Optimization for Computer Science Assignment 5

January 17, 2017

Submission: Upload your implementation and report zipped (`MatrNr.zip`) to the TU-Graz TeachCenter using the MyFiles extension.
Deadline: February 7, 2017 at 23:58h.

1 Shortest Path

In this assignment we will revisit the shortest path problem from the first assignment. This time, you will implement your own solver for the linear program (LP). In order to give the problem more practical relevance, we provide functions to convert images of labyrinths to an appropriate graph structure (see fig. 1). Your task is the implementation of the solver based on the affine scaling method.

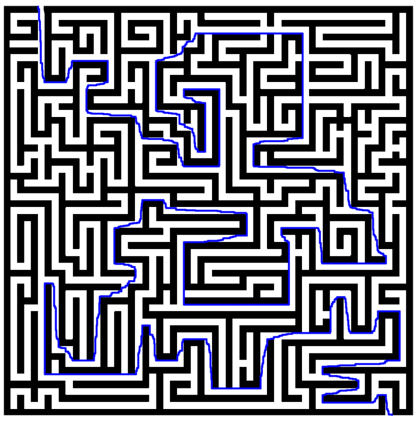

Figure 1: Automatic computation of the shortest path through the labyrinth.

1.1 Method

The construction of the graph and appropriate constraint matrix A, constraint vector b and weight vector c are already provided in the framework. The meaning of the terms is the same as in assignment 1. Thus we can set up a LP with equality constraints to compute the shortest path

$$\min_x c^T x \quad \text{s.t. } Ax = b,\ x \geq 0, \tag{1}$$

where x is the solution vector that indicates which edges of the graph are part of the shortest path. The LP (1) is of the form that we can apply a scaled projected gradient descent. The resulting algorithm is known as *affine scaling*. It falls in the category of so-called *interior point methods* and is very efficient for solving large scale LPs. The affine scaling algorithm is defined as

```
Data: Linear program of the form $\min_x c^T x \quad \text{s.t. } Ax = b,\ x \geq 0$.
  1. Choose a feasible starting point $x^0 \geq 0$
  2. Iterate for $k \geq 0$ :
while True do
    (i) $H^k = \left(\text{diag}(x^k)\right)^{-2}$
    (ii) Gradient step: $x^{k+1} = x^k - \alpha^k (H^k)^{-1} \left[ I - A^T \left( A(H^k)^{-1}A^T \right)^{-1} A(H^k)^{-1} \right] c$
         The step size $\alpha^k$ has to be chosen such that $x^{k+1} > 0$ holds. This can be done
         in closed form by computing the largest $\bar{\alpha}^k$ that still ensures $x^{k+1} \geq 0$ and
         setting $\alpha^k = 0.99\bar{\alpha}^k$.
    if stopping criterion then
        exit
    end
end
```

Algorithm 1: Affine scaling algorithm

The first step of the algorithm is the selection of a feasible starting point. Often this turns out to be a quite hard problem itself, sometimes as hard as the solution of the LP. In our case, a good strategy for finding a feasible x^0 is

$$x^0 = \bar{x} - A^T \left[\left(AA^T \right)^{-1} (A\bar{x} - b) \right], \tag{2}$$

where $\bar{x}$ is any vector fulfilling the constraint $\bar{x} \geq 0$, e.g. $\bar{x} = \mathbf{1}$ (a vector consisting of ones).
Computation of the optimal stepsize (step (ii) in alg. 1) can be done explicitly in closed form. To solve the linear problems that arise in the affine scaling method, you can use any direct solver[1].

1.2 Tasks

- Implement the algorithm to solve the LP.
- Define a suitable stopping criterion for alg. 1.

[1]python: `scipy.linalg.solve` for dense matrices and `scipy.sparse.linalg.spsolve` for sparse matrices

- Provide a plot of the energy of the linear objective function of problem (1) during the iterations.
- Compare your solver to the off-the-shelf solver you used in assignment 1. What do you find?
- Try different sized labyrinths with both algorithms and compare the runtime. Try your own images and include your findings in the report!

1.3 Framework

We provide a framework with functionality to convert images to a graph structure and to set up the LP. There is also a method to visualize the solution as well as some other functions you might find useful. As this assignment will be tested automatically, you have to obey the submission rules. Otherwise, you are free to change the script to you liking.

Submission Upload your implementation and report zipped (filename: `MatrNr.zip`) to the TeachCenter. Don't forget to delete your old submission.

Adhere to the following submission rules:

- The script name must be `shortest_path.py` and it must run in python version 2.7
- The script must accept one parameter to load a labyrinth image (already implemented)
- The script must produce an image of the shortest path called `shortest_path.png` using the function `graph.visualize_solution()` (already implemented)
- The script must finish without waiting for user input, i.e. remove all `plt.show()` or other calls that block execution.

You must not import any additional python modules besides the ones that are already present!