15.094/SMA5223 Systems Optimization: Models and Computation
Assignment 2 (85 points)
Due March 2, 2004

1 Lagrangian Dual (10 points)

Consider the problem:

$$\begin{aligned} P: \quad & \min_x \quad f(x) = \tfrac{1}{2}x^T x + c^T x \\ & \text{s.t.} \qquad Ax = b \\ & \qquad\quad x \geq 0 \\ & \qquad\quad x \in \Re^n \ . \end{aligned}$$

Create dual variables u for the constraints "$Ax = b$" and v for the constraints "$x \geq 0$", and formulate the Lagrangian. What is the Lagrange dual of this problem?

2 The Frank-Wolfe Algorithm (25 points)

Consider the following instance of P:

$$\begin{aligned} P: \quad & \text{minimize} \quad f(x) \\ & \text{s.t.} \qquad x \in C \ , \end{aligned}$$

where

$$f(x) = -24x_1 + 0.5x_1^4 - 10x_2 + x_2^2$$

and

$$C = \{(x_1, x_2) \mid x_1 - x_2 \leq 1, 6x_1 + x_2 \leq 13, x_2 \leq 7, x_1 \geq 0, x_2 \geq 0\} \ .$$

Download the support files **fw.zip** which contains skeletal code for the Frank-Wolfe algorithm designed to be run using OPLStudio. You may choose to base your code on this skeletal code or not.

- Construct a version of the Frank-Wolfe method and solve this problem starting from the point $(x_1, x_2) = (0.5, 3.0)$.
- Plot the iterates (x_1^k, x_2^k).
- Hand in a hard copy of your code, coded in OPLStudio. What stopping criteria did you use?

3 Subgradient Algorithm for Lagrange Dual (20 points)

Consider the primal problem:

$$\begin{array}{lll} \text{OP}: & \text{minimum}_x & c^T x \\ & \text{s.t.} & Ax - b \leq 0 \\ & & x \in \{0,1\}^n \ . \end{array}$$

Here $g(x) = Ax - b$ and $P = \{0,1\}^n = \{x \mid x_j = 0 \text{ or } 1, j = 1, \ldots, n\}$.

We create the Lagrangian:

$$L(x,u) := c^T x + u^T (Ax - b)$$

and the dual function:

$$L^*(u) := \quad \text{minimum}_{x \in \{0,1\}^n} \quad c^T x + u^T (Ax - b)$$

The dual problem then is:

$$\begin{array}{lll} \text{D}: & \text{maximum}_u & L^*(u) \\ & \text{s.t.} & u \geq 0 \end{array}$$

Now let us choose $\bar{u} \geq 0$. Notice that an optimal solution $\bar{x}$ of $L^*(\bar{u})$ is:

$$\bar{x}_j = \begin{cases} 0 & \text{if } (c - A^T \bar{u})_j \geq 0 \\ 1 & \text{if } (c - A^T \bar{u})_j \leq 0 \end{cases}$$

for $j = 1, \ldots, n$. Also,

$$L^*(\bar{u}) = c^T \bar{x} + \bar{u}^T (A\bar{x} - b) = -\bar{u}^T b - \sum_{j=1}^{n} \left[(c - A^T \bar{u})_j \right]^- \ .$$

Also

$$g := g(\bar{x}) = A\bar{x} - b$$

is a subgradient of $L^*(\bar{u})$.

Now consider the following data instance of this problem:

$$A = \begin{pmatrix} 6 & -8 \\ -2 & -3 \\ 7 & 5 \\ -4 & 6 \\ 3 & 14 \end{pmatrix} \ , \ b = \begin{pmatrix} 12 \\ -1 \\ 45 \\ 20 \\ 42 \end{pmatrix}$$

and

$$c^T = \begin{pmatrix} -5 & -4 \end{pmatrix} \ .$$

Construct a subgradient algorithm for solving Lagrange dual problems, coded in OPLStudio. You may use the skeletal code **sg.zip** as a starting point. Use your algorithm to solve problem D for the data instance (A, b, c) above, starting at $u^1 = (1, 1, 1, 1, 1)^T$, with the following step-size choices:

- $\alpha_k = \frac{1}{k}$ for $k = 1, \ldots$.
- $\alpha_k = \frac{1}{\sqrt{k}}$ for $k = 1, \ldots$.
- $\alpha_k = 1 \times (0.99)^k$ for $k = 1, \ldots$.
- $\alpha_k = .9 \times (0.9775)^k$ for $k = 1, \ldots$.
- a stepsize rule of your own.

Which step-size rules work best for this problem?

4 Solution Strategies for the Cutting Stock Problem (30 points)

This question asks you to compare different formulations and solution strategies for solving the cutting stock problem. The goal of this question is to demonstrate the power of alternative formulations of an optimization problem, as well as to demonstrate the power of column generation as a solution strategy.

We first revisit the Kantorovich formulation of the cutting stock problem discussed in lecture. We will use the following notation, which is different from the notation used in the lecture.

