IBM® ILOG CPLEX®
Optimization Studio 12.2
What’s New

Fully integrated development environment for prototyping, building and deploying analytical decision support applications using constraint and/or mathematical programming
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - Multi-commodity flow cuts
  - Pseudo-cost updates for infeasible nodes
  - Deterministic parallel barrier solver
  - Parallel algorithms available with default license

- Computational results
Interoperability

- Microsoft Excel, MATLAB, Python
- Not a new concept for CPLEX
  - C, C++, Java, C# and .NET
    - Accessible from other languages that can call C (Fortran, Delphi, APL, Cobol…)
  - OPL, AMPL, GAMS, AIMMS, MPL, …
  - Interactive optimizer (not always an API)
- Standard feature of CPLEX 12
  - No extra cost
  - CPLEX’s 20 year history of continuous feature and performance improvements
  - Online help plus other documentation formats
  - Numerous examples
Interoperability with MS Excel ®

- Previously, CPLEX could solve models that referenced spreadsheet data
  - modeling languages (OPL, AMPL, MPL, GAMS, AIMMS, etc.) all can input Excel spreadsheets
    - manage data within Excel, models within modeling language
  - Visual Basic and VBA could call the CPLEX C, .NET APIs
Interoperability with MS Excel®

- New Excel add-in enables model management in spreadsheet
  - manage data and model inside Excel
  - no additional tools needed
Diet Selection Problem: Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

<table>
<thead>
<tr>
<th>Problem Data</th>
<th>Foods</th>
<th>Food Cost</th>
<th>Food Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Feed1</td>
<td>Feed2</td>
<td>Feed3</td>
</tr>
<tr>
<td>Food Cost</td>
<td>1.84</td>
<td>2.19</td>
<td>1.84</td>
</tr>
<tr>
<td>Food Supply</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>Nutrients per Food Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbohydrate</td>
<td>510 370 500 370 420 320 220 346 110 80</td>
</tr>
<tr>
<td>Fat</td>
<td>34   36  42  31  42  25  27  12  20  20</td>
</tr>
<tr>
<td>Protein</td>
<td>28   24  25  14  31  3  15  9  1  1</td>
</tr>
<tr>
<td>Calcium</td>
<td>15   15  6   2   8   0   4   10  2</td>
</tr>
<tr>
<td>Vitamin E</td>
<td>6    10  2   0   15  15  0   4   4</td>
</tr>
<tr>
<td>Vitamin C</td>
<td>30   20  25  15  15  0   20  30  2</td>
</tr>
<tr>
<td>Vitamin D</td>
<td>20   20  20  10  8   2   15  0   2</td>
</tr>
</tbody>
</table>

The variables are the amounts of each food to collect for the diet.

Amount of Food

The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.

Diet Cost

The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states that the sum of the amount of the nutrient in the food times the amount of each food must be more than the minimum nutritional requirement but less than the maximum requirement.

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>Minimum</th>
<th>Provided</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbohydrate</td>
<td>3000</td>
<td>0</td>
<td>9999</td>
</tr>
<tr>
<td>Fat</td>
<td>300</td>
<td>0</td>
<td>375</td>
</tr>
<tr>
<td>Protein</td>
<td>85</td>
<td>0</td>
<td>9999</td>
</tr>
<tr>
<td>Calcium</td>
<td>100</td>
<td>0</td>
<td>9999</td>
</tr>
<tr>
<td>Vitamin E</td>
<td>100</td>
<td>0</td>
<td>9999</td>
</tr>
<tr>
<td>Vitamin C</td>
<td>100</td>
<td>0</td>
<td>9999</td>
</tr>
<tr>
<td>Vitamin D</td>
<td>100</td>
<td>0</td>
<td>9999</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 2008, 2009. All Rights Reserved.
### Diet Selection Problem

Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

#### Problem Data

<table>
<thead>
<tr>
<th>Foods</th>
<th>Food1</th>
<th>Food2</th>
<th>Food3</th>
<th>Food4</th>
<th>Food5</th>
<th>Food6</th>
<th>Food7</th>
<th>Food8</th>
<th>Food9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food Cost</td>
<td>1.64</td>
<td>2.19</td>
<td>1.84</td>
<td>1.84</td>
<td>1.84</td>
<td>1.44</td>
<td>2.29</td>
<td>0.77</td>
<td>0.77</td>
</tr>
<tr>
<td>Food Supply</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Nutrients

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>510</th>
<th>370</th>
<th>500</th>
<th>370</th>
<th>420</th>
<th>220</th>
<th>345</th>
<th>110</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fat</td>
<td>34</td>
<td>34</td>
<td>45</td>
<td>39</td>
<td>42</td>
<td>26</td>
<td>26</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>Protein</td>
<td>28</td>
<td>24</td>
<td>25</td>
<td>28</td>
<td>25</td>
<td>14</td>
<td>31</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Calcium</td>
<td>15</td>
<td>15</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Vitamin A</td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>Vitamin C</td>
<td>30</td>
<td>30</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>30</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Vitamin D</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Variable Cells

The variables are the amounts of each food to select for the diet.

<table>
<thead>
<tr>
<th>Amount of Food</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

#### Diet Cost

The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.

Diet Cost: 0

#### Constraints

The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states that the sum of the amount of the nutrient in the food times the amount of each food must be more than the minimum nutritional requirement but less than the maximum requirement.

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>Minimum</th>
<th>Provided</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbohydrate</td>
<td>2000.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Fat</td>
<td>350.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Protein</td>
<td>55.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Calcium</td>
<td>100.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Vitamin A</td>
<td>100.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Vitamin C</td>
<td>100.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
<tr>
<td>Vitamin D</td>
<td>100.00</td>
<td>0.00</td>
<td>9999.00</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 2008, 2009. All Rights Reserved.
### Objective Function

The objective function is given by:

$$ c^T x $$

where $c$ represents the cost coefficients and $x$ represents the decision variables.
### Diet Selection Problem

Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

#### Constraints:

\[ Ax \]

#### Food Data

<table>
<thead>
<tr>
<th>Foods</th>
<th>Food Cost</th>
<th>Food Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.84</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.19</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1.84</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.29</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.77</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1.29</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.72</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Nutrients

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>Carbohydrate</th>
<th>Protein</th>
<th>Fat</th>
<th>Calcium</th>
<th>Vitamin A</th>
<th>Vitamin C</th>
<th>Vitamin D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>510</td>
<td>370</td>
<td>300</td>
<td>370</td>
<td>390</td>
<td>220</td>
<td>236</td>
</tr>
<tr>
<td></td>
<td>220</td>
<td>310</td>
<td>342</td>
<td>290</td>
<td>42</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>15</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>15</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>3</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>30</td>
<td>20</td>
<td>15</td>
</tr>
</tbody>
</table>

#### Amount of Food

- The variables are the amounts of each food to select for the diet.

#### Diet Cost

The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.

Cost \[= \text{SUMPRODUCT}(\text{B10:J10}, \text{B$19:J$19})\]
Diet Selection Problem: Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diet Selection Problem: Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Problem Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Foods</td>
<td>Food1</td>
<td>Food2</td>
<td>Food3</td>
<td>Food4</td>
<td>Food5</td>
<td>Food6</td>
<td>Food7</td>
<td>Food8</td>
</tr>
<tr>
<td>4</td>
<td>Food Cost</td>
<td>1.84</td>
<td>2.19</td>
<td>1.84</td>
<td>1.44</td>
<td>2.29</td>
<td>0.77</td>
<td>1.59</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>Food Supply</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Nutrients</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Nutrients per Food Unit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Carbohydrate</td>
<td>510</td>
<td>570</td>
<td>500</td>
<td>370</td>
<td>420</td>
<td>220</td>
<td>345</td>
<td>110</td>
</tr>
<tr>
<td>9</td>
<td>Fat</td>
<td>34</td>
<td>36</td>
<td>42</td>
<td>38</td>
<td>42</td>
<td>26</td>
<td>27</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>Protein</td>
<td>28</td>
<td>24</td>
<td>25</td>
<td>14</td>
<td>31</td>
<td>3</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>Calcium</td>
<td>15</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>Vitamin B</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>Vitamin C</td>
<td>30</td>
<td>20</td>
<td>25</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>Vitamin D</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

