Combinatorial and Global Optimization, pp. 111–121
P.M. Pardalos, A. Migdalas and R. Burkard, Editors
© 2002 World Scientific Publishing Co.

Solving Quadratic Knapsack Problems by Reformulation and Tabu Search. Single Constraint Case

Fred Glover
University of Colorado at Boulder

Gary Kochenberger
University of Colorado at Denve

Bahram Alidaee
University of Mississippi

Mohammad Amini
University of Memphis

Abstract

Several recent papers have presented new approaches to special cases of the quadratic knapsack (QK) problem. Despite the advances reported, these problems remain, in general, very challenging to solve. Models containing more than 100 variables tax current methods to their limits. Moreover, current methods can only handle special instances of quadratic objective functions, leaving a large range of important problems unconsidered. In this paper, we show that it is possible to handle quadratic knapsack problems of dramatically greater size than those within the capacity of previous methods, and in addition to handle entirely general quadratic objectives. This outcome results by the device of reformulating the QK problem as an unconstrained binary quadratic problem, and applying a recently developed metaheuristic to this latter version. We report computational experience that discloses the viability of our approach.

Keywords: quadratic knapsack problem, unconstrained binary quadratic problem, tabu search

1 Introduction

The Quadratic (multidimensional) Knapsack problem is one of several classic NP-hard problems that has been the object of considerable research over the years. The general problem can be stated as

$$\begin{array}{ll} \mathrm{QK}: & \max\ xQx \\ \mathrm{st} & \\ & Ax \leq b \\ & x \in \{0,1\} \end{array}$$

We assume the coefficients of the matrix A are all non-negative, and those of b are all positive. No sign conditions are assumed for Q. Note that any linear term originally appearing in the objective function can be absorbed into the diagonal of Q, since $x_j = x_j^2$ for any 0-1 variable x_j. In this paper we focus on the single knapsack case, where A consists of a single row vector, although the approach taken can be applied to the more general case with multiple knapsack constraints.

Program QK has many applications, having been shown to address problems arising in capital budgeting, flexible manufacturing, earth station site selection for satellite communication systems, task assignments in host satellite systems, and a variety of other problems. The references given, and particularly the paper by Gallo, Hammer and Simeone [4], give discussions of various applications. A significant number of these applications involve the single knapsack case we examine here, and most of the algorithms developed for QK also focus on this case. It is also possible, via the types of re-formulations recently highlighted by Kochenberger, Alidaee, and Amini [12], to recast many traditional models into the form of QK. Thus, the applicability of QK is quite broad in its scope.

Several notable papers have recently appeared in the literature advancing our ability to solve special cases of the single knapsack QK. Billionnet and Calmels [2] discuss a branch and bound algorithm based on linearizations while Michelon and Veilleax [14] present a branch and bound approach based on Lagrangian decomposition and Lagrangian relaxation. Each of these algorithms assumes the matrix Q is positive. Both approaches employ variable fixing routines and both report computational experience with random problems of size up to 40 variables.

More recently, Hammer and Rader [8] gave a branch and bound algorithm for the single knapsack QK based on repeated use of the L_2 -best linear approximation. Variable fixing procedures, based on Lagrangian relaxation, are explored and computational experience on problems with up to 100 variables is reported. The Hammer and Rader algorithm, like the two methods cited above, assumes a positive Q matrix.

Since QK is NP-Hard, exact algorithms are likely to degrade in performance as the problem size grows. As expected, the experience reported in the three algorithms referenced above illustrates the rapid growth in computation times as problem size grows.

The state of the art, even for this special class of problems (positive Q) with a single knapsack constraint, is rather limited in terms of the size of problem that can be solved by exact methods. Larger problems, and those involving a general Q matrix, are computationally demanding and more amenable to heuristics than exact methods.

