Second obligatory project in DM85 Spring 2007

Jørgen Bang-Jensen

The project period starts on Monday April 23 and the report must be handed in at the latest Monday May 21 at noon in JBJ's mailbox.

The report may be written in Danish or in English. You may work in groups of up to 3 persons on a project and it does not have to be the same groups as for project 1.

The project is a continuation of Project 1 and all definitions and results from that project is assumed to be known.

Problem 1 (10 points)

The constraints in your mathematical model for E2AUG express that no cut in the network with capacity less than two (where "capacity" is measured by the sum over x-values on edges plus the number of edges from H across the cut) may exist.

In order to apply Branch & Cut, you need a *separation routine*, i.e. an algorithm, which given a potential solution returns a violated inequality if one exists. Explain how a maximum flow algorithm can be used as separation routine and estimate the complexity of this approach. Make sure to try and reduce the number of maximum flow calculations as much as possible and use the most appropriate flow algorithm. Also explain how you handle the problem that the graph is undirected.

Problem 2 (25 points)

Consider the following variants of E2AUG and E1-2AUG called E2AUG(S) and E1-2AUG(S). Besides G and H we are given a subset $S \subseteq V$ of V and we are only interested in having at least two edges across every cut that separates two vertices from S.

(a) Describe mathematical models for E2AUG(S) and E1-2AUG(S).

(b) Consider the special case of E1-2AUG(S) when $S = \{x, y\}$ and H is a hamiltonian path from x to y. Show by an example that the heuristic greedy cover from Project 1 may not find an optimal solution to E1-2AUG(S), even in this very special case. Hint: you can make such an example where all edges in E' have weight 1.

(c) Now consider the following idea to solve E1-2AUG(S) when S and H are as above. Make a directed graph D by orienting H as a directed path P from y to x and orienting[1] every edge of E' so that it forms a directed cycle with the part of P which it covers. For an example see Figure 1.The weight of the arcs not in P is the same as the weight of the corresponding edge in E'. Argue that the length of a shortest (x, y)-path in D equals the value of an optimal solution to E1-2AUG(S).

[1] We may also replace each edge xy of E' by a directed 2-cycle xyx, but when H is a path we don't need both directions and it is simpler to argue when we use only the proposed direction on the arcs

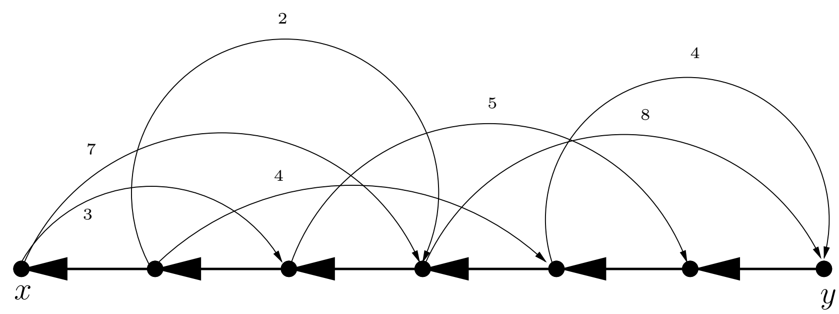

Figure 1: The digraph D constructed by orienting H as a directed path from y to x and all edges of E' in the "opposite" direction. Weights on arcs not on P are inherited from E'. All arcs of P have weight 0.

(d) Demonstrate the algorithm on the example in Figure 1 and show the resulting solution to E1-1AUG(S).

(e) Suppose now that $S = \{x, y\}$ still holds but now H is a general spanning tree of G and not necessarily a path. Argue that we can still solve E1-2AUG(S) optimally, using just one shortest path calculation in a suitably defined directed graph which we obtain from G by orienting some edges only one way and others in both directions. Hint: again we are just interested in covering the xy path in H but now we may have to use edges from E' that connect other parts of H. Use a suitable orientation on the edges of the xy path in H so as to make sure that if your (x, y)-path uses one of these edges, then this edge is covered by at least one of the new edges you will add.

Problem 3 (20 points)

In this problem we consider E2AUG where H will always be the spanning graph with no edges, i.e. $H = (V, \emptyset)$. Thus we are looking for a minimum cost spanning 2-edge-connected subgraph of G. In Project 1 you found that when G is the Petersen graph and H has no edges, the optimum value of the linear programming relaxation of your E2AUG model is 10 while the optimum solution to E2AUG has value 11. Hence the cut inequalities (demanding at least 2 edges across every cut) are not sufficient to ensure an integer optimum solution (all variables integer). In particular the following two fractional solutions y and y' are both optimal:

- y has value $\frac{1}{2}$ on every arc of the two 5-cycles 123451 and 6810796 (the vertex numbering is as in Figure 1 of the first project) and value 1 on the five remaining edges
- y' has value $\frac{2}{3}$ on every edge.

The purpose of this problem is to derive valid cuts for E2AUG which cut off both of these LP solutions when we add the new cuts to the original E2AUG model.

(a) Derive a valid in-equality which says that every integer solution x to E2AUG satisfies that the sum of the x-values around the 5-cycle 123451 (and hence by symmetry a lot of other 5 cycles such as 6810796 and 169721) is at least 3. Hint: take suitable combinations of the degree in-equalities that say that the sum over x on the 3 edges incident to any vertex is at least 2 (expressed as e.g. $-x_{12} - x_{15} - x_{16} \leq -2$) and the in-equalities that say $x \leq 1$ for every edge.

