Reoptimization in Lagrangian methods for the 0-1 quadratic knapsack problem

Lucas Létocart[1], Anass Nagih[2], Gérard Plateau[1]

November 8, 2010

1. LIPN - UMR CNRS 7030
Institut Galilée, Université Paris 13
99, Avenue J-B. Clément
93430 Villetaneuse, France.
{*lucas.letocart, gerard.plateau*}*@lipn.univ-paris13.fr*

2. LITA
UFR MIM, Université Paul Verlaine
Ile du Saulcy
57045 Metz Cedex 1, France.
anass.nagih@metz.fr

Abstract

The 0-1 quadratic knapsack problem consists of maximizing a quadratic objective function subject to a linear capacity constraint. To exactly solve large instances of this problem with a tree search algorithm (e.g., a branch and bound method), the knowledge of good lower and upper bounds is crucial for pruning the tree but also for fixing as many variables as possible in a preprocessing phase. The upper bounds used in the best known exact approaches are based on Lagrangian relaxation and decomposition. It appears that the computation of these Lagrangian dual bounds involves the resolution of numerous 0-1 linear knapsack subproblems. Thus, taking this huge number of resolutions into account, we propose to embed reoptimization techniques for improving the efficiency of the preprocessing phase of the 0-1 quadratic knapsack resolution. Namely, reoptimization is introduced to accelerate each independent sequence of 0-1 linear knapsack problems induced by the Lagrangian relaxation as well as the Lagrangian decomposition. Numerous numerical experiments validate the relevance of our approach.

1 Introduction

The 0-1 quadratic knapsack problem (QKP) consists of maximizing a quadratic objective function subject to a linear capacity constraint. It can be formulated as follows:

$$(QKP)\begin{cases} \max & \sum_{i\in N}\sum_{j\in N} c_{ij}x_i x_j \\ s.t. & \sum_{j\in N} a_j x_j \leq b \\ & x_j \in \{0,1\} \quad j \in N \end{cases} \tag{1}$$

where $N = \{1, \ldots, n\}$, the coefficients a_j $(j \in N)$ and c_{ij} $(i \in N, j \in N)$ are positive integers and b is an integer such that $max\{a_j : j \in N\} \leq b < \sum_{j\in N} a_j$.

Problem (QKP) is a generalization of the 0-1 linear knapsack problem, in which all the $c_{ij} = 0$, for all $i \neq j$ and it is known to be *NP*-hard. Literature surveys on solution approaches and applications (e.g., several classical graph theory problems (max clique, graph bipartitioning problem, k-clustering); economical problems (localization of railway stations and airports); compiler design problem) can be found in [17] and [31].

An important phase in the resolution of any combinatorial optimization problem is the preprocessing phase, which is all the more important since the instance sizes are large. It includes heuristics and relaxation methods for finding lower and upper bounds on the value of the optimal solution of the problem. Another important step for 0-1 programming problems aims at the reduction of the instance size (generally by fixing some variables at their optimal value) by exploiting these bounds. In addition, to solve exactly large instances with a tree search algorithm (e.g., a branch and bound method), the knowledge of good lower and upper bounds is also crucial for pruning the tree.

The choice of bounding methods is usually a tradeoff between the tightness of the bound obtained and the computation time to achieve it. As far as the 0-1 quadratic knapsack problem is concerned, the upper bounds produced by the best known exact approaches are based on Lagrangian relaxation and decomposition (see, e.g., [19, 27, 28, 33]) or semidefinite programming (see, e.g., [20, 21]).

To our knowledge, the best known exact approach is designed by Pisinger, Rasmussen and Sandwik [32] (denoted by *method PRS*). This approach combines a Lagrangian relaxation proposed by Caprara, Pisinger and Toth [11] (denoted by *method CPT*) and a Lagrangian decomposition proposed by Billionnet, Faye and Soutif [7, 8] (denoted by *method BFS*), taking advantage of the fact that the two methods *CPT* and *BFS* complete each other.

It appears that the computation of these Lagrangian dual bounds involves the resolution of numerous 0-1 linear knapsack subproblems. Thus, taking this huge number of resolutions into account, we propose to embed reoptimization techniques for improving the efficiency of the preprocessing phase for the 0-1 quadratic knapsack resolution.

In this paper, we propose to introduce *reoptimization* to accelerate each independent sequence of 0-1 continuous linear knapsack problems induced by the Lagrangian relaxation of Caprara, Pisinger and Toth's method (CPT) and the Lagrangian decomposition of Billionnet, Faye and Soutif's method (BFS). Two reasons for this choice: first, these two methods are the basic bricks of method PRS, and second, it was possible to get the two original software from the authors. Thus, this work propose to lay the foundations of *reoptimization* for (QKP) by a straightforward exploitation of the two existing codes.

Section 2 recalls the principle of methods CPT and BFS to compute upper bounds for problem (QKP). Section 3 deals with the introduction of reoptimization in the resolution of the 0-1 continuous linear knapsack problems solved in the preprocessing phase for the 0-1 quadratic knapsack problem. In this way, we propose to exploit reoptimization tools introduced successfully to improve the computation time of iterative algorithms (like subgradient methods) for a 0-1 bidimensional knapsack problem Lagrangian dual [36, 37]. Finally, numerous numerical experiments validate the relevance of our approach (Section 4).

