Computers & Operations Research 39 (2012) 3–11

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A computational study on the quadratic knapsack problem with multiple constraints

Haibo Wang [a,*], Gary Kochenberger [b], Fred Glover [c]

[a] Sanchez School of Business, Texas A&M International University, Laredo, TX 78041, USA
[b] School of Business Administration, University of Colorado at Denver, Denver, CO 80217, USA
[c] OptTek Systems, Boulder, CO 80302, USA

ARTICLE INFO

Available online 11 January 2011

Keywords:
0-1 quadratic programming
Knapsack problem
Mixed integer quadratic program
Branch-and-cut
Linearization

ABSTRACT

The quadratic knapsack problem (QKP) has been the subject of considerable research in recent years. Despite notable advances in special purpose solution methodologies for QKP, this problem class remains very difficult to solve. With the exception of special cases, the state-of-the-art is limited to addressing problems of a few hundred variables and a single knapsack constraint.

In this paper we provide a comparison of quadratic and linear representations of QKP based on test problems with multiple knapsack constraints and up to eight hundred variables. For the linear representations, three standard linearizations are investigated. Both the quadratic and linear models are solved by standard branch-and-cut optimizers available via CPLEX. Our results show that the linear models perform well on small problem instances but for larger problems the quadratic model outperforms the linear models tested both in terms of solution quality and solution time by a wide margin. Moreover, our results demonstrate that QKP instances larger than those previously addressed in the literature as well as instances with multiple constraints can be successfully and efficiently solved by branch and cut methodologies.

1. Introduction

Quadratic knapsack problem arises from knapsack problem with all binary variables, positive coefficients in constraint(s) and nonnegative coefficients in objective function to maximize a nonlinear objective function subject to knapsack constraint(s). Much of the literature on the quadratic knapsack problem concerns the construction and testing of special purpose methods for solving QKP problems with single knapsack constraint. While a few articles have proposed heuristic methods to be applied directly to QKP (see for instance Refs. [1,2]), most approaches described in the literature employ linearizations of one kind or another designed to convert QKP into an equivalent mixed integer linear program. In turn, this enables QKP to be solved by well-known approaches for optimizing MIPs. Illustrative of such approaches are the special purpose branch-and-bound methods proposed in Refs. [3–6]. In addition, other well-known methods proposed on general 0-1 programming, which can be applied to 0-1 QKP, include semi-definite programming [7,8], strong convex quadratics programming relaxation [9] and reformulation-linearization technique [10]. An overview of these and other methods is given in the recent survey paper by Pisinger [11] on quadratic knapsack problems but not with general 0-1 QKP. While advances in solution methodology have been reported in the literature, this class of problems remains very difficult to solve and the best known solution methods for QKP are limited in application to problems with a few hundred variables and a single knapsack constraint. Pre-processing and reduction techniques, such as those proposed by Pisinger et al. [12], enable larger instances to be solved in certain cases.

Rather than considering special purpose methods, we restrict our attention here to the general purpose optimizers for mixed integer linear programming (MILP) and mixed integer quadratic programming (MIQP) that come standard as part of CPLEX. No specialization is undertaken for the class of problems considered here. We use these optimizers to compare the performance of three substantially different formulations for QKP on a new test bed of challenging problems.

This paper extends the literature in several important ways.

1. This is the first paper in the literature to compare the performance of two well-known linearizations and one recently published linearization with the original quadratic formulation for finding optimal solutions to QKP problems.
2. This is the first paper in the literature to address general QKP instances with multiple knapsack constraints.

* Corresponding author. Tel.: +1 956 326 2503; fax: +1 956 326 2494.
E-mail addresses: hwang@tamiu.edu (H. Wang), gary.kochenberger@ucdenver.edu (G. Kochenberger), glover@opttek.com (F. Glover).

0305-0548/$ - see front matter Published by Elsevier Ltd.
doi:10.1016/j.cor.2010.12.017

3. We present extensive computational experience reporting our success in solving problems considerably larger and more complex than previously addressed in the literature.
4. We highlight the effectiveness of a quadratic programming based branch and cut approach for solving large instances of QKP.
5. Finally, we introduce a new set of challenging test problems to the research community.

The rest of this paper is organized as follows. In the next section we present the formulations considered in this research. Then in Section 3 we present our computational experience followed by our summary and conclusions in Section 4.

2. Formulations tested

In this section we present the four models tested and compared in this paper. The first is QKP in its original form. The second and third models are based on popular, common linearizations of QKP. The fourth model is based on recently published paper [13]. These four models are presented here and tested in Section 3. We note that while several linearizations have appeared in the literature, the three employed here are representative of the "linear" approach to QKP. Moreover, they can be implemented straight-away without the pre-processing or other special attention required of certain other linearizations.

2.1. Original quadratic model

The standard statement of the quadratic knapsack problem is

$$\text{QP}: \quad \max f(x) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_i x_j \tag{1}$$

st

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1,m \tag{2}$$

$$x \in \{0,1\}^n \tag{3}$$

For the computational work carried out and reported in this paper, we assume c_j, c_{ij}, a_{ij}, and $b_i \ge 0$. This formulation, which we will refer to as QP, will be compared with the following three linearizations.

In the computations carried out, we solved QP using CPLEX's MIQP solver. In general, this routine is designed to solve linearly constrained quadratic binary problems with objective functions of the form

$$\min x_0 = cx + x'Qx \tag{4}$$

where Q is a positive semi-definite (PSD) matrix. This PSD requirement can always be satisfied for the class of problems considered here by modifying c and Q using standard diagonal perturbation techniques (see, for instance, Refs. [14,15]). For the testing reported in this paper, we transformed each problem using the minimum eigenvalue transformation to ensure that the required convexity conditions were satisfied for each problem before applying MIQP.