The variables are the amounts of each food to select for the diet.

- Amount of Food
- The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.
- The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states that the sum of the amount of the nutrient in the food times the amount of each food must be more than the minimum nutritional requirement but less than the maximum requirement.

© Copyright IBM Corp. 2008, 2009. All Rights Reserved.
Diet Selection Problem: Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

The variables are the amounts of each food to select for the diet.

The Diet Cost cell contains the objective function, the sum of the cost of each food times the amount of each food.

The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states that the sum of the amount of the nutrient in the food times the amount of each food must be more than the minimum nutritional requirement but less than the maximum requirement.
### Diet Selection Problem

Choose the amount of foods in a diet to meet nutritional requirements at the lowest cost.

|          | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| **Problem Data** |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| **Foods** | Food1 | Food2 | Food3 | Food4 | Food5 | Food6 | Food7 | Food8 | Food9 | FoodA | FoodB | FoodC | FoodD | FoodE | FoodF | FoodG | FoodH | FoodI | FoodJ |
| **Food Cost** | 1.84 | 2.19 | 1.84 | 1.44 | 2.29 | 0.77 | 1.29 | 0.37 | 0.72 | 10  | 10  | 10  | 10  | 10  | 10  | 10  | 10  | 10  | 10  |
| **Food Supply** | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

### Nutrients per Food Unit

<table>
<thead>
<tr>
<th></th>
<th>Carbohydrate</th>
<th>Protein</th>
<th>Fat</th>
<th>Calcium</th>
<th>Vitamin A</th>
<th>Vitamin C</th>
<th>Vitamin B</th>
<th>Vitamin D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Food1</strong></td>
<td>510</td>
<td>34</td>
<td>15</td>
<td>15</td>
<td>8</td>
<td>30</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td><strong>Food2</strong></td>
<td>350</td>
<td>36</td>
<td>15</td>
<td>22</td>
<td>10</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food3</strong></td>
<td>500</td>
<td>40</td>
<td>6</td>
<td>31</td>
<td>2</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food4</strong></td>
<td>370</td>
<td>42</td>
<td>6</td>
<td>31</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food5</strong></td>
<td>420</td>
<td>42</td>
<td>2</td>
<td>31</td>
<td>0</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food6</strong></td>
<td>220</td>
<td>26</td>
<td>2</td>
<td>31</td>
<td>0</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food7</strong></td>
<td>345</td>
<td>27</td>
<td>2</td>
<td>31</td>
<td>0</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food8</strong></td>
<td>345</td>
<td>27</td>
<td>2</td>
<td>31</td>
<td>0</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td><strong>Food9</strong></td>
<td>345</td>
<td>27</td>
<td>2</td>
<td>31</td>
<td>0</td>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

### Solution

- **Amount of Food**
  - Food1: 4.38226
  - Food2: 0
  - Food3: 0
  - Food4: 0
  - Food5: 0
  - Food6: 0.61754
  - Food7: 0
  - Food8: 0
  - Food9: 0.38226
  - FoodA: 0
  - FoodB: 0
  - FoodC: 0
  - FoodD: 0
  - FoodE: 0
  - FoodF: 0
  - FoodG: 0
  - FoodH: 0
  - FoodI: 0
  - FoodJ: 0

- **Diet Cost**
  - $14.008574

### Constraints

The constraints on the diet are to meet minimum and maximum nutritional requirements. Each constraint states that the sum of the amount of the nutrient in the food times the amount of each food must be more than the minimum nutritional requirement or less than the maximum requirement.

