Um algoritmo *Branch-and-Cut* para o Problema da Mochila Quadrática 0-1

Jesus Ossian Cunha
jossian@cos.ufrj.br

Luidi Simonetti
luidi@ic.uff.br

Abilio Lucena
abiliolucena@cos.ufrj.br

RESUMO

Este artigo introduz um algoritmo exato para o Problema da Mochila Quadrática. O algoritmo, do tipo *Branch-and-Cut*, se baseia numa linearização da formulação clássica do problema, fortalecida com a utilização de algumas famílias de desigualdades válidas. A primeira delas está associada à quadratização da desigualdade de definição do problema, ou seja, sua desigualdade da mochila. As demais definem facetas do Politopo Booleano Quadrático e do Politopo da Mochila 0-1. Desigualdades dos dois últimos tipos são separadas dinamicamente, à medida que se fazem necessárias. Por sua vez, aquelas do primeiro tipo são incorporadas à reformulação linear, desde o início. Experimentos computacionais atestam que o algoritmo é uma opção atraente para a resolução exata do problema.

PALAVRAS CHAVE: Problema da mochila quadrática, algoritmo *Branch-and-Cut*, desigualdades válidas.

ABSTRACT

This paper introduces an exact solution algorithm for the Quadratic Knapsack Problem. The Branch-and-Cut type algorithm is based on a linearization of a classical formulation of the problem, strengthened with the use of some families of valid inequalities. The first type of inequality results from the quadratization of the problem defining inequality, i.e., its corresponding knapsack inequality. The other two types of inequalities are associated with facet defining inequalities for the Boolean Quadric Polytope and the 0-1 Knapsack Polytope. Inequalities of the latter types are separated dynamically, as they become necessary. In turn, inequalities of the former type are incorporated to the linear reformulation, right from the beginning. Computational experiments indicate that the algorithm is an attractive option for solving the problem exactly.

KEYWORDS: Quadratic Knapsack Problem, Branch-and-Cut algorithm, valid inequalities.

1 Introdução

Suponha que estão disponíveis um conjunto de n objetos $N = \{1, \ldots, n\}$, com pesos $W = \{w_i \in \mathbb{N} : i \in N\}$, e uma mochila com capacidade $c \in \mathbb{N}$, sendo $\max_{i \in N} w_i \leq c \leq \sum_{i \in N} w_i$. Colocar na mochila um objeto $i \in N$ traz um benefício $p_{ii} \in \mathbb{N}$ e deseja-se escolher um subconjunto de objetos $N' \subseteq N$ que, carregados na mesma, leve a um benefício total máximo. Denomina-se Problema da Mochila 0-1 (KP) [15, 19, 23] a escolha de um tal N'.

Se benefícios cruzados p_{ij}, são também obtidos quando $i, j \in N'$, a escolha de N' passa a se chamar Problema da Mochila Quadrática 0-1 (QKP) [3–5, 7, 12, 20, 22, 24].

Introduzindo um conjunto de variáveis $x = \{x_i \in \{0, 1\} : i \in N\}$, tomando $q_i = p_{ii}$ e notando que $x_i = x_i \times x_i$, QKP pode ser formulado como:

$$\max \sum_{i \in N} q_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} x_i x_j \tag{1}$$

$$\text{s.a.} \sum_{i \in N} w_i x_i \leq c \tag{2}$$

$$x_i \in \{0, 1\}, \quad i \in N. \tag{3}$$

QKP pode ser interpretado em termos de Teoria dos Grafos. Nesse sentido, seja $G = (V, E)$ um grafo completo não direcionado com n vértices onde cada vértice $i \in V$ leva a um lucro p_{ii} e tem um peso w_i. Da mesma forma, cada aresta $e = \{i, j\} \in E$, onde $i < j$, leva a um lucro p_{ij}. Definido G, pede-se para encontrar um subconjunto de vértices $S \subseteq V$ onde $\sum_{i \in S} w_i \leq c$ e o lucro total auferido seja o máximo.

Apesar de sua fácil formulação, o QKP é um problema desafiador, tendo sido amplamente estudado nas últimas três décadas [3–5, 7, 12, 20, 22, 24]. QKP é um problema NP-difícil pois admite KP, que pertence àquela classe de problemas, como um caso particular.

QKP possui uma grande quantidade de aplicações, podendo ser citadas, dentre outras:

- Witzgall descreve em [26] um problema de telecomunicação onde um conjunto de estações de satélites tem de ser selecionado, tal que o tráfico entre as estações é maximizado e uma retrição de orçamento tem de ser respeitada. Ou seja, um problema similar a QKP.
- Johnson, Mehrotra, Nemhauser em [13] citam um problema de projeto de compiladores que pode ser formulado como um QKP.
- O problema de encontrar cliques maximais de um grafo [7, 14, 22] pode ser formulado como um QKP [7, 22]. Para tanto, assuma que um grafo esparso não direcionado $G = (V, E)$ é dado. Denomina-se uma clique a um subgrafo completo de G e o problema em questão é o de encontrar a maior clique contida naquele grafo.

Ao longo das últimas três decadas diversos limitantes superiores e algoritmos tem sido propostos para o QKP.

