Optim Lett
DOI 10.1007/s11590-015-0851-1

ORIGINAL PAPER

Exact solutions to generalized vertex covering problems: a comparison of two models

Gary Kochenberger · Mark Lewis · Fred Glover · Haibo Wang

Received: 11 May 2013 / Accepted: 7 January 2015
© Springer-Verlag Berlin Heidelberg 2015

Abstract The generalized vertex cover problem (GVCP) was recently introduced in the literature and modeled as a binary linear program. GVCP extends classic vertex cover problems to include both node and edge weights in the objective function. Due to reported difficulties in finding good solutions for even modest sized problems via the use of exact methods (CPLEX), heuristic solutions obtained from a customized genetic algorithm (GA) were reported by Milanovic (Comput Inf 29:1251–1265, 2010). In this paper we consider an alternative model representation for GVCP that translates the constrained linear (binary) form to an unconstrained quadratic binary program and compare the linear and quadratic models via computations carried out by CPLEX's branch-and-cut algorithms. For problems comparable to those previously studied in the literature, our results indicate that the quadratic model efficiently yields optimal solutions for many large GVCP problems. Moreover, our quadratic model dramatically

G. Kochenberger (✉)
Business Analytics Department, School of Business Administration,
University of Colorado at Denver, Denver, CO 80217, USA
e-mail: gary.kochenberger@cudenver.edu

M. Lewis
Craig School of Business, Missouri Western State University,
St Joseph, MO 64507, USA
e-mail: mlewis14@missouriwestern.edu

F. Glover
OptTek Inc, Boulder, CO 80302, USA
e-mail: glover@opttek.com

H. Wang
IB&TS Division, Sanchez School of Business,
Texas A&M International University, Laredo, TX 78041, USA
e-mail: hwang@tamiu.edu

outperforms the corresponding linear model in terms of time to reach and verify optimality and in terms of the optimality gap for problems where optimality is unattained.

Keywords Vertex cover · Performance evaluation · Combinatorial problems

1 Introduction

The classic form of the vertex cover problem has the goal of finding a subset of nodes of minimum cardinality such that every edge in the graph has at least one endpoint in the subset of nodes chosen. A variant of this model that allows for node weights yields the so-called minimum weight vertex cover problem. Contemporary papers addressing the challenge of solving these problems and highlighting their importance are given in references [2–6,9,11–13]. Recently Hassin and Levin [7] and then Milanovic [12] introduced a further generalization of the vertex cover problem arising in practical applications that incorporates edge weights as well as node weights, where the cost associated with a given edge depends on whether one, both or neither end point is in the subset of nodes chosen. For example, upgrades to telecommunications network coverage areas involving combinations of node as well as transmission media edge costs [14]. This most recent version of the vertex cover problem is referred to as the Generalized Vertex Cover Problem (GVCP).

The paper by Milanovic [12] reports computational experience on some modest sized problems (up to 500 nodes with 100,000 edges) indicating that the exact method of CPLEX was unable to solve many of the test problems under consideration but that a specially crafted GA efficiently produced solutions of higher quality than those produced by CPLEX (using a reasonable limit on the computation time consumed). In this paper we re-visit the challenge of finding exact or near-optimal solutions to GVCP problems similar to those considered by Milanovic [12] as well as much larger GVCP problems. To this end we propose an alternative formulation for GVCP consisting of an unconstrained quadratic binary model and compare the linear model of the literature with our quadratic model on a set of test problems with characteristics similar to those found in the literature. Testing and comparison of the two models is carried out using the algorithms available via CPLEX for mixed integer and quadratic optimization.

In Sect. 2 below we present the linear binary model for GVCP as given by Milanovic [12] along with our quadratic alternative. Section 3 presents our computational experience and comparisons, followed by summary and conclusions in Sect. 4.

2 Alternative models for GVCP

Consider an undirected graph G = (V, E) with a vertex set V = {1, 2, 3 .. n}. For GVCP we seek a minimum cost subset of nodes and edges where the cost structure consists of: (1) c_i = cost of node i;(2) $d_k(e)$ = cost contributed by edge e if exactly k endpoints are in the subset of nodes chosen where $d_0(e) \geq d_1(e) \geq d_2(e)$.

In other words, the cost $d_0(e)$ for not including an edge e is higher than including a single endpoint of the edge $d_1(e)$. Including both endpoint nodes of an edge, $d_2(e)$

has the lowest of the three costs. Thus GVCP allows the possibility that a given edge, at an appropriate cost, can have neither endpoint in the subset of nodes chosen.