<table>
<thead>
<tr>
<th>Nutrients</th>
<th>Minimum</th>
<th>Provided</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbohydrate</td>
<td>33000</td>
<td>33000</td>
<td>59999</td>
</tr>
<tr>
<td>Fat</td>
<td>30</td>
<td>500</td>
<td>750</td>
</tr>
<tr>
<td>Protiem</td>
<td>30</td>
<td>300</td>
<td>475</td>
</tr>
<tr>
<td>Calcium</td>
<td>100</td>
<td>100</td>
<td>99999</td>
</tr>
<tr>
<td>Vitamin A</td>
<td>132</td>
<td>132</td>
<td>99999</td>
</tr>
<tr>
<td>Vitamin C</td>
<td>200</td>
<td>200</td>
<td>99999</td>
</tr>
<tr>
<td>Vitamin B</td>
<td>100</td>
<td>100</td>
<td>99999</td>
</tr>
<tr>
<td>Vitamin D</td>
<td>100</td>
<td>100</td>
<td>99999</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 2008, 2009. All Rights Reserved.
Interoperability with MS Excel ® – Summary

- Excel add-in gives power and reliability of CPLEX from spreadsheets
  - direct use from the Excel user interface
    - use Excel functions to create the linear and quadratic expressions in the model
  - application support via VBA macros

- Supports the features CPLEX users expect
  - all CPLEX problem types:
    - LP, QP, QCP, MIP, MIQP, MIQCP
  - solution analysis
  - algorithmic parameter setting

- Supports Excel 2003 and 2007
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
Interoperability with MATLAB ®

- What is MATLAB?
  - mathematical toolkit containing a wide variety of math functions to enable rapid development of technical computing applications
  - many additional features besides optimization
  - computing language and an interactive environment
  - graphics and visualization features
  - used in numerous industries and disciplines
Interoperability with MATLAB ®

- CPLEX for MATLAB
  - simple API for assembling model, solving, querying solution
Interoperability with MATLAB ®

cplex.readModel('sosex3.lp')
cplex.Model
Interoperability with MATLAB®

cplex.solve()
Interoperability with MATLAB ®
Interoperability with MATLAB ®

- CPLEX for MATLAB
  - simple API for assembling model, solving, querying solution
  - optimization functions have similar or identical arguments to corresponding MATLAB toolkit functions
    - easy to test drive CPLEX within an existing MATLAB application
Interoperability with MATLAB®

```matlab
f = [-1 -2 -3 -1]';
Aineq = [-1 1 1 10; ...];
[x,fval,exitflag,output] = cplexmilp(f, Aineq, bineq, Aeq, beq, ...);
```
Interoperability with MATLAB ®

- CPLEX for MATLAB
  - simple API for assembling model, solving, querying solution
  - optimization functions have similar or identical arguments to corresponding MATLAB toolkit functions
    - easy to test drive CPLEX within an existing MATLAB application
  - CPLEX class enables additional functionality
    - advanced starts
    - conflict refinement for analysis of infeasible models
    - parameter tuning
    - decomposition/sequential optimization
Interoperability with MATLAB®

cplex = Cplex('mipex1');
cplex.addRows([-inf;-inf;0], ...);
cplex.addCols(1, [-1;1;0], ...);
cplex.Model.obj = [1;2;3;1];
cplex.Model.lb = [0;0;0;2];
cplex.Model.ub = [40;inf;inf;3];
cplex.Model.A = [-1 1 1 10; ...];
cplex.solve();
disp(cplex.Solution.x);
Interoperability with MATLAB ®

- Example: a quadratic model where Q was reported to be nonconvex (not positive semi-definite)

- Question: is Q indefinite due to round off error, or truly indefinite?

