Hopcroft–Karp Algorithm

An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs

Chen CUI, Zhenhao DOU, Yuxin ZHAO

May 18, 2018

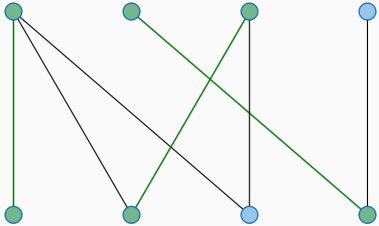


Table of Contents

1. Introduction

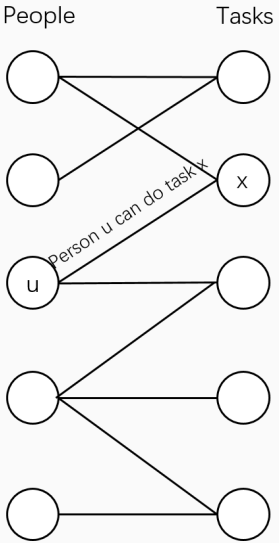
2. Matchings and Augmenting Paths

3. The Bipartite Case

Introduction

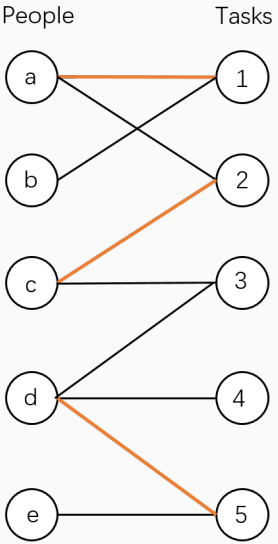
Bipartite Graphs

- Suppose we have a set of people L and set of tasks R
- Each person can do only some of the tasks
- Can model this as a bipartite graph $\rightarrow$



Bipartite Matching

- A **matching** gives an assignment of people to tasks
- Want to get as many tasks done as possible
- So, want a **maximum matching**: one that contains as many edges as possible.
- (This one is not maximum.)



Hopcroft–Karp Algorithm

Figure 1: John Hopcroft. 1986, Turing Award

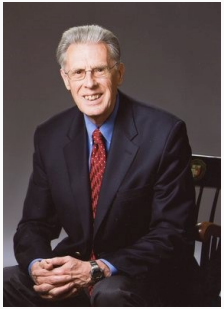


Figure 2: Richard M. Karp. 1985, Turing Award

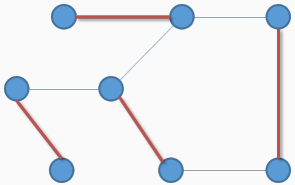
- Hungarian algorithm: time complexity $O(|E||V|)$
- Hopcroft–Karp algorithm: time complexity $O(|E|\sqrt{|V|})$
- Alt et al. algotithm:$O(|V|^{1.5}\sqrt{|E|/log|V|})$

Matchings and Augmenting Paths

Matching

Definition

- Let $G = (V, E)$ be a finite *undirected* graph (without loops, multiple edges, or isolated vertices)
- A set $M \subseteq E$ is a matching if **no** vertex $v \in V$ is incident with more than one edge in M
- Maximum matching is a matching of maximum cardinality
- A vertex v is free if it is incident with **no** edge in M



Augmenting Path

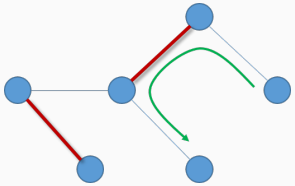
Definition

- A path (without repeated vertices)

$$P = (v_1, v_2), (v_2, v_3), \cdots, (v_{2k-1}, v_{2k})$$

  is called an augmenting path if its endpoints v_1 and v_{2k} are both free, and its edges are alternatively in $E - M$ and in M

- P has an odd number of edges
- $|P \cap (E - M)| = |P \cap M| + 1$
- $|P| = |P \cap (E - M)| + |P \cap M| = 2|P \cap M| + 1$



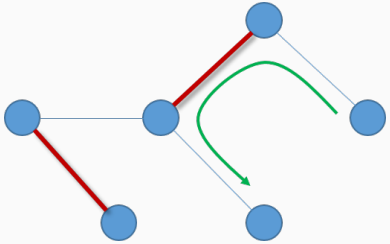
Lemma 1

Lemma

If M is a matching and P is an augmenting path relative to M, then $M \oplus P$ is a matching, and $|M \oplus P| = |M| + 1$.

