

# **Practical Large Scale Optimization with illustrations in Matlab**

**S. Elhedhli**

Last updated: Feb. 2025

**Book Draft. Do not distribute without permission from the author.**

# Contents

<b>2 Lagrangean Relaxation</b>	<b>2</b>
2.1 Relaxation and Bounding . . . . .	2
2.2 Lagrangean relaxation . . . . .	2
2.3 Finding the best Lagrangean multipliers . . . . .	3
2.3.1 Subgradient optimization . . . . .	4
2.3.2 Constraint Generation/Cutting Plane Method: . . . . .	5
2.4 The relationship to Dantzig-Wolfe decomposition: Branch-and-price . . . . .	8

# Chapter 2

## Lagrangean Relaxation

### 2.1 Relaxation and Bounding

For many difficult large scale optimization problems, including mixed integer programs, the only approach to deal with the difficulty and the size of the problem, is to solve a relaxed version of the problem. Relaxation can take different forms

- LP relaxation: ignores integer requirements
- Constraint relaxation: Removes some constraints all together
- Lagrangean relaxation: removes constraints and penalizes them in the objective function

A solution to the relaxed problem achieves a better objective than the relaxed problem. Therefore, it constitutes a lower bound for a minimization problem, and an upper bound for a maximization problem.

These bounds are very useful in assessing the proximity of a feasible solution to the optimal. Feasible solutions are usually found heuristically. This is illustrated below

$$\begin{array}{c|c|c} \text{lower bound} & \leq & \text{Optimal objective} \\ \text{based on Relaxation} & & \text{of a minimization problem} \\ \hline & & \leq \\ & & \text{upper bound} \\ & & \text{based on Heuristics} \end{array}$$

Example:

The bin packing problem with item lengths  $D_l$  and bin capacity  $V$ .

- The LP lower bound is  $\sum D_l/V$ .
- A combinatorial lower bound can be given by ignoring the capacity constraints and solving the resulting LP (also IP).
- A Lagrangean lower bound can be obtained by relaxing the assignment constraints.
- Note:  $\lceil \sum D_l/V \rceil$  is a better bound than  $\sum D_l/V$ .

### 2.2 Lagrangean relaxation

Consider the following example from <sup>1</sup>:

$$\begin{aligned} \max \quad & 16x_1 + 10x_2 + 4x_4 \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_1 + x_2 \leq 1 \quad \rightarrow u_1 \geq 0 \\ & x_3 + x_4 \leq 1 \quad \rightarrow u_2 \geq 0 \\ & x_i \in \{0, 1\}, \quad i = 1, 2, 3, 4. \end{aligned}$$

<sup>1</sup>M. Fisher, An Applications Oriented Guide to Lagrangian Relaxation, Interfaces vol 15 No 2 pp 10-21 (1985).

Let us relax the second and the third constraints with Lagrangean multipliers  $u_1$  and  $u_2$ . The resulting problem is

$$\begin{aligned} \max \quad & 16x_1 + 10x_2 + 4x_4 + u_1(1 - x_1 - x_2) + u_2(1 - x_3 - x_4) \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_i \in \{0, 1\}, \quad i = 1, 2, 3, 4. \end{aligned}$$

which, by rearranging terms, is equivalent to:

$$\begin{aligned} z_{SP} = \max \quad & (16 - u_1)x_1 + (10 - u_1)x_2 + (0 - u_2)x_3 + (4 - u_2)x_4 \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_i \in \{0, 1\}, \quad i = 1, 2, 3, 4. \end{aligned}$$

As  $u_1 + u_2$  do not effect the optimization in  $x$ . For any given values of  $u_1 \geq 0, u_2 \geq 0$ , a **Lagrangean upper bound** is given by  $UB(u_1, u_2) = u_1 + u_2 + z_{SP}$ . The best such bound is the **Lagrangean bound**, which is the solution of **Lagrangean dual problem**:

$$\min_{u_1 \geq 0, u_2 \geq 0} \{u_1 + u_2 + z_{SP}\}.$$

For the above example, let us find a Lagrangean upper bound for different values of  $u_1, u_2$ . Let us start with  $u_1 = u_2 = 0$ . The subproblem is

$$\begin{aligned} z_{SP} = \max \quad & 16x_1 + 10x_2 + 4x_4 \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_i \in \{0, 1\}, \quad i = 1, 2, 3, 4. \end{aligned}$$

whose solution is  $x_1 = 1; x_2 = 1; x_3 = 0, x_4 = 0$  with an objective of 26. The corresponding bound is  $26 + 0 + 0 = 26$ . The following table lists bounds for other values of  $u_1$  and  $u_2$ .

