Extensions to an Optimization Services Instance Language

Robert Fourer, Jun Ma
Industrial Engineering & Management Sciences
McCormick School of Engineering and Applied Science
Northwestern University
{4er, maj}@iems.northwestern.edu

Kipp Martin
Graduate School of Business
University of Chicago
kmartin@gsb.uchicago.edu

IFORS 2005
Honolulu, Hawaii — Thursday, July 14, 2005 — RA-14.3
Abstract

- Optimization problems of interest today go beyond the traditional linear, integer, quadratic, and smooth nonlinear types. A language for problem instances must be extended accordingly. This presentation describes prospective extensions to OSiL, our proposed language standard, in such areas as combinatorial optimization and constraint programming, stochastic programming, and semidefinite and cone programming.
Why Extensions?

Better describe problem instances
- Describe broader variety
- Describe more concisely
- Make more natural and understandable

More readily transform models and problems
- Modeling system → OSiL
- OSiL → Solver
Outline

*Piecewise-linear terms*

*Logic & combinatorial constraints*

*Complementarity constraints*

*Stochastic programming*

*Cone & semidefinite programming*
Piecewise-Linear Terms

Design considerations

- Univariate function of a numerical expression
- Defined by alternating breakpoints and slopes
  - Start and end with slopes
  - Value at zero is zero unless overridden
- Ordering of pieces must be unambiguous

Tentative extensions

- Series of \(<piece\>\) elements
  - one slope and one breakpoint attribute
  - last \(<piece\>\) has no breakpoint
  - sorted by breakpoint value
- Optional \(<level\>\) element gives value at zero
- A final element specifies the operand as an expression tree
Example

```xml
<piecewiseLinear>
  <piece slope="-2" breakpoint="-3"/>
  <piece slope="1" breakpoint="-2"/>
  <piece slope="0.333333" breakpoint="1"/>
  <piece slope="-1" breakpoint="2"/>
  <piece slope="1.25"/>
  <level value="0.666667"/>
  <var idx="7"/>
</piecewiseLinear>
```
Logic and Combinatorial Constraints

Design considerations

- Expression types
- Constraint types
- New operators

Examples of tentative extensions

- Logic operators
- Counting operator
- “All different” operator
- Variable indexed by a variable
Expression Types

Numerical
   ➢ Value is a number
       var Trans {ORIG, DEST} >= 0;

Logical
   ➢ Value is “true” or “false”

Object
   ➢ Value is a member of some set
       var JobForSlot {SLOTS} in JOBS;

Set
   ➢ Value is a set of numbers or objects:
       var MEMBERS {PROJECTS} within VOLUNTEERS;
Constraint Types

Range constraints

- $\text{lowerBound} \leq \text{numExpr} \leq \text{upperBound}$
- For one-sided constraint,
  $\text{lowerBound} = -\infty$ or $\text{upperBound} = +\infty$
- For equality, $\text{lowerBound} = \text{upperBound}$

Logic constraints

- $\text{logicExpr}$
- Logical
  $(\text{Mk}[i] = 0 \text{ and } \text{Mk}[i] = 0) \text{ or } \text{Mk}[i] + \text{Mk}[i] \geq \text{ld}$
- Counting
  atmost $\text{mxsrv} \{j \text{ in } D\} \ (\text{sum } \{p \text{ in PRD}\} \text{ Tr}[i,j,p] \geq 10)$
- Special-structure
  alldiff $\{j \text{ in } \text{Jobs}\} \ (\text{MachineForJob}[j])$
New Operators

Numerical-valued on constraints

- Counting
  \[ \text{count} \{ j \in D \} \left( \sum \{ p \in \text{PRD} \} \ Tr[i,j,p] \geq 10 \right) \]

Logic-valued on constraints

- Logical
  \[ (Mk[1] = 0 \text{ and } Mk[2] = 0) \text{ or } (Mk[1] + Mk[2] \geq 100) \]

- Counting
  \[ \text{atmost} \ mxsrv \{ j \in D \} \left( \sum \{ p \in \text{PRD} \} \ Tr[i,j,p] \geq 10 \right) \]