Gallo, Hammer and Simeone [12] propuseram um algoritmo tipo *Branch-and-Bound* [18, 27] para resolver o problema. Isto é feito calculando limitantes superiores através da substituição da função objetivo $f(x) = \sum_{i \in N} \sum_{j \in N} p_{ij} x_i x_j$ por uma função linear $g(x)$ tal que $g(x)$ domina $f(x)$ em todos os ponto viáveis.

Especificamente, limitantes superiores são obtidos pela resolução do seguinte problema:

$$\begin{aligned} &\max\ g(x) \\ &\text{s.a} \sum_{i \in N} w_i x_i \leq c \\ &\quad x_i \in \{0,1\}\,,\ i \in N. \end{aligned}$$

Chaillou, Hansen and Mahieu em [8] também propuseram um algoritmo do tipo *Branch-and-Bound* para o QKP. Entretanto, calculam limitantes superiores através de uma Relaxação Lagrangeana [2] da formulação utilizada.

Em outro algoritmo do tipo *Branch-and-Bound*, Michelon e Veuilleux [20] se utilizam de Decomposição Lagrangeana [30] para calcular limitantes superiores.

Mais uma vez utilizando Relaxação Lagrangeana, Billionet e Calmels [5] propuseram um algoritmo do tipo *Branch-and-Bound* para o problema.

Posteriormente, Billionet, Faye e Soutif [3] propuseram em um algoritmo do tipo *Branch-and-Bound* baseado em Decomposição Lagrangeana e partição de conjuntos.

Ainda utilizando Relaxação Lagrangeana, Palmeira [22] propôs um algoritmo do tipo *Branch-and-Bound* onde limitantes superiores são calculados através de um algoritmo *Relax-and-Cut* [10, 16, 17], ou seja, um análogo Lagrangeano de algoritmos de planos-de-corte.

No que é provavelmente o algoritmo exato mais citado na literatura do QKP, Caprara, Pisinger e Toth [7] propuseram um algoritmo do tipo *Branch-and-Bound*, baseado numa reformulação do problema. Neste algoritmo, limitantes superiores são calculados através de Relaxação Lagrangeana e técnicas de fixação de variáveis e redução do problema são também utilizadas.

Ao contrário dos algoritmos anteriores que, com uma única excessão, se utilizam de Relaxação Lagrangeana para o cálculo de limitantes superiores, o algoritmo proposto por Billionet e Soutif [6] se utiliza de Programação Linear para cumprir tal função. Da mesma forma, utiliza de resolvedores de Programação Inteira Mista (MIP) para obter soluções comprovadamente ótimas para o problema.

Finalmente, Rodrigues, Quadri, Michelon e Gueye [25] sugeriram recentemente um algoritmo do tipo *Branch-and-Bound* para o problema. Este, é baseado em um esquema de linearização, denominado t-linearização pelos autores, e se utiliza de Programação Linear para a obtenção de limitantes superiores.

Para maiores detalhes sobre o QKP, sugerimos [15, 19, 23] como fontes de consulta.

2 Desigualdades válidas para o QKP 0-1

Considere a restrição da mochila,

$$\sum_{i \in N} w_i x_i \leq c. \tag{4}$$

Multiplicando a mesma por x_j, para cada $j \in N$, obtemos as seguinte desigualdades quadratizadas:

$$\sum_{i \in N \setminus \{j\}} w_i x_i x_j \leq (c - w_j)\, x_j,\ j \in N. \tag{5}$$

Tais desigualdades, que são claramente válidas para o QKP, foram propostas por Adams e Sherali em [1].

Para efeito de linearização dos termos quadráticos da função objetivo do QKP e das desigualdades (5), considere o uso de variáveis $y = \{y_{ij} \in \{0,1\} :\ i < j,\ i, j \in N\}$. Dessa forma, $x_i x_j$ deve ser então substituído por y_{ij}. Adicionalmente, desigualdades válidas propostas por Padberg [21]

para o Politopo Booleano Quadrático (BQP) podem ser utilizadas para ajudar a aproximar o valor de y_{ij} daquele correspondente a $x_i x_j$. Nesse sentido, as desigualdades

$$y_{ij} \leq x_i,\ i < j,\ i,j \in N \tag{6}$$
$$y_{ij} \leq x_j,\ i < j,\ i,j \in N \tag{7}$$
$$x_i + x_j \leq 1 + y_{ij},\ i < j,\ i,j \in N. \tag{8}$$

se mostram particularmente úteis.

As desigualdades (5) e (6)-(8) foram utilizadas por Billionet e Calmels [5], Palmeira, Porto e Lucena [22] e Caprara, Pisinger e Toth [7]. Nesta última referência, para efeito de uma reformulação de QKP, variáveis adicionais $\{y_{ij} \in \{0,1\} : i > j,\ i,j \in N\}$ são também utilizadas, sob a restrição adicional de que $y_{ij} = y_{ji}$, para qualquer par $i,j \in N$ onde $i \neq j$. Ainda naquela referência, o benefício cruzado p_{ij}, relativo a um par $i,j \in N$, onde $i \neq j$, é dividido igualmente entre as duas variáveis y_{ij} e y_{ji} que lhe são correspondentes.

