The OS Dip Solver:
A Generic Block-Angluar Decomposition Algorithm

Horand Gassmann, Jun Ma, Kipp Martin

September 18, 2010

Abstract

In this document we describe how to use the Decomposition in Integer Programming (Dip) package with the Optimization Services (OS) package. The code for this example is contained in the folder `ApplicationTemplates/osDip`.

1 Building and Testing the OS-Dip Example

Currently, the Decomposition in Integer Programming (**Dip** package is not a dependency of the Optimization Services (**OS**) package. In order to run the OS Dip solver it is necessary to download both the **OS** and **Dip** projects. Download order is irrelevant. In the discussion that follows we assume that for both **OS** and **Dip** the user has successfully completed a `configure`, `make`, and `make install`. We also assume that the user is working with the trunk version of both **OS** and **Dip.**

The OS Dip solver C++ code is contained in `TemplateApplication/osDip`. The `configure` will create a `Makefile` in the `TemplateApplication/osDip` folder. The `Makefile` must be edited to reflect the location of the **Dip** project. The `Makefile` contains the line

```
DIPPATH = /Users/kmartin/coin/dip-trunk/vpath-debug/
```

This setting assumes that there is a **lib** directory:

```
/Users/kmartin/coin/dip-trunk/vpath-debug/lib
```

with the **Dip** library that results from `make install` and an `include` directory

```
/Users/kmartin/coin/dip-trunk/vpath/include
```

with the **Dip** header files generated by `make install`. The user should adjust

```
/Users/kmartin/coin/dip-trunk/vpath/
```

to a path containing the **Dip** `lib` and `include` directories. After building the executable by executing the `make` command run the `osdip` application using the command:

```
./osdip --param osdip.parm
```

This should produce the following output.

```
FINISH SOLVE
Status= 0 BestLB= 16.00000   BestUB= 16.00000   Nodes= 1
SetupCPU= 0.01 SolveCPU= 0.10 TotalCPU= 0.11 SetupReal= 0.08
SetupReal= 0.12 TotalReal= 0.16
Optimal Solution
-------------------------
Quality = 16.00
0       1.00
1       1.00
12      1.00
13      1.00
14      1.00
15      1.00
17      1.00
```

If you see this output, life is good and things are working. If this doesn't work, I almost certainly did something stupid and forget to fix it. The file `osdip.parm` is a parameter file. The use of the parameter file is explained in Section 4.

2 The OS Dip Solver – Code Description

The OS Dip Solver uses **Dip** to implement a Dantzig-Wofe decomposition algorithm for block-angular integer programs.

2.1 General Philosophy

2.2 The code

The following C++ files are used.

OSDip˙Main.cpp

OSDipBlockSolver.cpp

OSDipBlockCoinSolver.cpp

OSDipInterface.cpp

OSDipApp.cpp

3 Defining the Problem Instance and Blocks

Here we describe how to use the OSoption stuff and OSInstance.

4 The Parameter File

Look at the osdip.parm file. You can see by commenting and uncommenting you can run one of three problems that will also get downloaded.

sp1.osil – a simple plant location problem spl2.osil – a second simple plant location problem genAssign.osil – a generalize assignment problem

The osol files (the option files) determine behavior. For example, if you use

osolFiles/spl1-b.osol

then the assingment constraints are the block constraints. If you use

osolFiles/spl1.osol

then the setup forcing constraints are the block constraints. This new example also exhibits the problems I filed ticked on.

5 Simple Plant/Lockbox Location Example

The problem minimizing the sum of the cost of capital due to float and the cost of operating the lock boxes is the problem.
Parameters:

$m-$ number of customers to be assigned a lock box

$n-$ number of potential lock box sites

$c_{ij}-$ annual cost of capital associated with serving customer j from lock box i

f_i- annual fixed cost of operating a lock box at location i

Variables:

$x_{ij}-$ a binary variable which is equal to 1 if customer j is assigned to lock box i and 0 if not

y_i- a binary variable which is equal to 1 if the lock box at location i is opened and 0 if not

The integer linear program for the lock box location problem is

$$
(LB) \qquad \begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{i=1}^{n} f_i y_i & & (1)\\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,m & & (2)\\
& x_{ij} - y_i \le 0, \qquad i = 1,\ldots,n,\ j = 1,\ldots,m & & (3)\\
& x_{ij},\ y_i \in \text{–}0,1\text{”}, \quad i = 1,\ldots,n,\ j = 1,\ldots,m. & & (4)
\end{aligned}
$$

The objective (1) is to minimize the sum of the cost of capital plus the fixed cost of operating the lock boxes. The requirement that every customer be assigned a lock box is modeled by constraint (2). Constraints (3) are forcing constraints and play the same role as constraint set (**??**) in the dynamic lot size model.

