Pesquisa Operacional (2014) 34(1): 49-72

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope

AN EFFICIENT HYBRID HEURISTIC METHOD FOR THE 0-1 EXACT *k*-ITEM QUADRATIC KNAPSACK PROBLEM

Lucas Létocart[1*], Marie-Christine Plateau[2] and Gérard Plateau[1]

Received March 11, 2013 / Accepted February 21, 2014

ABSTRACT. The 0-1 exact k-item quadratic knapsack problem ($E-kQKP$) consists of maximizing a quadratic function subject to two linear constraints: the first one is the classical linear capacity constraint; the second one is an equality cardinality constraint on the number of items in the knapsack. Most instances of this NP-hard problem with more than forty variables cannot be solved within one hour by a commercial software such as CPLEX 12.1. We propose therefore a fast and efficient heuristic method which produces both good lower and upper bounds on the value of the problem in reasonable time. Specifically, it integrates a primal heuristic and a semidefinite programming reduction phase within a surrogate dual heuristic. A large computational experiments over randomly generated instances with up to 200 variables validates the relevance of the bounds produced by our hybrid dual heuristic, which yields known optima (and prove optimality) in 90% (resp. 76%) within 100 seconds on the average.

Keywords: quadratic programming; 0-1 knapsack; quadratic convex reformulation; semidefinite programming; surrogate duality; hybridization, experiments.

1 INTRODUCTION

The 0-1 exact k-item quadratic knapsack problem consists of maximizing a quadratic function subject to a linear capacity constraint with an additional equality cardinality constraint:

$$(E-kQKP)\begin{cases} \max & f(x)=\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_ix_j \\ \text{s.t.} & \sum_{j=1}^{n} a_jx_j \le b \quad (1) \\ & \sum_{j=1}^{n} x_j = k \quad (2) \\ & x_j \in \{0,1\} \quad j=1,\ldots,n \end{cases}$$

*Corresponding author
[1]Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, (UMR 7030), F-93430 Villetaneuse, France.
E-mails: {lucas.letocart, gerard.plateau}@lipn.univ-paris13.fr
[2]GDF SUEZ, 2, place Samuel de Champlain, F-92930 Paris La Défense cedex, France.
E-mail: marie-christine.plateau@gdfsuez.com

where n denotes the number of items, and all the data, k (number of items to be placed in the knapsack), a_j (weight of item j), c_{ij} (profit associated with the selection of items i and j) and b (capacity of the knapsack) are nonnegative integers. Without loss of generality, matrix $C = (c_{ij})$ is assumed to be symmetric.

Moreover, we assume that $\max_{j=1,\ldots,n} a_j \leq b < \sum_{j=1}^{n} a_j$ in order to avoid either trivial solutions or variable fixing via constraint (1). Let us denote by $k_{\max}$ the largest number of items which could be filled in the knapsack, that is the largest number of the smallest a_j whose sum does not exceed b. We assume that $k \in \{2, \ldots, k_{\max}\}$, where $k_{\max}$ can be found in $O(n)$ time [3, 20]. If not, either the value of the problem is equal to $\max_{i=1,\ldots,n} c_{ii}$ (for $k = 1$), or the domain of $(E - kQKP)$ is empty (for $k > k_{\max}$).

Let us also note that on the one hand, by dropping constraint (1), $(E-kQKP)$ is a k-cluster problem [6], and that on the other hand, by dropping constraint (2), it becomes a classical quadratic knapsack problem [32]. We can therefore conclude that problem $(E - kQKP)$ is NP-hard, as it includes two classical NP-hard subproblems.

Numerous approaches have been proposed for solving general 0-1 quadratic problems (QP). Many are devoted to reformulations before searching an optimal solution of (QP), including 0-1 linear reformulations [1, 36] and 0-1 convex quadratic reformulations [8, 9]. Various heuristic and exact methods have been designed for (QP), including algebraic, dynamic programming, cutting plane, penalty and enumerative methods as well as metaheuristics, using different types of relaxations, such as roof duality, Lagrangian relaxation and decomposition, semidefinite programming, convex quadratic relaxation or convex hull relaxation (see, e.g., [12, 16, 25, 26, 30, 37]).

To the best of our knowledge, however, problem $(E - kQKP)$ has not been studied before. It is an extension of the 0-1 exact k-item linear knapsack problem [13, 19, 28] which may be considered as a subproblem of the 0-1 collapsing knapsack problem (see, e.g., [34]). Its applications cover those found in previous references for k-cluster or classical quadratic knapsack problems (*e.g.*, task assignment problems in a client-server architecture with limited memory), but also multivariate linear regression and portfolio selection. Specific heuristic and exact methods including branch-and-bound and branch-and-cut with surrogate relaxations have been designed for these applications (see, e.g., [4, 5, 10, 31, 35]).

Section 2 shows that this NP-hard problem can be considered as unsolvable to optimality for sizes exceeding 40 variables by using a state-of-the-art software (e.g., CPLEX 12.1). More specifically, it cannot be guaranteed that instances of equal or larger sizes can be solved exactly within an hour (Section 2.2). Even using a quadratic convex reformulation [8, 9] as a preprocessing phase, CPLEX cannot solve to optimality instances with sizes greater than 90 variables (Section 2.3).

After presenting a basic primal heuristic (Section 3), we present efficient dual heuristics that produce both good lower and upper bounds on the value of problem $(E - kQKP)$ in reasonable time. For this purpose, we propose to explore the potential of different types of duals of $(E - kQKP)$ based on either Lagrangian relaxations, or a positive semidefinite relaxation or a surrogate relaxation (this work extends our previous works on this subject [29]).

Testing different types of Lagrangian relaxations and Lagrangian decompositions produces poor computational results, namely, weak upper bounds and large computation times. This lead us to exclude the construction of effective bounds in this way.

Section 4 describes our first effective and fast dual heuristic. It is based on the quadratic convex reformulation [8, 9] of Section 2.3 associated with the solution of a semidefinite relaxation. More specifically, each step of this iterative procedure consists in fixing variables to zero in a solution produced by a semidefinite relaxation, before using our fast primal heuristic over the reduced problem.

Section 5 details another efficient dual heuristic. It is based on a surrogate relaxation after relaxing constraint (2) as an inequality constraint, and exploits the tree searches used at each step of the solution of the surrogate dual.

Finally, in Section 6 we propose a fast and efficient hybrid heuristic method that combines our previous three heuristics. It integrates the primal heuristic and the surrogate dual heuristic within the semidefinite programming reduction phase.

A number of computational results obtained for randomly generated instances with up to 200 variables is presented throughout the paper (see Section 2.1 for details about the computational environment). They validate the relevance of the bounds produced by our hybrid heuristic method, which yields known optima (and prove optimality) in 90% (resp. 76%) within 100 seconds on the average.

2 PRACTICAL DIFFICULTY OF THE PROBLEM

This section proposes to detail computational experiments to exhibit the practical difficulty of the NP-hard problem ($E - kQKP$). For this purpose, we study the experimental behaviour of the state-of-the-art software CPLEX 12.1 for solving exactly the benchmark instances described in Section 2.1. We conduct two series of experiments by using CPLEX 12.1 with default settings: the first one, on the original models, that is in particular without any previous reformulation of the problem (Section 2.2); the second one, on the modified models produced by the quadratic convex reformulation method QCR of the problem [8, 9] (Section 2.3). They show that CPLEX either without or with this previous reformulation, is not able to solve to optimality instances with more than 90 variables in a reasonable time.

2.1 Computational environment

All the experiments have been carried out on an Intel Xeon bi-processor dual core 3 GHz with 2 GB of RAM (using 4 cores for Sections 2.2 and 2.3, and only one core for all the other Sections).

For the benchmark, we always present average values over 10 instances. For $n \in \{10, 20, 30, \ldots, 200\}$, we fix $k \in [1, n/4]$, $b \in [50, 30k]$, and a_j, $c_{ij} \in [1, 100]$. These data intervals have been empirically chosen with the aim to get hard instances.

We always use the standard solver CPLEX 12.1 with default settings [18]. To solve semidefinite programs, we choose the software CSDP, integrated into COIN-OR [17], which applies the interior point method developed by Borchers [11].