Outra família de desigualdades válidas para o BQP, denominadas desigualdades do triângulo, foram também propostas Padberg [21]. Estas foram utilizadas por Palmeira em [22] e são descritas por:

$$x_i + x_j + x_k - y_{ij} - y_{ik} - y_{jk} \leq 1,\ i < j < k,\ i,j,k \in N \tag{9}$$
$$-x_i + y_{ij} + y_{ik} - y_{jk} \leq 0,\ i < j < k,\ i,j,k \in N \tag{10}$$
$$-x_j + y_{ij} + y_{ik} - y_{jk} \leq 0,\ i < j < k,\ i,j,k \in N \tag{11}$$
$$-x_k + y_{ij} + y_{ik} - y_{jk} \leq 0,\ i < j < k,\ i,j,k \in N. \tag{12}$$

De forma a introduzir desigualdades válidas adicionais para o QKP, considere o conjunto

$$K = \left\{ \mathbf{x} : \sum_{i \in N} w_i x_i \leq c,\ x_i \in \{0,1\}^n,\ i \in N \right\},$$

associado ao KP.

Definição 2.1 *Seja $C \subseteq N$. Dizemos que C é uma cover se $\sum_{i \in C} w_i > c$. Uma cover é mínima se $C \setminus \{i\}$ não é uma cover para algum $i \in C$.*

Considere um vetor de incidência (ou vetor caracteristico), x^C, associado a C e definido como se segue: $x_i = 1$ se $i \in C$ e $x_i = 0$ se $i \notin C$. Assim sendo, C é uma *cover* se e somente se x^C é inviável para K.

Proposição 2.1 *Seja $C \subseteq N$ uma cover para K. A desigualdade de cover,*

$$\sum_{i \in C} x_i \leq |C| - 1, \tag{13}$$

é válida para K.

Prova 2.1 *Veja Wolsey [27].*

Definição 2.2 *Seja C uma cover de K, uma extensão de C é dada pelo subconjunto:*

$$E(C) = C \cup \{i \in N \setminus C\ :\ w_i \geq w_j,\ \forall\, j \in C\}.$$

Proposição 2.2 *Seja C uma cover para K. Então a desigualdade de extended cover,*

$$\sum_{i \in E(C)} x_i \leq |C| - 1, \tag{14}$$

é válida para K.

Prova 2.2 *Veja Wolsey [27].*

Desigualdades de *cover* serão aqui utilizadas para gerar desigualdades válidas para o QKP. De forma a introduzir tais desigualdades, vamos primeiramente mostrar como desigualdades de *cover* violadas podem ser identificadas. Seja $\mathbf{x}^*$ uma solução ótimo para o Problema de Relaxação Contínua de KP, formulado como

$$\max \sum_{i \in N} p_i x_i \tag{15}$$

$$\text{s.a} \sum_{i \in N} w_i x_i \leq c \tag{16}$$

$$x_i \in [0,1]\,,\ i \in N. \tag{17}$$

Associado a $\mathbf{x}^*$, o Problema de Separação (PS) das desigualdades de *cover* consiste em identificar uma desigualdade de *cover* violada por $\mathbf{x}^*$ ou estabelecer que nenhuma desigualdade desse tipo existe.

Um procedimento para resolver o PS definido acima, foi sugerido por Crowder, Johnson e Padberg em [9]. Para descrevê-lo, considere o KP

$$\gamma = \min \left\{ \sum_{i \in N} (1 - x_i^*) t_i : \sum_{i \in N} w_i t_i > c + 1,\ t_i \in \{0,1\}\,,\ i \in N \right\} \tag{18}$$

e o teorema que se segue, relativo a ele.

Teorema 2.1 *Veja Wolsey [27]*

1. *se $\gamma \geq 1$, $\mathbf{x}^*$ satisfaz a todas as desigualdades de cover.*
2. *se $\gamma < 1$, com uma solução ótima para (18) dada por $\mathbf{t}^R$, a desigualdade de cover $\sum_{i \in R} x_i \leq |R| - 1$ é violada por $\mathbf{x}^*$ em uma quantidade $1 - \gamma$.*

Prova 2.3 *Veja Wolsey [27]*

Seja $conv(K)$ o fecho convexo definido pelas soluções viáveis de KP. Balas [29] e Wolsey [28] mostraram que, dada uma desigualdade de *cover* mínima, existe no mínimo uma faceta definida pela desigualdade:

$$\sum_{i \in C} x_i + \sum_{i \in N \setminus \{C\}} \alpha_i x_i \leq |C| - 1 \tag{19}$$

onde $\alpha_i \geq 0$ para todo $i \in N \setminus \{C\}$. Esta desigualdade é denominada desigualdade de *lifting*.

Wolsey [27] descreve um procedimento para encontrar desigualdades de *lifting* que definem facetas para a $conv(K)$, quando C é uma *cover* mínima e $w_i \leq c$, para todo $i \in N$.

