Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Физтех-школа Прикладной Математики и Информатики
Кафедра «Дискретная математика»
Зерцалов Алексей Вячеславович, Б05-923

Поиск наименее плотного разреза

Содержание

1 Постановка задачи

В рамках данного проекта разбирается решение задачи наименее плотного разреза. Если быть более формальными, то требуется разбить вершины графа на 2 группы, так чтобы отношения числа ребер между ними к размеру меньшей группы было как можно меньше.

Необходимо изучить и имплемнтировать алгоритм Лейтона-Рао приближенного решения с точностью $O(log(n))$

2 Общий случай

2.1 Задачи с несколькими товарами

В своей статье Лейтон и Рао определяют и доказывают результаты для более общего случая, нежели в постановке нашей задачи.

2.2 Задачи с несколькими товарами (MFP)

В отличие от привычного определения задачи нахождения потока, рассматриваются задачи с несколькими товарами, которые необходимы доставить из истока s_i в сток t_i. Также, у каждого товара есть спрос D_i, который означает, что необходимо доставить D_i единиц товара.

В связи с определением возникает проблема: а как же определять максимальный поток, учитывая, что источников может быть больше одного? Авторы остановились на определении, старающемся максимизировать общую долю f каждого товара.

Определение 2.1. *Максимальный поток в MFP - это такое максимальное число f, что для каждого i из истока в сток может быть доставлено fD_i единиц товара одновременно со всеми остальными товарами, без превышения пропускных способностей ребер.*

Определение 2.2. *Минимальный разрез в MFP - это такое Φ для графа с множеством вершин V, что:*

$$\Phi = \min_{U \in V} \frac{C(U, \overline{U})}{|U||\overline{U}|}$$

где

$$C(U,\overline{U}) = \sum_{e \in <U,\overline{U}>} C(e)$$

$$D(U,\overline{U}) = \sum_{\{i | s_i \in U \cap t_i \in \overline{U} \cup t_i \in U \cap s_i \in \overline{U}\}} D_i$$

То есть, это минимальное отношение пропускной способности разреза к сумме спроса тех товаров, исток и сток которых лежат по разные стороны от разреза. Нетрудно заметить, что такое определение является обобщением случая, когда исток, сток, и товар один.

Теперь покажем, что максимальный поток всегда не больше минимального разреза. Допустим, $i_1, ..., i_r$ - множество индексов тех товаров, исток и сток которых разделены некоторым разрезом. По определению максимального потока, мы знаем, что

$$\sum_k f D_{i_k} \leqslant C(U,\overline{U})$$

$$\sum_k D_{i_k} = D(U,\overline{U})$$

Откуда следует, что

$$f \leqslant \frac{C(U,\overline{U})}{D(U,\overline{U})}$$

2.3 Равномерный вариант задачи. Результат

Для того, чтобы добиться логарифмического решения, авторы сужают поставленную ранее задачу.

Определение 2.3. *UMFP - сужение задачи MFP, в которой для любой пары вершин существует товар и спрос D_i всех товаров одинаков. Без ограничения общности $D_i = 1$.*

Для такой задачи был получен результат: минимальный разрез в UMFP не больше, чем в $O(log(n))$ раз больше максимального потока. Также существует пример, на котором достигается такая оценка.

Заметим, что при такой постановке задачи D - это произведение мощностей множеств, полученных разрезом (для каждого элемента-истока из одного множества все элементы из другого множества являются стоками):

$$D(U,\overline{U}) = |U||\overline{U}|$$

3 Теорема о минимальном разрезе в UMFP

Далее будет изложено доказтельство того факта, что минимальный разрез максимум в порядка $log(n)$ раз больше, чем максимальный поток. То есть, будет доказано, что:

$$\frac{C(U,\overline{U})}{|U||\overline{U}|} \leqslant O(flog(n))$$

где f - максимальный поток в графе.

Учитывая предыдущие результаты, сформулируем главную теорему:

Теорема 3.1. *Для любого UMFP, верно следующее:*

$$\Omega(\frac{\Phi}{log(n)}) \leqslant f \leqslant \Phi$$

Авторы статьи при доказательстве теоремы ссылаются на другой результат: на задачу, двойственную к MFP. А именно, это задача линейного программирования, где требуется найти такие расстояния $d(u,v)$, что

$$\sum_i D_i \cdot d(s_i,t_i) \geqslant 1$$

При этом следующая сумма минимизирована:

$$\sum_e C(e) \cdot d(e)$$

В частном случае UMFP:

$$\sum_{u,v} d(u,v) \geqslant 1$$

$$W = \sum_e C(e) \cdot d(e)$$

Оказывается, из теории линейного программирования следует, что $f = W$. Таким образом, мы свели часть задачи к решению задачи линейного программирования. Как известно, существуют полиномиальные решения таких задач. Дальнейшие выкладки помогут построить минимальный разрез с помощью W.