$u_1$	$u_2$	SP objective	$x_1$	$x_2$	$x_3$	$x_4$	$z_{SP}$	UB
0	0	$16x_1 + 10x_2 + 4x_4$	1	1	0	0	26	26
20	20	$-4x_1 - 10x_2 - 20x_3 - 16x_4$	0	0	0	0	0	40
13	0	$3x_1 - 3x_2 + 4x_4$	0	0	0	1	4	17

According to the table,  $u_1 = 13, u_2 = 0$  gave a better bound than the other listed combinations. In addition, the solution of the corresponding subproblem  $x_1 = 0; x_2 = 0; x_3 = 0, x_4 = 1$  is feasible to the original problem. Its corresponding objective is 4, which provides a lower bound.

LB=4 (obtained from a feasible solution)	$\leq$	Optimal	$\leq$	UB=17 (obtained from Lagrangean relaxation)
--	--------	---------	--------	---

The gap for the above solution is  $\frac{\|4-17\|}{17} = 76.74\%$  indicating that it may be very far from the optimal.

## 2.3 Finding the best Lagrangean multipliers

To find the Lagrangean multipliers that yield the sharpest bound, we need to solve the Lagrangean dual problem:

$$\min_{u_1 \geq 0, u_2 \geq 0} \{u_1 + u_2 + z_{SP}\}.$$

The Lagrangean dual function  $LR(u_1, u_2) = u_1 + u_2 + z_{SP}$  is in fact convex. It is continuous, but non-differentiable at points where  $z_{SP}$  has multiple optimal solutions. One approach to solve it is to use a subgradient method:

### 2.3.1 Subgradient optimization

---

Start with an initial solution  $u^0$ .  
At iteration  $k$ , update the solution:  
choose a search direction  $d^k = -\text{subgradient}$   
choose a step size  $t^k$   
update  $u^k = \max\{0, u_1^{k-1} + t^k d^k\}$

---

The subgradient is given by

$$g^k = \begin{bmatrix} 1 - x_1^k - x_2^k \\ 1 - x_3^k - x_4^k \end{bmatrix}.$$

and the search direction  $d^k = -g^k$ . The step size should be a good compromise between small steps that will guarantee that one does not step over the optimal solution , and large ones that converge quickly. Ideally, big steps should be taken when far from the optimal solution, and short ones when close to it. In a result by Held, Wolfe and Crowder (1974)<sup>2</sup> it was shown that as long as

$$t_k \rightarrow 0; \text{ and } \sum_{i=1}^k t_i \rightarrow \infty$$

Here is a Matlab code that implements the subgradient optimization algorithm for the previous example:

```

mu1=0;
mu2=0;

Aineq=[8 2 1 4];
bineq=10;
iter=0

options2 = optimoptions('intlinprog','display','off');

while iter<50
    iter=iter+1;
    f=[16-mu1;10-mu1;-mu2;4-mu2];

    [x,z]=intlinprog(-f,[1:4],Aineq,bineq,[],[],zeros(4,1),ones(4,1),[],options2);

    z=-z;

    UB=mu1+mu2+z;

    fprintf('iter= %g; mu1 = %3.2f; mu2 = %3.2f; UB = %3.2f \n', iter, mu1,mu2,UB);

    pause
    t=.3;

    mu1=max([0;mu1-t*(1-x(1)-x(2))]);
    mu2=max([0;mu2-t*(1-x(3)-x(4))]);

    end;

```

Note:

---

<sup>2</sup>M. Held, P. Wolfe, and H. P. Crowder, Validation of subgradient optimization, Math. Program., 6,. No. 1 (1974), 62-88

- The step size is crucial to the success of the method. A common choice for it is:

$$t_k = \frac{\lambda_k(UB^k - \bar{Z})}{\|b - Ax\|}$$

where  $UB^k$  is the Lagrangean bound at iteration  $k$ ,  $\bar{Z}$  is the objective value of the best known feasible solution, and  $\lambda_k$  is a scalar between 0 and 2.

