Advanced Core in Algorithm Design #9
算法設計要論 第 9 回

Yasushi Kawase
河瀬 康志

Dec. 6th, 2022

last update: 12:32pm, December 6, 2022

Schedule

Lec. #	Date	Topics
1	10/4	Introduction, Stable matching
2	10/11	Basics of Algorithm Analysis, Greedy Algorithms (1/2)
3	10/18	Greedy Algorithms (2/2)
4	10/25	Divide and Conquer (1/2)
5	11/1	Divide and Conquer (2/2)
6	11/8	Dynamic Programming (1/2)
7	11/15	Dynamic Programming (2/2)
—	11/22	Thursday Classes
8	11/29	Network Flow (1/2)
9	12/6	Network Flow (2/2)
10	12/13	NP and Computational Intractability
11	12/20	Approximation Algorithms (1/2)
12	12/27	Approximation Algorithms (2/2)
13	1/10	Randomized Algorithms

Outline

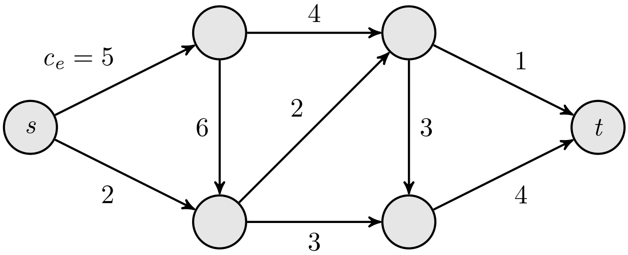
Max-flow problem

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t flow of maximum value $\mathrm{val}(f)$

$\sum_{v\colon e=(s,v)\in E} f_e - \sum_{v\colon e=(v,s)\in E} f_e$

Example



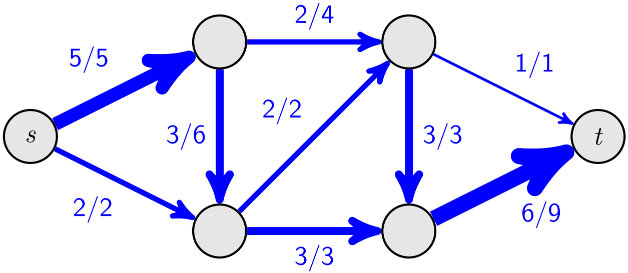
Max-flow problem

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t flow of maximum value $\mathrm{val}(f)$

$\sum_{v:\, e=(s,v)\in E} f_e - \sum_{v:\, e=(v,s)\in E} f_e$

Example



$\mathrm{val}(f) = 7$

Algorithms for the max-flow problem

This lecture

- Ford–Fulkerson algorithm → $\mathrm{O}(mC)$ time (pseudo polynomial-time)
- Scaling algorithm → $\mathrm{O}(m^2 \log C)$ time (weak polynomial-time)
- Edmonds–Karp algorithm → $\mathrm{O}(m^2 n)$ time (strong polynomial-time)

State of the Art

- $\mathrm{O}(mn)$ time [Orlin 2013]
- $m^{1+o(1)} \log C$ time [Chen et al., 2022]

Ford–Fulkerson algorithm

$\mathrm{Augment}(f, c, P)$

$\delta \leftarrow$bottleneck capacity of augmenting path P;
foreach $e \in P$ **do**
  if $e \in E$ **then** $f_e \leftarrow f_e + \delta$;
  else $f_{\bar{e}} \leftarrow f_{\bar{e}} - \delta$;
Return f;

Output of $\mathrm{Augment}(f, c, P)$ is a flow

Ford–Fulkerson algorithm

$f_e \leftarrow 0$ for each $e \in E$;
$G_f \leftarrow$ residual network of G with respect to flow f;
while $\exists$ *an s–t path P in G_f* **do**
  $f \leftarrow \mathrm{Augment}(f, c, P)$;
  Update G_f;
Return f;

Edmonds–Karp algorithm

Choosing augmenting path that uses the fewest edges (can be found via BFS)

