Progress and issues with SDP-based convexification and convex hull matheuristic for 0-1 quadratic NLPs

Monique Guignard-Spielberg (UPenn),
joint work with Lucas Letocart (Paris 13) and Michael Bussieck (GAMS)

MINLP Workshop -- CMU -- June 2014
We consider combinatorial optimization problems of the form

\[(Q01) \quad \text{Min} \quad q(x) = x^T Qx + c^T x \]
\[\text{s.t.} \quad Ax = b \]
\[A' x \leq b' \]
\[x \in \{0,1\}^n \]
The objective function is quadratic:

\[ q(x) = c^T x + x^T Qx \]

\[ = \sum_i c_i x_i + \sum_{i,j} x_i Q_{i,j} x_j \]

in addition to the individual costs \( c_i \), there are extra costs incurred when \( x_i \) and \( x_j \) are simultaneously equal to 1 (interaction costs).

For instance: correlated decisions (portfolio selection)  
traffic between locations (quadratic assignment)  
traffic inside a cross-dock depends on door assignments
Examples considered in this talk:

QAP or quadratic assignment problem (oldest, very hard)

GQAP or generalized quadratic assignment problem (just as hard or maybe harder)

CDAP or cross-dock assignment problem (a new problem in logistics, very hard too)

QKP and EkQKP or quadratic knapsack problems without or with a cardinality constraint.
FIRST PROBLEM: COMPUTING (IMPROVED) BOUNDS

- for nonconvex problems:
  - reformulation via linearization
  - via convexification

- for convex problems
  - improved bounds via convexification of the set of 0-1 feasible solutions
SIMILAR REFORMULATION TOOLS

(1) use the fact that $x_j \in \{0,1\} \Leftrightarrow (x_j)^2 = x_j$

(2) introduce new variables $y_{ij}$ (RLT) to replace the products $x_i x_j$, do not write $y_{ij} = x_i x_j$ because it is nonlinear, but add constraints that come from the definition of the new variables (y is viewed as a vector, like x):

- nonlinear equations:
  
  $y_{ij} = \min(x_i, x_j)$
  
  $y_{ij} = \max(0, x_i + x_j - 1)$

- or linear inequalities
  
  $y_{ij} \leq x_i, y_{ij} \leq x_j$
  
  $y_{ij} \geq 0, y_{ij} \geq x_i + x_j - 1$
(3) **introduce new variables** $X_{ij}$ (SDP) to replace the products $x_i x_j$, do not write $X_{ij} = x_i x_j$ because it is nonlinear, but add constraints that come from the definition of these new variables (X is viewed as a matrix)

defined by

$$X = \begin{pmatrix} x_1 x_1 & \cdots & x_1 x_n \\ \vdots & \ddots & \vdots \\ x_n x_1 & \cdots & x_n x_n \end{pmatrix} = xx^T$$

this implies that

- $X$ is of rank 1 (ignore)
- $X - xx^T = 0$ (still nonlinear !) therefore relaxed as $X - xx^T \succeq 0$ (nonlinear, or equivalently (*))

$$\begin{pmatrix} 1 \\ x \end{pmatrix}^T x^T \succeq 0$$

- $X_{ii} = (x_i)^2 = x_i$
(*) Schur’s lemma says that

\[
\begin{pmatrix} A & y \\ x & C \end{pmatrix} \succeq 0 \iff C - xA^{-1}y \succeq 0
\]
(4) **linearize the objective function:**

in RLT:

\[
 f(x) = x^T Q x = \sum_{i,j} x_i Q_{ij} x_j = \sum_{i,j} Q_{ij} x_i x_j = \sum_{i,j} Q_{ij} y_{ij}
\]

in SDP:

\[
 f(x) = x^T Q x = \sum_{i,j} x_i Q_{ij} x_j = \sum_{i,j} Q_{ij} x_i x_j = \sum_{i,j} Q_{ij} X_{ij} = Q \cdot X = \text{tr}(QX)
\]

(5) **add constraints that strengthen the continuous bound (RLT)**

multiply each previous constraint by each variable and by \((1\text{-that\
variable})\) and linearize:

\[
x_j \ast (A_k x = b_k) \Rightarrow \sum_i x_j A_{ki} x_i = b_k x_j \quad \Rightarrow \quad \sum_i A_{ki} y_{ij} = b_k x_j
\]
\((1-x_j)^* (A_k x = b_k) \Rightarrow \sum_i A_{ki} x_i - \sum_i x_j A_{ki} x_i = b_k - b_k x_j \Rightarrow \)

\[ \sum_i A_{ki} x_i + b_k x_j - \sum_i A_{ki} y_{ji} = b_k \]

one can also use symmetry:

\[ y_{ij} = y_{ji} \]

(6) the general idea is that the 0-1 models are “equivalent” in the following sense:

\[ \forall x \in FS(P), \exists y : \text{the o.f. are equal and } (x, y) \in FS(RLT) \]

and

\[ \forall (x, y) \in FS(RLT), \text{the o.f. are equal and } x \in FS(P) \]
(7) one can apply the same process “expanded” to the new set of variables and \textbf{iterate}:

\[
\nu(\overline{P}) \otimes \nu(\text{RLT-1}) \otimes \nu(\text{RLT-2}) \otimes \ldots \otimes \nu(\text{RLT-n}) = \nu(P)
\]

where $\otimes$ stands for "is at best as strong as"

(8) \textbf{reformulation by convexification of the objective:}

add to the objective function terms which are 0 for every 0-1 feasible solution of the problem and such that

- the expanded objective function is convex
- it yields the best continuous bound among all convex functions of the same type.
(9) if the objective function is convex, one may try to flatten it to improve the continuous bound.

