Lecture Note on Benders' Decomposition

Uffe Gram Christensen (uffe@diku.dk)
Anders Bjerg Pedersen (andersbp@diku.dk)

October 22, 2008

1 Introduction

Benders' Decomposition is an algorithm that allows us to solve certain optimisation problems very quickly. The algorithm can be used on any kind of optimisation problem but requires a certain substructure within the problem to be efficient.

In this lecture note we will explain what Benders' Decomposition does, and how it achieves faster solutions of optimisation problems. We will also give an example of how Benders' Decomposition can be used on a facility location problem in order to illustrate the mechanics of the algorithm.

In order to do this we will start by introducing some notation and terminology of linear programming. Hopefully most of this will be familiar to the reader (you), otherwise we recommend looking in [3] or [4].

The lecture note is based on an article by J.N. Hooker [1] and a chapter from the book by R.K. Martin [2].

1.1 Reminder on LP-formulations and duality

Let us begin our formal introduction by reminding the reader of the notation used for standard LP problems. Consider the LP problem below:

$$\begin{aligned} \min\ & z = cx \\ \text{s.t. } & Ax \geq b \\ & x \geq a \end{aligned} \tag{1}$$

where $A_{(m \times n)}$ are our constraint coefficients, $b_{(n)}$ are our right-hand sides, and $c_{(n)}$ are our cost function coefficients.

Recall from the theory of duality that the dual of (1) can be stated as follows:

$$\begin{aligned} \max\ & w = ub \\ \text{s.t. } & uA \leq c \\ & u \geq a \end{aligned} \tag{2}$$

where $u_{(n)}$ are our dual variables. An optimal solution to (2) is also an optimal solution to (1). We shall later use this fact in justifying why Benders' Decomposition actually works.

1.2 Inference duality

We now describe a more general concept of duality, namely *inference duality*, of which linear duality is a small corner. Consider two propositions, P and Q, whose truth or falsehood depends on a variable (or set of variables) named x and let $D = \{x \mid x \geq a\}$ be a domain for the variables. We then define the following:

Definition 1. *P implies Q with respect to D (denoted $P \xrightarrow{D} Q$) if Q is true for any $x \geq a$ for which P is true.*

To understand this definition, consider again the problem (1). We now construct the *inference dual* of this problem:

$$\begin{aligned} \max \quad & \beta \\ \text{s.t.} \quad & Ax \geq b \xrightarrow{D} cx \geq \beta \end{aligned} \tag{3}$$

The purpose of the dual problem is to derive a bound (indeed, in our case the strongest lower bound) on the objective function value. One can easily prove the following theorem, which describes the importance of inference duality:

Theorem 1. *The optimisation problem (1) has the same optimal solution as its inference dual (3).*

The proof is left to the curious reader of this note.

Example 1. *Consider the trivial problem*

$$\begin{aligned} \min \quad & z = 4x \\ \text{s.t.} \quad & x \geq 5 \\ & x \leq 10 \\ & x \geq 0 \end{aligned} \tag{4}$$

Let $D = \{x \mid x \geq 0\}$ and let $S = \{x \mid x \geq 5 \wedge x \leq 10\}$. The inference dual of this problem can be stated:

$$\begin{aligned} \max \quad & \beta \\ \text{s.t.} \quad & x \in S \xrightarrow{D} 4x \geq \beta \end{aligned} \tag{5}$$

We see from Definition 1 that we need to maximize β such that $4x \geq \beta$ for ANY $5 \leq x \leq 10$. Clearly the solution to this problem is $x = 5$ with $\beta = 20$, hereby giving us a tight lower bound on the objective function value of (4) (this also happens to be the optimal solution!).

1.3 General method theory

We now carry on to describe Benders' Decomposition and how to use it. The following section will outline the method without putting much attention to examples, as this will be thoroughly taken care of in a later part of this note. Also, due to the scope of the paper, little attention will be given to actually proving the correctness of the algorithm.

The first step of the algorithm consist of fixing a certain amount of variables in our original problem (1), hereby making the resulting sub-problem easy to solve. The essence of Benders' Decomposition lies in determining *which* variables to fix, such as to simplify the resulting sub-problem. This decision will often require specific knowledge of the problem at hand as well as known ways to solve similar problems quickly. It is therefore not possible for us within the scope of this article to give a complete overview of solution methods for all problems. However, in Section 2 we give a concrete example on how to choose the variables to be fixed, based on the difficulty of the resulting sub-problems.

