Model Solutions for Homework Assignment 1
CS798

Francisco Claude

September 27, 2008

1 Note

Keshav : These solutions are lightly edited versions of the original solutions. Please compare your own solutions with these and determine areas where you may need to deepen your understanding of the material.

2 Modeling

You have been hired as the head of CHYM FM's balloon operations. Too much money is being spent for each flight! Your job is to make flight profitable again (the number of flights is not negotiable).

For each flight, you can control where you take off from (there is a finite set of take-off locations) and the duration of the flight, as long as the flight lasts at least 15 minutes. The cost of a flight depends on its duration (to pay for natural gas, the pilot's wages, and for the chase vehicle), where the balloon takes off from, and how far the landing site is from a road (the further away it is from a road, the more it has to be dragged over a farmer's field). Moreover, you can have up to 9 passengers (in addition to at least one pilot), and charge them what you wish. Of course, the number of passengers decreases (say linearly) with the cost of the ticket.

What are the fixed parameters? What are the input and output parameters? What are the control variables? Come up with plausible transfer and objective functions. How would you empirically estimate the transfer function?

Solution. This problem admits many solutions. Here is *one* possible solution:

Control variables (these are from the statement of the problem):

- x_i : starting point of flight i
- d_i : duration of flight i, $d_i \geq 15$
- The cost of ticket for the ith flight, t_i.
- The number of passengers in a flight is not a control parameter, because it is related to the cost of a ticket - once the cost of the ticket is determined, the number of passengers cannot be independently controlled.

Fixed parameters:

- The possible take-off locations, V.
- The cost of chase vehicle (gas needed, maintenance, etc.) per kilometer of travel.
- Location of the roads, R.

- The cost of the natural gas, g, per minute of flight.
- Pilot's wages, w, per minute of flight.

The input parameters are:

- The wind speed and direction for flight i

The transfer functions are:

- Where a balloon lands, as a function of starting point, wind speed and direction, and flight duration.
- The number of passengers p as a function of cost of a ticket.
- A function that, given the cost of every process in the business, computes the cost of flight i.

The output parameters are:

- For flight i, the distance of the balloon landing spot from a road
- Number of passengers for flight i, p_i
- The cost of flight i, denoted f_i

The objective function is to maximize $p_i t_i - f_i$.
For empirically estimating the transfer functions:

- For each starting point, for each wind speed and direction, empirically determine the landing spot.
- The number of passengers for a cost can be determined by trial and error or by doing a market study.
- The cost of flight can be determined empirically as a linear combination of the fixed parameters.

3 Optimizing a function of two variables

Consider the following system:

$$O(x_1, x_2) = 10x_1 - 3x_2, \text{ where}$$
$$2x_1 - x_2 = 1$$
$$x_1 \geq 0, x_2 \geq 0$$

Geometrically find the optimal value of O.

Solution. Figure 1 shows the plot for the curve $2x_1 - x_2 = 1$ subject to $x_1 \geq 0$ and $x_2 \geq 0$. The system does not impose any constraint on how much x_1 and x_2 can grow, so the maximum value is unbounded. We know that the minimal value is at a vertex, in this case we only have one $(0.5, 0)$. If we evaluate the function in $(0.5, 0)$ and at a randomly chosen point, say $(3, 5)$ we get:

$$\begin{aligned} O(0.5, 0) &= 5 \\ O(3, 5) &= 15 \end{aligned}$$

Using this information, we know that the minimum value of O, given the constraints, is 5 at $(0.5, 0)$. There is no maximum value, since O is unbounded in the space defined by the constraints.

($Keshav$: I made a mistake here of not upper-bounding the variables!)

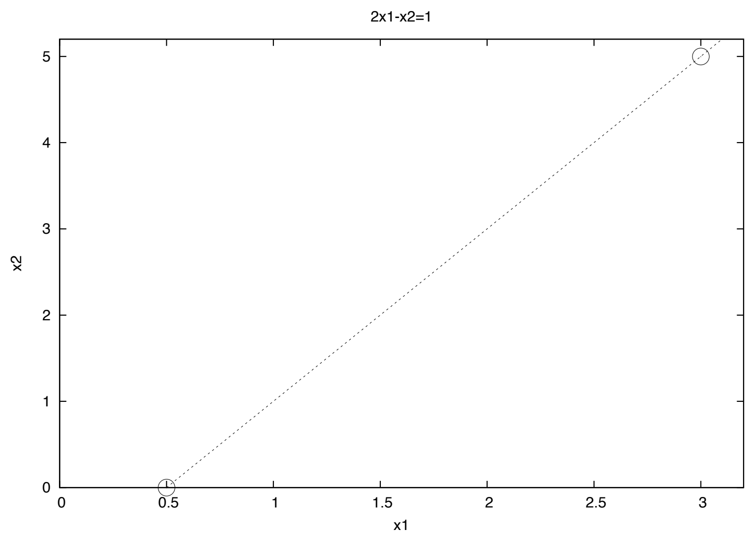

Figure 1: Geometrical representation of the constraints for problem 2.

