大数据算法-2025 春

Lecture 4: 最小割的 Karger 算法

2025.3.18

Lecturer: 丁虎 *Scribe:* 王运韬

1 图上的最小割

给定一个无向图 $G = (V, E)$ 和边上的容量 $c : E \to \mathbb{R}_{\geq 0}$。一个割 (S, T) 是顶点集 V 的一个划分，其中 $S \cup T = V$ 且 $S \cap T = \emptyset$。割的容量定义为：

$$c(S, T) = \sum_{u \in S, v \in T} c(u, v)$$

其中，如果 $(u, v) \notin E$，则约定 $c(u, v) = 0$。

我们的问题是：找到一个割 (S, T)，使得 $c(S, T)$ 的值最小。

1.1 Karger 的随机收缩算法

Karger 的算法是一种随机化的方法，用于近似求解最小割问题。其核心思想是通过重复随机选择并收缩图中的边，直到图中只剩下两个顶点为止。具体步骤如下：

1. 当图中顶点的数量超过 2 个时，执行以下操作：
   (a) 从边集 E 中按照均匀分布随机选择一条边 (u, v)。
   (b) 将顶点 u 和 v 合并成一个新的顶点 $S_{u,v}$，并移除边 (u, v)。
   (c) 对于每条原来与 u 或 v 相连的边，例如 (x, u) 或 (x, v)，将其替换为一条新的边 $(x, S_{u,v})$，并赋以容量 $\boldsymbol{c}(x, u) + c(x, v)$。
2. 对得到的图重复上述步骤，直到图中只剩下 2 个顶点 $S_X, S_{\bar{X}}$ 为止。
3. 最后，返回由这两个顶点定义的割 $(X, \bar{X})$ 及其容量。

需要注意的是，每次运行该算法时，返回的割是随机的。

1.2 成功的概率

假设图 G 的最小割的大小为 k，即 $\min_{S,T} c(S,T) = k$。我们希望分析 Karger 算法能够成功找到这个最小割的概率。

首先，图 G 中至少有 k 条边横跨最小割 (S^*, T^*)。因为 G 是一个无向图，每个顶点的度数必须至少为 k。如果某个顶点的度数小于 k，那么以这个顶点为一侧的割的容量将小于 k，这与 k 是最小割大小的假设矛盾。因此，图 G 的总边数满足 $|E| \geq \frac{kn}{2}$，其中 $n = |V|$ 是图中顶点的总数。

现在，我们分析算法在第一次收缩时的行为：此时，图中的边总数为 $|E| \geq \frac{kn}{2}$。m 横跨最小割 (S^*, T^*) 的边数恰好为 k。随机选择一条边时，选择一条横跨最小割的边的概率为：

$$\Pr[\text{选择的边横跨}(S^*, T^*)] = \frac{k}{|E|} \leq \frac{k}{\frac{kn}{2}} = \frac{2}{n}$$

因此，选择一条不横跨最小割的边的概率至少为：

$$1 - \frac{2}{n}$$

接下来，考虑第二次收缩。此时，图中的顶点数变为 $n-1$。由于每个顶点的度数仍然至少为 k，边数虽然减少，但总边数仍然满足 $|E'| \geq \frac{k(n-1)}{2}$。类似地，选择一条不横跨最小割的边的概率至少为：

$$1 - \frac{2}{n-1}$$

以此类推，在算法的每次迭代中，顶点数逐渐减少。总体而言，算法成功找到最小割的概率（即在整个过程中从未选择横跨最小割的边）可以表示为：

$$\prod_{i=3}^{n} \left(1 - \frac{2}{i}\right)$$

我们计算这个乘积：

$$\prod_{i=3}^{n} \left(1 - \frac{2}{i}\right) = \frac{(n-2)(n-3)\cdots 3 \cdot 2 \cdot 1}{n(n-1)(n-2)(n-3)\cdots 3} = \frac{(n-2)!}{n!/(2 \cdot 1)} = \frac{(n-2)!}{\frac{n(n-1)(n-2)!}{2}} = \frac{2}{n(n-1)}$$

因此，Karger 算法单次运行成功找到最小割的概率至少为：

$$\frac{2}{n(n-1)}$$

这表明成功概率的量级为 $\Omega\left(\frac{1}{n^2}\right)$。

1.3 改进成功概率

单次运行 Karger 算法找到最小割的概率较低，仅为 $\Omega(1/n^2)$。然而，我们可以通过多次独立运行该算法并从所有运行结果中选择容量最小的割来显著提高成功概率。

