Using COIN-OR to Solve the Uncapacitated Facility Location Problem

Ted Ralphs\textsuperscript{1}  Matthew Saltzman\textsuperscript{2}

\textsuperscript{1}COR@L Lab
Department of Industrial and Systems Engineering
Lehigh University

\textsuperscript{2}Department of Mathematical Sciences
Clemson University

Outline

1. The Uncapacitated Facility Location Problem
   - UFL Formulation
   - Cutting Planes

2. Developing a Solver
   - The ufl Class
   - COIN Tools
   - Putting It All Together

3. Additional Resources
The following are the input data needed to describe an instance of the uncapatitated facility location problem (UFL):

**Data**
- A set of depots $N = \{1, \ldots, n\}$, a set of clients $M = \{1, \ldots, m\}$,
- The transportation cost $c_{ij}$ to service client $i$ from $j$,
- The fixed cost $f_j$ for using depot $j$

**Variables**
- $x_{ij}$ is the amount of the demand for client $i$ satisfied from depot $j$
- $y_j$ is 1 if the depot is used, 0 otherwise
The following is a mathematical programming formulation of the UFL

<table>
<thead>
<tr>
<th>UFL Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimize</strong></td>
</tr>
</tbody>
</table>
| \[
\sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j \]
| **subject to** |
| \[
\sum_{j \in N} x_{ij} = d_i \quad \forall i \in M, \quad (2) \]
| \[
\sum_{i \in M} x_{ij} \leq (\sum_{i \in M} d_i) y_j \quad \forall j \in N, \quad (3) \]
| \[
y_j \in \{0, 1\} \quad \forall j \in N \quad (4)\]
| \[
0 \leq x_{ij} \leq d_i \quad \forall i \in M, j \in N \quad (5)\] |
Given the current LP solution, $x^*, y^*$, this method searches for violated logical constraints of the form

$$x_{ij} - d_j y_j \leq 0.$$ 

To generate such inequalities dynamically, we first we get the current solution. Then, we check if

$$x_{ij}^* - d_j y_j^* > \epsilon, \forall i \in M, j \in N.$$ 

We can also generate inequalities valid for generic MILPs.

If a violation is found, we can add the constraint to the current LP relaxation.
Tightening the Initial Formulation

Here is the basic loop for tightening the initial formulation using the dynamically generated inequalities from the previous slide.

**Solving the LP relaxation**

1. Form the initial LP relaxation and solve it to obtain \((\hat{x}, \hat{y})\).
2. Iterate
   1. Try to generate a valid inequality violated by \((\hat{x}, \hat{y})\).
   2. Optionally, try to generate an improved feasible solution by rounding \(\hat{y}\).
   3. Solve the current LP relaxation of the initial formulation to obtain \((\hat{x}, \hat{y})\).
   4. If \((\hat{x}, \hat{y})\) is feasible, STOP. Otherwise, go to Step 1.
The Uncapacitated Facility Location Problem
Developing a Solver
COIN Tools
Additional Resources

Data Members

C++ Class

```cpp
class UFL {
private:
    OsiSolverInterface * si;
    double * trans_cost; // c[i][j] -> c[xindex(i, j)]
    double * fixed_cost; // f[j]
    double * demand; // d[j]
    int M; // number of clients (index on i)
    int N; // number of depots (index in j)
    double total_demand; // sum{j in N} d[j]
    int *integer_vars;

    int xindex(int i, int j) {return i*N + j;}
    int yindex(int j) {return M*N + j;}
};
```

Ted Ralphs, Matthew Saltzman
Using COIN-OR to Solve the Uncapacitated Facility Location Problem
C++ Class

class UFL {
public:
    UFL(const char* datafile);
    ~UFL();
    void create_initial_model();
    double tighten_initial_model(ostream *os = &cout);
    void solve_model(ostream *os = &cout);
};
Open Solver Interface

- Uniform API for a variety of solvers: CBC, CLP, CPLEX, DyLP, FortMP, GLPK, Mosek, OSL, Soplex, SYMPHONY, the Volume Algorithm, XPRESS-MP supported to varying degrees.
- Manages multiple problem instances.
- Reads input from MPS or CPLEX LP files or constructs instances online using COIN-OR data structures.
- Inspects and modifies instances and saves to MPS or LP file.
- Sets solver parameters.
- Calls LP solver for LP or MIP LP relaxation.
- Manages interaction with CGL to add cutting planes and resolve.
- Calls MIP solver.
- Inspects solution.
A collection of cutting-plane generators and management utilities.

Interacts with OSI to inspect problem instance and solution information and get cuts added to the problem.

