Evangelos Panos :: Energy Economics Group :: Paul Scherrer Institute

CPLEX\Barrier options for TIMES models

ETSAP Webinar, 11th March 2022
Model sizes increase as more complex questions seek for answers.

- 1970: Accounting frameworks
- 1980: Process-oriented models
- 1990: Energy-economy coupling models, with micro-economic foundations and high technical detail
- 2000: Integrated assessment models with high temporal, technical and spatial resolution
- 2010: Integrated assessment models
- 2020: Socio-Energy-economy-climate models at high temporal, spatial, technical, resolutions

Problem focus:

- 1970: Energy balance frameworks
- 1980: Energy markets and economy
- 1990: Climate change mitigation
- 2000: Consumers & sector coupling

Solution time tends also to exponentially increase as policy questions become more complex.
What will be covered in this webinar:
• Insights on CPLEX\Barrier
• Key CPLEX\Barrier options to reduce solution times with examples
• When to use an advanced basis in CPLEX\Simplex

What will not be covered in this webinar:
• Numerical difficulties and instability
• Ill conditioning
• Degeneracy
• Assessing the quality of the solution obtained
A few words about the CPLEX\Barrier algorithm

with red are related CPLEX parameters
(click to go to the corresponding section in the manual)
The constraints define a polyhedron in $n$ dimensions (shaded area is the feasible area)

- If a solution exists, it will be at a vertex

- Simplex determines the edge to be followed next towards the optimal vertex

- Barrier goes through the inside of the feasible space with a combination of:
  - an optimization term to move towards the objective (predictor or growth)
  - a centering term or constraint to keep away from the boundaries (corrector)

Source of the figure: [https://ibmdecisionoptimization.github.io/tutorials/html/Linear_Programming.html](https://ibmdecisionoptimization.github.io/tutorials/html/Linear_Programming.html)
Its performance is affected by:

- The presence of \textit{dense} columns
- The total non-zeros in factor
- The total FP operations to factor
- The number of nonzeros in lower triangle of $A^*A'$

Sample output from the CPLEX\Bar\
rier log file:

\begin{verbatim}
CPXPARAM_BARRIER_ALGORITHM 1
CPXPARAM_BARRIER_ORDERING 3
CPXPARAM_BARRIER_LIMITS_ITERATION 999999
CPXPARAM_TIME_LIMIT 500000
CPXPARAM_TUNED_TIME_LIMIT 100000

Tried aggregator 1 time.
DUAL formed by presolve
LP Presolve eliminated 388864 rows and 399011 columns.
Aggregator did 243985 substitutions.
Reduced LP has 837186 rows, 1077942 columns, and 8028292 nonzeros.
Presolve time = 9.17 sec. (8220.72 ticks)
Parallel mode: using up to 7 threads for barrier.

***NOTE: Found 186 dense columns.

Number of nonzeros in lower triangle of $A^*A'$ = 30819597
Elapsed ordering time = 8.79 sec. (10000.00 ticks)
Total time for nested dissection ordering = 10.96 sec. (12549.98 ticks)
Summary statistics for Cholesky factor:
  Threads = 7
  Rows in Factor = 837372
  Integer space required = 8718128
  Total non-zeros in factor = 1322482575
  Total FP ops to factor = 9781244195485
\end{verbatim}
Main differences CPLEX\Simplex vs CPLEX\Barrier

- Objective values may be the same, but nature of solutions differs:
  - Barrier relies on convergence criteria and cannot move to the exact optimal solution
  - Barrier does not produce a basic solution → for this needs the Crossover* to be invoked
  - Without a basic solution advanced start basis information cannot be used

- Barrier requires significantly more memory than Simplex

- Barrier works well when $A^*A'$ is sparse and can also solve degenerate problems

- Barrier is sensitive to the presence of unbounded optimal faces

* Crossover transforms a typically near optimal Barrier solution to an optimal extreme-point solution
  - In case of a unique optimal solution the Barrier method converges well to the exact optimal solution
  - Crossover does not benefiting much from parallel hardware
  - Crossover is time consuming when initiated on an interior point solution very distant from the optimal solution
CPLEX\Barrier solution with and w/o Crossover

CPLEX\Barrier log file with Crossover (invoked by default) (solution time 8':31'’)