Proof.

- $|P| = 2|P \cap M| + 1$
- $|M \oplus P| = |M| + |P| - 2|P \cap M|$
- $|M \oplus P| = |M| + 1$
- vertices in the complement of P have the same neighbors in M as in $M \oplus P$, and vertices in P have **exactly** one neighbor in $M \oplus P$



□

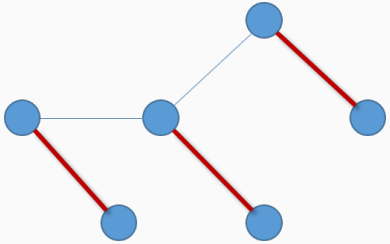
Lemma 1

Lemma

If M is a matching and P is an augmenting path relative to M, then $M \oplus P$ is a matching, and $|M \oplus P| = |M| + 1$.

Proof.

- $|P| = 2|P \cap M| + 1$
- $|M \oplus P| = |M| + |P| - 2|P \cap M|$
- $|M \oplus P| = |M| + 1$
- vertices in the complement of P have the same neighbors in M as in $M \oplus P$, and vertices in P have **exactly** one neighbor in $M \oplus P$

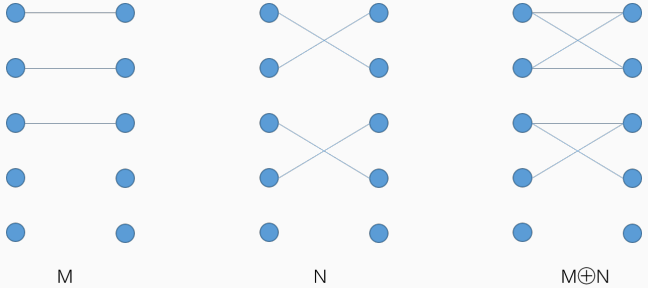


□

Theorem 1

Theorem

Let M and N be matchings. If $|M| = r$, $|N| = s$ and $s > r$, then $M \oplus N$ contains ***at least*** *$s - r$ vertex-disjoint augmenting paths relative to M.*



Theorem 1

Theorem

Let M and N be matchings. If $|M| = r$, $|N| = s$ and $s > r$, then $M \oplus N$ contains ***at least*** *$s - r$ vertex-disjoint augmenting paths relative to M.*

Proof.

- $\overline{G} = (V, M \oplus N)$
- $\forall v \in V$, v indicent with **at most one** edge from $N - M$ and $M - N$ respectively
- each connected component of $\overline{G}$ is either
  1. an isolated vertex
  2. a cycle of **even** length, with edges alternatively in $M - N$ and in $N - M$
  3. a path whose edges are alternatively in $M - N$ and in $N - M$

□

Theorem 1

Theorem

Let M and N be matchings. If $|M| = r$, $|N| = s$ and $s > r$, then $M \oplus N$ contains ***at least*** *$s - r$ vertex-disjoint augmenting paths relative to M.*

Proof.

- Let the components of $\overline{G}$ be $C_1, C_2, \cdots, C_g$, where $C_i = \{V_i, E_i\}$
- let $\delta(C_i) = |E_i \cap N| - |E_i \cap M| \in \{-1, 0, 1\}$
- $\delta(C_i) = 1$ **iff** C_i is an augmenting path relative to M

$$\begin{aligned}
\sum_i \delta(C_i) &= \sum_i (|E_i \cap N| - |E_i \cap M|) = \sum_i |E_i \cap N| - \sum_i |E_i \cap M| \\
&= |\bigcup_i E_i \cap N| - |\bigcup_i E_i \cap M| = |M \oplus N \cap N| - |M \oplus N \cap M| \\
&= |N - M| - |M - N| = (|N - M| + |M \cap N|) - (|M - N| + |M \cap N|) \\
&= |N| - |M| = s - r
\end{aligned}$$

