Code Jam 2009 - World Finals

Analysis: Wi-fi Towers

Connection graph

We start by representing the problem as a graph problem. Each tower is a vertex in the graph and has a weight equal to its score. If a tower A has another tower B in its range, we represent this fact as a directed edge from A to B. The problem is to choose a set of vertices with maximum weight such that for every edge from A to B, if A is chosen then B is also chosen. In the following, we assume the number of towers (vertices) is **V** and the number of edges is **E**.

Reduce to an instance of MIN-CUT

To reduce the problem to an instance of MIN-CUT, we create a flow network as follows. Create a source vertex, a sink vertex, and one vertex for each tower. Suppose a tower has score **s**. If **s** > 0, create an edge from the vertex to the sink with capacity **s**. If **s** < 0, create an edge from the source to this vertex with capacity |**s**|. Finally, for every edge in the connection graph, create a similar edge in the flow network with infinite capacity. The network has **V** + 2 = O(**V**) vertices and O(**V** + **E**) edges.

Now every finite cut in the graph represents a choice of towers - we choose every tower on the same side of the cut as the source. The infinite capacity edges enforce that the choice follows the given constraints (otherwise we get a cut of infinite weight). The edges from the source and to the sink penalize the solution appropriately for choosing towers with negative scores and for not choosing towers with positive scores. If the value of the best cut is **C**, the answer is **S** - **C**, where **S** is the sum of positive tower scores.

Solving MIN-CUT

By the max-flow min-cut theorem, we can solve the MIN-CUT instance generated above by computing the maximum flow in the same graph. To compute the maximum flow, we can use the Edmonds-Karp algorithm (a variant of Ford-Fulkerson which selects augmenting paths using BFS), which results in complexity bounded by $O(\mathbf{V}(\mathbf{V}+\mathbf{E})^2) = O(\mathbf{V}^5)$. In practice, this was fast enough to solve all possible test cases.

One could also use a more complicated push-relabel max-flow algorithm which, with a FIFO vertex selection rule, results in complexity $O(\mathbf{V}^3)$.

Yet another algorithm, thanks to integral capacities, is the capacity scaling variant of Ford-Fulkerson (start by searching for augmenting paths with weights being large powers of 2 first, and then decrease). This results in $O((\mathbf{V}+\mathbf{E})^2 \log \mathbf{F}) = O(\mathbf{V}^4 \log \mathbf{F})$ where **F** is the maximum value of the flow.

Reducing the number of edges

Finally, there is a geometric trick using which we can reduce the number of edges **E** to O(**V**), thus reducing the complexity of the algorithms above. The complexity of Edmonds-Karp becomes $O(\mathbf{V}^3)$ and of capacity scaling: $O(\mathbf{V}^2 \log \mathbf{F})$.

First, notice that we can remove edges without changing the final answer as long as the transitive closure of the graph stays the same.

The crucial trick is to see that if there are two directed edges A-C and B-C, and the angle ACB is smaller than 60 degrees, then we can remove the longer edge. Suppose A-C is longer than B-C. Then if we remove the edge A-C, there is still going to be an indirect connection A-B-C (using shorter or equal length edges), thanks to the fact that the range of A is a circle.

If we keep doing this, every vertex will end up with at most 6 incoming edges, thus reducing the total number of edges to at most 6**V**.

More information

This problem is also equivalent to the Minimum Closure Problem, which was studied in the 1970s and has applications in the mining industry.

Max Flow Min Cut Theorem