CPLEX\Barrier log file without Crossover (solution time 3’:29’’)

Difference in the objective function: 0.0000012%
CPLEX\Barrier solution with and w/o Crossover

GDX solution file with Crossover (invoked by default)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Dim</th>
<th>Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_OUT</td>
<td>Parameter</td>
<td>6</td>
<td>78454</td>
</tr>
</tbody>
</table>

GDX solution file without Crossover

<table>
<thead>
<tr>
<th>Entry</th>
<th>Name</th>
<th>Type</th>
<th>Dim</th>
<th>Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>1146</td>
<td>F_OUT</td>
<td>Parameter</td>
<td>6</td>
<td>89553</td>
</tr>
</tbody>
</table>

Notice that:

a) without crossover there are more entries in F_Out and
b) these entries have values close to zero
It has two main phases involving Simplex:
- primal variables are pushed to the bounds
- dual variables are pushed to the bounds

- CPLEX initiates primal and dual Simplex in a concurrent optimisation:
  - both primal and dual Crossover are invoked, and in both primal and dual pushes occur
  - usually the dual Crossover log is displayed
  - if primal Simplex finishes first it is indicated in the log

- After Crossover the basis is either optimal or CPLEX starts to re-optimize with Simplex
Primal push indicated with PMoves in the log (dual push is similar):

- Begin with a square full rank basis $x_B$

$$Ax^* = Bx_B + Sx_s = b$$

- While there is a superbasic* variable $l_j < x_j < u_j$ (with $l_j, u_j$ the lower and upper bound of $x_j$):
  - either push $x_j \rightarrow (l_j$ or $u_j)$ by adjusting $x_B$ (push)
  - or, move $x_j$ into basis and push another basic variable $x_i \rightarrow (l_j$ or $u_j)$ (exchange)

As Crossover is based on Simplex, the options for Simplex may be applied to accelerate it

* Superbasic variable is a non-basic variable that is not on one of its bounds

---

Sample output from the CPLEX\Barrier log file with Crossover

| 42 | 2.9284042e+05 | 2.9284042e+05 | 1.31e-06 | 3.44e-08 | 6.63e-02 | 1.29e+08 |
| 43 | 2.9284042e+05 | 2.9284042e+05 | 8.15e-07 | 1.08e-08 | 2.34e-02 | 4.24e+08 |

Barier time = 25.78 sec. (24008.51 ticks)

Parallel mode: deterministic, using up to 19 threads for concurrent optimization:
* Starting dual Simplex on 1 thread...
* Starting primal Simplex on 1 thread...

Dual crossover.
Dual: Fixing 181532 variables.
181531 PMoves: Infeasibility 8.21656627e-03 Objective 2.92840421e+05
180635 DMoves: Infeasibility 6.08732927e-03 Objective 2.92840421e+05
...
0 DMoves: Infeasibility 1.48538034e-01 Objective 2.92840437e+05
Dual: Pushed 128908, exchanged 68922.
Primal: Fixing 18078 variables.
18077 PMoves: Infeasibility 2.86482058e-03 Objective 2.92840422e+05
15715 PMoves: Infeasibility 1.86455138e-03 Objective 2.92840422e+05
...
0 PMoves: Infeasibility 5.72728385e-04 Objective 2.92840423e+05
Primal: Pushed 12819, exchanged 5259.
Elapsed time = 108.81 sec. (193276.72 ticks, 1 iterations)
Primal simplex solved model.
Total crossover time = 132.84 sec. (180911.98 ticks)
Total time on 19 threads = 162.39 sec. (129699.81 ticks)

--- LP status (1): optimal
--- Cplex Time: 162.44sec (det. 129701.02 ticks)

Optimal solution found.
Simplex iterations after crossover: 21347
Objective: 292840.421485
Internally Barrier solves the primal problem by turning all inequalities to equalities:

\[
\begin{align*}
\min & \quad c^T x \\
Ax &= b & (\text{dual } y) \\
x + s &= u & (\text{dual } w) \\
x &\geq l & (\text{dual } z)
\end{align*}
\]

The corresponding dual problem is:

\[
\begin{align*}
\max & \quad b^T y - u^T w + l^T z \\
A^T y - w + z &= c & (\text{dual } x) \\
w, z &\geq 0
\end{align*}
\]