2.2. First linearization

The first linearization we consider is a classic in the literature (see Refs. [3,6,16]) where each quadratic term in the objective function, $x_i x_j$, is replaced by a new binary variable, w_{ij}, and the new constraints

$$w_{ij} \le x_i, \quad w_{ij} \le x_j, \quad \text{and} \quad x_i + x_j \le 1 + w_{ij} \tag{5}$$

are added to the model to require that $w_{ij}=1$ if and only if $x_i=1$ and $x_j=1$. For the QKP problems considered here, the last of the three constraints in Eq. (5) is not necessary and we have the linearization

$$\text{LIN1}: \quad \max f(x) = \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} w_{ij} \tag{6}$$

st

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i i = 1,m \tag{7}$$

$$w_{ij} \le x_i \quad i = 1,\ldots,n-1, \quad j = i+1,\ldots,n \tag{8}$$

$$w_{ij} \le x_j \quad i = 1,\ldots,n-1, \quad j = i+1,\ldots,n \tag{9}$$

$$x_j, w_{ij} \ \textit{binary} \tag{10}$$

In the testing that follows, we refer to this model as LIN1. Note that this formulation grows rapidly in size with both n and problem density. Nonetheless Ref. [6] report that this simple formulation compares quite well with several other linearizations and that it is particularly well suited for low-density problems where it outperformed the alternative linearizations they tested.

2.3. Second linearization

For our second linearization we take a substantially different approach than that of Section 2.2 above. Here, we adopt an alternative linearization approach based on Ref. [17] (see also Refs. [11,18,19]). This development starts by noting that $f(x)$, from Eq. (1), can be written as

$$f(x) = \sum_{j=1}^{n} x_j g_j(x) \tag{11}$$

where

$$g_j(x) = c_j + \sum_{i=j+1}^{n} c_{ji} x_i \quad j = 1 \ldots (n-1) \tag{12}$$

and for $j=n$ we have

$$g_n(x) = c_n$$

Let

$$z_j = x_j g_j(x) \tag{13}$$

and define

$$U_j = \textit{upper bound on } g_j(x)$$

$$L_j = \textit{lower bound on } g_j(x)$$

Given this, a linearization of QKP (see Ref. [17]) is

$$\text{Max} \sum_{j=1}^{n} z_j \tag{14}$$

st

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad i = 1,m \tag{15}$$

$$L_j x_j \le z_j \le U_j x_j \tag{16}$$

$$g_j(x) - U_j(1-x_j) \le z_j \le g_j(x) - L_j(1-x_j) \tag{17}$$

$$x \ \textit{binary} \tag{18}$$

An observation of Glover [20] permits this linearization to be accomplished with fewer constraints, but within the present context we observe that it can be conveniently simplified to produce the following streamlined formulation:

$$\text{LIN2}:\quad \text{Max}\sum_{j=1}^{n} z_j \tag{19}$$

st

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i \quad i=1,m \tag{20}$$

$$z_j \le U_j x_j \tag{21}$$

$$z_j \le g_j(x) \tag{22}$$

$$x \ \textit{binary},\ z_j \ge 0. \tag{23}$$

Proposition. LIN2 is equivalent to the formulation (14)–(18).

Proof. It suffices to establish that Eqs. (16), (17) and (18) can be replaced by Eqs. (21), (22) and (23). The replacement of Eq. (16) by Eq. (21) is immediate by observing that we may legitimately take $L_j=0$, and by incorporating the resulting lower bound on z_j in Eq. (23) as a standard non-negativity restriction, This also justifies replacing $g_j(x)-L_j(1-x_j)$ in Eq. (17) with $g_j(x)$ to give Eq. (22). It remains to show that nothing is lost by dropping the term $g_j(x)-U_j(1-x_j)$ that bounds z_j from below in Eq. (17) to yield Eq. (22), even though this term may well be more restrictive than the non-negativity bound on z_j. Specifically, due to the form of the objective function (14) (=(19)) and the fact that z_j only appears in the constraints (16) and (17) in the original formulation, we are assured that the optimum value of z_j will be given by $z_j=\text{Min}(U_jx_j,g_j(x))$. If $z_j=g_j(x)$, then clearly the lower bound on z_j in Eq. (17) is redundant. On the other hand, if $z_j=U_jx_j$, we must show that $U_jx_j \ge g_j(x)-U_j(1-x_j)$. The outcome is established by observing that in both of the two cases $x_j=0$ and $x_j=1$, this latter inequality reduces to $U_j \ge g_j(x)$. □

Note that the U_j values can be taken to be the sum of all the coefficients in $g_j(x)$ in Eq. (12). Relative to LIN1, LIN2 is a very compact linearization involving just n new (continuous) variables and $2n$ new constraints. Our preliminary computational experience with this model, confirmed by the additional work reported below, suggests that LIN2 is quite effective compared to LIN1 for class of QKP instances considered in this paper.

2.4. Third linearization

After the completion of the initial study with LIN1 and LIN2, we learned about a new linearization proposed by Hansen and Meyer [13]. To evaluate the performance of this recently published linearization, we include this linearization in our study and refer this model as LIN3. The quadratic objective function can be rewritten as follows:

$$f(x)=g^{(0)}(x)+\sum_{j=1}^{n}(x_j g_j^{(1)}(x)+(1-x_j)g_j^{(2)}(x)) \tag{24}$$

where $g^{(0)}(x),g_i^{(1)}(x),g_i^{(2)}(x)$ are linear functions which can be derived from a posiform of the quadratic objective function. In the case of QKP, $g^{(0)}(x)$ is equal to a constant. The complete details of this new linearization approach are given in Ref. [13]. Let $\phi_j(x)=g_j^{(1)}(x)-g_j^{(2)}(x)$ and define$U_j=\textit{upper bound on } \phi_j(x)$ and $L_j=\textit{lower bound on } \phi_j(x)$

$$\text{LIN3}:\quad \text{Max}\sum_{j=1}^{n} z_j \tag{25}$$

st

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i \quad i=1,m \tag{26}$$

$$z_j \ge g_j^{(1)}(x)-(1-x_i)U_j, \quad j=1,n \tag{27}$$

$$z_j \ge g_j^{(2)}(x)+x_jL_j, \quad j=1,n \tag{28}$$

$$x \ \textit{binary},\ z_j \ge 0.$$

3. Computational experience

To test and compare the four models, we generated a new set of test problems ranging in size from 10 to 800 variables and from 3 to 15 knapsack constraints. These test problems were generated following procedure proposed by Gallo et al. [21]. Specifically, the a_{ij} coefficients are randomly generated with uniformly distributed in [1,50], the c_j are randomly generated in the range [1.100], and the c_{ij} values ($c_{ij}=c_{ji}$) are zero with probability equal to the problem density and otherwise randomly generated with uniformly distributed in the range [1,100]. Finally, the right-hand side values, b_i for each of the knapsack constraints are randomly generated with uniformly distributed in the range $[50,\sum_{j=1}^{n} a_{ij}]$. This approach to generating QKP test problems is widely accepted in the literature.