Edmonds–Karp algorithm

$f_e \leftarrow 0$ for each $e \in E$;
$G_f \leftarrow$ residual network of G with respect to flow f;
while $\exists$ *an s–t path in G_f* **do**
    $P \leftarrow$ a shortest s–t path in G_f;
    $f \leftarrow \mathrm{Augment}(f, c, P)$;
    Update G_f;
Return f;

Overview of analysis

$d(f)$: the length (number of edges) of a shortest augmenting path

Lemma 1

$d(f)$ never decreases during the execution

Lemma 2

$d(f)$ increases at least once per m iterations

Theorem

strongly polynomial time

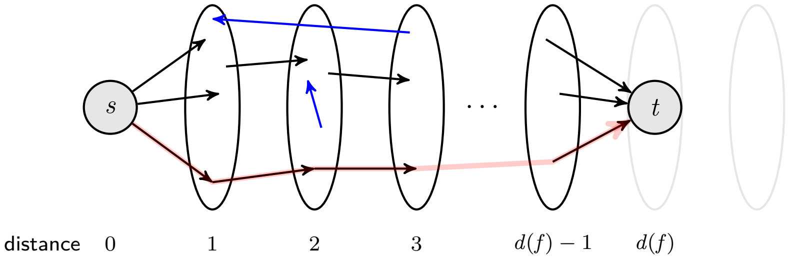
Edmonds–Karp algorithm can be implemented to run in $\mathrm{O}(m^2 n)$ time

- at most $n - 1$ different lengths
- at most m augmentation paths of length k
- $\mathrm{O}(m)$ time to find a shortest augmenting path

Lemma 1

$d(f)$ never decreases during the execution

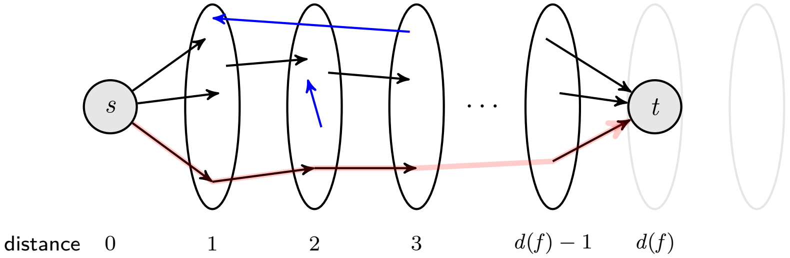
- classify vertices based on their distance from s in G_f
- three types of edges: forward, sideways, backwards
- new edges added to G_f by augmentation must be backwards



Lemma 2

$d(f)$ increases at least once per m iterations

- at least one forward edge is deleted from G_f per augmentation
- no forward edge will be added until $d(f)$ increases
- within m iterations, there will be no paths using only forward edges



Outline

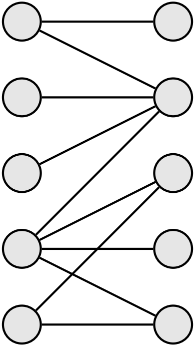
Bipartite matching problem

Problem

- Input: bipartite graph $G = (A, B; E)$
- Goal: find a maximum cardinality matching

a set of pairwise non-adjacent edges

Example



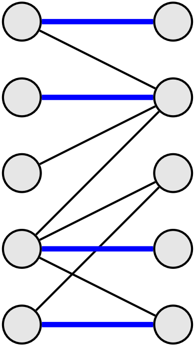
Bipartite matching problem

Problem

- Input: bipartite graph $G = (A, B; E)$
- Goal: find a maximum cardinality matching

