Matchings

Design and Analysis of Algorithms
Andrei Bulatov

Matchings

A matching M of a graph G = (V,E) is a set of edges such that every vertex is incident to at most one edge from M

Bipartite graphs: bipartition X, Y

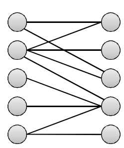
The Bipartite Matching Problem

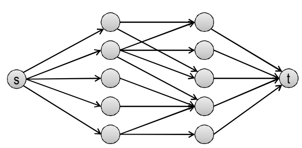
Instance:
A bipartite graph G

Objective:
Find a matching in G of maximal size

Algorithm

We show how to reduce the Bipartite Matching problem to Network Flow
Let G be a bipartite graph with bipartition X, Y

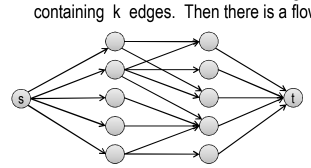




- orient all edges from X to Y
- add source s and sink t
- add arcs from s to all nodes in X, and from all nodes in Y to t
- set the weight of all arcs to be 1

Analysis

Lemma
Suppose there is a matching of G $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ containing k edges. Then there is a flow in G' of value k



Proof
Straightforward

Analysis (cntd)

Lemma
Suppose there is a flow in G' of value k, then there is a matching of G containing k edges.

Proof
Let f be a flow in G' of value k.
Since all capacities in G' are integer, there is an integer flow of value at least k. So we can assume f is integer.
f(e) equals 0 or 1 for every edge e
Let M be the set of arcs with the flow value 1

Analysis (cntd)

M contains k edges
Indeed, consider the cut (A,B) with A = X ∪ {s}
The value of the flow through the cut equals the number of arcs from X to Y where the flow is non-zero
The set of such arcs is exactly the set M

Every node from X is the beginning of at most one arc from M
It follows straightforwardly from the conservation property

Every node from Y is the end of at most one arc from M
Same argument
Therefore M is a matching

Running Time

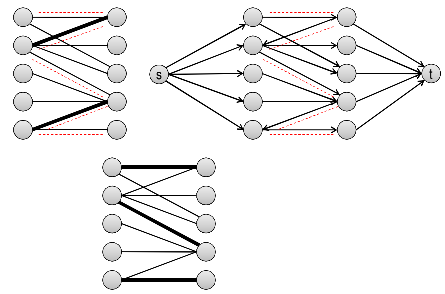
Theorem
The Ford-Falkerson algorithm can be used to find a maximal matching in a bipartite graph in O(mn) time

Proof
We can assume that G has no isolated vertices, and so $m \geq n/2$
The maximal value of a flow in G' does not exceed $C = c(s) = |X| \leq n$
By the theorem on the running time of the F.-F. algorithm, it runs in $O(mC) = O(mn)$ time

QED

Augmenting Paths in Bipartite Graphs

There is another algorithm for Bipartite Matching. It finds alternating paths

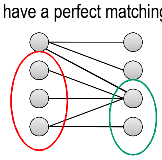


Perfect Matching and Hall's Theorem

If both parts of a bipartite graph have the same number of elements , a perfect matching can exist, that is a matching that includes all vertices of the graph

How is it possible that a bipartite graph does not have a perfect matching

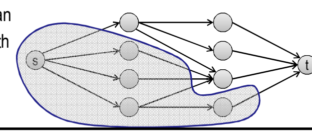
If there is $A \subseteq X$ such that for the set of neighbors N(A)
$|N(A)| < |A|$
(or same for Y)



Theorem (Hall)
If G is a bipartite graph, and for any $A \subseteq X$ and any $B \subseteq Y$, we have $|A| \leq |N(A)|$, $|B| \leq |N(B)|$, then there is a perfect matching of G.