The test bed considered here consists of three sets of problems: 24 *small* instances ranging in size from 10 to 100 variables and 3 to 6 knapsack constraints; 24 *medium-sized* instances ranging in size from 200 to 500 variables and 2 to 10 knapsack constraints; and, 8 *large* instances of size 800 variables with 5, 10, and 15 knapsack constraints. For all three size categories, densities range from 25% to 75%. This set of test problems contains larger instances than other test beds appearing in the literature and is the first to include problems with multiple knapsack constraints. As shown in the solution tables to follow, these problems, particularly the larger instances, proved to be very challenging for CPLEX and all four formulations tested. An additional discussion of problem difficulty is given in the appendix of this paper.

Tables 1–3 give the problem characteristics. Column 1 denotes the problem ID. Column 2 denotes the value of n and column 3 denotes the value of m, which the number of constraints in the original quadratic programming model. Column 4 denotes the problem density, which represents the percentage of nonzero coefficients of the quadratic terms. Columns 5, 7 and 9 denote the number of variables associated with each model and Columns 6, 8 and 10 denote the number of constraints associated with each model. All problems are available at www.tamiu.edu/~hwang/research/qkp/.

Each of the four model formulations was applied to this problem test bed. For the three linear formulations, LIN1, LIN2 and LIN3, the problems were solved using CPLEX's branch-and-cut optimizer (MILP, version 10.2) designed for linear mixed integer problems. Parameter values for CPLEX were set as follows: *rinsheur* in a range of 1–40, and *heuristicfreq* in a range of 5–40. For the quadratic formulation (QP), the problems were solved using CPLEX's branch-and-cut optimizer (MIQP, version 10.2) designed for quadratic mixed integer problems. The same set of parameters was used for both the linear and the quadratic optimizer. The key difference is that the model QP is solved via quadratic relaxations while the three linear models are solved utilizing LP relaxations. All runs are carried out by a 1.3 GHz Pentium IV PC.

Table 1
Description of small quadratic knapsack problems.

ID	*n*	*m*	Density	LIN1		LIN2		LIN3		QP	
				# vars	#cons	#vars	#cons	#vars	#cons	#vars	#cons
sqkp1	10	3	25%	22	27	20	23	20	23	10	3
sqkp2			50%	37	57	20	23	20	23	10	3
sqkp3			75%	44	71	20	23	20	23	10	3
sqkp4		6	25%	24	34	20	26	20	26	10	6
sqkp5			50%	34	54	20	26	20	26	10	6
sqkp6			75%	46	78	20	26	20	26	10	6
sqkp7	20	3	25%	87	137	40	43	40	43	20	3
sqkp8			50%	118	199	40	43	40	43	20	3
sqkp9			75%	165	293	40	43	40	43	20	3
sqkp10		6	25%	71	108	40	46	40	46	20	6
sqkp11			50%	109	184	40	46	40	46	20	6
sqkp12			75%	166	298	40	46	40	46	20	6
sqkp13	50	3	25%	376	655	100	103	100	103	50	3
sqkp14			50%	645	1193	100	103	100	103	50	3
sqkp15			75%	977	1857	100	103	100	103	50	3
sqkp16		6	25%	348	602	100	106	100	106	50	6
sqkp17			50%	658	1222	100	106	100	106	50	6
sqkp18			75%	984	1874	100	106	100	106	50	6
sqkp19	100	3	25%	1305	2413	200	203	200	203	100	3
sqkp20			50%	2597	4997	200	203	200	203	100	3
sqkp21			75%	3790	7383	200	203	200	203	100	3
sqkp22		6	25%	1303	2412	200	206	200	206	100	6
sqkp23			50%	2580	4966	200	206	200	206	100	6
sqkp24			75%	3841	7488	200	206	200	206	100	6

Table 2
Description of medium-sized quadratic knapsack problems.

ID	*n*	*m*	Density	LIN1		LIN2		LIN3		QP	
				# vars	# cons	# vars	# cons	# vars	# cons	# vars	# cons
mqkp1	200	2	25%	5200	10,046	400	402	400	402	200	2
mqkp2			50%	9995	19,824	400	402	400	402	200	2
mqkp3			75%	14,990	30,080	400	402	400	402	200	2
mqkp4		4	25%	5108	9884	400	404	400	404	200	4
mqkp5			50%	10,040	19,926	400	404	400	404	200	4
mqkp6			75%	14,881	29,838	400	404	400	404	200	4
mqkp7	300	3	25%	11,359	22,235	600	603	600	603	300	3
mqkp8			50%	22,311	44,421	600	603	600	603	300	3
mqkp9			75%	33,428	67,183	600	603	600	603	300	3
mqkp10		6	25%	11,474	22,444	600	606	600	606	300	6
mqkp11			50%	22,482	44,828	600	606	600	606	300	6
mqkp12			75%	33,549	67,478	600	606	600	606	300	6
mqkp13	400	4	25%	20,391	40,136	800	804	800	804	400	4
mqkp14			50%	39,745	79,310	800	804	800	804	400	4
mqkp15			75%	59,438	119,552	800	804	800	804	400	4
mqkp16		8	25%	20,134	39,606	800	808	800	808	400	8
mqkp17			50%	39,854	79,578	800	808	800	808	400	8
mqkp18			75%	59,475	119,590	800	808	800	808	400	8
mqkp19	500	5	25%	31,413	62,039	1000	1005	1000	1005	500	5
mqkp20			50%	62,188	124,259	1000	1005	1000	1005	500	5
mqkp21			75%	92,991	186,933	1000	1005	1000	1005	500	5
mqkp22		10	25%	31,441	62,090	1000	1010	1000	1010	500	10
mqkp23			50%	62,358	124,564	1000	1010	1000	1010	500	10
mqkp24			75%	93,209	187,354	1000	1010	1000	1010	500	10

3.1. *Results for small problems*

Table 4 presents the solution times and optimal objective function values for the small problems described in Table 1. All four models quickly produced optimal solutions for each of the 24 problems. The times shown in the table are the total time for the branch and cut procedure to terminate naturally with a proven optimal solution. The smallest solution time for each problem is highlighted in bold.

