Benders' decomposition with AMPL[1]

Stefano Nasini

Dept. of Statistics and Operations Research

Universitat Politécnica de Catalunya

Abstract. Benders' decomposition (named after Jacques F. Benders) is a decomposition technique that allows the solution of very large linear programs, provided that they have a special block structure. The algorithm adds new constraints as it progress towards the solution and is therefore referred to as a row generation approach, in accordance with its counterpart, Dantzig–Wolfe decomposition, which is referred to as a column generation approach. These notes provide an introductory explanation about the Benders' decomposition and two applications to mixed integer linear programs: the first one is a stochastic programming problem and the second a multi-commodity network flow problem.

1. Introduction

The Benders' decomposition is generally applied to linear problems, where the variables can be separated in two parts in such a way that, when fixing the value of one part the problem associated with the remaining ones has a straightforward solution. This is often the case of mixed integer linear problems, where the LP resulting from fixing the value of the integer variables is easily solvable, whereas the overall problem is difficult to deal with.

Consider the linear program:

[1] These notes have been obtained by revising and reformulating assignments of the courses in *Stochastic Programming* and *Great Scale Optimization*, of the Master degree program in Statistics and Operational Research (Faculty of Mathematics and Statistics, UPC, Barcelona).

$$\begin{cases} \min\limits_{x,y} \quad c^T x + q^T y \\ st \\ \quad Tx + Wy \geq b \\ \quad x \in X \\ \quad x \geq 0 \end{cases} \tag{1}$$

Problem (1) can be formulated in the equivalent form

$$\min_x \left\{ c^T x + \min_y \left\{ q^T y : Wy \geq (b - Tx),\ y \geq 0 \right\},\ x \in X \right\} \tag{2}$$

Dualizing the inner minimization we obtain

$$\min_x \left\{ c^T x + \max_\lambda \left\{ \lambda^T (b - Tx) : W^T \lambda \leq c,\ \lambda \geq 0 \right\},\ x \in X \right\} \tag{3}$$

A key observation is that feasible region of the dual formulation does not depend on the value of $x \in X$, which only affects the objective function. If the inner minimization problem in (2) is infeasible for some $x \in X$, then there must exist a $\lambda \geq 0$ verifying $W^T \lambda \leq c$ for which $\lambda^T (b - Tx) > 0$, so that a ray $\bar{v} = \lambda$ representing an improving direction in the dual polyhedron can be defined. By contrast, if the inner minimization problem in (2) is feasible for a given $x \in X$, then $\lambda^T (b - Tx) \leq 0$, for whatever $\lambda \geq 0$ verifying $W^T \lambda \leq c$, and $\bar{u} = \lambda$ defines an extreme point of the dual polyhedron. Thus, by enumerating extreme points and rays one can write problem (1) as follows:

$$\begin{cases} \min\limits_{z} \quad z \\ st \\ \quad z \geq c^T x + \bar{u}^T (b - Tx) \qquad \bar{u} \in P \\ \quad \bar{v}^T (b - Tx) \leq 0 \qquad \bar{v} \in R \\ \quad x \in X \end{cases} \tag{4}$$

where *P* and *R* represent the sets of extreme points and extreme rays respectively. These two sets are clearly unknown and a procedure to dynamically generate them is in order. This procedure can be constructed by considering that for each value of $x \in X$, the dual objective function represent lower bound of the primal, so that a sequence of cuts could be generating by iteratively solving the dual problem for a fixed $\bar{x} \in X$. At the same time, the sequence of $\bar{x}_1 \ldots \bar{x}_k$ is getting closer to the optimal solution as long as the set of generated cuts achieve *P* and *R.*

$$\begin{cases} \min\limits_{z} \quad z \\ st \\ \quad z \geq c^T x + \bar{u}_i^T (b - Tx) & i = 1 \ldots k_p \\ \quad \bar{v}_i^T (b - Tx) \leq 0 & i = 1 \ldots k_r \\ \quad x \in X \end{cases} \tag{5}$$

The two mathematical programs used to generate lower bounds of the objective value and sequence of $\bar{x}_1 \ldots \bar{x}_k$ are called *Sub-problem* and *Master Problem*, respectively. Hence, the procedure could be stated as follows:

Initialization

Let $k_r = 0$, $k_p = 0$;

Set an initial feasible solution $\bar{x}_k$;

Set $UB := \infty$, $LB := -\infty$;

while $UB - UL \geq \varepsilon$

Solve $\max\limits_{\lambda} \{ c^T \bar{x}_k + \lambda^T (b - T\bar{x}_k) : W^T \lambda \leq c, \lambda \geq 0 \}$ and

If $\bar{\lambda}$ is a ray (unbounded sub-problem)

Let $k_r = k_r + 1$

Let $\bar{v}_k = \bar{\lambda}$ (add a cut $\bar{v}_k^T (b - Tx) \leq 0$ to the master problem);

If else $\bar{\lambda}$ is an extreme point (bounded sub-problem)

Let $k_p = k_p + 1$

Let $\bar{u}_k = \bar{\lambda}$ (add a cut $z \geq c^T x + \bar{u}_k^T (b - Tx)$ to the master problem);

$UB = \min \{ UB, c^T \bar{x}_k + \bar{u}_k^T (b - T\bar{x}_k) \}$;

end if

Solve the master problem

$$\begin{cases} \min\limits_{z} \quad z \\ st \\ \quad z \geq c^T x + \bar{u}_i^T (b - Tx) & i = 1 \ldots k_p \\ \quad \bar{v}_i^T (b - Tx) \leq 0 & i = 1 \ldots k_r \\ \quad x \in X \end{cases}$$

Let $LB = \bar{z}$

end while

The computational performance of this procedure will be analyzed in the following sections by considering three well known applications.