The approach we take in this paper is to reformulate QK as an unconstrained binary quadratic program (QP) and then solve the reformulated, equivalent model by recently developed heuristics designed to solve QP. This approach is motivated by the success reported on new solution approaches for QP. See for example, the recent papers by Pardalos and Rodgers [15], Chardaire and Sutter [3], Glover, Kochenberger and Alidaee [5], Glover, Kochenberger, Alidaee, and Amini [6], Hammer, Boros, and Sun [7], and Lodi, Allemand, and Liebling [13]. In this case we make use of the method of [6]. Although heuristic, this method has been demonstrated to be so effective that it has obtained optimal solutions to all benchmark problems that exact methods have been able to solve, while requiring greatly reduced computational time by comparison to the exact methods.

In the sections that follow, we present the transformation used to convert QK into QP. We then present our computational experience followed by some summary comments.

2 Reformulation

Transformations for reformulating integer programs in the manner employed here have been proposed by several authors. Discussions relevant to our work, and upon which we draw, are given by Bazarra and Goode [1], Hammer and Rudeanu [9], Hansen [10], Hansen et al [11], and Sinclair [16]. For a recent paper highlighting such reformulations in a variety of model settings, the reader is referred to Kochenberger, Alidaee, and Amini [12]. In the work reported here, we convert Quadratic Knapsack models into an unconstrained quadratic program (QP) by employing a quadratic infeasibility penalty. While other representations are possible, we restrict our efforts to quadratic penalties in order to take advantage of our ability to solve QP.

Adding slack variables (in binary expansion form), we can write the quadratic knapsack problem as:

$$\begin{aligned} \mathrm{QK}: \quad & \max\ x_0 = xQx \\ \mathrm{st} \quad & \\ & Ax = b \\ & x \in \{0,1\} \end{aligned}$$

where x, A, and Q have been augmented to include the slack variables. Then, the equality constraints can be accommodated by introducing a positive penalty, PEN, and the associated penalty term

$$\mathrm{PEN}(Ax - b)'(Ax - b).$$

Subtracting this term from the objective function yields the equivalent penalized program

$$\begin{aligned} \mathrm{QK(PEN)}: \quad \max x_0 &= xQx - \mathrm{PEN}(Ax - b)'(Ax - b) \\ &= xQx - xDx + cx + \text{constant} \\ \text{st} \quad & \\ & x \in \{0,1\} \end{aligned}$$

where $D = \mathrm{PEN} * A^tA$, $c = 2\mathrm{PEN} * b^tA$, and the constant $= -\mathrm{PEN} * b^tb$. The additive constant does not effect the optimization and can be ignored for the time being. The remaining three terms can be combined into a single term allowing us to represent the equivalent unconstrained quadratic program simply as:

$$\begin{aligned} \mathrm{QK(PEN)}: \quad & \max x\hat{Q}x \\ \text{st} \quad & \\ & x \in \{0,1\} \end{aligned}$$

where $\hat{q_{ij}} = q_{ij} - d_{ij}$ for $i \neq j$ and $\hat{q_{ii}} = q_{ii} - d_{ii} + c_i$ for the diagonal terms. As previously remarked, we solve QK(PEN) using the Tabu Search (TS) approach reported in Glover, Kochenberger, Alidaee, and Amini [6]. An overview of our TS method is given in the appendix.

Example 1. To illustrate the reformulation procedure, consider the example used by Hammer and Rader [8]:

$$\begin{aligned} \text{maximize } x_0 &= 2x_1 + 5x_2 + 2x_3 + 4x_4 + 8x_1x_2 + 6x_1x_3 \\ &\quad +10x_1x_4 + 2x_2x_3 + 6x_2x_4 + 4x_3x_4 \\ \text{st} \quad & \\ & 8x_1 + 6x_2 + 5x_3 + 3x_4 \leq 16 \end{aligned}$$