Special-structure (“global”)

- All-different
  \[ \text{alldiff} \{ j \in \text{Jobs} \} (\text{MachineForJob}[j]) \]

- Distribution
  \[ \text{numberof} \ 3 \ in \{ j \in 1..n\text{Jobs} \} \ (\text{MachineForJob}[j]) \]
New Operators (cont’d)

Indexing

- Variables in subscripts of parameters or variables

```
param mCLI integer > 0;
param nLOC integer > 0;
param srvCost {1..mCLI, 1..nLOC} > 0;
param bdgCost > 0;

var Serve {1..mCLI} integer >= 1, <= nLOC;
var Open {1..nLOC} integer >= 0, <= 1;

minimize TotalCost:
    sum {i in 1..mCLI} srvCost[i,Serve[i]] +
    bdgCost * sum {j in 1..nLOC} Open[j];

subject to OpenDefn {i in 1..mCLI}:
    Open[Serve[i]] = 1;
```
New Operators (cont’d)

Indexing

- Variables constrained by subscripts

```plaintext
set ABLE within {1..mCLI, 1..nLOC};
param srvCost {ABLE} > 0;

......
minimize TotalCost:
  sum {i in 1..mCLI} srvCost[i,Serve[i]] + ...
```

... (i, Serve[i]) must be in ABLE

With set operands

- Set valued: union, intersection, difference
- Numerical valued: cardinality
- Logic valued: membership, containment
- Special-structure: all-disjoint
Logic Extensions to OSiL

Design
- Use same “nonlinear” expression tree
- Define new nodes to represent new operators

Implementation
- Extend API to give solvers access to constraint expressions
Example: Logic Operators

\[(Mk[i] = 0 \text{ and } Mk[i] = 0) \text{ or } Mk[i] + Mk[i] \geq lbd\]

\[
<\text{or}>
  <\text{and}>
    <\text{and}>
      <\text{eq}>
        <\text{var} \text{idx}="23"/>
        <\text{number} \text{value}="0"/>
      </\text{eq}>
      <\text{eq}>
        <\text{var} \text{idx}="103"/>
        <\text{number} \text{value}="0"/>
      </\text{eq}>
    </\text{and}>
  <\text{geq}>
    <\text{plus}>
      <\text{var} \text{idx}="23"/>
      <\text{var} \text{idx}="103"/>
    </\text{plus}>
    <\text{number} \text{value}="150"/>
  </\text{geq}>
</\text{or}>
\]
Example: “at most” operator

\[
\text{atmost mxsrv \{j in D\} (sum \{p in PRD\} Tr[i,j,p] \geq lim[j])}
\]

```xml
<atMost>
  <number value="2"/>
  <geq>
    <sum>
      <var idx="20"/>
      <var idx="21"/>
      <var idx="22"/>
    </sum>
    <number value="10"/>
  </geq>
  <geq>
    <sum>
      <var idx="30"/>
      <var idx="31"/>
      <var idx="32"/>
    </sum>
    <number value="27"/>
  </geq>
  ...
</atMost>
```
Examples (cont'd)

\[ \text{alldiff \{j in Jobs\} (MachineForJob[j])} \]

\[
\begin{aligned}
&\text{<alldiff>}
&\quad \text{<var idx="27"/>}
&\quad \text{<var idx="37"/>}
&\quad \text{<var idx="47"/>}
&\text{</alldiff>}
\end{aligned}
\]

\[ \text{Open[Serve[7]] where mCLI = 40, nLOC = 15} \]

- Serve corresponds to Var[0], ..., Var[39]
- Open corresponds to Var[40], ..., Var[54]

\[
\begin{aligned}
&\text{<var>}
&\quad \text{<plus>}
&\quad \text{<number value="39"/>}
&\quad \text{<var idx="6"/>}
&\text{</plus>}
&\text{</var>}
\end{aligned}
\]
Complementarity

Definition

- Two inequalities must hold . . .
- At least one of them with equality

Applications

- Equilibrium problems in economics and engineering
- Optimality conditions for nonlinear programs, bi-level linear programs, etc.