2 Lagrangian upper bounds for problem (QKP)

The Lagrangian approaches CPT [11] and BFS [7, 8] for problem (QKP) combine linearization or decomposition techniques with Lagrangian relaxation or Lagrangian decomposition. The main steps of these dual methods are described below: the Lagrangian relaxation of method CPT in Section 2.1 and the Lagrangian decomposition of method BFS in Section 2.2. This emphasizes the need to solve a huge number of 0-1 linear knapsack problems (e.g., about n problems of size n at each step of a subgradient procedure for method CPT).

2.1 The Lagrangian relaxation in method CPT

Using a linear reformulation of (QKP), Caprara, Pisinger and Toth [11] propose a computation of an upper bound by considering its Lagrangian relaxation (after a linearization step), which is solvable through numerous 0-1 linear knapsack problems.

In short, the first step in this upper bounding procedure is a reformulation-linearization technique (RLT) like linearization [1]. In a second step, by relaxing n^2 symmetric constraints, the authors prove that each Lagrangian relaxation may be solved by means of n independent 0-1 linear knapsack subproblems plus one global 0-1 linear knapsack problem depending on the solutions of the n previous ones. In order to speed up the subgradient algorithm used to solve the Lagrangian dual, only the continuous relaxations of these $n + 1$ linear knapsack problems are considered at each iteration.

Namely, the first step of method CPT [11] consists of linearizing (QKP) as follows:

.

$$(ILP)\begin{cases} \max & \sum_{j\in N} c_{jj}x_j + \sum_{j\in N}\sum_{i\in N\setminus\{j\}} c_{ij}y_{ij} & & \\ \text{s.t.} & \sum_{j\in N} a_j x_j \leq b & & \\ & \sum_{i\in N\setminus\{j\}} a_i y_{ij} \leq (b-a_j)x_j & \forall j\in N & (2) \\ & 0 \leq y_{ij} \leq x_j \leq 1 & \forall (i,j)\in N^2, i\neq j & \\ & y_{ij} = y_{ji} & \forall (i,j)\in N^2, i<j & (3) \\ & x_i, y_{ij} \in \{0,1\} & \forall (i,j)\in N^2, i\neq j & \end{cases}$$

For this linearization, the binary variables y_{ij} are introduced to replace the product $x_i x_j$. These new variables are linked to the old one by suitable inequalities. Indeed, constraints (2) consists of multiplying the capacity constraint by each variable and replacing x_j^2 by x_j since the variables are binary variables. They are redundant as long as the integer restriction on the variables is imposed [1, 3, 17, 26]. Caprara, Pisinger and Toth solve this problem by a branch and bound method. Their evaluation of nodes is based on a Lagrangean relaxation $(LR_{ILP}(\mu))$ which relaxes the symmetric constraints (3), by denoting μ_{ij} the dual multipliers associated with y_{ij}. The authors highlight this decomposition result: for each $\mu \in \mathbb{R}^{n^2}$, solving $(LR_{ILP}(\mu))$ is equivalent to solve $n+1$ linear 0-1 knapsack problems. Namely, relaxing the integer conditions on variables (for monitoring time complexity), these knapsack problems are considered in a two level way as follows:

1. First, n 0-1 continuous linear knapsack problems $(CKP_j(\mu))$ $j \in N$:

$$\begin{cases} \max & \sum_{i\in N\setminus\{j\}} (c_{ij}+\mu_{ij})z_{ij} & \\ \text{s.t.} & \sum_{i\in N\setminus\{j\}} a_i z_{ij} \leq (b-a_j) & \\ & 0 \leq z_{ij} \leq 1 & \forall i \in N\setminus\{j\} \end{cases}$$

   associated with constraint (2) are solved. For each $j \in N$, let us denote by $\overline{z}_j$ an optimal solution of $(CKP_j(\mu))$ and by $\overline{p}_j$ its value.

2. Second, it remains to solve the following 0-1 continuous linear knapsack problem (CKP) associated with the capacity constraint (1), whose objective function includes the results $\overline{p}_j$ $j \in N$ of the first level:

$$\begin{cases} \max & \sum_{j\in N} (\overline{p}_j + c_{jj})x_j & \\ \text{s.t.} & \sum_{j\in N} a_j x_j \leq b & \\ & 0 \leq x_j \leq 1 & \forall j \in N \end{cases}$$

   Let us denote by $\overline{x}$ its optimal solution.

Therefore, an optimal solution $\overline{y}$ of $(LR_{ILP}(\mu))$ can be simply computed in this way:

$$\overline{y}_{ij} = \overline{z}_{ij}\overline{x}_j \quad \forall i,j \in N, i \neq j.$$

Thus, this evaluation is realized in $0(n^2)$ time since each continuous linear knapsack problem may be performed in $0(n)$ time [15, 16].

As far as numerical experiments are concerned, Caprara, Pisinger and Toth highlight that their approach leads to a good quality bound (denoted further by ub_{CPT}) for high density problems and needs low computational time.

2.2 The Lagrangian decomposition in method BFS

Using a preliminary data partition of the problem, Billionnet, Faye and Soutif [7] propose the computation of an upper bound ub_{BFS} for $v(QKP)$ based on Lagrangian decomposition.