Within this small problem category, LIN1 has a slight edge for the $n=10$ and $n=20$ variable problems in terms of solution times but there is little difference here in the performance of all four models. (For the $n=20$ problems, the time performance of LIN1 is closely followed by that of LIN3, LIN2 and QP in this order.) Over

Table 3
Description of large quadratic knapsack problems.

ID	n	m	Density	LIN1		LIN2		LIN3		QP	
				#vars	#cons	#vars	#cons	#vars	#cons	#vars	#cons
lqkp1	800	5	25%	80,652	160,079	1600	1605	1600	1605	800	5
lqkp2			50%	160,192	320,243	1600	1605	1600	1605	800	5
lqkp3			75%	238,741	478,993	1600	1605	1600	1605	800	5
lqkp4		10	25%	80,352	159,476	1600	1610	1600	1610	800	10
lqkp5			50%	159,938	319,654	1600	1610	1600	1610	800	10
lqkp6			75%	238,571	478,744	1600	1610	1600	1610	800	10
lqkp7		15	25%	80,781	160,320	1600	1615	1600	1615	800	15
lqkp8			50%	159,765	319,359	1600	1615	1600	1615	800	15
lqkp9			75%	239,182	480,031	1600	1615	1600	1615	800	15

Table 4
Computation times and optimal solutions for the small problems.

ID	LIN1		LIN2		LIN3		QP	
	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution
sqkp1	**0.00**	648	**0.00**	648	**0.00**	648	**0.00**	648
sqkp2	**0.02**	505	0.03	505	0.02	505	0.03	505
sqkp3	0.06	290	0.05	290	**0.03**	290	0.06	290
sqkp4	**0.01**	383	0.02	383	**0.01**	383	0.02	383
sqkp5	**0.00**	963	0.01	963	0.02	963	0.01	963
sqkp6	**0.00**	234	0.01	234	0.02	234	0.02	234
sqkp7	0.25	1919	0.57	1919	**0.12**	1919	0.74	1919
sqkp8	0.33	521	**0.14**	521	0.19	521	0.24	521
sqkp9	1.61	4274	0.58	4274	**0.34**	4274	1.02	4274
sqkp10	**0.14**	1011	0.42	1011	0.16	1011	0.46	1101
sqkp11	**0.2**	1762	0.31	1762	0.38	1762	0.65	1762
sqkp12	3.50	3186	4.62	3186	**1.18**	3186	2.93	3186
sqkp13	**0.35**	11,790	0.67	11,790	0.55	11,790	1.01	11,790
sqkp14	15.43	11,223	3.55	11,223	**1.89**	11,223	2.19	11,223
sqkp15	5.54	19,512	3.23	19,512	3.26	19,512	**2.40**	19,512
sqkp16	2.87	5573	1.84	5573	**1.01**	5573	1.93	5573
sqkp17	3.93	3658	**2.39**	3658	2.42	3658	2.99	3658
sqkp18	3.48	3408	2.65	3408	3.34	3408	**2.57**	3408
sqkp19	11.28	17,140	4.53	17,140	4.54	17,140	**3.50**	17,140
sqkp20	327.95	26,029	119.10	26,029	9.31	26,029	**4.41**	26,029
sqkp21	1454.75	15,267	21.25	15,267	17.78	15,267	**3.41**	15,267
sqkp22	1121.95	10,071	136.2	10,071	**76.21**	10,071	144.18	10,071
sqkp23	3470.62	30,403	452.77	30,403	175.34	30,403	**117.86**	30,403
sqkp24	39.92	3427	3.26	3427	18.32	3427	**1.30**	3427

the six problem instances for $n=10$ and $n=20$, each of the three linear models gave the smallest CPU time on at least one instance. For the $n=50$ problems, the relative performance of LIN1 and LIN2 drops off with LIN3 and QP providing the best time performance. Finally, for the $n=100$ problems, QP generally provides the best time performance followed by LIN3, LIN2 with LIN1 coming in at a distant 4th place. In fact, for $n=100$ the solution times for QP are typically an order of magnitude smaller than those of LIN1.

Table 5 presents the number of nodes to termination and the value of the initial relaxation for each model for the 24 small problems. LIN1 gave the strongest relaxation on all 24 of the problems. LIN3 gave the weakest relaxation on 6 problems, while LIN2 gave the weakest relaxation on 3 problems and QP produced the weakest relaxation on 15 problems in this problem set. The node counts given in Table 5 need to be interpreted with the understanding that the branch and cut procedure used on all four models employs CPLEX standard heuristic procedures intended to enhance the relaxations found before any branching takes place. For these small problems, this heuristic enhancement was very productive for all four models. For example, on 18 of the 24 problems, optimal solutions were obtained via LIN1 without any branching at all. Similar results were obtained for LIN2, LIN3 and, to a lesser extent, for QP. As expected, the largest problems in this set (i.e., the $n=100$ problems) generally took the most branching for all four models.

Tables 4 and 5 illustrate the somewhat confounding relationship between strength of the relaxation, total CPU time and node count as the efficiency of the tree search is influenced by not only the strength of the initial bound but how expensive it is to compute the bound as well as the impact of the heuristic enhancement. As commented earlier, LIN1 had the strongest relaxation for each of the 24 problems yet gave the best time performance on only 7 of the smaller problems in this set. Moreover, on the 50 variable problem sqkp16, LIN3 had the weakest relaxation yet turned in the best time performance on this problem. Among three linear models, LIN3 requires more computation time when the density increases in this set, which is not observed in LIN1 and LIN2. Generally, though, for the larger problems in this set, the time advantage shifted to QP due to the smaller size of the relaxations to be solved. This trend is even more apparent for the medium and large problems of Tables 6 and 8.

Table 5
Total node count and initial relaxation for small problems.