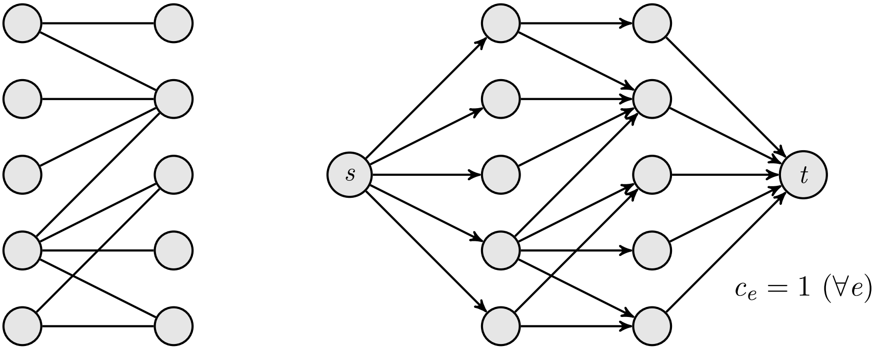
a set of pairwise non-adjacent edges

Example



Bipartite matching problem can be reduced to max-flow problem

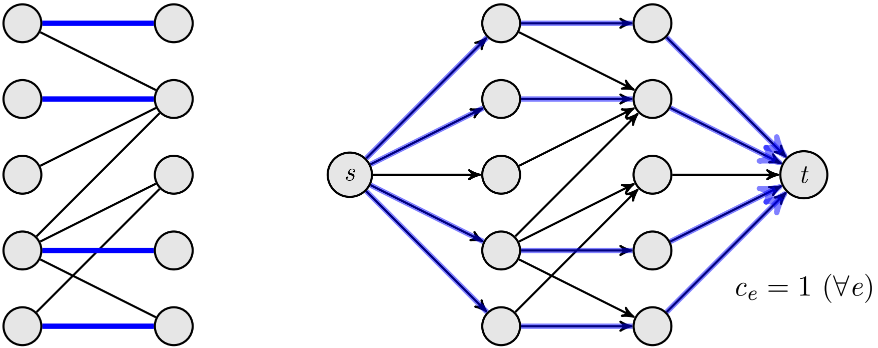
Recall that there exists an integral max-flow



Reduction

Bipartite matching problem can be reduced to max-flow problem

Recall that there exists an integral max-flow



Application of Ford–Fulkerson algorithm

Theorem

Ford–Fulkerson algorithm finds a maximum matching in $\mathrm{O}(mn)$ time

- The size of the maximum matching = the value of the maximum flow
- The size of the maximum matching is $\mathrm{O}(n)$
- Ford–Fulkerson: $\mathrm{O}(n)$ augmentations, each one takes $\mathrm{O}(m)$ time

  → $\mathrm{O}(mn)$ time

Hall's theorem

$G = (A, B; E)$: bipartite graph

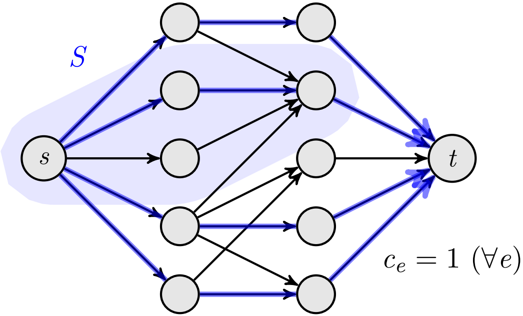
Theorem

vertices adjacent to X

$\exists$ matching M of size $|A| \iff |\Gamma(X)| \geq |X|\ (\forall X \subseteq A)$

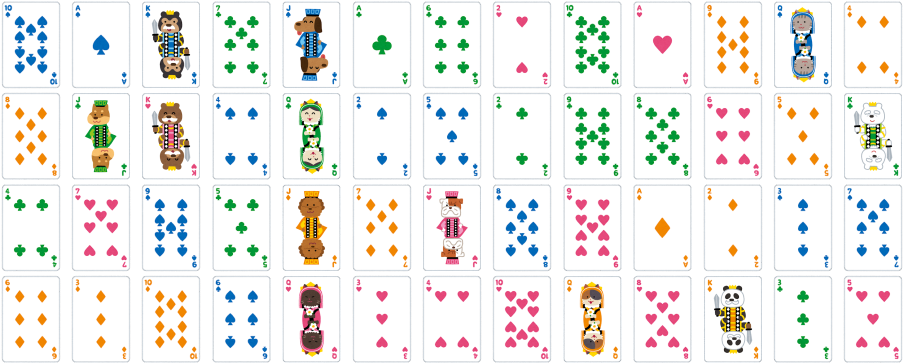
($\Rightarrow$) $\because \Gamma(X)$ contains all the partners of X in M

($\Leftarrow$) $\because |\Gamma(S \cap A)| < |S \cap A|$ for the set of vertices S reachable from s in G_f