2. Management problem with random parameters

Benders' decomposition is commonly applied to stochastic optimization problems with resources, where the matrix structure of the LPs has a straightforward column bipartition in the form of (1).

Consider an automatic coffee machine located in a public library. Every two days the supplier fills the machine up with coffee, milk and coins for change. Let x denote the amount of coins introduced in the machine. Since there is a risk for these coins to be stolen, the supplier must decide the correct amount x to avoid both excess and lack of coins. Clearly, the need of coins for change depends on the demand of milk and coffee. The supplier estimated that for each sold cap of coffee the expected need of coins is t^c and for each sold cap of milk the expected need of coins is t^m. The demand of milk and coffee in the machine is considered as discrete random variables ζ and γ with *s* values, that is to say, $\varsigma_1 \cdots \varsigma_s$ and $\gamma_1 \cdots \gamma_s$ with probabilities $p_1^c \cdots p_s^c$ and $p_1^m \cdots p_s^m$ respectively. The coin box of the coffee machine has a maximum capacity of u euros and the supplier estimated that the cost of leaving them in the coin box can be quantified as c euros per each deposited euro.

If the demand of milk and coffee is such that the required coins for change exceed x, then many customers will complain and the supplier shall have a cost of q for each euro missing in the coin box, that is, for each excess of demand. To solve this problem the supplier formulates the following mathematical program:

$$\text{canonical form} \begin{cases} \min \quad cx + E[\underbrace{Q(x,\zeta,\gamma)}] \\ st \\ \quad 0 \le x \le u \end{cases} \longrightarrow \begin{cases} \min \quad q(y^c + y^m) \\ st \\ \quad x + y^c \ge t^c \zeta \\ \quad x + y^m \ge t^m \gamma \\ \\ \quad y^c, y^m \ge 0 \end{cases}$$

Considering that the random variables ζ and γ can only take *s* values with probabilities $p_1^c \cdots p_s^c$ and $p_1^m \cdots p_s^m$ respectively. The extended form of the problem can be formulated as

$$
\text{Extended form}\begin{cases}
\min \quad c_1 x_1 + c_2 x_2 + \sum_{i=1}^{s}\sum_{j=1}^{s} p_i^c\, p_j^m\, q(y_{ij}^c + y_{ij}^m) & \\
st & \\
x + y_{ij}^c \geq t^c \zeta^i & i = 1..s,\ j = 1..s \\
x + y_{ij}^m \geq t^m \gamma^i & i = 1..s,\ j = 1..s \\
y_{ij}^c, y_{ij}^c \geq 0 & i = 1..s,\ j = 1..s \\
0 \leq x \leq u &
\end{cases} \qquad (6)
$$

The estimated data concerning the stochastic demand of milk and coffee, as well as the cost structure are the following:

c =	15
q =	9
u =	110
s =	3
t^c =	2
t^m =	1.5

i	p_i^c	ζ_i
1	0.25	40
2	0.50	80
3	0.25	120

j	p_i^m	γ_i
1	0.25	30
2	0.50	80
3	0.25	130

Let us see how the implementation of the problem (6) in AMPL.

COFFEE MATHINE MANAGEMENT PROBLEM (coffee.mod)

```
set SCENARIOS_c;
set SCENARIOS_m;

param d_c {SCENARIOS_c};
param d_m {SCENARIOS_m};
param p_c {SCENARIOS_c};
param p_m {SCENARIOS_m};

param c := 5;
param q := 10;
param u := 110;

var x >= 0, <= 110;
var y_c {SCENARIOS_c, SCENARIOS_m} >= 0;
var y_m {SCENARIOS_c, SCENARIOS_m} >= 0;

minimize Total_Cost:
x*c + sum{i in SCENATIO_m, j in SCENATIO_c}  p_c[i]* p_m[j]*q(y_c[i,j]+
y_m[i,j]);

subj to Demand_c {i in SCENARIO_c, j in SCENARIO_m}:
x + y_c[i,j] >= t_c*d_c[i];

subj to Demand_m {i in SCENARIO_c, j in SCENARIO_m}:
x + y_m[i,j] >= t_m*d_m[j];
```

COFFEE MATHINE MANAGEMENT PROBLEM (coffee.dat)

```
param: d_c :=
       1      40
       2      80
       3      120;

param: d_m :=
       1      30
       2      80
       3      130;

param: p_c :=
       1      0.25
       2      0.50
       3      0.25;

param: p_m :=
       1      0.25
       2      0.50
       3      0.25;
```

The following result corresponds to the direct solution of (6) by the simplex method (CPLEX 12.5.1.0).