- The smallest eigenvalue should give strong evidence:
  - function qpeig(varargin)
  - cpx = Cplex();
  - cpx.readModel(varargin{1});
  - lambda = eig(cpx.Model.Q);
  - disp('The minimum eigenvalue of Q matrix is:')
  - disp(min(lambda))
Interoperability with MATLAB ®

- **CPLEX for MATLAB**
  - simple API for assembling model, solving, querying solution
  - optimization functions have similar or identical arguments to corresponding MATLAB toolkit functions
    - easy to test drive CPLEX within an existing MATLAB application
  - CPLEX class enables additional functionality
    - advanced starts
    - conflict refinement for analysis of infeasible models
    - parameter tuning
    - decomposition/sequential optimization
  - supports all of the problem types CPLEX solves
    - LP, QP, QCP, MIP, MIQP, MIQCP
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
Interoperability with Python®

- What is Python?
  - interpreted scripting language like perl, ruby, awk, …
  - numerous object oriented features
    - classes
    - exception handling
    - lists
    - iterators
Interoperability with Python ®

- Overview of the CPLEX 12 Python API
  - based on the sparse matrix data structures of the C API
    - but more expressive due to Python’s object oriented features
  - flexibility and control of the C API
    - but with potentially faster development time
  - additional interactive functionality not available in CPLEX Interactive Optimizer
    - user requests for Interactive Optimizer features, which required writing a program, e.g., using C or C++ API
      - complex abort criteria
      - inspecting basis inverse rows
      - display solution with more than 6 digits
      - …
    - more efficient debugging and troubleshooting
Interoperability with Python ®

- Display solution with 10 digits:

```python
>>> import cplex
>>> c = cplex.Cplex("afiro.mps.gz")
>>> c.solve()
>>> x = c.solution.get_values()
>>> for j in range(c.variables.get_num()):
    print "x[%d] = %17.10f" % j,x[j]
```

```plaintext
x[0] = 80.0000000000
x[1] = 25.5000000000
...
x[29] = 383.9428571429
x[30] = 0.0000000000
x[31] = 0.0000000000
```
Interoperability with Python ®

- Complex abort criteria:

```python
import cplex
from cplex.callbacks import MIPInfoCallback

class AbortCallback(MIPInfoCallback):
    def __call__(self):
        if self.has_incumbent():
            gap = self.get_MIP_relative_gap()
            timeused = time.time() - self.starttime
            if timeused > 2 and gap < 0.05:
                print "Good enough solution after", timeused, "sec."
                self.abort()

c = cplex.Cplex("aflow30a.mps.gz")
abort_cb = c.register_callback(AbortCallback)
abort_cb.starttime = time.time()
c.solve()
```
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
Multi-Commodity-Flow Cuts

- Capacitated network flow is a common substructure of MIP models
- Numerous papers deal with separating cuts for network design problems
- Idea:
  - automatically identify network structure in general MIPs
  - apply cut separation from network design literature
- Goals:
  - help for models that contain networks
  - minimally degrade performance on other models
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
Pseudocost Updates for Infeasible Nodes

- How to update pseudocosts if a node is infeasible?
  - CPLEX 11 strategy: just ignore those nodes

- New CPLEX 12 strategy
  - associate infeasible nodes with a fake objective value
  - compute the fake objective value from the average pseudocost recorded on feasible nodes
  - ignore infeasible nodes after first feasible has been found

- Results
  - Improvement in time to optimality for satisfiability models and other special models
  - Improvement in time to first solution for regular models
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
    - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
Deterministic Parallel Barrier Solver

- Useful enhancement in its own right
- You can use it in deterministic parallel MIP
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
    - parallel algorithms available with default license

- Computational results
Parallel Algorithms Available with Default License

- Parallel MIP in CPLEX
  - Feature has been available since 1996
  - CPLEX was first solver to introduce deterministic parallel MIP (CPLEX 11, 2007)
  - CPLEX also continues to offer the opportunistic parallel algorithm for additional performance boost