Introducing a slack activity, bounded above by 3, as $1x_5 + 2x_6$, the transformation can be used to give an equivalent unconstrained problem in 6 binary variables. (The upper bound on the slack is selected to be large enough to admit an optimal solution. This can be done by several approaches, which we do not address here.) Choosing PEN to be 10, we get QP(PEN) with $\hat{Q}$ given by

$$\hat{Q} = \begin{bmatrix} 1922 & -476 & -397 & -235 & -80 & -160 \\ -476 & 1565 & -299 & -177 & -60 & -120 \\ -397 & -299 & 1352 & -148 & -50 & -100 \\ -235 & -177 & -148 & 874 & -30 & -60 \\ -80 & -60 & -50 & -30 & 310 & -20 \\ -160 & -120 & -100 & -60 & -20 & 600 \end{bmatrix}$$

and an additive constant of -2560. Solving this unconstrained problem, QP(PEN), gives the solution $x_0 = 2588$ which is found at $x = \{1,0,1,1,0,0\}$. Adjusting for the additive constant, we have the original objective function value of 28. In this example, both slack variables are equal to zero and thus the knapsack constraint is tight at the optimal solution given. We now turn to the computational work we carried out and the results obtained.

3 Computational Experiments

To test our approach to quadratic knapsack problems, 44 random quadratic knapsack problems were generated, re-formulated as unconstrained quadratic binary programs, and solved using our tabu search algorithm. Each problem was generated with q_{ij} between 0 and 25, a_j between 1 and 10, and b chosen between 10 and $\sum a_j$. Problem sizes ranged from 10 to 500 variables with densities varying from 25% to 100 %. Each problem had a single knapsack constraint that was converted to an equality with the addition of six additional binary slack variables. This allowed for a slack activity up to size 63 (which proved to more than sufficient).

Tables 1 and 2 summarize our results. For each problem, we report problem characteristics, along with the solution obtained in the allotted number of Tabu Search oscillation cycles and computation times. The solutions shown in Table 1, with the exception of problem C03, are in fact optimal, as verified by the branch and bound algorithm of Pardalos and Rodgers (P&R) [15]. Problem C03 could not be solved to completion by the exact method of [15] and thus an optimal solution to this problem is not known. Likewise, optimal solutions for the problems of Table 2 are yet to be established. These problems, while readily solved by our Tabu Search heuristic, proved to be beyond the capability of the branch and bound algorithm. (Branch and bound (P&R) run times in excess of six hours failed to produce optimal solutions for the smallest of these problems.) The results shown in Table 2 are the best values found within the arbitrary cycle limit imposed in our tabu search approach.

Certainly the choice of the parameter PEN is key to the approach we are taking. Theoretically, it is sufficient to choose PEN to be greater than the sum of the absolute values of the elements of Q. Our experience indicates, however, that feasible solutions can be obtained with much smaller values for PEN. PEN needs to be large enough to make infeasible choices unattractive to the search/optimization process. Yet choosing PEN somewhat larger than necessary may cause the elements of $\hat{Q}$ to become unnecessarily large, masking the relevance of Q and making the problem, in principle, harder to solve. This potential difficulty, as discussed later in this section, did not prove to be a problem for us. The approach taken in the work reported here was to start with a small value for PEN and raise it as needed in order to produce

Table 1: COMPUTATIONAL RESULTS

Problem ID	# variables	Density	PEN	Solution	# TS Cycles	Time(sec)
A01	10	0.25	50	6220	100	<1
A01	10	0.25	50	24373	100	<1
A03	10	0.5	50	1357	100	<1
A04	10	0.5	50	45342	100	<1
A05	10	0.75	50	3408	100	<1
A06	10	0.75	50	34220	100	<1
A07	10	1	50	58612	100	<1
A08	10	1	50	42462	100	<1
B01	20	0.25	50	180947	100	<1
B02	20	0.25	50	76794	100	<1
B03	20	0.5	50	73279	100	<1
B04	20	0.5	50	226432	100	<1
B05	20	0.75	50	347331	100	<1
B06	20	0.75	50	55570	100	<1
B07	20	1	50	194715	100	<1
B08	20	1	50	32605	100	<1
C01	30	0.25	250	156990	100	<2
C02	30	0.25	50	2635	100	<2
C03	30	0.5	250	1262937	100	<2
C04	30	0.5	250	814340	100	<2
C05	30	0.75	50	4631	100	<2
C06	30	0.75	250	273512	100	<2
C07	30	1	250	422510	100	<2
C08	30	1	50	6472	100	<2

