CS4234: Optimization Algorithms

Week 7 Questions: answered!

Week 7 Questions

Here I try to give some answers to the questions that were asked in the IVLE survey. This week, there were only a couple questions.

Question 1 *In the flow decomposition lemma, when a flow F is split into flows f_1 to f_k, is each f_j a single path from s to t?*

Yes. The idea of the flow decomposition lemma is that the initialy flow F is split into flows $f_1, \ldots, f_k$ where each f_j is a st-path. Also important, the sum $\sum |f_j| = |F|$, the total flow. Finally, there are $\leq m$ such flows, i.e., $k \leq m$.

As we talked about in tutorial, there is also an efficient algorithm that finds the flow decomposition, and if the capacities are all integral and if F is integral, then the resulting decomposed flows are also integral.

The basic idea of the algorithm is to start with a flow F and de-augment: find any st-path with positive flow; create a new f_j from that path, and subtract that from the flow. You can repeat that until all the flow is gone. (There may be left circulations, i.e., flow going in circles in the network. But these circulations do not matter as they do not change the overall st-flow.)

Question 2 *For the maximum flow problem, what if the capacity is not on the vertex but on the nodes?*

As we saw in tutorial, we can often change vertex limitations to edge limitations by replacing every node u with two nodes u_{in} and u_{out}: all incoming edges to u attach to u_{in}, all outgoing edges from u leave from u_{out} and there is one directed edge from u_{in} to u_{out}. The capacity of the edge (u_{in}, u_{out}) is set equal to the vertex capacity.

Question 3 *How do we reduce the k-Clique problem to $(n-k)$-vertex-cover? For a general graph, is a relation between minimum vertex cover and maximum matching?*

See the notes for more details. The general idea in the reduction is to observe:

- Given a graph G, define a new graph G' that is the inverse, i.e., there is an edge $(u, v) \in G'$ if an only if there is *no* edges $(u, v) \in G$.
- Every clique in G is an independent set in G'. Every independent set in G' is a clique in G'.
- There is an independent set of size k in G if and only if there is a vertex cover of size $n - k$ in G. Similarly, there is an independent set of size k in G' if and only if there is a vertex cover of size $n-k$ in G'. Why? If a set of nodes C is a vertex cover, then the set $V \setminus C$ is independent—any edge between nodes in $V \setminus C$ would not be covered by a node in C. Similarly, if a set of nodes I is an independent set, then the set $V \setminus I$ is a vertex cover, because every edge is adjacent to some node not in I.
- Putting these facts together, we conclude that there is a clique in G of size k if and only if there is an independent set in G' of size k if and only if there is a vertex cover of size $n-k$ in G'. Similarly, there is a vertex cover in G of size k if and only if there is an indepedent set in G of size $n-k$ if and only if there is a clique of size $n-k$ in G'.

In general, we know that the size of the minimum vertex cover is always $\geq$ the size of the maximum matching, for all graphs. That is always true.

Unfortunately, the converse is not true: the size of the minimum vertex cover is not always equal to the size of the maximum matching. In fact, we know that they can differ by up to a factor of 2—which is the integrality gap of the linear program for vertex cover.

When we look at duality, we will try to see why this is the case: vertex cover and matching are almost duals (and they are duals in bipartite graphs), but not quite, not in general.

Question 4 *How did people come up with the push relabel algorithm? Is this algorithm related to linear programming? Can we represent the max flow problem as a linear programming problem?*

Good question! I'm always curious how a given algorithm was invented.

The basic idea of trying to "push flow around" until it converges is, I think, a reasonably natural approach. The breakthrough idea, I think, was to find a way to ensure that it converged, i.e., that eventually it reached the target.

The idea of using heights to avoid cycles is actually a common one. There are a variety of routing problems where it is useful to assign a unique height to every node in a graph, which yields a directed acyclic graph. We can then use all our favorite DAG algorithms.

I would guess that the inventors of Push-Relabel started there: assign heights to ensure it is acyclic. Then they experimented with different relabeling strategies until they found one that worked. Then they simplified, until they observed that basically any relabeling scheme works! At least that's my best guess.

We can use linear programming to represent maximum flow problems. It is, at first glance, an integer linear program but we can show that it will always find integral solutions. We saw one example of this in tutorial, where we looked at the minimum cost flow problem.

We will come back to this when talking about duality (and the relationship of max flow and min cut in duality theory).