Institut for Matematik og Datalogi | September 19, 2012
Syddansk Universitet | JBJ

DM208 – Fall 2012 – Weekly Note 5

Handout material in Week 38
Korte and Vygen, Combinatorial Optimization 3rd edition Springer 2008, Chapter 13.

Stuff covered in week 38

- Non-bipartite Matching. PS Chapter 10.4-10.5 and SCH 5.1-5.2 (We will not cover weighted non-bipartite matching. Still you should know that this problem is solvable in polynomial time).
- We also (via the "guest lecture of Matthias Kriesell) covered the more general f-factors, where we have a graph $G = (V, E)$ and a specification $f(v) \leq d(v)$ at every vertex and we want to select a subset E' of E so that these induce a spanning graph $F = (V, E')$ with $d_F(v) = f(v)$ for each $v \in V$. Such an F is called an f-factor of G. In particular, when $f(v) \equiv k$ for all $v \in V$ we call F a k-factor of G.

  To see that this problem can be solved using a matching algorithm, lets create a new graph H from $G = (V, E)$ and f as follows: Replace each vertex v of G with two sets $A(v)$ and $B(v)$ of vertices with $|A(v)| = d(v)$ and $|B(v)| = d(v) - f(v)$. The edges of H are all edges between $A(v), B(v)$ for all $v \in V$ and for each edge $uw \in E$, put a single edge between $A(u)$ and $A(w)$ such that each vertex of $A(v)$, $v \in V$ belongs to exactly one such edge. Now it is easy to show that H has a perfect matching if and only if G has an f-factor.
  This is a polynomial reduction (remind yourself why!) so we get a polynomial algorithm for checking the existence of an f-factor in a given graph from the polynomial algorithm for the maximum matching problem.

Lecture September 25, 2012:
NB, note that we will be in U26!

- Arc-disjoint branchings. This is BJG Section 9.5.
- Minimum cost branchings. This is based on pages 338-341 in the second edition of BJG. These pages are available from the course page. **Note that in the contracted graph we work with the weight function** c'. This is not written explicitly in the section (although it is clear from the proof).
- If there is more time we will start on the exercises.

Lecture September 26, 2012:
I plan to spend at least 3 of the 4 "hours" on the lecture(s).

- Matroid intersection. This is based on PS section 12.5, SCH 10.4-10.5 and Korte and Vygen sections 13.5-13.7
- Matroid Union (partition) This is based on PS section 12.5, SCH 10.4-10.5 and Korte and Vygen sections 13.5-13.7

Problems and applications to discuss on September 25,2012:

- Prove that if a graph is 2-connected (that is, there are at least two internally disjoint (s,t)-paths for every choice of distinct vertices $s,t \in V(G)$), then for every vertex s and edge uv of G there is a cycle C which contains s and the edge uv.
- Show that a graph G has a strongly connected orientation (we replace each edge uv by one of the arcs $u \to v, v \to u$) if and only if G is 2-edge-connected. Also describe an algorithm to find such an orientation or a bad cut.
- Let G be a tree. How many new edges must be added to G to make it 2-edge-connected? Try to construct an algorithm which adds as few edges as possible and try to formulate a min-max result.
- Let $G = (V,E)$ be a k-edge-connected graph and let H be a minimal set of new edges such that $G' = (V, E \cup H)$ is $(k+1)$-edge-connected. Prove that the edges of H form a forest.
- Prove that every minimally k-edge-connected graph has at most $k(n-1)$ edges. Hint: recall what you learned about max-back forrests.
- Show that in the case when G is a bipartite graph we can solve the f-factor problem by transforming the problem into a maximum flow problem.
- SCH 5.1, 5.4
- SCH 5.7 page 84.
- 2-processor scheduling: Suppose we are given a task consisting of 8 jobs a,b,c,d,e,f,g,h with the following precedence relations, where we only list the one that do not follow by transitivity (if x is before y and y before z, then automatically x is before z so we don't write it in the list):

  $$\{a < c, a < f, b < d, c < e, c < g, d < f, f < e, f < g, f < h\}$$

  Find an optimal schedule for processing on 2 processors and prove that it is optimal.