RESULTS SOLVING THE WHOLE PROBLEM BY THE SIMPLEX METHOD

```
ampl: option solver cplex;
ampl: model coffee.mod;
ampl: data coffee.dat;
ampl: solve;
CPLEX 12.5.0.0: optimal solution; objective 2358.75
2 dual simplex iterations (0 in phase I)
ampl: display x;
x = 80

ampl: display y_c;
y_c :=
1 1     0
1 2     0
1 3     0
2 1    80
2 2    80
2 3    80
3 1   160
3 2   160
3 3   160;

ampl: display y_m;
y_m :=
1 1     0
1 2    40
1 3   115
2 1     0
2 2    40
2 3   115
3 1     0
3 2    40
3 3   115;

ampl:
```

To apply a Benders decomposition of this problem, let as consider the dualization of the inner problem, that is, the one corresponding to the second stage decision.

$$\text{canonical form} \begin{cases} \min \quad cx + E[\underbrace{Q(x,\zeta,\gamma)}] \\ st \\ \quad 0 \le x \le u \end{cases} \longrightarrow \begin{cases} \min \quad \sum_{i=1}^{s}\sum_{j=1}^{s} \lambda_{ij}^{c}(t^{c}\zeta_i - x) + \sum_{i=1}^{s}\sum_{j=1}^{s} \lambda_{ij}^{m}(t^{m}\gamma_j - x) \\ st \\ \quad \lambda_{ij}^{c} \le p_i^{c} p_j^{m} q \qquad i=1..s,\ j=1..s \\ \quad \lambda_{ij}^{m} \le p_i^{c} p_j^{m} q \qquad i=1..s,\ j=1..s \\ \quad \lambda_{ij}^{c}, \lambda_{ij}^{m} \ge 0 \qquad i=1..s,\ j=1..s \end{cases}$$

The matrix structure is the following

$$\begin{array}{cccccccccc|c|c}
x & +y_{11}^{c} & & & & & & & & & & t^{c}\zeta_1 \\
x & & +y_{12}^{c} & & & & & & & & & t^{c}\zeta_1 \\
x & & & +y_{13}^{c} & & & & & & & & t^{c}\zeta_1 \\
x & & & & +y_{21}^{c} & & & & & & & t^{c}\zeta_2 \\
x & & & & & +y_{22}^{c} & & & & & \ge & t^{c}\zeta_2 \\
x & & & & & & +y_{23}^{c} & & & & & t^{c}\zeta_2 \\
x & & & & & & & +y_{31}^{c} & & & & t^{c}\zeta_3 \\
x & & & & & & & & +y_{32}^{c} & & & t^{c}\zeta_3 \\
x & & & & & & & & & +y_{33}^{c} & & t^{c}\zeta_3
\end{array}$$

$$\begin{array}{cccccccccc|c|c}
x & +y_{11}^{m} & & & & & & & & & & t^{m}\gamma_1 \\
x & & +y_{12}^{m} & & & & & & & & & t^{m}\gamma_2 \\
x & & & +y_{13}^{m} & & & & & & & & t^{m}\gamma_3 \\
x & & & & +y_{21}^{m} & & & & & & & t^{m}\gamma_1 \\
x & & & & & +y_{22}^{m} & & & & & \ge & t^{m}\gamma_2 \\
x & & & & & & +y_{23}^{m} & & & & & t^{m}\gamma_3 \\
x & & & & & & & +y_{31}^{m} & & & & t^{m}\gamma_1 \\
x & & & & & & & & +y_{32}^{m} & & & t^{m}\gamma_2 \\
x & & & & & & & & & +y_{33}^{m} & & t^{m}\gamma_3
\end{array}$$

When fixing the value of x, the solution of the problem involving variables y_{ij}^{c} and y_{ij}^{c}, for $i=1..s,\ j=1..s$, is trivial. The AMPL implementation of this problem is the following.

SUB-PROBLEM (coffee1.mod**)**

```
param S := 3;

set SCENARIO_c := 1..S;
set SCENARIO_m := 1..S;

param X;
param d_c {SCENARIO_c};
param d_m {SCENARIO_m};
param p_c {SCENARIO_c};
param p_m {SCENARIO_m};
```

```
param t_c := 2;
param t_m := 1.5;
param q := 9;

var Lambda_c {SCENARIO_c, SCENARIO_m} >= 0;
var Lambda_m {SCENARIO_c, SCENARIO_m} >= 0;

maximize Dual_Cost:
      sum {i in SCENARIO_c, j in SCENARIO_m} Lambda_c[i,j]*(t_c*d_c[i]-X) +
sum {i in SCENARIO_c, j in SCENARIO_m} Lambda_m[i,j]*(t_m*d_m[j]- X);

subj to Dual_c {i in SCENARIO_c, j in SCENARIO_m}:
      Lambda_c[i,j] <= p_c[i]*p_m[j]*q;

subj to Dual_m {i in SCENARIO_c, j in SCENARIO_m}:
       Lambda_m[i,j] <= p_c[i]*p_m[j]*q;
```

MASTER PROBLEM (coffee2.mod**)**

```
param nCUT >= 0 integer;
param cut_type {1..nCUT} symbolic within {"point","ray"};

param lambda_c {SCENARIO_c, SCENARIO_m, 1..nCUT};
param lambda_m {SCENARIO_c, SCENARIO_m, 1..nCUT};
param u := 110;
param c := 15;

var x >= 0, <=u;
var z;

minimize Total_Cost:
c*x + z;

subj to Cut_Defn {k in 1..nCUT}:
if cut_type[k] = "point" then z >= sum {i in SCENARIO_c, j in SCENARIO_m}
lambda_c[i,j,k]*(t_c*d_c[i]-x) +
sum {i in SCENARIO_c, j in SCENARIO_m} lambda_m[i,j,k]*(t_m*d_m[j]- x);
```

The iterative procedure which allows implementing the course of generation of vertices and rays, on the one hand, and 1-stage solutions, on the other, is reported below.