4 Optimizing a function of three variables

Geometrically find the optimal value of O where:

$$O(x_1, x_2, x_3) = 5x_1 + 2x_2 - x_3$$
$$x_1 + x_2 + x_3 = 1$$
$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

Solution. Figure 2 shows the plot for the curve $x_1 + x_2 + x_3 = 1$ for x_1, x_2, x_3 non-negative. The resulting plot serves to find the optimal values of O. The optimal value has to be in a vertex, so we evaluate the three of them:

$$\begin{aligned} O(1,0,0) &= 5 \\ O(0,1,0) &= 2 \\ O(0,0,1) &= -1 \end{aligned}$$

Clearly, the maximum value is reached at point $(1, 0, 0)$ and the minimum value is reached at point $(0, 0, 1)$.

5 Representing a linear program

Make and state the appropiate assumptions to model the system in Problem 1 in standard form.

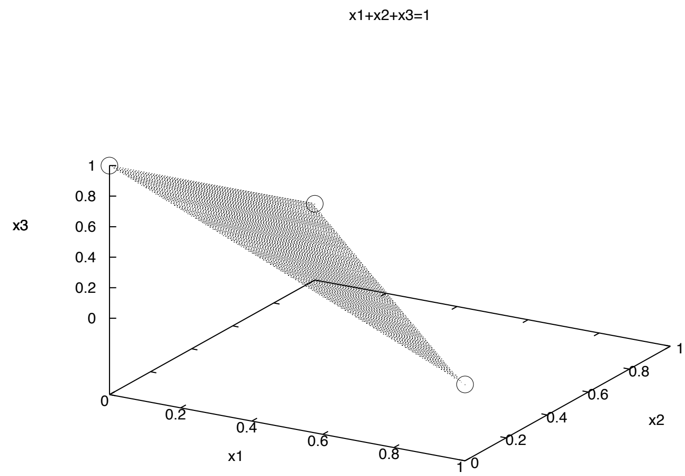

Figure 2: Geometrical representation of the constraints for problem 3.

Solution. We will assume we have the transfer functions $landing\ spot = L(starting\ point, windspeed)$; $p_i = T(t)$, and $flight\ cost f_i = C(landing\ spot, g, w, d_i ...)$.

We define the objective function $O = p_i t_i - f_i$. The function is subject to $p_i \leq 9$ and that $d_1 \geq 15$. It is important to notice that the system is not necessarily linear.

6 Network flow

Model the network flow problem where the warehouses have finite bounded capacity as a linear program.

Solution. We will consider the problem where there is only one source node s. Otherwise, if we have many sources we create a new source with unbounded capacity to transfer to all sources. We do the same for the sink t.

Let $G = (V, E)$ be the graph, f_{ij} the flow between nodes v_i and v_j, and c_{ij} the capacity for the link between v_i and v_j. The classical problem is stated as:

$$\begin{gathered} O = \textstyle\sum_i f_{si},\ subject\ to \\ \textstyle\sum_i f_{ij} = \sum_k f_{jk} \quad \forall j \notin \{s,t\} \\ f_{ij} \leq c_{ij} \quad \forall i,j \end{gathered}$$

We can interpret the 'capacity of a warehouse' in two ways. One way to interpret it (the easy way) is that no more than cap_j flow can go through warehouse j. To model this, we should add the following constraint:

$$\sum_i f_{ij} \leq cap_j \quad \forall j$$

A more complex interpretation is that warehouses have limited storage capacity. This would allow the ingress flow to exceed the egress flow for a limited duration of time. Specifically, if the storage capacity of warehouse j is B_j, then, denoting the flow on link ij at time t by $f_{ij}(t)$,

$\sum_i \int_{t_1}^{t_2} f_{ij}(t)dt \leq \sum_k \int_{t_1}^{t_2} ff_{jk}(t)dt + B_j \quad \forall j \notin \{s,t\}, \forall t_1, t_2$

so that the ingress flows, integrated over all time periods, never exceed the egress flows, integrated over the same time period, taking into account the possibility of storage of B_j in the warehouse.

7 Integer linear programming

Generalize Example 6 to the case where n users can schedule jobs on one of the k machines, such that each user incurs a specific cost and gains a specific benefit on each machine at each of the m time periods. Write out the ILP for this problem.

Solution. We use x_{ijh} as the variable whose value is 1 if the user i is assigned to schedule a job on machine h at period j. We consider c_{ijh}/g_{ijh} the cost/gain for user i on assigning a job in machine h at period j. The function we want to optimize is:

$$O = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{h=1}^{k} (g_{ijh} - c_{ijh}) x_{ijh}$$

The constraints are that each $x_{ijh} \in \{0, 1\}$, which makes the problem an ILP. The second constraint is that at each time, each machine can have at most one job scheduled, so:

$$\sum_{i=1}^{n} x_{ijh} \leq 1 \quad \forall j, h$$

8 Weighted bipartite matching

Suppose you have K balls that need to be placed in M urns such that the payoff from placing the k-th ball in the m-th urn is p_{km}, and no more than 2 balls can be placed in each urn. Model this as a weighted bipartite matching problem.