Algorithm 1 Procedimento para encontrar desigualdades de *lifting* [27]

$r = |N \setminus C|$;
Ordene os indices $i \in N \setminus C$ em ordem crescente de acordo com cada peso w_i;
for $t = 1$ **to** r **do**
Resolva o problema,

$$\begin{aligned} \zeta_t = \max & \sum_{j=1}^{t-1} \alpha_{i_j} x_{i_j} + \sum_{i \in C} x_i \\ \text{s.a} & \sum_{j=1}^{t-1} w_{i_j} x_{i_j} + \sum_{i \in C} w_i x_i \leq c - w_{i_j} \\ & x \in \{0,1\}^{|C|+t-1} . \end{aligned}$$

$\alpha_{i_t} = |C| - 1 + \zeta_t$;
end for

3 Exemplo numérico

Considere a seguinte instância para o QKP 0-1.

$$\begin{aligned} \max\ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + \\ & 14y_{12} + 14y_{13} + 49y_{14} + 37y_{15} + 33y_{16} + 3y_{17} + \\ & 13y_{23} + 36y_{24} + 6y_{25} + 2y_{26} + y_{27} + \\ & 26y_{34} + 21y_{35} + 11y_{36} + 2y_{37} + \\ & 12y_{45} + 46y_{46} + 10y_{47} + \\ & 24y_{56} + 46y_{57} + \\ & 19y_{67} \\ \text{s.a. } & 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 + x_7 \leq 19 \\ & x_i \in \{0,1\}\,,\ i \in \{1,\ldots,7\} \end{aligned} \tag{20}$$

Utilizando o resolvedor *Xpress Optimizer*, obtemos a seguinte solução para Problema de Relaxação Contínua associado a (20):

$$\overline{\mathbf{x}} = \left[\ 0,00 \quad 0,12 \quad 0,58 \quad 0,92 \quad 0,92 \quad 0,92 \quad 0,92\ \right] \text{e}$$

$$\overline{\mathbf{y}} = \begin{bmatrix} 0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 \\ & 0,00 & 0,12 & 0,12 & 0,06 & 0,12 \\ & & 0,50 & 0,50 & 0,50 & 0,50 \\ & & & 0,92 & 0,92 & 0,92 \\ & & & & 0,92 & 0,92 \\ & & & & & 0,92 \end{bmatrix},$$

para uma função objetivo de valor igual a $363,46$.

Resolvendo o Problema de Separação associado às desigualdade de *cover*, obtemos a solução

$$z = \left[\ 0,00 \quad 0,00 \quad 1,00 \quad 1,00 \quad 1,00 \quad 1,00 \quad 0,00\ \right],$$

com valor ótimo igual a $0,65$. Dessa maneira, temos que $C = \{3,4,5,6\}$ e

$$x_3 + x_4 + x_5 + x_6 \leq 3$$

é a desigualdade de *cover* encontrada. Esta, de acordo com (18) e o Teorema (2.1), é violada por $\overline{\mathbf{x}}$.

Quadratizando a desigualdade de *cover* obtida acima, obtemos as seguintes desigualdades:

$$y_{13} + y_{14} + y_{15} + y_{16} \leq 3x_1 \quad (21)$$
$$y_{23} + y_{24} + y_{25} + y_{26} \leq 3x_2 \quad (22)$$
$$x_3 + y_{34} + y_{35} + y_{36} \leq 3x_3 \quad (23)$$
$$y_{34} + x_4 + y_{45} + y_{46} \leq x_4 \quad (24)$$
$$y_{35} + y_{45} + x_5 + y_{56} \leq x_5 \quad (25)$$
$$y_{36} + y_{46} + y_{56} + x_6 \leq 3x_6 \quad (26)$$
$$y_{37} + y_{47} + y_{57} + y_{67} \leq 3x_7. \quad (27)$$

Em particular, para a solução $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, as desigualdades (21) e (22) são não-violadas, enquanto (23), (24), (25), (26) e (27) são violadas.

Observe que, dada uma desigualdade de *cover* violada, é possível obter uma desigualdade de *cover* quadratizada não-violada onde o termo referente a quadratização está definido no intervalo $(0, 1)$ (na verdade, aquele termo pode ser definido no intervalo $(0, 1]$). Para tanto, observe que $y_{ij} \leq x_i$ e $y_{ij} \leq x_j$ se aplicam e, dessa forma, $0 < x_j \leq 1$ e $y_{ij} \neq x_i x_j$ resultam. Isto, por sua vez, implica em um decréscimo do lado esquerdo da desigualdade de *cover* quadratizada. Já para o caso onde $x_i = 0$, temos que a desigualdade quadratizada obtida será sempre não-violada.

Dada uma desigualdade de *cover* não-violada, seria possível obter uma desigualdade de *cover* quadratizada, a ela associada, e que seja violada? Note que, nesse caso, a desigualdade de *cover* quadratizada seria obtida pela multiplicação da desigualdade de *cover* por um termo pertencente ao conjunto que a define. A resposta a pergunta aqui formulada será dada através da construção de um exemplo.