2.1 Linear model for GVCP

Following the notation of Milanovic, the linear model for GVCP_Lin is:

$$\min \sum\nolimits_{i=1}^{n} c_i x_i + \sum\nolimits_{(i,j)\in E}^{n} \left\{ d_2(i,\ j) z_{ij} + d_1(i,\ j)\,(y_{ij} - z_{ij}) + d_0(i,\ j)\,(1 - y_{ij}) \right\}$$

subject to $(\forall (i,\ j) \in E)$

$$y_{ij} \le x_i + x_j \tag{1}$$

$$z_{ij} \le x_i \tag{2}$$

$$z_{ij} \le x_j \tag{3}$$

$$x_i,\ y_{ij}, z_{ij} \text{ binary}$$

where $x_i = 1$ if node i is in the subset; 0 otherwise; $z_{ij} = 1$ if *both* vertices i and j are in the subset; 0 otherwise; $y_{ij} = 1$ if at *least* one of the vertices i and j are in the subset; 0 otherwise.

Note that GVCP_Lin is a large 0/1 linear program with two variables (y_{ij} and z_{ij}) for each edge in the graph. Thus, GVCP_Lin grows rapidly with both node count and with edge density. Note also that by setting $y_{ij} = 1$, $z_{ij} = 0$, and eliminating the edge costs, GVCP_Lin reduces to the weighted vertex cover problem WVCP. In addition, if we set all node costs to 1, GVCP_Lin reduces to the classic vertex cover problem.

2.2 Quadratic alternative formulation

Our alternative quadratic model modifies the linear model by making appropriate quadratic substitutions for y_{ij} and z_{ij} in a manner that allows these edge variables along with the constraints of (1), (2), and (3) to be eliminated, yielding a compact equivalent model in the form of an unconstrained quadratic binary program. To illustrate this we first re-write GVCP_Lin in a slightly more convenient form:

$$\min x_0 = \sum\nolimits_{i=1}^{n} c_i x_i - \sum\nolimits_{(i,j)\in E} a_{ij} z_{ij} - \sum\nolimits_{(i,j)\in E} b_{ij} y_{ij} + C \tag{4}$$

subject to $(\forall (i,\ j) \in E)$

$$y_{ij} \le x_i + x_j \tag{5}$$

$$z_{ij} \le x_i \tag{6}$$

$$z_{ij} \le x_j \tag{7}$$

$$x_i,\ y_{ij},\ z_{ij} \text{ binary}$$

where $a_{ij} = d_1(i,j) - d_2(i,j) \geq 0; b_{ij} = d_0(i,j) - d_1(i,j) \geq 0;\ C = \sum_{(i,j)\in E} d_0(i,j)$. From (4) we see that both y_{ij} and z_{ij} will tend to be 1 unless the constraints force them to be zero. The constraints of (5) imply that y_{ij} will equal 0 when *both* x_i and x_j are zero which, in turn, means that we can replace y_{ij} with the quadratic expression:

$$y_{ij} = x_i + x_j - x_i x_j \tag{8}$$

Furthermore, the constraints of (6) and (7) imply that z_{ij} will be 0 when *either or both* x_i and x_j are equal to 0 which means we can replace z_{ij} with the quadratic expression:

$$z_{ij} = x_i x_j \tag{9}$$

Making the substitutions of (8) and (9) into the objective function of (4) yields

$$\begin{aligned} \text{GVCP\_xQx}: \ \min x_0 = \sum_{i=1}^{n} c_i x_i - \sum_{(i,j)\in E} a_{ij} x_i x_j \\ - \sum_{(i,j)\in E} b_{ij}(x_i + x_j - x_i x_j) + C \end{aligned} \tag{10}$$

This provides an unconstrained quadratic binary program of the form: min $x_0 = x'Qx + C$ where Q is an n-by-n symmetric matrix of constants whose diagonal elements are the node costs and whose remaining elements are associated edge costs. A similar approach for reformulations with penalty functions is reported in a recent study by Hsia and Wang [8].

3 Computational experience

For the purpose of seeking exact solutions, our interest in this section is to see if either GVCP_Lin or GVCP_xQx has a computational advantage over the other and whether or not one of these formulations may actually enable an exact method (in this case CPLEX) to solve instances of GVCP of the size and characteristics found in the literature in a reasonable amount of time. In addition to comparing the alternative models with one another, we also provide a comparison of our results to those reported earlier by Milanovic [12].