In all the tables:

- δ represents the density of the quadratic matrix,
- the CPU time values are given in seconds,
- the duality *gap* for each *upper bound* proposed in this paper is defined as $\frac{upper\ bound - opt}{opt} \times 100$, where *opt* is the best known lower bound for the value of the problem, that is either the optimal value, or the best lower bound found by one of our heuristics,
- in the same way, the duality *gap* for each *lower bound* proposed in this paper is defined as $\frac{opt - lower\ bound}{opt} \times 100$, where *opt* is the best known upper bound for the value of the problem, that is either the optimal value, or the best upper bound computed by one of our dual algorithms or by CPLEX 12.1.

Note that to compute these gaps, some optimal values and some best upper bounds were obtained using CPLEX on a supercomputer (40 cores and 500 GB of RAM).

2.2 Numerical experiments with CPLEX 12.1

As the default preprocessing of CPLEX is able to convexify a non convex 0-1 problem, it is possible to run this state-of-the-art software without any previous reformulation of the problem. Results of Tables 1 and 2 highlight how huge the duality gaps are, i.e., the gaps between the optimal continuous and integer values. For smaller densities (Table 1), all instances can be solved within one hour, except for $\delta = 50\%$, where one instance cannot be solved to optimality. For larger densities (Table 2), for 40 and 50 variables, one or two instances cannot be solved exactly (see the numbers in brackets).

Table 1 – Exact solutions with CPLEX 12.1 for smaller densities.

	$\delta = 25$ %		$\delta = 50$ %	
n	Gap %	CPU time (s)	Gap %	CPU time (s)
10	174.24	0.01	251.67	0.01
20	154.87	0.02	316.22	0.07
30	159.93	0.27	293.30	0.77
40	135.56	2.36	295.09	25.68
50	157.28	245.48	214.56	>3600 (1)

2.3 A quadratic convex reformulation of the problem

Recent work by Billionnet, Elloumi & M-C. Plateau [9] and Billionnet, Elloumi & Lambert [8] highlights the great interest of an adequate convexification of the problem. Indeed, the perfor-

Table 2 – Exact solutions with CPLEX 12.1 for larger densities.

	$\delta = 75\ \%$		$\delta = 100\ \%$	
n	Gap %	CPU time (s)	Gap %	CPU time (s)
10	255.52	0.02	321.09	0.02
20	311.45	0.23	353.79	0.58
30	306.56	9.01	725.26	21.36
40	364.78	>3600 (1)	414.04	>3600 (1)
50	752.33	>3600 (1)	726.20	>3600 (2)

mance of a state-of-the-art solver is greatly improved when it is applied after their quadratic convex reformulation called QCR, i.e., these authors observed a drastic improvement in bound quality as well as in efficiency. This is the case for the instances of our benchmark (see the results of Section 2.3.2). Section 2.3.1 details before the application of the QCR method for our problem.

2.3.1 The QCR method

It consists of reformulating a nonconvex 0-1 quadratic maximization problem into an equivalent 0-1 program with a concave quadratic function. Then, the reformulated problem can be efficiently solved by a classical branch-and-bound algorithm based on continuous relaxation. The fundamental objective of this convexification method is therefore to construct a model whose continuous relaxation is as tight as possible.

In summary, QCR is a two-phase method, whose main phase is the preprocessing one:

- **Phase 1 –** Replace the objective function $f(x)$ by a concave quadratic objective function $f_{\alpha,u}(x)$ depending on two parameters α and u both in $\mathbb{R}^n$ to get an equivalent convex 0-1 program (see below).
- **Phase 2 –** Apply a standard 0-1 convex quadratic solver to this new problem.

To be more specific, the first phase consists of perturbing the objective function by subtracting two functions, equal to zero on the feasible set and depending on two parameters.

Billionnet, Elloumi & M-C. Plateau [9] replace $f(x)$ by:

$$f_{u,\alpha}(x) = f(x) - \sum_{i=1}^{n} u_i \left(x_i^2 - x_i\right) - \sum_{i=1}^{n} \alpha_i x_i \left(\sum_{j=1}^{n} x_j - k\right) \qquad \text{with } u \in \mathbb{R}^n \text{ and } \alpha \in \mathbb{R}^n .$$

Following the results of Faye & Roupin [21], Billionnet, Elloumi & Lambert [8] propose to replace $f(x)$ by:

$$f_{u,v}(x) = f(x) - \sum_{i=1}^{n} u_i \left(x_i^2 - x_i\right) - v \left(\sum_{j=1}^{n} x_j - k\right)^2 \qquad \text{with } u \in \mathbb{R}^n \text{ and } v \in \mathbb{R} .$$

The use of either of these new functions leads to identical bounds at the end of phase 1 of the QCR method. The use of the second, however, consumes the least amount of computation time. We propose thus to exploit $f_{u,v}(x)$ for our experiments.

The aim is thus to determine the best values of the parameters $u^* \in \mathbb{R}^n$ and $v^* \in \mathbb{R}$ so that the new function $f_{u^*,v^*}(x)$ is concave and the upper bound obtained by its continuous relaxation is minimal. They are the values that optimize the following problem:

$$(P) \quad \min_{u\in\mathbb{R}^n,\ v\in\mathbb{R}} \quad \max_{x\in[0,1]^n\ :\ \sum_{j=1}^n a_j x_j \le b,\quad \sum_{j=1}^n x_j = k} \quad f_{u,v}(x)$$

To obtain these optimal parameters, following Billionnet et al. [8], we solve the following semi-definite relaxation of $(E-kQKP)$:

$$(E\text{-}kQKP_{SDP}) \begin{cases} \max & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} X_{ij} & & \\ \text{s.t.} & X_{ii} = x_i & i = 1,\dots,n & (3) \\ & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij} - 2k\sum_{j=1}^{n} x_j = -k^2 & & (4) \\ & \displaystyle\sum_{j=1}^{n} a_j x_j \le b & & \\ & \displaystyle\sum_{j=1}^{n} x_j = k & & \\ & \begin{pmatrix} 1 & x^t \\ x & X \end{pmatrix} \succeq 0 & & \\ & x \in \mathbb{R}^n,\ X \in \mathbb{R}^{n^2} & & \end{cases}$$

where constraint (4) has been introduced to tighten the optimal value of the problem. It consists of replacing each product $x_i x_j$ by a variable X_{ij} in $\left(\sum_{j=1}^n x_j - k\right)^2 = 0$ (i.e., the square of constraint (2)).

The optimal values u_i^* $(i = 1,\dots,n)$ (resp. v^*) of problem (P) are simply given by the optimal values of the dual variables of $(E-kQKP_{SDP})$ associated with constraints (3) (resp. (4)).

2.3.2 Numerical experiments

As it is shown in [8, 9], the general QCR method produces better upper bounds and so better CPU times (including times taken to solve SDP) for different classes of 0-1 quadratic programming problems. Applying the QCR reformulation before solving our instances with CPLEX 12.1, it is now possible to solve efficiently instances up to 90 items, except for one instance with $\delta = 75\%$ (see Tables 3 and 4, where CPU times include the QCR reformulation effort). For larger sizes, up to 5 instances per class cannot be solved to optimality (see the numbers into brackets). Thus, in spite of the major improvement realized by using the QCR method (see for n equal to 50, the

large gap between results reached by CPLEX alone and CPLEX via QCR), all these experiments reinforce our purpose to elaborate dual heuristics which are able to produce within reasonable time both good lower and upper bounds for the value of problem $(E - kQKP)$.

Table 3 – Exact solutions with CPLEX 12.1 via QCR for smaller densities.

	$\delta = 25\ \%$		$\delta = 50\ \%$	
n	Gap %	CPU time (s)	Gap %	CPU time (s)
50	41.22	0.69	36.38	0.87
60	149.24	0.58	21.65	1.65
70	46.84	4.08	80.44	11.57
80	42.06	17.38	64.10	40.41
90	129.34	81.72	92.22	145.19
100	82.78	183.17	21.83	>3600 (1)
110	84.61	>3600 (2)	20.49	>3600 (3)

Table 4 – Exact solutions with CPLEX 12.1 via QCR for larger densities.

	$\delta = 75\ \%$		$\delta = 100\ \%$	
n	Gap %	CPU time (s)	Gap %	CPU time (s)
50	114.26	0.52	69.45	1.66
60	150.84	3.91	69.06	3.90
70	63.77	22.96	66.18	31.18
80	92.31	101.41	49.19	240.64
90	42.12	>3600 (1)	29.93	376.36
100	27.22	>3600 (4)	106.57	>3600 (2)
110	148.16	>3600 (3)	24.24	>3600 (5)

3 A PRIMAL HEURISTIC

This section presents a first lower bound for the value of problem $(E - kQKP)$. This basic primal heuristic, denoted by H_{pri}, is derived from a classical heuristic devoted to the quadratic knapsack problem [7]. As this straightforward approach exclusively provides a lower bound and consumes negligible computation times, it should be considered as a basic tool for our dual heuristics (see next Sections).

