An Evolutionary Heuristic for the Minimum Vertex Cover Problem

Sami Khuri
San José State University
Dept. of Mathematics & Computer Science
One Washington Square
San José, CA 95192–0103
U.S.A.

Thomas Bäck
University of Dortmund
Dept. of Computer Science, LS XI
D–44221 Dortmund
Germany

Introduction

The minimum vertex cover problem (mvcp) belongs to the class of NP-hard problems. The maximum clique problem can be reduced to it [3]. Thus, the search for an optimal solution is intractable (unless of course P = NP). Due to its numerous applications, especially in various matching problems, the problem is not abandoned. The goal is to find heuristics: approximation algorithms that have polynomial running times that return near-optimal solutions.

In this work we describe the results of applying a genetic algorithm [6, 4] to the mvcp. The latter is a highly constrained combinatorial optimization problem. Unlike traditional approaches that use domain-specific knowledge, and specialized genetic operators, the approach presented here makes use of a graded penalty term incorporated in the fitness function to penalize infeasible solutions. The fitness function itself is quite simple and needs to be added to GENEsYs, the genetic algorithm software package we use in this work. This package is based on Grefenstette's widely used GENESIS [5].

Following the formal introduction of the mvcp, the best known heuristic algorithm for that problem is introduced. The study then focuses on the genetic-based heuristic. Several problem instances are used with both algorithms and the results are compared. Our work concludes with some observations about our findings, and some suggestions on the use of evolutionary heuristics for other combinatorial optimization problems.

The Minimum Vertex Cover Problem

The mvcp of an undirected graph $G = (V, E)$ where V is the set of vertices and E denotes the set of edges, consists in finding the smallest subset $V' \subseteq V$ such that $\forall \langle i, j \rangle \in E$, we have $i \in V'$ or $j \in V'$ (or both). V' is said to be a vertex cover of G. The following is a formal definition of the mvcp in which we make use of Stinson's terminology for combinatorial optimization problems [14]:

Problem instance: A graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of vertices and $E \subseteq V \times V$ the set of edges. An edge between vertices i, j is denoted by the pair $\langle i, j \rangle \in E$. We define the *adjacency matrix* (e_{ij}) according to

$$e_{ij} = \begin{cases} 1 & , \text{ if } \langle i, j \rangle \in E \\ 0 & , \text{ otherwise} \end{cases} .$$

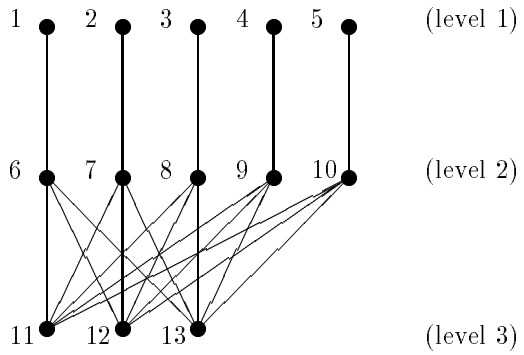

Figure 1: Construction of the regular graph after Papadimitriou and Steiglitz.

Feasible solution: A set V' of nodes such that $\forall \langle i, j \rangle \in E$: $i \in V' \vee j \in V'$.

Objective function: The size $|V'|$ of the vertex cover V'.

Optimal solution: A vertex cover V' that minimizes $|V'|$.

Since we are interested in covering all the edges of the given graph by using as few nodes as possible, one might be tempted to use a greedy-based heuristic to tackle the mvcp. The algorithm consists in repeatedly selecting a vertex of highest degree (the node that covers as many of the remaining edges as possible), and removing all of its incident edges. This is not a good strategy as was demonstrated by Papadimitriou and Steiglitz ([10], p. 407). They considered regular graphs, each of which consists of three levels. The first two levels have the same number of nodes while the third level has two nodes less than the number of nodes found on the previous two levels. More precisely, each graph consists of $n = 3k + 4$ ($k \geq 1$) nodes; $k + 2$ nodes on the first level, labeled $1, \ldots, k+2$; followed by $k+2$ nodes on the second level, labeled $k + 3, \ldots, 2k + 4$, while the k nodes of the third level are labeled $2k + 5, \ldots, 3k + 4$.

The regular graph for $k = 3$ can be found in [10] (p. 407) and is reproduced in figure 1.