- Termination is either based on the number of iterations, or when the multipliers fail to change for a specified number of iterations.

### 2.3.2 Constraint Generation/Cutting Plane Method:

The solution of [SP] can be thought of as finding the vector  $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ , among all the possible feasible solutions to  $\{x \in \{0,1\} : 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10\}$  that has the highest objective  $(16 - u_1)x_1 + (10 - u_1)x_2 - u_2x_3 + (4 - u_2)x_4$ . Suppose there are  $H$  such vectors:

$$\begin{bmatrix} x_1^{h=1} \\ x_2^{h=1} \\ x_3^{h=1} \\ x_4^{h=1} \end{bmatrix}, \begin{bmatrix} x_1^{h=2} \\ x_2^{h=2} \\ x_3^{h=2} \\ x_4^{h=2} \end{bmatrix}, \dots, \begin{bmatrix} x_1^h \\ x_2^h \\ x_3^h \\ x_4^h \end{bmatrix}, \dots, \begin{bmatrix} x_1^H \\ x_2^H \\ x_3^H \\ x_4^H \end{bmatrix}.$$

Then [SP] can be written as:

$$\begin{aligned} z_{SP} &= \max \{(16 - u_1)x_1^{h=1} + (10 - u_1)x_2^{h=1} + (-u_2)x_3^{h=1} + (4 - u_2)x_4^{h=1}; \\ &\quad (16 - u_1)x_1^{h=2} + (10 - u_1)x_2^{h=2} + (-u_2)x_3^{h=2} + (4 - u_2)x_4^{h=2}; \\ &\quad \dots; \\ &\quad (16 - u_1)x_1^{h=H} + (10 - u_1)x_2^{h=H} + (-u_2)x_3^{h=H} + (4 - u_2)x_4^{h=H}\} \\ &= \max_{h=1,\dots,H} \{(16 - u_1)x_1^h + (10 - u_1)x_2^h + (-u_2)x_3^h + (4 - u_2)x_4^h\}. \end{aligned}$$

Given this, the best Lagrangean upper bound is

$$\min_{u_1, u_2} \left\{ u_1 + u_2 + \max_{h=1,\dots,H} \{(16 - u_1)x_1^h + (10 - u_1)x_2^h + (-u_2)x_3^h + (4 - u_2)x_4^h\} \right\}.$$

Let us define

$$\theta = \max_{h=1,\dots,H} \{(16 - u_1)x_1^h + (10 - u_1)x_2^h + (-u_2)x_3^h + (4 - u_2)x_4^h\},$$

which implies that

$$\begin{aligned} \theta &\geq (16 - u_1)x_1^h + (10 - u_1)x_2^h + (-u_2)x_3^h + (4 - u_2)x_4^h \text{ for all } h = 1, \dots, H. \\ \Rightarrow \theta + (x_1^h + x_2^h)u_1 + (x_3^h + x_4^h)u_2 &\geq 16x_1^h + 10x_2^h + 4x_4^h \text{ for all } h = 1, \dots, H \end{aligned}$$

Putting everything together, we get

$$\begin{aligned} [MP] : \min \quad & u_1 + u_2 + \theta \\ \text{s.t.} \quad & \theta + (x_1^h + x_2^h)u_1 + (x_3^h + x_4^h)u_2 \geq 16x_1^h + 10x_2^h + 4x_4^h; \quad h = 1, \dots, H \\ & u_1 \geq 0, u_2 \geq 0 \end{aligned}$$