ID	LIN1		LIN2		LIN3		QP	
	# Nodes	Relaxation	# Nodes	Relaxation	# Nodes	Relaxation	# Nodes	Relaxation
Sqkp1	0	672.7179	0	672.8684	0	683.4271	0	701.972
Sqkp2	0	604.9259	0	735.6531	0	725.3521	0	719.1087
Sqkp3	0	544.6291	0	678.9119	1	691.9716	0	607.913
Sqkp4	0	474.1621	2	478.7965	0	512.3386	0	532.9974
Sqkp5	0	969.4545	0	1008.165	1	1044.3126	0	1053.578
Sqkp6	0	407.7234	0	562.0818	2	584.2413	0	522.7983
Sqkp7	0	2130.142	0	2374.584	0	2357.07	0	2315.051
Sqkp8	0	843.6738	0	1160.06	1	1184.128	0	1117.33
Sqkp9	0	4561.95	0	4975.374	4	5035.286	4	5047.538
Sqkp10	0	1187.776	8	1487.334	3	1445.894	4	1395.088
Sqkp11	0	2041.573	0	2386.868	2	2401.236	0	2382.024
Sqkp12	0	3936.877	0	4365.794	0	4376.751	4	4348.526
Sqkp13	3	11862.92	0	12555.26	0	12597.35	0	12654.26
Sqkp14	0	11680.86	0	13533.13	1	13742.48	2	13978.25
Sqkp15	0	20205.62	0	22971.62	1	23274.25	2	23758.16
Sqkp16	0	5908.856	0	7022.584	2	7069.351	0	6931.953
Sqkp17	0	4350.807	0	6020.706	0	6421.793	4	6643.167
Sqkp18	0	4865.899	0	6889.734	3	7328.895	0	7754.318
Sqkp19	12	17309.21	0	20178.97	4	20436.76	0	20641.99
Sqkp20	32	30761.81	532	37466.65	683	38531.94	24	38988.69
Sqkp21	112	19748.36	114	27574.47	231	31742.89	8	32237.85
Sqkp22	374	12753.96	584	15579.49	712	15970.32	192	16088.91
Sqkp23	496	37415.03	1302	42827.29	1413	44817.46	356	45362.44
Sqkp24	0	4523.556	0	8254.965	2	9314.688	0	11975.47

Table 6
Computation times and best solutions found within a CPU time limit (7200 s) for the medium sized problems.

ID	LIN1		LIN2		LIN3		QP	
	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution
mqkp1	7200	177,030	1096.22	177,272	367.21	177,272	**24.10**	177,272
mqkp2	7200	11,533	1193.03	11,648	144.57	11,648	**4.57**	11,648
mqkp3	7200	465,570	665.51	470,218	246.13	470,218	**124.67**	470,218
mqkp4	7200	11,281	87.01	11,777	**37.12**	11,777	49.67	11,777
mqkp5	7200	61,856	1705.6	67,621	644.37	67,621	**77.21**	67,621
mqkp6	7200	26,044	861.98	26,696	229.01	26,696	**31.60**	26,696
mqkp7	7200	53,060	59.13	53,653	**26.38**	53,653	104.86	53,653
mqkp8	7200	691,414	3491.23	696,397	1216.92	696,397	**52.94**	696,397
mqkp9	7200	36,123	362.43	91,006	**189.25**	91,006	340.26	91,006
mqkp10	7200	26,098	2750.46	27,876	1635.43	27,876	**1074.06**	27,876
mqkp11	7200	43,743	1505.51	47,227	**879.23**	47,227	1226.28	47,227
mqkp12	7200	81,635	143.57	84,838	51.77	84,838	**4.86**	84,838
mqkp13	7200	257,511	3816.36	**259,319**	**3190.23**	259,**319**	7200	259,307
mqkp14	7200	63,506	**40.68**	64,168	68.14	64,168	549.53	64,168
mqkp15	7200	224,701	7200	618,252	7200	618,412	**2768.57**	620,562
mqkp16	7200	6531	933.23	8322	225.34	**8322**	**130.73**	8322
mqkp17	7200	99,846	7200	134,003	7200	135,013	**2973.91**	**135,675**
mqkp18	7200	**	7200	574,713	7200	579,931	**4389.21**	**588,088**
mqkp19	7200	55,697	7200	71,321	7200	72,004	**6537.59**	**72,684**
mqkp20	7200	3499	39.29	69,919	22.37	69,919	**7.45**	69,919
mqkp21	7200	343,995	7200	1,199,141	7200	1,199,084	**5054.64**	**1,199,742**
mqkp22	7200	77,887	7200	170,899	5731.2	170,939	**2719.44**	**170,939**
mqkp23	7200	**	33.30	29,697	**19.68**	29,697	42.68	29,697
mqkp24	7200	102,226	7200	607,902	**7200**	605,328	**3875.29**	**611,297**

3.2. *Results for medium-sized problems*

Table 6 reports the timing and solution value results for the medium-sized problems described in Table 2. These problems, ranging in size from 200 to 500 variables and 2 to 10 knapsack constraints, were given a 7200 s (i.e., 2 h) time limit. The values reported in the table are the best objective function values found in the time limit allowed.

LIN1 was unable to solve any of the 24 problems in this problem set within the allotted time. The LPs associated with LIN1 are too time-consuming to enable natural termination within 7200 s. In fact for two of the problems, the initial relaxation of LIN1 could not be solved in the time given. For all 24 problems, the best solutions found via LIN1 were inferior to those coming from LIN2, LIN3 and QP.

In contrast to the performance of LIN1, the other three models were relatively successful on these problems. As shown in Table 6, LIN2 gave optimal solutions for 17 out of 24 problems while LIN3 produced optimal solution for 18 out of 24 problems and QP reported optimal solutions for 23 out of the 24 problems. On the 16 problems where all three models gave the optimal solution, QP had a smaller CPU time on 10 problems followed by LIN3 with 5 and

LIN2 with 1 in this order. Generally, however, the CPU time to find and prove optimality was considerable shorter for QP than for LIN2 and LIN3. On those problems where the three models terminated naturally, QP reported the optimal solution in an average time of 255 s while LIN2 took an average of 995 s and LIN3 used an average of 690.24 s. However, the impact of density on CPU time found in small problem set with LIN3 was not observed in this set.

Table 7 presents the node counts at termination and the initial relaxations for these problems. For these 24 problems, QP gave the weakest initial relaxations. Yet, QP generally had lower nodes counts and clearly delivered the best overall performance for these medium-sized problems in terms of solution quality and solution time.