Infeasibility

<table>
<thead>
<tr>
<th>Infeasibility</th>
<th>In log file</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>primal</td>
<td>Prim Inf</td>
<td>(</td>
</tr>
<tr>
<td>upper</td>
<td>Upper Inf</td>
<td>(</td>
</tr>
<tr>
<td>dual</td>
<td>Dual Inf</td>
<td>(</td>
</tr>
</tbody>
</table>

Infeasibility ratio

It is applicable only for specific Barrier algorithm variants; it should increase to a large number towards the optimal solution.
In contrast to Simplex, Barrier uses relative measures and tolerances of primal feasibility, dual feasibility and complementarity slackness to assess convergence to an optimal solution.

The feasibility of a primal solution is assessed with respect to the variables’ values in the solution:

$$\frac{\|Ax - b\|}{1 + \|x\|} \leq \text{barepcomp}$$

The optimality criterion is “usually” based on a normalized duality gap* (primal-dual pairs):

$$\frac{|c^T x - y^T b|}{1 + |c^T x|} \leq \text{barepcomp}$$

CPLEX\Barrier may accept a solution with large violations $\|Ax - b\|$ if solution values $\|x\|$ are also large

- Constraint “violation” is $|Ax - b| \neq 0$, while “infeasibility” is $\frac{\|Ax - b\|}{1 + \|x\|} > \text{barepcomp}$

- A violation which does not cause $\frac{\|Ax - b\|}{1 + \|x\|} > \text{barepcomp}$ is not an infeasibility

Tuning the performance of the CPLEX\Barrier algorithm

with red are related CPLEX parameters (click to go to the related section in the CPLEX manual)
Our test model used as an example

- It is a multi-region TIMES model with 5 regions
  - Milestone years 2018, 2020, 2024, 2050, 2060 and 2070
  - 96 timeslices

- The size of the model constraint matrix A, before any reductions made by CPLEX Presolve:
  - 1 652 244 equations (rows)
  - 1 480 090 variables (columns)
  - 10 174 369 non-zero coefficients in the constraint matrix A

- The size of the model constraint matrix A, after reductions made by CPLEX Presolve:
  - 836 966 equations
  - 1 077 792 variables
  - 8 028 092 non-zeros

- CPLEX Presolve reduces the size of the model matrix by ~20%

- In the following examples, Presolve related options are not discussed or changed
• Interior point algorithms should be preferred when parallel hardware is available

• RAM requirements increase with the number of CPLEX threads

• The solution times exhibit diminishing returns or even deteriorate above a threshold of threads

Model statistics before presolve: equations 1652244, variables 1480090, non-zeros 10174368
Model statistics after presolve: equations 1019245, variables 837216, non-zeros 7969795
Hardware: HPE Apollo XL230K Gen10 blade with 2 Intel® Xeon® Gold 6152 Scalable Processors @ 2.10GHz (2 x 22 cores, 384GB RAM)
Fluctuations in the solution time are due to the variable cluster workload where the tests were made
When solving our test TIMES model with the default CPLEX options:

- The DUAL problem is formed by Presolve
- The non-zeros in factor are 1305.8 million
- The total FP ops is 1.01E+13
- The Barrier algorithm must perform calculations with a very dense Cholesky factor and with many floating point operations
- CPLEX\Barrier needs 5 hours and Crossover takes 4 more hours → 9 hours in total

Test 1: Using the default CPLEX options → solution time 9 hours
Primal vs dual problem and performance gains

• Simplex is influenced by the number of constraints $m$ than by the number of variables $n$
  – If $(m \ll n)$ or $(m \gg n$ and $\frac{n}{m} > 0.5)*$ the primal problem should be solved
  – Otherwise, solving the dual problem could lead to performance gains

• Barrier is influenced by the number of dense columns that result in a dense $A^*A'$ matrix:
  – Dense columns in the primal problem become dense rows in the dual
  – Dense columns in the dual become dense rows in the primal
  – It is not always straightforward if the primal performs better than the dual problem

• The CPLEX option `Predual` overrides default behavior in choosing which problem to solve

* Klotz and Newman, 2013. Practical guidelines for solving difficult linear problems, 
Forcing Presolve to pass the primal problem

- **Predual** = -1 is set to force solving the primal problem

- The dense columns increased to 319 (+72% from the previous case)