Forms

- “Square” systems of complementarity conditions
  - # of variables = # of complementarity constraints + # of equality constraints
- Mathematical programs with complementarity constraints (MPCCs)
Classical Complementarity

LP with nonnegative variables

- Complementary slackness conditions

\[
\begin{align*}
\text{Primal Constr} \{i \in I\}: \\
\quad \sum_{j \in J} a[i, j] \times X[j] &= b[i]; \\
\text{Primal Bounds} \{j \in J\}: X[j] &\geq 0; \\
\text{Dual Constr} \{j \in J\}: \\
\quad \sum_{i \in I} Y[i] \times a[i, j] + Z[j] &= c[j]; \\
\text{Dual Bounds} \{i \in I\}: Z[i] &\geq 0; \\
\text{Complementarity} \{j \in J\}: \\
\quad X[j] = 0 &\quad \text{or} \quad Z[j] = 0;
\end{align*}
\]

- Multiplicative alternative

\[
\begin{align*}
\text{Complementarity} \{j \in J\}: \\
\quad X[j] \times Z[j] &= 0;
\end{align*}
\]
Mixed Complementarity

LP with bounded variables

- Complementary slackness conditions

\[
\text{PrimalConstr } \{i \in I\}: \sum_{j \in J} a[i, j] \cdot X[j] = b[i];
\]
\[
\text{PrimalBounds } \{j \in J\}: l[j] \leq X[j] \leq u[j];
\]
\[
\text{DualConstr } \{j \in J\}:
\quad \sum_{i \in I} Y[i] \cdot a[i, j] + Z[j] = c[j];
\]
\[
\text{Complementarity } \{j \in J\}:
\quad X[j] = l[j] \text{ implies } Z[j] \geq 0 \text{ and }
\quad X[j] = u[j] \text{ implies } Z[j] \leq 0 \text{ and }
\quad l[j] < X[j] < u[j] \text{ implies } Z[j] = 0;
\]

- Variational inequality alternative

\[
\text{Complementarity } \{j \in J\}:
\quad \forall \{Y[j] \text{ in interval}[l[j], u[j]]\} \quad (Y[j] - X[j]) \cdot Z[j] \geq 0;
\]
New complements Operator

LP with nonnegative variables

\[
\text{PrimalConstr } \{i \in I\}:
\sum \{j \in J\} \ a[i,j] \ * \ X[j] = b[i];
\]
\[
\text{DualConstr } \{j \in J\}:
\sum \{i \in I\} \ Y[i] \ * \ a[i,j] + Z[j] = c[j];
\]
\[
\text{Complementarity } \{j \in J\}:
X[j] >= 0 \text{ complements } Z[j] >= 0;
\]

LP with bounded variables

\[
\text{PrimalConstr } \{i \in I\}:
\sum \{j \in J\} \ a[i,j] \ * \ X[j] = b[i];
\]
\[
\text{DualConstr } \{j \in J\}:
\sum \{i \in I\} \ Y[i] \ * \ a[i,j] + Z[j] = c[j];
\]
\[
\text{Complementarity } \{j \in J\}:
l[j] <= X[j] <= u[j] \text{ complements } Z[j];
\]
\textbf{\ldots without Auxiliary Variable Z}[j]\ldots

\textit{LP with nonnegative variables}

\begin{itemize}
  \item PrimalConstr \{i in I\}:
    \[ \sum_{j in J} a[i,j] \times X[j] = b[i]; \]
  \item Complementarity \{j in J\}:
    \[ X[j] \geq 0 \text{ complements} \sum_{i in I} Y[i] \times a[i,j] \leq c[j]; \]
\end{itemize}

\textit{LP with bounded variables}

\begin{itemize}
  \item PrimalConstr \{i in I\}:
    \[ \sum_{j in J} a[i,j] \times X[j] = b[i]; \]
  \item Complementarity \{j in J\}:
    \[ l[j] \leq X[j] \leq u[j] \text{ complements} \]
    \[ c[j] - \sum_{i in I} Y[i] \times a[i,j]; \]
\end{itemize}
Nonlinear