This problem is usually called the master problem. The solution process proceeds as follows

```

    1. Start with any values of  $u \geq 0$ 
While    $LB \neq UB$ 
    2. Solve  $SP$ , get a solution  $x^h$  and an upper bound  $UB_h$ 
    3. Update the upper bound  $UB = \min(UB, UB_h)$ 
    4. Use the solution  $x^h$  to add one more cut to  $MP$ 
    5. Solve the new  $MP$ , get a solution  $u$  and a lower bound  $LB$ 
End     while

```

This procedure iterates between solving the subproblem and the master problem, changing the objective value of the first and adding one more constraint to the second.

$$\begin{aligned}
& [MP] : \\
& \min \quad u_1 + u_2 + \theta \\
& \text{s.t.} \quad \theta + (x_1^h + x_2^h)u_1 + (x_3^h + x_4^h)u_2 \geq 16x_1^h + 10x_2^h + 4x_4^h; \\
& \quad h = 1, \dots, H \\
& \quad \downarrow u_1, u_2 \quad \uparrow x_1^h, x_2^h, x_3^h, x_4^h \\
& [SP] : \\
& \max \quad (16 - u_1)x_1 + (10 - u_1)x_2 + (0 - u_2)x_3 + (4 - u_2)x_4 \\
& \text{s.t.} \quad 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\
& \quad x_i \in \{0, 1\}, i = 1, 2, 3, 4.
\end{aligned}$$

Let us apply this procedure to the above example starting with  $u_1 = 20, u_2 = 20$ .

**Iteration 1 ( $h = 1$ ):**

$u_1 = 20, u_2 = 20$ . This lead to the subproblem

$$\begin{aligned}
& \max \quad (16 - 20)x_1 + (10 - 20)x_2 + (0 - 20)x_3 + (4 - 20)x_4 \\
& \quad = -4x_1 - 10x_2 - 20x_3 - 16x_4 \\
& \text{s.t.} \quad 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\
& \quad x_i \in \{0, 1\}, i = 1, 2, 3, 4.
\end{aligned}$$

which gives a solution of  $x_1^{h=1} = x_2^{h=1} = x_3^{h=1} = x_4^{h=1} = 0$ , with an objective of 0. The corresponding Lagrangean upper bound is 40, which is the first value that  $UB$  takes, i.e.  $UB = 40$ . Using the subproblem solution  $[0, 0, 0, 0]$ , we create our first constraint in  $[MP]$ .

$$\begin{aligned}
& \theta + (x_1^{h=1} + x_2^{h=1})u_1 + (x_3^{h=1} + x_4^{h=1})u_2 \geq 16x_1^{h=1} + 10x_2^{h=1} + 4x_4^{h=1} \\
& \Rightarrow \theta \geq 0.
\end{aligned}$$

Hence the first  $[MP]$  to solve is

$$\begin{aligned}
& [MP] : \min \quad u_1 + u_2 + \theta \\
& \text{s.t.} \quad \theta \geq 0 \\
& \quad u_1, u_2 \geq 0
\end{aligned}$$

This gives a solution of  $u_1 = u_2 = \theta = 0$ , with an objective of 0. So  $LB = 0$ .

**Iteration 2 ( $h = 2$ ):**

$u_1 = 0, u_2 = 0$ . This leads to the subproblem

$$\begin{aligned}
& \max \quad 16x_1 + 10x_2 + 4x_4 \\
& \text{s.t.} \quad 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\
& \quad x_i \in \{0, 1\}, i = 1, 2, 3, 4.
\end{aligned}$$

which gives a solution of  $x_1^{h=2} = x_2^{h=2} = 1, x_3^{h=2} = x_4^{h=2} = 0$ , with an objective of 26. The corresponding Lagrangean upper bound is 26.  $UB = \min\{40, 26\} = 26$ . Using the subproblem solution, we create a second constraint in  $[MP]$ .

$$\theta + 2u_1 \geq 26$$

Hence the second [MP] to solve is

$$\begin{aligned} [MP] : \min \quad & u_1 + u_2 + \theta \\ \text{s.t.} \quad & \theta \geq 0 \\ & \theta + 2u_1 \geq 26 \\ & u_1, u_2 \geq 0 \end{aligned}$$

This gives a solution of  $u_1 = 13, u_2 = \theta = 0$ , with an objective of 13. So  $LB = 13$ .