In brief, their method BFS consists of three major steps:

- *Creation of clusters*: partition of the variable set into m disjoint classes k, $k \in \{1, \ldots, m\}$.
- *Lagrangian decomposition*: splitting the problem into m independent 0-1 linear knapsack subproblems, associated with the m clusters.
- *Computation of the upper bound*: solving the Lagrangian dual problem by means of a subgradient algorithm.

By denoting n_k the number of variables for cluster k, $k = 1, \ldots, m$, at each iteration of their subgradient algorithm, 2^{n_k} 0-1 linear knapsack problems have to be solved for each cluster k. But, in order to speed up the dual resolution, the authors propose to relax the integrality conditions on the variables. Thus, like for method CPT described in the previous section, they only consider the continuous relaxation of all the 0-1 linear knapsack subproblems.

Namely, for each cluster $k \in \{1, \ldots, m\}$, we propose to describe the generic problem which has to be solved among the 2^{n_k} 0-1 linear knapsack problems, at each iteration of the associated Lagrangian decomposition dual resolution. For this, let us give the main points of the method.

First, m clusters of variables are created (the authors do not propose any order to realize this partition but suggest to fix empirically a common size (e.g., 5) to each cluster). Let us denote by I_k the index set associated with variables of cluster k, $k = 1, \ldots, m$ (so $n_k = |I_k|$). A copy y of variable x is then created in this way:

$$y_j^k = x_j, \quad j \in N \backslash I_k, \quad k = 1, \ldots m$$

in order to consider a new formulation of problem (QKP):

$$(QKP') \begin{cases} max \sum_{k=1}^{m} f_k(x,y) & & \\ \text{s. t.} & & \\ \quad x_j = y_j^k & k = 1,...,m; j \in N \backslash I_k & (4) \\ \quad \sum_{i \in I_k} a_i x_i + \sum_{j \in N \backslash I_k} a_j y_j^k \leq b & k = 1,...,m & (5) \\ \quad x_i \in \{0,1\} & i \in I_k; k = 1,...,m & (6) \\ \quad y_j^k \in \{0,1\} & j \in N \backslash I_k; k = 1,...,m & (7) \end{cases}$$

where

$$f_k(x,y) = \sum_{i \in I_k} c_i x_i + \sum_{i \in I_k} \sum_{i' \in I_k, i' \neq i} c_{ii'} x_{i'} x_i + \sum_{i \in I_k} \sum_{j \in N \backslash I_k} c_{ij} x_i y_j^k$$

By relaxing the copy constraints (4) using the following multipliers:

$$\lambda = (\lambda_j^k)_{1 \leq k \leq m, j \in N \backslash I_k}$$

we get a Lagrangian relaxation:

$$\begin{cases} max \sum_{k=1}^{m} (f_k(x,y) + \sum_{j \in N \backslash I_k} \lambda_j^k (x_j - y_j^k)) & & \\ \text{s. t.} & & \\ \quad \sum_{i \in I_k} a_i x_i + \sum_{j \in N \backslash I_k} a_j y_j^k \leq b & k = 1,...,m & \\ \quad x_i \in \{0,1\} & i \in I_k; k = 1,...,m & \\ \quad y_j^k \in \{0,1\} & j \in N \backslash I_k; k = 1,...,m & \end{cases}$$

which may be decomposed into m subproblems. When variables x_i are fixed at 0 or 1, these subproblems are nothing else that 0-1 linear knapsack problems (in y) with size $n - n_k$.
In order to reduce computation times, the integrality constraints over variables y_j are relaxed for considering continuous knapsack problems, but on the other side, to improve the upper bound quality, the constraints of (QKP') are expanded by the following ones:

$$x_i y_j^k = x_j y_i^h \quad i \in N \backslash I_h; j \in N \backslash I_k; 1 \leq k < h \leq m$$

And by relaxing them, using the following multipliers:

$$\mu = (\mu_{ij}^{kh})_{i \in N \backslash I_h; j \in N \backslash I_k; 1 \leq k < h \leq m}$$

the resulting Lagrangian decomposition differs from the previous one only by a modification of the objective function which becomes:

$$\sum_{k=1}^{m}(f_k(x,y)+\sum_{j\in N\setminus I_k}\lambda_j^k(x_j-y_j^k)+\sum_{i\in N\setminus I_h;j\in N\setminus I_k;1\leq k<h\leq m}\mu_{ij}^{kh}(x_iy_j^k-x_jy_i^h))$$

and does not disturb the knapsack structure of the subproblems.

In summary, at each iteration of the Lagrangian decomposition dual resolution, for each cluster $k\in\{1,\ldots,m\}$, by considering the 2^{n_k} fixings of variables x_i in I_k, a 0-1 linear knapsack problem (in y_j^k) with size $n-n_k$ is solved in continuous. According to the data k,x,λ and μ (a large part of the previous objective function becomes a constant), it has the following form ($KP^k(x,\lambda,\mu)$):

$$\begin{cases} max\sum_{i\in I_k}\sum_{j\in N\setminus I_k}c_{ij}x_iy_j^k-\sum_{j\in N\setminus I_k}\lambda_j^ky_j^k+\sum_{i\in N\setminus I_h;j\in N\setminus I_k;1\leq k<h\leq m}\mu_{ij}^{kh}(x_iy_j^k-x_jy_i^h) \\ \text{s.t.} \\ \quad\sum_{j\in N\setminus I_k}a_jy_j^k\leq b-\sum_{i\in I_k}a_ix_i \\ \quad y_j^k\in\{0,1\}\quad j\in N\setminus I_k \end{cases}$$

It should be noted that m Lagrangian decomposition duals have to be solved to get the upper bound ub_{BFS}.