(10) if the objective function is pseudoconvex, one can compute the continuous bound over the convex hull of all 0-1 feasible solutions, this is the convex hull relaxation of the original model, and this bound is at least as good as the continuous relaxation bound.
QUADRATIC CONVEX REFORMULATION (QCR)


---

**MQCR**


Convexify \( q(x) = x^T Q x \)

**First attempt: Hammer and Rubin**

If \( Q \) is not sdp, its smallest eigenvalue \( \mu \) is negative, add to \( Q \) a matrix \( |\mu - \varepsilon| I \) and subtract \( |\mu - \varepsilon| x \). The “new” matrix \( Q + |\mu - \varepsilon| I \) will have all its eigenvalues positive and the new objective function will be equal to the old one for any binary vector \( x \). This amounts to changing all diagonal elements by the same amount \( (-\mu + \varepsilon) \).

**Second attempt: Billionnet and Elloumi.** Add different amounts to all diagonal elements of \( Q \), call \( u_i \) the change in \( Q_{ii} \), and choose the vector \( u \) to get the best lower bound among all possible \( u \)’s such that \( Q + u I \) is psd: this is an SDP problem.
Third attempt: MC Plateau. Add to the previous change a multiple of $x_j (A_i x-b_i)$ where $A_i x=b_i$ is the $i$th equality constraint. Let the coefficient be $\alpha_{ij}$, chosen as to yield the best lower bound among all new psd quadratic matrices.

Fourth attempt: Ji, Zheng and Sun (2011)

In addition, enforce some of the conditions that make $X_{ij}$ more like the product of the 0-1 variables $x_i$ and $x_j$. Indeed, without that, one gets $X_{ij}$ different from $x_i x_j$. 
<table>
<thead>
<tr>
<th>Instance</th>
<th>with u</th>
<th>CSDP sec.</th>
<th>with uv</th>
<th>CSDP sec.</th>
<th># 0-1 vars</th>
<th>number of $X(i,j,ip,jp) &gt; x(i,j)$ with u</th>
<th>%</th>
<th>number of $X(i,j,ip,jp) &gt; x(i,j)$ with uv</th>
<th>%</th>
<th>number of $X(i,j,ip,jp) &lt; \max(0,x.l(i,j)+x(ij,ip)-1)$</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x7</td>
<td>-3687670</td>
<td>0.851</td>
<td>-648309</td>
<td>1.013</td>
<td>112</td>
<td>1611</td>
<td>13%</td>
<td>1257</td>
<td>10%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>20x10</td>
<td>-2218027</td>
<td>1.912</td>
<td>-1676906</td>
<td>2.634</td>
<td>200</td>
<td>10261</td>
<td>26%</td>
<td>9885</td>
<td>25%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>20-15-35</td>
<td>-4003849</td>
<td>3.974</td>
<td>-3931993</td>
<td>5.03</td>
<td>300</td>
<td>22351</td>
<td>25%</td>
<td>22337</td>
<td>25%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>20-15-55</td>
<td>-3438241</td>
<td>3.714</td>
<td>-3418839</td>
<td>5.442</td>
<td>300</td>
<td>13226</td>
<td>15%</td>
<td>13476</td>
<td>15%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>30-06-95</td>
<td>237147</td>
<td>1.435</td>
<td>353601</td>
<td>3.086</td>
<td>180</td>
<td>7999</td>
<td>25%</td>
<td>7529</td>
<td>23%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>30-07-75</td>
<td>-3384130</td>
<td>1.749</td>
<td>-2441652</td>
<td>3.805</td>
<td>210</td>
<td>11132</td>
<td>25%</td>
<td>11128</td>
<td>25%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>50-10-95</td>
<td>-14369254</td>
<td>12.2</td>
<td>-13468819</td>
<td>73.6</td>
<td>500</td>
<td>40088</td>
<td>16%</td>
<td>41286</td>
<td>17%</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>moccia60x40</td>
<td>-153320314</td>
<td>1489</td>
<td>-153320271</td>
<td>1652</td>
<td>2400</td>
<td>1345008</td>
<td>23%</td>
<td>1344976</td>
<td>23%</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>
M (for Matrix) QCR

Considerons first the case of a quadratic problem without equality constraints.