**Iteration 3 ( $h = 3$ ):**

$u_1 = 13, u_2 = 0$ . This leads to the subproblem

$$\begin{aligned} \max \quad & 3x_1 - 3x_2 + 4x_4 \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_i \in \{0, 1\}, i = 1, 2, 3, 4. \end{aligned}$$

which gives a solution of  $x_1^{h=3} = x_2^{h=3} = x_3^{h=3} = 0, x_4^{h=3} = 1$ , with an objective of 4. The corresponding Lagrangean upper bound is  $17. UB = \min\{26, 17\} = 17$ . Using the subproblem solution, we create our third constraint in [MP].

$$\theta + u_2 \geq 4$$

Hence the third [MP] to solve is

$$\begin{aligned} [MP] : \min \quad & u_1 + u_2 + \theta \\ \text{s.t.} \quad & \theta \geq 0 \\ & \theta + 2u_1 \geq 26 \\ & \theta + u_2 \geq 4 \\ & u_1, u_2 \geq 0 \end{aligned}$$

This gives a solution of  $u_1 = 11, u_2 = 0, \theta = 4$ , with an objective of 15. So  $LB = 15$ .

**Iteration 4 ( $h = 4$ ):**

$u_1 = 11, u_2 = 0$ . This leads to the subproblem

$$\begin{aligned} \max \quad & 5x_1 - 1x_2 + 4x_4 \\ \text{s.t.} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 \leq 10 \\ & x_i \in \{0, 1\}, i = 1, 2, 3, 4. \end{aligned}$$

which gives a solution of  $x_1^{h=1} = 1; x_2^{h=1} = x_3^{h=1} = x_4^{h=1} = 0$ , with an objective of 5. The corresponding Lagrangean upper bound is  $16. UB = \min\{17, 16\} = 16$ . Using the subproblem solution, we create our fourth constraint in [MP].

$$\theta + u_1 \geq 16$$

Hence the fourth [MP] to solve is

$$\begin{aligned} [MP] : \min \quad & u_1 + u_2 + \theta \\ \text{s.t.} \quad & \theta \geq 0 \\ & \theta + 2u_1 \geq 26 \\ & \theta + u_2 \geq 4 \\ & \theta + u_1 \geq 16 \\ & u_1, u_2 \geq 0 \end{aligned}$$

This gives a solution of  $u_1 = 10.35, u_2 = 0, \theta = 5.65$ , with an objective of 16. So  $LB = 16$ . since  $UB = LB = 16$ , we stop. The optimal Lagrange multipliers are  $u_1^* = 11; u_2^* = 0$  giving the Lagrangean bound of 16.

The corresponding Matlab code is given below

```
%Solve Master problem
```

```

f=[1;1;1];
A=[];
lb=[0;0;0];
ub=[inf;inf;inf];
b=[];

z_lower=-inf;
z_upper=inf;

options=optimoptions('linprog','display','off');
options2 = optimoptions('intlinprog','display','off');

%initializations
u1=0;u2=0; iter=0;
while (abs(z_upper-z_lower)>0.0001)
iter=iter+1;
%solve SP
obj=[16-u1;10-u1;-u2;4-u2]';
weight=[8,2,1,4];

x=zeros(4,1);
x=intlinprog(-obj,[1:4],weight,10,[],[],0*obj,0*obj+1,[],options2);

z_upper=min(z_upper,u1+u2+obj*x);
new_const=[x(1)+x(2),x(3)+x(4),1];
A=[A;new_const];
b=[b;[16,10,0,4]*x];

[uutheta,z_lower]=linprog(f,-A,-b,[],[],lb,ub,[],options);
u1=uutheta(1);
u2=uutheta(2);
fprintf('iter=%g: u1= %2.2f, u2 = %2.2f, x1 = %2g,x2 = %2g,x3 = %2g,x4 = %2g, z_lower =%2.2f , z_u
end;