- The non-zeros in factor are 552.4 million (-60% from the previous case)

- The total FP ops is $2.34 \times 10^{12}$ (i.e. -77% from the previous case)

- CPLEX\Barrier needs 33 minutes and Crossover takes 80 more minutes → 113 minutes in total

Test 2: Adding the **Predual** CPLEX option → solution time 113’
The role of the “dense columns” in performance

- Dense columns degrade performance: a dense column has a given variable in many rows
- Barrier invocation techniques to handle the detected dense columns and reduce their impact

Barrier reports the detected dense columns

--- test_2.run(116997) 1803 Mb
--- 1,652,244 rows, 1,480,090 columns 10,1
--- Executing CPLEX: elapsed 6:00:33.896
...

Reading parameter(s) from "/data/user/panos_e
>> lpmethod 4
>> threads 22
>> pordual -1
Finished reading from "/data/user/panos_e/ETS
Reading data...
Starting Cplex...
...
Tried aggregator 3 times.
LP Presolve eliminated 388864 rows and 398831
Aggregator did 244135 substitutions.
Reduced LP has 1019245 rows, 837216 columns,
Presolve time = 8.53 sec. (6434.65 ticks)
Parallel mode: using up to 22 threads for bar
***NOTE: Found 319 dense columns.
Number of nonzeros in lower triangle of A*A'

FP Operations in trillion (1E12)
(full range of dense columns)

FP Operations in trillion (1E12)
(zoom in the first 5000 columns)

Dual: 186
Primal: 319
Minimum at 2906 dense columns
• Produce the column histogram of matrix A with respect to the number of non-zeros coefficients, either with:
  – ** bardisplay** =2 writes the histogram in the log, or,
  – the **CONVERT** solver to export the Jacobian which can be processed for the histogram*

• Check the histogram and identify the #non-zeros which shows presence of dense columns

• Set the **barcolnz** option around to this value

* Elementary code in Python available upon request
Changing the default number of dense columns

- **barcolnz** = 150 is set to treat more columns as dense

- The dense columns increased to 2906 (+811% from the previous case)

- The non-zeros in factor are 276.0 million (-53% from the previous case)

- The total FP ops is 1.31E+12 (i.e. -50% from the previous case)

- CPLEX\ Barrier needs 29 minutes and Crossover takes 55 more minutes \(\rightarrow\) 84 minutes in total

---

Test 3: Adding the **barcolnz** CPLEX option \(\rightarrow\) solution time 84'
• Barrier permutes the rows of A to reduce the fill in the Cholesky factor (fill-in = the difference in the non-zeros between the Cholesky factor and the lower triangle of A*A’)

• The option barorder can override the default selection of the algorithm, if needed:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatic</td>
</tr>
<tr>
<td>1</td>
<td>Approximate Minimum Degree (AMD)</td>
</tr>
<tr>
<td>2</td>
<td>Approximate Minimum Fill (AMF)</td>
</tr>
<tr>
<td>3</td>
<td>Nested Dissection (ND)</td>
</tr>
</tbody>
</table>

– The AMD algorithm (barorder=1) provides good quality in moderate time
– The AMF algorithm (barorder=2) usually results in 5-10% smaller factors than AMD
– The ND algorithm (barorder=3) produces significantly smaller factors from the three in most cases

**Note:** “Automatic” or “Default” for CPLEX does not mean simply selecting one of the available options in some cases, thus it is good to explicitly try also the default options.
In our test case, Barrier automatically selects the ND that gives the smaller size of the Cholesky factor (number of non-zeros) → however the default \texttt{barorder}=0 is better than selecting \texttt{barorder}=3.

Barrier ordering algorithm and their effect in non-zeros (in millions) for our test model case.