We first split our variables into two groups: let y and x (both belonging to some domain D) be a separation of the variables in our initial problem. This gives us the following reformulation:

$$\begin{aligned} \min \ & z = cx + f(y) \\ \text{s.t. } & Ax + g(y) \geq b \\ & x, y \in D \end{aligned} \tag{6}$$

Here $g(y)$ is a vector of functions on each of the variables in y. We now fix the variables in y to some trial values $\bar{y}$ (and move a few terms around), hereby generating the sub-problem:

$$\begin{aligned} \min \ & z = cx + f(\bar{y}) \\ \text{s.t. } & Ax \geq b - g(\bar{y}) \\ & x \in D \end{aligned} \tag{7}$$

Even though this problem may be simple enough to solve directly, recall from section 1.2 that the inference dual of a problem gives us a lower bound on the optimal value of the main problem. In general Benders' Decomposition we solve the inference dual:

$$\begin{aligned} \max \quad & \beta \\ \text{s.t.} \quad & Ax \geq b - g(\bar{y}) \xrightarrow{x, \bar{y} \in D} cx + f(\bar{y}) \geq \beta \end{aligned} \tag{8}$$

If we restrict our attention to linear sub-problems only, though, the classical separation lemma (see [1], Theorem 2) allows the sub-problem dual to be stated as follows:

$$\begin{aligned} \max \ & w = u(b - g(\bar{y})) + f(\bar{y}) \\ \text{s.t. } & uA \leq c \\ & u \in D \end{aligned} \tag{9}$$

Solving the sub-problem (inference) dual gives us a lower bound β^* on the objective function value that can be inferred from the constraints in (6) (assuming that $y = \bar{y}$). We now try to use our knowledge of the simplicity of the sub-problem to try and derive some general knowledge about *any* fixing of the variables in y, not just the current trial values of $\bar{y}$. In other words, what we want to achieve is a function $\beta_{\bar{y}}(y)$ that gives us a valid lower bound for *any* value of y. This means that in the above case, $\beta_{\bar{y}}(\bar{y}) = \beta^*$. As before mentioned, finding this function often requires specific knowledge of the problem at hand.

For problems with linear sub-problems we see that if the solution u is finite, the dual gives us a way to derive a bound on the form $z \geq u(b - g(\bar{y})) + f(\bar{y})$, which we would like to generalise to be valid for all y, not just $y = \bar{y}$. However, noticing that our solution u is feasible for any y, we can generalise the bound to be

$$z \geq u(b - g(y)) + f(\bar{y}).$$

The function derived from solving the sub-problem inference dual yields what we will from now on call a *Benders cut*: $z \geq \beta_{\bar{y}}(y)$. In the case of linear sub-problems, this cut then becomes:

$$z \geq \beta_{\bar{y}}(y) = u(b - g(y)) + f(\bar{y}).$$

As we will iterate the above method we remember to take notice of what $\bar{y}$ we used to derive this particular cut. The cut represents information achieved on the lower bound, assuming the current trial values of $\bar{y}$. All the cuts we generate are grouped together in what is normally referred to as the *Benders' master problem*:

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & z \geq \beta_{y^k}(y), \quad k = 1, \ldots, K \\ & y \in D_y \end{aligned} \tag{10}$$

Here y^k represents the trial values used in the k'th iteration of the algorithm. Solving this master problem gives us the next assignment of the variables in y to try out, i.e. the $\bar{y}$ for the next iteration.

1.4 The algorithm

In the previous section we have outlined the various steps to take in using Benders' Decomposition. We will now set up a more formal algorithm which can be implemented when solving concrete problem instances.

We begin by initialising $k = 0$ (our iteration counter) and $\bar{z} = -\infty$ (our initial best guess on a lower bound on the objective function value). Furthermore we choose our initial assignment $\bar{y}$ of the variables in y. How these are chosen may of course have an impact on the number of iterations of the algorithm, but for now we choose them either randomly or using some specific knowledge of the instance we are trying to solve. We now solve the sub-problem inference dual with respect to the assignment $\bar{y}$. If it does not have a feasible solution, we stop the algorithm as the original problem is then

unbounded ($\bar{z} = -\infty$). If $\beta = \bar{z}$ we stop the algorithm, as it has found the optimal solution $\bar{z}$. Otherwise let $\beta > \bar{z}$ denote our solution to the sub-problem inference dual and let $\beta_{\bar{y}}(y)$ be our bounding function (as described above) with $\beta_{\bar{y}}(\bar{y}) = \beta$.