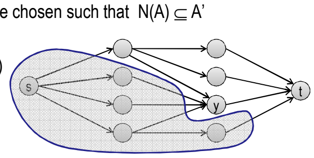
Perfect Matching and Hall's Theorem (cntd)

Proof
We use graph G'. Assume $|X| = |Y| = n$
If there is no perfect matching of G, a maximal flow in G' has value less than n
We use this fact to find a set A (a subset of X or Y) such that $|N(A)| < |A|$
Since the value of maximal flow equals the capacity of a minimal cut, there is a cut (A', B') with capacity $< n$
Set A' contains s, but can contain vertices from both sides
Set $A = X \cap A'$



Perfect Matching and Hall's Theorem (cntd)

We show that (A', B') can be chosen such that $N(A) \subseteq A'$
Take a node $y \in B' \cap N(A)$
Prove that $(A' \cup \{y\}, B - \{y\})$ is a cut of capacity not exceeding that of (A',B')



Indeed, the new cut crosses the arc (y,t),
but since $y \in N(A)$, there is at least one arc arriving to y from A, and so now it is not crossed
Consider the capacity of (A',B') assuming $N(A) \subseteq A'$
The only arcs out of A' are those leaving s, or arriving to t

Perfect Matching and Hall's Theorem (cntd)

Thus
$$c(A',B') = |X \cap B'| + |Y \cap A'|.$$
Observe that $|X \cap B'| = n - |A|$, and $|Y \cap A'| \geq |N(A)|$
Then the assumption $c(A',B') < n$ implies
$$n - |A| + |N(A)| \leq |X \cap B'| + |Y \cap A'| = c(A',B') < n$$
We get
$$|A| > |N(A)|$$

QED

Disjoint Paths

Design and Analysis of Algorithms
Andrei Bulatov

Disjoint Paths Problem

A set of paths are said to be disjoint if they do not have common edges

The Directed Edge-Disjoint Paths Problem

Instance:
A digraph G, and distinguished vertices s, t of G

Objective:
Find a maximum number of edge-disjoint paths from s to t

The Undirected Edge-Disjoint Paths Problem

The same only for undirected graphs

Directed Paths vs. Flows

Let G be a digraph, s, t distinguished nodes

We can always assume that s is a source, and t is a sink

Why?

Define a flow network by making s and t the distinguished source and sink, resp., and setting the capacity of each arc to be 1

Lemma

If there are k edge-disjoint paths in a directed graph G from s to t, then the value of the maximum flow in G is at least k

Proof

Set f(e) = 1 if e belongs to one of the paths, and f(e) = 0 otherwise

QED

Directed Paths vs. Flows (cntd)

We can choose an integer maximal flow. Its values are 0 and 1

Lemma

If f is a flow with values 0 and 1 of value k, then the set of edges with flow value f(e) = 1 contains a set of k edge-disjoint paths.

Proof

We proceed by induction on k

Base Case: If k = 0 then there is nothing to prove.

Induction Hypothesis: Suppose the claim is true for all flows of value < k

Induction Step:

Construct a sequence of arcs as follows:

start with s.

Directed Paths vs. Flows (cntd)

Take any edge e = (s,u) such that f(e) = 1

By Conservation property, there is an edge e’ = (u,w) with f(e’) = 1

Continue until

either we reach t , and so obtain a path P from s to t

or we reach some node v for the second time

In the first case set f(e) = 0 for all arcs e from P

We obtain a flow of value k – 1 (why?), and get the result by the Induction Hypothesis.

In the second case, we remove the cycle between the two appearances of v

QED

Finding Disjoint Directed Paths

Algorithm:

- apply the F.-F. algorithm
- use the inductive procedure from the proof (it is called path decomposition)

Theorem

The Ford-Falkerson algorithm can be used to find a maximal set of edge-disjoint paths in a digraph in O(mn) time

Undirected Paths vs. Flows

Let G be a an undirected graph, s, t distinguished vertices

Finding paths in G can be reduced to finding paths in a directed graph as follows:

Replace every edge of G with 2 arcs going into opposite directions

Remove arcs coming into s, and going out of t

Problem:

Paths in the digraph can use the arcs going opposite directions.

Lemma

For any flow network, there is a maximum flow f where for all opposite directed arcs e = (u,v) and e' = (v,u), either f(e) = 0, or f(e') = 0

Undirected Paths vs. Flows (cntd)

Proof

Take any (integer) maximal flow f such that $f(e) \neq 0$ and $f(e') \neq 0$ for some e = (u,v), e' =(v,u)

Let k be the smallest of these two values

Decreasing f(e) and f(e') by k, we obtain a flow that is 0 on one of these two opposite arcs.

QED

Theorem

The Ford-Falkerson algorithm can be used to find a maximal set of edge-disjoint paths in an undirected graph in O(mn) time