Assignment 4 – Solutions

COMP3121/9101 22T1

Released April 5, due April 19

This document gives model solutions to the assignment problems. Note that alternative solutions may exist.

1. (20 points) You are making an exam of k questions, which is divided into m sections. The ith section must contain exactly p_i questions, where $\sum p_i = k$. There are n questions available in the question bank, each of which can be used at most once in the exam. There is a further restriction: for each section i and question j, you are given a boolean $a_{i,j}$ which records whether question j can be placed in section i or not.

   Design an algorithm which runs in time polynomial in m and n and determines whether a valid exam can be formed.

Solution

Construct a flow network with:

- source s and sink t
- a vertex x_i for each question
- a vertex y_j for each section
- an edge from s to each x_i with capacity 1
- for each pair (i, j) such that $a_{i,j}$ is true, an edge from each x_i to each y_j with capacity 1
- an edge from each y_j to t with capacity p_j.

We claim that a valid exam can be constructed if and only if the value of the maximum flow is k. Observe that each unit of flow in this graph assigns a question to a section in which it can be placed. The edges from the source enforce that no question is used more than once, and the edges to the sink enforce the section limits. The exam can be formed if and only if all sections can attain their limit at once, i.e. if each edge to the sink is fully occupied with flow, in which case the maximum flow has value $\sum p_i = k$.

The maximum flow is found using the Edmonds-Karp algorithm in $O(VE^2)$. This is polynomial in n and m since the graph has $n + m + 2$ vertices and up to $nm + n + m$ edges.

2. (20 points) You are going on holiday. There are n cities in the world. There are also m airlines, the ith of which operates flights in both directions between cities a_i and b_i. You live in city 1, and you would like to travel to city n and back. You don't mind stopovers in other cities, but you don't want to have more than one stopover in any city.

   Design an algorithm which runs in time polynomial in n and m and determines whether it is possible to travel from city 1 to city n and back to city 1 without going through any other city more than once.

Solution

Construct a flow network with:

- source vertex 1^{out} and sink vertex n^{in}
- for each city i where $1 < i < n$, two vertices i^{in} and i^{out}, with an edge of capacity 1 from i^{in} to i^{out}
- for each airline j joining a_j and b_j, edges of capacity ∞ from a_j^{out} to b_j^{in} and an edge from b_j^{out} to a_j^{in} (omitting any edges to 1^{in} or from n^{out}).

We claim that the journey can be completed if and only if the maximum flow is at least 2. Each unit of flow in this graph corresponds to a path from city 1 to city n, and we can follow one of these paths in reverse to return from city n to city 1. Importantly, the in- and out-vertices and the edges joining them enforce that each city is used at most once for a stopover.

The answer is determined using the Ford-Fulkerson algorithm, with early exit after a second augmenting path is found. This algorithm runs in $O(E) = O(n + m)$, which is clearly polynomial in n and m.

3. (20 points) You are manufacturing integrated circuits from a rectangular silicon board that is divided into an m by n grid of squares. Each integrated circuit requires two adjacent squares, either vertically or horizontally, that are cut out from this board. However, some squares in the silicon board are defective and cannot be used for integrated circuits. For each pair of coordinates (i, j), you are given a boolean $d_{i,j}$ representing whether the square in row i and column j is defective or not.

   Design an algorithm which runs in time polynomial in m and n and determines the maximum number of integrated circuits that can be cut out from this board.

Solution

Construct a graph with:

- a vertex $v_{i,j}$ for each non-defective cell, and
- an edge between each pair of neighbouring non-defective cells.

This graph is bipartite, as each edge joins a vertex where $i + j$ is odd with a vertex where $i + j$ is even.

Placing a circuit corresponds to selecting an edge in this graph. No cell can be part of more than one circuit, so no two edges can share a vertex, i.e. our selection of edges must constitute a matching. Therefore, the maximum number of circuits that can be placed is exactly the size of the maximum matching in this bipartite graph. This can be found using the Edmonds-Karp algorithm in $O(VE)$ time, and since $V \leq mn$ and $E \leq 4mn$ the runtime is clearly polynomial in m and n.

4. (20 points) You are in charge of an electrical network. In this network, power is transmitted from x generators to y customers through z substations, so there are $n = x + y + z$ facilities in total. There are also m one-way wires, the ith of which goes from facility a_i to facility b_i and costs c_i to disconnect. You are guaranteed that all c_i are integers. Note that:

   - there are no wires to any generator or from any customer,
   - there can be a wire from a generator directly to a customer, and
   - there can be wires between substations.

   Some urgent repairs are needed, so you will have to take the entire network down, that is, disconnect wires so that power cannot travel from any generator to any customer. However, you will also have to make a trip to the site of each chosen wire, and you would prefer to leave your office as little as possible.

   Design an algorithm which runs in time polynomial in n and m and determines a set of wires which:

   - has minimum cost to disconnect, and
   - requires the fewest trips out of all minimum cost sets.

Solution

Construct a flow network G with:

- source s and sink t
- a vertex for each facility
- an edge from s to each generator, with infinite capacity
- an edge corresponding to each wire, with capacity c_i
- an edge from each customer to t, with infinite capacity.

Any finite-capacity cut in this graph will disconnect all generators from all customers (since the edges from s and to t have infinite capacity). To minimise the cost, we must find a minimum cut. We also have to minimise the number of trips, so we look for a minimum cut comprised of as few edges as possible.

To achieve this, we construct a new graph G' by changing only the (finite) capacities from c_i to $(m + 1)c_i + 1$. Now, for a cut (S, T) with capacity $c(S, T)$ in G would have capacity $(m + 1)c(S, T) + |S, T|$ in G', where $|S, T|$ is the number of forward edges across the cut. Note that $1 \leq |S, T| \leq m$, so the minimum cut in G' must minimise $c(S, T)$ first, and then $|S, T|$, which exactly corresponds to the criteria in our problem.

To find which wires to disconnect, we first run the Edmonds-Karp algorithm, and after it terminates we perform one more breadth-first search from s. The minimum cut will then be (S, T) where S contains all vertices seen in this search, and T contains all others (including t). We disconnect those wires which go from a vertex in S to a vertex in T.

Constructing both graphs takes $O(V + E)$ time, and finding a minimum cut in G' takes $O(VE^2)$. Since $V = n + 2$ and $E = x + y + m \leq n + m$, the time complexity is clearly polynomial in n and m.