```

## 2.4 The relationship to Dantzig-Wolfe decomposition: Branch-and-price

Let us write the Dantzig-Wolfe reformulation of the above example where the constraint  $8x_1 + 2x_2 + x_3 + 4x_4 \leq 10$  is replaced by its equivalent representation in terms of the feasible solutions  $x^h$ ,  $h = 1, \dots, H$ :

$$\begin{aligned}
x &= \sum_{h=1}^H \alpha_h x^h \\
x &= \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \alpha_1 \begin{bmatrix} x_1^{h=1} \\ x_2^{h=1} \\ x_3^{h=1} \\ x_4^{h=1} \end{bmatrix} + \alpha_2 \begin{bmatrix} x_1^{h=2} \\ x_2^{h=2} \\ x_3^{h=2} \\ x_4^{h=2} \end{bmatrix} + \dots + \alpha_h \begin{bmatrix} x_1^h \\ x_2^h \\ x_3^h \\ x_4^h \end{bmatrix} + \dots + \alpha_H \begin{bmatrix} x_1^H \\ x_2^H \\ x_3^H \\ x_4^H \end{bmatrix} \\
x_1 &= \sum_{h=1}^H \alpha_h x_1^h, x_2 = \sum_{h=1}^H \alpha_h x_2^h, x_3 = \sum_{h=1}^H \alpha_h x_3^h, x_4 = \sum_{h=1}^H \alpha_h x_4^h.
\end{aligned}$$

where  $\sum_{h=1}^H \alpha_h = 1$  and  $\alpha_h \in \{0, 1\}$ ,  $h = 1, \dots, H$ . Hence, the original problem is equivalent to:

$$\begin{aligned} [DW] : \max & \sum_{h=1}^H (16x_1^h + 10x_2^h + 4x_4^h)\alpha_h \\ \text{s.t.} & \sum_{h=1}^H (x_1^h + x_2^h)\alpha_h \leq 1 \\ & \sum_{h=1}^H (x_3^h + x_4^h)\alpha_h \leq 1 \\ & \sum_{h=1}^H \alpha_h = 1 \\ & \alpha_h \in \{0, 1\}, \quad h = 1, \dots, H. \end{aligned}$$

Its LP relaxation is

$$\begin{aligned} [DMP] : \max & \sum_{h=1}^H (16x_1^h + 10x_2^h + 4x_4^h)\alpha_h \quad (\text{dual var}) \\ \text{s.t.} & \sum_{h=1}^H \alpha_h = 1 \quad \rightarrow \theta \\ & \sum_{h=1}^H (x_1^h + x_2^h)\alpha_h \leq 1 \quad \rightarrow u_1 \\ & \sum_{h=1}^H (x_3^h + x_4^h)\alpha_h \leq 1 \quad \rightarrow u_2 \\ & \alpha_h \geq 0, \quad h = 1, \dots, H. \end{aligned}$$

Taking the dual, we get

$$\begin{aligned} [MP] : \min & u_1 + u_2 + \theta \\ \text{s.t.} & \theta + (x_1^h + x_2^h)u_1 + (x_3^h + x_4^h)u_2 \geq 16x_1^h + 10x_2^h + 4x_4^h; \quad h = 1, \dots, H \end{aligned}$$

Which is nothing but the full Master problem [MP]. Then we could apply the same procedure as the Lagrangean relaxation procedure where we solve [DMP] instead of [MP]. This procedure is called the Dantzig-Wolfe decomposition and is summarized as follows:

While $LB \neq UB$ <ul style="list-style-type: none"> <li>1. Start with any values of <math>u \geq 0</math></li> <li>2. Solve [SP], get a solution <math>x^h</math> and an upper bound <math>UB_h</math></li> <li>3. Update the upper bound <math>UB = \min(UB, UB_h)</math></li> <li>4. Use the solution <math>x^h</math> to add one more <b>column</b> to MP</li> <li>5. Solve the new [DMP], get a <b>dual</b> solution <math>u</math> and a lower bound <math>LB</math></li> </ul>
End      while

This procedure iterates between solving the subproblem and the master problem, changing the objective value of the first and adding one more column to the second.