Cuts include:

- Combinatorial cuts: AllDifferent, Clique, KnapsackCover, OddHole
- Flow cover cuts
- Lift-and-project cuts
- Mixed integer rounding cuts
- General strengthening: DuplicateRows, Preprocessing, Probing, SimpleRounding
COIN LP Solver

- High-quality, efficient LP solver.
- Simplex and barrier algorithms. QP with barrier algorithm.
- Fine control through OSI or direct calls.
- Tight integration with CBC (COIN-OR Branch and Cut MIP solver).
The \texttt{initialize\_solver()} Method

Initializing the LP solver

```cpp
#include <OsiClpSolverInterface.hpp>
typedef OsiClpSolverInterface OsiXxxSolverInterface;
#include "CbcModel.hpp"
#endif

OsiSolverInterface* UFL::initialize\_solver() {
    OsiXxxSolverInterface* si = new OsiXxxSolverInterface();

    return si;
}
```
The create_initial_model() Method

Creating Rim Vectors

CoinIotaN(integer_vars, N, M * N);
CoinFillN(col_lb, n_cols, 0.0);

int i, j, index;

for (i = 0; i < M; i++) {
    for (j = 0; j < N; j++) {
        index = xindex(i, j);
        objective[index] = trans_cost[index] / demand[i];
        col_ub[index] = demand[i];
    }
}
CoinFillN(col_ub + (M*N), N, 1.0);
CoinDisjointCopyN(fixed_cost, N, objective + (M * N));
The \texttt{create\_initial\_model()} Method

Creating the Constraint Matrix

```cpp
CoinPackedMatrix * matrix =
    new CoinPackedMatrix(false, 0, 0);

matrix->setDimensions(0, n_cols);
for (i = 0; i < M; i++) { //demand constraints:
    CoinPackedVector row;
    for (j = 0; j < N; j++) row.insert(xindex(i, j), 1.0);
    matrix->appendRow(row);
}

for (j = 0; j < N; j++) { //linking constraints:
    CoinPackedVector row;
    row.insert(yindex(j), -1.0 * total_demand);
    for (i = 0; i < M; i++) row.insert(xindex(i, j), 1.0);
    matrix->appendRow(row);
}
```
Loading the Problem in the Solver

```cpp
si->loadProblem(*matrix, col_lb, col_ub,
                objective, row_lb, row_ub);
```
The tighten_initial_model() Method

Tightening the Relaxation—Custom Cuts

```cpp
cost double* sol = si->getColSolution();
int newcuts = 0, i, j, xind, yind;
for (i = 0; i < M; i++)
    for (j = 0; j < N; j++)
        xind = xindex(i, j); yind = yindex(j);

if (sol[xind] - (demand[i] * sol[yind]) >
    tolerance) { // violated constraint
    CoinPackedVector cut;
    cut.insert(xind, 1.0);
    cut.insert(yind, -1.0 * demand[i]);
    si->addRow(cut, -1.0 * si->getInfinity(), 0.0);
    newcuts++;
}
```

Ted Ralphs, Matthew Saltzman
Tightening the Relaxation—CGL Cuts

OsiCuts cutlist;
si->setInteger(integer_vars, N);
CglGomory * gomory = new CglGomory;
gomory->setLimit(100);
gomory->generateCuts(*si, cutlist);
CglKnapsackCover * knapsack = new CglKnapsackCover;
knapsack->generateCuts(*si, cutlist);
CglSimpleRounding * rounding = new CglSimpleRounding;
rounding->generateCuts(*si, cutlist);
CglOddHole * oddhole = new CglOddHole;
oddhole->generateCuts(*si, cutlist);
CglProbing * probe = new CglProbing;
probe->generateCuts(*si, cutlist);
si->applyCuts(cutlist);
Calling the Solver (Built-In MIP)

```cpp
si->setInteger(integer_vars, N);

si->branchAndBound();
if (si->isProvenOptimal()) {
    const double * solution = si->getColSolution();
    const double * objCoeff = si->getObjCoefficients();
    print_solution(solution, objCoeff, os);
}
else
    cerr << "B&B failed to find optimal" << endl;
return;
```
The \texttt{solve\_model()} Method

Calling the Solver (CLP Requires Separate MIP)

```cpp
CbcModel model(*si);
model.branchAndBound();
if (model.isProvenOptimal()) {
    const double * solution = model.getColSolution();
    const double * objCoeff = model.getObjCoefficients();
    print_solution(solution, objCoeff, os);
} else {
    cerr << "B&B failed to find optimal" << endl;
    return;
}
```
<project> is one of Osi, Cgl, Clp, Cbc, etc.

- **Project home pages:**
  https://projects.coin-or.org/<project> (Trac pages).

- **Documentation:** http://www.coin-or.org/Doxygen/<project> (Doxygen), http://www.coin-or.org/Clp/userguide/, http://www.coin-or.org/Cbc/userguide/

- **Mailing lists:** http://list.coin-or.org (see coin-discuss, coin-osi-devel, cgl, coin-lpsolver—note lists will be reorganized soon).