3.1 Test problems

Our comparisons are based on three sets of test problems denoted P1, P2, P3. All problems were created using the procedure adopted from [11] which creates difficult GVCP problems via node costs that are averaged over one hundred randomly generated covers. All problems are available electronically from the first author. Set P1 was designed to mimic the size and characteristics of the problems used by Milanovic in [12]. To illustrate the effect of node costs on problem difficulty, a second set of problems, P2, was generated from P1 using the same topology and same edge costs but with node costs simply being doubled from their P1 values. In addition, a third set of larger problems, P3, was generated with up to 5,000 nodes and more than 10 million edges in order to test whether the types of comparisons we found based on P1 and P2 carried over to larger problem instances.

All problems were solved with CPLEX v12 on a 4-core 3.4 GHz Intel i7-2600 cpu using 8 threads and Windows 7 with 16 GB RAM. For the purpose of comparing with previous results from the literature, the same time limit of 7,200 s used by Milanovic [12] was imposed in all cases. It is customarily recommended that for stability reasons difficult problems should not be run in hyper-threaded mode. However, with hyper-threading enabled we obtained a performance increase of about 50 %, as measured by the number of iterations and nodes explored, and had no computer stability problems. In all cases, CPLEX default settings were utilized. While CPLEX has many customizable parameter settings, the default settings allow it to modify parameters in accordance with the CPLEX presolve and heuristic routines and provide for a legitimate comparison of the quadratic and linear models where performance differences cannot be attributed to the user's choice of parameter settings. That said, the CPLEX working memory parameter was changed from the default 128 MB to 12 GB, without which the larger problems could not have been attempted.

3.2 Results for problem sets P1 and P2

Tables 1 and 2 summarize the average results obtained for problem sets P1 and P2, respectively. The results in our tables do not include the additive constant, C. For all 28 P1 problem instances, both the GVCP_xQx and the GVCP_Lin quickly found optimal solutions. The linear model was able to quickly prove optimality for all 28 problems in P1 while the quadratic model proved optimality in the allotted time on 26 of the 28 problems. In the 26 cases where both models yielded proven optimal solutions, the quadratic model did so in an average of 5 s while the linear model took an average of 11 s.

The problems of set P1 were designed to be as comparable as possible to the problems reported in the paper by Milanovic [12] which reported using CPLEX v10 on a 2.5 GHz single threaded machine with 1 GB RAM. The work reported in [12] succeeded in exactly solving only 10 small and/or sparse problems out of 28 and reported the average time to solution of 4,488 s, while our corresponding average time for the linear model was 10.5 s (about 420 times faster) to solve all problems to proven optimality. This difference in the performance of CPLEX can be largely attributed to the difference in computers and in our use of parallel hyper-threading, as well as to the difference in alternative versions of CPLEX.

Table 1 Average results from problem set P1 categorized by number of edges

# edges	Time to opt solution		Time to prove opt		[12] time to solution
	xQx	LIN	xQx	LIN	GA
Small	0.01	0.02	0.09	0.03	0.5
Medium	7	1.5	14	2	5
Large	0.02	39.5	0.02	40	47

Table 2 Average results from problem set P2 categorized by number of edges

# edges	Time to opt solution		Time to prove opt	
	xQx	LIN	xQx	LIN
Small	0.02	0.03	0.07	0.03
Medium	40	3	538	3
Large	0.2	1252	13	1321

Milanovic also reported results for a genetic algorithm (GA) heuristic applied to the 28 problems [12]. With the GA, each problem was run 20 times yielding results that matched the optimal solution for 5 small problems, found best known solutions, perhaps not optimal, on 15 problems, and failed to match best known solutions on 8 problems. Across all 28 problems, the average time to reach the best solution found was 13 s. In comparison, our average times to find the known optimal solutions were 0.1 and 7.6 s for GVCP_xQx and GVCP_Lin respectively. Our average time to prove that the solutions found were optimal was 5 s for GVCP_xQx and 11 s for GVCP_Lin.

