Linear Programming

Homework 2

Nick Morris

03/15/2016

Algorithm

Code

The computer language used to model this algorithm is R. The code is in the impS_HW 2.R file provided. If you do not have R, you can open up this file in notepad or notepad++. If you open it in notepad++ then go to Language → R → R and this will highlight key words and comments to make reading the code easier. There are three sections: the algorithm, the solution, and the iteration table.

Improvement Direction

The improvement direction is determined by the gradient of the objective function at the current basic feasible solution. If the objective function improves by minimization then we compute:

$$-\nabla obj(x_1, x_2, \ldots, x_n) \bullet \nabla obj(x_1, x_2, \ldots, x_n)$$

If this dot product results in a value less than zero then the improvement direction is chosen as $-\nabla obj(x_1, x_2, \ldots, x_n)$. If the objective function improves by maximization then we compute:

$$\nabla obj(x_1, x_2, \ldots, x_n) \bullet \nabla obj(x_1, x_2, \ldots, x_n)$$

If this dot product results in a value greater than zero then the improvement direction is chosen as $\nabla obj(x_1, x_2, \ldots, x_n)$. For both cases, if their specified condition for their dot product isn't met then there is no direction of improvement, so a zero vector is chosen as the improvement direction.

Step Size

The maximum step size for each constraint is determined by reducing each constraint to one of the following equations:

```
[+, -] λ [<, <=, =, >=, >] ( b – A*BFS ) / |A*∇x|
```

where `[...]` *indicates possible options, resulting in different equations*
where `λ` *is a scalar representing the maximum step size*
where `b` *is a vector of the right-handside of the given constraints*
where `A` *is a matrix of the left-handside of the given constraints*
where `BFS` *is a vector of the current solution*
where `∇x` *is a vector of the improvement direction*

This would be best explained by a couple examples, first let `( b – A*BFS ) / |A*∇x| = [+, -] C`, where `C` is a real non-negative scalar. The first example is as follows:

```
- λ <= [+, -] C
```

This would result in `λ` taking a value of infinity because it can increase in value infinitely while being less than the right-hand side of the constraint. Another example is as follows:

```
- λ > - C
```

This would result in `λ` taking a value up to `C` but not equal to, so for our purposes we choose `λ = C – 0.1`.

Every constraint is reduced, and then evaluated to determine its maximum value of λ, which can only take four values: zero, infinity, C, or approximately C. After the maximum λ is determined for every constraint, the chosen step size is the minimum of these values. The chosen step size is then multiplied by a positive value less than or equal to one. The reason for reducing the value of the chosen step size is to prevent step sizes that always move to constraint boundaries, which can result in stepping over local optima's in the feasible region.

Optimality

Optimality is checked at two points in the algorithm for every iteration. The first point is after the improvement direction at the current BFS has been calculated. If the improvement direction is the zero vector, then we've reached a local optimal where there is no direction of improvement. The second point is after the minimum step size has been chosen. If the step size is zero, then we've reached a local optimal where stepping forward in the improvement direction would result in a constraint being violated.

Unboundedness

Unboundedness is checked after the minimum step size has been chosen. If the step size is infinity then the problem is unbounded because the objective function will improve infinitely in the improvement direction. The algorithm will stop immediately before this improvement occurs.

New Starting Point

The new starting point is computed after Optimality and Unboundedness hasn't occurred. The computation is as follows:

$$BFS` = BFS + \lambda * \nabla x$$

where BFS` *is a vector representing the new starting point*
where BFS *is a vector of the current solution*
where λ *is a scalar representing the step size*
where ∇x *is a vector of the improvement direction*

The next iteration occurs beginning with determining the improvement direction at the new starting point. This algorithm will iterate continuously until optimality or unboundedness occurs.

Results

Iterations

The total number of iterations run to obtain the local optimal solution was `9291`. Also, the scalar multiple `1/3` was used to reduce each step size.

Objective Value

The value of the objective at the local optimal solution is `0.9901772`. Given that the objective function is a product of sine and cosine functions, the theoretical maximum value is `1`.

Solution

The local optimal solution is `(x, y) = (0.01945391, 4.850613).` This was found by the starting point `(x, y) = (4.5, 13)`.

Iteration Table

Each row of the iteration table indicates the iteration number of the `BFS`, the objective value of the `BFS`, the step size used to reach the `BFS`, the direction vector used to reach the `BFS`, and the `BFS` itself. Due to the large size of the iteration table, it can be found in the excel file: HW 2, in the tab: Solution, that has been provided.