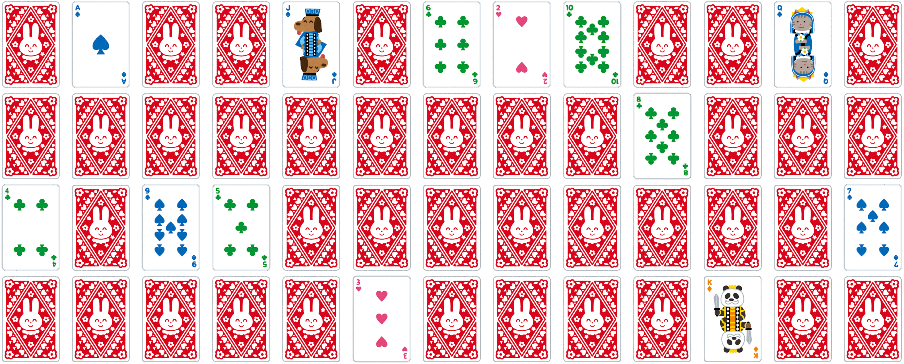
Application of Hall's theorem: Card magic

- Deal a standard deck of cards out into 13 piles of 4 cards each
- Is it always possible to select exactly 1 card from each pile, such that the 13 selected cards contain exactly one card of each rank?



Application of Hall's theorem: Card magic

- Deal a standard deck of cards out into 13 piles of 4 cards each
- Is it always possible to select exactly 1 card from each pile, such that the 13 selected cards contain exactly one card of each rank?



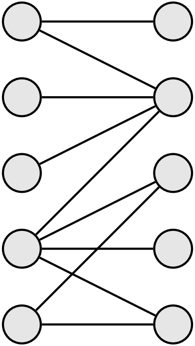
Bipartite minimum vertex cover problem

Problem

- Input: bipartite graph $G = (A, B; E)$
- Goal: find a minimum cardinality vertex cover

a set of vertices that includes at least one endpoint of every edge

Example



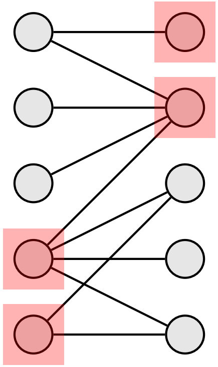
Bipartite minimum vertex cover problem

Problem

- Input: bipartite graph $G = (A, B; E)$
- Goal: find a minimum cardinality vertex cover

a set of vertices that includes at least one endpoint of every edge

Example



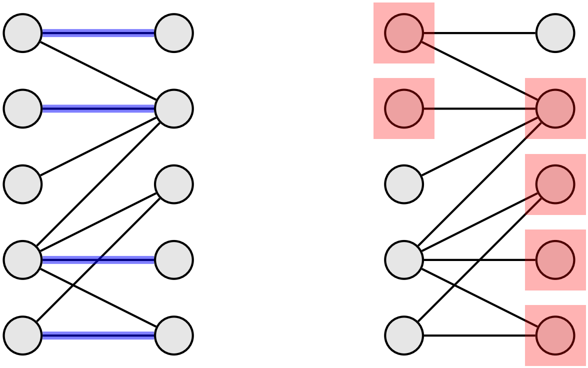
Weak duality

Proposition

$|M| \leq |C|$ for any matching M and vertex cover C

$\because$ each $e \in M$ must be covered by a distinct vertex

Example



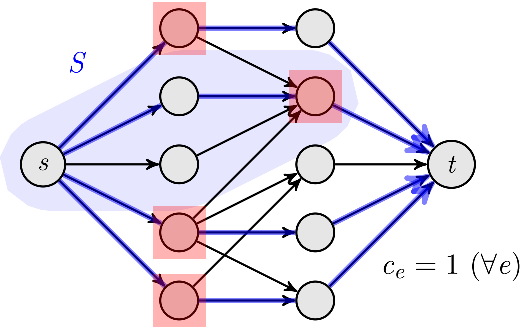
Strong duality

the residual graph at the end of Ford–Fulkerson algorithm

Kőnig's theorem

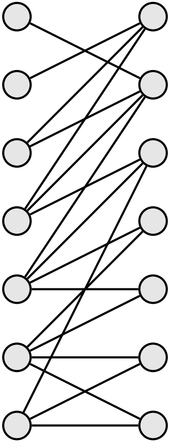
$\max_{M:\,\text{matching}} |M| = \min_{C:\,\text{vertex cover}} |C|$

- S: the set of vertices reachable from s in G_f
- $(A \setminus S) \cup (B \cap S)$ is VC as $\nexists (u, v) \in E$ such that $u \in A \cap S$ and $v \in B \setminus S$



Quiz

Find a maximum matching and a minimum vertex cover.