3.1 The heuristic H_{pri}

The primal heuristic method proposed below, denoted by H_{pri}, is an adaptation of a well-known heuristic developed by Billionnet & Calmels [7] for the classical quadratic knapsack problem (QKP). Thus, our approach links together a destructive phase and a constructive phase, the latter including a greedy algorithm and a local search. These two phases are summarized below (see Fig. 1 for more details):

- **Destructive phase –** A greedy algorithm derived from that of Chaillou, Hansen & Mahieu [15], is performed to produce a solution $\underline{x}$ starting from the vector $e = 1^n$ of $\mathbb{R}^n$. This destructive procedure is simply adapted to provide a solution satisfying the capacity constraint and with a number of items is less than or equal to k (i.e., $a\underline{x} \leq b$ and $e\underline{x} \leq k$). This is achieved by removing iteratively item i with the smallest utility function of the type

$$\sum_j \frac{c_{ij}}{a_i}.$$

- **Constructive phase –** This part performs fill up and exchange procedures. Derived from Gallo, Hammer & Simeone's method [23], its objective is two-fold: first, to get from $\underline{x}$ a feasible solution of $(E - kQKP)$, that is a solution $\underline{x}'$ satisfying the constraint capacity whose number of items placed in the knapsack is now exactly equal to k (i.e., $a\underline{x}' \leq b$ and $e\underline{x}' = k$); second, to find finally a solution $\underline{x}^{pri}$ which improves the quality of solution $\underline{x}'$.

3.2 Numerical experiments

As a general remark, we note that our primal heuristic H_{pri} is very fast, since for all instances it took less than 0.04 seconds. The duality gaps are rather small, for all instance sizes (see Fig. 2 and Table 5). The best gap values are obtained for the larger densities. Nevertheless, these gaps are markedly greater than those obtained for the classical quadratic knapsack problem, which are generally less than 1%.

4 A SEMIDEFINITE PROGRAMMING HEURISTIC

Our first effective and fast dual heuristic, denoted by H_{sdp}, is based on semidefinite programming (Section 4.1). Each step of this iterative procedure consists in fixing variables to zero in a solution produced by a semidefinite relaxation of the problem, before using our very fast primal heuristic over the reduced problem (Section 4.2). The computational experiments highlight the gain obtained by this algorithmic tool which dominates heuristics H_{pri} for all instances (Section 4.3). H_{sdp} is very fast, since for all instances it tooks less than 4 seconds on the average.

4.1 The semidefinite programming relaxation

Let us recall [8] that from the semidefinite relaxation $(E - kQKP_{SDP})$ defined in Section 2.3.1, we get optimal multipliers $u^* \in \mathbb{R}^n$ and $v^* \in \mathbb{R}$ for the QCR method such that:

$$v(E - kQKP_{SDP}) = \max_{x \in [0,1]^n \,:\, ax \leq b, \quad ex = k} f_{u*,v*}(x)$$

We can therefore expect that problem $(E - kQKP_{SDP})$ produces good upper bounds for problem $(E - kQKP)$ and at the same time, produces a solution, denoted by x^{sdp}, which is not too far from a good solution of $(E - kQKP)$ (see [26, 27] for results concerning other types of 0-1 quadratic problems).

/*Initial phase: provides a trivial solution $\underline{x}^0$ such that $a\underline{x}^0 \leq b$ and $e\underline{x}^0 = k$*/
$J \leftarrow arg\ k\ smallest\ (a_j)_{j\in\{1,\ldots,n\}}$
for j **from** 1 **to** n **do**
  if $j \in J$ **then** $\underline{x}^0_j \leftarrow 1$ **else** $\underline{x}^0_j \leftarrow 0$ **endif**
enddo
$\underline{x}^{pri} \leftarrow \underline{x}^0$; $v(H_{pri}) \leftarrow f(\underline{x}^0)$
/*Destructive phase: provides a solution $\underline{x}^1$ such that $a\underline{x}^1 \leq b$ and $e\underline{x}^1 \leq k$*/
$\underline{x}^1 \leftarrow e = \{1, \ldots, 1\}$
$U \leftarrow \{1, \ldots, n\}$
while $\sum_{j=1}^{n} a_j > b$ **or** $card(U) > k$ **do**
  $i^* \leftarrow arg\ min(\sum_{j\in U} c_{ij}/a_i)$.
  $\underline{x}^1_{i^*} \leftarrow 0$
  $U \leftarrow U - \{i^*\}$
endwhile
if $e\underline{x}^1 = k$ **and** $f(\underline{x}^1) > v(H_{pri})$ **then** $\underline{x}^{pri} \leftarrow \underline{x}^1$; $v(H_{pri}) \leftarrow f(\underline{x}^1)$ **endif**
/*Constructive phase: fill up and exchange procedures to improve $\underline{x}^1$ */
$\underline{x}^2 \leftarrow \underline{x}^1$
$bestgain \leftarrow 1$
while $bestgain$ **do**
  $bestgain \leftarrow 0$; $fillup \leftarrow false$; $exchange \leftarrow false$
  for all variable x_i such that $\underline{x}^2_i = 0$ **do**
    if $e\underline{x}^2 < k$ **and** $a\underline{x}^2 + a_i \leq b$ **then**
      $gain \leftarrow \delta f_i$ /*first derivative of function $f = c_i + \sum_{j=1}^{n} c_{ij}\underline{x}^2_j$*/
      if $gain > bestgain$ **then**
        $bestgain \leftarrow gain$; $i^{fill} \leftarrow i$, $fillup \leftarrow true$
      endif
    else /*$e\underline{x}^2 = k$*/
      for all variable x_j such that $\underline{x}^2_j = 1$ **do**
        if $a\underline{x}^2 - a_j + a_i \leq b$ **then**
          $gain \leftarrow \delta f_i - \delta f_j - c_{ij} - c_{ji}$ /*second derivative of function f*/
          if $gain > bestgain$ **then**
            $bestgain \leftarrow gain$; $(i^{ex}, j^{ex}) \leftarrow (i, j)$; $exchange \leftarrow true$
          endif
        endif
      endfor
    endif
  endfor
  if $exchange$ **then**
    $\underline{x}^2_{iex} \leftarrow 1$; $\underline{x}^2_{jex} \leftarrow 0$
  else
    if $fillup$ **then** $\underline{x}^2_{i\,fill} \leftarrow 1$ **endif**
  endif
endwhile
if $e\underline{x}^2 = k$ **and** $f(\underline{x}^2) > v(H_{pri})$ **then** $\underline{x}^{pri} \leftarrow \underline{x}^2$; $v(H_{pri}) \leftarrow f(\underline{x}^2)$ **endif**

Figure 1 – The primal heuristic H_{pri}.

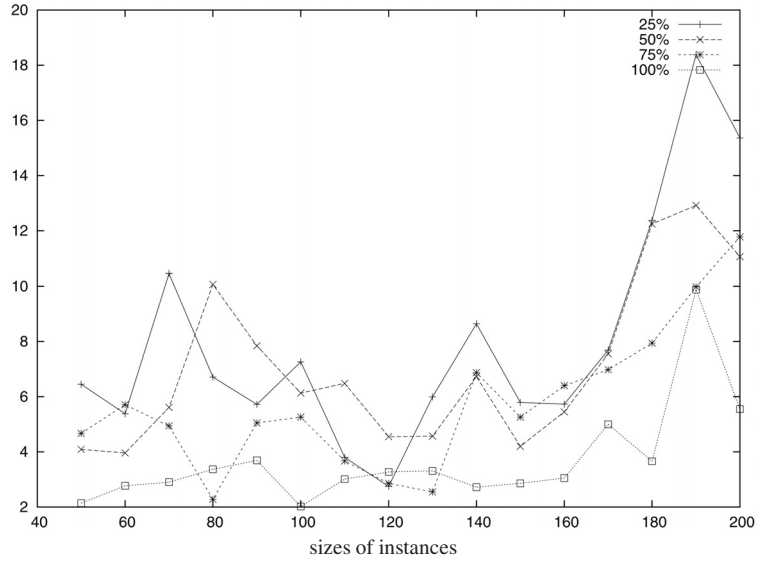


Figure 2 – Heuristic H_{pri}: duality gaps for each density of matrix C.