Location Example 1: A three-by-five example.

		CUSTOMER					
		1	2	3	4	5	FIXED COSTS
	1	2	3	4	5	7	2
PLANT	2	4	3	1	2	6	3
	3	5	4	2	1	3	3

Location Example 2: A three-by-three example.

$$\min 2x_{11} + x_{12} + x_{13} + x_{21} + 2x_{22} + x_{23} + x_{31} + x_{32} + 2x_{33} + y_1 + y_2 + y_3$$

$$
\begin{aligned}
\text{s.t.} \quad & x_{11} + x_{21} + x_{31} = 1 & \\
& x_{12} + x_{22} + x_{32} = 1 & \qquad Ax \ge b \text{ constraints}\\
& x_{13} + x_{23} + x_{33} = 1 &
\end{aligned}
$$

$$
\begin{aligned}
& x_{11} \le y_1 \le 1 \\
& x_{12} \le y_1 \le 1 \\
& x_{13} \le y_1 \le 1 \\
& x_{21} \le y_2 \le 1 \\
& x_{22} \le y_2 \le 1 \qquad Bx \ge b \text{ constraints}\\
& x_{23} \le y_2 \le 1 \\
& x_{31} \le y_3 \le 1 \\
& x_{32} \le y_3 \le 1 \\
& x_{33} \le y_3 \le 1 \\
& x_{ij}, y_i \ge 0,\ i = 1,\ldots,n,\ j = 1,\ldots,m.
\end{aligned}
$$

6 Generalized Assignment Problem Example

A problem that plays a prominent role in vehicle routing is the *generalized assignment problem.* The problem is to assign each of n tasks to m servers without exceeding the resource capacity of the servers.
Parameters:

$n-$ number of required tasks

$m-$ number of servers

$f_{ij}-$ cost of assigning task i to server j

b_j- units of resource available to server j

$a_{ij}-$ units of server j resource required to perform task i

Variables:

$x_{ij}-$ a binary variable which is equal to 1 if task i is assigned to server j and 0 if not

The integer linear program for the generalized assignment problem is

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}x_{ij} \tag{5}$$

$$(GAP) \qquad \text{s.t.} \quad \sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1, \ldots, n \tag{6}$$

$$\sum_{i=1}^{n} a_{ij}x_{ij} \leq b_j, \qquad j = 1, \ldots, m \tag{7}$$

$$x_{ij} \in –0, 1'', \; i = 1, \ldots, n, \; j = 1, \ldots, m. \tag{8}$$

The objective function (5) is to minimize the total assignment cost. Constraint (6) requires that each task is assigned a server. The requirement that the server capacity not be exceeded is given in (7).

The test problem

```
 min     2 X11 + 11 X12 + 7 X21 + 7 X22 + 20 X31 +
2 X32 + 5 X41 + 5 X42
s.t.
X11 + X12 =    1
X21 + X22 =    1
X31 + X32 =    1
X41 + X42 =    1

3 X11 + 6 X21 + 5 X31 + 7 X41 <=   13
2 X12 + 4 X22 + 10 X32 + 4 X42 <=   10
```

7 Implementing A Block Solver

Describe the Factory Code

8 Issues to Fix

- Enhance solveRelaxed to allow parallel processing of blocks. See ticket 30.
- Does not work when there are 0 integer variables. See ticket 31.
- Be able to set options in C++ code. See ticket 41.
- Problem with Alps bounds at node 0. See ticket 43
- Figure out how to use BranchEnforceInMaster or BranchEnforceInSubProb so I don't get the large bonds on the variables. See ticket 47.