A minimum vertex cover is obtained by choosing all the nodes of the second level, but the greedy strategy would start with the nodes of the third level, since these have highest degree. Consequently, the greedy strategy

mvcp100-01		mvcp100-02		mvcp100-03		mvcp100-04		mvcp100-05		PS100	
$f_{2\cdot10^4}(\vec{x})$	N	$f_{2\cdot10^4}(\vec{x})$	N	$f_{2\cdot10^4}(\vec{x})$	N	$f_{2\cdot10^4}(\vec{x})$	N	$f_{2\cdot10^4}(\vec{x})$	N	$f_{2\cdot10^4}(\vec{x})$	N
53	1	55	34	55	77	55	96	55	99	34	65
54	1	57	3	56	1	67	1	83	1	66	35
55	3	59	2	59	6	68	1				
56	6	60	3	63	3	75	1				
57	4	61	10	64	3	90	1				
58	4	62	1	66	1						
59	9	63	8	67	1						
60	4	64	1	68	1						
61	6	65	1	70	1						
62	12	66	6	71	1						
63	5	67	6	74	1						
> 63	45	> 67	25	> 74	4						
$\bar{f} = 62.61$		$\bar{f} = 62.75$		$\bar{f} = 57.62$		$\bar{f} = 55.80$		$\bar{f} = 55.28$		$\bar{f} = 45.20$	

Table 1: Experimental results obtained by the genetic algorithm for five random graphs of size $n = 100$ with edge density $d = 0.1$ ("mvcp100-01"), $d = 0.2$ ("mvcp100-02"), $d = 0.3$ ("mvcp100-03"), $d = 0.4$ ("mvcp100-04") and $d = 0.5$ ("mvcp100-05") and the regular graph of size $n = 100$ from Papadimitriou and Steiglitz ("PS100").

finds a solution of size $2k + 2$ (since it has to select $k + 2$ nodes in addition to the nodes of the third level), while the optimal solution has size $k + 2$.

It can be shown that the greedy algorithm never produces a solution which is more than $\ln(n)$ times the optimum, where n is the number of vertices ([9], p. 323). More precisely, the algorithm performs with a relative error $\frac{\ln(n)}{n}$ (which grows as fast as $\ln(n)$) ([10], p. 408).

The following simple algorithm is surprisingly the best approximation algorithm known for the mvcp ([9], p. 301). It can be shown that the size of the vertex cover it returns is guaranteed to be no more than twice the size of an optimal vertex cover ([1], p. 968).

ALGORITHM 1 (vercov)

$C := \{\}$;
  {C contains the vertex cover being constructed}
$E' := E$;
while $E' \neq \emptyset$ **do**
  randomly choose $\langle u, v\rangle \in E'$;
  $C := C \cup \{u, v\}$;
  $E' := E' - \{\langle x, y\rangle \mid x = u \vee y = v\}$;
    {remove from E' every edge incident on
     either u or v};
od
return C;

To compare the performance of the above algorithm with the genetic algorithm, a cover V' is represented by a binary vector $\vec{x} = x_1 x_2 \ldots x_n$ where $x_i = 1$ if the i^{th} node is in V', and $x_i = 0$ if it is not. Using this representation, we developed the following fitness function to be minimized by the genetic algorithm:

$$f(\vec{x}) \quad = \quad \sum_{i=1}^{n} \left(x_i + n \cdot (1 - x_i) \cdot \sum_{j=i}^{n} (1 - x_j) e_{ij} \right)$$

The term $\sum_{i=1}^{n} x_i$ of $f(\vec{x})$ determines the size of the potential vertex cover represented by $\vec{x}$, while the term $n \cdot \sum_{i=1}^{n} \sum_{j=i}^{n} (1 - x_i) \cdot (1 - x_j) \cdot e_{ij}$ penalizes sets V' that are not covers by adding a penalty of magnitude n for each edge e_{ij} for which $i \notin V'$ and $j \notin V'$. Consequently, the second term drops to zero for feasible solutions.

The fitness function was developed according to the following design principles that are important for a successful penalty function approach [11, 13, 7]:

- The penalty should be graded, i.e., fitness values should improve as solutions approach (in terms of the Hamming distance) feasible regions of the search space.
- Infeasible binary vectors are guaranteed to yield fitness values which are inferior to fitness values of even the worst feasible solutions.