Table 5 – H_{sdp} vs. H_{pri}: Gap quality.

δ	Gaps %	
	H_{pri}	H_{sdp}
25%	8.08	1.69
50%	7.09	1.67
75%	5.76	1.64
100%	3.70	1.04
Global results	6.16	1.51

Numerical experiments. Solving the semidefinite relaxation ($E - kQKP_{SDP}$) is rather fast: CPU times increase with the sizes of the instances without exceeding five seconds. Observing gap quality for all densities of matrix C in Figure 4 (a), the behavior of this relaxation is clearly unstable. Even if the average gap can be large (up to 150% for some instances), we still propose to use this semidefinite program for designing our heuristic H_{sdp}.

4.2 The heuristic H_{sdp}

This heuristic method solves first the semidefinite relaxation ($E - kQKP_{SDP}$) defined in Sections 2.3.1 and 4.1. Then an iterative procedure based on the associated solution x^{sdp} is initiated. More specifically, each step of this procedure consists in fixing more and more variables to zero in x^{sdp}, before using our very fast primal heuristic H_{pri} over the reduced problem. An attempt to update the solution produced by H_{pri} is then applied by performing the fill-up and exchange procedure of Figure 1 on the global problem, i.e., with the n variables (see Fig. 3 for more details).

$x^{sdp} \leftarrow$ solution of the semidefinite relaxation $(E-kQKP_{SDP})$ of the problem
$\epsilon \leftarrow 0$
$n_{reduce} \leftarrow n$
$\underline{x}^{sdp} \leftarrow 0$
while $n_{reduce} \geq 1$ **do**
  repeat
    $\epsilon \leftarrow \epsilon + 0.01$
  until at least one variable is less than ϵ in x^{sdp}
  Eliminate variables with values less than ϵ in x^{sdp}
  Update n_{reduce}
  Perform heuristic H_{pri} over the *reduced problem*
  Update $\underline{x}^{pri}$ by performing the fill-up and exchange procedure on the *global problem*
  if $f(\underline{x}^{pri}) > f(\underline{x}^{sdp})$
  then
    $\underline{x}^{sdp} \leftarrow \underline{x}^{pri}$
  endif
endwhile
$v(H_{sdp}) \leftarrow f(\underline{x}^{sdp})$

Figure 3 – The semidefinite programming heuristic H_{sdp}.

4.3 Numerical experiments

The performance of heuristic H_{sdp} is analyzed through the numerical results laid out in Tables 5, 6 and 7. Table 5 shows a great improvement regarding the quality of the duality gap. It is about three times better than that obtained by H_{pri}. Tables 6 and 7 complete this information by splitting the results into two categories: small sizes ($n \leq 100$) and larger sizes ($n \geq 110$). They reveal that CPU times remain rather low, since heuristic H_{sdp} tooks less than 4 seconds on the average.

5 A SURROGATE DUAL HEURISTIC FOR $(E-kQKP)$

Let us consider another effective dual heuristic, denoted by H_{sur}, based on a surrogate relaxation. Eventually, this algorithmic tool will prove to be an efficient and robust method: it proves optimality for 46% of the 640 instances, within one minute on the average.

5.1 A surrogate dual algorithm

We actually use a double relaxation. We first consider the following problem for which the equality cardinality constraint (2) is transformed into an inequality constraint (2′):

$$(kQKP)\begin{cases} \max & f(x) & \\ \text{s.t.} & ax \leq b & (1) \\ & ex \leq k & (2') \\ & x \in \{0,1\}^n & \end{cases}$$

Second, we consider a surrogate relaxation of this 0-1 quadratic bi-knapsack problem, by combining the two constraints (1) and (2′). To solve the associated surrogate dual problem, we use the scheme of the algorithm $SADE^2$ developed by Fréville & G. Plateau [22] for the surrogate dual of *linear* 0-1 bi-dimensional knapsack problems. $SADE^2$ is a one-dimensional dichotomic search over the compact interval [0,1], which revisits Glover's method [24] for guaranteeing optimality in a finite number of iterations.

In short, given a multiplier $\mu \in [0, 1]$, algorithm $SADE^2$ determines first an order on the two constraints (1) and (2'), so that the surrogate relaxation can be written as:

$$(S(\mu)) \begin{cases} \max & f(x) \\ \text{s.t.} & (A_1 + \mu A_2)x \leq (b_1 + \mu b_2) \\ & x \in \{0, 1\}^n \end{cases}$$

with either

$$(A_1, b_1) = (a, b) \quad \text{and} \quad (A_2, b_2) = (e, k)$$

or

$$(A_1, b_1) = (e, k) \quad \text{and} \quad (A_2, b_2) = (a, b)$$

For a given μ, let $x^*(\mu)$ be an optimal solution of problem $(S(\mu))$, if $x^*(\mu)$ satisfies simultaneously the two constraints, it is obviously an optimal solution of problem $(kQKP)$. If, in addition, $ax^*(\mu) = k$, the problem $(E - kQKP)$ is solved. Otherwise, procedure $SADE^2$ performs its dichotomic search by using the fact that $x^*(\mu)$ satisfies the first (resp. second) constraint and violates the second (resp. first) one, and by exploiting the ratio $(b_1 - A_1 x^*(\mu))/(A_2 x^*(\mu) - b_2)$ to update the interval bounds (see [22]).

As the theoretical results for this method depend only on properties of the bi-constraint system, its adaptation for the quadratic case is straightforward. In addition, at each iteration of the procedure, we propose to exploit the approach of Caprara, Pisinger & Toth [14] for solving the associated quadratic 0-1 knapsack instance (this requires to adapt their code for accepting a constraint with real data). The first step of Caprara et al.'s method consists of linearizing $(S(\mu))$ as follows:

$$\begin{cases} \max & f(x, y) = \sum_{j \in N} c_{jj} x_j + \sum_{j \in N} \sum_{i \in N \setminus \{j\}} c_{ij} y_{ij} & & \\ \text{s.t.} & \sum_{j \in N} (a_{1j} + \mu a_{2j}) x_j \leq b_1 + \mu b_2 & & \\ & \sum_{i \in N \setminus \{j\}} (a_{1i} + \mu a_{2i}) y_{ij} \leq (b_1 + \mu b_2 - (a_{1j} + \mu a_{2j})) x_j \quad \forall j \in N & & (5) \\ & 0 \leq y_{ij} \leq x_j \leq 1 & \forall (i, j) \in N^2, i \neq j & \\ & y_{ij} = y_{ji} & \forall (i, j) \in N^2, i < j & (6) \\ & x_i, y_{ij} \in \{0, 1\} & \forall (i, j) \in N^2, i \neq j & \end{cases}$$

where N denotes the set of $\{1, \ldots, n\}$. For this 0-1 linear problem, denoted by $(LP(\mu))$, 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 (5) result from 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. Caprara et al. solve this problem by a branch-and-bound method based on a depth-first search strategy with a static order of the variables for branching. It uses a preprocessing phase which reduces the size of the problem through variable fixing. Its effectiveness is obtained by algorithmic choices and data structures that favor simplicity and incrementality (as for the algorithm designed for the linear 0-1 knapsack problem in [33]). In particular, their evaluation of nodes is based on a Lagrangian relaxation of $(LP(\mu))$ which dualizes the symmetric constraints (6). The authors highlight a beautiful decomposition result which shows that solving this relaxation is equivalent to solve $n + 1$ linear 0-1 knapsack problems. Furthermore, for controling time complexity, the authors decide to relax the integer conditions on variables. This leads to an evaluation which can be produced in $0(n^2)$ time since each continuous linear knapsack problem can be solved in $0(n)$ time [20].

From a practical point of view, in order to control CPU times (i.e., to speed up achieving good upper bound for $v(kQKP)$ and thus for $v(E - kQKP)$), we propose to search an approximate value of the surrogate dual (S) of $(kQKP)$ by means of the following two algorithmic adaptations:

– *approximation 1 (solving problem* $(S(\mu))$*)*:

We add a CPU time limit (one minute, in our case) in the branch-and-bound search of Caprara et al.'s method. Obviously, this stopping criterion is effective when the Lagrangian relaxation evaluation is inefficient. In practice, this occurs for values of μ close to zero, or for the greater instance sizes (around 200 variables). In this situation, we have retained first the upper bound $v^{up}(S(\mu))$ on the value of $(S(\mu))$, equal to the maximal evaluation over the pending nodes of the search tree, and also the associated solution $x^{up}(\mu)$ of $(S(\mu))$.

