CS787 Homework 1

Mikola Lysenko

February 19, 2009

1. "Cuts in $(S, \mathcal{I})$ are precisely cycles in (S, I)"

We start by showing the forward direction of the equivalence: For any cut C of $(S, \mathcal{I})$ and maximal element $I \in \mathcal{I}$, there exists some non-empty $c \subseteq C \cap I$. By duality, $I^* = S - I$ is maximal in $(S, \mathcal{I}^*)$ and $c \cap I^* = \emptyset$, so $c \cup I^* \notin \mathcal{I}^*$. Removing any element from C would result in there being some I' where $C \cap I' = \emptyset$, resulting in C being independent in $\mathcal{I}^*$. Therefore C is a cycle of $(S, \mathcal{I}^*)$.

And now we finish our proof by showing the other direction: Given a cycle C of $(S, \mathcal{I}^*)$, for every maximal element $I^* \in \mathcal{I}^*$ there exists some non-empty $c \subset C$ such that $C - c \subseteq I^*$. Because $c \not\subset I^*$ and $I = S - I^*$, $c \subset I$ with I maximal in $(S, \mathcal{I})$. Because C is minimal, removing any element would cause there to be some maximal set in $\mathcal{I}^*$ containing C, so there would have to be some other maximal set in $\mathcal{I}$ disjoint from C. Therefore, C must be a cut of $(S, \mathcal{I})$.

"The Blue rule in $(S, \mathcal{I})$ is the same as the Red rule in $(S, \mathcal{I}^*)$ with the ordering of weights reversed"

We show this by playing a word-game. The Blue rule for $(S, \mathcal{I})$ is:

```
Find a cut with no blue element, pick an uncolored element of the cut of minimum weight and color it blue.
```

By the above result, this is the same as saying for $(S, \mathcal{I}^*)$:

```
Find a cycle with no blue element, pick an uncolored element of the cycle of minimum weight and color it blue.
```

Switching the term blue to red, and swapping orders gives the rule:

```
Find a cycle with no red element, pick an uncolored element of the cycle of maximum weight and color it red.
```

Which is precisely the red Rule for $(S, \mathcal{I}^*)$.

2. Suppose that $|A| \neq |B|$, then without loss of generality let $|A| < |B|$. By definition there exists some $b \in B$ such that $A' = A \cup \{b\} \in \mathcal{I}$. However, this contradicts the statement that A is maximal. Therefore we must conclude that all maximal elements of $\mathcal{I}$ are the same cardinality.

3. We first prove that each matroid, $(S, \mathcal{I})$, is a set system with the prescribed properties. Observe that for any $A \subset S$, $(S \cap A, \mathcal{I} \cap 2^A)$ is a matroid (by the second matroid axiom). From Problem 2 we know that for any maximal $I, J \in \mathcal{I} \cap 2^A$, $|I| = |J|$. Thus, the matroid $(S, \mathcal{I})$ satisfies the hypothesis.

To prove the opposite direction, we next show that some set system $(S, \mathcal{I})$ having the stated properties is a matroid. The first two matroid axioms are trivially satisfied by the non-emptiness and downward closure of $\mathcal{I}$[1]. To prove the third axiom, let $I, J \in \mathcal{I}$ with $|I| < |J|$ and pick maximal elements $K_I, K_J \in \mathcal{I} \cap 2^{I \cup J}$ such that $I \subseteq K_I$ and $J \subseteq K_J$. From the hypothesis, we know that $|K_I| = |K_J|$, $|K_J| \geq |J|$ and $|J| > |I|$, thus $|K_I| > |I|$. Therefore there must exist some $x \in K_I - I$ such that $I \cup \{x\} \in \mathcal{I}$. Moreover $K_I - I \subseteq J$, so $x \in J$, proving that $(S, \mathcal{I})$ is a matroid.

4. Let $(S, \mathcal{I})$ be a matroid with $I \in \mathcal{I}$ maximal, $x \notin I$ and $I' = I \cup \{x\}$. By definition $I' \notin \mathcal{I}$, so I' must contain at least one cycle. Suppose that there exist two distinct cycles $C, C' \subset I'$ and pick some $y \in C - C'$ to construct $I'' = I' - \{y\}$. By the cycle interchange lemma (proved in class), I'' is maximal in $\mathcal{I}$. However, $C' \subset I''$ which contradicts $C' \notin \mathcal{I}$, and therefore the cycle C is unique.

5. Suppose $A, B \subseteq G$ are transitive reductions of G and $A \neq B$. Because $|A| = |B|$, there must exist some edge $x \in A - B$ such that the transitive closure of $G - x$ is the transitive closure of G (because B is a transitive reduction and $B - x = B$). However, because G is directed, acyclic and simple, if some edge $x = (a, b)$ is in the transitive reduction, then there can be no other path from vertex a to b. Therefore, removing x from G must changes the transitive closure of G and we have a contradiction.

Because of the above result, we may examine each edge of G is any order, removing only those which change the transitive closure of G. The resulting minimal graph is then the unique transitive reduction of G:

```
TRANSITIVE-REDUCTION(G = (V, E))
1  G' ← G
2  for each e ∈ E
3        do if TRANSITIVE-CLOSURE(G' − e) = TRANSITIVE-CLOSURE(G)
4           REMOVE(G', e)
5  return G'
```

Because TRANSITIVE-CLOSURE can be computed in polynomial time and the main loop is called exactly once per edge, the total running time of TRANSITIVE-REDUCTION is polynomial in $|G|$.

[1] Errata added from email

6. First, observe that within any circuit there are an even number of edges (one in and one out each time a vertex is visited), thus every vertex of an Eulerian circuit has even degree.

Next, we prove the existence of an Eulerian circuit constructively for some graph $G = (V, E)$ with each vertex having even degree. To do this, we first show that for any vertex $v \in V$, there exists some cycle containing v. Initially, construct any path, P starting at v which does not contain any repeat edges. If P loops back to v at some point, then take this smaller loop as our circuit and be done with it. Otherwise, it must be possible to extend P because each vertex of G has even degree, so there must exist some other edge coming out from the head of P which may be added to P. The only case where this fails to hold is at v, but once P has reached v, then we have constructed a circuit and we are done. Next, observe that if there are two cycles C, C' sharing a common vertex, then the concatenation CC' is also a circuit. Finally, we conclude our argument by observing that the removal of some circuit, C from G, does not change the parity of the degrees of G, and moreover for any vertex of C with edges in $G - C$ must be part of a circuit in within a connected component. Because G is connected this construction may be expanded until all edges of G have been exhausted, thus G must contain an Eulerian circuit. Algorithmically this yields the following:

```
EULERIAN-CIRCUIT(v)
1  R ← []
2  while HAS-EDGE(v)
3        do
4           e ← GET-EDGE(v)
5           REMOVE-EDGE(e)
6           R ← EULERIAN-CIRCUIT(NEXT(e, v)) + e + R
7  return R
```

The running time of EULERIAN-CIRCUIT is dominated by the while-loop, which is executed only while v has an edge. Because each iteration removes one edge, the total running time is bounded by $O(|E|)$.