具体来说：假设我们运行算法 n^2 次。每次运行失败（即未找到最小割）的概率为：

$$1 - \frac{2}{n(n-1)}$$

n^2 次运行全部失败的概率为：

$$\left(1 - \frac{2}{n(n-1)}\right)^{n^2}$$

因为 $\frac{2}{n(n-1)} \approx \frac{2}{n^2}$（当 n 较大时），我们可以近似计算：

$$\left(1 - \frac{2}{n(n-1)}\right)^{n^2} \leq \left(1 - \frac{1}{n^2}\right)^{n^2}$$

根据不等式 $1 - x \leq e^{-x}$，我们有：

$$\left(1 - \frac{1}{n^2}\right)^{n^2} \leq e^{-\frac{n^2}{n^2}} = e^{-1} \approx 0.3679$$

因此，至少有一次运行成功的概率为：

$$1 - \left(1 - \frac{2}{n(n-1)}\right)^{n^2} \geq 1 - e^{-1} \approx 0.6321$$

这意味着，通过运行 n^2 次 Karger 算法，我们以超过 63% 的概率能够找到最小割。通过运行 $O(n^2)$ 次，成功概率可以提升到一个常数级别。每次运行算法要进行 $n-1$ 次边收缩, 每次边收缩要用时 $O(n)$, 总的算法复杂度为 $O(n^4)$. 相当低效.

2 改进的 Karger-Stein 算法

在上一节中，我们分析了 Karger-Stein 算法的基本版本。现在，我们提出一个改进版本，通过更细致的递归调用策略来进一步提高效率。

2.1 算法描述

改进的 Karger-Stein 算法如下：

1. 如果图 G 的顶点数 $|V| \leq 6$，则使用暴力枚举法找到最小割

2. 否则：

   (a) 设 $t = \lceil 1 + |V|/\sqrt{2} \rceil$

   (b) 构造两个图 G_1 和 G_2，每个都是通过将 G 收缩到 t 个顶点得到的

   (c) 递归计算 G_1 和 G_2 的最小割

   (d) 返回两个结果中较小的那个

2.2 运行时间分析

该算法的运行时间由以下递归关系给出：

$$T(n) = 2T\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil\right) + O(n^2)$$

我们可以使用主定理来分析这个递归关系。设收缩到 t 个顶点需要 $O(n^2)$ 时间。

Theorem 2.1. *改进的 Karger-Stein 算法的运行时间为 $O(n^2 \log n)$。*

Proof. 展开递归树：

- 第一层：2 次调用 (生成两个子问题, 分别关于 G_1 和 G_2)，规模为 $\approx n/\sqrt{2}$
- 第二层：4 次调用 (G_i 各自又需构造两个图)，规模为 $\approx n/2$
- 第三层：8 次调用，规模为 $\approx n/(2\sqrt{2})$
- ...
- 第 k 层：2^k 次调用，规模为 $\approx n/((\sqrt{2})^k)$

递归在 $n/((\sqrt{2})^k) \leq 6$ 时停止，解得 $k \approx 2\log n$。
每层的总工作量为 $O(n^2)$，因为：

$$\text{第}k\text{层工作量} = 2^k \cdot O\left(\left(\frac{n}{(\sqrt{2})^k}\right)^2\right) = O(n^2)$$

总共有 $O(\log n)$ 层，因此总运行时间为 $O(n^2 \log n)$。 □

2.3 成功概率分析

首先是前 $n-t$ 次收缩后保留最小割, 然后是两个结果有至少一次成功, 得到成功概率的递归关系为:

$$P(n) \geq \frac{1}{2}\left(1-\left(1-P\left(\left\lceil 1+\frac{n}{\sqrt{2}}\right\rceil\right)\right)^2\right)$$

Theorem 2.2. 改进的 *Karger-Stein* 算法找到最小割的成功概率为 $\Omega\left(\frac{1}{\log n}\right)$。

Proof. 我们通过归纳法证明 $P(n) \geq \frac{1}{c\log n}$，其中 c 是适当选择的常数。

基例：对于小的 n，可以直接计算。

归纳步骤：假设对于所有 $m < n$ 成立，则：

$$\begin{aligned} P(n) &\geq \frac{1}{2}\cdot\left(1-\left(1-\frac{1}{c\log(n/\sqrt{2})}\right)^2\right) \\ &\approx \frac{1}{c(\log n - 0.5)} \quad (a^2-b^2=(a+b)(a-b)) \\ &\geq \frac{1}{c\log n} \quad (\text{当 } c \text{ 足够大}) \end{aligned}$$

因此，通过选择足够大的 c，可以满足归纳假设。 □

2.4 进一步改进

通过运行算法 $O(\log n)$ 次并取最佳结果，我们可以将成功概率提高到常数级别，同时保持 $O(n^2\log^2 n)$ 的总运行时间。

Corollary 2.3. 存在一个 $O(n^2\log^2 n)$ 时间的随机算法，能以高常数概率找到图的最小割。