- This enabling technology allowed us to complete the migration to default parallelism in CPLEX 12
  - Default parallel without determinism would be a disaster for users
  - Think of parallel as a standard performance feature for MIP and Barrier, like cuts and Cholesky ordering

- Supported for shared-memory architectures
Outline

- Interoperability
  - Excel connector
  - MATLAB connector
  - Python connector

- New CPLEX 12 ingredients
  - multi-commodity flow cuts
  - pseudo-cost updates for infeasible nodes
  - deterministic parallel barrier solver
  - parallel algorithms available with default license

- Computational results
## Comparison of CPLEX 11 and 12 – MIP sequential

<table>
<thead>
<tr>
<th>sup(time)</th>
<th>#</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1)</td>
<td>985</td>
<td>1.01</td>
</tr>
<tr>
<td>[1,10)</td>
<td>469</td>
<td>1.09</td>
</tr>
<tr>
<td>[10,100)</td>
<td>391</td>
<td>1.22</td>
</tr>
<tr>
<td>[100,1k)</td>
<td>284</td>
<td>1.30</td>
</tr>
<tr>
<td>[1k,10k)</td>
<td>201</td>
<td>1.69</td>
</tr>
<tr>
<td>[1,10k]</td>
<td>1436</td>
<td>1.32</td>
</tr>
<tr>
<td>[100,10k]</td>
<td>576</td>
<td>1.61</td>
</tr>
<tr>
<td>[1k,10k]</td>
<td>292</td>
<td>2.01</td>
</tr>
</tbody>
</table>
# CPLEX 12 – Speed-up summary relative to CPLEX 11

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>≥ 1 s</th>
<th>≥ 10 s</th>
<th>≥ 100 s</th>
<th>≥ 1000 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP Primal Simplex</td>
<td>no performance improvements</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP Dual Simplex</td>
<td>1.03</td>
<td>1.04</td>
<td>1.04</td>
<td>1.09</td>
</tr>
<tr>
<td>LP Barrier (1 thread)</td>
<td>1.06</td>
<td>1.07</td>
<td>1.09</td>
<td>1.12</td>
</tr>
<tr>
<td>LP Barrier (4 threads)</td>
<td>1.22</td>
<td>1.23</td>
<td>1.18</td>
<td>1.10</td>
</tr>
<tr>
<td>QP (Barrier 1 thread)</td>
<td>1.06</td>
<td>1.08</td>
<td>1.11</td>
<td>1.20</td>
</tr>
<tr>
<td>MIP (1 thread)</td>
<td>1.32</td>
<td>1.45</td>
<td>1.61</td>
<td>2.01</td>
</tr>
<tr>
<td>MIP (deterministic 4 threads)</td>
<td>1.30</td>
<td>1.43</td>
<td>1.54</td>
<td>1.74</td>
</tr>
<tr>
<td>MIQP (1 thread)</td>
<td>1.21</td>
<td>1.24</td>
<td>1.48</td>
<td>2.38</td>
</tr>
<tr>
<td>MIQCP (1 thread)</td>
<td>1.06</td>
<td>1.01</td>
<td>1.14</td>
<td>1.23</td>
</tr>
</tbody>
</table>
New in CPLEX Optimization Studio 12.2 – CPLEX Optimizer

- **Enhanced speed**
  - 2.2X* on MIP problems taking more than 100 secs
  - 16%* on LP problems solved with Barrier > 1 sec

- Numerical condition number diagnosis tool for MIP
  - Provides summary of LP condition numbers during Branch & Cut

- New ports
  - 32-bit x86 Solaris
  - IBM systems zLinux and pLinux

* on average, using 4 threads, compared to CPLEX 12.1 using 4 threads

- Drivers:
  - Improved heuristics
  - Improved cuts
  - Reduced parallel overhead
  - General software engineering
## CPLEX 12.2 MILP Enhancement – Details