(b) Use the valid in-equalities (for the 5-cycles) which you derived in (a) and the degree in-equalities to obtain a pair of in-equalities of the form either $a \cdot x \geq b$ or $a' \cdot x \geq b'$ where a, a' are coefficients for the edges and b, b' are constants. Your new in-equalities should be able to cut off the LP-solution y' and you must show why they do so. Hint: consider what happens if you have equality in the new in-equality from (a).

(c) Are these new in-equalities (and all the equivalent ones that follow by symmetry) enough to force an integer optimum to the augmented version of the model for the Petersen graph (with all in-equalities these added)?

Problem 4 (25 points)

On the course page you can find an example of an OPL script for solving the minimum spanning tree problem using a cutting plane approach. On weekly note 8 you were asked to improve this script by considering the orientation version, using that if T is a spanning tree with vertex set $\{1, 2, 3, \ldots, n\}$, then you can always orient the edges of T such that the tree becomes directed out of 1 (we call this an *out-tree* rooted at vertex 1) and has the property that every vertex except 1 has in-degree (number of arcs entering it) precisely one. This is equivalent to saying that the oriented version of T contains a directed path from 1 to every other vertex. If you managed to make such a script you will have seen that it is much faster than the naive version I first gave you. Now you should use the same approach to construct an OPL script for solving the E1-2AUG problem. Here the main observations needed are the following which you are NOT asked to prove:

1. A graph is 2-edge-connected if and only if it can be oriented as a strong digraph.

2. A digraph D is strongly connected if and only if D contains a spanning out-tree rooted at 1 and a spanning in-tree rooted at 1 (in an in-tree every vertex has a directed path towards the root) for every vertex $v \in V(D)$.

So in order to solve E1-2AUG you need to find a cheapest possible set of edges X from E' to include so that adding these to H gives a graph $G' = (V, F + X)$ which has an orientation containing both an in-tree and an out-tree (not necessarily arc-disjoint!!) rooted at vertex 1.

(a) Describe a mathematical model for the orientation version of E1-2AUG. Thus you need a variable y to describe which edges from E' should be taken and another variable x to describe which orientation (each edge ij with $y_{ij} = 1$ should be given precisely one orientation) to give the chosen edges AND (a subset of) the edges of

H so that the set of arcs for which $x = 1$ form a strong digraph containing every vertex of G. Since edges in H have cost 0 you may choose to force all these to be oriented in your model.

(b) Explain how to use this model in a cutting plane approach. I.e. what is the model you solve repeatedly and which are the cuts that you add in each iteration.

(c) Use your script to solve the problem on the instance from Figure 1 in Project 1.

(d) Use your script to solve the three instances "G1.dat,G2.dat,G3.dat" supplied on the home page of the course (with a description of the data format) I need the value of the solution, which edges are chosen and the solution time.

In order to solve the problems supplied by me you will probably need the full version of OPL studio (which you call by the command 'oplst37'). Since we have only one license available, you must quit the program as soon as you have made your test so that others have a chance to use it!

Problem 5 (20 points)

Your boss suddenly runs into your room and cries: "Drop everything - I have an urgent job for you. Find the best assignment of persons to jobs in the following situation:

I have 5 persons and 5 jobs, a profit matrix N showing in entry (i, j) the profit earned from letting person i do job j, and a cost matrix O showing the cost of educating person i for job j (the matrices are shown below). The educational budget is 18, and exactly one person has to be assigned to each job." **Note that we do not care what the education price is, as long as it does not exceed 18. Thus this cost should not be included in the objective function.**

$$N = \begin{pmatrix} 5 & 11 & 7 & 6 & 5 \\ 2 & 10 & 4 & 10 & 3 \\ 6 & 13 & 8 & 9 & 9 \\ 6 & 13 & 8 & 12 & 8 \\ 6 & 13 & 9 & 14 & 8 \end{pmatrix}$$

$$O = \begin{pmatrix} 4 & 5 & 6 & 5 & 3 \\ 5 & 8 & 24 & 3 & 6 \\ 4 & 10 & 8 & 3 & 5 \\ 2 & 9 & 2 & 10 & 8 \\ 2 & 5 & 3 & 1 & 4 \end{pmatrix}$$

(a) First formulate and solve the problem using OPL studio.

(b) To get some training in using optimization techniques (perhaps to convince you boss that you can do so competently) you decide to solve the problem to optimality using Branch-and-Bound. Here you are not allowed to use the information you got by solving the problem using OPL studio. That is, you must demonstrate that your solution is optimal by other means and you cannot use the OPL solution as the incumbent. You may, however, use OPL studio to solve the assignment problems that you generate during your algorithm.

You first convert the problem to a cost minimization problem, which you then solve. The lower bound computation should be done using Lagrangian relaxation as described in the next section, i.e. you have to use the objective value of the assignment problem resulting from the relaxation of the budget constraint with a non-negative multiplier.

(c) In order to explain the method to your boss, you have to explain why any non-negative multiplier gives rise to a lower bound for your given problem.

Suggested line of work for Problem 5:

Converting the profit maximization problem to a cost minimization problem with non-negative cost is done by multiplying all values in the profit matrix by -1 and adding the largest element of the original profit matrix to each element. Formulation of the Lagrangian relaxation can be done by subtracting the right-hand side of the budget constraint from the left-hand side, multiplying the difference with λ and adding the term to the objective function. Collecting terms to get the objective function coefficient for each x_{ij}, you get an assignment problem for each fixed value of λ. Choose a value of λ, e.g. 1, and solve the assignment problem. If the budget constraint is not fulfilled, then try to increase λ and resolve. If an optimal solution cannot be found through adjustments of λ (what are the conditions, under which an optimal solution to the relaxed problem is also optimal for the original problem?), choose a variable to branch on and create two new subproblems, each of which is treated as the original problem.