBIG results are based on ten runs for each graph.

OBIG solution quality dominated traditional and PDI algorithms on each of the 12 smallest irand and sg graphs, discovering the optimum cover for 11 of 12 graphs. Table 1 shows similar solution quality results for the large irand and sg graphs. The best (minimum) solution found is reported for the GH algorithm, while the range of solutions discovered is reported for PDI and OBIG. As is true for the small graphs, PDI provides excellent results - dominating those of GH (as well as MM and LRT not repeated here). OBIG, however, yields superior covers for each graph except sg1770. More interestingly, OBIG also requires orders of magnitude less time to compute the better quality solutions. Table 1 includes runtimes on the 8 large graphs for which PDI results were reported [15]. OBIG data is averaged over ten runs[2].

The excellent performance of OBIG (or alternatively of the greedy approach to IS) on these vertex cover problem classes is promising, but doesn't directly address the potential utility of BDD-IG. Table 2 reports the results of an initial study made to assess the effectiveness of the encoding within an EA framework. A traditional elitist GA was implemented using the BDD-IG encoding/fitness evaluation function. The GA used the basic generational structure of an SGA, but used binary tournament selection and uniform crossover ($P_x = 0.85$). Because the BDD encoding is effectively variable length, the next-bit mutation rate was arbitrarily fixed at 0.01 for all problems. As suggested in section 3, the initial population was biased towards the all-zero chromosome: each bit was initialized to zero with probability (P_{0init}) = 0.99. The resulting GA (labeled BDDGA in the table) was terminated after 20 generations with population size 20. Ten runs each were made on the 11 graphs (10 large graphs plus irand85) for which OBIG had not already discovered the optimum cover.

Table 2 also includes results reported for some of the same graphs in [12] and [16]. Shrivastava described a Hopfield neural net algorithm (NN) for vertex cover that produced superior quality results to PDI on certain graph instances, but required significant computational time. As described previously, Kommu implemented a

---

[2] A recent OBIG implementation is about 4 times faster than that used during these experiments. Average run times are reported because, unlike PDI, variance between runs is negligible. OBIG data was collected on a 166 MHz Pentium PC while PDI data represents aggregate CPU time on a 14 processor Encore.

Table 2. GA/NN best (minimum) solutions for large graphs

graph	NN	KGA	OBIG	BDDGA
irand85			59	58
sg269	174	172	173	172
irand500	381		353	350
sg698	426	404	450	446
sg821	418	394	392	391
irand1000a	648		591	588
irand1000b	753		698	694
irand1300	1076		1014	1009
sg1742	1170	1109	1108	1100
sg1770	1180	1018	1063	1047
sg2083	1272	1158	1149	1139

variety of GAs for VC and reported excellent solution qualities for some of the same graphs. The best cover discovered by any of his various GAs (KGA) is reported in the table. He did not report computational requirements for the best of the algorithms; however, sample runtimes for GAs yielding poorer quality solutions were reported as 1000s to 10,000s of seconds for the larger graph instances [12]. As can be seen from the table, the BDD-IG encoded GA showed excellent performance on this test suite. BDDGA found better covers than OBIG for each graph, and dominated the best solutions previously discovered using PDI. The simple BDDGA also typically found better covers than any of the various algorithms summarized under NN or KGA. BDDGA solution quality was worse than KGA only for graphs sg698 and sg1770. In the case of graph sg2083, the optimal cover discovered by BDDGA was validated as a globally optimum cover by comparing with the best known bounding results [7]. The strong solution quality obtained with BDDGA still required orders of magnitude less runtime than any of the competing algorithms. For the graphs reported previously in table 1, BDDGA required between 1.95 and 95.0 seconds per run.

A follow-on study was conducted to examine the effectiveness of several potentially more robust evolutionary algorithm approaches to some of the most difficult of these vertex cover problems, including the two for which BDDGA had not already discovered the best known cover. Various EAs were investigated, including traditional elitist serial and island parallel GAs, as well as both mutation-only and recombination-only algorithms. Each algorithm was implemented using the BDD based encoding. Although detailed results are not provided here for brevity, each BDD-IG encoded EA provided superior quality results over the various traditional, NN, and penalty/validation based KGA approaches described earlier. The best overall performance was provided by incorporating a mutation-only hill climbing post-processing phase (HC) with the recombination-only Hypergamous Parallel GA (HPGA) [6]. The resulting hybrid algorithm (HPGA+HC) discovered the best known solution for each of the graphs examined [7].

Additional empirical investigation of OBIG and BDD-IG was conducted using the DIMACS maximum clique problems [17]. Two of the DIMACS graph classes known to be among the most difficult [1] are reported here: keller and mann. Performance of the OBIG algorithm and a BDD-IG encoded HPGA+HC is shown in table 3. The

Table 3. Best (maximum) IS solutions for keller and mann benchmarks

graph	|V|	optimum	OCH2	OCH20	GMCA	OBIG	HPGA+HC
mann_a9	45	16	16			16	
mann_a27	378	126	126	126	125	126	
mann_a45	1035	345	343	345	337	343	345
mann_a81	3321	≥1100				1097	1099
keller4	171	11	11	11	11	11	
keller5	776	27	25	27	18	26	26
keller6	3361	≥59				52	54

table also includes results reported for Aggarwal's OCH GA for IS and Bui's GA for maximum clique (GMCA) [5]. OCH had previously displayed exceptionally strong performance compared to various traditional algorithms for independent set on the DIMACS benchmarks [1]. Each of Aggarwal's algorithms for IS, along with the algorithms for vertex cover reported here, were run on the complementary graphs of the original benchmarks. Solutions thus correspond to maximum clique solutions found for the original graphs. Values reported in the table represent the size of the best (maximum) independent set or clique discovered (or alternatively as $|V-V_C|$).

