Finding Disjoint Paths

We can determine if there is a path of G from one vertex s to another t using depth-first or breadth-first search. Often, we actually want to find a large number of such paths which are internally disjoint. Here, we present an algorithm which finds a set of internally disjoint s-t paths of maximum cardinality.

There are actually four versions of this problem, as the graph can be directed or undirected and we can either insist that the paths are vertex disjoint, or allow them to share vertices and insist only that they are edge disjoint, Thus we consider:

UE paths
Input: A graph G and two specified vertices s and t
Ouput: A maximum cardinality set $P_1, ..., P_k$ of edge disjoint paths from s to t.

UV paths
Input: A graph G and two specified vertices s and t
Ouput: A maximum cardinality set $P_1, ..., P_k$ of internally vertex disjoint paths from s to t.

DE paths
Input: A digraph (= directed graph) D and two specified vertices s and t
Ouput: A maximum cardinality set $P_1, ..., P_k$ of edge disjoint paths from s to t.

DV paths
Input: A digraph D and two specified vertices s and t
Ouput: A maximum cardinality set $P_1, ..., P_k$ of internally vertex disjoint paths from s to t.

We now discuss an efficient algorithm which solves all of these problems.

The Reductions

To begin, we note that if we want to solve UV Paths on input (G, s, t) then it is enough to solve DV Paths on (D, s, t) where D is obtained from G by replacing each edge uv by the two arcs uv and vu. For, a set of k internally vertex disjoint paths in G clearly yields k internally vertex disjoint paths in D, and vice-versa.

So, to solve an instance of UV paths, we can first reduce it to an instance of DV paths, find the solution and then transform it to a solution for the original UV Paths instance. Since both reductions can be carried out in linear time (an exercise), if we have a polynomial time algorithm for DV paths, then we also have a polynomial time algorithm for UV paths.

Next we note that to solve DV paths on input (D, s, t) it is enough to solve DE Paths on input (D', s, t) where D' is an auxiliary directed graph defined as follows:

(i) $V(D') = \{v_{in} | v \in V(D)\} \cup \{v_{out} | v \in V(D)\}$

(ii) $E(D') = \{v_{in}v_{out} | v \in V(D)\} \cup \{u_{out}v_{in} | uv \in E(D)\}$

Finally, we show that if we want to solve UE Paths on input (G, s, t) then it is enough to solve DE Paths on (D, s, t) where D is obtained from G by replacing each edge uv by the two arcs uv and vu. Clearly, there is a 1-1 correspondence between the directed s-t paths in D and the s-t paths in G. Furthermore, k edge disjoint paths in G yields k edge disjoint paths in D, however, the reverse implication is not so clear, as we may have two disjoint directed s-t paths in D which do not correspond to disjoint paths of G. In particular, one of the paths may use an arc arc uv while the other using the arc vu,

However, it is clear that if we have k edge disjoint paths of D which also satisfy:

(*) If uv is used on some path then vu is used on no path.

Then they correspond to a set of k disjoint directed paths in G.

We show that if G contains k edge disjoint s-t paths then it contains k edge disjoint paths which satisfy (*). To do so, we actually prove the result for walks and then noted that we could shortcut walks to obtain paths.

That is we proved

(1) If D contains k edge disjoint paths then it contains k edge disjoint s-t walks (since a path is a walk),

(2) If D contains k edge disjoint walks then it contains k edge disjoint walks such that if uv is used on some walk then vu is used on no walk (proof discussed below).

(3) If D contains k edge disjoint walks such that if uv is used on some walks then vu is used on no walk, then it contains k edge disjoint s-t paths satisfying (*) (since the edge set of each walk contains an s-t path).

Combining (1)-(3) with our remarks above we see that G contains k edge disjoint s-t paths if and only if D does.

In proving (2) we consider a set of k edge disjoint s-t walks, chosen so as to minimize the sum of the edges of all the paths. Then each walk is a path as otherwise we could shortcut one of the walks, and get a new sets of walks whose total number of edges is smaller. We then suppose for a contradiction that there are two paths W_1 and W_2 in our set with $uv \in W_1$ and $vu \in W_2$. We write $W_1 = W_1'W_1''$ and $W_2 = W_2'W_2''$ where W_1' ends at u, W_2' ends at v, W_1'' begins at v, and W_2'' begins at u. We note that we obtain an s-t walk W_3 by concatenating W_1' and W_2'' and another s-t walk W_4 by concatenating W_2'' and W_1'. Furthermore, they are edge disjoint because (i) no edge is used twice on W_1 or W_2 since they are paths and (ii) no edge is used in both W_1 and W_2

because they are edge disjoint. But this clearly contradicts our choice of a set of walks, as we could have used W_3 and W_4 instead of W_1, W_2. This proves (2).

These reduction show that an algorithm to solve DE Paths can be used to solve UV Paths, UE Paths, and DV paths.

The Algorithm

Consider an instance (D, s, t) of DE paths. Given a set $\{P_1, ..., P_k\}$ of s-t walks in D, we define an auxiliary graph D' as follows:

$$V(D') = V(D)$$
$$E(D') = E(D) - (\cup_{i=1}^k E(P_i)) \cup \{uv | vu \in \cup_{i=1}^k E(P_i)\}.$$

We used breadth-first or depth first search to look for an s-t path in D'. If we found such a path P', then we let F be the digraph with edge set $(\cup_{i=1}^k E(P_i)) \cup E(P')$. By essentially repeatedly applying the reduction above we can obtain $k+1$ vertex disjoint s-t paths in F which use no edge uv of F such that vu is also in F (we are more precise about how we do this below). Thus, our new set of paths uses only edges of D (since for any edge uv of $F - D$, we know vu is on some P_i and hence in F).

If we find no such path then we can find a subset S of V with $s \in S, t \notin S$ such that there is no edge uv of D' from S to $V - S$. This implies that

(i) There is no edge of $E(D) - (\cup_{i=1}^k E(P_i))$ from S to $V - S$,

(ii) There is no edge of any P_i from $V - S$ to S and hence exactly one edge of each P_i from S to $V - S$.

Thus, we see that there are exactly k edges of D from S to $V - S$ and since each s to t path uses such an edge we see that there do not exist $k+1$ vertex disjoint paths from s to t in D. Thus, our current set of k paths is optimal.

By the above, we have an iterative procedure that finds a maximum set of edge disjoint paths s-t in a digraph D. The main step in each iteration is a breadth first search, so each iteration takes $O(|E|)$ time. There are at most $|V|$ edges out of s, and hence at most $|V|$ iterations so the total time complexity is $O(|V||E|)$. If we only want to find k paths, then the algorithm takes $O(k|E|)$ time.

We have also obtained an algorithmic proof of the following result.

Definition: An s-t cut is a set of edges whose removal leaves a graph without a path from s to t.

Menger's Theorem: The maximum number of edge disjoint paths of D is equal to the minimum number of edges in an s-t cut.