$s+\partial$

论文报告 $(x \mod 4 \equiv 0)$

502022330024 李晨曦

November 22, 2023

引入：Edmonds-Karp 算法

流

给定图 $G(V,E)$ 和源汇点 s,t，一个流 $f: V\times V\to\mathbb{R}$ 应该满足：

$$\forall vw\in E.\ f(v,w)\le c(v,w) \tag{1}$$

$$\forall vw\in E.\ f(v,w)=-f(w,v) \tag{2}$$

$$\sum_{vw\in E} f(v,w)=0 \tag{3}$$

注

上面是本论文中使用的定义，和课程用书中的定义（非负）不一致但等价。

最大流是使得

$$\sum_{vw \in E} |f(v, w)|$$

最大的流。要求解最大流，一个常用的算法是 Edmonds-Karp 算法。

Edmonds-Karp 算法

首先找到一条合法的流，然后尝试使得这条流最大。

对偶（duality）

最大流是使得

$$\sum_{vw \in E} |f(v, w)|$$

最大的流。要求解最大流，一个常用的算法是 Edmonds-Karp 算法。

Edmonds-Karp 算法

首先找到一条合法的流，然后尝试使得这条流最大。

对偶（duality）

首先找到一条最大的流，然后尝试使得这条流合法。

问题

- 存在这样的算法吗?
- 什么是「不合法」的流?
- 在尝试使它合法的过程中，能一直保持最大吗?

Push-relabel 算法

本文提出了一种算法，即 push-relabel 算法。该算法按照与 Edmonds-Karp 算法相对偶的直觉设计并实现。

文章证明了，在朴素实现下，算法的复杂度为 $O(n^2 m)$；在特定的操作顺序下，算法的复杂度为 $O(n^3)$；利用动态树（dynamic tree）方法进行优化后，算法的复杂度可以达到 $O(nm\log(\frac{n^2}{m}))$.

预流（preflow）

预流是一个函数 $f: V \times V \to \mathbb{R}$，它满足 1 和 2. 对于 3，进行如下修改：

$$\sum_{vw \in E} f(v, w) = e(v) \geq 0$$

预流（preflow）

预流是一个函数 $f: V \times V \to \mathbb{R}$，它满足 1 和 2. 对于 3，进行如下修改：

$$\sum_{vw \in E} f(v, w) = e(v) \geq 0$$

理解

从直觉上理解，预流和流的区别在于预流允许流入的流量大于流出的流量。

预流

预流（preflow）

预流是一个函数 $f: V \times V \to \mathbb{R}$，它满足 1 和 2. 对于 3，进行如下修改：

$$\sum_{vw \in E} f(v, w) = e(v) \geq 0$$

理解

从直觉上理解，预流和流的区别在于预流允许流入的流量大于流出的流量。

活跃

一个点 v 在论文中被叫做是活跃（active）的，当且仅当 $e(v) > 0$

剩余网络

如何保证预流是最大的?

剩余网络

如何保证预流是最大的?