Two versions of OCH were reported. OCH2 corresponds to the best solution of two experiments; OCH20 reports the best solutions of twenty experiments. Aggarwal did not implement OCH for mann_a81 and keller6, "the reason being not so much the method itself, but the fact that there was not enough memory to load the graphs" [1]. OBIG was implemented for each of the graphs listed. A BDD-IG encoded algorithm (HPGA+HC) was run on each graph instance for which OBIG failed to find the optimum independent set (cover). Following the Aggarwal study, the value reported is the best of two experiments. Implementation details are described in [7]. The resulting solution qualities show OBIG and BDD-IG performance similar to that seen earlier for the irand and sg graphs. OBIG or the BDD-IG based hybrid algorithm obtained results equal to or better than the strong results provided by OCH on all graphs except keller5. (The algorithms were also successfully run on the very large graphs containing ~3000 vertices). As before, these results were obtained with modest computational requirements: OBIG ran at least an order of magnitude faster than OCH2 (normalized to SGI Challenge equivalent performance), while the BDD-IG encoded EA required time comparable to OCH20. Ten OBIG runs on mann_a81 and keller6 required 0.44 and 17.7 seconds respectively.

5 Conclusions

This paper introduced an encoding for the important graph theoretic problems of vertex cover and independent set. The approach relies on a binary decision diagram embedded within an effective traditional heuristic for the underlying vertex cover problem. Each bit of the chromosome corresponds not to a particular vertex (as is true in most prior EAs for VC and related problems), but instead to the next decision to be made during synthesis of a feasible cover. The resulting BDD-IG encoding and fitness evaluation function was shown to encode a subset of the search space's feasible region guaranteed to include optimal solutions. The encoding thus naturally obviates the need to apply validation or penalty methods to infeasible solutions derived from direct encodings. Because BDD-IG is based on the skeleton of a strong traditional heuristic for VC, the approach also allows initial EA populations to be seeded in known promising regions of the search space. The encoding was empirically examined on a variety of vertex cover problem classes, including graphs derived from ISCAS-89 circuit benchmarks and graphs included in the DIMACS benchmark suite for maximum clique. Performance of the BDD-IG based algorithms was exceptionally strong compared to heuristic algorithms for VC and IS that are among the best previously reported.

References

[1] Aggarwal, C. C., Orlin, J. B., and Tai, R. P. "Optimized Crossover for the Independent Set Problem," *Operations Research*, 45(2):226-234, March-April 1997.

[2] Bäck, T. and Khuri, S. "An Evolutionary Heuristic for the Maximum Independent Set Problem," in *Proc. First IEEE Conference on Evolutionary Computation*, IEEE Press, pp. 531-535, 1994.

[3] Bar-Yehuda, R. and Even, S. "A Local-Ratio Theorem for Approximating the Weighted Vertex Cover Problem," *Annals of Discrete Mathematics*, 25:27-45, 1985.

[4] Brglez, F., Bryan, D. and Kozminski, K. "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. Intl. Symp. on Circuits and Systems*, pp. 1929-1934, 1989.

[5] Bui, T. N. and Eppley, P. H. "A Hybrid Genetic Algorithm for the Maximum Clique Problem," in *Proc. of the 6th ICGA*, Morgan Kaufmann, pp. 478-484, 1995.

[6] Evans, I. K. "Embracing Premature Convergence: The Hypergamous Parallel Genetic Algorithm," to appear in *Proc. of the 5th IEEE Intl. Conf. on Evolutionary Computation*.

[7] Evans, I. K. "Reemphasizing Recombination in Evolutionary Search: Heuristics for Vertex Cover," Ph.D. Thesis, University of Iowa, 1997.

[8] Feo, T. A., Resende, M. and Smith, S. H. "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set," *Operations Research*, 42(5):860-878, September-October 1994.

[9] Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[10] Halldórsson, M. M. and Radhakrishnan, J. "Greed is Good: Approximating Independent Sets in Sparse and Bounded-degree Graphs," *Proc. 26th ACM Symposium on Theory of Computing*, pp. 439-448, 1994.

[11] Hochbaum, D. S., ed. *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, 1997.

[12] Kommu, V. "Enhanced Genetic Algorithms in Constrained Search Spaces with Emphasis in Parallel Environments," Ph.D. Thesis, The University of Iowa, 1993.

[13] Motwani, R. "Lecture Notes on Approximation Algorithms - Volume I," Technical Report, Department of Computer Science, Stanford University, 1992.

[14] Pramanick I. and Kuhl, J. G. "A Practical Method for Computing Vertex Covers for Large Graphs," in *Proc. Intl. Symposium on Circuits and Systems* pp. 1859-1862, 1992.

[15] Pramanick I. and Kuhl, J. G. "An Inherently Parallel Method for Heuristic Problem-Solving: Part II-Example Applications," *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1016-1028, 1995.

[16] Shrivastava, Y., Dasgupta, S. and Reddy, S. M. "Guaranteed Convergence in a Class of Hopfield Networks," *IEEE Transactions on Neural Networks*, 3(6):951-960, 1992.

[17] DIMACS Challenge Problems for Maximum Clique, available via anonymous FTP at dimacs.rutgers.edu.