Price-dependent demands

```plaintext
var Price {i in PROD};
var Level {j in ACT};

subject to Pri_Compl {i in PROD}:
    Price[i] >= 0 complements
    sum {j in ACT} io[i,j] * Level[j] >= demzero[i] - demrate[i] * Price[i];

subject to Lev_Compl {j in ACT}:
    Level[j] >= 0 complements
    sum {i in PROD} Price[i] * io[i,j] <= cost[j];
```

... not obviously an optimality condition for an optimization problem
From Applications

Prices of coal shipments

subject to delct \{cr in creg, u in users\}:

\[ 0 \leq ct[cr,u] \text{ complements} \]

\[ ctcost[cr,u] + cv[cr] \geq p["C",u]; \]

Height of membrane

subject to dv \{i in 1..M, j in 1..N\}:

\[ lb[i,j] \leq v[i,j] \leq ub[i,j] \text{ complements} \]

\[ (dy/dx) \ast (2\ast v[i,j] - v[i+1,j] - v[i-1,j]) \]
\[ + (dx/dy) \ast (2\ast v[i,j] - v[i,j+1] - v[i,j-1]) \]
\[ - c \ast dx \ast dy; \]

\[ \ldots \text{more at Complementarity Problem Net} \]
http://www.cs.wisc.edu/cpnet/
Operands: Always Two Inequalities

Two single inequalities

- single-ineq1 complements single-ineq2
  - Both inequalities must hold,
  - at least one at equality

One double inequality

- double-ineq complements expr
  - The double-inequality must hold, and
  - if at lower limit then $expr \geq 0$,
    - if at upper limit then $expr \leq 0$,
    - if between limits then $expr = 0$
Complementarity Extensions to OSiL

Design
- Introduce new `<complements>` element to expression tree
- Require two child nodes

Implementation
- Check for “two inequalities” requirement after the validation phase
Example

- \( p[1] \geq 0 \) complements
  
  \[
  400 \cdot h[0]^3 \cdot p[1] / \exp(1.416 \cdot p[1]) -
  400 \cdot h[1]^3 \cdot (p[2] - p[1]) / \exp(1.416 \cdot (p[2] + p[1])) +
  121.14 \cdot h[1] - 121.14 \cdot h[0] \geq 0;
  \]

\[
<\text{complements}>
  <\text{geq}>
    <\text{var} \text{idx}="25"/>
    <\text{number} \text{value}="0"/>
  </\text{geq}>
  <\text{geq}>
    <\text{sum}>
      <\text{times}>
        <\text{number} \text{value}="400"/>
        <\text{times}>
          <\text{power}>
            <\text{var} \text{idx}="47"/>
            <\text{number} \text{value}="3"/>
          </\text{power}>
          ...
        </\text{times}>
      </\text{times}>
    </\text{sum}>
    <\text{number} \text{value}="0"/>
  </\text{geq}>
</\text{complements}>

Example *(more completed)*

```xml
<complements>
  <geq>
    <var idx="25"/>
    <number value="0"/>
  </geq>
  <geq>
    <sum>
      <times>
        <number value="400"/>
        <times>
          <power>
            <var idx="47"/>
            <number value="3"/>
          </power>
          ...
        </times>
      </times>
      ...
      <var idx="47" coef="-121.14"/>
    </sum>
    <number value="0"/>
  </geq>
</complements>
```
Stochastic Programming

Core optimization problem

Stage information

Stochastic information
What about SMPS Format?