As we get the best ordering and smallest FPs to factor by default, no need to change the algorithm by setting the \texttt{barorder} option in our test case.

\begin{itemize}
\item ***NOTE: Found 2906 dense columns.***
\item Number of nonzeros in lower triangle of $A\times A' = 36716686$
\item Using Nested Dissection ordering
\item Total time for automatic ordering = 55.05 sec. (63682.77 ticks)
\item Summary statistics for Cholesky factor:
  \begin{itemize}
  \item Threads = 22
  \item Rows in Factor = 1822151
  \item Integer space required = 7358991
  \item Total non-zeros in factor = 276809970
  \item Total FP ops to factor = 1310992918584
  \end{itemize}
\end{itemize}
Additional tuning parameters

with red are related CPLEX parameters (click to go to the corresponding section in the manual)
Tuning parameters affecting iterations performance

• CPLEX\Barrier offers additional tuning parameters that affect the speed of iterations
  – These options do not affect the dense columns or size of the Cholesky factor

• It is recommended to explore them after reducing the complexity of the Cholesky factor
  – algorithm to be used in iterations : baralg
  – estimated starting point in iteration 0 : barstartalg
Barrier algorithm to be used in the iterations

- CPLEX\Barrier offers three algorithms to choose using the `baralg` option (default value is 0):

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Same as 1 for MIP subproblems, 3 otherwise</td>
</tr>
<tr>
<td>1</td>
<td>Infeasibility-estimate start</td>
</tr>
<tr>
<td>2</td>
<td>Infeasibility-constant start</td>
</tr>
<tr>
<td>3</td>
<td>standard barrier algorithm</td>
</tr>
</tbody>
</table>

- The standard barrier algorithm `baralg`=3 performs well in most of the cases.

- Algorithms `baralg`=1 and `baralg`=2 can also detect infeasibilities and can be helpful in a problem with numerical difficulties to improve the quality of the solution, but are slower than the standard algorithm.

  - `baralg`=1 and `baralg`=2 differ from each other on how the estimate the starting point of Barrier.

*Note:* “Automatic” or “Default” for CPLEX does not mean simply selecting one of the available options in some cases, thus it is good to explicitly try also the default options.
By forcing \texttt{baralg} = 3 the obtained solution time of our test model drops to 36 minutes.

The improvement gains are mainly sought on Crossover:
- Barrier: 21 min. (-28\% from the previous test)
- Crossover: 15 min. (-73\%) due to a better starting point

While \texttt{baralg} = 3 is the default by explicitly setting it here it improves the solution time:
- it looks like “automatic” or “default” in CPLEX goes beyond simply choosing one of the available options.

---

Test 4: Adding the \texttt{baralg} CPLEX option → solution time 36'
• CPLEX\Barrier offers four heuristics to estimate a starting point, with the option `barstartalg` (default 1)

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>default primal, dual is 0</td>
</tr>
<tr>
<td>2</td>
<td>default primal, estimate dual</td>
</tr>
<tr>
<td>3</td>
<td>primal average, dual is 0</td>
</tr>
<tr>
<td>4</td>
<td>primal average, estimate dual</td>
</tr>
</tbody>
</table>

• When we pass the primal problem `barstartalg` = 1 or 2 works well

• When we pass the dual problem is worthy also to check `barstartalg` = 2 or 4

• Starting algorithm `barstartalg` = 3 seems to be slower in most cases

• When numerical difficulties are present is worthy to check alternatives algorithms for the starting point
Implications of applying a different starting point

- In our test example, the choice of the starting point affects the number of barrier iterations, the starting point of the crossover algorithm, and the number of simplex iterations after crossover.

