Trabalho Prático - Push Relabel

DCC199 Teoria dos Grafos

Yuri Diego Santos Niitsuma
Breno Rodrigues Marques da Silva

Introdução

Um dos problemas mais conhecidos de otimização é o problema do fluxo máximo. Ele é um caso especial de um problema de circulação, onde deseja-se encontrar o maior fluxo possível de uma fonte s para um sumidouro t.

Um dos primeiros algoritmos propostos para a solução desse problema foi o Ford-Fulkerson. Esse método utiliza uma estratégia gulosa para procurar um caminho da fonte ao sumidouro pelo grafo residual. Uma vez identificado, um fluxo igual à menor capacidade das arestas é enviado por ele. Uma implementação específica do Ford-Fulkerson é o algoritmo de Edmonds-Karp, que define o método de busca de caminhos através de uma busca em largura.

Esse trabalho por sua vez fez uma implementação do algoritmo de push-relabel. Em comparação aos descritos anteriormente, esse método desloca o fluxo de vértice em vértice, identificando qual o excesso de fluxo em cada um, e movimentando esse excesso para os adjacentes. Para isso, um grafo dirigido modela o problema:

Seja $G = (V, A)$ um grafo com $s, t \in V$ respectivamente a fonte e o sumidouro do problema.

A capacidade de uma aresta $(u, v) \in E$ é denominada $c(u, v) : E \to \mathbb{Z} | c(u, v) > 0$ e representa o fluxo máximo que pode passar por ela.

O fluxo de uma aresta $(u, v) \in E$ é denominado $f(u, v) : E \to \mathbb{Z} | f(u, v) > 0$. Ele possui as seguintes restrições:

- Restrição de capacidade: $f(u, v) \leq c(u, v)$, ou seja, o fluxo de uma aresta não pode exceder a sua capacidade
- Conservação de fluxo: $\sum_{u:(u,v)\in E} f(u, v) = \sum_{u:(v,u)\in E} f(v, u)$, $\forall v \in V \setminus \{s, t\}$, ou seja, o fluxo total que entra em um vértice deve ser igual ao fluxo total que sai dele, exceto para a fonte e para o sumidouro.

O valor do fluxo f é definido como $f = \sum_{v:(s,v)\in E} f(s, v)$, onde s é a fonte de G. Ele representa a quantidade de fluxo total que passa da fonte para o sumidouro.

O problema de fluxo máximo é maximizar f, ou seja, achar o maior fluxo possível de s para t.

Push-Relabel

O algoritmo de *push-relabel* pode ser visualizado através de uma analogia: as arestas são canos de água e os vértices são suas junções, cada uma a uma altura diferente. A fonte é considerada como a junção mais alta e o sumidouro a mais baixa. Cada junção manda água para as adjacentes. Uma vez que alguma junção possua um excesso de água, ela envia (operação de *push*) água para alguma outra de altura menor. Se a água ficar presa em alguma junção, a sua altura deve ser aumentada (operação de *relabel*).

Dessa forma, duas variáveis são associadas a cada vértice: a sua altura, que é usada para determinar se ele pode enviar fluxo para algum dos seus adjacentes ou não, e o excesso de fluxo, que é a diferença entre o fluxo total que entra no vértice e o fluxo total que sai dele. Abaixo segue o *struct* em C.

```
typedef struct{
  int height, eFlow;
} Vertex;
```

Dado o vértice v, definimos que:

- $height = h(v)$ é altura do vértice
- $eFlow = e(v)$ o excesso de fluxo no vértice.

Dado $u, v \in V - \{s\}$, temos que o excesso de fluxo é

$$e(u) = \sum_{v\ \in\ V} f(v,\ u) - \sum_{v\ \in\ V} f(u,\ v)$$

Mas aplicaremos uma relaxação na restrição de conservação de fluxo.

$$\sum_{v\ \in\ V} f(v,\ u) - \sum_{v\ \in\ V} f(u,\ v) \geq 0 \Rightarrow e(u) \geq 0$$

O que nos dará mais flexibilidade pois nos dara a liberdade de aumentar o fluxo olhando apenas em um vértice e suas arestas adjacentes.