Table 1 illustrates the effects of the number of edges on problem difficulty for P1, where small problems are up to 500 edges, medium up to 5,000 edges and large problems having up to over 100,000 edges. The 28 problems in P1 consist of seven sets of problems having from 30 to 500 nodes, each with 4 variations of edge density ranging from 1 to 84 %. A similar table was presented in [12] and representative results from that paper for the genetic algorithm tested are included in the last column. The table shows that for the dense problems the quadratic form clearly outperforms the linear form. One of the 28 problems, a 300 node 1,081 edge graph proved difficult for GVCP_xQx, taking 121 seconds to find the optimal solution and affecting the overall average reported. However, all other problems in that category GVCP_xQx found the optimal solution in under 0.2 s and in general the time to find and the time to prove exact solutions using GVCP_xQx compare very favorably with the heuristic GA results reported in [12].

Table 2 illustrates that doubling the node costs made the P1 problems more difficult for both the quadratic and the linear binary representations. Doubling the node costs changes the relative magnitude distribution between the diagonal and off-diagonal elements of Q, a relationship that is known to affect problem difficulty as we see in the table. Table 2 also shows that GVCP_xQx did not perform as well on problems with a medium number of edges, with the majority of these problems being edge sparse 300, 400 and 500 node problems, where edge sparse indicates that only about 10 % of all possible edges are enumerated in the problem. In all problem sets, GVCP_xQx outperforms on edge dense problems (>30 %). The average time taken to prove optimality was $6\times$ longer for the problems in P2 than for those in P1. Average time to find the optimal solution for all 28 problems in the more difficult P2 set was 16 s for GVCP_xQx and 279 s for GVCP_Lin.

3.3 Large problem set (P3)

Larger problem instances (P3) were investigated to determine if GVCP_xQx, would continue to dominate GVCP_Lin in solution time and in solution quality, as measured

Table 3 Performance on larger problem instances (P3)

Nodes	Edges	Time to Sol		MIP gap		Best solution
		Quad	Linear	Quad (%)	Linear (%)	
750	2,0725	4	113	0.19	0.00	−239,815
	506,95	1	1,020	0.15	0.04	−668,695
	103,160	1	7,100	0.09	0.94	−1,363,015
	235,146	0.5	7,000	0.06	21.20	−3,074,643
1,000	50,139	1	916	0.28	0.05	−639,205
	150,086	0.7	5,000	0.12	18.00	−2,000,200
	299,952	1	7,000	0.07	21.00	−3,946,731
	417,802	0.5	4,000	0.05	24.00	−5,4645,02
2,000	200,255	3	7,000	0.14	11.86	−2,641,880
	400,224	5	6,000	0.12	24.00	−5,267,672
	800,100	4	[a]	0.06	[a]	−10,486,869
	1,672,728	6	[a]	0.03	[a]	−21,618,900
3,000	3,688,461	14	[a]	0.02	[a]	−47,476,288
5,000	10,454,451	60	[a]	0.02	[a]	−134,149,816
	Averages	7.3	4217	0.1	12.1	

[a] Unable to find a feasible solution in 7,200 s time limit

by the optimality gap. Results for problems ranging from 750 nodes to 5,000 nodes are shown in Table 3. The table shows the problem size and solution time for the optimality gap indicated, subject to a 7,200 s time limit. GVCP_xQx, always found a solution as good as, or better than, the one found by GVCP_Lin. For the problems containing from 750 to 2,000 nodes, GVCP_xQx consistently yielded high quality solutions within an average optimality gap of 0.1 % in 2 s. Conversely, GVCP_Lin yielded lower quality solutions within an average optimality gap of 12 % in about 4,400 s (more than 2,000 times slower). This performance difference is explained in large part due to the vast difference in size between the problem representations generated by GVCP_Lin and GVCP_xQx.

Table 3 also provides a comparison of the two models on the largest and densest instances reported in the literature, 3,000 node 82 % dense and 5,000 node 84 % dense problems. The mere size of these problems creates a computational challenge for GVCP_Lin. For example, the 5,000 node problem has over 18 million columns and 26 million rows after CPLEX presolve eliminated over 5 million rows and close to 3 million columns. The corresponding reduced MIQP of GVCP_xQx for this problem had only 2,509 binary variables. As shown in Table 3, GVCP_xQx, enabled high quality solutions to be quickly found for both of these large problems while GVCP_Lin failed to produce a feasible solution in the allotted time limit of 7,200 s.

Experiments with CPLEX parameters designed to emphasize optimality and improve best bound were tested on GVCP_xQx with little effect on either the optimality gap or the number of nodes explored.

4 Summary and conclusions

In this paper we show how the standard linear model for the Generalized Vertex Cover Problem, GVCP_Lin, can be reformulated as an equivalent, more compact quadratic model (GVCP_xQx). Computational comparisons carried out using the branch-and-cut optimizers available from CPLEX reveal the superiority of the new, quadratic model.

