Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
CNRS UMR 7024

CAHIER DU LAMSADE

260

Juillet 2007

Exact solution method to solve large scale integer quadratic multidimensional knapsack problems

Dominique Quadri, Eric Soutif, Pierre Tolla

Exact solution method to solve large scale integer quadratic multidimensional knapsack problems[1]

D. Quadri*, E. Soutif§, P. Tolla*

Abstract

In this paper we develop a branch-and-bound algorithm for solving a particular integer quadratic multi-knapsack problem. The problem we study is defined as the maximization of a concave separable quadratic objective function over a convex set of linear constraints and bounded integer variables. Our exact solution method is based on the computation of an upper bound and also includes pre-procedure techniques in order to reduce the problem size before starting the branch-and-bound process. We lead a numerical comparison between our method and three other existing algorithms. The approach we propose outperforms other procedures for large-scaled instances (up to 2000 variables and constraints).

Key words : Integer programming, separable quadratic function, linearization, surrogate relaxation, branch-and-bound

1 Introduction

We develop in this paper an exact solution method for solving a particular class of non-linear knapsack problems. The integer quadratic multidimensional knapsack problem $(QMKP)$, which is known to be NP-hard [13] (Lueker, 1975), involves the maximization of a concave quadratic and separable function over a convex set of linear constraints

[1]A extended abstract of this paper appeared in LNCS 4362 pp. 456-464, 2007

[†]LAMSADE, Université Paris-Dauphine, 75775 Paris cedex 16, France. `{quadri, tolla}@lamsade.dauphine.fr`

[¶]CEDRIC, Conservatoire des Arts et Métiers, 292 rue Saint Martin, 75003 Paris, France. `soutif @cnam.fr`

(called knapsack constraints) and bounded (pure) integer variables. It can be written as:

$$(QMKP)\left\{\begin{array}{ll} \max & f(x)=\sum_{j=1}^{n} f_j(x_j)=\sum_{j=1}^{n} c_j x_j - d_j x_j^2 \\ s.t. & \left| \begin{array}{l} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, i=1,...,m \\ 0 \leq x_j \leq u_j, j=1,...,n \\ x_j \in \mathbb{N}, j=1,...,n \end{array}\right. \end{array}\right.$$

where the coefficients c_j, d_j, a_{ij}, b_i are nonnegative. The bounds u_j of variables x_j are integers such that $u_j \leq (\frac{c_j}{2d_j})$ because of the concavity of each f_j, $x_j^* \leq (\frac{c_j}{2d_j})$, where x_j^* is the optimal solution of the program $\max_{x_j \geq 0} f_j(x_j)$.

Problem $(QMKP)$ has wide applications, including financial models [4] (Djerdjour et al., 1988), [5] (Faaland, 1974) production and inventory management [2] (Bretthauer and Shetty, 2002). However, most of the knapsack problems literature has addressed attention to specialized versions. For instance, there are large bodies of literature focusing on knapsack problems with linear objective function and a single constraint or m contraints (see [6] (Fayard and Plateau, 1974), [14] (Martello and Toth, 1983), [7] (Fréville and Hanafi, 2005), on knapsack problem with the quadratic objective function of $(QMKP)$ over a single linear constraint and 0-1 or pure integer variables (see [1] (Billionnet and Soutif, 2004), [16] (Pisinger et al., 2007), [2], [15] (Mathur and Salkin, 1983)). Nevertheless several applications require the use of pure integer variables and m capacity constraints, such as capital budgeting.