Corollary 1

Corollary

*M is a maximum matching **iff** there is no augmenting path relative to M.*

Proof.

maximum matching $\Rightarrow$ no augmenting path

- Lemma 1

no augmenting path $\Rightarrow$ maximum matching

- Suppose that M^* is a matching of maximum cardinality, and that $|M| < |M^*|$
- $M \oplus M^*$ contains at least 1 vertex-disjoint augmenting paths relative to M

□

Naive Algorithm

Algorithm 1: Naive iterative scheme for computing a maximum matching

$M = \emptyset$

repeat

  Find an augmenting path P with respect to M

  $M = M \oplus P$

until *there is no augmenting with respect to M*

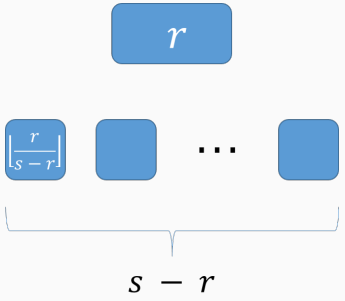
Corollary 2

Corollary

Let M be a matching. Suppose $|M| = r$, and suppose that the cardinality of a ***maximum*** *matching is s, $s > r$. Then there exists an augmenting path relative to M of length $\leq 2\lfloor r/(s-r) \rfloor + 1$.*

Proof.

- Let N be a **maximum** matching
- $M \oplus N$ contains $s - r$ vertex-disjoint augmenting paths relative to M
- **at most** r edges from $M \Rightarrow$ *one of them* **at most** $\lfloor r/(s-r) \rfloor$ edges from M
- **at most** $\leq 2\lfloor r/(s-r) \rfloor + 1$ *all altogether*



□

Theorem 2

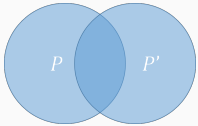
Theorem

Let M be a matching, P a shortest augmenting path relative to M, and P' an augmenting path relative to $M \oplus P$. Then

$$|P'| \geq |P| + |P \cap P'|$$

Proof.

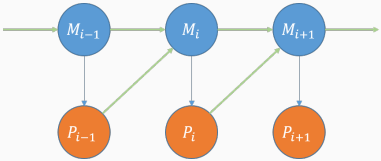
- Let $N = M \oplus P \oplus P'$. Then N is a matching and $|N| = |M| + 2$
- $M \oplus N$ contains 2 vertex-disjoint augmenting paths relative to M; call them P_1 and P_2
- $M \oplus N = M \oplus (M \oplus P \oplus P') = P \oplus P' \Rightarrow |P \oplus P'| \geq |P_1| + |P_2| \geq 2|P|$
- $|P \oplus P'| = |P| + |P'| - 2\,|P \cap P'| \Rightarrow |P'| \geq |P| + 2|P \cap P'|$ (Warning!)



□

Notations

- Initially $M_0 = \emptyset$
- Compute a sequence $M_0, M_1, M_2, \cdots, M_i, \cdots$
- P_i is a shortest augmenting path relative to M_i
- $M_{i+1} = M_i \oplus P_i$



Corollary

$|P_i| \leq |P_{i+1}|$

Proof.

P_{i+1} is a augmenting path relative to $M_i \oplus P_i$, then

$$|P_{i+1}| \geq |P_i| + |P_i \cap P_{i+1}| \geq |P_i|$$

□

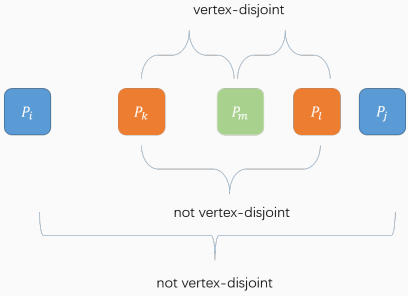
Corollary 4

Corollary

For all i and j such that $|P_i| = |P_j|, (i \neq j)$, P_i and P_j are vertex-disjoint.