BENDERS' DECOMPOSITION PROCEDURE (coffee.run**)**

```
model coffee.mod;
data coffee.dat;

option solver cplexamp;
option cplex_options 'mipdisplay 2 mipinterval 100 primal';

option omit_zero_rows 1;
option display_eps .000001;

problem Master: x, z, Total_Cost, Cut_Defn;
problem Sub: Lambda_c, Lambda_m, Dual_Cost, Dual_c, Dual_m;

suffix unbdd OUT;

let nCUT := 0;
let z := 0;
let X := 1;

param GAP default Infinity;

repeat { printf "\nITERATION %d\n\n", nCUT+1;
```

```
   solve Sub;
   printf "\n";

   if Sub.result = "unbounded" then { printf "UNBOUNDED\n";
      let nCUT := nCUT + 1;
      let cut_type[nCUT] := "ray";
      let {i in SCENARIO_c, j in SCENARIO_m} lambda_c[i,j,nCUT] :=
Lambda_c[i,j].unbdd;
      let {i in SCENARIO_c, j in SCENARIO_m} lambda_m[i,j,nCUT] :=
Lambda_m[i,j].unbdd;

      }
  else {
      let GAP := min (GAP, Dual_Cost - z);
      option display_1col 0;
      display Dual_Cost;
      display z;
      if Dual_Cost <= z + 0.00001 then break;
      let nCUT := nCUT + 1;
      let cut_type[nCUT] := "point";
       let {i in SCENARIO_c, j in SCENARIO_m} lambda_c[i,j,nCUT] :=
Lambda_c[i,j];
      let {i in SCENARIO_c, j in SCENARIO_m} lambda_m[i,j,nCUT] :=
Lambda_m[i,j];
      }

   printf "\nRE-SOLVING MASTER PROBLEM\n\n";

   solve Master;
   printf "\n";
   option display_1col 20;
   display x;

   let X := x;
};

option display_1col 0;
display Dual_Cost;
display x;
```

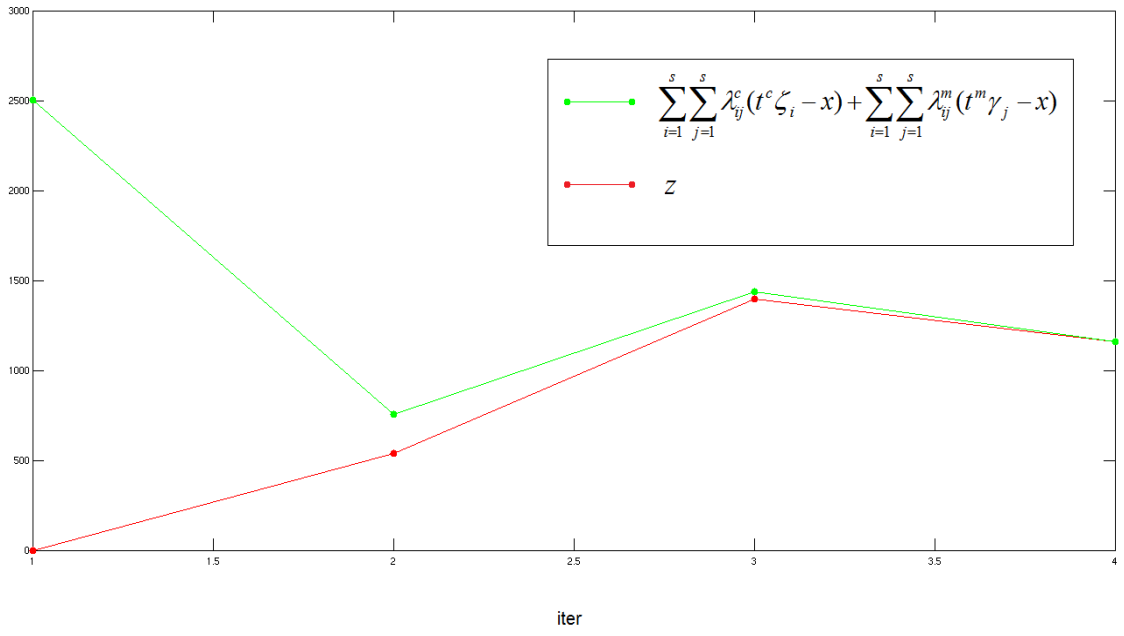
The following result corresponds to the iterative solution of the master problem and sub-problem, in accordance with the Benders' decomposition. The optimal solution is obtained in 4 iterations.

RESULTS SOLVING THE WHOLE PROBLEM BY B&B

```
ampl: include coffee.run;

ITERATION 1 ----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 2502
0 dual simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 2190
0 dual simplex iterations (0 in phase I)

ITERATION 2 ----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 2
```

```
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 753.75
6 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 2332.5
1 dual simplex iterations (0 in phase I)

ITERATION 3 ------------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 1434.375
3 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 2358.75
0 simplex iterations (0 in phase I)

ITERATION 4 ------------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 2
mipinterval 100
CPLEX 12.5.0.0: optimal solution; objective 1158.75
0 simplex iterations (0 in phase I)

Dual_Cost = 1158.75
z = 1158.75
x = 80
```