Billionnet and Soutif [8] highlight that their approach embedded in a branch and bound scheme, consumes more computation time but produces better experimental results than method CPT for low and middle density instances.

Summary

On one hand, we note that these two previous approaches involve a huge number of 0-1 linear knapsack resolutions at each step of a subgradient algorithm. On the other hand, the works of Thiongane, Nagih and Plateau [37] have shown that reoptimization techniques can be applied successfully to solve a sequence of 0-1 linear knapsack problems in a subgradient scheme. Thus, combining these two situations, as method PRS uses a hybridization of CPT and BFS methods (see Section 1), we propose now to introduce specific reoptimization techniques in these two basic approaches separately.

3 Reoptimization for problem (QKP)

In this paper, *reoptimization* consists of exploiting the resolution of a given instance of a problem (P) to accelerate the resolution of another instance (P') next to the

previous one (e.g., with a slight modification of part of the data). Thus, having reoptimization in mind, while solving (P) it is important to store information that might be useful for the resolution of (P'). Reoptimization techniques might be applied for a sequence of two instances or more generally for a larger sequence of instances (see, e.g., [5, 6, 12, 18, 22, 23, 25, 38, 39]).

As it is pointed out in the previous section, one of the critical things in the resolution of each Lagrangian dual of (QKP) is the huge number of *continuous 0-1 linear knapsack problems* to be solved. For improving the efficiency of the computation of the upper bounds ub_{CPT} and ub_{BFS}, we propose to reduce the global computation time of methods CPT and BFS by exploiting a part of the reoptimization techniques of method MARIE (Method combining Approximation and Reoptimization for Integer programming Evoluating instances) [35, 37], in the resolution of the (finite) sequences of 0-1 linear knapsacks.

Section 3.1 details the specific reoptimization tool we propose to embed in methods CPT and BFS for bounding and reducing problem (QKP). Section 3.2 describes the general reoptimization framework we design for the preprocessing phase of these two methods.

3.1 Reoptimization for the continuous 0-1 knapsack problem

The Lagrangian heuristic method MARIE was designed by Thiongane, Nagih and Plateau for the 0-1 linear bidimensional knapsack problem [35, 37]. MARIE combines a variable fixing phase and a local search for improving the lower bound on the value of the bidimensional knapsack produced by Lagrangian heuristics. Moreover, the sequence of 0-1 linear one-dimensional knapsack instances obtained from the algorithm are solved by using reoptimization techniques in order to reduce the total computational time effort. Computational results reported in [37] show that duality gaps produced by MARIE are close to zero with small CPU times.

Here we focus on the specific reoptimization tool that we propose to be embedded in methods (CPT) and (BFS) for bounding and reducing problem (QKP). We consider the following generic 0-1 linear knapsack problem:

$$(K(u)) \begin{cases} \max & \sum_{j\in I} c(u)_j x_j \\ s.t. & \sum_{j\in I} a_j x_j \leq cap \\ & x_j \in \{0,1\} \quad j \in I \end{cases}$$

where u is a positive real number, the index set I is a subset of N, the coefficients a_j $(j \in I)$ are positive integers, the coefficients $c(u)_j$ $(j \in I)$ are real numbers, and cap is an integer such that $max\{a_j : j \in I\} \leq cap < \sum_{j\in I} a_j$. This model stands for each knapsack problem solved in subgradient algorithms (with the Lagrangian

multiplier u) for solving the duals detailed in Section 2. Namely, all along the subgradient algorithms several sequences of 0-1 knapsack problems have to be solved. And in each sequence, all the problems have the same constraint and only differ, one to another, on the objective function via the parameter u.

Solving the LP relaxation ($\overline{K}(u)$) of ($K(u)$) is equivalent to determining an optimal multiplier $\mu(u)$ associated with the constraint of ($K(u)$). It is well known that:

$$\exists\ i \in I\ such\ that\ \mu(u) = \frac{c(u)_i}{a_i}$$

and two sets $U(u)$ and $L(u)$ forming a tripartition of I which satisfy the two properties:

$$(i)\ \forall\ j \in U(u)\ \frac{c(u)_j}{a_j} \geq \mu(u) \geq \frac{c(u)_l}{a_l}\ \forall\ l \in L(u)$$

$$(ii)\ \sum_{j \in U(u)} a_j \leq cap < \sum_{j \in U(u)} a_j + a_i$$

The LP solution $\overline{x}$ is then obtained by:

$$\overline{x}_j = \begin{cases} 1 & j \in U(u) \\ 0 & j \in L(u) \\ c(u)_i(\frac{cap - \sum_{j \in U(u)} a_j}{a_i}) & j = i \end{cases}$$

From the set of ratios $\frac{c(u)_j}{a_j}$, a sequence of tripartitions is realized from different target values while checking if condition (ii) holds. Different versions of the algorithm detailed in Figure 3.1 appeared simultaneously in 1977 in three papers reported in [4, 13, 24]. The version proposed here has an average linear time complexity [2, 15] (note that, in expectation of reoptimization, index i is saved even if $\overline{x}$ is an optimal solution of ($K(u)$) (in this case $\overline{x}_j$ is equal to 0 or 1)).