- $w_i, i = 1, \ldots, m$: the width of item i. (data)
- $b_i, i = 1, \ldots, m$: the demand for stock of width w_i. (data)

- W: the width of the large rolls. (data)
- K: the number of available rolls. (data)
- y^k: 1 if roll k is cut, 0 otherwise, $k = 1, \ldots, K$. (decision variable)
- x_i^k: number of times that an item of width w_i is cut on roll k. (decision variable)

The objective is to minimize the number of rolls used in order to satisfy demand.

$$\min_{x,y} \sum_{k=1}^{K} y^k$$

We must satisfy demand for each item $i = 1, \ldots, m$:

$$\sum_{k=1}^{K} x_i^k \geq b_i$$

We cannot exceed the width of each roll $k = 1, \ldots, K$:

$$\sum_{i=1}^{m} w_i x_i^k \leq W y^k$$

The decision variables must also satisfy their bounds and integrality:

$$\begin{aligned} x_i^k &\geq 0 \\ 0 \leq y^k &\leq 1 \\ x, y &\text{ are integer} \end{aligned}$$

Note that this formulation solves for the number of rolls that must be cut as well as the pattern $(x_1^k, \ldots, x_m^k)$ that is used for each roll $k = 1, \ldots, K$.

Question (a): **Solution using the Kantorovich formulation.** Use the files **kant.mod** and **kant.osc** as a starting point. Finish constructing the OPL model for this formulation and then solve the problem using the data file **cs1.dat**. Next modify the data file to gain some more experience with the ease and/or difficulty of solving the cutting stock problem using the Kantorovich formulation. Try varying K, the number of rolls available. Also try varying the number m of items, as well as the widths $w_i, i = 1, \ldots, m$. Next try varying the width W of the large rolls. What do you observe? What is a possible explanation for your observation? Finally, relax the integrality constraints. What happens?

As a common basis of comparison for a sections of this problem use the data files **cs1.dat**,**cs2.dat**,**cs3.dat**, **chv1.dat**,**chv2.dat**,**chv3.dat**, and **chv4.dat**.

You should have found some difficulty when trying to solve the Kantorivich formulation for some data configurations. Let us also think about an alternative formulation of the problem. The concept is simple. For a given set of widths, $w_i, i = 1, \ldots, m$, it is "easy" to determine all feasible patterns into which a large roll of width W could be cut. This enumeration could take some time to compute, but it is a straightforward exercise. Once we enumerate all of the feasible cutting patterns, we can then determine the fewest number of rolls to cut in order to satisfy demand.

First, we introduce some more notation:

- n: number of possible patterns, $j = 1, \ldots, n$. (data)
- a_{ij}: The number of items of width w_i which are cut in pattern j, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. (data)
- z_j: The number of rolls that we will cut using pattern j. (decision variable)

Using the above notation, there are n different patterns. Each pattern j is represented using $(a_{1j}, a_{2j}, \ldots, a_{mj})$ for the j^{th} pattern, and satisfies the requirement:

$$\sum_{i=1}^{m} a_{ij} w_i \leq W$$

Now, as before, we want to minimize the total number of rolls cut. But now we minimize the sum (over all possible patterns) of the number of rolls cut using each pattern. (A deeper connection between the two formulations was discussed in lecture.) The objective function is:

$$\min_{z} \sum_{j=1}^{n} z_j$$

We must satisfy demand for each item $i = 1, \ldots, m$:

$$\sum_{j=1}^{n} a_{ij} z_i \geq b_i$$

The decision variables must satisfy their bounds and integrality:

$$z_j \geq 0$$
$$z_j \text{ is integer}$$

Note that the x_i^k from the Kantorovich formulation are *not* the same as the a_{ij} of this formulation.

Question (b): **Solution using the pattern formulation.** Use the file **fullmastercs.osc**, **fullmaster.mod**, **patterns.osc** and as a starting point. Finish constructing the OPL model for the pattern formulation of the cutting stock problem and then solve the problem using the data files **cs1-3.dat**, and **chv1-4.dat**. Next modify the data file to gain some more experience with the ease and/or difficulty of solving the cutting stock problem using the pattern formulation. Try varying the width W of the large roll. Also try varying the number m of items. What do you observe? What is a possible explanation for your observation? Finally, relax the integrality constraints. What happens?

You should have found some difficulty when running the OPL model of the pattern formulation. Even when the number of rolls is small, the number of feasible patterns can be huge. Forming the coefficient matrix that corresponds to all possible patterns is clearly impractical. However, as was discussed in lecture, this problem is a perfect candidate for column generation. Instead of pre-generating all possible patterns, we generate feasible patterns as necessary.

Question (c): Use the files **CS.osc**, **generatecol.mod**, **selectcol.mod**, as a starting point, and finish constructing a column generation algorithm in OPL script to solve the cutting stock problem. Then solve the problem using the data files **cs1-3.dat**, and **chv1-4.dat**. Next modify the data file to gain some more experience with the ease and/or difficulty of solving the cutting stock problem using column generation. Try varying the width W of the large rolls. Also try varying the number m of items, as well as the widths $w_i, i = 1, \ldots, m$. What do you observe? What is a possible explanation for your observation? What do you observe when comparing this formulation and implementation to the previous formulations and implementations?