Assignment #5

Greg Plaxton

November 2, 2012

There are two parts to this assignment. The first part, described in Section 1 below, consists of a programming task that is due by 8pm on Monday, November 12. Students are allowed to work on the first part with a partner, and are strongly encouraged to do so. Each team should turn in only one solution for the first part. If you are having trouble finding a partner, send me an email and I will match you up with someone else in the same situation.

The second part, described in Section 3 below, consists of a number of exercises related to the lecture material, and is due at the beginning of class on Monday, November 19. Each student should work on this part separately.

NOTE: The next (and last) assignment will not include a programming task, so you should feel free to use up all of your remaining slack time (up to a maximum of one week) on the programming portion of this assignment.

1 Programming & Problem Solving: Mooov Around

In this assignment you will implement a polynomial-time algorithm for the Mooov Around problem with general preferences. The algorithm that you are to implement is a generalization of the algorithm that we developed in Assignments 2 and 3 for the special case of strict preferences. The input-output format is the same as in Assignment 1.

1.1 Configurations

Recall that we have n students, numbered from 0 to $n-1$, and n spaces, numbered from 0 to $n-1$.

We define a *configuration* $G = (U, V, E)$ as a bipartite graph with a set U of vertices on the "left", one for each student, a set V of vertices on the "right", one for each space, and a set E of directed edges satisfying the following constraints: (1) each edge goes from a student to a space, or from a space to a student; (2) each space has exactly one outgoing edge; (3) each student has exactly one incoming edge. The n space-to-student edges encode an allocation, as follows: If the outgoing edge of space j goes to student i, then space j is allocated to student i.

For any configuration $G = (U, V, E)$ and any space j in V, we define *student*(G, j) as the unique student i such that edge (j, i) belongs to E. For any configuration $G = (U, V, E)$ and any student i in U, we define *space*(G, i) as the unique space j such that *student*$(G, j) = i$.

For any configuration $G = (U, V, E)$, and any student i in U, we define *out*(G, i) as $\{j \mid (i, j) \in E\}$.

The *initial configuration* is the configuration $G = (U, V, E)$ in which *out*(G, i) is empty for all students i in U, and there is an edge from space i to student i for $0 \leq i < n$. Thus the allocation associated with the initial configuration is the university allocation.

For any configuration $G = (U, V, E)$, we define *satisfied*(G) as the set of all students i in U such that *space*(G, i) belongs to *out*(G, i), and we define *unsatisfied*(G) as $U \setminus$ *satisfied*(G).

For any configuration $G = (U, V, E)$ and any space j in V, we define *distance*(G, j) as the length of a shortest path from j to a student in *unsatisfied*(G). If there is no such path, we define *distance*(G, j) as ∞.

For any configuration $G = (U, V, E)$ and any student i in U, we define *next*(G, i) as follows: if *distance*$(G, j) = \infty$ for all j in *out*(G, i), then *next*$(G, i) =$ *nil*; otherwise, letting V' denote the set of all spaces j in *out*(G, i) minimizing *distance*(G, j), we define *next*(G, i) as the minimum space in V'. (Example: If V' consists of spaces 3, 7, and 11, then *next*(G, i) is space 3.)

For any configuration $G = (U, V, E)$, we define *pruned*(G) as the configuration $G' = (U, V, E \setminus E')$ where E' denotes

$$\{(i, j) \in E \mid i \in U \ \wedge \ j \neq \textit{next}(G, i)\},$$

and we define *cycles*(G) as the set of all directed cycles in *pruned*(G).

For any configuration $G = (U, V, E)$ and any cycle C in *cycles*(G), we define *trade*(G, C) as the configuration $(U, V, (E \setminus E') \cup E'')$ where E' denotes

$$\{(j, i) \in V \times U \mid j \in C \ \wedge \ \textit{student}(G, j) = i\}$$

and E'' denotes

$$\{(j, i) \in V \times U \mid i \in C \ \wedge \ \textit{next}(G, i) = j\}.$$

For any configuration G, we define *exhausted*(G) as the set of all students i in *unsatisfied*(G) such that *next*$(G, i) =$ *nil*.

1.2 Preferences

Let P denote the student preferences, i.e., all of the preference information provided by the students. We define *configs*(P) as the set of all configurations $G = (U, V, E)$ such that for any student i in U and any space j in *out*(G, i), student i prefers space j to any space in $V \setminus$ *out*(G, i).

Remark: It is easy to prove that for any student preferences P, configuration $G = (U, V, E)$ in *configs*(P), and any cycle C in *cycles*(G), the configuration *trade*(G, C) belongs to *configs*(P). End of Remark.

For any student preferences P, any student i in U, and any subset V' of V, we define $top(P, i, V')$ as the set of spaces j in V' such that student i likes space j at least as well as any other space in V'.

For any student preferences P, any configuration $G = (U, V, E)$ in $configs(P)$, and any student i in $exhausted(G)$, we define $reveal(P, G, i)$ as the configuration $(U, V, E \cup E')$ where

$$E' = \{(i, j) \mid j \in top(P, i, V \setminus out(G, i))\}.$$

Remark: It is easy to prove that for any student preferences P, any configuration $G = (U, V, E)$ in $configs(P)$, and any student i in $exhausted(G)$, the configuration $reveal(P, G, i)$ belongs to $configs(P)$. End of Remark.

2 Algorithm C

In this assignment you will implement the following nondeterministic algorithm, referred to as Algorithm C. Like Algorithms A and B (of Assignments 2 and 3, respectively), Algorithm C starts from the initial configuration, and iteratively updates the current configuration by performing an arbitrary enabled action until a configuration is reached in which no actions are enabled. Once the algorithm terminates, the output allocation is defined by the space-to-student edges in the last configuration.

It remains to define the set of enabled actions in a given configuration. Fix the given student preferences P, and let $G = (U, V, E)$ be an arbitrary onfiguration in $configs(P)$. For any cycle C in $cycles(G)$, a trading action is enabled for C in G, and the effect of performing this action is to replace G with $trade(G, C)$. For any student i in $exhausted(G)$, a revelation action is enabled for i in G, and the effect of performing this action is to replace G with $reveal(P, G, i)$.

Remark: It turns out that Algorithm C enjoys the same confluence property as we established for Algorithm B in Assignment 4. Thus all correct implementations of Algorithm C produce the same output, regardless of how the nondeterminism is resolved. End of Remark.

3 Exercises

1. Let $G = (V, E)$ be a flow network, and let f and f' be two maximum flows in G. Let S (resp., S') be the set of vertices that are reachable from the source in the residual network G_f (resp., $G_{f'}$). Prove that $S = S'$.

2. Let $G = (V, E)$ be a flow network, and let (S, T) and (S', T') be two minimum-capacity cuts of G. Prove that $(S \cap S', T \cup T')$ and $(S \cup S', T \cap T')$ are also minimum-capacity cuts of G.

3. The input to the longest path problem is a connected undirected graph $G = (V, E)$ with a pair of distinguished vertices s and t. The output is a longest simple path

between s and t. The decision version of the longest path problem takes the same input except that we are also give a positive integer k, and the goal is to determine whether there is a path between s and t of length at least k. In what follows we refer to the decision version of the longest path problem as LP.

(a) Prove that LP belongs to NP.

(b) See the description of the Hamiltonian Cycle (HC) problem on page 474 of the text. The HC problem is known to be NP-complete. Give a polynomial-time reduction from HC to LP, i.e., prove that HC$\leq_P$LP.