剩余网络

类似于书上的概念，在论文的流定义下也可以定义剩余网络 $G_f = (V, E_f)$。边 vw 的剩余容量（residual capacity）为：

$$r_f(v, w) = c(v, w) - f(v, w)$$

易见 $r_f(v, w) \geq 0$. 论文中定义，如果 $r_f(v, w) = 0$，那么 $vw \notin E_f$

定理

类似于书上的定理 7.6 ，有论文中的定理 3.2:
一个流 f 是最大的，当且仅当 G_f 中不含 f 增广路。

剩余网络

定理

类似于书上的定理 7.6 ，有论文中的定理 3.2:
一个流 f 是最大的，当且仅当 G_f 中不含 f 增广路。

只要保证在算法运行时，预流 f 的剩余网络 G_f 不含 f 增广路，那么一定有当算法结束，预流 f 成为合法的流时，f 是最大流。

问题

如何保证 G_f 中不含 f 增广路（即 s 到 t 的有向路）？

标号

问题

如何保证 G_f 中不含 f 增广路（即 s 到 t 的有向路）？

标号（label）

点 v 的标号 $d(v)$ 是一个整数，一个标号是合法（valid）的，当且仅当：

- $d(s) = n$
- $d(t) = 0$
- $\forall vw \in E_f \,.\, d(v) \leq d(w) + 1$

引理 3.3

对于预流 f 与在 G_f 上合法的标号 d，G_f 中不存在一个 $s-t$ 路。

标号

引理 3.3

对于预流 f 与在 G_f 上合法的标号 d，G_f 中不存在一个 $s-t$ 路。

证明

假设存在一条 $s-t$ 路 $p = v_0 v_1 \dots v_n$ ，那么由于对于任何 $v_i \neq t$，$d(v_i) \leq d(v_{i+1}) + 1$，那么对长度进行归纳，易证 $d(t) \geq 1$. 而 $d(t) = 0$，所以显然不成立。

引理 3.3 说明了，只要我们能够一直维护一个合法的标号，那么在算法结束时，流 f 就会是最大流。

算法过程

初始化

- $d(s) = n$
- $d(t) = 0$
- $d(v) = 0, v \neq s, v \neq t$

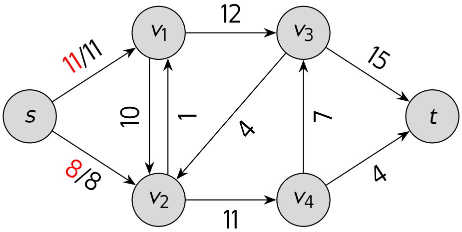
由于把所有的非 s, t 顶点标号都设置为 0，如果 G_f 中还存在 sv 边的话，那么 $d(s) \leq d(v) + 1$ 是肯定不成立的。所以:

$$\forall sv \in E \,.\, f(s, v) = c(s, v)$$

其他情况，$f(v, w) = 0$.

操作

现在我们得到了一个这样的预流：



可以看到，v_1 和 v_2 节点是活跃的。活跃的节点可以进行两个操作，即 *push* 和 *relabel*

Relabel

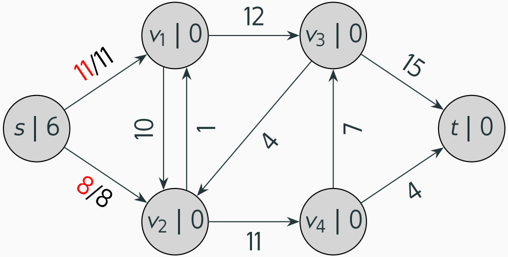
操作条件

v 是活跃的，且 $\forall w \in V\ .\ r_f(v, w) > 0 \to d(v) \le d(w)$

操作过程

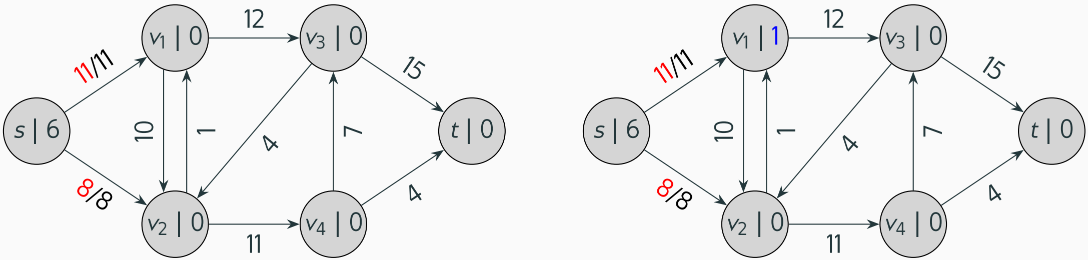
$d(v) := \min\{\ vw \in E \mid d(w) + 1\ \}$

Relabel



$s \mid 6$
$v_1 \mid 0$
$v_2 \mid 0$
$v_3 \mid 0$
$v_4 \mid 0$
$t \mid 0$
11/11
8/8
12
10
1
4
7
15
11
4

Relabel



s | 6
v1 | 0
v2 | 0
v3 | 0
v4 | 0
t | 0
11/11
8/8
12
10
1
4
7
15
4
11
s | 6
v1 | 1
v2 | 0
v3 | 0
v4 | 0
t | 0
11/11
8/8
12
10
1
4
7
15
4
11