Suponha que temos a seguinte solução para relaxação contínua do QKP previamente definido:

$$\overline{\mathbf{x}} = \left[\begin{array}{cccccc} 0,00 & 0,50 & 0,00 & 0,50 & 0,00 & 0,50 \end{array} \right].$$

Considere uma *cover* definida pelo conjunto $C = \{2, 4, 6\}$ e observe que a mesma é não-violada por $\overline{\mathbf{x}}$. Quadratizando a desigualdade de *cover* correspondente a C, para $j \notin C$, obtemos

$$y_{12} + y_{14} + y_{16} \leq 2x_1 \quad (28)$$
$$y_{23} + y_{34} + y_{36} \leq 2x_3 \quad (29)$$
$$y_{25} + y_{45} + y_{56} \leq 2x_5. \quad (30)$$

Estas, são não-violadas, pois o máximo valor que y_{ij} podem assumir é 0, de acordo com as desigualdades $y_{ij} \leq x_i,\ i < j,\ i, j \in N$ e $y_{ij} \leq x_j,\ i < j,\ i, j \in N$. Efetuando o mesmo procedimento para $j \in C$, obtemos

$$x_2 + y_{24} + y_{26} \leq 2x_2 \quad (31)$$
$$y_{24} + x_4 + y_{46} \leq 2x_4 \quad (32)$$
$$y_{26} + y_{46} + x_6 \leq 2x_6. \quad (33)$$

Observe então que temos $y_{24} \leq x_2,\ y_{24} \leq x_4,\ y_{26} \leq x_2,\ y_{26} \leq x_6,\ x_2 + x_4 \leq 1 + y_{24}$ e $x_2 + x_6 \leq 1 + y_{26}$, onde o maior valor que y_{24} e y_{26} podem assumir é $0,50$. Dessa forma, temos que

$$y_{24} + y_{26} + x_2 = 1,5$$

e

$$(|C|-1)x_2 = 2 \times 0,50 = 1,00$$

resultam. Logo,

$$y_{24} + y_{26} + x_2 > (|C|-1)x_2$$

se aplica e temos uma desigualdade quadratizada não-violada.

Vamos agora encontrar uma desigualdade de *lifting*. Utilizando o Algoritmo 1, obtemos a seguinte desigualdade de *lifting*,

$$2x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 4,$$

definida para $L = \{1,2,3,4,5,6\}$. Esta desigualdade é violada por $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$.

Observe que se a desigualdade de *cover* obtida é violada, a desigualdade de *lifting* também o será.

Quadratizando a desigualdade de *lifting*, obtemos as seguintes desigualdades

$$2x_1 + y_{12} + y_{13} + y_{14} + y_{15} + y_{16} \leq 3x_1 \quad (34)$$
$$2y_{12} + x_2 + y_{23} + y_{24} + y_{25} + y_{26} \leq 3x_2 \quad (35)$$
$$2y_{13} + y_{23} + x_3 + y_{34} + y_{35} + y_{36} \leq 3x_3 \quad (36)$$
$$2y_{14} + y_{24} + y_{34} + x_4 + y_{45} + y_{46} \leq 3x_4 \quad (37)$$
$$2y_{15} + y_{25} + y_{35} + y_{45} + x_5 + y_{56} \leq 3x_5 \quad (38)$$
$$2y_{16} + y_{26} + y_{36} + y_{46} + y_{56} + x_6 \leq 3x_6 \quad (39)$$
$$2y_{17} + y_{27} + y_{37} + y_{47} + y_{57} + y_{67} \leq 3x_7. \quad (40)$$

Observe que (34) não é violada por $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, enquanto (35), (36), (37), (38), (39) e (40) são.

De acordo com o exemplo acima, temos que, dados os elementos pertencentes ao conjunto das *lifting*, L, ao quadratizamos uma desigualdade de *lifting* em relação a tais elementos, as desigualdades resultantes podem ou não serem violadas pela solução contínua em mãos.

Considere o conjunto que define uma *cover*, $C = \{1, ..., p\}$, e o conjunto dos elementos fora da cobertura, $N \setminus C = \{p+1, \ldots, n\}$. Para encontramos uma desigualdade de *lifting*, de acordo com o Algoritmo 1, ordenamos decrescentemente os elementos do conjunto $N \setminus C$, de acordo com $w_i,\ i \in N \setminus C$. Feito isso, resolvemos, para $1 < t < |N \setminus C|$, os seguintes problemas:

$$\begin{aligned} \zeta_t = \max & \sum_{j=1}^{t-1} \alpha_{i_j} x_{i_j} + \sum_{i \in C} x_i \\ \text{s.a} & \sum_{j=1}^{t-1} w_{i_j} + \sum_{i \in C} w_i \leq b - w_t \\ & x \in \{0,1\}^{|C|+t-1}, \end{aligned}$$

onde $\alpha_{i_t} = |C| - 1 - \zeta_t$.

Suponha, para algum j tal que $1 < j < |N \setminus C|$, que $\alpha_{i_j} = 0$ se aplica, ou seja, que $\zeta_j = 1 - |C|$. Observe que, para $k \in \{t+1, t+2, \ldots, |N \setminus C|\}$ temos $b - w_k \leq b - w_t$. Assim sendo, chegamos a

$$\zeta_k \geq \zeta_t,$$

o que implica que $\alpha_{ik} \leq \alpha_{i_t}$. Como $\alpha_{i_k} \geq 0$ para todo k, temos então que $\alpha_{i_k} = 0$, para todo $k > i_t$.