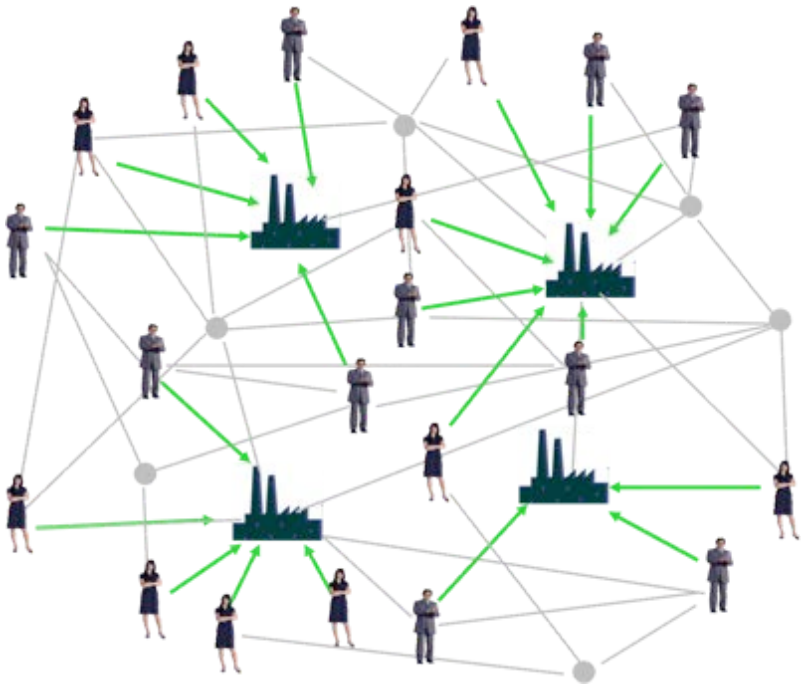
The illustration below shows the primal and dual objective functions associated to the gap along the iterations.



3. The facility location problem

The facility location problem is a mixed integer linear programming problem concerned with the optimal placement of facilities to minimize transportation costs while considering factors like avoiding placing hazardous materials near housing and competitors' facilities. Common application of this problem is the statistical cluster analysis, the assignment of costumers to servers, etc.

Suppose we have a set of m warehouses and a set of n stores. We wish to assign each store to a warehouse in such a way that the cost of delivering the commodities from the warehouses to the store is minimized, while satisfying the demand of each store.



The mathematical programming formulation of this problem requires two types of decision variables: $x_i \in \{0,1\}$, that is the binary decision concerning whether to build the warehouse *i*, for *i = 1…m*, and $y_{ij} \geq 0$, that is the amount of commodity that the warehouse *i* deliver to store *j*, for *i = 1…m, j = 1…n*. The parameters of the problem are the cost c_i of building the warehouse *i*, for *i = 1…m*, and the cost of delivering the commodity from warehouse *i* to store *j,* for *i = 1…m, j = 1…n.* The mathematical programming formulation is the following:

$$
\begin{cases}
\min\limits_{z} \quad \sum\limits_{i=1}^{m} c_i x_i + \sum\limits_{j=1}^{n}\sum\limits_{i=1}^{m} q_{ij} y_{ij} & \\
st & \\
\quad \sum\limits_{i=1}^{n} y_{ij} \le s_i x_i & i = 1 \ldots m \\
\quad \sum\limits_{j=1}^{m} y_{ij} = d_j & j = 1 \ldots n \\
\quad x_i \in \{0,1\} & i = 1 \ldots m \\
\quad y_{ij} \ge 0 & i = 1 \ldots m,\ j = 1 \ldots n
\end{cases} \tag{6}
$$

The AMPL implementation of the original problem is the following:

FACILITY LOCATION PROBLEM (trnlocd.mod**)**

```
set ORIG;   # shipment origins (warehouses)
set DEST;   # shipment destinations (stores)

param s {ORIG} > 0;
param d {DEST} > 0;

var x {ORIG} binary;    # 1 iff it is built
param c {ORIG} default 500000;

var y {ORIG,DEST} >= 0;  # amounts shipped
param q {ORIG,DEST} > 0;

minimize Total_Cost:
   sum {i in ORIG} c[i] * x[i] +
   sum {i in ORIG, j in DEST} q[i,j] * y[i,j];

subj to Supply {i in ORIG}:
   sum {j in DEST} y[i,j] <= s[i] * x[i];

subj to Demand {j in DEST}:
   sum {i in ORIG} y[i,j] = d[j];
```

The model must be saved in a file, which we called `trnlocd.mod`. A file containing the initialization of the parameters is also needed. The following file.dat is an example of data format for the models in `trnlocd.mod`.

DATA (trnloc1d.dat**)**

```
param: ORIG: supply :=
   1 23070    6 21090   11 22690   16 17110   21 16720
   2 18290    7 16650   12 19360   17 15380   22 21540
   3 20010    8 18420   13 22330   18 18690   23 16500
   4 15080    9 19160   14 15440   19 20720   24 15310
   5 17540   10 18860   15 19330   20 21220   25 18970;

param: DEST: demand :=
        A3    12000
        A6    12000
        A8    14000
        A9    13500
        B2    25000
        B4    29000;
```

```
param q:

          A3      A6      A8       A9      B2      B4  :=
    1   73.78   14.76   86.82    91.19   51.03   76.49
    2   60.28   20.92   76.43    83.99   58.84   68.86
    3   58.18   21.64   69.84    72.39   61.64   58.39
    4   50.37   21.74   61.49    65.72   60.48   56.68
    5   42.73   35.19   44.11    58.08   65.76   55.51
    6   44.62   39.21   44.44    48.32   76.12   51.17
    7   49.31   51.72   36.27    42.96   84.52   49.61
    8   50.79   59.25   22.53    33.22   94.30   49.66
    9   51.93   72.13   21.66    29.39   93.52   49.63
   10   65.90   13.07   79.59    86.07   46.83   69.55
   11   50.79    9.99   67.83    78.81   49.34   60.79
   12   47.51   12.95   59.57    67.71   51.13   54.65
   13   39.36   19.01   56.39    62.37   57.25   47.91
   14   33.55   30.16   40.66    48.50   60.83   42.51
   15   34.17   40.46   40.23    47.10   66.22   38.94
   16   41.68   53.03   22.56    30.89   77.22   35.88
   17   42.75   62.94   18.58    27.02   80.36   40.11
   18   46.46   71.17   17.17    21.16   91.65   41.56
   19   56.83    8.84   83.99    91.88   41.38   67.79
   20   46.21    2.92   68.94    76.86   38.89   60.38
   21   41.67   11.69   61.05    70.06   43.24   48.48
   22   25.57   17.59   54.93    57.07   44.93   43.97
   23   28.16   29.39   38.64    46.48   50.16   34.20
   24   26.97   41.62   29.72    40.61   59.56   31.21
   25   34.24   54.09   22.13    28.43   69.68   24.09;
```