Proof.

- By contradiction, $|P_i| = |P_j|, i < j, P_i$ and P_j are **not** vertex-disjoint.
- $\exists k, l$ such that $i \leq k < l \leq j$, P_k and P_l are **not** vertex-disjoint **and** $\forall m \in (k, l)$, P_m is vertex-disjoint from P_k and P_l



□

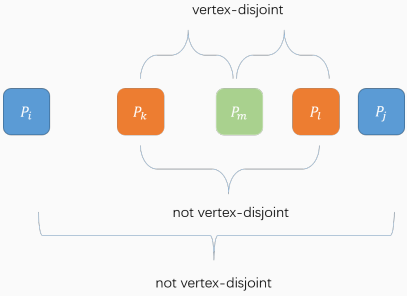
Corollary 4

Corollary

For all i and j such that $|P_i| = |P_j|$, $(i \neq j)$, P_i and P_j are vertex-disjoint.

Proof.

- P_l is an augmenting path relative to $M_k \oplus P_k$
  - $\forall m \in (k, l)$, P_m is vertex-disjoint from P_k and P_l
- $\Rightarrow |P_l| \geq |P_k| + |P_k \cap P_l|$
- But $|P_l| = |P_k|$, so $|P_k \cap P_l| = 0$
  $\Rightarrow P_k$ and P_l have no edges in common
- if P_k and P_l had a vertex v in common
  - they would have a common edge in $M_k \oplus P_k$



□

Theorem 3

Theorem

Let s be the cardinality of a maximum matching. The number of distinct integers in the sequence

$$|P_0|, |P_1|, \cdots, |P_i|, \cdots$$

is less than or equal to $2\lfloor\sqrt{s}\rfloor + 2$.

Proof.

- Let $r = \lfloor s - \sqrt{s}\rfloor$. Then $|M_r| = r$ and, by Corollary 2,

$$|P_r| \leq 2\lfloor s - \sqrt{s}\rfloor/(s - \lfloor s - \sqrt{s}\rfloor) + 1 \leq 2\lfloor\sqrt{s}\rfloor + 1$$

- for each $i < r$, $|P_i|$ is one of the $\lfloor\sqrt{s}\rfloor + 1$ positive odd integers $\leq 2\lfloor\sqrt{s}\rfloor + 1$
- Also $|P_{r+1}|, \cdots, |P_s|$ contribute at most $\lceil\sqrt{s}\rceil$ distinct integers
- total $\leq \lfloor\sqrt{s}\rfloor + 1 + \lceil\sqrt{s}\rceil \leq 2\lfloor\sqrt{s}\rfloor + 2$

□

Maximum Matching Algorithm

Algorithm 2: Maximum matching algorithm

$M = \emptyset$

repeat

$l =$ the length of a **shortest** augmenting path relative to M

Find a **maximal** set of paths $\{Q_1, Q_2, \cdots, Q_t\}$ with the properties that

1. $\forall Q_i, Q_i$is an augmenting path relative to M and $|Q_i| = l$
2. the Q_i are vertex-disjoint

$M = M \oplus Q_1 \oplus Q_2 \oplus \cdots \oplus Q_t$

until *no such path exists*

Corollary 5

Corollary

If the cardinality of a maximum matching is s, then Algorithm constructs a maximum matching within $2\lfloor\sqrt{s}\rfloor + 2$ *executions of Find step.*

Proof.

Consider the ith and (i+1)th executions of Find steps.