Push

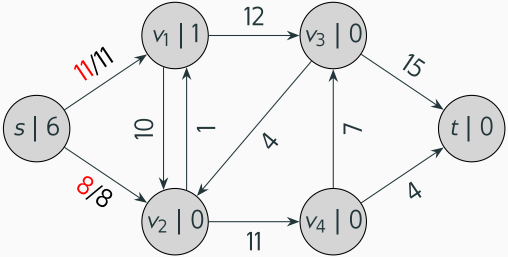
操作条件

v 是活跃的，$r_f(v, w) > 0$ 且 $d(v) = d(w) + 1$

操作过程

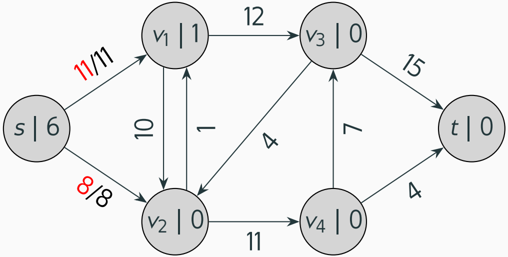
- $\delta = \min(e(v), r_f(v, w))$
- $f(v, w) := f(v, w) + \delta$
- $f(w, v) := f(w, v) - \delta$

Push

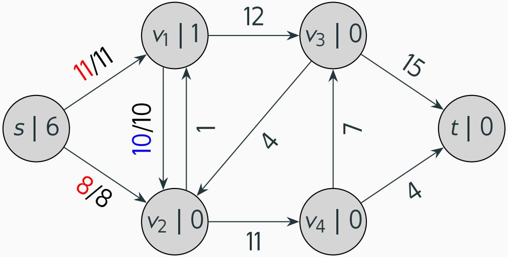


$s \mid 6$
$v_1 \mid 1$
$v_2 \mid 0$
$v_3 \mid 0$
$v_4 \mid 0$
$t \mid 0$
11/11
8/8
12
10
1
4
7
15
11
4

Push



s | 6
v1 | 1
v2 | 0
v3 | 0
v4 | 0
t | 0
11/11
8/8
10
1
12
4
7
15
11
4



s | 6
v1 | 1
v2 | 0
v3 | 0
v4 | 0
t | 0
11/11
8/8
10/10
1
12
4
7
15
11
4

不动点

算法的过程可以描述为：对预流 f 不断进行 push 和 relabel 操作，直到达到不动点。（即不能再进行 push 和 relabel 操作）

直观理解

这个算法的过程就是先把源点到周围顶点的流都流满，之后不断 push 和 relabel，一开始超过最大流的流量最后会流回源点。

算法性质

正确性

引理 3.1

对于任意顶点 v，如果它是活跃的，那么它要么可以进行 push 操作，要么可以进行 relabel 操作

引理 3.1 的逆否命题即证明了算法的正确性：在算法结束时，任何顶点 v 都满足 $e(v) = 0$，这样以来预流 f 就是真正的流 f. （最大已经得到保证）

停机性 $\wedge$ 时间复杂度

引理 3.7

在算法的任意时刻，对任意顶点 v, $d(v) \leq 2n-1$

引理 3.10

push 操作次数的上限为 $4n^2m$

定理 3.11

算法的时间复杂度为 $O(n^2m)$

算法优化

各种优化

- 通过先入先出的操作顺序，可以获得 $O(n^3)$ 复杂度（本文 Section 4.）
- 通过动态树，可以获得 $O(nm\log(\frac{n^2}{m}))$ 复杂度（本文 Section 5.）
- 通过每次都选择 $d(v)$ 最大的节点，可以获得 $O(n^2\sqrt{m})$ 复杂度

参考文献

Ravindra K. Ahuja, Murali S. Kodialam, Ajay K. Mishra, and James B. Orlin.
Computational investigations of maximum flow algorithms.
European Journal of Operational Research, 97:509–542, 1997.

Andrew V. Goldberg and Robert E. Tarjan.
A new approach to the maximum-flow problem.
J. ACM, 35(4):921–940, oct 1988.