It is important to recall that the computational time of this algorithm will go down as each target value (also denoted below by *pivot*), chosen to realize the sequence of partitions, satisfies at least one of the following properties:

- *to be close to the median of the set of concerned ratios* $\frac{c(u)_j}{a_j}$
  This choice allows to create a sequence of equitable partitions of the ratio sets (see, e.g., [10, 29, 30, 34]). Namely, for each set of the sequence, about half of elements of the previous set is considered. This tends to approach a worst case linear time complexity, in a simpler way than the median method [2, 9] used in Lawler's algorithm [24].

- *to be close to the optimal multiplier* $\mu(u)$
  This property may be reached for the first pivot of the process when it is possible to exploit the data distribution of the generated instances (see for example, [14]). This choice tends to reduce drastically the number of

/* initialization */
$L \leftarrow I$; $U \leftarrow \emptyset$; $\sigma \leftarrow 0$

/* iterative algorithm */
while $\sigma < cap$ **do**
  choose an item h in L
  $L_1 \leftarrow \{j \in L |\ \mathrm{c(u)}_j/\mathrm{a}_j \geq (c(u)_h/a_h\} \backslash \{h\}$
  $L_2 \leftarrow \{j \in L |\ \mathrm{c(u)}_j/\mathrm{a}_j < c(u)_h/a_h\}$
  if $(\sigma + \sum_{L_1} a_j > cap)$ **then** $L \leftarrow L_1$
  else
    $\sigma \leftarrow \sigma + \sum_{j \in L_1} a_j; U \leftarrow U \cup L_1$
    if $(\sigma < cap)$ **then**
      $U \leftarrow U \cup \{h\}$; $\sigma \leftarrow \sigma + a_h$; $L \leftarrow L_2$
    endif
  endif
end while
$i \leftarrow h$ /*in expectation of reoptimization*/
$\overline{x}_j \leftarrow 1, j \in U$; $\overline{x}_j \leftarrow 0, j \notin U$
if $\sigma = cap$ **then**
  $\overline{x}$ is feasible and optimal for $(K(u))$; $v(K(u)) \leftarrow c(u)\overline{x}$
else
  $\sigma \leftarrow \sigma - a_i$
  $\overline{x}_i \leftarrow (cap - \sigma)/a_i; U(u) \leftarrow U \backslash \{h\}; L(u) \leftarrow N \backslash U; v(\overline{K}(u)) \leftarrow c(u)\overline{x}$
endif

Figure 3.1: Solving the continuous knapsack problem $(\overline{K}(u))$ in linear time complexity

iterations of the algorithm. For the best situation (the chosen initial pivot is optimal), this leads to the direct resolution of $(\overline{K}(u))$ with a single iteration. Our reoptimization process comes into this scope.

For the Lagrangian dual context, we recall that all along the subgradient algorithms, we have to solve a sequence of 0-1 knapsack problems which have the same constraint and only differ, one to another, on the objective function. In order to accelerate the computational time of the iterative procedures, which determine the optimal tripartition of I, we propose to reduce the number of its exchanges by using the optimal multiplier found at the previous iteration of the subgradient algorithm. In other words, let us consider at iteration l the index i^l of the optimal basic variable of the LP relaxation $(\overline{K}(u^l))$. For solving $(\overline{K}(u^{l+1}))$, we consider $c(u^{l+1})_{i^l}/a_{i^l}$ as the *first pivot*. In this way, we expect that this target value is not far from $\mu(u^{l+1})$ and that the first tripartition of I will be close to the optimal tripartition characterizing $\mu(u^{l+1})$ (see Figure 3.2).

3.2 Reoptimization in methods CPT and BFS

The computation of the Lagrangian dual bounds ub_{CPT} and ub_{BFS} involves the resolution of numerous 0-1 continuous linear knapsack subproblems (Section 2). In fact, each iterative preprocessing phase of methods CPT and BFS consists of a dynamic reduction of variables of problem (QKP), which both update the upper and lower bounds and the problem size until the reduction process fails. This leads to a huge number of 0-1 continuous knapsack problems in which we propose to embed the reoptimization technique detailed in Section 3.1.

We design a generic reoptimization framework (Figure 3.3), which respects our wish to realize a straightforward exploitation of the codes used by the authors in papers [11] for method CPT and [7, 8] for method BFS. As we only add adequate conditional instructions without changing the structure of the codes, we have to point out that, first the computation times could be greatly improved by using adequate data structures, and second, the reoptimization used for the variable fixing phase is far from the best possible procedure which should imply a complete rewriting of parts of the codes.

