European Social Fund Prague & EU: We invests in your future.

Combinatorial optimisation
The shortest paths in a graph

Přemysl Šůcha (suchap@fel.cvut.cz)

March 19, 2014

1 The shortest path problem

The shortest path problem, which is often used in many applications, is believed to be one of the most important problem in graph theory. This problem is specified by directed graph G where each edge $e \in E(G)$ is weighted by real number $a(e)$ corresponding to the length between incidence nodes of edge e [1]. After that, the path length can be calculated as the sum of edge lengths of the path edges. Let x and y are graph nodes then distance $u(x, y)$ from node x to node y is defined as the shortest path from x to y. If the path does not exist then value $u(x, y)$ is set to inf. One of the most important properties of this problem is *triangle inequality*.

Theorem 1.1 *If the graph is without negative cycles then the following inequality is satisfied for all triple of nodes.*

$$u(i, j) \leq u(i, k) + u(k, j). \tag{1}$$

To solve the shortest path problem *Dijkstra's algorithm* can be used. The algorithm works out correctly if edge weights are positive. It is possible to modify the algorithm to deal with negative edges at the cost of higher algorithmic complexity.

Algorithm 1 Dijkstra's algorithm.

Require: Graph $G(V, E)$.
Require: Starting node r.
Require: Edge weights $a : E(G) \to \mathbb{R}^+$ such that for all edges is satisfied $a(e) \geq 0$.
Ensure: The shortest paths from r, i.e. $U(v)$, $FROM(v)$.

```
U(r) := 0
U(v) := inf        ∀v ≠ r
D := ∅
FROM(v) := inf     ∀v ∈ V
while arbitrary v ∈ (V \ D) such that U(v) ≠ inf do
    From set V \ D select node x with the minimal U(x) value.
    D = D ∪ (x)
    for all e ∈ E+(x) do
        y = end node of e
        if U(x) + a(e) < U(y) then
            U(y) := U(x) + a(e)
            FROM(y) = x
        end if
    end for
end while
return U, FROM
```

The list of closed nodes (i.e. the distance from node r is final) is stored in D, to be able to reconstruct the shortest paths $FROM$ array is required. Value $FROM(y)$ is the index to the next-to-last node x of the shortest path $p(r, y)$, therefore edge $e = (x, y)$ is the last edge of the shortest path. Algorithm complexity of the standard Dijkstra's algorithm is $O(n^2)$.

2 Approximating functions

Approximation of piecewise linear functions is lesser-known, however quite an interesting application of the shortest path problem [2]. As an input a set of n points is given, i.e. $\{(x_i, f(x_i))\}$ (see Figure 1). The goal is to select the smallest possible number of points on condition that approximation error is reasonable (see Figure 2).

```
   x = [0   1.26   2.51   3.77   5.03   6.28];
f = [0.01   1.16   0.70   -0.34   -0.80   0.2100];
```

Figure 1: Input data of the function.

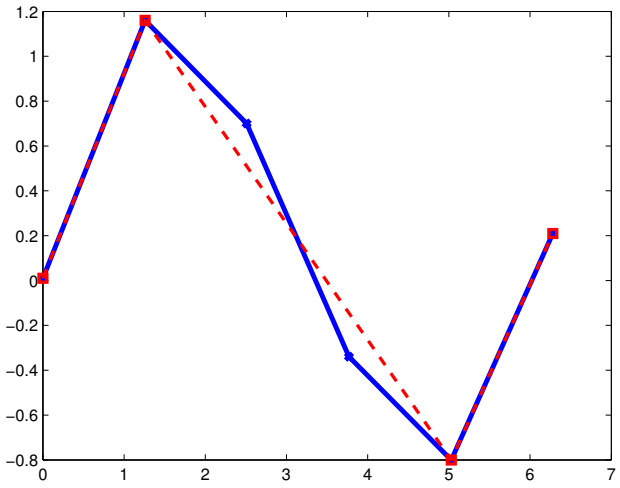

Figure 2: The input function (blue) and the approximated function (red).

This problem can be solved using the shortest path problem applied to graph G where each node v_i corresponds to a data point $(x_i, f(x_i))$ and each edge $e = (i, j)$ (such that $i < j$) has weight $c_{i,j}$ which corresponds to a penalty if the part of the function from $(x_i, f(x_i))$ to $(x_j, f(x_j))$ is approximated by a line segment. The penalty consists of the approximation error (weighted by β) and the number of required samples (weighted by α). An example of such graph can be seen in Figure 4 and penalty $c_{i,j}$ is calculated as follows.

$$c_{i,j} = \alpha + \beta \left[\sum_{k=i}^{j} \left(f(x_k) - f'(x_k) \right)^2 \right] \tag{2}$$

$f'(x_k)$ is a precomputed value of the approximating function.

$$f'(x) = f(x_i) + (x - x_i) \cdot \frac{f(x_j) - f(x_i)}{x_j - x_i} \tag{3}$$

The matrix of weights W and graph G corresponding to the data in Figure 1 are shown in Figure 3 and 4. Using the algorithm for the shortest path problem the optimal approximating function can be found.

```
c = [ inf     2.00  8.46 26.57 32.94 29.66; ...
      inf     inf   2.00  2.82  2.74 28.54; ...
      inf     inf   inf   2.00  2.84 23.42; ...
      inf     inf   inf   inf   2.00  7.42; ...
      inf     inf   inf   inf   inf   2.00; ...
      inf     inf   inf   inf   inf   inf];
```

Figure 3: The matrix of weights W (i.e. $c_{i,j}$ values for $\alpha = 2, \beta = 10$).