3.3. Results for large problems

Tables 8 and 9 present the computational results for the $n=800$ problems of Table 3. For these problems we restrict our attention to the models LIN3, LIN2 and QP as the LPs associated with LIN1 are too large to make it competitive. Table 8 lists the best solution found for each problem within an allotted time of 14,400 s (4 h). Note that none of three models produced a proven optimal solution within the time limit given but that QP dominated LIN2 and LIN3 across the board in terms of solution quality.

In an effort to see how close the results of Table 8 are to optimal solutions, we arbitrarily chose lqkp2 on model QP and solved it again with the time limit removed. This produced an optimal solution of 3,304,703 at a CPU time of 42,060 seconds (more than 11 h). Thus the gap between the optimal solution and the best solution found (objective function=3,304,401) within 14,400 s is roughly .009%. The gaps for the other problems are not known at this time but we suspect they are small as well.

3.4. Overall assessment

Based on the results given in the above tables, no conclusions can be drawn about the impact of either density or the number of knapsack constraints on CPU time. For a given number of variables and knapsack constraints, sometimes, but not always, low-density problems take longer to solve than higher density problems except

Table 8
Computational results for large problems: best solutions found within a CPU limit of 14,400 s.

ID	LIN2		LIN3		QP	
	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution
lqkp1	14,400	386,714	14,400	386,706	14,400	**414,796**
lqkp2	14,400	3,291,313	14,400	3,291,768	14,400	**3,304,401**
lqkp3	14,400	2,857,042	14,400	2,877,312	14,400	**2,941,027**
lqkp4	14,400	264,911	14,400	270,458	14,400	**283,141**
lqkp5	14,400	1,054,419	14,400	1,054,587	14,400	**1,106,566**
lqkp6	14,400	758,346	14,400	773,197	14,400	**801,725**
lqkp7	14,400	723,419	14,400	731,940	14,400	**742,162**
lqkp8	14,400	526,883	14,400	530,871	14,400	**553,301**
lqkp9	14,400	1,280,871	14,400	1,290,073	14,400	**1,304,561**

Table 9
Computational results for large problems: total # nodes, and initial relaxations within a CPU Limit of 14,400 s.

ID	LIN2		LIN3		QP	
	# Nodes	Relaxation	# Nodes	Relaxation	# Nodes	Relaxation
lqkp1	305	745648.9	598	779145.6	723	838164.7
lqkp2	260	4015129.0	477	4033187.3	990	4207809.0
lqkp3	500	4070389.0	377	4116588.1	486	4371211.0
lqkp4	157	630446.2	514	683945.7	750	717126.6
lqkp5	536	1814310.0	378	1931456.1	646	2024964.0
lqkp6	155	1823784.0	363	1942358.8	544	2106518.0
lqkp7	1543	1101091.0	1026	1104275.0	618	1193577.0
lqkp8	225	1163669.0	433	1284391.3	1100	1380316.0
lqkp9	1999	1992258.0	1209	2128634.6	450	2321545.0

Table 7
Total node count and initial relaxation for the medium sized problems

ID	LIN1[a]		LIN2		LIN3		QP	
	# Nodes	Relaxation	# Nodes	Relaxation	# Nodes	Relaxation	# Nodes	Relaxation
mqkp1	37814	179409.2	918	189679.1	1381	188534.2	338	195952.0
mqkp2	297	13563.04	4476	26308.02	6642	28409.6	0	37647.54
mqkp3	571	484504.4	1114	513803.1	2315	517025.4	502	538595.7
mqkp4	315	14283.55	174	23090.7	411	25134.8	180	27737.15
mqkp5	17	100738.7	581	113915.8	1186	116745.5	204	124054.9
mqkp6	169	27938.32	0	47268.65	2	51809.7	10	67471.34
mqkp7	457	53299.39	170	76124.64	374	83214.6	872	89230.27
mqkp8	57	710377.7	3486	756355.7	5128	775908.3	228	791227.9
mqkp9	1	110427.6	354	162576.2	647	185463.4	1568	209261.8
mqkp10	45	38509.74	1265	58628.16	3193	63712.4	3622	71986.15
mqkp11	5	54772.74	1048	88549.9	2514	103241.5	476	116356.5
mqkp12	146	82606.3	296	138670.6	303	175939.6	6	188541.6
mqkp13	31	265988.6	3218	327432.7	4528	329672.0	26,380	345776.6
mqkp14	29	69101.97	674	118285.0	817	1173581.4	3108	175772.4
mqkp15	0	639669.2	1484	813689.7	1891	834712.5	3500	880660.2
mqkp16	9	16402.96	112	28368.61	183	33758.51	1208	41632.36
mqkp17	1	158269.1	1962	234792.8	1581	268141.7	13,206	290768.8
mqkp18	[b]	[b]	560	845227.1	734	863241.5	1896	927377.1
mqkp19	0	102,519	828	154908.4	621	171623.3	5684	196443.6
mqkp20	0	69466.03	70	136993.8	37	179023.7	2	218551.1
mqkp21	0	123857.9	5140	1521982	6186	1561731	1046	1621316
mqkp22	0	269308.8	3700	305345.8	4109	318906.1	1080	337596.1
mqkp23	[b]	[b]	40	67164.11	57	83461.4	324	133694.3
mqkp24	0	688,730	14,300	887967.1	17,214	931671.8	1898	1015191

[a] No optimal solution is found in LIN1.
[b] Unable to solve initial relaxation within 7200 s.

the LIN3 model in some small problems. Likewise, for a fixed number of variables and density, sometimes, but not always, instances with fewer knapsack constraints are more readily solved than those with a larger number of knapsack constraints. Additional testing will be needed to clarify these issues.

Overall, our computational results indicate that all four formulations worked well on the small problems. However, with growth in problem size, the performance of LIN1 lags behind that of LIN2, LIN3 and QP. Furthermore, as the growth continues, LIN2 and LIN3 lag behind QP. Considering just the three linearizations, LIN3 generally gave the best performance. While all three linearizations are very competitive with QP on small instances, QP performed well across all three problem sets. For the large instances, approaches based on linearizations run into computational difficulty due to the size of the LPs that need to be solved. Given that efficient implementations of branch and cut algorithms are now readily available for the QP formulation, QP stands out as the best of the four models considered here. Moreover, given the growth in size of the LPs inherent to any approach based on linearizations, the QP formulation may prove to be a very good choice compared to any linearization that might be considered.