For giving an evaluation of the number of 0-1 continuous linear knapsack solved for computing bounds and fixing variables in the two codes, we denote by *tour* the number of iterations of the preprocessing phase and *iter* the number of iterations of the subgradient procedure for solving the Lagrangian dual (we denote by p the number of knapsack problems solved at each iteration). Moreover, we always denote by n the size of the current problem (QKP) at the beginning of a preprocessing phase (i.e., after possible reduction(s) of the problem):

Reoptimization in method CPT

The iterative preprocessing phase stops when no new variable can be fixed. This drastic stopping criterion gives value of parameter *tour*. The stopping criterion for the computation of ub_{CPT} is simply a prefixed number of iterations: *iter* is equal to $n+200$. Finally, parameter p is equal to $n+1$. We recall that it corresponds to n 0-1 continuous linear knapsack problems $(CKP_j(\mu))$ with size $n-1$ following by a unique 0-1 continuous linear knapsack problem (CKP) with size n (see Section 2.1).

Reoptimization in method BFS

The number of iterations of the preprocessing phase *tour* depends on stopping criteria based on size reduction threshold values. The number of iterations *iter* for computing ub_{BFS} derives from stopping criteria, which combine threshold values both for the standard duality *gap* defined as $\frac{upperbound-lowerbound}{lowerbound}$ and for the sizes of the Lagrangian multipliers. The parameter p is equal to $\sum_{k=1}^{m} 2^{n_k}$. Namely, for each cluster $k \in \{1, \ldots, m\}$, and for each of the 2^{n_k} fixings of variables x_i in I_k,

/* initialization */
$i^l \leftarrow$ *index of the optimal basic variable of* $(\overline{K}(u^l))$
$L \leftarrow$ *the index set of items in the order found at the end of the resolution of* $(\overline{K}(u^l))$
$h \leftarrow i^l$
$ratio \leftarrow c(u^{l+1})_h/a_h$
$L_1 \leftarrow \{j \in L | c(u^{l+1})_j/a_j \geq ratio\} \backslash \{h\}$
$L_2 \leftarrow \{j \in L | c(u^{l+1})_j/a_j < ratio\}$
if $(\sum_{L_1} a_j > cap)$ **then** $L \leftarrow L_1$
else
  $\sigma \leftarrow \sum_{j \in L_1} a_j$
  if $(\sigma = cap)$ **or** $(\sigma + a_h = cap)$ **then**
    /* $(\overline{K}(u^{l+1}))$ and $(K(u^{l+1}))$ are both solved */
    if $(\sigma = cap)$ **then** $U(u^{l+1}) \leftarrow L_1$ **else** $U(u^{l+1}) \leftarrow L_1 \cup \{h\}$ **endif**
    $i^{l+1} \leftarrow h$; $\overline{x}_j \leftarrow 1, j \in U(u^{l+1})$; $\overline{x}_j \leftarrow 0, j \notin U(u^{l+1})$; $v(K(u^{l+1})) \leftarrow c(u^{l+1})\overline{x}$
  else
    $U \leftarrow L_1$
    if $\sigma + a_h > cap$ **then**
      /* $(\overline{K}(u^{l+1}))$ is solved */
      $i^{l+1} \leftarrow h$; $\overline{x}_h \leftarrow (cap - \sigma)/a_h$; $U(u^{l+1}) \leftarrow U$
      $\overline{x}_j \leftarrow 1, j \in U(u^{l+1})$; $\overline{x}_j \leftarrow 0, j \notin U(u^{l+1}) \cup \{h\}$; $v(\overline{K}(u^{l+1})) \leftarrow c(u^{l+1})\overline{x}$
    else /* $\sigma + a_h < cap$*/
      $\sigma \leftarrow \sigma + a_h; U \leftarrow U \cup \{h\}; L \leftarrow L_2$

/* iterative algorithm */
  see Figure 3.1 with $u = u^{l+1}$

    endif
  endif
endif

/* information stored for the next problem instance */
  save i^{l+1} *and the current order of set* L

Figure 3.2: Reoptimization for the continuous knapsack problem $(\overline{K}(u^{l+1}))$ after solving $(\overline{K}(u^l))$

$tour \leftarrow 1$
repeat
  $l \leftarrow 1$
  /* upper bound computation */
  repeat
    apply *reoptimization* for each of the p independent 0-1 continuous linear knapsack problems $(\overline{K_j}(u^{l+1}))$ knowing the resolution of $(\overline{K_j}(u^l))$ $j \in \{1, \ldots p\}$ (see Figure 3.2)
    $l \leftarrow l + 1$
  until (stopping criteria)
  /* variable fixing */
  /* let us denote by *iter* the number of iterations for computing the upper bound, and by $\underline{x}$ the solution associated with the best known lower bound*/
  for each free variable x_f **do**
    apply *reoptimization* for each of the p independent 0-1 continuous linear knapsack problems $(\overline{K_j}(u^{iter})|x_f$ fixed at $1 - \underline{x}_f)$ knowing the resolution of $(\overline{K_j}(u^{iter}))$ $j \in \{1, \ldots p\}$ with the aim to fix x_f at $\underline{x}_f$ (see Figure 3.2)
  endfor
$tour \leftarrow tour + 1$
until (stopping criteria)

Figure 3.3: Generic reoptimization framework

a 0-1 continuous linear knapsack problem with size $n - n_k$ has to be solved (see problem $(KP^k(x, \lambda, \mu))$ of Section 2.2).