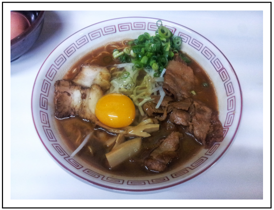
Outline

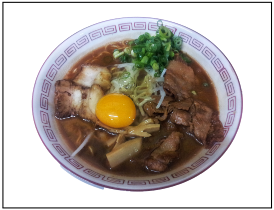
Image segmentation

Problem

- Input: image
- Goal: label each pixel as either the foreground or the background

Example

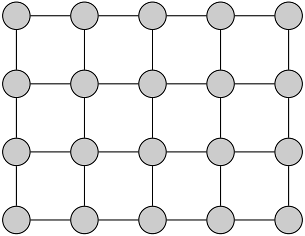




Segmentation problem

Problem

- Input:
  - undirected graph $G = (V, E)$ (V: pixel, E: neighbor)
  - likelihood $a_i \in \mathbb{R}_+$ that $i \in V$ belongs to the foreground
  - likelihood $b_j \in \mathbb{R}_+$ that $j \in V$ belongs to the background
  - separation penalty $p_{ij} \in \mathbb{R}_+$ for each $\{i, j\} \in E$
- Goal: find $X \subseteq V$ that maximizes $q(X) := \sum_{i \in X} a_i + \sum_{j \in V \setminus X} b_j - \sum_{\{i,j\} \in \delta(X)} p_{ij}$

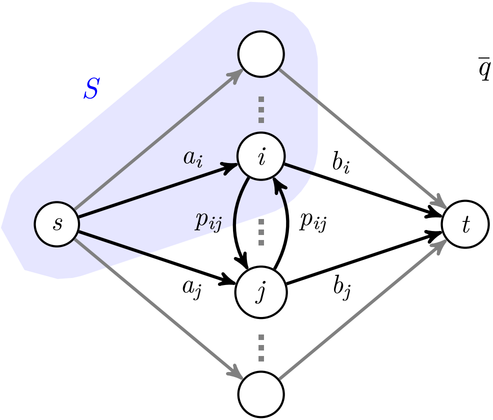


Reduction to minimum cut

Observation

maximizing $q(X)$ $\iff$ minimizing $\bar{q}(X)$

- $q(X) \coloneqq \sum_{i \in X} a_i + \sum_{j \in V \setminus X} b_j - \sum_{\{i,j\} \in \delta(X)} p_{ij}$
- $\bar{q}(X) \coloneqq \sum_{i \in V}(a_i + b_i) - q(X) = \sum_{i \in X} b_i + \sum_{j \in V \setminus X} a_j + \sum_{\{i,j\} \in \delta(X)} p_{ij}$



$\bar{q}(S \setminus \{s\}) = \mathrm{cap}(S)$

Result

Algorithm

1. Construct the corresponding flow network;
2. Find the minimum-cut S for the network;
3. **Return** $S \setminus \{s\}$;

Theorem

The solution to the segmentation problem can be obtained by a minimum-cut algorithm (e.g., $O(|E|^2|V|)$ time)

Outline

1. Edmonds–Karp algorithm
2. Bipartite matching
3. Image segmentation
4. Densest subgraph

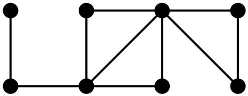
Densest subgraph problem

Problem

- Input: undirected graph $G = (V, E)$
- Goal: find a nonempty $S \subseteq V$ that maximizes $\mathrm{dens}(S) := |E(S)|/|S|$