<table>
<thead>
<tr>
<th></th>
<th>barstartalg 1</th>
<th>barstartalg 2</th>
<th>barstartalg 3</th>
<th>barstartalg 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting iteration primal obj value</td>
<td>8.0E+11</td>
<td>8.0E+11</td>
<td>1.4E+10</td>
<td>1.4E+10</td>
</tr>
<tr>
<td>Starting iteration dual obj value</td>
<td>-4.5E+10</td>
<td>-3.8E+10</td>
<td>-4.5E+10</td>
<td>-3.8E+10</td>
</tr>
<tr>
<td>Starting iteration primal infes</td>
<td>8.2E+09</td>
<td>8.2E+09</td>
<td>4.1E+07</td>
<td>4.1E+07</td>
</tr>
<tr>
<td>Starting iteration dual infes</td>
<td>1.4E+10</td>
<td>1.4E+10</td>
<td>1.4E+10</td>
<td>1.4E+10</td>
</tr>
<tr>
<td>Number of barrier iterations</td>
<td>201</td>
<td>180</td>
<td>225</td>
<td>211</td>
</tr>
<tr>
<td>End Barrier iteration primal obj value</td>
<td>6.5900073E+09</td>
<td>6.5900073E+09</td>
<td>6.5900073E+09</td>
<td>6.5900073E+09</td>
</tr>
<tr>
<td>End Barrier iteration dual obj value</td>
<td>6.5900070E+09</td>
<td>6.5900070E+09</td>
<td>6.5900070E+09</td>
<td>6.5900070E+09</td>
</tr>
<tr>
<td>End Barrier iteration primal infes</td>
<td>3.8E-02</td>
<td>4.2E-02</td>
<td>4.3E-02</td>
<td>4.6E-02</td>
</tr>
<tr>
<td>End Barrier iteration dual infes</td>
<td>5.5E+00</td>
<td>6.7E+00</td>
<td>2.8E+00</td>
<td>6.9E+00</td>
</tr>
<tr>
<td>Dual crossover first iteration obj value</td>
<td>6.59000706E+09</td>
<td>6.59000709E+09</td>
<td>6.59000709E+09</td>
<td>6.59000703E+09</td>
</tr>
<tr>
<td>Dual crossover first iteration dual infes</td>
<td>3.8E+00</td>
<td>4.5E+00</td>
<td>5.3E+00</td>
<td>3.8E+00</td>
</tr>
<tr>
<td>Simplex iterations after crossover</td>
<td>4437</td>
<td>4033</td>
<td>11313</td>
<td>2640</td>
</tr>
<tr>
<td>Scaled infeasibilities occur</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Optimal objective value</td>
<td>6590007223.3373</td>
<td>6590007223.3374</td>
<td>6590007223.3373</td>
<td>6590007223.3373</td>
</tr>
</tbody>
</table>

Note: Due to fluctuating cluster load where the test made, we can argue that practically there is no difference between 1, 2, and 4.
• Barrier relies on convergence criteria and may struggle to converge by performing iterations without improving the objective
  – In such cases, often Barrier reverts to a previous iteration when terminating and this is indicated with an * in the .log file

• The performance might be improved if increasing the Barrier convergence tolerance barepcomp to avoid the additional iterations:
  – important is not to alter “too much” the starting point of Crossover

• Usually such cases also denote presence of numerical difficulties

**“Wasted” Barrier iterations**

Barrier reverts to iteration 183 and ~20 iterations are “wasted”
(based on the barstartalg=1 case requiring 38’ to be solved)
### Increasing the CPLEX\Barrier convergence tolerance

- To increase the tolerance we need to calculate an approximation of the duality gap:

\[
\frac{|c^T x - y^T b|}{1 + |c^T x|}
\]

- The calculation shows that after iteration 182 the duality gap remains at 4.552347E-8

- The default value of `barepcomp` is 1E-8

- So in this example we can increase the tolerance to stop Barrier around iteration 182, e.g., by setting it to 4.7E-8

### Table

<table>
<thead>
<tr>
<th>Itn</th>
<th>Primal Obj</th>
<th>Dual Obj</th>
<th>Prim Inf</th>
<th>Upper Inf</th>
<th>Dual Inf</th>
<th>Duality gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.99E+11</td>
<td>-4.50E+10</td>
<td>8.19E+09</td>
<td>4.62E+08</td>
<td>1.39E+10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7.24E+11</td>
<td>-1.06E+11</td>
<td>7.19E+09</td>
<td>4.06E+08</td>
<td>1.12E+10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6.12E+11</td>
<td>-2.03E+11</td>
<td>5.34E+09</td>
<td>3.01E+08</td>
<td>8.69E+09</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>1.98E-02</td>
<td>1.94E-10</td>
<td>4.94E+00</td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>2.27E-02</td>
<td>2.24E-10</td>
<td>5.11E+00</td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.61E-02</td>
<td>1.97E-10</td>
<td>5.28E+00</td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.61E-02</td>
<td>3.02E-10</td>
<td>5.54E+00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.81E-02</td>
<td>3.75E-10</td>
<td>5.56E+00</td>
<td></td>
</tr>
<tr>
<td>182</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.81E-02</td>
<td>4.23E-10</td>
<td>5.70E+00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>5.13E-10</td>
<td>6.00E+00</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>5.33E-10</td>
<td>6.15E+00</td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>5.57E-10</td>
<td>6.54E+00</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>5.77E-10</td>
<td>6.42E+00</td>
<td></td>
</tr>
<tr>
<td>195</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>6.04E-10</td>
<td>6.76E+00</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>6.19E-10</td>
<td>6.96E+00</td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>6.42E-10</td>
<td>7.11E+00</td>
<td></td>
</tr>
<tr>
<td>198</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>6.56E-10</td>
<td>7.57E+00</td>
<td></td>
</tr>
<tr>
<td>199</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.82E-02</td>
<td>6.63E-10</td>
<td>7.52E+00</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.92E-02</td>
<td>7.94E-10</td>
<td>8.52E+00</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>6.59E+09</td>
<td>6.59E+09</td>
<td>3.81E-02</td>
<td>4.55E-10</td>
<td>5.53E+00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Duality gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.053835E+00</td>
</tr>
<tr>
<td>1.145917E+00</td>
</tr>
<tr>
<td>1.330828E+00</td>
</tr>
<tr>
<td>2.124429E-07</td>
</tr>
<tr>
<td>2.124429E-07</td>
</tr>
<tr>
<td>1.517449E-07</td>
</tr>
<tr>
<td>1.517449E-07</td>
</tr>
<tr>
<td>6.069796E-08</td>
</tr>
<tr>
<td>6.069796E-08</td>
</tr>
<tr>
<td>6.069796E-08</td>
</tr>
<tr>
<td>6.069796E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
<tr>
<td>4.552347E-08</td>
</tr>
</tbody>
</table>