(QP) \( \max q(x) = x^T Q x \)

s.t. \( Ax \leq b \)

\( x \in \{0,1\}^n \)

Rewrite \( q(x) = x^T Q x = x^T (Q - M) x + x^T Mx \)

where \( M \) is a matrix such that \( Q - M \preceq 0 \)

and \( M = \text{Diag}(\rho) + P + N \in S_n \)

with \( P \in P \) and \( N \in N \).
The following always hold for $x \in \{0,1\}^n$:

$$x_i x_j = \min(x_i, x_j) \text{ and } x_i x_j = \max(0, x_i + x_j - 1)$$

thus

$$q(x) = x^T [Q - \text{Diag}(\rho) - P - N] x + x^T [\text{Diag}(\rho) + P + N] x$$

$$= x^T [Q - \text{Diag}(\rho) - P - N] x + \rho^T x + \sum_{i,j} [P_{ij} x_i x_j + N_{ij} x_i x_j]$$

$$= x^T [Q - \text{Diag}(\rho) - P - N] x + \rho^T x + \sum_{i,j} [P_{ij} s_{ij} - N_{ij} t_{ij}]$$

where $s_{ij} = \min(x_i, x_j)$ and $t_{ij} = -\max(0, x_i + x_j - 1)$. 
(QP) can be rewritten as (QP(\(\rho\), \(P\), \(N\))):

Max \( q(x) = x^T[Q - \text{Diag}(\rho) - P - N]x + \rho^Tx + \sum_{i,j}[P_{ij}s_{ij} - N_{ij}t_{ij}] \)

s.t. \( Ax \leq b \),
\( x \in \{0,1\}^n \)
\( s_{ij} = \min(x_i, x_j) \) and \( t_{ij} = -\max(0, x_i + x_j - 1) \)

where \( Q - \text{Diag}(\rho) - P - N \leq 0 \), \( P \geq 0 \) and \( N \leq 0 \).

\( s_{ij} = \min(x_i, x_j) \) and \( t_{ij} = -\max(0, x_i + x_j - 1) \) can be replaced by
\( s_{ij} \leq x_i \) and \( s_{ij} \leq x_j \) and \( t_{ij} \leq 0 \) and \( t_{ij} \leq 1 - x_i - x_j \).
(QP) can finally be rewritten as

\[
\begin{align*}
\text{(QP}(\rho, P, N)):
\text{Max} & \quad q(x) = x^T [Q - \text{Diag}(\rho) - P - N] x + \rho^T x + \sum_{i,j} [P_{ij}s_{ij} - N_{ij}t_{ij}] \\
\text{s.t.} & \quad Ax \leq b \\
& \quad s_{ij} \leq x_i, \quad s_{ij} \leq x_j, \\
& \quad t_{ij} \leq 0, \quad t_{ij} \leq 1 - x_i - x_j. \\
& \quad x \in \{0,1\}^n
\end{align*}
\]

where \( Q - \text{Diag}(\rho) - P - N \preceq 0 \), \( P \succeq 0 \) and \( N \preceq 0 \).
The continuous relaxation of \((\text{QP}(\rho, P, N))\) is \((\overline{\text{QP}}(\rho, P, N))\):

\[
\begin{align*}
\text{Max} \quad q(x) &= x^T [Q - \text{Diag}(\rho) - P - N] x + \rho^T x + \sum_{i,j} [P_{ij}s_{ij} - N_{ij}t_{ij}] \\
\text{s.t.} \quad Ax &\leq b \\
&\quad s_{ij} \leq x_i, \quad s_{ij} \leq x_j, \\
&\quad t_{ij} \leq 0, \quad t_{ij} \leq 1 - x_i - x_j, \\
&\quad 0 \leq x \leq e
\end{align*}
\]

and one wants to determine the values of \(\rho, P\) and \(N\) that will minimize its optimal value, which is an upper bound on \(v(\text{QP})\):

\[
\begin{align*}
(\text{UB}) \quad \text{Min} \quad v(\overline{\text{QP}}(\rho, P, N)) \\
\text{s.t.} \quad Q - \text{Diag}(\rho) - P - N &\preceq 0 \\
\rho &\in \mathbb{R}^n, \ P \geq 0 \text{ and } N \preceq 0.
\end{align*}
\]
(UB) can be solved using any SDP solver.
Extension to the case of equality constraints.

Assume that there is also a linear equality constraint

\[ Fx = d, \]

then one can extend the analysis and convexify the problem as follows.
The extended SDP model is

\[ \text{max } Q.X \]
\[ \text{s.t. } X_{ii} = x_i, \quad i = 1, \ldots, n \quad (1) \]
\[ X_{ij} \leq x_i, \quad X_{ij} \leq x_j, \quad i, j = 1, \ldots, n \quad (2) \]
\[ X_{ij} \geq x_i + x_j - 1, \quad X_{ij} \geq 0, \quad i, j = 1, \ldots, n \quad (3) \]
\[ Ax \leq b \quad (4) \]
\[ Fx = d \quad (5) \]
\[ \sum_j F_{kj} X_{ij} = d_k x_i, \quad i, j = 1, \ldots, n, \quad k = 1, \ldots, m, \quad (6) \]
\[ 0 \leq x \leq e \quad (7) \]
\[ \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0 \quad (8) \]
The convexified function will be

\[ q_{P,N,u,v}(x) = x^T (Q-P-N) x + \sum_i \rho_i (x_i^2 - x_i) + \sum_k \alpha_{ki} x_i (F_k x - d_k) \]

or, in a cheaper version,

\[ q_{P,N,u,v}(x) = x^T (Q-P-N) x + \sum_i \rho_i (x_i^2 - x_i) + \nu \|Fx - d\|^2 \]

\[ + \sum_{i,j} (P_{ij} s_{ij} - N_{ij} t_{ij}) \]

with \( P = B+C \) and \( N = -(D+E) \),
and the convexified model’s constraints will be

\[ Ax \leq b \]

\[ Fx = d \]

\[ s_{ij} \leq x_i, \quad s_{ij} \leq x_j, \quad i, j=1,\ldots,n \]

\[ t_{ij} \leq 0, \quad t_{ij} \leq 1 - x_i - x_j, \quad i, j=1,\ldots,n \]

\[ x_{ij} \in \{0,1\}. \]
Columns 3-5 and Columns 7-9 of the tables record the average CPU time, average number of
nodes explored and the average relative errors of the two reformulations.