4 Computational results

This section is devoted to an experimental study over randomly generated instances. Using a uniform distribution, the data belong to the following intervals: $c_{ij} \in [1, 100]$, $a_j \in [1, 50]$ and $b \in [1, \sum_{j \in N} a_j]$ for $i, j \in \{1, \ldots, n\}$ with $n \in \{100, \ldots, 600\}$ for method CPT and $n \in \{100, \ldots, 400\}$ for method BFS. In the four tables, δ represents the density of the quadratic matrix (25 %, 50 %, 75 % and 100 %) and the results are average values over 10 instances. All the experiments have been carried out on an Intel Xeon bi-processor dual core 3 GHz with 2 GB of RAM (only one core has been used).

The experiments are designed to appreciate the intrinsic impact of reoptimization in methods CPT and BFS. Starting from the original codes, we only expand them by our reoptimization tools without modifying the parameters worked by the authors. In particular, we keep the original size (5) for the clusters in method BFS (it should be noted that the gains are of the same order whatever the cluster sizes). Moreover, for the original code of method BFS, core memory problems appear from instances with 500 variables. This explain the limitation of experiments for BFS.

Each column *Aver.* reports the average values of savings in terms of exchanges (Tables 1 and 3) and of CPU times (Tables 2 and 4) realized into the continuous knapsack resolutions of the preprocessing phase. The average values of standard deviations of columns *Dev.* refine these results.

The results show that the use of reoptimization techniques in the preprocessing phase leads to an average exchange saving of 28.53% for method CPT and 27.73% for method BFS. Moreover, we note that without implementing any dedicated data structures, we get an average time saving of 4.15 % for method CPT and 21.27 % for method BFS. Standard deviations highlight regular gains for method CPT. On the other hand, the more important standard deviation values indicate more chaotic gains for method BFS, even if all the mean gains remain positive in all cases.

n \ δ	25 %		50 %		75 %		100 %	
	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
100	14.13	10.31	31.41	12.30	46.97	8.62	56.80	4.33
200	17.13	10.88	24.52	7.95	46.41	9.38	58.05	4.56
300	20.59	11.35	22.19	9.24	18.23	13.66	53.38	2.40
400	12.50	8.09	12.90	9.85	32.76	5.01	53.30	3.86
500	10.90	4.91	16.77	13.36	32.51	14.08	53.52	3.35
600	7.86	2.68	27.27	10.16	27.88	9.93	51.13	3.91

Table 1: Number of exchanges saved for method CPT (%)

n \ δ	25 %		50 %		75 %		100 %	
	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
100	6.11	7.43	7.66	6.73	20.67	15.94	17.60	7.69
200	3.70	7.73	4.81	20.36	12.70	12.36	19.60	8.84
300	8.11	11.73	2.35	6.88	5.12	8.70	11.73	8.60
400	3.31	5.42	1.22	4.11	4.15	4.21	13.70	5.42
500	0.68	6.54	2.70	8.59	7.36	11.14	13.43	6.82
600	1.85	3.59	2.08	4.05	0.80	11.60	11.89	6.95

Table 2: CPU time (s) saved for method CPT (%)

n \ δ	25 %		50 %		75 %		100 %	
	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
100	33.66	33.65	19.94	32.89	51.09	74.54	74.81	17.86
200	1.19	1.56	67.94	20.12	36.76	54.94	22.67	34.56
300	63.27	32.30	43.69	42.17	0.37	5.62	75.15	58.10
400	41.45	43.80	62.50	31.00	6.87	11.58	0.11	23.31

Table 3: Number of exchanges saved for method BFS (%)

n \ δ	25 %		50 %		75 %		100 %	
	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.	Ave.	Dev.
100	8.29	12.64	5.00	4.39	28.59	28.06	37.47	15.57
200	0.78	0.86	34.15	37.97	25.45	44.40	8.26	15.95
300	43.96	29.28	29.54	21.29	1.57	5.29	36.69	33.51
400	27.93	31.39	42.96	20.21	11.56	17.97	2.65	11.68

Table 4: CPU time (s) saved for method BFS (%)

5 Conclusion

Caprara, Pisinger and Toth [11] and Billionnet, Faye and Soutif [7, 8] have proposed efficient Lagrangian based methods for solving the 0-1 *integer quadratic* knapsack problem. As their preprocessing phases involve the resolution of numerous 0-1 *integer linear* knapsack problems, we have proposed to improve the efficiency of these two well known methods by introducing reoptimization techniques. We designed a generic reoptimization framework for these two different Lagrangian methods by exploiting the resolution of the huge number of 0-1 *continuous linear* knapsack problems embedded in their iterative schemes. Using the original codes of the authors expanded by our reoptimization tools, the numerical experiments validate the relevance of our approach. In the near future, we intend to redesign the variable reduction phase by using adequate data structures of the two codes.

References

[1] W. P. Adams and H. D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32:1274–1290, 1986.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Pub. Company, 1976.

[3] E. Balas, S. Ceria, and G. Cornuejols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[4] E. Balas and E. Zemel. Solving large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[5] T. Belgacem and M. Hifi. Sensitivity analysis of the knapsack sharing problem: perturbation of the weight of an item. *Computers and Operations Research*, 35:295–308, 2008.

[6] T. Belgacem and M. Hifi. Sensitivity analysis of the optimum to perturbation of the profit of a subset of items in the binary knapsack problem. *Discrete Optimization*, 5:755–761, 2008.