To increase the tolerance we need to calculate an approximation of the duality gap:

\[
\frac{|c^T x - y^T b|}{1 + |c^T x|}
\]

The calculation shows that after iteration 182 the duality gap remains at 4.552347E-8.

The default value of `barepcomp` is 1E-8.

So in this example we can increase the tolerance to stop Barrier around iteration 182, e.g., by setting it to 4.7E-8.
The effect of increasing the convergence tolerance

By increasing the tolerance from $1 \times 10^{-8}$ to $4.7 \times 10^{-8}$ the overall solution time decreased by 3’
- Barrier time from 23’ to 19’
- Crossover time from 15’ to 17’

- Crossover time increased due to a “worse” starting point

- Trade-off between Barrier and Crossover time

Test 5: With default tolerance → solution time 39’
Test 6: Increasing barecomp → solution time 36’
CPLEX parameters affecting the Crossover performance

with *red* are related CPLEX parameters
(click to go to the related section in the [CPLEX manual](http://example.com))
Crossover typically comprises a small percentage time unless if:
- the initiated suboptimal interior point solution has a “significant distance” from the optimal
- numerical difficulties, ill conditioning or degeneracy are present

Options that can improve the crossover performance:
- `barcrossalg` : selects which Crossover method to be used
- `barepcomp` : reducing it may provide a better starting point for Crossover
- `ppriind, dprind` : sets the pricing algorithm of Simplex when choosing variables for the base

Crossover can be turned off with `solutiontype=2` that instructs CPLEX not to seek a basic solution:
- the interior point midface solution from Barrier is reported
- can be useful for a quick insight of the approx. optimal solution, if crossover takes long time
Changing the Crossover algorithm

• By default CPLEX invokes primal and dual Crossover in parallel when multiple threads are available
  – Otherwise, the choice depends on the type of the problem solved

• The parameter `barcrossalg` can be used to instruct CPLEX to use a specific Crossover algorithm

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatic</td>
</tr>
<tr>
<td>1</td>
<td>Primal crossover</td>
</tr>
<tr>
<td>2</td>
<td>Dual crossover</td>
</tr>
</tbody>
</table>
Recall that in our example we solve the primal problem.

It therefore makes sense to set `barcrossalg` = 1 to force CPLEX to use primal Crossover.

The solution time is reduced to 30’.

Compared to slide 32:
- the Barrier time is almost unaffected to 20’,
- the Crossover time reduced from 17’ (in slide 31) to 10’

Test 7: Adding `barcrossalg` → solution time 30’
Reducing convergence tolerance to provide a better starting point to Crossover

- If...
  a) Barrier iterations are not excessive
  b) Show no long tail of iterations with little progress
  c) Crossover (or Simplex after Crossover) need a lot of iterations

- ....then reducing the convergence tolerance `barepcomp` may provide a better starting point for Crossover and improve performance

- However, reducing the tolerance well below the default value 1E-8 entails the risk that Barrier finds the model infeasible
In our example, we increased the tolerance to 4.7E-8 to avoid entirely the wasted iterations in Barrier.