Even if many theoretical approaches have been proposed to solve integer quadratic problems (see for instance [3] (Cooper, 1981), [11, 12] (Korner 1985 and 1990), only few of these methods have been implemented in practice so few experimental results can be found. There have been a limited number of papers studying $(QMKP)$. Djerdjour et al. (1988) [4] proposed a branch-and-bound algorithm specifically designed to solve $(QMKP)$, unfortunately the computational results were limited to instances up to 20 variables and constraints. However, the real-world applications require effective methods which are able to solve large scale instances of the problems. Since, $(QMKP)$ is NP-hard, one should not expect to find a polynomial time algorithm for solving it exactly. Hence, we are usually interested in developing branch-and-bound algorithm. A key step in designing an effective exact solution method for such a maximization problem is to establish a tight upper bound on the optimal value. In a previous study [17] (Quadri et al., 2007), we present an effective upper bound method based on a linearization and surrogate relaxation. We have compared analytically and computationally our upper bound to the bound suggested by Djerdjour et al. (1988) [4], the basic LP-relaxation of $(QMKP)$, and the LP-relaxation of the equivalent linearized formulation (MKP). Simulation results, over a set of large instances (up to 2000 variables and constraints), showed that our upper bound is of good quality and closer to the optimum than the three other. Finally, the proposed upper bound can be computed in a very quick CPU time.

We propose in this present study, using our tight upper bound [17] (Quadri et al., 2007) to develop an exact solution method to solve $(QMKP)$. Our algorithm also incorporate preprocessing procedures in order to reduce the size of the initial problem before starting the branch-and-bound and a heuristic to find a good feasible solution given in [17] (Quadri et al., 2007). We then compare the computational time of our method with three other existing methods which include the three upper bounds previously mentioned. Our computational testing, presented latter in this paper, over a set of large instances (up to 2000 variables and constraints) evidences the computational performance of our branch-and-bound and its ability to solve real-sized instances.

The paper is organized as follows. The next section is dedicated to a description of the existing exact solution methods to $(QMKP)$. Section 3 summarizes the upper and lower bounds algorithms we proposed in a previous work and we use in the proposed branch-and-bound to solve $(QMKP)$. Our branch-and-bound also incorporates pre-processing procedures which are details in Section 3. The computational results are reported in Section 4. We finally conclude in Section 5.

In the remainder of this paper, we adopt the following notations: letting (P) be an integer or a 0-1 program, we will denote by $(\overline{P})$ the continuous relaxation problem of (P). We let $Z[P]$ be the optimal value of the problem (P) and $Z[\overline{P}]$ the optimal value of $(\overline{P})$.

2 Existing exact solution methods

In this section we describe three existing exact solution methods to tackle $(QMKP)$. Each of them include an upper bound whose quality has been established in a previous work in [17] (Quadri et al., 2007). We will compare the computational performance of the exact solution method we proposed with these three other branch-and-bound algorithms in Section 4.

The Djerdjour, Mathur and Salkin exact solution method (DMS) [4]. These authors develop a branch-and-bound algorithm based on the computation of an upper bound of better quality than the one provided by solving the LP-relaxation of $(QMKP)$. This upper bound is computed solving a polynomial problem derived from a linearization of the initial problem and a surrogate relaxation. The authors first use a direct expansion originally proposed by Glover [8] (Glover, 1975) of the integer variables and apply a piecewise linear interpolation to the initial objective function initially used by [15] (Mathur and Salkin, 1983).

A zero-one linearization branch-and bound (LBB). Mathur and Salkin develop in [15] (Mathur and Salkin, 1983) an exact solution method for the single constraint integer quadratic knapsack problem. These authors transform the initial quadratic problem into a 0-1 linear equivalent formulation by applying a direct expansion on the integer variables and a linear interpolation on each quadratic function f_j. They simply solve the equivalent problem using a solver so as to obtain the optimal value of the original problem. The 0-1 linearization branch-and-bound we suggest to test to solve to optimality $(QMKP)$ is a straightforward application of [15] (Mathur and Salkin, 1983) to the case of m capacity constraints. Following the approach of these authors we convert $(QMKP)$ to a 0-1 multidimensional knapsack problem (MKP) which optimal value is equal to the one of $(QMKP)$ (the formulation of (MKP) is given in Section 3.1). The corresponding branch-and-bound algorithm computes at each node of the search tree the LP-relaxation of (MKP) which provides an upper bound of the optimal value of (MKP). This approach has never been used to solve directly $(QMKP)$.