Experimental Results

The experiments reported in this section are performed by using a genetic algorithm with a population size $\mu = 50$, a mutation rate $p_m = 1/n$, crossover rate $p_c = 0.6$, proportional selection, and two-point crossover. As reported in [2, 12], the latter is expected to perform better than the traditional one-point crossover. In order to apply the genetic algorithm to the minimum vertex cover problem, no component of this general genetic algorithm — except, of course, the fitness function — has to be modified. This fact reflects the wide applicability

mvcp100-01		mvcp100-02		mvcp100-03		mvcp100-04		mvcp100-05		PS100	
$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N
80	6	80		80		80	1	80		66	100
82	17	82	1	82		82		82	1		
84	23	84	2	84	1	84	1	84			
86	20	86	4	86	6	86		86	5		
88	17	88	18	88	12	88	5	88	7		
90	14	90	17	90	15	90	13	90	10		
92	1	92	31	92	18	92	24	92	21		
94	2	94	20	94	28	94	34	94	27		
96		96	4	96	16	96	19	96	17		
98		98	1	98	4	98	3	98	9		
100		100		100		100		100	3		
$\bar{f} = 86.46$		$\bar{f} = 89.22$		$\bar{f} = 92.22$		$\bar{f} = 92.96$		$\bar{f} = 93.12$			

Table 2: Experimental results obtained by the vercov heuristic for five random graphs of size $n = 100$ with edge density $d = 0.1$ ("mvcp100-01"), $d = 0.2$ ("mvcp100-02"), $d = 0.3$ ("mvcp100-03"), $d = 0.4$ ("mvcp100-04") and $d = 0.5$ ("mvcp100-05") and the regular graph of size $n = 100$ from Papadimitriou and Steiglitz ("PS100").

and robustness of genetic algorithms when compared to problem-specific heuristics.

For the experimental tests, random graphs of size $n = 100$ with different edge densities d, $d \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, are used. For example, an edge density of $d = 0.1$ means that an edge is placed between two nodes with a probability of 0.1. For each of these problems, a total of $N = 100$ runs of the genetic algorithm is performed. The results are summarized in table 1 for the best results that were encountered during the 100 runs for each test problem. For each problem instance, we record the different fitness values that were obtained and their frequencies. Furthermore, the average fitness value $\bar{f}$ over all 100 runs is indicated at the bottom of the table. The total number of function evaluations per single run is chosen to be $2 \cdot 10^4$, such that only an extremely small fraction of the search space is tested by the genetic algorithm.

In addition to the randomly constructed graphs, the regular graphs introduced by Papadimitriou and Steiglitz ([10], pp. 406–409) and described in the previous section of this paper, are used to compare the behavior of the genetic algorithm with the vercov heuristic.

Recall that these graphs contain $n = 3k + 4$ ($k \geq 1$) nodes distributed on three levels. They can be scaled up by choosing high values for k. We choose problem instances of the regular graph of sizes $n = 100$ ($k = 32$) and $n = 202$ ($k = 66$).

For each of the problems a total of $N = 100$ independent runs of the vercov heuristic is performed, and the results are summarized in table 2.

The same experiments were also performed for graphs of size $n = 200$ in order to test the behavior of the genetic algorithm as well as the vercov heuristic for an even larger problem size. In this case, the genetic algorithm was allowed to run for $4 \cdot 10^4$ function evaluations per experiment. The corresponding results obtained by the genetic algorithm are shown in table 3, while the results from the vercov heuristic are presented in table 4.

From these tables, a number of interesting conclusions can be drawn. For the random graphs, which were used to compare the genetic algorithm and the vercov heuristic, it is known by construction that an optimum of quality 55 (respectively 110) exists, which is likely to be the global optimum. This optimum is found by the genetic algorithm at least once within the $N = 100$ runs that were performed.

Moreover, the randomly constructed problems quickly become simpler for the genetic algorithm when the edge density is increased. For an edge density of $d = 0.5$, the genetic algorithm almost surely finds the global optimum of the problems.

In case of the regular graph after Papadimitriou and Steiglitz, the genetic algorithm is able to find the global optimum in about 2/3 of all runs, while the remaining runs identify a solution of quality $2k + 2$.

On the other hand, the vercov heuristic performs poorly. For the randomly constructed problems, the heuristic finds best solutions of quality 80 (for graphs of size 100) and 172 (for graphs of size 200), respectively. Surprisingly, the average performance of the vercov heuristic decreases as the edge density is increased, i.e., the problems become even harder for the vercov heuristic while they become much simpler for the genetic algorithm when the edge density increases.