– *approximation 2 (in the dichotomic search)*:

The performance of Caprara et al.'s method is poor for a problem like $(S(0))$, whose constraint coefficients are all identical (equal to one here). When no solution is found by Caprara et al.'s method within the CPU time limit, we propose to increase multiplier μ step by step from zero (i.e., we start with value 0.01, then double that value each time). This process stops as soon as a multiplier μ^t is found such that problem $(S(\mu^t))$ can be solved to optimality within the CPU time limit.

While *approximation 2* implies that the set M of multipliers generated by our approximate dual algorithm may miss the optimal multiplier μ^* for (S), with *approximation 1*, we can conclude that our dual algorithm finds an upper bound $\overline{v}(S)$ of $v(S)$ defined as follows:

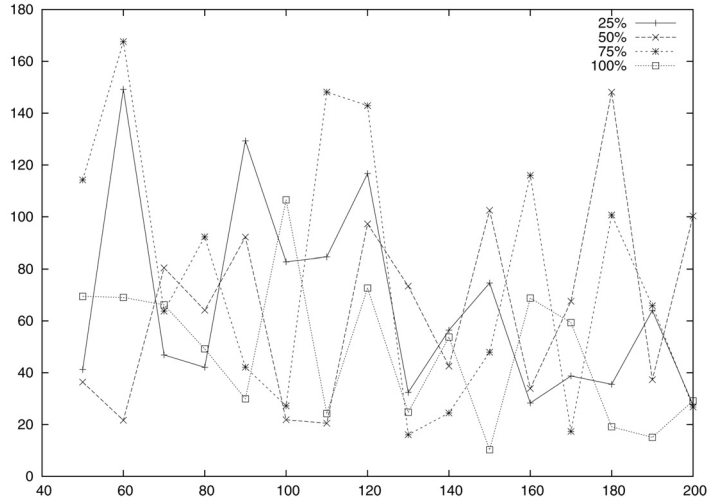
$$\overline{v}(S) = \min_{\mu \in M} \overline{v}(S(\mu))$$

where $\overline{v}(S(\mu))$ is equal to $v^{up}(S(\mu))$ if *approximation 1* is being used, or $v(S(\mu))$ otherwise. We can then state the following result:

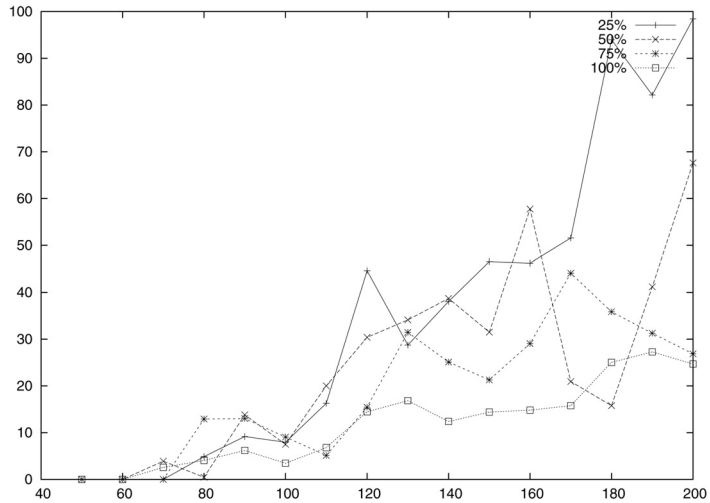
Proposition.

$$v(E-kQKP) \leq v(kQKP) \leq v(S) = \min_{\mu\in[0,1]} v(S(\mu)) \leq \overline{v}(S) = \min_{\mu\in M} \overline{v}(S(\mu))$$

Finally, in the algorithm of Figure 5, for any μ, *solving* $(S(\mu))$ may be *solve* $(S(\mu))$ *approximately* when *approximation 1* occurs. In this case, we exploit the solution $x^{up}(\mu)$ (see above) that may differ from $x^*(\mu)$. All along our approximate dual method, we propose to denote this solution by $\overline{x}(\mu)$, which is equal to $x^{up}(\mu)$ if *approximation 1* is being used, or $x^*(\mu)$ otherwise.



(a) Convexification via QCR



(b) Surrogate dual

Figure 4 – Duality gaps for each density of matrix C.

Numerical experiments. As for the linear case, our approximate surrogate dual value is reached in a finite number of iterations (at most 10 iterations here). From the results obtained with this dichotomic procedure, two global remarks may be mentioned: first, CPU times do not exceed seven minutes on the average and above all the quality of upper bounds is very good (see Fig. 4(b)). Second, more than 75% of gap values are improved in comparison with those produced by the semidefinite programming relaxation (see Section 4).

5.2 The surrogate dual heuristic H_{sur}

Classically, our surrogate dual heuristic, denoted by H_{sur}, keeps the best feasible solution of $(E - kQKP)$, denoted by $\underline{x}^{sur}$, produced by the dichotomic procedure described in Figure 5. It computes both a lower bound $f(\underline{x}^{sur})$ and an upper bound $\overline{v}(S)$ (see Section 5.1) of $v(E - kQKP)$.

More specifically, at each iteration of our dichotomic procedure, that is for each multiplier μ in the set M defined above, we retain the best feasible solution – denoted by $\underline{x}(\mu)$ – produced by the branch-and-bound depth first search, which satisfies the cardinality constraint exactly (i.e., $a\underline{x}(\mu) \leq b$ and $e\underline{x}(\mu) = k$).

As the procedure starts by using the very fast heuristic H_{pri}, the solution $\underline{x}^{sur}$ produced by our heuristic H_{sur} is such that:

$$f\left(\underline{x}^{sur}\right) = \max\left\{f(\underline{x}^{pri}), \max_{\mu \in M} f(\underline{x}(\mu))\right\}$$

Using an analysis of the experimental results of the previous section, the single adjustment of our procedure consists in reducing to half a minute the CPU time threshold for solving $(S(\mu))$ approximately (its solution is denoted by $\overline{x}(\mu)$ (see Section 5.1)). This allows first, to speed up the search with a negligible loss in gap quality and second, to get a dual heuristic which is both effective and robust. Finally, it should be noted that problem $(E - kQKP)$ is solved exactly when a problem $S(\mu)$ is solved exactly for a given μ in M and when its optimal solution $\overline{x}(\mu)$ is primal feasible for $(E - kQKP)$ (see Proposition of Section 5.1).

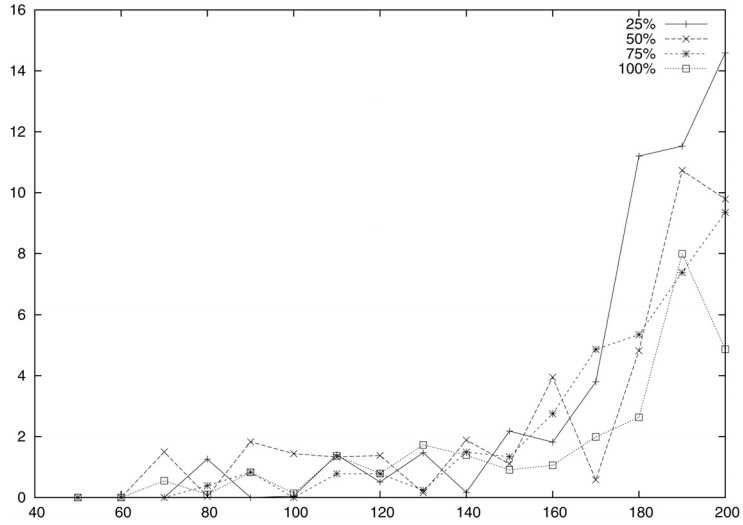
Numerical experiments. The numerical results of Figure 6 (associated with those of Table 6 and Table 7) show the high quality of the lower bounds provided by our surrogate dual heuristic H_{sur}. It tends to dominate heuristic H_{sdp} on the instances of small sizes ($n \leq 100$), while the reverse is true for the larger sizes ($n \geq 110$). It should be noted that 46% of the instances are in fact exactly solved by our surrogate dual heuristic. Although CPU times of H_{sur} increase with the size of the instances, they never exceed two minutes. Moreover, for each density, the mean time is around one minute.