[7] A. Billionnet, A. Faye, and E. Soutif. A new upper bound for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 112:664–672, 1999.

[8] A. Billionnet and E. Soutif. Using a mixed integer programming tool for solving the 0-1 quadratic knapsack problem. *INFORMS Journal on Computing*, 16:188–197, 2004.

[9] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and Systems Sciences*, 7(4):448–461, 1972.

[10] Bourgeois Ph. and G. Plateau. *Selected algorithmic tools for the resolution of the 0-1 knapsack problem*, EURO XII-TIMS XXXI Joint International Conference, Helsinki (Finland), 1992.

[11] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.

[12] A. Crema. An algorithm for the multiparametric 0-1 integer linear programming problem relative to objective function. *European Journal of Operational Reshearch*, 125:18–24, 2000.

[13] D. Fayard and G. Plateau. *Reduction algorithm for single and multiple constraints 0-1 linear programming problems*, Methods of Mathematical Programming, Zakopane (Poland), 1977.

[14] D. Fayard and G. Plateau. *Contribution à la résolution des programmes mathématiques en nombres entiers*, Thèse d'état, Université de Lille 1 (France), 1979.

[15] D. Fayard and G. Plateau. Algorithm 47: an algorithm for the solution of the 0-1 knapsack problem. *Computing*, 28:269–287, 1982.

[16] D. Fayard and G. Plateau. An exact algorithm for the 0-1 collapsing knapsack problem. *Discrete Applied Mathematics*, 49:175–187, 1994.

[17] G. Gallo, P.L. Hammer, and B. Simeone. Quadratic knapsack problems. *Mathematical Programming Study*, 12:132–149, 1980.

[18] A. M. Geoffrion and R. Nauss. Parametric and postoptimality analysis in integer linear programming. *Management Science*, 23:453–466, 1977.

[19] S. Gueye and Ph. Michelon. Miniaturized linearizations for quadratic 0/1 problems. *Annals of Operations Research*, 140:235–261, 2005.

[20] C. Helmberg and F. Rendl. Solving quadratic (0,1) problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82:291–315, 1998.

[21] C. Helmberg and F. Rendl. A semidefinite programming approach to quadratic knapsack problems. *Journal of Combinatorial Optimization*, 4:197–215, 2000.

[22] G. Hifi, H. Mhalla, and S. Sadfi. Sensitivity of the optimum to perturbations of the profit or weight of an item in the binary knapsack problem. *Journal of Combinatorial Optimization*, 10:239–260, 2005.

[23] S. Holm and D. Klein. Three methods for post-optimal analysis in integer linear programming. *Mathematical Programming*, 21:97–109, 1984.

[24] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4:339–356, 1979.

[25] M. Libura. Integer programming problems with inexact objective function. *Control and Cybernetics*, 9:189–202, 1980.

[26] L. Lovasz and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[27] Ph. Michelon and N. Maculan. Lagrangian methods for 0-1 quadratic knapsack problems. *Discrete Applied Mathematics*, 42:257–269, 1993.

[28] Ph. Michelon and L. Veilleux. Lagrangian methods for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 92:326–341, 1996.

[29] A. Nagih and G. Plateau. A Lagrangean decomposition for 0-1 hyperbolic programming problems. *International Journal of Mathematical Algorithms*, 1:299–314, 2000.

[30] A. Nagih and G. Plateau. A partition algorithm for 0-1 unconstrained hyperbolic programming problems. *Investigacion Operativa*, 9(1):167–178, 2000.

[31] D. Pisinger. The quadratic knapsack problem - a survey. *Discrete Applied Mathematics*, 155:623–648, 2007.

[32] D. Pisinger, A.B. Rasmussen, and R. Sandvik. Solution of large-sized quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing*, 19(2):280–290, 2007.

[33] Rodrigues C. D., D. Quadri, Ph. Michelon, S. Gueye, and M. Leblond. *Applying the T-linearization to the quadratic knapsack problem*, ROADEF 2009, Nancy (France), 2009.

[34] R. Sedgewick. Implementing quicksort programs. *Communications of the Association for Computing Machinery*, 21:847–857, 1978.

[35] B. Thiongane. *Réoptimisation dans le dual lagrangien du biknapsack en variables 0-1.* PhD thesis, Université Paris 13 (France), 2003.

[36] B. Thiongane, A. Nagih, and G. Plateau. An Adapted Step Size Algorithm for a 0-1 Biknapsack Lagrangean Dual. *Annals of Operations Research*, 139(1):353–373, 2005.

[37] B. Thiongane, A. Nagih, and G. Plateau. Lagrangean heuristics combined with reoptimization for the 0-1 bidimensional knapsack problem. *Discrete Applied Mathematics*, 154:2200–2211, 2006.

[38] N. Touati. *Amélioration des performances du schéma de la génération de colonnes : Application aux problèmes de tournées de véhicules.* PhD thesis, Université Paris 13 (France), 2008.

[39] Touati N., L. Létocart and A. Nagih. *Diversification and reoptimization procedures in column generation for the resolution of the acyclic vehicle routing problem with time windows*, Proceedings of INOC 2009 : International Network Optimization Conference, Pise (Italy), 6 pages, 2009.