feasible solutions. We arbitrarily took PEN to be 50, 250, 500, 750, 1000, or 1500. For the "A", "B", "C", "D", and "E" problems, PEN was initially set to 50. Each problem was then solved and the solutions obtained were checked for feasibility.

Problems that were infeasible were solved again with successively larger values of PEN until the knapsack constraint was satisfied. The "F" problems, based on what we learned from A-E, were solved with PEN = 1500 from the outset. Clearly PEN = 1500 (or something larger) would have worked for all the problems. Tables 1 and 2 report the values of PEN used to yield the solutions shown. In all cases, the solutions listed are feasible with respect to the original knapsack constraints.

Choosing PEN too large, as mentioned above, has the *potential* to make a problem more difficult to solve than an appropriately chosen smaller value of PEN. For the testing we have done to date, this potential difficulty did not materialize. This is illustrated in Table 3 where we report results obtained by solving a particular problem,

Table 2: COMPUTATIONAL EXPERIENCE (CON'T)

Problem ID	# variables	Density	PEN	Solution	# TS cycles	Time(sec)
D01	40	0.25	50	888033	100	<4
D02	40	0.25	50	1607975	100	<4
D03	40	0.5	50	177190	100	<4
D04	40	0.5	50	1272126	100	<4
D05	40	0.75	50	1076775	100	<4
D06	40	0.75	50	1389799	100	<4
D07	40	1	50	1549279	100	<4
D08	40	1	50	40931	100	<4
E01	100	0.25	50	2659975	100	<9
E02	100	0.25	50	1953310	100	<9
E03	100	0.5	500	4241440	100	<9
E04	100	0.5	750	10993576	100	<9
E05	100	0.75	500	2119812	100	<9
E06	100	0.75	1000	11040166	100	<9
E07	100	1	750	13696762	100	<9
E08	100	1	1000	5943174	100	<9
F01	500	0.25	1500	35491046	500	240
F02	500	0.5	1500	35544448	500	240
F03	500	0.75	1500	76078805	500	240
F04	500	1	1500	76145674	500	240

C08, with various values of PEN ranging from 250 to 3000. Table 3 lists the value of PEN, the optimal solution value, and the Tabu Search cycle at which the optimal solution was obtained for these runs.

In all cases, the solutions shown in Table 3 are feasible and optimal. Note that the performance of the heuristic was fairly uniform over the range of PEN values. While there is some variation in the cycle count required to reach the optimal solution, no discernible pattern is suggested by the table. Similar computations on other problems have also yielded only small variations in solution burden associated with differing PEN values. This is a somewhat surprising discovery since the folklore about using "infeasibility penalties" in discrete and nonlinear optimization is that their size can materially effect computational efficiency. The absence of this effect here suggest that our TS method is highly robust.

Our approach had no trouble finding high quality, feasible solutions to all problems in the test set. And, compared to alternative methods from the literature, these solutions were found very quickly. The papers by Michelon and Veilleux [14] and Billionnet and Calmels [2] report computational experience on problems up to size n = 40 only. Their problems are comparable in size and density to our "A", "B", "C", and "D"

Table 3: PERFORMANCE ON PROBLEM C08 FOR VARIOUS VALUES OF PEN

PEN	Optimal Value	Cycles Yielding Optimal
250	30500	16
500	60750	17
750	91000	29
1000	121250	10
1500	181750	19
3000	363250	29

problems. For the 40 variable problems, Michelon and Veilleax report solution times of more than 900 seconds (SUN 4.0) while Billionet and Calmels report solutions times of approximately 500 seconds (HP 9000) for their 40 variable problems.