From Tables 1-4, we see that the improved reformulation \((QKP(\rho^*, P^*, N^*))\) is much more efﬁcient than the diagonal perturbed reformulation \((QKP(\bar{\rho}))\) for the randomly generated instances of \((QKP)\) in terms of the CPU time, number of nodes and relative errors. For both relaxation options in the MIQP solver, the number of nodes explored by the MIQP solver for \((QKP(\rho^*, P^*, N^*))\) is much less than that for \((QKP(\bar{\rho}))\). This is mainly because the continuous relaxations of the subproblems of \((QKP(\rho^*, P^*, N^*))\) are tighter than the subproblems of \((QKP(\bar{\rho}))\) during the branch-and-bound process. We also see that the relative error achieved in solving \((QKP(\rho^*, P^*, N^*))\) is much smaller than that of \((QKP(\bar{\rho}))\) when the branch-and-bound method is terminated with the 1800 seconds maximum CPU time. We observe that the CPU time for computing the parameters of \((QKP(\rho^*, P^*, N^*))\) is longer than that of \((QKP(\bar{\rho}))\). However, this additional CPU time is neglectable when compared the total CPU time used by the branch-and-bound method.

Table 1: Comparison results for \((QKP)\) with LP relaxation \((m = 1)\)

<table>
<thead>
<tr>
<th>(n)</th>
<th>((DPR_d))</th>
<th>((QKP(\bar{\rho})))</th>
<th>((TUB_d))</th>
<th>((QKP(\rho^<em>, P^</em>, N^*)))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>time</td>
<td>nodes</td>
<td>rel.error(%)</td>
</tr>
<tr>
<td>60</td>
<td>1.00</td>
<td>613.33</td>
<td>1467872</td>
<td>1.46</td>
</tr>
<tr>
<td>70</td>
<td>1.60</td>
<td>1638.20</td>
<td>3718590</td>
<td>3.60</td>
</tr>
<tr>
<td>80</td>
<td>1.78</td>
<td>1747.67</td>
<td>3975569</td>
<td>7.33</td>
</tr>
<tr>
<td>90</td>
<td>2.00</td>
<td>1800.00</td>
<td>3311959</td>
<td>2.67</td>
</tr>
<tr>
<td>100</td>
<td>2.37</td>
<td>1800.00</td>
<td>3581474</td>
<td>20.00</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>14.00</td>
<td>9429</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>6.60</td>
<td>119.20</td>
<td>12009</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>9.00</td>
<td>585.33</td>
<td>402603</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>12.00</td>
<td>686.67</td>
<td>277329</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>13.00</td>
<td>801.00</td>
<td>128992</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Table 2: Comparison results for \((QKP)\) with continuous relaxation \((m = 1)\)

<table>
<thead>
<tr>
<th>(n)</th>
<th>((DPR_d))</th>
<th>((QKP(\bar{\rho})))</th>
<th>((TUB_d))</th>
<th>((QKP(\rho^<em>, P^</em>, N^*)))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>time</td>
<td>nodes</td>
<td>rel.error(%)</td>
</tr>
<tr>
<td>60</td>
<td>1.00</td>
<td>1800.00</td>
<td>71350</td>
<td>5.33</td>
</tr>
<tr>
<td>70</td>
<td>1.60</td>
<td>1800.00</td>
<td>68077</td>
<td>8.40</td>
</tr>
<tr>
<td>80</td>
<td>1.78</td>
<td>1800.00</td>
<td>76147</td>
<td>21.67</td>
</tr>
<tr>
<td>90</td>
<td>2.00</td>
<td>1800.00</td>
<td>35954</td>
<td>6.00</td>
</tr>
<tr>
<td>100</td>
<td>2.37</td>
<td>1800.00</td>
<td>39933</td>
<td>31.67</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>822.67</td>
<td>4784</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>6.60</td>
<td>1694.20</td>
<td>7643</td>
<td>1.20</td>
</tr>
<tr>
<td></td>
<td>9.00</td>
<td>1659.00</td>
<td>7395</td>
<td>2.63</td>
</tr>
<tr>
<td></td>
<td>12.00</td>
<td>1800.00</td>
<td>4930</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>13.00</td>
<td>1800.00</td>
<td>4427</td>
<td>5.33</td>
</tr>
</tbody>
</table>

Tables 5-6 summarize the numerical results of the MIQP solver in CPLEX 12.1 with LP relaxation and continuous relaxation respectively for the two reformulations of \((k-QKP)\). We observe that the average performance of the improved reformulation \((k-QKP(\rho^*, \alpha^*, P^*, N^*))\) is slightly better than the reformulation \((k-QKP(\bar{\rho}, \bar{\alpha}))\) in terms of the CPU time, the number of nodes and the
The experiments were performed on a collection of QKP instances provided by Billionnet and Soutif (http://cedric.cnam.fr/~soutif/QKP/).