The following result corresponds to the direct solution by B&B (CPLEX 12.5.1.0).

RESULTS SOLVING THE WHOLE PROBLEM BY B&B

```
ampl: option solver cplex;
ampl: model trnloc1.mod;
ampl: data trnloc1.dat;
ampl: solve;

CPLEX 12.5.1.0: optimal integer solution; objective 5735206.9
1338 MIP simplex iterations
272 branch-and-bound nodes

ampl: display x;

x [*] :=
 1 0    4 0    7 0   10 0   13 0   16 0   19 0   22 1   25 1
 2 0    5 0    8 0   11 0   14 0   17 1   20 1   23 0
 3 0    6 0    9 0   12 0   15 0   18 1   21 0   24 1;

ampl: display y;

y [*,*]
:     A3     A6      A8     A9      B2      B4      :=
1       0       0       0       0       0       0
2       0       0       0       0       0       0
3       0       0       0       0       0       0
4       0       0       0       0       0       0
5       0       0       0       0       0       0
6       0       0       0       0       0       0
7       0       0       0       0       0       0
8       0       0       0       0       0       0
9       0       0       0       0       0       0
10      0       0       0       0       0       0
11      0       0       0       0       0       0
12      0       0       0       0       0       0
```

```
13      0       0       0       0       0       0
14      0       0       0       0       0       0
15      0       0       0       0       0       0
16      0       0       0       0       0       0
17      0       0    8810       0       0     960
18      0       0    5190   13500       0       0
19      0       0       0       0       0       0
20      0   12000       0       0    9220       0
21      0       0       0       0       0       0
22   5760       0       0       0   15780       0
23      0       0       0       0       0       0
24   6240       0       0       0       0    9070
25      0       0       0       0       0   18970;
```

Let us now turn to the Benders' decomposition and consider the equivalent form of (6)

$$\begin{cases} \min \quad \sum_{i=1}^{m} c_i x_i + \underbrace{Q(x_1 \dots x_m)}_{\longrightarrow} \\ st \\ \quad x_i \in \{0,1\} \qquad i = 1 \dots m \end{cases} \longrightarrow \begin{cases} \min\limits_{z} \quad \sum_{j=1}^{n}\sum_{i=1}^{m} q_{ij} y_{ij} \\ st \\ \quad \sum_{i=1}^{n} y_{ij} \le s_i x_i \qquad i = 1 \dots m \\ \quad \sum_{j=1}^{m} y_{ij} = d_j \qquad j = 1 \dots n \\ \quad y_{ij} \ge 0 \qquad i = 1 \dots m,\ j = 1 \dots n \end{cases}$$

Dualizing the inner minimization, we obtain

$$\begin{cases} \min \quad \sum_{i=1}^{m} c_i x_i + \underbrace{Q(x_1 \dots x_m)}_{\longrightarrow} \\ st \\ \quad x_i \in \{0,1\} \qquad i = 1 \dots m \end{cases} \longrightarrow \begin{cases} \max\limits_{\lambda,\mu} \quad \sum_{i=1}^{m} (s_i x_i)\lambda_i + \sum_{j=1}^{n} d_j \mu_j \\ st \\ \quad \lambda_i + \mu_j \le q_{ij} \qquad i = 1 \dots m,\ j = 1 \dots n \\ \quad \lambda_i \le 0 \end{cases}$$

Thus, the sub-problem and the master problem are respectively the following

$$\text{SP} \begin{cases} \max\limits_{\lambda,\mu} \quad \sum_{i=1}^{m} (s_i x_i)\lambda_i + \sum_{j=1}^{n} d_j \mu_j \\ st \\ \quad \lambda_i + \mu_j \le q_{ij} \quad i = 1 \dots m,\ j = 1 \dots n \\ \quad \lambda_i \le 0 \end{cases}$$

$$
\text{MP}\begin{cases}
\min\limits_{z} \quad z & \\
st & \\
\quad z \geq \sum\limits_{i=1}^{m} c_i x_i + \left(\sum\limits_{i=1}^{m} (s_i x_i)\lambda_i^p + \sum\limits_{j=1}^{n} d_j \mu_j^p \right) & i = 1 \ldots k_p \\
\quad \left(\sum\limits_{i=1}^{m} (s_i x_i)\lambda_i^r + \sum\limits_{j=1}^{n} d_j \mu_j^r \right) \leq 0 & i = 1 \ldots k_r \\
\quad x_i \in \{0,1\} & i = 1 \ldots m
\end{cases}
$$

Note that λ_i might be interpreted as the price of supplying s_i unit of commodity from the warehouse *i* and μ_j as the price for demanding d_j unit of the commodity from the store *j*.