4 *Algoritmo Branch-and-Cut para o QKP*

Um algoritmo *Branch-and-Cut* [27] é um algoritmo do tipo *Branch-and-Bound* [15] no qual planos de cortes são gerados para os diferentes problemas definidos nos nós da árvore de enumeração. Isso é feito com o intuito de fortalecer os limitantes duais associados aos mesmos e eventualmente reduzir o número de nós a enumerar na árvore.

Na prática existe um *trade-off* a ser respeitado ao se implementar um algoritmo *Branch-and-Cut*. Se muitos planos de cortes são adicionados em cada nó da árvore de enumeração, as reotimizações podem se tornar muito caras. Demandas adicionais relativas ao uso de memória RAM devem também ser consideradas, já que temos que armazenar todas as informações relativas aos problemas definidos em cada nó da árvore de enumeração. Dependendo do número de cortes gerados, essa demanda pode se tornar significativa.

Um *cut pool* é utilizada para armazenar informações referente aos cortes. Por exemplo, limites duais e bases Simplex a eles associados devem ser guardadas. Da mesma forma, é também necessário indicar quais planos de cortes são necessárias para reconstruir a formulação relativa a um dado nó da árvore de procura (os ponteiros para estas restrições devem também ser armazenados no *cut pool*).

O algoritmo *Branch-and-Cut* que implementamos para o QKP se baseia na separação de desigualdades válidas para o KP. Nossa implementação se utilizou do resolvedor de Programação Linear Inteira Mista, *Xpress Optimizer*. A escolha da variável a ser ramificada na árvore de enumeração foi determinada automaticamente pelo *Xpress Optimizer*, para percorrer a árvore de enumeração utilizamos a busca em profundidade. Desabilitamos o preprocessamento, estratégias de cortes e estratégias de heurísticas do *Xpress Optimizer*. Prioridade foi atribuida às variáveis $\mathbf{y}$. No mais, todas as funções de gerenciamento da árvore de enumeração foram deixados a cargo do resolvedor.

Em nossa implementação, utilizamos a seguinte reformulação linear do QKP:

$$\max \sum_{i\in N} q_i x_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} p_{ij} y_{ij} \tag{41}$$

$$\text{s.a.} \sum_{i\in N} w_i x_i \leq c \tag{42}$$

$$\sum_{i<j,\ i\in N} w_i y_{ij} + \sum_{i>j,\ i\in N} w_i y_{ij} \leq (c - w_j)x_j,\ j \in N \tag{43}$$

$$y_{ij} \leq x_i,\ i < j,\ i,j \in N \tag{44}$$

$$y_{ij} \leq x_j,\ i < j,\ i,j \in N \tag{45}$$

$$x_i + x_j \leq 1 + y_{ij},\ i < j,\ i,j \in N \tag{46}$$

$$x_i, y_{ij} \in \{0,1\},\ i < j,\ i,j \in N. \tag{47}$$

Especificamente, trabalhamos sobre a relaxação contínua de (41)-(47). Neste caso, substituimos as restrições $x_i, y_{ij} \in \{0,1\},\ i < j,\ i,j \in N$, pelas desigualdades $x_i,\ y_{ij} \in [0,1],\ i < j,\ i,j \in N$, respectivamente.

Além das desigualdades que utilizamos para reforçar a relaxação contínua de (41)-(47) e, por conseguinte, as relaxações contínuas dos problemas definidos nos demais nós de nossa árvore de enumeração, poderíamos também ter utilizado outras famílias de desigualdades válidas. No entanto, nossos experimentos computacionais indicaram que o preço computacional que teríamos que pagar para fazê-lo, não se justificaria. Optamos então por reforçar nossas formulações, em cada nó da árvore de enumeração, utilizando apenas as desigualdades (9), as desigualdades de *cover* (13), as desigualdades *extended* (14) e as desigualdades de *lifting* (19). Desigualdades violadas pertencentes a tais famílias são inseridas ao problema em mãos apenas quando violadas pela solução corrente.

Como visto na Seção 2, desigualdades de *cover* violadas são identificadas através da resolução de um KP auxiliar. Podemos resolver este KP utilizando, dentre outras alternativas, Programação Dinâmica [27] ou o próprio resolvedor *Xpress Optimizer* [11], por exemplo. O procedimento utilizado para identificar desigualdades (13), (14) e (19) violadas é aquele descrito no Seção 2. De acordo com nossos testes computacionais, utilizar o resolvedor *Xpress Optimizer* [11] para resolver tais problemas de separação se mostrou uma alternativa satisfatória e é a que aqui empregamos.