A standard branch-and-bound algorithm (SBB). Since the objective function of $(QMKP)$ is concave and the feasible set is convex the LP-relaxation of this problem can be found and thus provides an upper bound for $(QMKP)$. Consequently, a classical branch-and-bound consists of computed this bound at each node of the search tree.

3 The proposed branch-and-bound algorithm

In this section we develop a branch-and-bound algorithm which key step is the computation of a tight upper bound [17] (Quadri et al., 2007). This upper bound method involves two improvements of the algorithm proposed by Djerdjour et al. (1988) [4] summarized in Section 2. We also develop a heuristic which provides a good feasible solution we use as a starting point of the process. Before starting the solution procedure we implement three pre-processing procedures to reduce the problem size. We implemented the proposed branch-and-bound algorithm using a best bound strategy which chooses the node with the best (the lowest) upper bound.

3.1 Upper bound computation

The proposed upper bound is computed by applying a linearization and a surrogate relaxation presented in [17] (Quadri et al., 2007), which is derived from two improvements of the bound suggested by Djerdjour et al. (1988) [4]. The first one speeds up the computational time of the bound. The second improvement increase the quality of the bound. We now summarize the main steps of our method to get a tight upper bound for $(QMKP)$.

First an equivalent formulation of $(QMKP)$ is obtained by using a direct expansion of the integer variables x_j and by applying a piecewise linear approximation to the initial problem as employed in [4] (Djerdjour et al., 1988). Consequently, $(QMKP)$ is converted into a $0-1$ multidimensional knapsack problem (MKP):

$$(MKP)\left\{\begin{array}{ll} \max \sum_{j=1}^{n}(\sum_{k=1}^{u_j} s_{jk}y_{jk}) & \\ s.t. & \left| \begin{array}{l} \sum_{j=1}^{n}(a_{ij}\sum_{k=1}^{u_j} y_{jk}) \leq b_i,\ i=1,...,m \\ y_{jk}\in\{0,1\} \end{array}\right. \end{array}\right.$$

where $\sum_{k=1}^{u_j} y_{jk} = x_j$, $s_{jk} = f_{jk} - f_{j,k-1}$ and $f_{jk} = c_j k - d_j k^2$.

The second step consists in getting the surrogate problem associated to the LP-relaxation of (MKP) as also discussed in [4] (Djerdjour et al., 1988). The resultant problem (KP, w) can be written as

$$(KP,w)\left\{\begin{array}{ll} \max \sum_{j=1}^{n}(\sum_{k=1}^{u_j} s_{jk}y_{jk}) & \\ s.t. & \left| \begin{array}{l} \sum_{j=1}^{n}[\sum_{i=1}^{m} w_i a_{ij}]\sum_{k=1}^{u_j} y_{jk} \leq \sum_{i=1}^{m} w_i b_i \\ y_{jk}\in\{0,1\} \end{array}\right. \end{array}\right.$$

The LP-relaxation of (KP, w) provides an upper bound for $(QMKP)$ (a proof is given in [17] (Quadri et al., 2007)). Nevertheless the quality of this bound depends on the surrogate vector employed to establish the surrogate problem.

The optimal surrogate multiplier, denoted by w^*, corresponds to the optimal solution of the dual surrogate problem (SD) (see [9] (Glover, 1975)). The problem (SD) can be expressed as $\min_{w\geq 0} Z[\overline{KP,w}]$. Since the objective function of (SD) is quasi-convex Djerdjour et al. (1988) [4] solve (SD) via a local descent method which is shown in [17] (Quadri et al., 2007) to be very time consuming. We then suggest, in [17] (Quadri et al., 2007) , an alternative way to compute w^* which is assessed to be quicker than the local descent method, using the result that the optimal solution of dual of $(\overline{MKP})$ provides an optimal solution of (SD).

Experiment results assess the computational efficiency of this alternative way for computing w^* (see [17] (Quadri et al., 2007)).

The second improvement proceeds from an additional stage in which we solve (KP, w^*) in 0-1 variables rather than in continuous variables as it is done by Djerdjour et al. (1988) in [4]. Analytically the quality of the bound is increased. The experiments quantify the gap provides by this additional step.