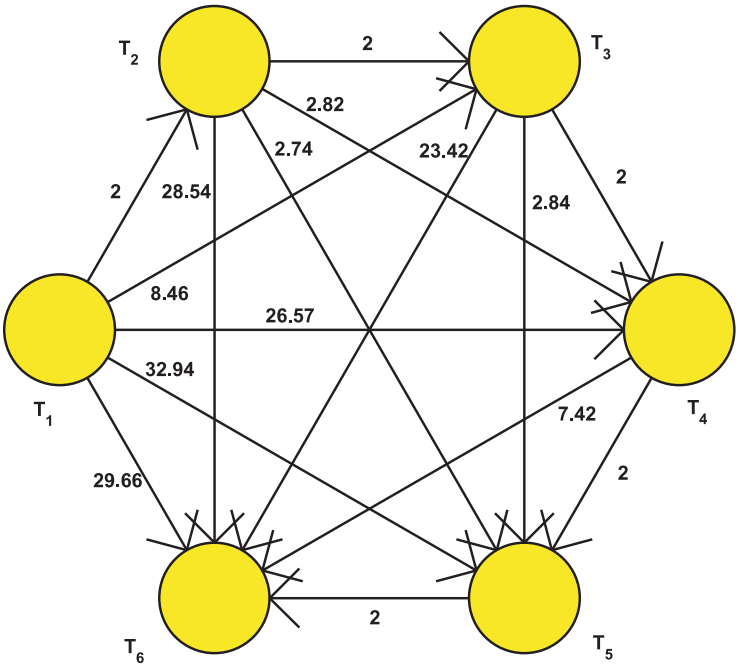

Figure 4: The corresponding graph G.

3 A seminar assignment

The following function is stated.

```
x = [0.8  2.15  2.9  4.1];
f = [0.1  0.7  1.3  0.65];
```

A seminar assignment: Calculate the matrix of weights (compute $c_{i,j}$ values for $\alpha = 2, \beta = 10$) and solve the shortest path problem to find approximating function. In the end sketch both original and approximating function.

4 A homework assignment

A homework assignment, part 1: Implement the Dijkstra's algorithm and use it to find the approximating function. The input data is in Figure 3 (the function is depicted in Figure 1). In the end the result should be printed as it is shown in Figure 2.

Hints: To display results you can use functions `plot` and `hold on`. To deal with the sets `union`, `intersect`, `setdiff` and `ismember` functions are helpful. To get the minimal value and its index in the vector `min` function can be used.

In a similar way the shortest path problem can be applied to some vector pictures which are composed of points (x_i, y_i). The difference is that weights $c_{i,j}$ are computed as a sum of squared distances from points (x_k, y_k) to the line segment defined by (x_i, y_i) and (x_j, y_j) points where $i < k < j$. An example of two-dimensional approximation, where the frontiers of the Czech Republic are input data, is depicted in Figure 5.

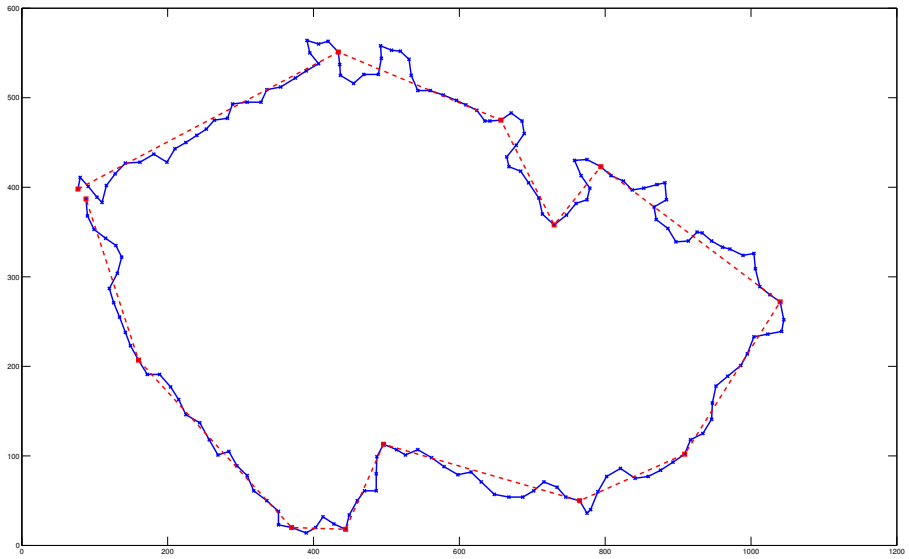

Figure 5: Entered points (blue) and selected points (red).

A homework assignment, part 2: Use your proposed solution to smooth frontiers of the Czech Republic. The procedure is almost the same with the difference in calculating weights $c_{i,j}$ – mentioned above Fig. 5. Parameters α and β should be reasonably selected with respect to computational requirements and the precision of the approximation. The data can be obtained from the subject homepage under the link to this tutorial.

References

[1] J. Demel, *Grafy a jejich aplikace*. Academia, second ed., 2002.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall; United States Ed edition, 1993.

[3] B. H. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer, third ed., 2006.