The comparison results between the two reformulations (QKP(\(\bar{\rho}\))) and (QKP(\(\rho^*, P^*, N^*\))), when using MIQP solver in CPLEX 12.1 with continuous relaxation, are summarized in Tables 1-2, where

- Columns 2 and 6 of the tables are the CPU time for computing the parameters by solving the SDP problems (DPR\(_d\)) and (TUB\(_d\)), respectively;
- Columns 3-5 and Columns 7-9 of the tables record the average CPU time, average number of nodes explored and the average relative errors of the two reformulations.

Please refer to the attached PDF file about the details.
Table 1: Comparison results for (QKP) with continuous relaxation

<table>
<thead>
<tr>
<th>Instance</th>
<th>(DPRd)</th>
<th>(QKP(ρ))</th>
<th>(TUBd)</th>
<th>(QKP(ρ*, P*, N*))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>time</td>
<td>nodes</td>
<td>rel.error(%)</td>
</tr>
<tr>
<td>100_100_1</td>
<td>3</td>
<td>3600</td>
<td>39187</td>
<td>12.94</td>
</tr>
<tr>
<td>100_100_2</td>
<td>3</td>
<td>3600</td>
<td>40160</td>
<td>4.82</td>
</tr>
<tr>
<td>100_100_3</td>
<td>2</td>
<td>3600</td>
<td>39392</td>
<td>1.20</td>
</tr>
<tr>
<td>100_100_4</td>
<td>3</td>
<td>3600</td>
<td>48886</td>
<td>17.15</td>
</tr>
<tr>
<td>100_100_5</td>
<td>3</td>
<td>3600</td>
<td>39526</td>
<td>1.04</td>
</tr>
<tr>
<td>100_100_6</td>
<td>2</td>
<td>3600</td>
<td>45209</td>
<td>22.69</td>
</tr>
<tr>
<td>100_100_7</td>
<td>5</td>
<td>3600</td>
<td>70218</td>
<td>49.39</td>
</tr>
<tr>
<td>100_100_8</td>
<td>2</td>
<td>3600</td>
<td>52620</td>
<td>25.04</td>
</tr>
<tr>
<td>100_100_9</td>
<td>6</td>
<td>3600</td>
<td>38654</td>
<td>1.18</td>
</tr>
<tr>
<td>100_100_10</td>
<td>3</td>
<td>3600</td>
<td>36432</td>
<td>3.47</td>
</tr>
<tr>
<td>100_75_1</td>
<td>4</td>
<td>16</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>100_75_2</td>
<td>3</td>
<td>3600</td>
<td>52415</td>
<td>13.32</td>
</tr>
<tr>
<td>100_75_3</td>
<td>4</td>
<td>3600</td>
<td>19704</td>
<td>19.19</td>
</tr>
<tr>
<td>100_75_4</td>
<td>2</td>
<td>3600</td>
<td>24415</td>
<td>20.74</td>
</tr>
<tr>
<td>100_75_5</td>
<td>2</td>
<td>3600</td>
<td>77056</td>
<td>35.02</td>
</tr>
<tr>
<td>100_75_6</td>
<td>3</td>
<td>3600</td>
<td>13380</td>
<td>3.08</td>
</tr>
<tr>
<td>100_75_7</td>
<td>3</td>
<td>3600</td>
<td>20513</td>
<td>9.91</td>
</tr>
<tr>
<td>100_75_8</td>
<td>2</td>
<td>3600</td>
<td>66529</td>
<td>53.50</td>
</tr>
<tr>
<td>100_75_9</td>
<td>3</td>
<td>3600</td>
<td>21753</td>
<td>11.14</td>
</tr>
<tr>
<td>100_75_10</td>
<td>3</td>
<td>3600</td>
<td>36064</td>
<td>3.50</td>
</tr>
</tbody>
</table>

The experiments were performed on a collection of QKP instances provided by Billionnet and Soutif (http://cedric.cnam.fr/~soutif/QKP/).

The comparison results between the two reformulations (QKP(ρ)) and (QKP(ρ*, P*, N*)), when using MIQP solver in CPLEX 12.1 with continuous relaxation, are summarized in Tables 1-2, where

- Columns 2 and 6 of the tables are the CPU time for computing the parameters by solving the SDP problems (DPRd) and (TUBd), respectively;
- Columns 3-5 and Columns 7-9 of the tables record the average CPU time, average number of nodes explored and the average relative errors of the two reformulations.