5 Resultados Computacionais

As instâncias utilizadas em nossos testes computacionais foram geradas aleatoriamente de acordo com [5, 7, 8, 12, 20]. Tais instâncias obdecem as regras de geração que descreveremos a seguir. O símbolo $\%$ define a densidade da instância, ou seja, o percentual de benefícios $p_{ij},\ i \leq j,\ i, j \in N$ não-nulos. Cada peso w_j é distribuído aleatoriamente em $[1, 50]$ para cada $j \in N$, enquanto os benefícios $p_{ij},\ i \leq j,\ i, j \in N$ com densidade $\%$ são distribuídos aleatoriamente entre$[1, 100]$. A capacidade c é distribuída aleatoriamente em $[50, \sum_{j=1}^{n} w_j]$.

Em nossos testes computacionais utilizamos uma máquina com a seguinte configuração: *Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz, 2 GB, gcc/g++ 4.6.3, SO Ubuntu*. Utilizamos ainda o resolvedor *Xpress Optimizer*, *release* 23.

As tabelas 1 e 2 trazem os resultados referente ao algoritmo *Branch-and-Cut* descrito na Seção 4, onde comparamos entre si as desigualdades (9), *cover*, *extender* e *lift* adicionados como plano de cortes, quando violadas, ao longo da árvore de *Branch-and-Bound*. Mostramos ainda resultados referente ao modelo (41)-(47) apenas, onde passamos o mesmo para o resolvedor *Xpress Optimizer* e deixamos a resolução a cargo do resolvedor, sem alterar parâmetros.

Nas tabelas 1 e 2 temos que a coluna n indica o tamanho do problema e $\%$ sua densidade, a coluna *xpress* indica os resultados do modelo (41)-(47) passado para o resolvedor, enquanto as colunas *triângulo*, *cover*, *extended* e *lifting* indicam desigualdades do *triângulo*, *cover*, *extended* e de *lifting* respectivamente adicionas como plano de cortes, quando violadas ao modelo (41)-(47), em um algoritmo tipo *Branch-and-Cut*. Temos ainda que a coluna *cut* indica o número de cortes inseridos ao longo da árvore de busca, a coluna *no* indica o números de nós visitados na árvore de busca do *Branch-and-Cut* e a coluna *t(s)* o tempo de execução do *Branch-and-Cut*. Cada linha refere-se a uma média de 10 instâncias.

Tabela 1: *Branch-and-Cut* para o QKP, onde $n \in \{20, \ldots, 80\}$

		xpress		*triângulo*			*cover*			*extended*			*lifting*		
n	*%*	*no*	*t(s)*	*cut*	*no*	*t(s)*	*cut*	*no*	*t(s)*	*cut*	*no*	*t(s)*	*cut*	*no*	*t(s)*
20	25	9	0,4	204	7	0,8	7	7	0,3	4	6	0,3	5	6	0,5
	50	14	0,5	164	10	0,8	10	11	0,7	6	8	0,5	6	8	0,7
	75	21	0,6	117	15	0,7	20	14	0,9	6	11	0,7	4	8	0,9
	100	25	0,5	136	20	0,8	16	20	1,0	4	8	0,6	4	8	0,9
40	25	33	3,6	1614	35	10,5	20	24	3,6	14	24	3,3	13	19	3,7
	50	35	4,8	646	27	8,2	18	22	4,7	12	21	4,6	5	17	4,1
	75	64	6,3	779	54	8,6	24	25	5,7	5	13	4,0	7	13	4,6
	100	254	11,9	1968	205	15,5	116	181	14,3	13	40	9,2	14	31	8,1
60	25	29	10,9	4947	69	135,0	26	35	12,3	13	29	10,0	23	38	14,0
	50	158	37,3	3751	120	133,6	67	104	37,0	30	85	29,6	26	82	35,6
	75	312	62,4	4632	269	123,3	129	219	65,1	34	117	47,5	53	121	51,2
	100	2366	233,2	18836	1796	370,6	799	1477	234,0	68	184	70,1	19	63	40,0
80	25	95	53,4	9000	103	881,6	51	47	39,5	56	74	53,2	27	56	45,6
	50	115	86,3	8328	150	1844,9	66	89	88,3	25	75	78,2	23	57	77,8
	75	2003	616,5	51688	1500	1266,9	671	1091	632,6	99	204	209,9	58	135	163,8
	100	6513	1592,4	49653	7914	2236,0	1462	2746	1605,2	20	89	128,0	22	76	149,3

Tabela 2: *Branch-and-Cut* para o QKP, onde $n \in \{100, 120\}$

		xpress		*cover*			*extended*			*lifting*		
n	%	*no*	*t(s)*	*cut*	*no*	*t(s)*	*cut*	*no*	*t(s)*	*cut*	*no*	*t(s)*
100	25	109	132,7	35	38	75,2	31	46	83,6	20	36	73,7
	50	110	195,3	85	114	260,9	24	60	173,6	39	122	277,7
	75	310	367,2	98	170	286,4	97	213	410,4	28	62	184,6
	100	600	530,8	234	508	534,7	15	54	186,7	54	134	337,2
120	25	274	333,6	56	152	252,4	17	99	129,9	36	132	198,1
	50	482	686,9	72	105	558,3	25	62	329,7	30	83	378,4
	75	1138	2453,0	1193	2351	3891,6	288	742	2099,4	57	123	828,4
	100	8656	7855,1	1510	2779	4158,6	13	61	457,7	25	82	550,6

6 Conclusões

Como indicado em nossa investigação, desigualdades válidas para o KP podem ser utilizadas em um algoritmo *Brach-and-Cut* para o QKP. Isto é feito com o intuito de reduzir o número de nós percorridos na árvore de enumeração do algoritmo *Branch-and-Bound* e, eventualmente, o tempo total de CPU.