The AMPL implementation of master problem and sub-problem associated to the Benders' decomposition are the following

SUB-PROBLEM (trnloc1d.mod)

```
set ORIG;    # shipment origins (warehouses)
set DEST;    # shipment destinations (stores)

param s {ORIG} > 0;
param d {DEST} > 0;
param c {ORIG} > 0;
param q {ORIG,DEST} > 0;
param X {ORIG} binary;          # = 1 iff warehouse built at i

var mu {ORIG} <= 0;
var lambda {DEST};

maximize Dual_Ship_Cost:
   sum {i in ORIG} Mu[i] * (s[i]*X[i]) +
   sum {j in DEST} lambda[j] * d[j];

subj to Dual_Ship {i in ORIG, j in DEST}:
   mu[i] + lambda[j] <= q[i,j];
```

MASTER PROBLEM (trnloc1d.mod)

```
param nCUT >= 0 integer;
param cut_type {1..nCUT} symbolic within {"point","ray"};
param lambda {ORIG,1..nCUT} <= 0.000001;
param mu {DEST,1..nCUT};

var x {ORIG} binary;            # = 1 iff warehouse built at i
var z;

minimize Total_Cost:
   sum {i in ORIG} c[i] * x[i] + z;

subj to Cut_Defn {k in 1..nCUT}:
   if cut_type[k] = "point" then z >=
      sum {i in ORIG} lambda[i,k] * s[i]*x[i] +
      sum {j in DEST} mu[j,k]* d[j];
```

The `file.run` which implements the iterative procedure associated to the Benders' decomposition is the following:

BENDERS' DECOMPOSITION PROCEDURE

```
model trnloc1d.mod;
data trnloc1.dat;

option solver cplexamp;
option cplex_options 'mipdisplay 2 mipinterval 100 primal';
option omit_zero_rows 1;
option display_eps .000001;

problem Master: x, z, Total_Cost, Cut_Defn;
problem Sub: lambda, mu, Dual_Ship_Cost, Dual_Ship;

suffix unbdd OUT;

let nCUT := 0;
let Max_Ship_Cost := 0;
let {i in ORIG} build[i] := 1;

param GAP default Infinity;

repeat { printf "\nITERATION %d\n\n", nCUT+1;

   solve Sub;
   printf "\n";

   if Sub.result = "unbounded" then { printf "UNBOUNDED\n";
      let nCUT := nCUT + 1;
      let cut_type[nCUT] := "ray";
      let {i in ORIG} supply_price[i,nCUT] := Supply_Price[i].unbdd;
      let {j in DEST} demand_price[j,nCUT] := Demand_Price[j].unbdd;
      }

   else {
      if Dual_Ship_Cost <= Max_Ship_Cost + 0.00001 then break;

      let GAP := min (GAP, Dual_Ship_Cost - Max_Ship_Cost);
      option display_1col 0;
      let nCUT := nCUT + 1;
      let cut_type[nCUT] := "point";
      let {i in ORIG} supply_price[i,nCUT] := Supply_Price[i];
      let {j in DEST} demand_price[j,nCUT] := Demand_Price[j];
      }

   printf "\nRE-SOLVING MASTER PROBLEM\n\n";

   solve Master;
   printf "\n";
   option display_1col 20;
   let {i in ORIG} build[i] := Build[i];
};

option display_1col 0;
display Dual_Ship;
```

Let us analyze the computational performance of the two ways of solving the facility location problem: the direct solution of the whole problem by B&B (CPLEX 12.5.1.0), and the Benders' decomposition. The following result corresponds to the iterative solution of the master problem and sub-problem, in accordance with the Benders' decomposition.

RESULTS SOLVING THE WHOLE PROBLEM BY B&B

```
ampl: include trnloc1d.run;

ITERATION 1 ----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2661907.9
30 dual simplex iterations (11 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution; objective 2876165
0 MIP simplex iterations
0 branch-and-bound nodes

ITERATION 2 ----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: unbounded problem.
23 simplex iterations (0 in phase I)
variable.unbdd returned

UNBOUNDED

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5188260.8
1 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 482.625, relmipgap = 9.30226e-05

ITERATION 3 ----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 4397580.4
28 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5624373.2
5425 MIP simplex iterations
6423 branch-and-bound nodes
absmipgap = 55931.8, relmipgap = 0.00994453
```

```
ITERATION 4 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 3847883.3
25 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
468 MIP simplex iterations
252 branch-and-bound nodes
absmipgap = 54380.6, relmipgap = 0.00960464

ITERATION 5 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2818044.7
23 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
217 MIP simplex iterations
65 branch-and-bound nodes
absmipgap = 53584.9, relmipgap = 0.00946411

ITERATION 6 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2757807.5
4 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
240 MIP simplex iterations
62 branch-and-bound nodes
absmipgap = 53500.8, relmipgap = 0.00944925

ITERATION 7 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2892428.2
```