Hammer and Rader (H&R) [8] report computational experience with problems up to size n = 100. They report solving completely dense, 40 variable problems in roughly 20 seconds (SPARC station 1+). For their 100 variable problems, H&R solve low density problems with admirable efficiency. For instance, 25% problems are solved in little more than 2 minutes. However, 100 % dense problems, of size 100, were solved in roughly 24 minutes. None of these authors report experience on larger problems.

Since our approach is heuristic in nature, comparing the solution times we report in Tables 1 and 2 with the branch and bound times given above must be done with considerable care. Such comparisons are further confounded by the fact that different computers were used in the various studies and that our Tabu Search approach was run for an arbitrary number of oscillation cycles. Nonetheless, a quick comparison gives the reader some indication of the speed with which our approach generates high quality solutions to these problems. As shown in Tables 1 and 2, we produce optimal solutions to test problems up to 30 variables (and varying densities) in about 2 seconds on a Pentium 200 computer. Thus, as in the case of unconstrained problems, we obtain optimal solutions to all problems where such solutions are known, and we obtain such solutions in at least an order of magnitude faster than the exact approach. In many cases we are faster by two orders of magnitude.

The remaining problems belong to the group that could not be solved by the exact methods. We produced high quality feasible solutions to the 40 variable and 100 variable problems in roughly 4 and 9 seconds, respectively. For our 500 variable problems, which we ran for 500 oscillation cycles, run times were approximately 4 minutes. To the best of our knowledge, ours is the first study to report on problems as large as 500 variables. The previous limit is represented by the 100 variable problems in the Hammer and Rader paper.

It is interesting to note that our solution times for a given problem size are not very sensitive to problem density. Problems with 500 variables are solved in about 4 minutes whether they are 25% dense or 100% dense. This behavior is in sharp contrast to the branch and bound algorithms whose performance degrades considerably with density. For example, on their 100 variable problems, H&R report approximate times of 2, 3, 8, and 24 minutes, respectively, for densities 25, 50, 75, and 100%. The time invariant nature of our approach is due to the fact that regardless of the density of the original matrix Q, the transformation produces a matrix $\hat{Q}$ which is very dense. This is illustrated in the example presented earlier where Q was fairly sparse and $\hat{Q}$ was 100 % dense. It should also be pointed out that the transformation destroys any nice properties (like all non-negative elements) that Q may have exhibited. Neither the absence of special structure nor presence of high density pose a problem for our approach.

4 Summary and Conclusions

For quadratic knapsack problems of modest size and special properties, efficient exact methods, like that of Hammer and Rader, exist. However, quadratic knapsack problems with hundreds of variables pose computational difficulties in excess of what can reasonably be handled by exact methods. Current exact methods also are unable to handle QK problems that have negative elements in their objective functions. For these larger and/or more general problems, heuristic methods must be employed.

In this paper we have demonstrated that the unconstrained binary quadratic program, QP, is a viable alternative approach for modeling and solving QK problems. The reformulated problems are readily solved by our tabu search algorithm to yield high quality feasible solutions, which are optimal in all cases that can be solved by exact methods. Our results illustrate that large instances can be readily solved.

For the class of problems considered here, choosing an appropriate value for the parameter PEN was straightforward. An appropriate choice, which for any problem depends on both problem size and problem data, can always be found by a minimum of experimentation, or, more formally, by simple target analysis. However, we found (surprisingly) that taking PEN somewhat larger than necessary did not significantly affect computational performance.