4. Summary and conclusions

In this paper we compared three linearizations and the original quadratic formulation on a new set of test problems for the quadratic knapsack problem. Much of the work reported previously in the literature concerns the relative strength of the relaxations coming from various linearizations with little attention paid to actually finding optimal solutions and no attention at all paid to problems with multiple knapsack constraints. Our computational results indicate that the strength of the root relaxation is not necessarily a good indication of overall performance. This is consistent with findings reported in related papers.

Our results also indicate that problems with multiple knapsack constraints can be effectively solved by branch and cut methodologies and that the quadratic formulation, QP, was particularly effective for medium to large sized problems consisting of 200–500 variables and 800 variables, respectively.

In this study we successfully addressed larger problems than previously addressed in the literature. It is clear from our computational experience, however, that problems of the type considered here with 800 or more variables and multiple constraints remain computationally challenging for the exact methods we used in this study. Nonetheless, our results suggest that the original quadratic formulation, rather than resorting to linearizations of one kind or another, offers considerable promise as a modeling construct for solving the class of problems examined.

We note that the results reported here were obtained without the use of pre-processing intended to strengthen the relaxations associated with the four basic models considered here. Such efforts may lead to enhanced performance and will be investigated as part of our on-going research. Our future work will also explore the use of advanced starts coming from modern heuristics. Moreover, we intend to investigate the use of Semi-definite programming and other methods to improve the performance of QP as well as solving even larger instances than considered here.

Appendix. Report on level of difficulty of randomly generated test problems

For the large problem set, none of four models can be solved within the time limit of 3 h. This fact is a strong indicator of the difficulty of these test problems. However, for smaller problems, the level of difficulty cannot be measured solely by the CPU time even though QKP is NP-hard in general. Very few QPK studies include a discussion of problem difficulty. Hansen and Meyer [13] and Forrester [22] are rare exceptions. Both papers offer a discussion of difficulty and conclude that there is a positive correlation between variable orderings and CPU time. In this appendix, we give a brief report on two possible measures of difficulty. Specifically, we examine the impact of constraint coefficients (weight) to knapsack capacity as well as density of objective function coefficients on CPU time.

1. Ratio of object weight to knapsack capacity (WC ratio): Most greedy heuristics and surrogate constraint heuristics for QKP create the initial solution by comparing the sum of object weights a_{ij} in the constraint and the knapsack capacity b_i and select the objects whose total weight will not exceed b_i. If the ratio $\sum a_{ij}/b_i$ is large, fewer objects can be chosen in an initial solution. If the ratio $\sum a_{ij}/b_i$ is small, more objects can be chosen in the initial solution, suggesting that the problem may be more difficult to solve. For QKP with multiple constraints, we examine each constraint and take the smallest ratio value for each problem into consideration. Then for all the test problems we were able to solve optimally, we examine the impact of this ratio on the CPU time. We report our results in Figs. 1 and 2.
2. Density of objective function coefficients (Q matrix): Previous studies [13,22] reported a positive correlation between density and solution time. Given their findings, we

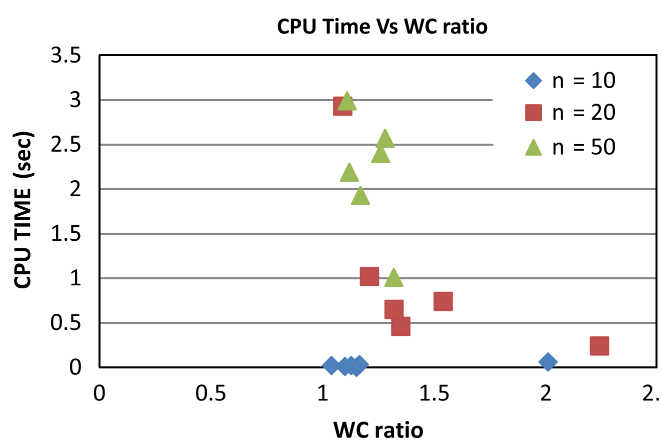

Fig. 1. The impact of WC-ratio on CPU time for $n=10, 20, 50$ in small problem set.

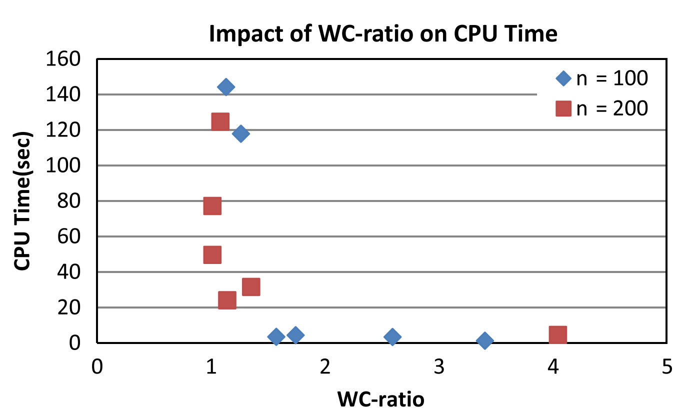

Fig. 2. The impact of WC-ratio on CPU time for $n=100,200$ in small to medium sized problem set.

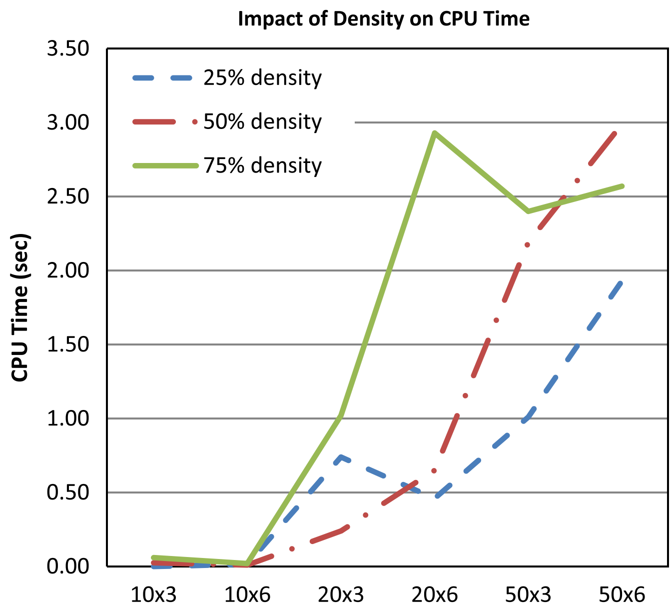

Fig. 3. The impact of density on CPU time for $n=10$, 20, 50 small problems.