This deteriorated the starting point of the Crossover.

By setting `barepcomp`=4.6E-8 we accept a couple of wasted iterations in Barrier to gain time in Crossover.

The new solution time is 28’.
The Crossover algorithm is based on Simplex
  - Simplex iterations are affected by the pricing algorithm used when selecting variables for the basis

Several pricing algorithms exist with the most common ones being:
  - Reduced-cost: computationally inexpensive but usually leads to many iterations
  - Steepest edge: most powerful, reduces iterations but has large work per iteration
  - Devex: approximation to steepest edge with less work per iteration and can reduce iterations

CPLEX offers variants of steepest edge implementations, useful for difficult problems
  - For primal, CPLEX by default uses hybrid reduced-cost and devex pricing
  - For dual, CPLEX by default uses steepest edge pricing

The choice of the pricing algorithm is affected by the options `ppriind` for primal and `dpriind` for dual
The default values for the pricing algorithm of Simplex work well for our test model.

A slight reduction in the solution time was obtained when *steep edge pricing with slack initial norms* was used, which reduces the computational intensity of the steepest edge algorithm, by setting $ppriind=3$.

The $ppriind$ was used because we solve the primal problem.

Test 9: Changing $ppriind$ CPLEX option → solution time 26'
Without Crossover (by setting `solutiontype`=2) the solution is not basic

- MILP problems benefit from a basic solution
- Duals can be different in a non-basic solution
- Simplex warm starts are not possible without a basic solution
- Solutions can be a bit “ugly” (see slide 9)
A few words about using and advanced basis to warm-up Simplex
The CPLEX option `advind` enables Simplex (not Barrier) to use an advanced basis (i.e. a basic solution from a previous run) for a warm start up, if the model is “slightly” modified.

- If primal feasibility is to be retained after modification, better to use Primal Simplex in the warm startup.
- If dual feasibility is to be retained then Dual Simplex in the warm startup could be more suitable.

### Problem modification vs Primal vs dual feasibility

<table>
<thead>
<tr>
<th>Problem modification</th>
<th>Primal vs dual feasibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add variables</td>
<td>Primal feasibility</td>
</tr>
<tr>
<td>Add rows</td>
<td>Dual feasibility</td>
</tr>
<tr>
<td>Change objective function coefficients</td>
<td>Primal feasibility</td>
</tr>
<tr>
<td>Change right-hand side coefficients</td>
<td>Dual feasibility</td>
</tr>
<tr>
<td>Change matrix coefficients of basic variables</td>
<td>Feasibility lost</td>
</tr>
<tr>
<td>Change matrix coefficients of non-basic variables</td>
<td>Primal feasibility</td>
</tr>
</tbody>
</table>

“The End”

Overview from the steps taken to accelerate a TIMES model
Summary of the steps taken to speed up our test model

- Default CPLEX options, 546 minutes
- Forcing primal problem
- Handling dense columns
- Forcing standard Barrier
- Chosing starting point
- Avoiding wasted iterations
- Forcing primal crossover
- Improving crossover starting point
- Selecting pricing algorithm
- Disable Crossover

Note: the solution time reductions achieved in each step depend on the model examined.

- Ipmethod 4 threads 22
- Predual -1
- Barcolnz 150
- Barstartalg 1
- Baralg 3
- Barepcomp 4.6E-08
- Barcrossalg 1
- Ppriind 3

- Final CPLEX options, 26 minutes

Solution time reductions -95% in solution time

Solution time in minutes

0 50 100 150 200 250 300 350 400 450 500

Default CPLEX options
Forcing primal problem
Handling dense columns
Forcing standard Barrier
Chosing starting point
Avoiding wasted iterations
Forcing primal crossover
Improving crossover starting point
Selecting pricing algorithm
Disable Crossover
Thank you very much for the attention
Dr. Evangelos Panos
Energy Economics Group
Laboratory for Energy Systems Analysis
e-mail: evangelos.panos@psi.ch

My thanks goes to:
Fred Fiand, GAMS
George Giannakidis, ETSAP