6 A HYBRID DUAL HEURISTIC

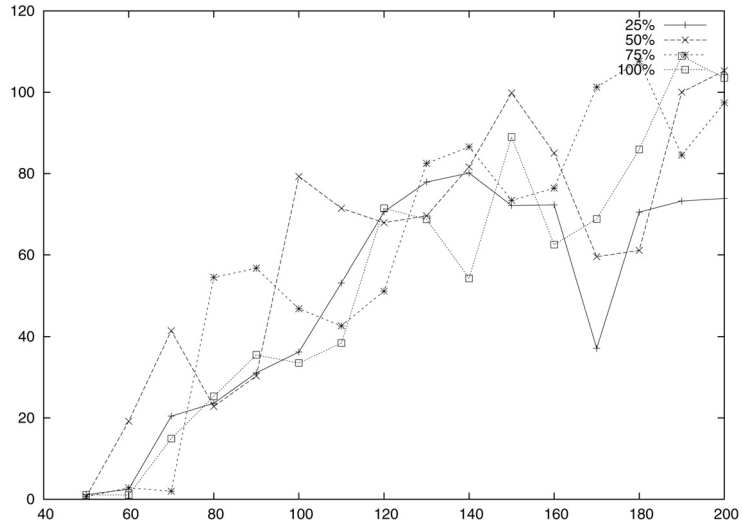
Our fast and efficient hybrid heuristic method, denoted by H_{hybrid}, combines the heuristics described in Sections 3, 4 and 5. It integrates the primal heuristic and the surrogate dual heuristic

Perform heuristic H_{pri} ; $\underline{x}^{sur} \leftarrow \underline{x}^{pri}$
/*Trust interval for the dichotomic procedure*/
$(A_1, b_1) \leftarrow (a, b)$; $(A_2, b_2) \leftarrow (e, k)$; $end \leftarrow false$
Solving of $(S(1)) : max\ f(x)\ \ s.t.\ \ (a+e)x \leq b+k;\quad x \in \{0,1\}^n$ /*associated solution $\overline{x}(1)$*/
if $f(\underline{x}(1) > f(\underline{x}^{sur})$ **then** $\underline{x}^{sur} \leftarrow \underline{x}(1)$ **endif**
if $\overline{x}(1)$ is primal feasible
then $v(kQKP) \leftarrow f(\overline{x}(1))$ /*$\overline{v}(S) = v(kQKP)$*/
if $e\overline{x}(1) = k$ **then** $\underline{x}^{sur} = \overline{x}(1)$ is optimal for $(E - kQKP)$; $end \leftarrow true$ **endif**
else $\alpha \leftarrow (b - a\overline{x}(1))/(e\overline{x}(1) - k)$
if $a\overline{x}(1) > b$
then
Solving of $(S(0)) : max\ f(x)\ \ s.t.\ \ ax \leq b;\quad x \in \{0,1\}^n$ /*associated solution $\overline{x}(0)$*/
if $f(\underline{x}(0) > f(\underline{x}^{sur})$ **then** $\underline{x}^{sur} \leftarrow \underline{x}(0)$ **endif**
if $e\overline{x}(0) \leq k$
then $v(kQKP) \leftarrow f(\overline{x}(0))$ /*$\overline{v}(S) = v(kQKP)$*/
if $e\overline{x}(0) = k$ **then** $\underline{x}^{sur} = \overline{x}(0)$ is optimal for $(E - kQKP)$; $end \leftarrow true$ **endif**
else $\alpha_l \leftarrow (b - a\overline{x}(0))/(e\overline{x}(0) - k)$; $\alpha_r \leftarrow \alpha$
endif
else /*$e\overline{x}(1) > k$*/
Solving of $(S(\infty)) : max\ f(x)\ \ s.t.\ \ ex \leq k;\quad x \in \{0,1\}^n$ /*associated solution $\overline{x}(\infty)$*/
if $f(\underline{x}(\infty) > f(\underline{x}^{sur})$ **then** $\underline{x}^{sur} \leftarrow \underline{x}(\infty)$ **endif**
if $a\overline{x}(\infty) \leq b$
then $v(kQKP) \leftarrow f(\overline{x}(\infty))$ /*$\overline{v}(S) = v(kQKP)$*/
if $e\overline{x}(\infty) = k$ **then** $\underline{x}^{sur} = \overline{x}(\infty)$ is optimal for $(E - kQKP)$; $end \leftarrow true$ **endif**
else $\alpha_l \leftarrow (e\overline{x}(0) - k)/(b - a\overline{x}(0))$; $\alpha_r \leftarrow 1/\alpha$
$(A_1, b_1) \leftarrow (e, k)$; $(A_2, b_2) \leftarrow (a, b)$ /*Permutation of the two constraints*/
endif
endif
endif
/*Extended dichotomy*/
$\mu_l \leftarrow 0$; $\mu_r \leftarrow 0$
while *not end* **do**
if $\alpha_l \geq \alpha_r$
then $\overline{v}(S) \leftarrow \min\{f(\overline{x}(\mu_l)), f(\overline{x}(\mu_r))\}$; $end \leftarrow true$
if $f(\underline{x}^{sur}) = \overline{v}(S)$ **then** $\underline{x}^{sur}$ is optimal for $(E - kQKP)$ **endif**
else $\mu \leftarrow (\alpha_l + \alpha_r)/2$; *Solving of* $(S(\mu))$ /*associated solution $\overline{x}(\mu)$*/
if $f(\underline{x}(\mu) > f(\underline{x}^{sur})$ **then** $\underline{x}^{sur} \leftarrow \underline{x}(\mu)$ **endif**
if $\overline{x}(\mu)$ is primal feasible
then $v(kQKP) \leftarrow f(\overline{x}(\mu))$ /*$\overline{v}(S) = v(kQKP)$*/
if $e\overline{x}(\mu) = k$ **then** $\underline{x}^{sur} = \overline{x}(\mu)$ is optimal for $(E - kQKP)$; $end \leftarrow true$ **endif**
else $\alpha \leftarrow (b_1 - A_1\overline{x}(\mu))/(A_2\overline{x}(\mu) - b_2)$
if $A_2\overline{x}(\mu) > b_2$ **then** $\alpha_l \leftarrow \alpha$; $\mu_l \leftarrow \mu$ **else** $\alpha_r \leftarrow \alpha$; $\mu_r \leftarrow \mu$ /*$A_1\overline{x}(\mu) > b_1$*/ **endif**
endif
endif
endwhile
$v(H_{sur}) \leftarrow f(\underline{x}^{sur})$

Figure 5 – The heuristic H_{sur}.



(a) Duality gaps



(b) CPU times (s)

Figure 6 – Heuristic H_{sur} (results for each density of matrix C).

within the semidefinite programming reduction phase. It proves to get best duality gaps in a reasonable time for all instances of our benchmark.

6.1 The heuristic H_{hybrid}

The aim of this heuristic (see Fig. 7) is to get better solutions than those provided by our heuristics H_{sdp} and H_{sur}. It starts by performing H_{sur} which includes heuristic H_{pri} (see Fig. 5). We then

apply heuristic H_{sdp} (see Fig. 3) in which heuristic H_{sur} is used for a second and last time. We are thefore sure to find a solution $\underline{x}^{hybrid}$ as good as that produced by both heuristics H_{pri}, H_{sdp} and H_{sur}. In addition, computation times remain reasonable since heuristic H_{sur} is used only two times. They are in addition to the reasonable time of H_{sdp}.