Em nossos testes computacionais observamos que o uso das desigualdades (9) se torna computacionalmente caro, com o aumento da dimensão das instâncias. Observamos ainda um elevado número de nós são visitados nas árvore de enumeração, quando utilizamos tais desigualdades. Por sua vez, o uso das desigualdades de *cover*, *extended* e *lifting* se mostraram particularmente interessantes, em nossos testes computacionais, sobretudo as últimas.

Realizamos ainda testes computacionais utilizando as desigualdades de *cover*, *extended* e *lifting*, quadratizando-as. Entretanto, os resultados computacionais obtidos não se mostraram atraentes.

Referências

[1] Adams, W. P. e Sherali, H. D., A tight linearization and an algorithm for zero-one quadratic programming problems, Manage. Sci., 1986.

[2] Beasley, J., Lagrangian Reaxation, The Manegement School - Imperial College, Londo, 1992.

[3] Billionnet, A. e Soutif, É., An exact method based on Lagrangian decomposition for the 0-1 quadratic knapsack problem, European Journal of Operational Research, 2004.

[4] Billionnet, A., Faye, A. e Soutif, É., A new upper bound for the 0-1 quadratic knapsack problem, European Journal of Operational Research, 1999.

[5] Billionnet, A. e Calmels, F., Linear programming for the 0-1 quadratic knapsack problem, European J. Oper. Res, 1996.

[6] Billionnet, A. e Soutif, É., Using a Mixed Integer Programming Tool for Solving the 0-1 Quadratic Knapsack Problem, INFORMS J. on Computing, 2004.

[7] Caprara, A., Pisinger, D. e Toth, P., Exact Solution of the Quadratic Knapsack Problem, INFORMS J. on Computing, Vol.11, 1999.

[8] Chaillou, P., Hansen, P. e Mahieu, Y., Best network flow bound for the quadratic knapsack problem, Combinatorial Optimization, Lecture Notes in Mathematics, 1989.

[9] Crowder, H., Johnson, E. e Padberg, M.W., Solving large-scale 0-1 linear programming programs, Oper. Res., 1983.

[10] Escudero, L. e Guignard, M. e Malik, K., A Lagrangean relax and cut approach forthe sequential ordering with precedence constraints, Annals of Operations Research, 1994.

[11] Fair Isaac Corporation, FICO Xpress Optimization Suite, 2012.

[12] Gallo, G., Hammer, P. L. e Simeone, B., Quadratic knapsack problems, Mathematical Programming, 1980.

[13] Johnson, Ellis J. L., Mehrotra, Anuj e Nemhauser, George L., Min-cut clustering, Math. Program., 1993.

[14] Johnson, D.S. e Trick, M., Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, American Mathematical Society, 1996.

[15] Kellerer, H. e Pferschy, U. e Pisinger, D., Knapsack Problems, Springer, Berlin, Germany, 2004.

[16] Lucena, A., Steiner problem in graphs: Lagrangean relaxation and cutting-planes, COAL Bulletin, Mathematical Programming Society, 1992.

[17] Lucena, A., Non delayed relax-and-cut algorithms, Annals of Operations Research, 2005.

[18] Maculan, N. e Fampa, M., Otimização linear, Brasília: Editora da Universidade de Brasília, 2006.

[19] Martello, S. e Toth, P., Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., 1990.

[20] Michelon, P. e Veilleux, L., Lagrangean methods for the 0-1 quadratic knapsack problem, European J. Oper. Res, 1996.

[21] Padberg, M.W., The Boolean Quadratic Polytope: Some Characteristics, Facets and Relatives, Mathematical Programming, 1989.

[22] Palmeira, M. M., Um Algoritmo Relax-and-Cut para o Problema Quadrático da Mochila Binária, PUC, Rio de Janeiro, RJ, Brasil, 1999.

[23] Pisinger, D. e Toth, P., Knapsack problems, in: D. Du, P. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, 1998.

[24] Pisinger, D., The quadratic knapsack problem - a survey, Discrete Applied Mathematics, 2007.

[25] Rodrigues, C. D., Quadri, Q., Michelon, P. e Gueye, S., 0-1 Quadratic Knapsack Problems: An exact approach based on t-linearization, SIAM Journal on Optimization, 2012.

[26] Witzgall, C., Mathematical methods of site selection for electronic message system (EMS), Technical Report, NBS Internal Report, 1975.

[27] Wolsey, L. A., Integer Programming, Wiley-Interscience Publication, 1998.

[28] Wolsey, L.A., Faces for linear inequalities in 0-1 variables, Math. Program, 1975.

[29] Balas, E., The prize collecting traveling salesman problem, Networks, 1989.

[30] Elloumi, S., Faye, A. e Soutif, E., Decomposition and Linearization for 0-1 Quadratic Programming, Annals of Operations Research, 2000.