09. Bipartite Matching

CPSC 535

Kevin A. Wortman

CALIFORNIA STATE UNIVERSITY FULLERTON

Bipartite Matching

So far, all our reductions to max-flow have been either straightforward flow simulations, or variations on max-flow.

Now we'll see a quite-different problem that also reduces to max-flow.

Partition of a Set

Intuitively: if $X = L \cup R$ is a *partition*, then every element of X is placed in L or R (but not both).

Formally: L and R partition X if

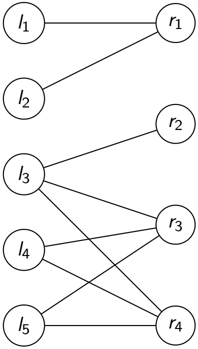
- $X = L \cup R$,
- $L \cap R = \emptyset$, and
- $L \neq \emptyset, R \neq \emptyset$.

Bipartite Matching

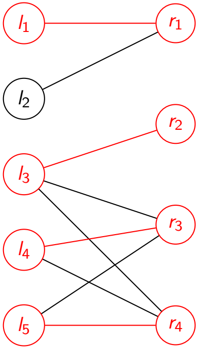
bipartite maximum matching
input: an undirected bipartite graph $G = (V, E)$ with parts $V = L \cup R$
output: a matching $M \subseteq E$ where the number of matched vertices is maximum

- *bipartite:* L, R are disjoint and edges only go between L, R
- *matching:* pick edges that "pair off" two vertices; goal is to maximize #paired-off
- intuitively, L is one kind of thing and R is another kind of thing

Bipartite Matching



Bipartite Matching



matching

$$M = \{\text{included edges}\} = \{\{l_1, r_1\}, \{l_3, r_3\}, \{l_4, r_3\}, \{l_5, r_4\}\}$$

$$|M| = 4$$

(other optimal matchings exist)

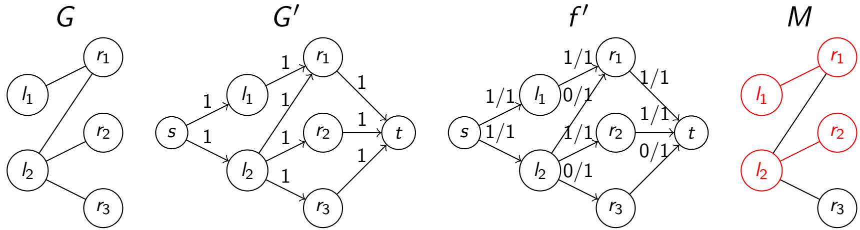
Bipartite Matching Applications

- any scenario where there are two kinds of things that can be paired
- goal is simply maximum number of pairings
- casting for a play: $L =$ set of actors; $R =$ set of roles; edge $\{l, r\}$ exists when l could play role r
- packing leftover food (one item/container): $L =$ set of food items; $R =$ available containers; edge $\{l, r\}$ exists when food l could fit in container r
- scheduling appointments: $L =$ set of clients; $R =$ set of time slots; edge $\{l, r\}$ exists when client l could meet appointment r
- might feel NP-hard, but actually in P

Formulating Bipartite Matching as Flow

- let $G = (V, E)$ be bipartite matching instance
- create $G' = (V', E')$ with $V' = V \cup \{s, t\}$ where s, t are new source/sink
- create edges in G':
  - (l, r) $\forall l \in L, r \in R, \{l, r\} \in E$
  - (s, l) $\forall l \in L$
  - (r, t) $\forall r \in R$
- every edge (v, w) has capacity $c(v, w) = 1$
- post-processing: edge $(l, r) \in M$ iff $f(l, r) = 1$
- observe $|V'| \in O(|V|), |E'| \in O(|E|)$, overhead is $O(|V| + |E|)$
- $\implies$ if this is correct, can solve bipartite matching in $O(|V|^3)$ time

Formulating Bipartite Matching as Flow



$$M = \{\{l, r\} \mid f'(l, r) = 1\} = \{\{l_1, r_1\}, \{l_2, r_2\}\}$$

(other max flows ⇔ matchings exist)

Correctness of this Formulation

Technical details:

- *integrality theorem:* if every capacity $c(u, v) \in \mathbb{Z}$ then every $f(u, v) \in \mathbb{Z}$ and $|f| \in \mathbb{Z}$
- $\exists$ matching M with cardinality $k = |M|$ iff $\exists$ some flow f with value $k = |f|$
  - key idea: pairing two vertices in the matching adds exactly one flow from $s \rightsquigarrow t$
  - there are no opportunities for flow aside from matched vertices
- $\implies$ a maximum flow in G' corresponds to a maximum matching in G

Summary

- classical max-flow problem can be solved in $O(|V|^3)$ time, in P
- robust max-flow problem (supports unreachable vertices, antiparallel edges, multiple sinks/sources) also in $O(|V|^3)$ time w/ worse constant factors, in P
- bipartite matching reduces to max-flow, so bipartite matching can be solved in $O(|V|^3)$ time, in P
- other practical, distinct problems reduce to max-flow or bipartite matching so take $O(|V|^3)$ time and are in P