Solution in TPK 4191 - Production optimization and control

Jørn Vatn
Email: jorn.vatn@ntnu.no

September 8, 2024

Problem 2.8

Model parameters

Assume that all travelling times from node i to node j, i.e., $d_{i,j}$ is saved in a matrix with corresponding element `d[i,j]`. Further assume that the start node (S) has number 0, and `destination` is the number of the destination node (5).

The essential recursive code is:

$$D(j) = min_i\{d_{i,j} + D(i)\}$$

where $D(j)$ is the lowest total travelling time from the start node to node j. The Python code for implementing $D(j)$ is:

```
def D(j, predecessor=[0]):
    if j == 0 :
        return 0
    else:
        best = 1E+30
        for i in range(destination):
            if i != j:
                if d[i, j] > 0:
                    test = d[i, j] + D(i)
                    if test < best:
                        best = test
                        iBest = i
        predecessor[0] = iBest
        return best
```

The main program reads:

```
predecessor=[0]
n = 5
while n > 0:
    dd=D(n, predecessor)
    print( "To node ", n, " from node " ,predecessor[0],  " -  total
                                                distance =",dd)
    n = predecessor[0]
```

Note we start calling `D(5)`, i.e., total distance to the end node (5). This function returns in `predecessor` the best node we could reach node (5) from. This appears to be `predecessor` = node (4). Then we call `D(predecessor)` and find which node it was best node we could reach node (4) from etc. In this manner we find the entire trajectory from the star node to the destination node. The print is shown in the opposite direction.