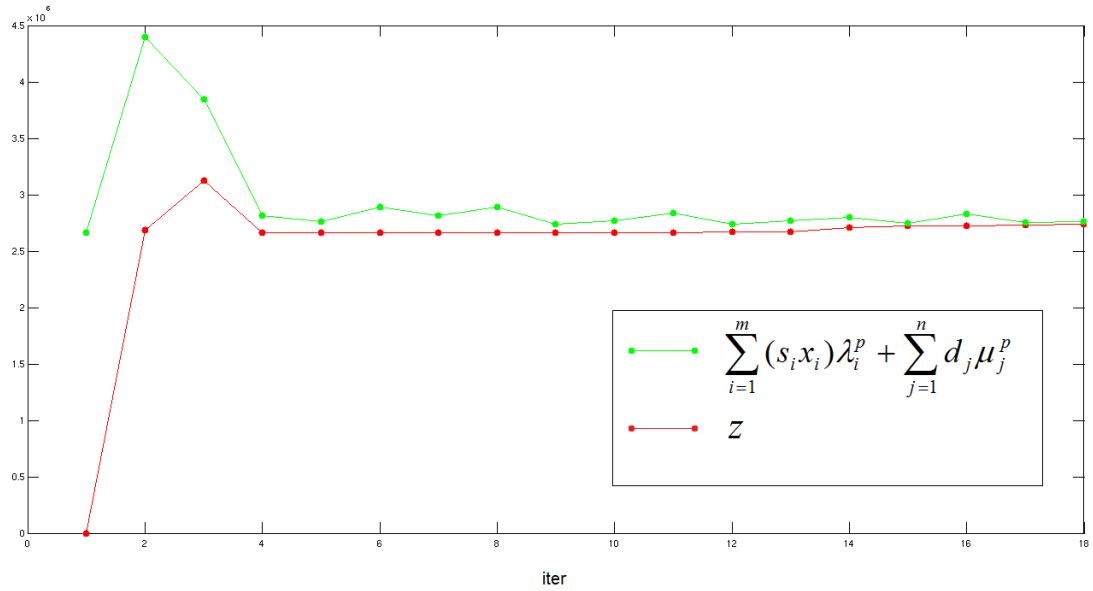
```
20 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
257 MIP simplex iterations
59 branch-and-bound nodes
absmipgap = 49963.8, relmipgap = 0.00882455

ITERATION 8 ------------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2811044.9
17 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
227 MIP simplex iterations
51 branch-and-bound nodes
absmipgap = 56233.8, relmipgap = 0.00993196

ITERATION 9 ------------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2886397.5
13 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
240 MIP simplex iterations
49 branch-and-bound nodes
absmipgap = 39820.9, relmipgap = 0.00703313

ITERATION 10 -----------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2735206.9
14 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
```

```
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
209 MIP simplex iterations
43 branch-and-bound nodes
absmipgap = 44304.5, relmipgap = 0.00782501

ITERATION 11 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2767331.8
14 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5661907.9
289 MIP simplex iterations
69 branch-and-bound nodes
absmipgap = 55142.2, relmipgap = 0.00973916

ITERATION 12 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2838645.9
8 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5670096.9
292 MIP simplex iterations
59 branch-and-bound nodes
absmipgap = 54973.2, relmipgap = 0.00969528

ITERATION 13 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2740440.4
22 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5673021.1
359 MIP simplex iterations
81 branch-and-bound nodes
absmipgap = 52274.3, relmipgap = 0.00921454
```

```
ITERATION 14 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2766209.9
17 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5711425.1
329 MIP simplex iterations
64 branch-and-bound nodes
absmipgap = 54662, relmipgap = 0.00957064

ITERATION 15 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2800953.4
15 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5723554.1
385 MIP simplex iterations
75 branch-and-bound nodes
absmipgap = 55830.3, relmipgap = 0.00975449

ITERATION 16 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2745154.1
8 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5726687.5
371 MIP simplex iterations
84 branch-and-bound nodes
absmipgap = 54887.2, relmipgap = 0.00958447

ITERATION 17 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2830540.5
```

```
9 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5734216.7
435 MIP simplex iterations
87 branch-and-bound nodes
absmipgap = 55956.2, relmipgap = 0.00975829

ITERATION 18 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2755659.9
8 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5735642.9
471 MIP simplex iterations
90 branch-and-bound nodes
absmipgap = 56132, relmipgap = 0.00978652

ITERATION 19 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2764530.5
5 simplex iterations (0 in phase I)

RE-SOLVING MASTER PROBLEM

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal integer solution within mipgap or absmipgap; objective
5735206.9
323 MIP simplex iterations
75 branch-and-bound nodes
absmipgap = 55482.5, relmipgap = 0.00967401

ITERATION 20 ------------------------------------------------------------

CPLEX 12.5.0.0: mipdisplay 0
mipinterval 100
primal
mipgap = 0.01
CPLEX 12.5.0.0: optimal solution; objective 2735206.9
14 simplex iterations (0 in phase I)
```

The procedure converges in 20 iterations. The illustration below shows the primal and dual objective functions, associated to the gap along the iterations. Red line represents the value of the master problem and green line the value of the sub-problem.



Notice that there is initial rapid convergence of the bounds to the optimal objective function value, followed by very slow convergence thereafter. This is a typical pattern that is observed in decomposition methods in general.

4. References

J.F.Benders. *Partitioning procedures for solving mixed variable programming problems.* Numerische Mathematik 4, 238-252, 1962.

D. Anderson. *Models for determining least-cost investments in electricity supply*. The Bell Journal of Economics, 3:267–299, 1972.

D. Bertsimas and J. Tsitisklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, 1997.

A.J. Conejo, E. Castillo, R. Minguez, R. García Bertrand, *Decomposition Thechniques in Mathematical Programming*, Springer, 2006.