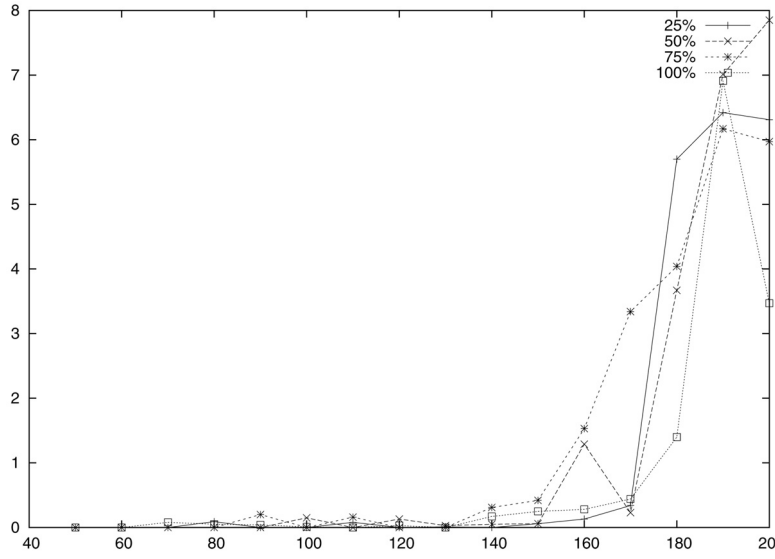
Perform heuristic H_{sur}
/*the value of $H_{sur} = f(\underline{x}^{sur})$ is at least equal to the value of H_{pri}=$f(\underline{x}^{pri}_{first})$ */
$\underline{x}^{hybrid} \leftarrow \underline{x}^{sur}$
$x^{sdp} \leftarrow$ solution of the semidefinite relaxation $(E - kQKP_{SDP})$ of the problem
$\epsilon \leftarrow 0$
$n_{reduce} \leftarrow n$
while $n_{reduce} \geq 1$ **do**
  repeat
    $\epsilon \leftarrow \epsilon + 0.01$
  untill at least one variable is less than ϵ in x^{sdp}
  Eliminate variables with values less than ϵ in x^{sdp}
  Update n_{reduce}
  Perform heuristic H_{pri} over the *reduced problem*
  if ($\underline{x}^{hybrid}$ and $\underline{x}^{pri}$ are not better than $\underline{x}^{pri}_{first}$) and $\epsilon = 0.05$
  then /*heuristic H_{sur} is performed for the second and last time*/
    Perform heuristic H_{sur} over the *reduced problem*
    if $f(\underline{x}^{sur}) > f(\underline{x}^{pri})$
    then
      $\underline{x}^{pri} \leftarrow \underline{x}^{sur}$
    endif
  endif
  Update $\underline{x}^{pri}$ by performing the fill-up and exchange procedure on the *global problem*
  if $f(\underline{x}^{pri}) > f(\underline{x}^{hybrid})$
  then
    $\underline{x}^{hybrid} \leftarrow \underline{x}^{pri}$
  endif
endwhile
$v(H_{hybrid}) \leftarrow f(\underline{x}^{hybrid})$

Figure 7 – The heuristic H_{hybrid}.

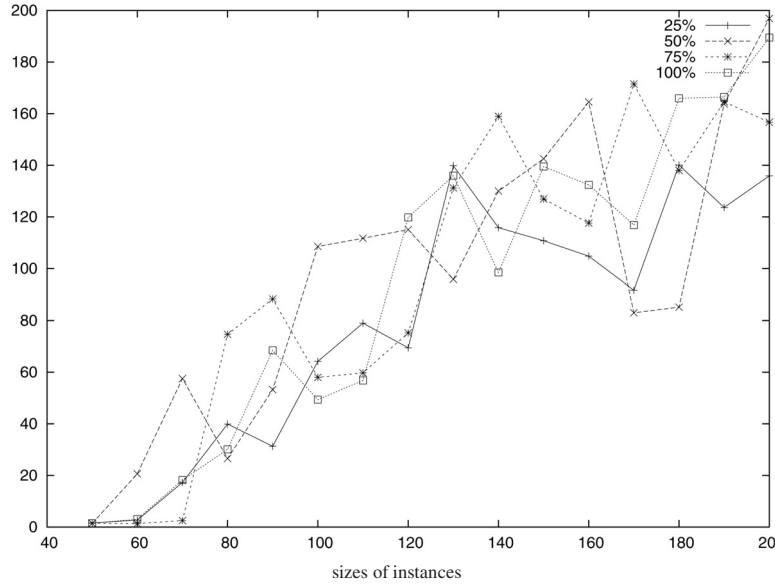
6.2 Numerical experiments

The computational experiments validate the relevance of the bounds produced by our hybrid heuristic method which combines quality (i.e., duality gap is about 1.20 % on the average) and efficiency (i.e., CPU time is about one minute and a half on the average).

Figure 8 with Tables 6, 7 and 8 summarize the performance of heuristic H_{hybrid}. Additional computational experiments have been realized by performing CPLEX 12.1, instance by instance, within the associated CPU time of heuristic H_{hybrid}. The results with this *limited time CPLEX*



(a) Duality gaps

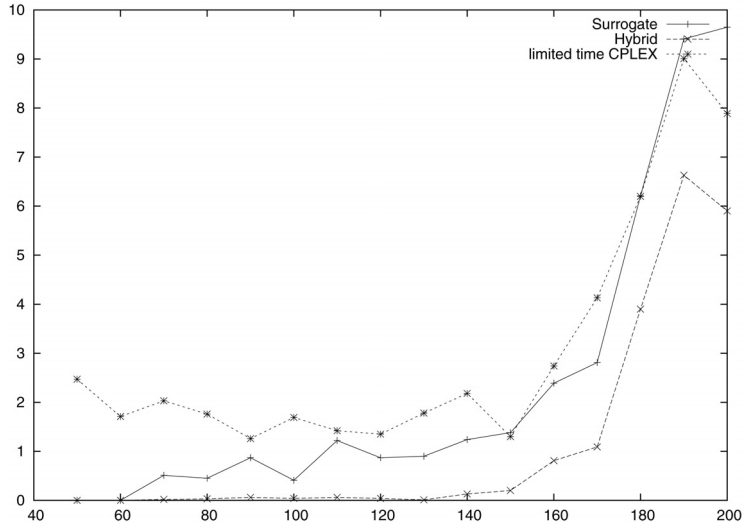


(b) CPU times (s)

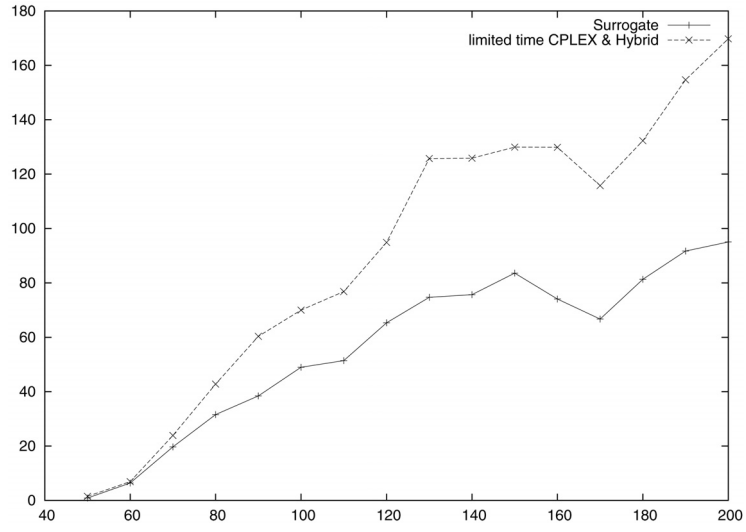
Figure 8 – Heuristic H_{hybrid} (results for each density of matrix C).

software, denoted by $LT-CPLEX$, are listed in Tables 6 and 7. Figure 9(a) (duality gaps) show clearly, instance size by instance size, the dominance of H_{hybrid} over H_{sur} and $LT-CPLEX$.

Table 8 counts the percentage of instances for which optimality is proved (i.e., lower and upper bounds coincide) or simply reached (550 of the 640 instances of our benchmark have a known optimal solution). Results in brackets relate to these 550 instances.



(a) Duality gaps (%)



(b) CPU time (s)

Figure 9 – H_{hybrid} vs. H_{sur} and $limited\ time\ CPLEX$.

For the set of 550 instances (resp. 640 instances), heuristic H_{hybrid} yields thus known optima in 90% (resp. 76%) of the cases and proves optimality in 63% (resp. 49%) of the cases, within less than 100 seconds on the average. These scores are even better for instances with sizes from 50 to 100 variables. For example, H_{hybrid} yields known optima in 96% of the cases (i.e., 232 of 240 instances).

Finally, Figure 9(b) shows that CPU time of H_{hybrid} increases with the size of the instances, but never exceed two minutes and a half.

Table 6 – H_{hybrid} vs. H_{sdp}, H_{sur} and $LT-CPLEX$: average values for $n = 50$ to 100.

	Heuristic H_{sdp}		Heuristic H_{sur}		Heuristic H_{hybrid}		$LT-CPLEX$
δ	Gap %	CPU (s)	Gap %	CPU (s)	Gap %	CPU (s)	Gap %
25 %	0.60	1.55	0.22	19.15	**0.01**	26.14	1.62
50 %	0.43	1.70	0.80	32.28	**0.03**	44.64	2.11
75 %	0.22	1.46	0.20	27.27	**0.03**	37.71	2.28
100 %	0.35	1.51	0.27	18.55	**0.03**	28.43	1.27
Global results	0.40	1.56	0.37	25.06	**0.03**	34.23	1.82

Table 7 – H_{hybrid} vs. H_{sdp}, H_{sur} and $LT-CPLEX$: average values for $n = 110$ to 200.