3.2 Lower bound computation

In this section we sum up the main steps to get a feasible solution for $(QMKP)$. This lower bound is used in [17] (Quadri et al., 2007) to assess the quality of our upper bound.

In this present work it will be used as a good starting solution. The main idea of the proposed heuristic is the following. Since it is proved in [17] (Quadri et al., 2007) that the optimal value of $(\overline{MKP})$ is closer to the optimum of $(QMKP)$ than the optimal value of $(\overline{QMKP})$, we start from the optimal solution, y^*, of $(\overline{MKP})$, to tackle a feasible solution for $(QMKP)$. Letting $\sum_k y^*_{jk} = \alpha_j$, for each variable x_j of $(QMKP)$, we add to $(QMKP)$ the constraint $\lfloor \alpha_j \rfloor \leq x_j \leq \lfloor \alpha_j \rfloor + 1$, where $\lfloor \alpha_j \rfloor$ denotes the greatest integer smaller or equal to α_j. Thus, each variable becomes bivalent, and since the objective function is separable, it can straightforwardly be shown that the resulting problem is a 0-1 linear multidimensional knapsack problem. Obviously, solving this knapsack problem yields a feasible solution for $(QMKP)$ which is not necessarily optimal for $(QMKP)$.

3.3 Pre-processing techniques

Most commercial integer programming solvers propose a pre-processing phase which simplifies the problem being solved before starting a branch-and-bound procedure. This pre-processing phase consists of reducing the number of variables and constraints of the initial problem, using logical issues and variable fixation techniques. In this section we describe the pre-processing procedures we implemented in order to speed up the solution of $(QMKP)$ and discuss the utility of such procedures.

We use three pre-processing procedures: the first one detects and discards some redundant constraints, the second one reduces the bound of the integer variables (using [10] (Hammer et al., 1975)), and the third and last one fixes some 0-1 variables to their optimal values.

3.3.1 Eliminating redundant constraints

We first discard constraints which are trivially implied by a strong one. Since $(QMKP)$ only includes knapsack constraints, this strong constraint is simply found by considering the constraint with the smallest right-hand-side, say b_{i_0}. Then we set $\alpha_i = \max_{1 \leq j \leq n} \frac{a_{ij}}{a_{i_0 j}}$. Suppose that a constraint i verifies $\alpha_i \leq \frac{b_i}{b_{i_0}}$, we have:

$$\forall j \in \{1, \ldots, n\}, \; \frac{a_{ij}}{a_{i_0 j}} \leq \alpha_i,$$

and since for all i and j $a_{ij} \geq 0$:

$$\forall x \text{ verifying constraint } i_0, \; \sum_{j=1}^{n} a_{ij} x_j \leq \sum_{j=1}^{n} \alpha_i a_{i_0 j} x_j \leq \alpha_i \sum_{j=1}^{n} a_{i_0 j} x_j \leq \alpha_i b_{i_0} \leq b_i$$

Consequently, when a constraint i satisfies $\alpha_i \leq \frac{b_i}{b_{i_0}}$, this constraint is immediately satisfied when constraint i_0 is verified. This criterion allows us to detect the constraints obviously dominated by constraint i_0.

We could iterate this technique using not only the constraint included the smallest right-hand-side but also the one with the second smallest right-hand-side (and so on). However, this idea does not appear experimentally efficient.

This simple technique, whose complexity is $O(mn)$, allowed us to significantly reduce the number of constraints of all the considered instances. As discussed in Section 4. its efficiency strongly depends of the structure of the considered instance. Indeed, we randomly generate three kinds of instances: squared instances ($n = m$), rectangular instances ($m = 5\%n$) and correlated instances. The reductions obtained are of good quality (between 52% and 56% of the constraints can be removed) concerning the squared and correlated instances.