*Shares drawbacks of the MPS format for LPs, MIPs*

- Limited precision
- Limited name length
- Expensive to process
- Restricted to linear problems
- Not entirely standard

*Yet doesn't cover all problems of interest*

- See extensions at
  
  sba.management.dal.ca/profs/hgassmann/smps2.htm
OSiL Stochastic Extensions

XML-based format

- **Core** LP or MIP described using OSiL
- Multi-stage partitioning using `<stages>` element
- Stochastic characteristics using `<stochastic>` element
Example (Birge & Louveaux, *Intro to Stoch Prog*)

**General information for core problem**

```
<problemDescription>
  <source>FinPlan_JohnBirge</source>
  <maxOrMin>max</maxOrMin>
  <objConstant>0</objConstant>
  <numberObjectives>1</numberObjectives>
  <numberConstraints>4</numberConstraints>
  <numberVariables>8</numberVariables>
</problemDescription>
```
Example (cont’d)

Constraint & variable data for core problem

```xml
<problemData>
  <constraints>
    <con name="p1" lb="55" ub="55" />
    <con name="p2" lb="0" ub="0" />
    <con name="p3" lb="0" ub="0" />
    <con name="p4" lb="80" ub="80" />
  </constraints>
  <variables>
    <var name="x11" />
    <var name="x21" />
    <var name="x12" />
    <var name="x22" />
    <var name="x13" />
    <var name="x23" />
    <var objCoef="1" name="y" />
    <var objCoef="-4" name="w" />
  </variables>
</problemData>
```
Example *(cont’d)*

*Coefficient column-start positions for core problem*

```
<coefMatrix>
  <start>
    <el>0</el>
    <el>2</el>
    <el>4</el>
    <el>6</el>
    <el>8</el>
    <el>10</el>
    <el>12</el>
    <el>13</el>
  </start>
</coefMatrix>
```
Example (cont’d)

Coefficients for core problem

```
<rowIdx>
  <el>0</el>
  <el>1</el>
  <el>0</el>
  <el>1</el>
  <el>0</el>
  <el>1</el>
  <el>2</el>
  <el>1</el>
  <el>2</el>
  <el>1</el>
  <el>2</el>
  <el>1</el>
  <el>3</el>
  <el>2</el>
  <el>3</el>
  <el>3</el>
  <el>3</el>
  <el>3</el>
</rowIdx>

<value>
  <el>1</el>
  <el>-1.25</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1.25</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1.25</el>
  <el>1</el>
  <el>-1.14</el>
  <el>1</el>
  <el>-1</el>
  <el>1</el>
</value>

</coefMatrix>
```
OSiL <stages> Element

Partition variables & constraints by stage number

- Required attribute gives number of stages

<implicitOrder>

- When already ordered by stage
- Just say where each stage begins

<explicitOrder>

- Specify a stage number for each variable and constraint
Example (cont’d)

Stage information

```
<stages number="4">
  <implicitOrder>
    <el startRowIdx="1" startColIdx="1" />
    <el startRowIdx="2" startColIdx="3" />
    <el startRowIdx="3" startColIdx="5" />
    <el startRowIdx="4" startColIdx="7" />
  </implicitOrder>
</stages>
```
OSiL `<stochastic>` Element

`<explicitScenario>`
- Discrete distributions
- Discrete approximations to distributions

`<implicitScenario>`
- Continuous distributions

`<penalties>`
- Simple recourse

`<riskMeasures>`
- Chance constraints
- Probabilistic objectives
Alternatives for Explicit Scenarios

<scenarioPath>

- Every child represents a scenario as a **path**
  - from the root of the scenario tree
  - to one of its leaves
- First child is the **root scenario**
  - defined by the core problem
- Each subsequent child branches
  - directly from the root
  - or indirectly from some previous branch

*Every scenario has a parent*

- Only differences from parent are specified
Example (cont’d)

