

# Using Dip With OS

Horand Gassmann, Jun Ma, Kipp Martin

September 12, 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 `OS/examples/osDip`.

## 1 Using 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 **osDip** example it is necessary to download both the **OS** and **Dip** package. Download order is not relevant. In the discussion that follows we assume that for both **OS** and **Dip** the user has successfully completed a **configure**, **make**, and **make install**.

I have tested this on several simple plant location problems and generalized assignment problems. I have tested on both the Mac and Linux and it seems to be working. The configure step should generate a working Makefile for your platform. The OS project still does not include Dip as an external so you need to point to the Dip root. There is a variable in the generated Makefile,

DIPPATH =

that need to be filled in after configure. This "should" be the only line in the Makefile you need to edit. So on my machine, I have

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

and there exists a directory

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

with the necessary libs and a directory

DIPPATH = /Users/kmartin/coin/dip-trunk/vpath/include

with the necessary headers.

After building the executable run

./osdip -param osdip.parm

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.

spl1.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 assignment 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.

## 2 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

$$\min \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} + \sum_{i=1}^n f_i y_i \quad (1)$$

$$(LB) \quad \text{s.t.} \quad \sum_{i=1}^n x_{ij} = 1, \quad j = 1, \dots, m \quad (2)$$

$$x_{ij} - y_i \leq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m \quad (3)$$

$$x_{ij}, y_i \in \{0, 1\}, \quad i = 1, \dots, n, \quad j = 1, \dots, m. \quad (4)$$

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

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

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

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

$$\begin{aligned} x_{11} &\leq y_1 \leq 1 \\ x_{12} &\leq y_1 \leq 1 \\ x_{13} &\leq y_1 \leq 1 \\ x_{21} &\leq y_2 \leq 1 \\ x_{22} &\leq y_2 \leq 1 \\ x_{23} &\leq y_2 \leq 1 \\ x_{31} &\leq y_3 \leq 1 \\ x_{32} &\leq y_3 \leq 1 \\ x_{33} &\leq y_3 \leq 1 \end{aligned} \quad Bx \geq b \text{ constraints}$$

$$x_{ij}, y_i \geq 0, \quad i = 1, \dots, n, \quad j = 1, \dots, m.$$

### 3 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

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

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

$$\sum_{i=1}^n a_{ij} x_{ij} \leq b_j, \quad j = 1, \dots, m \quad (7)$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \dots, n, \quad j = 1, \dots, m. \quad (8)$$

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

$$\min \quad 2 X_{11} + 11 X_{12} + 7 X_{21} + 7 X_{22} + 20 X_{31} + 2 X_{32} + 5 X_{41} + 5 X_{42}$$

s.t.

$$X_{11} + X_{12} = 1$$

$$X_{21} + X_{22} = 1$$

$$X_{31} + X_{32} = 1$$

$$X_{41} + X_{42} = 1$$

$$3 X_{11} + 6 X_{21} + 5 X_{31} + 7 X_{41} \leq 13$$

$$2 X_{12} + 4 X_{22} + 10 X_{32} + 4 X_{42} \leq 10$$

## 4 Implementing A Block Solver

## 5 Issues

- 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.