Please refer to the attached PDF file about the details.
<table>
<thead>
<tr>
<th>Instance</th>
<th>(DPR$_d$)</th>
<th>(QKP($\bar{\rho}$))</th>
<th>(TUB$_d$)</th>
<th>(QKP($\rho^<em>$, $P^</em>$, $N^*$))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>time</td>
<td>nodes</td>
<td>rel.error(%)</td>
</tr>
<tr>
<td>100,50,1</td>
<td>5</td>
<td>3600</td>
<td>37773</td>
<td>5.39</td>
</tr>
<tr>
<td>100,50,2</td>
<td>3</td>
<td>3600</td>
<td>18774</td>
<td>3.08</td>
</tr>
<tr>
<td>100,50,3</td>
<td>3</td>
<td>3600</td>
<td>29594</td>
<td>19.77</td>
</tr>
<tr>
<td>100,50,4</td>
<td>3</td>
<td>3600</td>
<td>19298</td>
<td>2.45</td>
</tr>
<tr>
<td>100,50,5</td>
<td>3</td>
<td>3600</td>
<td>23087</td>
<td>11.49</td>
</tr>
<tr>
<td>100,50,6</td>
<td>3</td>
<td>3600</td>
<td>31420</td>
<td>45.55</td>
</tr>
<tr>
<td>100,50,7</td>
<td>3</td>
<td>3600</td>
<td>25417</td>
<td>14.83</td>
</tr>
<tr>
<td>100,50,8</td>
<td>3</td>
<td>3600</td>
<td>22136</td>
<td>12.17</td>
</tr>
<tr>
<td>100,50,9</td>
<td>3</td>
<td>3600</td>
<td>22426</td>
<td>10.01</td>
</tr>
<tr>
<td>100,50,10</td>
<td>3</td>
<td>3600</td>
<td>32801</td>
<td>5.64</td>
</tr>
<tr>
<td>200,100,1</td>
<td>16</td>
<td>3600</td>
<td>3965</td>
<td>1.81</td>
</tr>
<tr>
<td>200,100,2</td>
<td>15</td>
<td>3600</td>
<td>5054</td>
<td>24.19</td>
</tr>
<tr>
<td>200,100,3</td>
<td>14</td>
<td>3600</td>
<td>7091</td>
<td>116.26</td>
</tr>
<tr>
<td>200,100,4</td>
<td>13</td>
<td>3600</td>
<td>6691</td>
<td>103.90</td>
</tr>
<tr>
<td>200,100,5</td>
<td>13</td>
<td>3600</td>
<td>3869</td>
<td>6.32</td>
</tr>
<tr>
<td>200,100,6</td>
<td>14</td>
<td>3600</td>
<td>7050</td>
<td>102.40</td>
</tr>
<tr>
<td>200,100,7</td>
<td>13</td>
<td>3600</td>
<td>3776</td>
<td>73.9</td>
</tr>
<tr>
<td>200,100,8</td>
<td>14</td>
<td>3600</td>
<td>3616</td>
<td>6.23</td>
</tr>
<tr>
<td>200,100,9</td>
<td>13</td>
<td>3600</td>
<td>3926</td>
<td>10.10</td>
</tr>
<tr>
<td>200,100,10</td>
<td>13</td>
<td>3600</td>
<td>4711</td>
<td>23.25</td>
</tr>
<tr>
<td>300,50,1</td>
<td>46</td>
<td>3600</td>
<td>2583</td>
<td>10.08</td>
</tr>
<tr>
<td>300,50,2</td>
<td>49</td>
<td>3600</td>
<td>1987</td>
<td>117.78</td>
</tr>
<tr>
<td>300,50,3</td>
<td>47</td>
<td>3600</td>
<td>1269</td>
<td>8.63</td>
</tr>
<tr>
<td>300,50,4</td>
<td>45</td>
<td>3600</td>
<td>1675</td>
<td>8.31</td>
</tr>
<tr>
<td>300,50,5</td>
<td>54</td>
<td>3600</td>
<td>1679</td>
<td>3.99</td>
</tr>
</tbody>
</table>
Table 6: Comparison results for (k-QKP) with continuous relaxation ($m = 1$, $K = \lfloor \frac{n}{8} \rfloor$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>(k-DPR$_d$)</th>
<th>(k-QKP($\bar{\rho}, \bar{\alpha}$))</th>
<th>(k-TUB$_d$)</th>
<th>(k-QKP($\rho^<em>, \alpha^</em>, P^<em>, N^</em>$))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time (s)</td>
<td>time (s)</td>
<td>nodes</td>
<td>rel.error(%)</td>
</tr>
<tr>
<td>60</td>
<td>1.00</td>
<td>1800.00</td>
<td>69836</td>
<td>5.84</td>
</tr>
<tr>
<td>70</td>
<td>2.00</td>
<td>1800.00</td>
<td>53224</td>
<td>8.90</td>
</tr>
<tr>
<td>80</td>
<td>2.00</td>
<td>1800.00</td>
<td>44465</td>
<td>13.05</td>
</tr>
<tr>
<td>90</td>
<td>2.20</td>
<td>1800.00</td>
<td>38035</td>
<td>13.52</td>
</tr>
<tr>
<td>100</td>
<td>3.00</td>
<td>1800.00</td>
<td>29276</td>
<td>15.74</td>
</tr>
</tbody>
</table>

have shown that the optimal parameters in the improved reformulation can be obtained by solving an SDP problem. Comparison numerical results suggest that the improved reformulation is more efficient than the existing reformulation based on diagonal perturbation in terms of the CPU time, the number of nodes explored and the relative errors when the MIQP solver in CPLEX 12.1 is used to solve the reformulations.

We point that the matrix decomposition method can be extended to any 0-1 quadratic program. Although the continuous relaxation of the new reformulation is always tighter than or at least as tight as that of the diagonal perturbed reformulation, the size of the new reformulation also increases significantly. There is a trade-off between the quality of the bound and its computation time in a branch-and-bound method. For this reason, we can use partial elements of the matrix $P$ and $N$ when constructing the new reformulation, as is the case in our numerical experiments.