We then increment k and set $y^k = \bar{y}$, before we add the Benders cut $z \geq \beta_{y^k}(y)$ to the master problem which we then try to solve. If we find that the master problem is infeasible, we stop the algorithm because our original problem is then infeasible. Otherwise we let $\bar{y}$ be our new optimal solution to the master problem (and hence our new trial value for the next iteration where we again solve the sub-problem inference dual with our new $\bar{y}$) with optimal value $\bar{z}$.

The above text can be formalised in the following algorithm:

Algorithm 1 BENDERSDECOMPOSITION()

```
Choose ȳ in original problem
z̄ ← −∞
k ← 0
while (sub-problem dual has feasible solution β ≥ z̄) do
    Derive lower bound function β_ȳ(y) with β_ȳ(ȳ) = β
    k ← k + 1
    y^k ← ȳ
    Add z ≥ β_ȳ(y) to master problem
    if (master problem is infeasible) then
        Stop. The original problem is infeasible.
    else
        Let (z̄, ȳ) be the optimal value and solution to the master problem.
return (z̄, ȳ)
```

To prove the correctness of the algorithm, Hooker ([1]) presents the following theorem:

Theorem 2. *Assume that after each iteration of Algorithm 1 the Benders cut $z \geq \beta_{\bar{y}}(y)$ is valid (meaning that any optimal solution (z^*, x^*, y^*) to our initial problem with divided set of variables (6) satisfies $z^* \geq \beta_{\bar{y}}(y^*)$). Then the following holds:*

i) If the algorithm stops with $(\bar{z}, \bar{y})$ as a finite solution to (10), then the initial problem (6) has optimal solution $(\bar{x}, \bar{y})$ with $f(x, \bar{y}) = \bar{z}$.

ii) If the algorithm stops with an infeasible master problem (10), then the initial problem (6) is infeasible.

iii) If the algorithm stops with a sub-problem, whose inference dual (8) is infeasible, then the initial problem (6) is unbounded.

Proof.

i) We let β^* be the last obtained optimal solution to (8). As (7) and (8) have the same optimal solution, β^* will also be optimal for (7) with an accompanying optimal solution $\bar{x}$. Any feasible solution to (6) with value $\bar{z}$ is optimal in (6), as $\bar{z}$ satisfies

$\bar{z} \geq \beta_{\bar{y}}(y)$. As we could not find a better y for our master problem, we have an optimal solution to (6).

ii) Since for every Benders cut we have that any feasible solution (x, y) to (6) satisfies that $\bar{z} \geq \beta_{\bar{y}}(y)$, an infeasible master problem (10) will imply that also our initial problem (6) is infeasible.

iii) As (8) is infeasible there is no lower bound for (7). Therefore we have found a case in which there is also no lower bound for (6), which is therefore unbounded.

□

Hooker also proves that the algorithm will actually stop when the sub-problem dual is solved to optimality.

2 Example

Having used the previous pages to explain how Benders' Decomposition works, we will now show how it can be used on a linear programming problem. First we will describe the general problem we are trying to solve, next we will describe what substructure we recognise that will allow us to efficiently apply the Benders' algorithm. We will the present how we will arrive at the Benders cuts for this problem and what the Benders cuts symbolise. Lastly we will present a specific instance of the problem and run some iterations of the algorithm to allow you to get a feel of how Benders Decomposition works.

2.1 Facility location problem

The problem we will be solving is known as the facility location problem ([4], page 10) and also under a range of similar names. The problem models a situation with n factories and m customers. Each customer has a demand that needs to be satisfied from one or more of the factories, in our formulation of the problem x_{ij} is how big a fraction of customer j's demand is satisfied from factory i. The cost of satisfying all demand for customer j from factory i is denoted c_{ij}. The cost of opening a factory is y_i and the variable f_i is a decision variable indication if factory i is open or closed. With these variables in place the LP-problem can be stated as follows:

$$
\begin{aligned}
\min \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \quad &+ \quad \sum_{i=1}^{n} f_i y_i \\
\text{s.t.} \sum_{i=1}^{n} x_{ij} \quad &\geq \quad 1,\ j = 1, \ldots, m \\
-x_{ij} + y_i \quad &\geq \quad 0,\ i = 1, \ldots, n,\ j = 1, \ldots, m \\
x_{ij} \quad &\geq \quad 0,\ i = 1, \ldots, n,\ j = 1, \ldots, m \\
y_i \quad &\in \quad \{0, 1\},\ i = 1, \ldots, n
\end{aligned}
$$