Overflowing é definido quando $e(u) > 0$.

Cada aresta também tem das variáveis associadas: o fluxo, que indica a quantidade atual de fluxo que passa por ela, e a capacidade, que indica qual o valor máximo de fluxo na aresta.

```
typedef struct{
  int flow, capacity;
  int u, v;
}Edge;
```

Dado dois vértices (u, v), definimos que:

- $flow = f(u, v)$ é o fluxo na aresta
- $capacity = c_f(u, v)$ a capacidade máxima de fluxo

Como consequência, $f(u, v) \leq c_f(u, v)$.

O algoritmo conta com três operações principais: o *preflow*, o *push* e o *relabel*.

preflow

O *preflow* inicializa os valores do grafo para que as demais operações possam ser feitas com sucesso. A altura e o excesso de fluxo de cada vértice é inicializado com 0, já que nenhum deles possui fluxo no início de execução. Ele define a altura da fonte igual ao número de vértices no grafo, já que ela deve possuir a maior altura da rede.

$$h(u) = \begin{cases} |V| & \text{if} u = s \\ 0 & \forall u \neq s \end{cases}$$

O fluxo de cada aresta é inicializado com 0, já que não há fluxo em nenhuma delas. Finalmente, são identificados os vértices adjacentes à fonte. Nas arestas que os ligam, o fluxo recebe o valor da sua capacidade, e o vértice recebe um excesso de fluxo com o mesmo valor.

$$f(u, v) = \begin{cases} c(u, v) & \text{if} u = s \\ 0 & \forall u \neq s \end{cases}$$

push

A operação de *push* é utilizada para retirar fluxo de algum vértice com excesso. Se houver algum vértice adjacente com altura menor, o fluxo é deslocado dele para o adjacente. O valor deslocado é definido pelo menor valor entre o excesso do vértice e a capacidade restante da aresta.

$\Delta = \min(e(u),\ c_f(u,v))$
if $(u,v) \in E$
then $f(u,v) = f(u,v) + \Delta$
else $f(u,v) = f(u,v) - \Delta$
endif
$e(u) = e(u) - \Delta$
$e(v) = e(v) + \Delta$

A condição $e(u) \geq 0$ sempre se mantém .

relabel

O *relabel* é utilizado quando algum vértice possuir um excesso de fluxo mas nenhum dos seus adjacentes tiver uma menor altura. Nesse caso devemos aumentar a sua altura. Seu novo valor é dado pela menor altura adjacente, somada a um.

$h(u) = 1 + min\{h(v) : (u,v) \in E_f\}$

O algoritmo chega ao fim quando não houverem mais vértices com excesso de fluxo. A distribuição corrente do fluxo será a distribuição otimizada para o problema. O valor do fluxo máximo é dado pelo valor de excesso de fluxo na fonte.

Corretude

Deve-se provar que se o algoritmo terminar, o pré-fluxo f é fluxo máximo.

1. Sabemos que a altura $h\,(u) \mid u \in V$ nunca diminui e que ocorre um relabel no vértice u, a sua altura $h(u)$ aumenta em pelo menos 1.
   Demonstração: Se u sofre um *relabel*, então para todo vértice v tal que $(u,v) \in E_f$ temos que $h\,(u) \leq h\,(v)$. Logo, $h\,(u) \leq min\{h\,(v) \,|\, (u,v) \in E_f\}$. Portanto o relabel aumenta o valor de $h(u)$.

2. Seja $G = (V,E)$ um fluxo com fonte s e sumidouro t. Durante a execução do algoritmo, h se mantém como uma função de altura.
   Demonstração: Por indução: em uma aresta residual $(u,v) \in E_f$, após o relabel temos que $h\,(u) \leq h\,(v) + 1$. Considerando outra aresta residual, esta (w,u), por **(1)**, $h\,(w) \leq h\,(u) + 1$ antes do relabel implica em $h(w) < h(u)$ após a operação. Logo, no relabel h é uma função de altura. Considerando a operação de push, ela pode adicionar uma aresta (v,u) ou remover uma aresta (u,v). No primeiro caso, $h\,(v) \; = \; h\,(u) - 1 < h\,(u) + 1$, e a variável h continua uma função de altura. No segundo caso, essa restrição é removida, e a variável h ainda continua uma função de altura.

