

Network Optimization (Fall 2008) – Brief Solutions to Homework 5

1. The set of paths and cycles can be $\mathcal{P} = \{P_1, P_2, P_3\} = \{1-2, 1-2-4-5, 1-3-2-4-5\}$, $\mathcal{W} = \{W_1, W_2\} = \{2-4-5-2, 2-4-5-3-2\}$ with $f(P_1) = 5$, $f(P_2) = 6$, $f(P_3) = 4$, $f(W_1) = 2$, and $f(W_2) = 1$. The decomposition is not unique.
2. We can prove this result using the related result on the necessary and sufficient condition for the existence of Eulerian cycle in an undirected graph.

Lemma 1.1. *An undirected graph has an Eulerian cycle iff every node has an even degree.*

Proof: (\Leftarrow) The Eulerian cycle “enters” and “leaves” a node once each time it visits the node, thus contributing an even degree.

(\Rightarrow) We adapt the flow decomposition algorithm to the case of undirected graphs (with no supply/demand nodes as well). Start from any node, do DFS. Since every node has even degree, we can reach a node, and also leave it. Remove from the graph each arc traversed, adjust degrees of nodes associated with the arc, and store any cycles as discovered. We are guaranteed to return to a node already visited (after traversing at most n arcs). If the current graph is empty of arcs, stop. Else, start from a node that is part of a cycle *already found*, and repeat. We are always guaranteed to find a cycle as long as there are arcs left in the graph, as all nodes remaining will have even degrees. This procedure is guaranteed to terminate, as there are only a finite number of arcs. The output is a set of *arc-disjoint* cycles, which fully cover the arc set of the original graph. We can merge these cycles to obtain an Eulerian cycle. Nodes with degree $2k$ for $k \geq 2$ (in the original graph) are exactly the nodes that share k cycles. \square

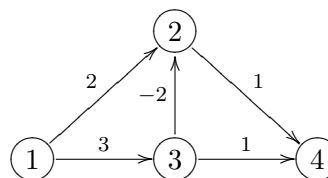
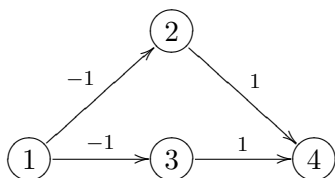
Now we prove the main result in question. First of all, notice that the statement makes sense only for $k > 0$ (when $k = 0$, we get a single Eulerian cycle). We consider a modification of the original graph, with $2k$ nodes having odd degrees, as follows. We add k arcs to the graph each between two of the odd-degree nodes. The resulting graph will have all node degrees even, and hence have an Eulerian cycle, by Lemma 1.1. Starting from the Eulerian cycle, we remove the k additional arcs added (in any order). The first arc removed changes the Eulerian cycle to an Eulerian *walk* of the original graph. Each subsequent arc removal breaks a bigger walk in to two smaller walks, incrementing the number of walks by one. Note that we are not removing any of the arcs from the original network. When we have removed all the extra arcs added, we will get k arc disjoint walks, which cover all arcs in the original network.

3. Create a graph with nodes $0, 1, \dots, n$. Add arcs (i, j) for all $i < j$. Arc (i, j) models the situation when all the books of heights H_{i+1}, \dots, H_j are arranged in shelves of height H_j . To this end, set its cost $c_{ij} = F_j + C_j \sum_{k=i+1}^j L_k$. The shortest path from 0 to n gives the cheapest arrangement of books.
4. We can essentially repeat the inductive argument that is used to prove the correctness of Dijkstra’s algorithm here. We perform induction on the number of marked nodes in the BFS, and simultaneously on the cardinality of the set S as maintained by Dijkstra’s algorithm. Assume that the `BFS_order` and `Dijkstra_order` are identical for the first k nodes – marked by BFS, and also in the set S of Dijkstra. In BFS, all the arcs in the out-arc list of the current node are examined for admissibility before going to the next node. Similarly, in Dijkstra’s algorithm, the distance labels of all arcs in the out-arc list of the current node are updated as it gets moved to S from \bar{S} . The $k + 1$ -st node marked by BFS will be connected by a single

admissible arc to the current node. The $k + 1$ -st node considered by Dijkstra will have the smallest $d(j)$ value among all $j \in \bar{S}$ (breaking ties arbitrarily), and will be connected to a node in S by a single arc. Since all $c_{ij} = 1$, and by the induction assumption, the heads of the nodes of all the admissible arcs from the current node in BFS will exactly be the same nodes $j \in \bar{S}$ with the identical minimum $d(j)$ value, and hence will be examined in the same order by Dijkstra's algorithm as well. Hence `BFS_order` and `Dijkstra_order` remain identical for step $k + 1$.

Since the BFS algorithm runs in $O(m)$ time, the result on the complexity of the shortest path problem follows for networks with unit cost arcs.

- The numbers listed on arcs are the costs. Dijkstra's algorithm works correctly on the first instance, but fails on the second instance.



- See course web page for MATLAB code `sp_dijkstra.m`.