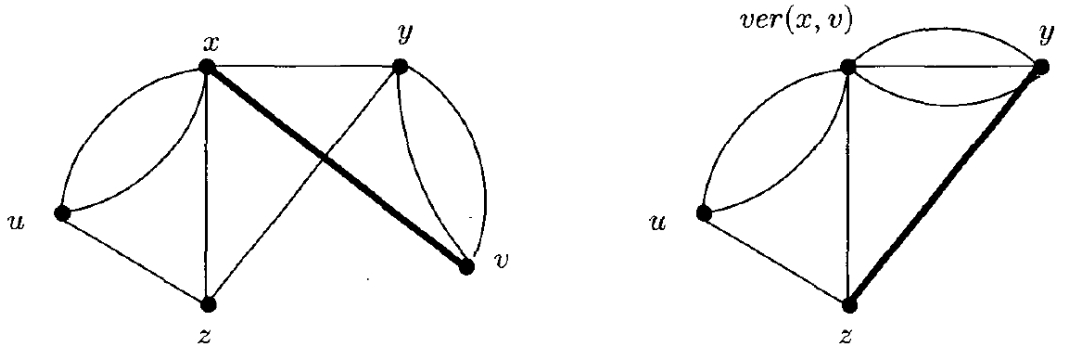
- 教材讨论
  - JH第5章第3节第5小节

问题1：随机算法

- 用随机算法解优化问题与判定问题有什么异同？
- 解优化问题的随机算法与近似算法有什么异同？
- 随机算法重复执行对解判定与解优化问题有什么异同？

问题2： Min-CUT

- 如何用最大流算法解Min-CUT问题？局限性？
- **Contraction**



(a) (b)

"any edge contraction does not reduce the size of a minimal cut in G."

Random Contraction

Algorithm 5.3.5.1. RANDOM CONTRACTION

Input: A connected multigraph $G = (V, E)$.
Output: a cut (V_1, V_2) of G.
Step 1: Set, for every $v \in V$, $label(v) = \{v\}$.
Step 2: **while** G has more than 2 vertices
do begin choose uniformly at random an edge $e = \{x, y\} \in E(G)$;
$G := G/\{e\}$;
set $label(z) = label(x) \cup label(y)$ for the new vertex z of G,
and replace the edges as described above
end
Step 3: **if** $G = (\{u, v\}, E')$ for a multiset E',
then output$(label(u), label(v))$.

How about ***RC*** algorithm?

Theorem 5.3.5.2. RANDOM CONTRACTION *is a polynomial-time randomized optimization algorithm that finds an optimal cut with a probability of at least* $\frac{2}{n\cdot(n-1)}$ *for any multigraph of* n *vertices.*

- 获得最优解的途径就是每一次都没用选中最优解中的边。$Prob\left(\bigcap_{i=1}^{n-2} Event_i\right)$

$$Prob(Event_1) = \frac{|E| - |E(C_{min})|}{|E|} = 1 - \frac{k}{|E|} \underset{(5.14)}{\geq} 1 - \frac{k}{k \cdot n/2} = 1 - \frac{2}{n}.$$

"the total number of edges of G *is at least* $n \cdot k/2$*"*

问题3：The Improvement of ***RC***

- Random Contraction算法可能无法得到最优解的关键是什么？如何避免这样的问题？

we see that the first contraction involves an edge from $E(C_{min})$ with small probabilities $\frac{2}{n}, \frac{2}{n-1}, \frac{2}{n-2}, \ldots$. The key observation is that these probabilities grow and may be very large (even 2/3 in the last contraction). The first natural idea could be to use RANDOM CONTRACTION to reduce G to some G/F of l vertices and then to compute a minimal cut of G/F by the best known deterministic algorithm.

问题4：*l*-Comb-Contract

- *l*-Comb-Contract算法思想？

Algorithm 5.3.5.4. l-COMB-CONTRACT

Input: a multigraph $G = (V, E)$ on n vertices, $n \in \mathbb{N}$.
Output: a cut (V_1, V_2) of G.
Step 1: Apply RANDOM CONTRACTION on G in order to get a multigraph G/F of $l(n)$ vertices.
Step 2: Apply a deterministic algorithm to compute a minimal cut of G/F. Output the corresponding cut of G.

$$l : \mathbb{N} \rightarrow \mathbb{N},\ 1 \leq l(n) \leq n$$

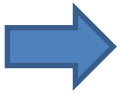
代价与结果

Theorem 5.3.5.5. *For any function* $l : \mathbb{N} \to \mathbb{N}$, $1 \leq l(n) \leq n$, *the randomized algorithm* l-COMB-CONTRACT *works in time* $O(n^2 + (l(n))^3)$, *and it finds a minimal cut with a probability of at least*

$$\binom{l(n)}{2} \Big/ \binom{n}{2}.$$

Corollary 5.3.5.6. *For every* $l : \mathbb{N} \to \mathbb{N}$, $1 \leq l(n) \leq n$, $\frac{n^2}{(l(n))^2}$ *repetitions of* l-COMB-CONTRACT *provide an optimal cut with probability at least* $1 - 1/e$.

$$O\left((n^2 + (l(n))^3) \cdot \frac{n^2}{(l(n))^2}\right) = O\left(\frac{n^4}{(l(n))^2} + n^2 \cdot l(n)\right)$$



the best choice for l is $l(n) = \lfloor n^{2/3} \rfloor$

问题5：RRC算法

- RRC算法思想？

Algorithm 5.3.5.7. RRC(G)

Input: A multigraph $G = (V, E)$, $|V| = n$.
Output: A cut (V_1, V_2) of G.
Procedure: **if** $n \leq 6$ **then** compute a minimal cut of G by a deterministic method.
else begin $h := \lceil 1 + n/\sqrt{2} \rceil$;
realize two independent runs of RANDOM CONTRACTION on G in order to obtain multigraphs G/F_1 and G/F_2 of the size h;
RRC(G/F_1);
RRC(G/F_2);
return the smaller of the two cuts produced by RRC(G/F_1) and RRC(G/F_2).
end.

RRC的效果

Theorem 5.3.5.8. *The algorithm* RRC *works in* $O(n^2 \log_2 n)$ *time and finds a minimal cut with a probability of at least*

$$\frac{1}{\Omega(\log_2 n)}.$$

问题6：算法讨论

- 三种算法
  - Random Contraction
  - *l*-Comb-Contract
  - Recursive Random Contraction

- 两种改进思路的异同

问题7：随机算法策略

- 如何理解？

(i) Execute the parts of the randomized computation where the probability of success decreases drastically in a completely deterministic way. This approach may be successful if the "derandomization" of these parts can be efficiently done.

(ii) Do not execute a lot of complete independent runs of the algorithm on the same input. Prefer to execute many independent runs of the computation parts in which the probability of success drastically decreases, and only a few independent runs of the computation parts, where the decrease of the probability of success is almost negligible. This approach is especially efficient if the computation parts essentially decreasing the success probability are short.