Comparisons, based on time to optimality and solution quality as measured by optimality gap, were made on a set of test problems that included instances similar in size and characteristics to those previously studied in the literature [12] for the linear model GVCP_Lin. We tested the two models on 70 problems ranging in size from 30 to 5,000 nodes (largest GVCP reported in literature) with varying edge densities. Across all problems, the quadratic model, GVCP_xQx, outperformed the linear model, GVCP_Lin, by a wide margin, producing exact or quantifiably high quality solutions in just a few seconds. The superiority of GVCP_xQx was particularly evident on the denser problem instances. We note that the exact GVCP_xQx results we report here, even allowing for differences in computers used, compare nicely with the heuristically produced (GA) results reported in [12] for the comparable problems.

These results highlight that the quadratic formulation allows much larger problem instances than previously considered in the literature to be solved exactly by off-the-shelf methods like CPLEX. All told, the quadratic model, for the class of problems considered here, exhibits a substantial computational advantage over the competing linear representation. This advantage derives from the fact that the quadratic model is substantially smaller than the linear model and represents a practical way to circumvent the combinatorial explosion associated with the much larger linear model.

Moreover, the results we report here for the Generalized Vertex Cover Problem are consistent with results we have found for other classes of problems where the standard linear model was compared with an equivalent quadratic alternative. For instance, see the paper "Computationally Attractive Non-Linear Models for Combinatorial Optimization," by Alidaee, et al. [1]. The results reported here add to the mounting evidence that quadratic alternatives to standard linear models, for many problem classes, can not only be computationally attractive but may enable high quality solutions to be found by commercial products (like CPLEX) for problem instances that are too large to solve in the more common linear form.

As part of our on-going research, we plan to extend the type of comparisons made here on GVCP to other forms of vertex cover problems. Results from these extensions will be reported in future papers.

References

1. Alidaee, B., Kochenberger, G., Lewis, K., Lewis, M., Wang, H.: Computationally attractive non-linear models for combinatorial optimization. Int. J. Math. OR. **1**, 9–19 (2009)
2. Bouamama, S., Blum, C., Boukerram, A.: Population-based iterated Greedy algorithm for the minimum weight vertex cover problem. A Appl. Soft Comput. **121**, 632–1639 (2012)
3. Cai, S., K. Su, Chen, Q.: EWLS: a new local search for minimum vertex cover. In: Proceedings of the 24^{th} AAAI Conference on Artificial Intelligence. pp. 45–50 (2010)

4. Cai, S., Su, K., Sattar, A.: Two new local search strategies for minimum vertex cover. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence. pp. 441–447 (2012)
5. Cygan, M., Pilipczuk, M.: Split vertex deletion meets vertex cover: new fixed-parameter and exact exponential-time algorithms. Inf. Process. Lett. **113**, 179–182 (2013)
6. Gomes, F.C., Meneses, C.N., Pardalos, P.M., Viana, G.V.R.: Experimental analysis of approximation algorithms for the vertex cover and set covering problems. Comput. Oper. Res. **33**, 3520–3534 (2006)
7. Hassin, R., Levin, A.: The minimum generalized vertex cover problem. ACM Trans. Algorithms **2**, 66–78 (2006)
8. Hsia, Y., Wang, Y.: A new penalty parameter for linearly constrained 0–1 quadratic programming problems. Optim. Lett. **7**, 765–778 (2013)
9. Javoanovic, R., Tuba, M.: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Appl. Soft Comput. **11**, 5360–5366 (2011)
10. Lewis, M.: On the use of guided design search for discovering significant decision variables in the fixed-charge capacitated multicommodity network design problem. Networks **53**(1), 6–18 (2008)
11. Likas, A., Stafylopatis, A.: A parallel algorithm for the minimum weighted vertex cover problem. Inf. Process. Lett. **53**(4), 229–234 (1995)
12. Milanovic, M.: Solving the generalized vertex cover problem by genetic algorithm. Comput. Inf. **29**, 1251–1265 (2010)
13. Shyu, S., Yin, P.-Y., Lin, B.M.T.: An ant colony optimization algorithm for the minimum weight vertex cover problem. Ann. OR. **131**, 283–304 (2004)
14. Voss, S., Fink, A.: A hybridized tabu search approach for the minimum weight vertex cover problem. JOH **18**, 869–876 (2012)