3.3.2 Reducing the bounds of the integer variables

Since the objective function f of problem $(QMKP)$ is separable into n concave subfunctions f_j, each integer variable x_j is naturally bounded by $\frac{c_j}{2d_j}$. Consequently, the bound u_j of variable x_j is always chosen between 1 and $\max\left(1, \frac{c_j}{2d_j}\right)$ when instances are randomly generated. However a high value of parameter u_j may considerably increase the difficulty to solve the problem (for example the most difficult instances generated in Section 4 involve integer bounds up to 70).

Hammer et al. (1975) propose in [10] a technique called constraint pairing to reduce the bounds of some integer variables verifying a set of linear inequalities. The idea of this method is to arbitrarily choose two inequalities from the set, say $\sum_{j=1}^{n} a_j x_j \leq a_0$ and $\sum_{j=1}^{n} b_j x_j \leq b_0$, and to establish a third surrogate one obtained by multiplying the second constraint by a positive parameter t and by summing it with the first constraint. The surrogate constraint is: $\sum_{j=1}^{n}(a_j + tb_j)x_j \leq (a_0 + tb_0)$. When t is judiciously chosen, the surrogate constraint may allow us to tighten the bounding interval of variable x_j, $[0, u_j]$, for instance by noting that variable x_j may no more take the value 0 or u_j without violating the surrogate constraint. The authors show it suffices to successively affect only $n + 2$ values to t in order to get as many conclusions as possible about the bounds of all the variables from the two constraints initially chosen. The complexity of the whole procedure is $O(n^2)$ for each couple of pairing inequalities.

This technique seems relatively useful if we consider the experimental results in Section 4. These experiments show for instance that the number of pure integer variables (those having $u_i > 1$) decrease on average from 70% to 40% for the correlated instances and from 38% to 21% for the squared instances.

3.3.3 Fixing some 0-1 variables to their optimal value

When applying the last pre-processing technique, the number of 0-1 variables has probably increased. Thus, it is natural to try to fix some of the 0-1 variables to their optimal value in order to decrease the problem size. Let a 0-1 variable, say x_i, we first need to guess its value in the optimal solution. We use the value of x_i in the best known feasible solution and try to prove that x_i cannot have a different value in any optimal solution. Considering a set I of variables indices that all are supposed to be equal to 0 in an optimal solution, we add the following constraint to the problem:

$$\sum_{i \in I} x_i \geq 1$$

then we compute an upper bound UB of the modified problem using the method presented in [17] (Quadri et al., 2007). If the computed upper bound is strictly lower than the value of the best known feasible solution then we can conclude that all the variables indexed in I can definitively be fixed to 0. A similar constraint can be added for fixing variables to 1 instead of 0 as explained in [1] (Billionnet and Soutif, 2004).

The size of the set I is an important parameter for the success of such a fixation method. Choosing $|I| = 1$ comes down to fix variables one by one, and choosing $|I|$ too large generally fails since no fixation can be done (the quality of the upper bound does not allow us to simultaneously fix all the variables indexed in I when I is too large). Experiments show that for $(QMKP)$ and the bound used to solve it, a good compromise consists in choosing $|I| = 2$.

This procedure allows us to fix on average 51% of the $0-1$ variables. However it appears much more time-consuming than the two previous ones. It can sometimes constitute more than 10% of the whole CPU-time necessary to find the exact solution using the branch-and-bound procedure.

4 Computational results

The computational performance of the four branch-and-bound procedures was tested through a set of 100 randomly generated squared problems (i.e. $n = m$) and correlated problems (i.e. $c_j = \sum_{i=1}^{m} a_{ij}$ and $d_j = c_{min}/2$, where c_{min} is the minimum of all c_j values). The correlated problems are well-known to be more difficult to solve in practice for $0-1$ linear multidimensional knapsack problem (MKP), which is a special case of $(QMKP)$, than squared instances. The number of variables take their values in the set $\{100, 500, 1000, 1500, 2000\}$, with 10 replications per n-value. As in [4] (Djerdjour et al., 1988) integer coefficients a_{ij}, c_j, d_j were uniformly drawn at random