3. Seja $G = (V,E)$ um fluxo em rede com fonte s e sumidouro t, seja f um pré-fluxo em G e seja h uma função de altura em V. Então não existe nenhum caminho de s até t na rede residual G_f.
   Demonstração: Por contradição, suponha que exista um caminho $p = \{s,\ v_0,\ v_1, \ldots\ ,\ \ v_j, t\ \}$. Já que p é um caminho simples, logo $j < |V|$. Com $i = \{0, 1, \ldots, j-1\}$, temos a aresta $(v_i,\ v_{i+1})$. Como h é uma função de altura, temos que $h\,(v_i) \leq h\,(v_{i+1}) + 1$ com $i = 0, 1, \ldots, j-1$. Aplicando essa desigualdade sobre o caminho p, temos $h\,(s) \leq h\,(t)$. Porém, como $h(t) = 0$, temos que $h\,(v_i) \leq \ j < |V|$, o que contradiz a restrição de $h\,(s) = |V|$ na função de altura. Logo, o caminho p não existe.

4. Se o algoritmo terminar ao ser executado sobre um fluxo de rede $G = (V,E)$ com fonte s e sumidouro t, então o pré-fluxo f que ele calcula será um fluxo máximo para G.

Demonstração: A inicialização por *preflow* faz com que f seja um pré-fluxo. As únicas operações durante a execução são as de *push* e *relabel*. Se f for um pré-fluxo antes das operações de *push* ou *relabel*, então ainda será um pré-fluxo depois. Ao término do algoritmo, cada vértice em $V - \{s, t\}$ deve ter um excesso de 0, pois caso contrário ele seria selecionado para *push* e como f é um pré-fluxo, não há nenhum vértice com *overflow*. Logo, f é um fluxo. Como h é uma função de altura, *(3)* nos diz que não existe nenhum caminho residual de s até t na rede residual G_f. Logo, pelo teorema do fluxo máximo e corte mínimo, f é um fluxo máximo.

5. Seja $G = (V, E)$ um fluxo de rede com fonte s e sumidouro t, e seja f um pré-fluxo em G. Então, para qualquer $x \in G$, existe um caminho de x para s no grafo residual G_f.
Demonstração: Para um vértice x tal que $e(x) > 0$, seja U o conjunto contendo vértice v tal que existe um caminho em G_f, e suponha por contradição que $s \notin U$.

$$\begin{aligned}
\sum_{u\in U} e(u) &= \sum_{u\in U}\left(\sum_{v\in V} f(v,u) - \sum_{v\in\{V-U\}} f(u,v)\right)\\
&= \sum_{u\in U}\left(\left(\sum_{v\in U} f(v,u) + \sum_{v\in\{V-U\}} f(v,u)\right) - \left(\sum_{v\in U} f(u,v) + \sum_{v\in\{V-U\}} f(u,v)\right)\right)\\
&= \sum_{u\in U}\sum_{v\in U} f(v,u) + \sum_{u\in U}\sum_{v\in\{V-U\}} f(v,u) - \sum_{u\in U}\sum_{v\in U} f(u,v) - \sum_{u\in U}\sum_{v\in\{V-U\}} f(u,v)\\
&= \sum_{u\in U}\sum_{v\in\{V-U\}} f(v,u) - \sum_{u\in U}\sum_{v\in\{V-U\}} f(u,v)
\end{aligned}$$

Como $e(x) > 0, x \in U$, então $\sum_{u\in U} e(u) > 0$, o que nos leva a:

$$\sum_{u\in U}\sum_{v\in\{V-U\}} f(v,u) - \sum_{u\in U}\sum_{v\in\{V-U\}} f(u,v) > 0 \Rightarrow \sum_{u\in U}\sum_{v\in\{V-U\}} f(v,u) > 0$$

Então deve existir pelo menos um par de vértices tal que $f(u', v') > 0$, então existe uma aresta residual (u', v'), o que significa que também existe um caminho $x \to u'$ e depois pra v', o que contradiz a definição do conjunto U.