Solution. We assume that we are trying to maximize the payoff (sum). The matching allows only to connect one ball to one urn, so we will modify the elements of the bipartite graph as follows:. We create a new set M', this new set contains $2n$ elements, where n is the number of urns in M. We will label this elements as u'_{im}, $i \in \{1, 2\}$. Every element in M', u_{im}, represents a position (1 or 2) in urn m. This allows to put, at most, two items in each urn. Now, we create the connections between K and M', for which $m_{k,im} = m_{k,m} \forall i \in \{1, 2\}$. And now the problem can be solved as a weighted bipartite matching.

9 Dynamic programming

You are given a long string L of symbols from a finite alphabet. Your goal is to find matches of a substring of L with a shorter string S from the same alphabet. The catch is that match need not be exact: each non-matching symbol gets penalty of 1. So, you can have a trivial 'match' with a penalty of $|S|$. What you want is to output all matching substring of L along with the corresponding penalty. Use dynamic programming to solve this problem.

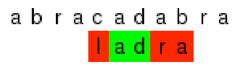

Figure 3: Example for the first version of approximate matching.

Solution. Let k be the number of errors in the match, $n = |L|$ and $m = |S|$. We will consider two versions of this problem. In the first version we will not allow deletion or insertion in any string for completing a match. Figure 3 shows an example of searching "ladra" inside "abracadabra" (at position 5) shows a match with $k = 3$, red cells represent errors and green cells represent matches. For this problem we define $D[i, j]$ which represents the score computed for $L[1, i]$ and $S[1, j]$. Note that $D[i, j]$ is undefined for $j < i$. We fill the table the following way:

$$D[i, j] = D[i-1][j-1] + \delta(L[i], S[j])$$

Where,

$$\delta(a, b) = \begin{cases} 1 \ \textit{if} \ a \neq b \\ 0 \ \textit{if} \ a = b \end{cases}$$

The table is filled by columns from left to right, and every column in filled from top to bottom. Note that the algorithm takes $O(nm)$ to fill the whole table, this is the same time that is required to compare every position using brute force.

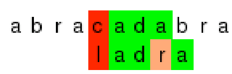

Figure 4: Example for the second version of approximate matching.

The second version considers the case where we can delete, insert or modify symbols in L or S. Figure 4 shows an example of searching "ladra" inside "abracadabra" (at position 5) shows a match with $k = 2$, green cells represent matches, red a modification and orange a deletion. In this case the dynamic programming changes a little bit, using the same δ function it is:

$$D[i, j] = \min(D[i-1][j-1] + \delta(L[i], S[j]), D[i-1][j], D[i][j-1])$$

In this case the algorithm takes $O(mn)$ time too, yet the brute force version does not achieve the same time so easily.

For both versions we assume that $D[i, j] = 0$ for $j \leq 0$ and $D[i, j] = j$ for $i \leq 0$. In the first version some of these values are not really defined in the table, but these conditions do not affect the solution.

10 Lagrangian optimization

Use Lagrangian optimization to find the optimal value of $z = x^3 + 2y$ subject to the condition that $x^2 + y^2 = 1$ (i.e., the points (x, y) lie on the unit circle).

Solution. Every function is continous and at least twice differentiable. We define $F(x, y, \lambda)$:

$$F(x, y, \lambda) = x^3 + 2y + \lambda(x^2 + y^2 - 1)$$

Now we impose that $\nabla F = 0$:

$$\frac{\partial F}{\partial x} = 3x^2 + 2\lambda y = 0 \quad (1)$$

$$\frac{\partial F}{\partial y} = 2 + 2\lambda y = 0 \quad (2)$$

$$\frac{\partial F}{\partial \lambda} = x^2 + y^2 - 1 = 0 \quad (3)$$

If we solve (1) we get two solutions for x, x_1 and x_2:

$$x_1 = 0, x_2 = -\frac{2\lambda}{3}$$

For x_1 we get that $y_1 = 1, y_2 = -1$ from (3) and $\lambda_1 = -1, \lambda_2 = 1$ from (2). Thus, the optimal values are 2 and -2.

For x_2 we get that $y = -\frac{1}{\lambda}$. Replacing x_1 and y in (3) we end up with the following equation:

$$\left(-\frac{2\lambda}{3}\right)^2 + \left(-\frac{1}{\lambda}\right)^2 = 1 \quad (4)$$

If we solve this equation, we obtain an imaginary value for λ, thus x and y are inmaginary too.

Finally, only x_1 leads to valid solutions for the system, so the maximum value of f, subject to the given constraints, is 2, achieved at the point $(0, 1)$. The minimum is -2, achieved at the point $(0, -1)$.

11 Hill climbing

Suppose you know that the objective function you are trying to maximize has more than K local optima. Outline an algorithm that is guaranted to find the global optimum using hill climbing.

Solution. We start with K random points and compute the optimal value reached at each point. If we have K, we return the best point. Otherwise, we eliminate the repeated results, lets say r of them, and then we start again with r points and repeat the process (remembering those already computed). When we reach K different points the algorithm finishes and knows the global optimum. Note that we could iterate infinitely until finding the K local optimas. However, without making any assumptions about the space we cannot assure a better method that finds the global optimum.