Table 1: Average computational time of the four methods for each problem type

n	m	SBB	LBB	DMS	Our BB
Squared					
100	100	1.31	0.63	2.13	1.5
500	500	120.12	16.1	-	11.5
1000	1000	346.56	283.3	-	50.5
1500	1500	1178.4	392.5	-	183.7
2000	2000	2557.9	1369.4	-	305.2
Correlated					
100	100	0.7	0.1	3.35	0.4
500	500	955.7	129.4	-	410.4
1000	1000	1480.24	650.13	-	1280.2
1500	1500	-	-	-	-
2000	2000	-	-	-	-

in the range $\{1..100\}$. Coefficients b_i and u_j are integers uniformly distributed such that $50 \leq b_i < \sum_{j=1}^{n} a_{ij}u_j$ and $1 \leq u_j \leq \lceil \frac{c_j}{2d_j} \rceil$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

The percentage of pure integer variables (with $u_j > 1$) rises 38.6% on average for squared instances rather than it is equal to 70% on average for correlated problems. The average value of the bound u_j for the pure integer variables reaches 21.2 with a standard deviation 2.24 over 50 instances for squared problems and rises 54.1 on average with a standard deviation 6 over 50 for correlated problems.

Our branch-and-bound as well as *DMS* were coded in C language whereas the optimal solution provided by *SBB* and *LBB* were got using a commercial solver ILOG-Cplex9.0. Simulations were run on a bi-xeon 3.4 Ghz with 4Go of main memory.

4.1 General comments on the comparison of the computational time of the four methods

Table 1 displays, for the four exact solution methods mentioned in this paper, the average computational time in CPU seconds over 10 replications for each problem type and size. We limited to 10800 seconds (3 hours) the solution time.

The results in Table 1 indicate clearly the poor performance of *DMS* which was not able to reach the optimum in the chosen limit time for all problems types with 500 variables and more. It is proved in [17] (Quadri et al.) that is due to the fact that *DMS* employs a very time consuming local descent search method to obtain the optimal surrogate multiplier w^* (see Sections 2 and 3).

Squared instances. Our branch-and-bound clearly outperforms the three other procedures. For large scaled instances ($n \geq 1000$) our exact solution method is approximately 4 times quicker than the second best approach *LBB* and it is almost 7 times faster than *SBB*. This conforts the ability of our method to solve large sized problems in a very competitive time with an average of $5''05$ minutes (305 seconds) for the 2000-variables instances.

Correlated instances. As expected the correlated problems are difficult to be solved by the four exact methods. The size of the instances only up to 1000 variables and constraints. Nevertheless, concerning 500-variable problems LBB is the most appropriate algorithm taking $10''00$ minutes on average to find the optimum. Our branch-and-bound still solves the same instances in a competitive CPU time (2 times slower than *LBB*).

4.2 Analysis of the experiments results

The improvement capability of our branch-and-bound can be explained by three features, namely: (i) the feasible solution, (ii) the upper bound and (iii) the pre-processing procedures.

Our feasible solution (see [17] (Quadri et al., 2007). Actually, providing to our branch-and-bound algorithm another feasible solution (of poorer quality) did not significantly debase its solution time. Consequently, the good quality of the initial feasible solution does not clearly explain the performance of our exact solution method.

Our upper bound and pre-processing procedures: squared instances. The impact of both the upper bound we proposed in [17] (Quadri et al., 2007) and the pre-procedures used clearly appear. Indeed, our upper bound is always closer to the optimum than the three other utilized in DMS, LBB and SBB. Moreover, this bound takes about 10 seconds on average to be computed. On the other hand, our method includes part of the procedure of Hammer et al. (1975) [10] to reduce the bounds u_j of the pure variables. This actually leads to convert many pure integer variables into 0-1 variables. The average proportion of pure integer variables decreases from 38% to 21.02%. The fixation technique eliminates 50.25% of variables which value is fixed to 1. Finally, the search of redundant constraints allows to keep 48% of the initial constraints. Consequently, both the upper bound method and the pre-procedure techniques are very effective for squared instances.

