An Evolutionary Heuristic for the Minimum Vertex Cover Problem

Sami Khuri[1] and Thomas Bäck[2]

[1] San José State University, Department of Mathematics & Computer Science, One Washington Square, San José, CA 95192–0103, U.S.A.
[2] Department of Computer Science, University of Dortmund, 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 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 [4].

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 [9]:

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

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

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 can be seen by considering Papadimitriou and Steiglitz's regular graphs ([6], p. 407). The graph consists of three levels; the first two have the same number of nodes while the third level has two nodes less than 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 third level are labeled $2k+5, \ldots, 3k+4$.

The regular graph for $k = 3$ can be found in ([6], p. 407) and is reproduced in figure 1.

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 ([5], p. 323). More precisely, the algorithm performs with a relative error $\frac{\ln(n)}{n}$ (which grows as fast as $\ln(n)$) ([6], p. 408).

The following simple algorithm is surprisingly the best approximation algorithm known for the mvcp ([5], 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' ≠ ∅ do
    randomly choose ⟨u, v⟩ ∈ E';
    C := C ∪ {u, v};
    E' := E' − {⟨x, y⟩ | x = u ∨ y = v};
        {remove from E' every edge incident
            on either u or v};
od
return C;
```

In order to compare this algorithm with a genetic algorithm, we developed the following fitness function to be minimized by the genetic algorithm:

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

The first term of $f(\mathbf{x})$ determines the size of the potential vertex cover $\mathbf{x}$, while the second term penalizes sets V' that are not covers by adding a penalty of n for each edge that is not represented by any node of V'.

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 (which is, according to the experimental results reported in [2, 8], expected to perform better than the traditional one-point crossover). In order to become applicable 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 and robustness of genetic algorithms in contrast to problem-specific heuristics.

For the experimental test random graphs of size $n = 100$ and with different edge density $d \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ of the graph are used. For each of these problems, a total of $N = 100$ runs of the genetic algorithm are performed, and the runs are evaluated according to the number of runs that yield solutions of identical quality. The results are summarized in table 1 for the best results that were encountered during the 100 runs for each test problem. For each experiment, the average final best 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 was 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 ([6], 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.

mvcp100-01		mvcp100-02		mvcp100-03		mvcp100-04		mvcp100-05		PS100	
$f_{2\cdot 10^4}(\mathbf{x})$	N	$f_{2\cdot 10^4}(\mathbf{x})$	N	$f_{2\cdot 10^4}(\mathbf{x})$	N	$f_{2\cdot 10^4}(\mathbf{x})$	N	$f_{2\cdot 10^4}(\mathbf{x})$	N	$f_{2\cdot 10^4}(\mathbf{x})$	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 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 from Papadimitriou and Steiglitz ("PS100").

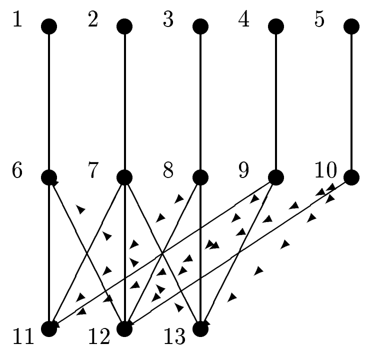

Fig. 1.: Construction of the regular graph after Papadimitriou and Steiglitz.

Recall that these graphs contain $n = 3k+4$ ($k \geq 1$) distributed on three levels. They can be scaled up by choosing high values for k.

In general, the graph contains $k+2$ nodes (labeled $1, \ldots, k+2$) in the first row, $k+2$ nodes (labeled $k+5, \ldots, 2k+4$) in the second row, and k nodes (labeled $2k+5, \ldots, 3k+4$) in the third row. Of course, a minimum vertex cover is obtained by choosing all nodes of the second row, but a greedy strategy would start with the nodes of the third row, since these have highest degree. Consequently, a greedy strategy finds a solution of size $2k+2$ (since it has to select another row of nodes in addition to the third row), while the optimal solution has size $k+2$. The genetic algorithm and the vercov heuristic are also compared on problem instances of size $n = 100$ ($k = 32$) and $n = 202$ ($k = 66$) of this regular graph.

For each of the problems a total of $N = 100$ independent runs of the vercov heuristic was 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

mvcp100-01		mvcp100-02		mvcp100-03		mvcp100-04		mvcp100-05		PS100	
$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	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 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 from Papadimitriou and Steiglitz ("PS100").

algorithm as well as the vercov heuristic for an even larger problem dimension. 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.

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.

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 Schaffer [7], pages 10–19.
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. 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.
5. C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, Reading, MA, 1994.
6. Ch. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Prentice Hall, 1982.
7. J. D. Schaffer, editor. *Proceedings of the 3rd International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers, San Mateo, CA, 1989.
8. 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 Schaffer [7], pages 51–60.
9. D. R. Stinson. *An Introduction to the Design and Analysis of Algorithms.* The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 2nd edition, 1987.

mvcp200-01		mvcp200-02		mvcp200-03		mvcp200-04		mvcp200-05		PS202	
$f_{4\cdot10^4}(\mathbf{x})$	N	$f_{4\cdot10^4}(\mathbf{x})$	N	$f_{4\cdot10^4}(\mathbf{x})$	N	$f_{4\cdot10^4}(\mathbf{x})$	N	$f_{4\cdot10^4}(\mathbf{x})$	N	$f_{4\cdot10^4}(\mathbf{x})$	N
110	2	110	55	110	95	55	96	55	99	68	60
113	1	120	10	128	1	67	1	83	1	134	40
116	4	121	7	129	6	68	1				
117	2	122	2	130	2	75	1				
119	3	123	7	134	1	90	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								
$\bar{f} = 132.50$		$\bar{f} = 119.07$		$\bar{f} = 111.01$		$\bar{f} = 55.80$		$\bar{f} = 55.28$			

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 from Papadimitriou and Steiglitz ("PS202").

mvcp200-01		mvcp200-02		mvcp200-03		mvcp200-04		mvcp200-05		PS202	
$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	N	$f(\mathbf{x})$	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 from Papadimitriou and Steiglitz ("PS202").