If we fix the values of y to $\bar{y}$ this problem becomes very simple indeed. We simply choose to supply a customer's full demand from the plant which can deliver it at the lowest cost. This can be done in $O(mn)$ time. This gives us the sub-problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{s.t.} \sum_{i=1}^{n} x_{ij} \quad & \geq \quad 1,\ j = 1,\ldots,m \\ -x_{ij} + \bar{y}_i \quad & \geq \quad 0,\ i = 1,\ldots,n,\ j = 1,\ldots,m \\ x_{ij} \quad & \geq \quad 0,\ i = 1,\ldots,n,\ j = 1,\ldots,m \end{aligned}$$

2.2 Solving the dual sub-problem

We now want to solve the dual problem. We use proposition 10.7 from [2] to find the dual variables. The terminology in this proposition is that $i \in O(\bar{y})$ denotes the *open* factories while $i \in C(\bar{y})$ denotes the *closed* factories. The variables v_j are the dual variables associated with the demand constraints and w_{ij} are the dual variables associated with the setup constraints. The proposition states the following for $j = 1,\ldots,m$:

$$\begin{aligned} v_j &= \min_{i \in O(\bar{y})} \{c_{ij}\} \\ w_{ij} &= \begin{cases} 0 & i \in O(\bar{y}) \\ \max_{i \in C(\bar{y})}\{(v_j - c_{ij}), 0)\} & i \in C(\bar{y}) \end{cases} \end{aligned}$$

The formal proof for the correctness can be found in [2], but we will here briefly explain the meaning of the proposition. $v_j = \min_{i \in O(\bar{y})}\{c_{ij}\},\ j = 1,\ldots,m$ simply states that we will select the open factory with the least cost for each customer. $w_{ij} = 0,\ i \in O(\bar{y})$ states that we will ignore the setup constraint if factory i is open. $w_{ij} = \max\{(v_j - c_{ij}), 0)\}$, $i \in C(\bar{y})$ states that if a factory is closed then we could improve our solution by $v_j - c_{ij}$ for customer j by opening factory i, if the value is positive.

We will be referring to the dual variable as u which means the combination (v, w). Using the proposition we get the following Benders cut:

$$\beta_{\bar{y}}(\bar{y}) = u(b - B\bar{y}) + f^t\bar{y}$$

Due to the structure of the problem this can also be written as

$$\beta_{\bar{y}}(\bar{y}) = \sum_{i=1}^{m} v_j + \sum_{i=1}^{n}\left(f_i - \sum_{j=1}^{m} w_{ij}\right) y_i.$$

2.3 An actual instance

We will now set up a small example problem and run the algorithm on this problem. Our example will use 3 possible factories and 5 possible customers. The following table lists the c_{ij}'s as well as the f_i's.

	Customers					
Factory	1	2	3	4	5	**Fixed costs**
1	2	3	4	5	7	2
2	4	3	1	2	6	3
3	5	4	2	1	3	3

To start the algorithm we simply select only to open factory 1. Thus $\bar{y}_1 = 1$, $\bar{y}_2 = 0$, $\bar{y}_3 = 0$. Next we find v_j which we find by solving $v_j = \min_{i \in O(\bar{y})}\{c_{ij}\}$, $j = 1, \ldots, m$. Since $O(\bar{y})$ only contains factory 1, this is very simple and gives us $v_j = (2, 3, 4, 5, 7)$. Next we find w_{ij} using the formulas from the proposition:

$$w_{1j} = 0$$

This is due to factory 1 being open.

$$w_{2j} = (0, 0, 3, 3, 1)$$

This says that there is nothing to be gained for customer 1 and 2 if we open factory 2, but there is a gain for customers 3 through 5.

$$w_{3j} = (0, 0, 2, 4, 4)$$

Again there is no gain for customers 1 and 2 but there is a gain for 3 through 5 if we open factory 3.