	Heuristic H_{sdp}		Heuristic H_{sur}		Heuristic H_{hybrid}		$LT-CPLEX$
δ	Gap %	CPU (s)	Gap %	CPU (s)	Gap %	CPU (s)	Gap %
25 %	2.42	6.91	4.95	68.10	**1.98**	111.11	3.99
50 %	2.41	6.12	3.57	80.16	**2.03**	128.87	4.41
75 %	2.51	6.14	3.43	80.36	**2.19**	130.05	4.15
100 %	1.45	6.66	2.47	75.17	**1.30**	132.19	2.65
Global	2.20	6.46	3.61	75.95	**1.88**	125.56	3.80

Table 8 – Heuristic H_{hybrid}: performance for all sizes.

			Optimality	
δ	Gap %	CPU (s)	Proved %	Reached %
25 %	1.25	79.25	47.95 (58.82)	80.14 (92.13)
50 %	1.28	97.28	51.33 (65.25)	77.33 (89.92)
75 %	1.38	95.43	47.02 (61.74)	72.85 (88.00)
100 %	0.82	93.28	51.32 (66.67)	74.34 (88.28)
Global	1.18	91.31	49.42 (63.11)	76.13 (89.59)

7 CONCLUSION

This paper proposes fast computations of lower and upper bounds for the 0-1 exact k-item quadratic knapsack problem ($E-kQKP$). Thanks to these bounds, our experiments highlight the possibility to solve now instances up to 200 variables whatever their density. Computational experiments point out the efficiency and the robustness of our hybrid heuristic method: it yields known optima in 90% of the cases and proves optimality in 76% of the cases, within 100 seconds in average. Further work concerns obviously the improvement of the lower bounds but we presume that we have to concentrate one's effort on the upper bounds for instances with large sizes. Thus, we plan to study new types of relaxations such that the convex hull relaxation (see [2]). A final perspective consists in embedding these high-quality bounds in a branch-and-bound scheme for the exact solving of ($E-kQKP$).

ACKNOWLEDGEMENTS

This work was partially conducted on the Magi cluster at Université Paris 13, and available at `http://www.univ-paris13.fr/calcul/wiki/`.

REFERENCES

[1] ADAMS WP, FORRESTER R & GLOVER F. 2004. Comparisons and enhance ment strategies for linearizing mixed 0-1 quadratic programs. *Discrete Optimization*, **1**(2): 99–120.

[2] AHLATŞOGLU A & GUIGNARD M. 2007. The convex hull relaxation for nonlinear programs with linear constraints. Technical report, OPIM Department, Wharton School (USA).

[3] BALAS E & ZEMEL E. 1980. An algorithm for large zero-one knapsack problems. *Operations Research*, **28**: 1130–1154.

[4] BERTSIMAS D & SHIODA R. 2009. Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, **43**: 1–22.

[5] BIENSTOCK D. 1996. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, **74**: 121–140.

[6] BILLIONNET A. 2005. Different formulations for solving the heaviest *k*-subgraph problem. *Information Systems and Operational Research*, **43**(3): 171–186.

[7] BILLIONNET A & CALMELS D. 1996. Linear programming for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, **92**: 310–325.

[8] BILLIONNET A, ELLOUMI S & LAMBERT A. Extending the QCR method to general mixed-integer programs. Mathematical Programming, Pub. on line, 21 pages, Doi: 10.1007/s10107-010-0381-7.

[9] BILLIONNET A, ELLOUMI S & PLATEAU M-C. 2009. Improving the performance of standard solvers via a tighter convex reformulation of constrained quadratic 0-1 programs: the QCR method. *Discrete Applied Mathematics*, **157**: 1185–1197.

[10] BONAMI P & LEJEUNE M. 2009. An exact solution approach for portfolio optimization problems under stochastic and integer constraints. *Operations Research*, **57**: 650–670.

[11] BORCHERS S. 1999. CSDP, a C library for semidefinite programming. *Optimization methods and Software*, **11**(1): 613–623.

[12] BOROS E & HAMMER P. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics*, **123**: 115–225.

[13] CAPRARA A, KELLERER H, PFERSCHY U & PISINGER D. 2000. Approximate algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, **123**: 333–345.

[14] CAPRARA A, PISINGER D & TOTH P. 1999. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11: 125–137.

[15] CHAILLOU P, HANSEN P & MAHIEU Y. 1986. Best network flow bounds for the quadratic knapsack problem. *Lecture Notes in Mathematics*, **1403**: 226–235.

[16] CHARDAIRE P & SUTTER A. 1995. A decomposition method for quadratic 0-1 programming. *Management Science*, **41**(4): 704–712.

[17] COIN-OR. COmputational INfrastructure for Operations Research. http://www.coin-or.org.

[18] CPLEX. 2009. IBM ILOG CPLEX Callable library version 12.1 Reference manual. http://docs.hpc.maths.unsw.edu.au/ilog/cplex/12.1.

[19] DUDZINSKI K. 1989. On a cardinality constrained linear programming knapsack problem. *Operations Research Letters*, **8**: 215–218.

[20] FAYARD D & PLATEAU G. 1982. Algorithm 47: an algorithm for the solution of the 0-1 knapsack problem. *Computing*, **28**: 269–287.

[21] FAYE A & ROUPIN F. 2007. Partial Lagrangian relaxation for general quadratic programming. *4OR*, **5**(1): 75–88.

[22] FRÉVILLE A & PLATEAU G. 1993. An exact search for the solution of the surrogate dual of the 0-1 bidimensional knapsack problem. *European Journal of Operational Research*, **68**: 413–421.

[23] GALLO G, HAMMER PL & SIMEONE B. 1980. Quadratic knapsack problems. *Mathematical Programming Study*, **12**: 132–149.

[24] GLOVER F. 1965. A multiphase dual algorithm for the 0-1 integer programming problem. *Operations Research*, **13**(6): 879–919.

[25] HAMMER P, HANSEN P & SIMEONE B. 1984. Roof duality, Complementation and Persistency in Quadratic 0-1 Optimization. *Mathematical Programming*, **28**: 121–155.

[26] HELMBERG C & RENDL F. 1998. Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, **8**(3): 291–315.

[27] HELMBERG C, RENDL F & WEISMANTEL R. 2000. A semidefinite programming approach to quadratic knapsack problems. *Journal of Combinatorial Optimization*, **4**: 197–215.

[28] KELLERER H, PFERSCHY U & PISINGER D. 2004. Knapsack Problems. Springer-Verlag.

[29] LÉTOCART L, PLATEAU MC & PLATEAU G. 2007. Lagrangean and convexification methods for the 0-1 *k*-item quadratic knapsack problem. In Proceedings of NCP07 (International Conference on Non Convex Programming), Rouen (France), December.

[30] MICHELON P & MACULAN N. 1991. Lagrangean decomposition for integer nonlinear programming with linear constraints. *Mathematical Programming*, **52**(2): 303–314.

[31] MITRA G, ELLISON F & SCOWCROFT A. 2007. Quadratic programming for portfolio planning: Insights into algorithmic and computational issues. Part ii: Processing of portfolio planning models with discrete constraints. *Journal of Asset Management*, **8**: 249–258.

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

[33] PLATEAU G & NAGIH A. 2010. 0-1 Knapsack Problems. In Combinatorial Optimization: Paradigms of Combinatorial Optimization (V. Paschos eds), ISTE ltd-John WILEY and Sons Pub., volume 2, chapter 8, 215–242.

[34] POSNER ME & GUIGNARD M. 1978. The collapsing 0-1 knapsack problem. *Mathematical Programming*, **15**: 155–161.

[35] SHAWA D, LIUB S & KOPMANB L. 2008. Lagrangean relaxation procedure for cardinality-constrained portfolio optimization. *Optimization Methods and Software*, **23**: 411–420.

[36] SHERALI HD & ADAMS WP. 1999. A Reformulation-Linearization Tech nique for Solving Discrete and Continuous Nonconvex Problems. Kluwer Academic Publ., Dordrecht, Boston, London.

[37] ZHU WX. 2003. Penalty parameter for linearly constrained 0-1 quadratic programming. *Journal of Optimization Theory and Applications*, **116**(1): 229–239.