Лемма 3.1. *Для любого графа G, с произвольными пропускными способностями ребер и функцией расстояния (с суммарным весом W), а также для любого $\delta > 0$, существует такое разбиение графа на компоненты радиуса не больше δ, так что пропускная способность ребер, соединяющих разные компоненты, будет не больше $4W \cdot log(\frac{n}{\delta})$*

Доказательство 3.1.1. *Пусть $C = \sum\limits_{e \in E} C(e)$ - суммарная пропускная способность графа G. Если $\delta \leqslant 4W \cdot log(\frac{n}{C})$, мы можем выделить каждую вершину в отдельную компоненту. Тогда пропускная способность между разными компонентами $\leqslant C \leqslant 4W \log(\frac{n}{\delta})$.*

Рассмотрим случай $\delta > 4W \cdot log(\frac{n}{C})$. Для дальнейших рассуждений построим новый граф G^+ следующим образом: заменим каждое ребро в G на цепочку из $\lceil \frac{Cd(e)}{W} \rceil$ ребер. Пропускную способность каждоого такого ребра сделаем равной $C(e)$, а расстояние 1. Далее будет показано, как сформировать нужные нам компоненты в G^+ и выделить их в исходном графе.

Алгоритм построения компонент: Сначала выбираем произвольную вершину v, соответсвующую какой-то вершине из G. Определим G_i^+ - подграф G^+, содержащий все вершины и ребра на расстоянии не более i от вершины v. Нетрудно заметить, что, так как длины ребер в G^+ равны 1, то расстояние между двумя врешинами в нем это просто кол-во ребер в кратчайшем пути между ними. Так обозначим за $C_0 = \frac{2C}{n}$, а для $i > 0$ обозначим за $C_i = \sum\limits_{e \in G_i^+} C(e)$. Найдем такое наименьшее j, что $C_{j+1} < (1+\varepsilon)C_j$, $\varepsilon = Wlog(\frac{n}{\delta C})$. Такое j точно найдется, поскольку с какого-то момента $C_i = C_{i+1}$, так как граф конечен. Найденный подграф G_j^+ формирует первую компоненту. Остальные компоненты найдем рекурсивно, удаляя ребра и вершины от предыдущей компоненты из графа G^+, пока не исчерпаем все такие v, которые лежат и в исходном графе G.

Пусть $C^+ = \sum\limits_{e \in G^+} C(e)$. Тогда, согласно определению G^+:

$$C^+ = \sum_{e \in E} C(e) \lceil \frac{Cd(e)}{W} \rceil \leqslant \sum_{e \in E} C(e)(1 + \frac{Cd(e)}{W}) = C + \frac{C}{W} \sum_{e \in E} C(e)d(e) = C + C = 2C$$

Пропускная способность ребер, выходящих из любой компоненты C_i, согласно неравенству $C_{j+1} < (1+\varepsilon)C_j$:

$$C_{i+1} - C_i < \varepsilon C_i$$

Посчитаем суммарную пропускную способность ребер, выходящих из компонент. Мы имеем 2 вида компонент: C_i, когда $i \neq 0$, и C_0. Во втором случае компонента просто состоит из 1 вершины, значит, таких может быть не больше n. В первом случае - суммарно они не больше C^+ по определению:

$$\varepsilon(\sum_{i \geqslant 1} C_i + nC_0) \leqslant \varepsilon C^+ + \varepsilon n C_0 \leqslant 2\varepsilon C + 2\varepsilon C = 4\varepsilon C$$

Теперь построим компоненты в G. Поместим все вершины из G, лежащие в одной компоненте в G^+, в одну компоненту в G. Тогда, заметим, что пропускная способность ребер, выходящих из компонент одинакова для обоих графов (и там и там разрезается ребро e, причем в G^+, разрезается «подребро» на нем, которое по построению имеет ту же пропускную способность $C(e)$. Значит, суммарная пропускная способность ребер в G, выходящих из компонент, такая же: $4\varepsilon C \leqslant 4W \cdot log(\frac{n}{\delta})$. Это доказывает первую часть теоремы.

Осталось показать, что все компоненты радиуса не больше δ.

Зафиксируем j и рассмотрим компоненту с этим радиусом в G^+. Исходя из способа построения компонент, понятно, что пропускная способность ребер в компоненте будет не меньше $(1+\varepsilon)^j C_0 = (1+\varepsilon)^j \frac{2C}{n}$. Поскольку пропускная способность всего графа, как мы доказали выше, не больше $2C$, можно составить и рещить следующее неравенство (в последнем пользуемся неравенством на логарифм с тем, что $\varepsilon < \frac{1}{4}$):

$$(1+\varepsilon)^j \frac{2C}{n} \leqslant 2C$$

$$j log(1+\varepsilon) \leqslant log(n)$$

$$j \leqslant \frac{log(n)}{log(1+\varepsilon)} \leqslant \frac{log(n)}{\varepsilon}$$

Любой путь в G^+ длиной l соответсвует пути в G длиной не больше $\frac{Wl}{C}$. Объединяя оба неравенства, получаем, что радиус любой компоненты не больше $\frac{W log(n)}{C_\varepsilon} = \delta$, что и требовалось доказать.