Explicit scenarios by path

```xml
<stochastic>
  <explicitScenario>
    <scenarioPaths>
      <rootScenario name="sc1" prob="0.125" base="none" stage="1">
        <el rowIdx="2" colIdx="1">1.25</el>
        <el rowIdx="2" colIdx="2">1.14</el>
        <el rowIdx="3" colIdx="3">1.25</el>
        <el rowIdx="3" colIdx="4">1.14</el>
        <el rowIdx="4" colIdx="5">1.25</el>
        <el rowIdx="5" colIdx="5">1.14</el>
        <bound varIdx="2" type="lower"/>
      </rootScenario>
      <scenario name="sc2" prob="0.125" parent="1" stage="4">
        <el rowIdx="4" colIdx="5">1.06</el>
        <el rowIdx="5" colIdx="5">1.12</el>
      </scenario>
      <scenario name="sc3" prob="0.125" parent="1" stage="3">
        <el rowIdx="3" colIdx="3">1.06</el>
        <el rowIdx="3" colIdx="4">1.12</el>
        <el rowIdx="4" colIdx="5">1.25</el> ...
      </scenario>
    </scenarioPaths>
  </explicitScenario>
</stochastic>
```
Alternatives for Explicit Scenarios

<scenarioTree>

- Every child represents a node of the scenario tree
  - by means of an <snode> element
- First <snode> corresponds to the root node
- Every <snode> may have <snode> children
  - defining branches of the tree

Every <snode> specifies the problem at its stage

- By listing differences from its parent, or
- By specifying a single-stage OSiL problem
Example (cont’d)

Explicit scenarios by tree node

```xml
<stochastic>
  <explicitScenario>
    <scenarioTree>
      <sNode prob="1" base="coreProgram">
        <changes>
          <el rowIdx="2" colIdx="1">1.06</el>
          <el rowIdx="2" colIdx="2">1.12</el>
        </changes>
      </sNode>
      <sNode prob="0.5" base="coreProgram">
        <sNode prob="0.5" base="coreProgram"/>
        <sNode prob="0.5" base="firstSibling">
          <changes>
            <el rowIdx="4" colIdx="5">1.06</el>
            <el rowIdx="5" colIdx="5">1.12</el>
          </changes>
        </sNode>
      </sNode>
    </scenarioTree>
  </explicitScenario>
</stochastic>
```
...
Alternatives for Implicit Scenarios

<distributions>
- Built-in univariate and multivariate
- User-defined

<stochasticElements>
- Series of <elementGroup> children
- Each <elementGroup> specifies a stochastic process

\[ Y_t = \sum_{i=1}^{p} M_i Y_{t-i} + \sum_{j=1}^{q} N_j v_{t-j} + c_t \]

- matrices \( M_i, N_j \)
- uncorrelated, identically distributed random vectors \( v_{t-j} \)
- constant \( c_t \)

\ldots an ARMA(p, q) process
<penalties> Element

Specifies penalties for violating constraints

- <simpleRecourse>: linear shortage and surplus penalties
- <robustOptimization>: quadratic penalties
- <piecewiseLinearQuadratic>
- <userDefinedPenalty>: shortage and surplus specified like other user-defined functions

... separate for each constraint
<riskMeasures> Element

Each child may take any of three types

- <simpleChance>
- <jointChance>
- <integratedChance> (see ...)

One (simple) or more (joint) rowIdx attributes

- rowIdx ≥ 0 implies chance constraint
  * probability that the constraint is satisfied
- rowIdx < 0 implies probabilistic objective
  * minimize or maximize the probability that
    the objective is ≥ or ≤ a constant
Cone and Semidefinite Programming

Design considerations

- Generalizations of $x \geq 0$
- Small number of cone types

Tentative extensions

- An element for each cone type
- Child elements and/or attributes indicate the variables involved
Cone Types

Second-order

- Quadratic cone
  \[ x_1^2 \geq \sum_{j=2}^{n} x_j^2 \]

- Rotated quadratic cone
  \[ 2x_1x_2 \geq \sum_{j=3}^{n} x_j^2 \]

Semidefinite

- Symmetric matrix \( X \) of variables is positive semi-definite
Example

Constraint & variable data for core problem

```xml
<cones>
  <quadraticCone>
    <el>1</el>
    <el>3</el>
  </quadraticCone>
  <quadraticCone>
    <el>2</el>
    <el mult="3" incr="1">4</el>
  </quadraticCone>
  <rotatedQuadraticCone startIndex="7" endIndex="9"/>
</cones>
```