<table>
<thead>
<tr>
<th>Min Solve Time (both)</th>
<th>Max Solve Time (both)</th>
<th># of Models</th>
<th>Avg Speed up</th>
<th>Wins / Losses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Second</td>
<td>1 Second</td>
<td>1060</td>
<td>1.0</td>
<td>44 / 78</td>
</tr>
<tr>
<td>1 Second</td>
<td>10 Seconds</td>
<td>594</td>
<td>1.0</td>
<td>198 / 242</td>
</tr>
<tr>
<td>10 Seconds</td>
<td>100 Sec</td>
<td>389</td>
<td>1.4</td>
<td>240 / 107</td>
</tr>
<tr>
<td>100 Sec</td>
<td>1000 Sec</td>
<td>308</td>
<td>1.7</td>
<td>212 / 74</td>
</tr>
<tr>
<td>1000 Sec</td>
<td>10000 Sec</td>
<td>207</td>
<td>2.3</td>
<td>148 / 42</td>
</tr>
<tr>
<td>1 Second</td>
<td>&gt; 10000</td>
<td>1627</td>
<td>1.5</td>
<td>891 / 492</td>
</tr>
</tbody>
</table>

A Win occurs when one version is at least 10% faster than the other.

**Comparison to CPLEX 12.1 (released July 2009)**
### CPLEX 12.2 Barrier LP Enhancement – Details

<table>
<thead>
<tr>
<th>Min Solve Time (both)</th>
<th>Max Solve Time (both)</th>
<th># of Models</th>
<th>Avg Speed up</th>
<th>Wins / Losses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Second</td>
<td>1 Second</td>
<td>1339</td>
<td>1%</td>
<td>50 / 2</td>
</tr>
<tr>
<td>1 Second</td>
<td>10 Seconds</td>
<td>300</td>
<td>18%</td>
<td>229 / 1</td>
</tr>
<tr>
<td>10 Seconds</td>
<td>100 Sec</td>
<td>189</td>
<td>16%</td>
<td>140 / 5</td>
</tr>
<tr>
<td>100 Sec</td>
<td>1000 Sec</td>
<td>146</td>
<td>15%</td>
<td>68 / 3</td>
</tr>
<tr>
<td>1000 Sec</td>
<td>&gt; 10000 Sec</td>
<td>44</td>
<td>14%</td>
<td>22 / 2</td>
</tr>
<tr>
<td>1 Second</td>
<td>&gt; 10000</td>
<td>635</td>
<td>16%</td>
<td>460 / 11</td>
</tr>
</tbody>
</table>

A Win occurs when one version is at least 10% faster than the other.

Comparison to CPLEX 12.1 (released July 2009)
CPLEX 12.2 MILP Enhancement – Historical perspective

- For comparison, here are some past marketing messages for CPLEX speedups in MIP:

  - CPLEX 7 (2000): 60% on “hard mixed integer problems”
  - CPLEX 8 (2002): 40% overall, 70% on “difficult problems”
  - CPLEX 9 (2003): 50% on “difficult customer models”
  - CPLEX 10 (2006): 35% overall, 70% on “particularly difficult models”
  - CPLEX 11 (2007): 15% under one minute, 3X on 1-60 minutes, 10X on one hour and up
  - CPLEX 12.0 (2009): 30% overall, 2X on 1000 seconds and up
  - CPLEX 12.2 (2010): 50% overall, 2.2X on 100 seconds and up

- The 12.2 release stacks up extremely favorably
Summary

- CPLEX 12 messages:
  - Interoperability
    - Excel connector
    - MATLAB connector
    - Python connector
  - MILP twice as fast on hard problems as ever before
    - Measurable performance boost on other problem classes too
  - Parallel algorithms available with default license
Thank you!

Questions?