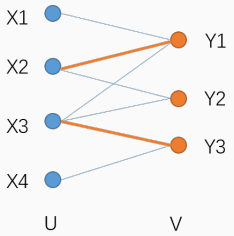
- $l_{i+1} \geq l_i$, by Corollary 3
- $l_{i+1} = l_i \Rightarrow \forall P_a \in phase_i, P_b \in phase_{i+1}$ P_a and P_b are vertex-disjoint

□

The Bipartite Case

Algorithm

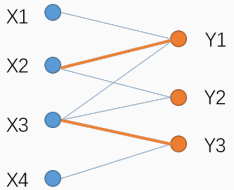
- let *U* and *V* be the two sets in the bipartition of *G*
- let the matching from *U* to *V* at any time be represented as the set *M*
- The algorithm is run in phases. Each phase consists of the following steps.
  - A breadth-first search partitions the vertices of the graph into layers.
  - finds a maximal set of vertex disjoint augmenting paths of minimum length by *DFS*
  - Use these augmenting paths to enlarge *M*

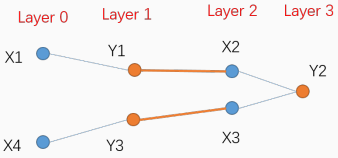


Partitions the Vertices

A breadth-first search partitions the vertices of the graph into layers.

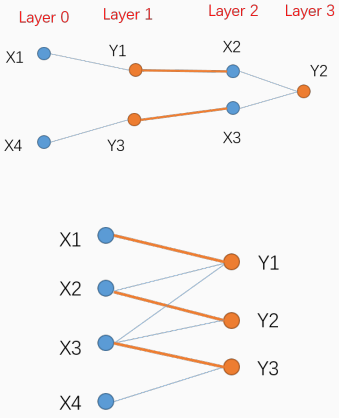
- The free vertices in U form the first layer of the partitioning.
- the traversed edges are required to alternate between matched and unmatched.
  - from a vertex in U, only unmatched edges may be traversed.
  - from a vertex in V only matched edges may be traversed.
- The search terminates at the first layer k where one or more free vertices in V are reached.





Maximal Set of Vertex Disjoint Augmenting Paths

- finds a *maximal* set of vertex disjoint augmenting paths of length k
  - the DFS is only allowed to follow edges that lead to an unused vertex in the next layer
  - paths in the DFS must alternate between matched and unmatched edges
  - Any vertex encountered during the DFS can immediately be marked as used.
- $O(|E|)$ running time for the DFS
- Every one of the paths found in this way is used to enlarge M



Hopcroft-Karp Algorithm

Algorithm 3: Hopcroft-Karp Algorithm

```
1 while exist augmenting paths
2     BFS(G, M)
3     used[] = false
4     for each free boy vertex u
5         DFS(G, M, u)
```

Construct Graph

Algorithm 4: BFS(G,M)

```
1  Create empty queue Q
2  Push all free boy vertices into Q
3  while Q ≠ ∅
4      u = DEQUEUE(Q)
5      if u.type= boy
6          for each v ∈ G.Adj[u]
7              if not visited v
8                  v.dist = u.dist + 1
9                  If not free_girl_found ENQUEUE(Q, v)
10     else if u.type= girl
11         if ∃(u, v) ∈ M
12             v.dist = u.dist + 1
13             ENQUEUE(Q, v)
14         else free_girl_found = true
```

Finding a matching for u

Algorithm 5: Depth-First-Search

```
1 Function DFS( G, M, u )
2     for each v ∈ G.Adj[u]
3         if not used[v] and v.dist = u.dist + 1
4             used[v] = true
5             if v not matched or DFS(G, M, the current macthing of v)
6                 M = M ∪ {(u, v)}
7                 return true
8     return false
```

Conclusion

- Time Complexity
  - a single phase may be implemented in $O(|E|)$ time.
  - the algorithm performs a total of at most $2\sqrt{|V|}$ phases
  - it takes a total time of $O(|E|\sqrt{|V|})$ in the worst case.
- Non-bipartite graphs
  - the task of finding the augmenting paths within each phase is more difficult
  - Micali, S.; Vazirani, V. V. (1980); "An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs"
  - Peterson, Paul A.; Loui, Michael C. (1988), "The general maximum matching algorithm of Micali and Vazirani"

Questions?

Thanks.