References


For 0-1 variables:

\[ x_i x_j = \frac{1}{2}(x_i + x_j)^2 - \frac{1}{2} x_i^2 - \frac{1}{2} x_j^2 = \frac{1}{2}(x_i + x_j)^2 - \frac{1}{2} x_i - \frac{1}{2} x_j \]

thus one can render the objective function convex.

Bound quality (all negative!):

<table>
<thead>
<tr>
<th></th>
<th>DJWhite</th>
<th>cplex 12.4</th>
<th>u</th>
<th>uv</th>
<th># 0-1 vars</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x7</td>
<td>-11,949,628</td>
<td>-3,882,635</td>
<td>-3,687,670</td>
<td>-648,099</td>
<td>112</td>
</tr>
<tr>
<td>20x10</td>
<td>-13,133,687</td>
<td>-2,908,816</td>
<td>-2,218,027</td>
<td>-1,676,910</td>
<td>200</td>
</tr>
<tr>
<td>27x21</td>
<td>-68,190,327</td>
<td>-11,643,384</td>
<td>-9,204,537</td>
<td>---</td>
<td>567</td>
</tr>
</tbody>
</table>
FLATTENING A CONVEX FUNCTION OF 0-1 VARIABLES

An Other Bounding Approach:

The CHR (Convex Hull Relaxation) Algorithm

(For Pseudoconvex MINLP Problems With Linear Constraints)
Based on Primal Relaxation For Nonlinear Integer Problems

(Guignard 1994)

For a nonlinear integer programming problem

\[
(P) \quad \text{Min}_x \{ f(x) \mid Ax = b, Cx \leq d, x \in X \}
\]

with a nonlinear (pseudo-)convex function \( f(x) \), one can construct a primal relaxation

\[
(PR) \quad \text{Min}_x \{ f(x) \mid Ax = b, x \in \text{Co}\{ x \in X \mid Cx \leq d \} \}.
\]

The constraint set is a polyhedron, and the objective function is still (pseudo-)convex.
Convex hull relaxation (CHR) for NLIP problems

(Albornoz 1998) (Ahlatcioglu, Guignard 2007)

For a nonlinear integer programming problem

\[(P) \quad \text{Min}_x \{ f(x) \mid x \in X \}\]

with a nonlinear convex function \(f(x)\), and \(X\) such that \(\text{Co}(X)\) is a polyhedron.

One can construct a **primal relaxation**

\[(PR) \quad \text{Min}_x \{ f(x) \mid x \in \text{Co}(X) \}\]

The constraint set is a polyhedron, and the objective function is still the same.
Special case: The CHR relaxation for linear constraints

\[(NLIP) \quad \min \{ f(x) \mid x \in Y, \ Ax = b \}\]

where \(f(x)\) is a nonlinear convex function of \(x \in \mathbb{R}^n\),
\(A\) is an \(m \times n\) constraint matrix, \(b\) is a resource vector in \(\mathbb{R}^m\),
\(Y\) is a subset of \(\mathbb{R}^n\) specifying integrality restrictions on \(x\).

The Convex Hull Relaxation of (NLIP) is

\[(CHR) \quad \min \{ f(x) \mid x \in \text{Co}\{x \in Y \mid Ax = b\}\}\].

This relaxation is a primal relaxation, in the \(x\)-space. It is actually a relaxation that does not “relax” any true constraint.
(CHR) looks difficult because of the implicit formulation of the convex hull.

However one can solve the problem exactly by decomposing it into a sub-problem and a master problem,

(same as in Michelon and Maculan (1992) for the linear case of Lagrangean relaxation).

Possible methods:

Frank & Wolfe (1956),
Von Hohenbalken’s Simplicial Decomposition (1977)
Hearn et al. ‘s Restricted SD (1987)
We use *restricted* simplicial decomposition.

To guarantee a *global optimal solution* of (CHR), several conditions must be satisfied:

(i) the feasible region must be *compact and convex*,

(ii) the objective function must be *(pseudo-*)convex, and

(iii) the constraints must be *linear*.

We solve a sequence of *LIP subproblems* and *NLP master problems*. 
LOWER BOUND:

Simplicial Decomposition finds an optimal solution $x^*$ to (CHR).

This provides a lower bound on $v$(NLIP):

$$\text{LB}_{CHR} = f(x^*).$$

UPPER BOUND

At each iteration $k$ of the subproblem, an extreme point, $y^*_k$, of the convex hull is found. It is an integer feasible point of (NLIP).

Each point $y^*_k$ yields an Upper Bound (UB) on the optimal value of (NLIP), and the best upper bound on $v$(NLIP) is

$$\text{UB}_{CHR} = \min f(y^*_1), f(y^*_2), \ldots, f(y^*_k).$$
USED AS A HEURISTIC FOR NONCONVEX PROBLEMS:

multistart. Use different initial linearization points.