Далее сформулируем несколько утверждений, который приведут нас к решению задачи.

Следствие 3.1.1. *Для любого графа G и функции расстояния с суммарным весом W, возможно одно из двух:*

- *Найдется компонента радиуса не больше $\frac{1}{2n^2}$, которая содержит 2/3 вершин графа*

- *Найдется разрез размера $O(W log(n))$*

Лемма 3.2. *Для любого графа G и фуукнции расстояния с суммарным весом W и множества вершин T, содержащем не менее 2/3 врешин графа, такого, что*

$$\sum_{u \in V-T} d(T,u) \geqslant \frac{1}{2n}$$

Существует разрез размером $O(W)$

Лемма 3.3. *Для любого графа G и функции расстояния с суммарным весом W, существует разрез размера $O(W log(n))$*

4 Сведение полученных результатов к задаче из условия

Как можно было увидеть, все это время решалась не та задача: нахождение минимума $\Phi = \min\limits_{U \in V} \frac{C(U,\overline{U})}{|U||\overline{U}|}$, тогда как как в условии нашей задачи стояло нахождение минимума

$$\Phi_1 = \min_{U \subseteq V} \frac{C(U,\overline{U})}{min(|U|,|\overline{U}|}$$

Покажем, что на самом деле, результаты применимы и к нашей постановке.

Заметим, что:

$$\Phi_1 = max(|U|,|\overline{U}|)\Phi$$

И так как $\frac{n}{2} \leqslant max(|U|,|\overline{U}|) \leqslant n$, то

$$\frac{n}{2}\Phi \leqslant \Phi_1 \leqslant n\Phi$$

Откуда

$$\Omega(\frac{\Phi_1}{nlog(n)}) \leqslant f \leqslant \frac{2}{n}\Phi_1$$

5 Описание построенного алгоритма

Соединив все вместе, получается следующий алгоритм:

- Решаем задачу линейного программирования нахождения фукнции расстояний для $C(e) = 1$
- Находим с помощью компоненты для $\delta = \frac{1}{2n^2}$
- Находим либо нужный разрез (и завершаем алгоритм), либо большую компоненту, содержащую 2/3 узлов
- Если разрез еще не найден, то строим разрез $O(W)$

6 Анализ и результаты для частных примеров

Полученный алгоритм был реализован на *Python3*, используя библиотеку *pulp* для линейного программирования.

Ниже приведены примеры некоторых графов и результаты алгоритма на них:

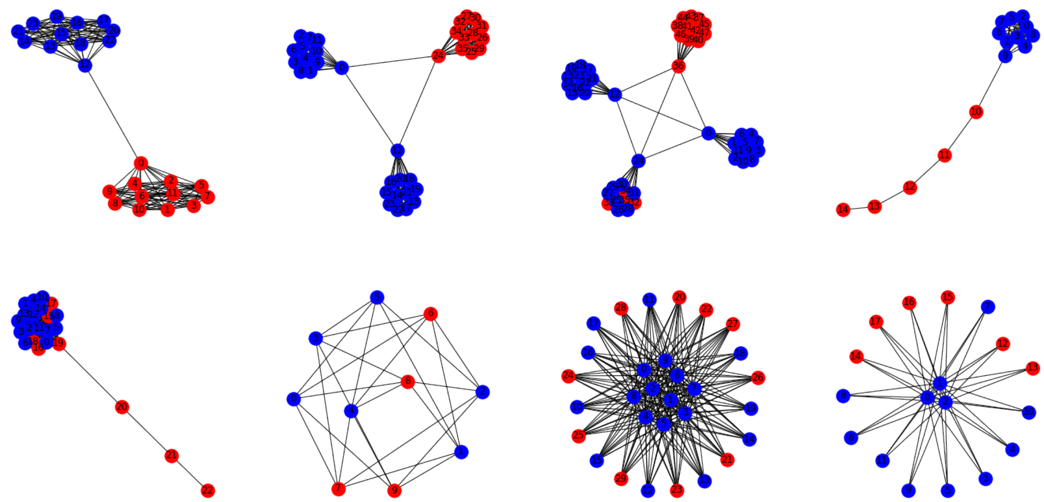

Рис. 1: Тестовые примеры работы алгоритма Лейтона-Рао

Как видно из примеров на "хорошо разделяемых"графах алгоритм выдает отличные результаты, что говорит о правильной реализации алгоритма. Также чем больше вершин в графе, тем более красивые и "визуально правильные"результаты получаются.

7 Литература

- Tom Leighton and Satish Rao. 1999. Multicommodity Max-Flow Min-Cut Theorems and Their Use In Designing Approximation Algorithms