For the regular graph instances, the vercov heuristic fails in all of the 100 runs that were performed for each of the graphs. Recall that the graphs are constructed so as to force the vercov heuristic into yielding solutions of quality $2k + 2$. The random choice of edges in the while-loop of the algorithm implies that nodes from either layers one and two, or from layers two and three

mvcp200-01		mvcp200-02		mvcp200-03		mvcp200-04		mvcp200-05		PS202	
$f_{4\cdot10^4}(\vec{x})$	N	$f_{4\cdot10^4}(\vec{x})$	N	$f_{4\cdot10^4}(\vec{x})$	N	$f_{4\cdot10^4}(\vec{x})$	N	$f_{4\cdot10^4}(\vec{x})$	N	$f_{4\cdot10^4}(\vec{x})$	N
110	2	110	55	110	95	110	99	110	100	68	60
113	1	120	10	128	1	140	1			134	40
116	4	121	7	129	6						
117	2	122	2	130	2						
119	3	123	7	134	1						
120	1	125	1								
121	2	126	1								
122	3	127	1								
124	3	129	1								
125	6	132	2								
126	2	134	1								
> 126	71	> 134	12								
$\bar{f} = 132.50$		$\bar{f} = 119.07$		$\bar{f} = 111.01$		$\bar{f} = 110.30$		$\bar{f} = 110.00$		$\bar{f} = 108.00$	

Table 3: Experimental results obtained by the genetic algorithm for five random graphs of size $n = 200$ with edge density $d = 0.1$ ("mvcp200-01"), $d = 0.2$ ("mvcp200-02"), $d = 0.3$ ("mvcp200-03"), $d = 0.4$ ("mvcp200-04") and $d = 0.5$ ("mvcp200-05") and the regular graph of size $n = 202$ from Papadimitriou and Steiglitz ("PS202").

of the graph are involved in the solution constructed, which in turn implies that two layers have to be part of the solution found by the heuristic [10].

Conclusion

In this work, we have demonstrated that genetic algorithms can be used in a straightforward way to find good approximate solutions of the minimum vertex cover problem. Moreover, the results found by the genetic algorithm are better than those obtained from the best known traditional heuristic, the vercov algorithm. These findings, in addition to other good results obtained on different, highly constrained combinatorial optimization problems such as the subset sum [7], minimum tardy task [7], and multiple knapsack problems [8], give strong evidence that genetic algorithms can yield good solutions for a wide range of hard combinatorial optimization problems for which solutions can be represented by binary strings.

Acknowledgements

The first author was partially supported by the California State University research award GS850-851. The second author gratefully acknowledges support by the German BMFT, grant 01IB403A, within the project EVOALG.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[2] L. J. Eshelman, R. A. Caruna, and J. D. Schaffer. Biases in the crossover landscape. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 10–19. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP–Completeness*. Freemann & Co., San Francisco, CA, 1979.

[4] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, Reading, MA, 1989.

[5] J. J. Grefenstette. GENESIS: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165, 1984.

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

[7] S. Khuri, Th. Bäck, and J. Heitkötter. An evolutionary approach to combinatorial optimization problems. In D. Cizmar, editor, *Proceedings of the 22nd Annual ACM Computer Science Conference*, pages 66–73, Phoenix, 1994. ACM Press, New York.

[8] S. Khuri, Th. Bäck, and J. Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In E. Deaton, D. Oppenheim, J. Urban, and H. Berghel, editors, *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 188–193. ACM Press, New York, 1994.

[9] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, 1994.

[10] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice Hall, 1982.

mvcp200-01		mvcp200-02		mvcp200-03		mvcp200-04		mvcp200-05		PS202	
$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N	$f(\vec{x})$	N
172	2	172		172		172		172		134	100
174	3	174		174	1	174		174			
176	13	176	1	176	1	176	1	176	1		
178	14	178	1	178	3	178	4	178	1		
180	23	180	5	180	5	180	1	180	5		
182	14	182	11	182	5	182	9	182	5		
184	15	184	18	184	17	184	8	184	5		
186	7	186	16	186	19	186	16	186	13		
188	5	188	24	188	16	188	16	188	11		
190	3	190	15	190	16	190	16	190	21		
192	1	192	4	192	9	192	15	192	19		
> 192	0	> 192	5	> 192	8	> 192	14	> 192	19		
$\bar{f} = 180.98$		$\bar{f} = 186.46$		$\bar{f} = 186.92$		$\bar{f} = 188.06$		$\bar{f} = 189.16$			

Table 4: Experimental results obtained by the vercov heuristic for five random graphs of size $n = 200$ with edge density $d = 0.1$ ("mvcp200-01"), $d = 0.2$ ("mvcp200-02"), $d = 0.3$ ("mvcp200-03"), $d = 0.4$ ("mvcp200-04") and $d = 0.5$ ("mvcp200-05") and the regular graph of size $n = 202$ from Papadimitriou and Steiglitz ("PS202").

[11] J. T. Richardson, M. R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 191–197. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[12] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[13] A. E. Smith and D. M. Tate. Genetic optimization using a penalty function. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 499–505. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[14] D. R. Stinson. *An Introduction to the Design and Analysis of Algorithms.* The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 2nd edition, 1987.