$\{\{u, v\} \in E \mid u, v \in S\}$

Example



$\mathrm{O}(mn)$ possibilities of density: ℓ/k for $k = 1, 2, \ldots, n$ and $\ell = 0, 1, \ldots, m$

→ sufficient to solve the existence of S s.t. $\mathrm{dens}(S) > \alpha$

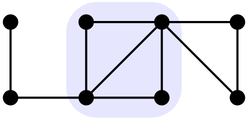
Densest subgraph problem

Problem

- Input: undirected graph $G = (V, E)$
- Goal: find a nonempty $S \subseteq V$ that maximizes $\mathrm{dens}(S) := |E(S)|/|S|$

$\{\{u, v\} \in E \mid u, v \in S\}$

Example



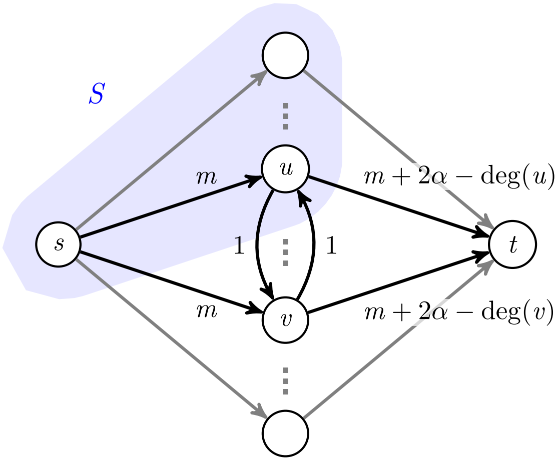
$\mathrm{O}(mn)$ possibilities of density: ℓ/k for $k = 1, 2, \ldots, n$ and $\ell = 0, 1, \ldots, m$

→ sufficient to solve the existence of S s.t. $\mathrm{dens}(S) > \alpha$

Reduction

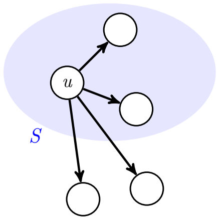
Proposition

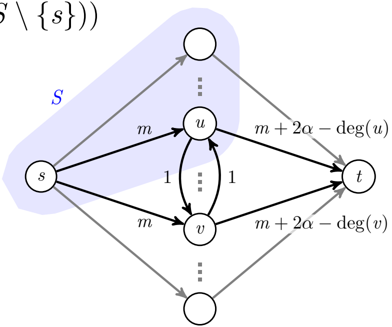
- $\mathrm{cap}(S) = mn + 2(|S| - 1)(\alpha - \mathrm{dens}(S \setminus \{s\}))$
- $\mathrm{cap}(S) < mn \iff \mathrm{dens}(S \setminus \{s\}) > \alpha$



Proof of Proposition

$$\begin{aligned}
\mathrm{cap}(S) &= \sum_{u\in S\setminus\{s\}} (m + 2\alpha - \deg(u)) + \sum_{u\in V\setminus S} m + \sum_{\{u,v\}\in E\colon u\in S,\, v\notin S} 1 \\
&= mn + 2\alpha(|S|-1) - \left(\sum_{u\in S\setminus\{s\}} \deg(u) - \sum_{\{u,v\}\in E\colon u\in S,\, v\notin S} 1 \right) \\
&= mn + 2\alpha(|S|-1) - 2\big|E[S\setminus\{s\}]\big| \\
&= mn + 2(|S|-1)(\alpha - \mathrm{dens}(S\setminus\{s\}))
\end{aligned}$$





Algorithm

- Sort $P \coloneqq \left\{\ell/k \mid k \in \{1, 2, \dots, n\},\ \ell \in \{0, 1, \dots, m\}\right\}$ $\mathrm{O}(mn \log n)$ time
- compute $\max\{\alpha \in P \mid \mathrm{dens}(S) \leq \alpha\ (\forall S)\}$ by binary search
  - $\mathrm{O}(\log n)$ min-cut problems (with $n + 2$ vertices, $m + 2n$ edges)
  - min-cut problem can be solve in $\mathrm{O}(m^2 n)$ time by Edmonds–Karp

→ total running time is $\mathrm{O}(m^2 n \log n)$