It should be noted that no attempt was made to tailor our Tabu Search algorithm for the work carried out here. We used the algorithm of [6] for general QP problems with no modifications. (The method is also not tuned to obtain best results for these QP problems.) For instance, employing an improvement routine, designed specifically for QK problems, at critical points in the search process would likely lead to better results. Our approach likewise does not depend on any particular property of the original Q matrix. Future research will address the solution of QK problems that include multiple knapsack constraints.

APPENDIX: Overview of our Tabu Search Algorithm

Our algorithm is built around strategic oscillations that alternate between constructive phases (progressively setting variables to 1, ADDS) and destructive phases (progressively setting variables to 0, DROPS). To control the underlying search process, we use a memory structure that is updated at locally optimal solutions called *critical events.* A parameter *span* is used to indicate the amplitude of oscillation about a critical event. We begin with *span* equal to 1 and gradually increase it to some limiting value. For each value of *span*, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, we begin to gradually decrease span, allowing again for a series of alternating constructive and destructive phases. When *span* reaches the value of 1, a *complete span cycle* has been completed and the next cycle is launched.

Information stored at critical events is used to influence the search process by penalizing adds (during constructive phases) and favoring drops (during destructive phases) for variables that have been in recent critical solutions at a level of one. Cumulative critical event information is also used to introduce a subtle long term bias into the search process. A complete description of the approach is given in the references sited.

References

[1] Bazarra, M. and J. Goode, "A Cutting-Plane Algorithm for the Quadratic Set Covering Problem," OR, Vol.23, no. 1, Jan-Feb (1975), pp.150-158.

[2] Billionnet, A., and Calmels, A.,"Linear Programming for the 0-1 Quadratic Knapsack Problem," EJOR, 92 (1996),310-325.

[3] Chardaire, P., and Sutter, A., "A Decomposition Method for Quadratic Zero-One Programming," Management Science, 41(1995),704-712.

[4] Gallo, G., Hammer, P., and Simeone,B., " Quadratic Knapsack Problems," Mathematical Programming, 12(1980),132-149.

[5] Glover, F., Kochenberger, G., and Alidaee, B., "Adaptive Memory Tabu Search for Binary Quadratic Programs," Management Science, 44(1998), 336-345.

[6] Glover, F., Kochenberger,G., Alidaee, B., and, Amini, M., "Tabu Search With Critical Event Memory: An Enhanced Application for Binary Quadratic Programs," (1997), 2nd International Conference on MetaHeuristics.

[7] Hammer, P., E. Boros, and X. Sun, "On Quadratic Unconstrained Binary Optimization," paper given at the Seattle INFORMS Meeting, Fall 1998.

[8] Hammer, P., and Rader, D., "Efficient Methods for Solving quadratic 0-1 Knapsack Problems," INFOR, 35(1997),170-182

[9] Hammer, P., and Rudeanu, S., Boolean Methods in OR and Related Areas, Springer-Verlag, New York, (1968).

[10] Hansen, P., "Methods of Nonlinear 0-1 Programming," Annals of Discrete Mathematics, 5(1079), 53-70.

[11] Hansen, P., Jaumard,F., and Mathon,V., "Constrained Nonlinear 0-1 Programming," ORSA Journal on Computing, 5(1993), 97-119.

[12] Kochenberger, G., Alidaee, B., and Amini, M., "Applications of the Unconstrained Binary Quadratic Program," Working Paper, University of Colorado at Denver, (1998)

[13] Lodi, A., Allemand, K., and Liebling, T., "An Evolutionary Heuristic for Quadratic 0-1 Programming," EJOR (to appear).

[14] Michelon, P., and Veilleau, L., "Lagrangean Methods for the 0-1 Quadratic Knapsack Problem," EJOR, 92 (1996), 326-341.

[15] Pardalos, P., and Rodgers, G., "Computational Aspects of a Branch and Bound Algorithm for Quadratic zero-One Programming," Computing, 45(1990), 131-144.

[16] Sinclair, M., "An Exact Penalty Function Approach for Nonlinear Integer Programming Problems," EJOR, 27(1986), 50-56.