For CDAP: from 2 real crossdocks

For GQAP: with original data from Lee and Ma, and Moccia

For QAP: QAPLib problems

Overall results: very fast. Average gap for known solutions less than 1%.
Example: Nonconvex GQAP
<table>
<thead>
<tr>
<th>Instance</th>
<th># 0-1 vars</th>
<th>OV</th>
<th>% gap</th>
<th>CH heuristic</th>
<th>Time (MIP, NLP, total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pb16x7</td>
<td>112</td>
<td>2809870</td>
<td>0</td>
<td>2809870</td>
<td>9,2,11</td>
</tr>
<tr>
<td>pb-20-15-35</td>
<td>300</td>
<td>1471896</td>
<td>0</td>
<td>1471896</td>
<td>5,2,7</td>
</tr>
<tr>
<td>pb-20-15-55</td>
<td>300</td>
<td>1723638</td>
<td>0</td>
<td>1723638</td>
<td>9,6,15</td>
</tr>
<tr>
<td>pb-20-15-75</td>
<td>300</td>
<td>1953188</td>
<td>0</td>
<td>1953188</td>
<td>11,3,14</td>
</tr>
<tr>
<td>pb-30-06-95</td>
<td>180</td>
<td>5160920</td>
<td>0</td>
<td>5160920</td>
<td>11,6,17</td>
</tr>
<tr>
<td>pb-30-07-75</td>
<td>210</td>
<td>4383923</td>
<td>0</td>
<td>4383923</td>
<td>17,4,21</td>
</tr>
<tr>
<td>pb-30-08-55</td>
<td>240</td>
<td>3501695</td>
<td>0</td>
<td>3501695</td>
<td>3,2,5</td>
</tr>
<tr>
<td>pb-30-10-65</td>
<td>300</td>
<td>3620959</td>
<td>0</td>
<td>3620959</td>
<td>17,5,23</td>
</tr>
<tr>
<td>pb-30-20-35</td>
<td>600</td>
<td>3379359</td>
<td>0</td>
<td>3379359</td>
<td>16,6,22</td>
</tr>
<tr>
<td>pb-30-20-55</td>
<td>600</td>
<td>3593105</td>
<td>0</td>
<td>3593105</td>
<td>15,5,20</td>
</tr>
<tr>
<td>pb-30-20-75</td>
<td>600</td>
<td>4050938</td>
<td>0</td>
<td>4050938</td>
<td>11,3,14</td>
</tr>
<tr>
<td>pb-30-20-95</td>
<td>600</td>
<td>5710645</td>
<td>0.28</td>
<td>5726530</td>
<td>219,2,222</td>
</tr>
<tr>
<td>pb-35-15-35</td>
<td>525</td>
<td>4456670</td>
<td>0</td>
<td>4456670</td>
<td>16,8,23</td>
</tr>
<tr>
<td>pb-35-15-55</td>
<td>525</td>
<td>4639128</td>
<td>0</td>
<td>4639128</td>
<td>14,8,22</td>
</tr>
<tr>
<td>pb-35-15-75</td>
<td>525</td>
<td>6301723</td>
<td>0</td>
<td>6301723</td>
<td>16,9,26</td>
</tr>
<tr>
<td>pb-35-15-95</td>
<td>525</td>
<td>6670264</td>
<td>0</td>
<td>6670264</td>
<td>334,13,347</td>
</tr>
</tbody>
</table>
Application: Crossdocking

Inbound trucks

Destinations

There is a transfer cost associated with every receiving/des- tination transfer pair.
Formulation of the CDAP

**Parameters**

- \( M \) - number of suppliers.
- \( N \) - number of customers.
- \( I \) - number of strip doors.
- \( J \) - number of stack doors.
- \( f_{mn} \) - the number of trips required by the material handling equipment to move items originating from supplier \( m \) to the stack door where freight destined for customer \( n \) is being consolidated.
- \( d_{ij} \) - distance between strip door \( i \) and stack door \( j \).

**Decision Variables**

- \( x_{mi} = 1 \) if supplier \( m \) is assigned to strip door \( i \), \( x_{mi} = 0 \) otherwise.
- \( y_{nj} = 1 \) if customer \( n \) is assigned to stack door \( j \), \( y_{nj} = 0 \) otherwise.
Formulation of the CDAP

\[
\min \left\{ \text{cost} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{m=1}^{M} \sum_{n=1}^{N} d_{ij} f_{mn} x_{mi} y_{nj} \right\}
\]

s.t.

\[\sum_{i=1}^{I} x_{mi} = 1 \text{ for all } m\] - each inbound truck is assigned one strip door

\[\sum_{m=1}^{M} s_{m} x_{mi} \leq S_i \text{ for all } i\] - capacity of strip door not exceeded

\[\sum_{j=1}^{J} y_{nj} = 1 \text{ for all } n\] - each destination is assigned one stack door

\[\sum_{n=1}^{N} r_{n} y_{nj} \leq R_j \text{ for all } j\] - capacity of stack door not exceeded

\[x_{mi}, y_{nj} \text{ are either 0 or 1}\]
CDAP DEMONSTRATION

Current operations

Modified operations
• For January 2012, the total daily cost could be reduced by utilizing the CDAP algorithm to assign the doors. By extrapolation on average, the CDAP cost is 11% lower than the Base Case cost, which translates to $10,000 of cost savings per month.

* The algorithm is very fast. It takes less than 50 seconds on a laptop to solve a practical instance with 56 incoming trucks, 16 destinations, 41 inbound and 16 outbound doors. It can be used real time in the crossdock.
Thank You!