We can now generate our Benders cut which will eliminate all solutions that yield a more expensive solution to the original problem.

$$\begin{aligned}
\beta_{\bar{y}}(\bar{y}) &= \sum_{i=1}^{m} v_j + \sum_{i=1}^{n} \left(f_i - \sum_{j=1}^{m} w_{ij} \right) y_i \\
&= 2 + 3 + 4 + 5 + 7 + (2 - 0)y_1 + (3 - (3 + 3 + 1))y_2 + (3 - (2 + 4 + 4))y_3 \\
&= 21 + 2y_1 - 4y_2 - 7y_3
\end{aligned}$$

This gives us an upper bound on the solution of 23. Inserting the Benders cut into the master problem makes it look like this:

$$\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & z \geq 21 + 2y_1 - 4y_2 - 7y_3 \\
& y \in \mathbb{B}^3
\end{aligned}$$

As can easily be seen the optimal solution to this problem is to choose $y = (0, 1, 1)$, that is keep factory 1 closed and open factories 2 and 3, yielding a solution of 10 (our new lower bound). This then becomes our new $\bar{y}$ and we can do one more iteration of the algorithm. The values of v_j and w_{ij} can now be found using the new value of $\bar{y}$ and the resulting values are:

$$\begin{aligned}
v_j &= (4, 3, 1, 1, 3) \\
w_{1j} &= (2, 0, 0, 0, 0) \\
w_{2j} = w_{3j} &= 0
\end{aligned}$$

Calculating the next Benders cut is then a breeze:

$$\begin{aligned}\beta_{\bar{y}}(\bar{y}) &= \sum_{i=1}^{m} v_j + \sum_{i=1}^{n} \left(f_i - \sum_{j=1}^{m} w_{ij} \right) y_i \\ &= (4+3+1+1+3) + (2-2)y_1 + (3-0)y_2 + (3-0)y_3 \\ &= 12 + 3y_2 + 3y_3\end{aligned}$$

This yields a new upper bound of 18, and the master problem now looks like this:

$$\begin{aligned}\min \quad & z \\ \text{s.t.} \quad & z \geq 21 + 2y_1 - 4y_2 - 7y_3 \\ & z \geq 12 + 3y_2 + 3y_3 \\ & y \in \mathbb{B}^3\end{aligned}$$

The solution to this problem is $y = (0, 0, 1)$ which again becomes our new $\bar{y}$, with the new lower bound of 15. The third iteration of the algorithm then yields the values:

$$\begin{aligned}v_j &= (5, 4, 2, 1, 3) \\ w_{1j} &= (3, 1, 0, 0, 0) \\ w_{2j} &= (1, 1, 1, 0, 0) \\ w_{3j} &= 0\end{aligned}$$

And the resulting Benders cut:

$$\begin{aligned}\beta_{\bar{y}}(\bar{y}) &= (5+4+2+1+3) + (2-4)y_1 + (3-3)y_2 + (3-0)y_3 \\ &= 15 - 2y_1 + 3y_3\end{aligned}$$

This yields no better upper bound, but naturally adds to the master problem which is now:

$$\begin{aligned}\min \quad & z \\ \text{s.t.} \quad & z \geq 21 + 2y_1 - 4y_2 - 7y_3 \\ & z \geq 12 + 3y_2 + 3y_3 \\ & z \geq 15 - 2y_1 + 3y_3 \\ & y \in \mathbb{B}^3\end{aligned}$$

Which has the optimal solution $y = (1, 0, 1)$ and yields a new lower bound of 16. These iterations can now be repeated until the upper and lower bound are the same which will then yield the optimal solution and prescribe which factories are to be opened.

In the end we arrive at the optimal solution of opening factories 1 and 3 with an optimal solution value of 16.

3 Conclusion

In this article we have presented an alternative way of solving certain optimisation problems. Choosing whether or not to impose this solving method depends heavily on the knowledge of the potential simplicity of certain "easy" sub-problems. We have described a more general notion of duality, namely inference duality, and used this in constructing an algorithm for solving optimisation problems using Benders' Decomposition. Finally, we have applied this algorithm to a concrete instance of the facility location problem and seen how it makes use of the simple sub-structure when fixing the open factories.

References

[1] J.N. Hooker: *Logic-Based Benders Decomposition* (1995)

[2] R.K. Martin: *Large scale linear and integer optimisation: A unified approach* (1999)

[3] H.A. Taha: *Operations Research - An Introduction, 7th edition*, Pearson Education (2003)

[4] L.A. Wolsey: *Integer Programming*, John Wiley & Sons, Inc. (1998)