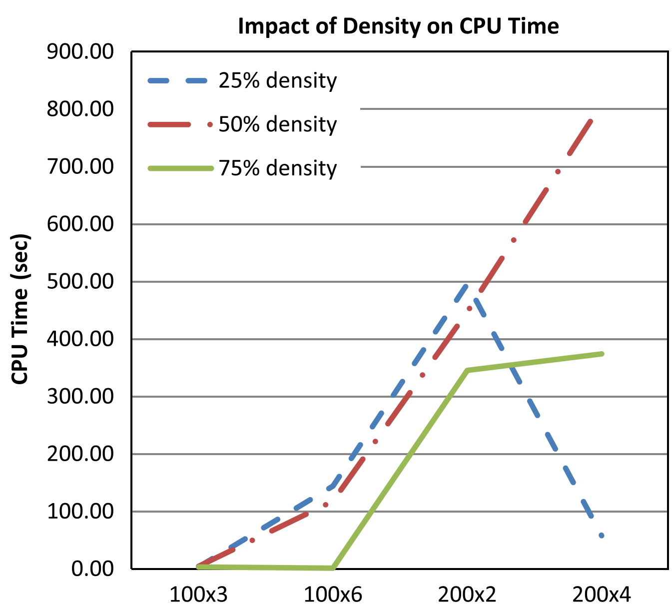

Fig. 4. The impact of density on CPU time for small to medium sized problems.

examined our results to see if similar patterns were on display. For the small and medium-sized problems we were able to solve to optimality, we examined the relationship between density of the Q matrix and CPU time. These results are presented in Figs. 3 and 4.

A.1. Tentative conclusions

Of the two measures highlighted in this appendix, the first approach (WC ratio) consistently indicates that problems with small ratios, as expected, tend to take more CPU time. Most of the problems in our test bed have small ratios indicating that they can be expected to be challenging. The $n=10$ variable problems are an exception to this. These problems are simply small and easy to solve regardless of the ratios.

Our results, however, are less clear with respect to our second measure where Fig. 3 and 4 show no clear relationship between density and CPU time. This is not consistent with earlier results reported in the literature and it suggests that further testing is necessary to more clearly establish the relationship between density and CPU time.

References

[1] Kochenberger G, Glover F, Alidaee B, Rego C. A unified modeling and solution framework for combinatorial optimization problems. Operations Research Spectrum 2004;26(2):237–50.
[2] Glover F, Kochenberger G, Alidaee B, Amini M. Solving Quadratic Knapsack Problems by Reformulation and Tabu Search: Single Constraint Case. In: Pardalos AMPM, Burkard R, editors. Combinatorial and Global Optimization. World Scientific Publishing Co.; 2002. p. 111–21.
[3] Billionnet A, Calmels F. Linear programming for the 0-1 quadratic knapsack problem. European Journal of Operational Research 1996;92(2):310–25.
[4] Michelon P, Veuilleux L. Lagrangean methods for the 0-1 quadratic knapsack problem. EJOR 1996;92:326–41.
[5] Caprara A, Pisinger D, Toth P. Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing 1999;11(2):125–37.
[6] Billionnet A, Soutif E. An exact method based on Lagrangian decomposition for the 0-1 quadratic knapsack problem. European Journal of Operational Research 2004;157(3):565–75.
[7] Lovász L, Schrijver A. Cones of matrices and set-functions and 0-1 optimization. SIAM Journal on Optimization 1991;1:166–90.
[8] Lasserre JB, editor. An explicit exact SDP relaxation for nonlinear 0-1 programs Lecture Notes in Computer Science ed. K.A.a.A.M.H. Gerards. Vol. 2081; 2001, p. 293–303.
[9] Billionnet A, Elloumib S, Plateaub M-C. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: the QCR method. Discrete Applied Mathematics 2009;157(6):1185–97.
[10] Sherali HD, Adams WP. A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Dordrecht, Boston, London: Kluwer Academic Publishers; 1999.
[11] Pisinger D. The quadratic knapsack problem: a survey. Discrete Applied Mathematics 2007;155(5):623–48.
[12] Pisinger WD, Rasmussen AB, Sandvik R. Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing 2007;19(2):280–90.
[13] Hansen P, Meyer C. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. Discrete Applied Mathematics 2009;157:1267–90.
[14] Hammer PL, Rubin AA. Some remarks on quadratic programming with 0/1 variables. Revue Francaise d'Informatique et al. Recherche Operationelle 1970;4(3):67–79.
[15] Billionnet A, Elloumi S. Using a mixed integer quadratic programming solver for unconstrained quadratic 0-1 problem. Mathematical Programming, Series A 2007;109:55–68.
[16] Glover F, Woolsey G. Converting the 0-1 polynomial programming problem to a 0-1 linear program. Operations Research 1974;22:180–2.
[17] Glover F. Improved linear integer programming formulations of non-linear integer problems. Management Science 1975;22:450–60.
[18] Grainger D, Letchford A. Improving a Formulation of the Quadratic Knapsack problem, in working paper. 2007, Lancaster University.
[19] Adams W, Forrester R. Linear forms of nonlinear expressions: new insights on old ideas. OR Letters 2007;35:510–8.
[20] Glover F. An improved MIP formulation for products of discrete and continuous variables. Journal of Information and Optimization Sciences 1984;5(1):69–71.
[21] Gallo G, Hammer P, Simeone B. Quadratic knapsack problems. Mathematical Programming Study 1980;12:132–49.
[22] Forrester RJ. Compact formulations of 0-1 quadratic program. 1997 [cited; Available from: http://www.theforresterfamily.com/dick/rjforrester_thesis.pdf.