$[DMP] :$	$\max \sum_{h=1}^H (16x_1^h + 10x_2^h + 4x_4^h) \alpha_h$ (dual var)
s.t.	$\sum_{h=1}^H \alpha_h = 1 \longrightarrow \theta$
	$\sum_{h=1}^H (x_1^h + x_2^h) \alpha_h \leq 1 \longrightarrow u_1$
	$\sum_{h=1}^H (x_3^h + x_4^h) \alpha_h \leq 1 \longrightarrow u_2$
	$\alpha_h \geq 0, h = 1, \dots, H.$
$\Downarrow u_1, u_2$	$\Updownarrow x_1^h, x_2^h, x_3^h, x_4^h$
$[SP] :$	
$\max$	$(16 - u_1)x_1 + (10 - u_1)x_2 - u_2x_3 + (4 - u_2)x_4$
s.t.	$8x_1 + 2x_2 + x_3 + 4x_4 \leq 10$
	$x_i \in \{0, 1\}, i = 1, 2, 3, 4.$

Note that at each iteration, a restricted version of  $DMP$  is solved, as the columns are only those for  $h = 1, \dots, \bar{H}$  (think of the rest as having their corresponding  $\alpha_h$  set to 0, for  $h = \bar{H} + 1, \dots, H\}.$ . The above procedure solves the LP relaxation of the Dantzig-Wolfe reformulation ( $0 \leq \alpha_h \leq 1$ ). To solve for  $\alpha_h \in \{0, 1\}$ , we need to apply branch-and-bound.

- At node 0 we apply the Dantzig Wolfe decomposition procedure, we get the best upper bound as well as a collection of the vectors  $x^h, h \in H^{node_0}$ .
  - If all the variables  $\alpha_h$  are either 0 or 1, we stop, otherwise, we pick a fractional  $0 < \alpha_{\bar{h}} < 1$  and branch, creating two nodes where on one,  $\alpha_{\bar{h}}$  is set to 0 and on the other  $\alpha_{\bar{h}}$  is set to 1.
  - It is better to branch on the original variables  $x_i = \sum_{h=1}^H \alpha_h x_i^h$ , where we pick a variable that is fractional and apply the following branching  $x_i = 0$  versus  $x_i = 1$ . Luckily in this process that the subproblems, which are knapsacks remain as such (i.e. knapsack problems).
  - The initial set of vectors that are generated at node 0  $x^h, h \in H^{node_0}$  are then screened for those that satisfy the branching condition. those that satisfy  $x_i = 0$  are used to initialize the node where  $x_i = 0$  and those that satisfy  $x_i = 1$  are used to satisfy  $x_i = 1$ .

The last two properties are very crucial for the success of branch-and-price algorithms: (1): the structure of the subproblems do not change, (2) the set of generated columns are used to warm start subsequent nodes.

Example:

Solving the first node amounts to what we did before. The set of columns needed for  $DW$  are

$$\begin{bmatrix} h = 1 & h = 2 & h = 3 & h = 4 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix}$$

Node 0:

$$\begin{aligned}
 [DMP] : \\
 \max \quad & 26\alpha_2 + 4\alpha_3 + 16\alpha_4 && \text{(dual var)} \\
 \text{s.t.} \quad & 2\alpha_2 + \alpha_4 \leq 1 && \longrightarrow u_1 \\
 & \alpha_3 \leq 1 && \longrightarrow u_2 \\
 & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1 \\
 & \alpha_h \geq 0, \quad h = 1, 2, 3, 4.
 \end{aligned}$$

the solution is  $\alpha_1^* = 0; \alpha_2^* = 0; \alpha_3^* = 0; \alpha_4^* = 1$  with an objective of 16. The corresponding  $x$  solution are

$$x^* = \sum_{h=1}^H \alpha_h^* x^h \quad [0, 0, 0, 1] \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Which is the optimal solution.