Our upper bound and pre-processing procedures: correlated instances. The upper bound is of poor quality in comparison with the result obtain for the squared problems.

Nevertheless, the relative gap is on average 20% and the bound is computed in $1''30$ on average (see [17] (Quadri et al., 2007)). On the other hand, the pre-procedure techniques are quite well effective. Indeed, 56.6% of constraints can be removed on average, 27% of variables are fixed to their bound. Before starting the pre-procedure process, only 30% of variables are binaries. The impact clearly appears: 60% of variables are $0-1$ after applying the techniques. In spite of the fact that these results are of good quality, our branch-and-bound does not solve 1000-variable problems faster than *LBB*.

5 Conclusion

We develop in this paper a branch-and-bound algorithm to solve the integer quadratic multidimensional knapsack problem. The upper bound and the pre-procedure techniques that we incorporate allow us to solve large-sized squared instances, up to 2000 variables and constraints. The classical branch-and-bound, which is usually used to solve such convex problems is of poor quality as well as the method proposed by Djerdjour et al. (1988) [4]. Moreover, the method suggested by [15] (Mathur and Salkin, 1983) we extended to the case of $(QMKP)$, which has never been utilized to solve $(QMKP)$ directly, outperforms the three other branch-and-bound concerning the correlated problems.

References

[1] A. Billionnet and E. Soutif. An exact method based on lagrangian decomposition for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 157 (3):565–575, 2004.

[2] K. Bretthauer and B. Shetty. The nonlinear knpasack problem - algorithms and applications. *European Journal of Operational Research*, 138 (3):459–472, 2002.

[3] M. Cooper. A survey of methods for pure nonlinear integer programming. *Management Science*, 27 (3):353–361, 1981.

[4] M. Djerdjour, K. Mathur, and H. Salkin. A surrogate-based algorithm for the general quadratic multidimensional knapsack. *Operation Research Letters*, 7 (5):253–257, 1988.

[5] B. Faaland. An integer programming algorithm for portfolio selection. *Management Science*, 20 (10):1376–1384, 1974.

[6] D. Fayard and G. Plateau. Reduction algorithms for single and multiple constraints 0-1 linear programming problems. *International conference methods of mathematical programming proceedings*, Zakopane, Poland, 1977.

[7] A. Fréville and S. Hanafi. The multidimensionnal 0-1 knapsack problems - bounds and computational aspects. *Annals of Operations Research*, 139:195–227, 2005.

[8] F. Glover. Improved linear integer progamming formulations of nonlinear integer problems. *Management Science*, 22 (4):455–460, 1975.

[9] F. Glover. Surrogate constraint duality in mathematical programming. *Operations Research*, 23:434–451, 1975.

[10] P.L. Hammer, M. W. Padberg, and U.N. Peled. Constraints pairing in integer programming. *INFOR*, 13 (1):68–81, 1975.

[11] F. Körner. Integer quadratic progarmming. *European Journal of Operational Research*, 19 (2):268–273, 1985.

[12] F. Körner. On the numerical realization of the exact penalty method for quadratic programming algorithms. *European Journal of Operational Research*, 46 (3):404–408, 1990.

[13] G.S. Lueker. Two np-complete problems in nonnegative integer programming. *Computer Science Labatory, Princeton, NJ*, report 178 (A6), 1975.

[14] S. Martello and P. Toth. Knapsack problems: algorithms and computer implementations. *John Wiley and Sons, Inc. New-York, USA*, 1983.

[15] K. Mathur and H. Salkin. A branch and bound algorithm for a class of nonlinear knapsack problems. *Operations Research Letters*, 2 (4):155–160, 1983.

[16] D. Pisinger, A. Rasmussen, and R. Sandvick. Solution of large quadratic knapsack problems through agressive reduction. *INFROMS Journal on Computing*, 19 (2), 2007.

[17] D. Quadri, E. Soutif, and P. Tolla. Upper bounds for large scale integer quadratic multidimensional knapsack problems. *International Journal of Operations Research*, To appear. A draft of this paper is available as a technical report CEDRIC 1117, 2007.