6. Seja $G = (V, E)$; um fluxo de rede com fonte s e sumidouro t. Durante a execução Push-Relabel em G, temos $h(u) \le 2|V| - 1 \ \ \forall u \in V$.
Demonstração: A altura da fonte s e do sumidouro t nunca muda porque estes vértices são por definição $e(s), e(t) \le 0$. Logo, sempre teremos $h(s) = |V|$ e $h(t) = 0$, e ambos não são maiores que $2|V| - 1$.
Agora considere qualquer vértice $u \in V - s, t$. Inicialmente temos $h(u) = 0$. Iremos mostrar que a cada *relabel*, ainda teremos $h(u) \le 2|V| - 1$. Quando u passa pelo processo de *relabel*, $e(u) > 0$, e pelo lema anterior, existe um caminho simples até s no grafo residual. Seja p um vetor contendo os vértices no caminho de u até s, temos que p tem dimensão menor que $|V| - 1$ pois é caminho simples e representaremos $p = (v_0, v_1, v_k)$. Para cada $(v_i, v_{i+1}) \in E_f$, e pelo lema sobre as alturas, $h(v_i) \le h(v_{i+1}) + 1$, expandindo estas inequações de v_0 k vezes, temos:
$h(u) = h(v_0) \le h(v_k) + k \le h(s) + (|V| - 1) = 2|V| - 1$.

Complexidade

O *relabel* só é executado até $(2|V| - 1)(|V| - 2)$ vezes. Já que a altura $h(u)$ não pode ser diminuída, o seu valor máximo é de $2|V| - 1$ para cada vértice. Nota-se que o *relabel* só pode ser aplicado a até $|V| - 2$ vértices. Logo, o *relabel* tem complexidade de $O\left(|V|^2\right)$.

A operação de push deve ser divida em duas: os *pushs* que saturam o grafo residual e aqueles que não o saturam. No primeiro tipo o arco (u, v) por onde a operação é feita é removido de G_f. Ele só poderá ser inserido novamente se houver um *relabel* sobre o vértice v, deve haver um push por (v, u) e então u deverá sofrer um *relabel* também. Nesse processo, a altura $h(u)$ aumenta em pelo menos dois. Logo, existem menos de $2|V|$ *pushs* saturantes entre u e v. Multiplicado ao número de arestas onde isso pode ocorrer, temos até $2|V||E|$ ocorrências no grafo, com complexidade $O(|V||E|)$.

A seguir devemos encontrar a quantidade de *pushs* não saturantes. Isso será feito pelo método do potencial, com função $\Phi = \sum_{[u \in V,\ e(u)>0]} h(u)$. O *relabel* de um vértice u aumenta o valor de Φ em menos de $2|V|$, pois o conjunto sobre o qual a soma é tomada se mantém o mesmo. Em um push saturante de um vértice u até um v, o valor de Φ aumenta em menos de $2|V|$, pois nenhuma altura muda e só o vértice v pode ganhar algum *overflow*.

Após um *push* não saturante de u para v, u, que tinha *overflow*, não tem nenhum excesso de fluxo mais, enquanto v deve ter algum, a não ser que seja a fonte. Logo, a função Φ diminuiu em $h(u)$ e aumentou por 0 ou por $h(v)$. Como $h(u) - h(v) = 1$, o resultado é que a função potencial diminuiu em pelo menos 1. Portanto, a quantidade de aumento em Φ é limitada por $(2|V|-1)(|V|-2) + (2|V|-1)(2|V||E|) \leq 4|V|^2|E|$. Logo, a complexidade de um push não saturante é de $O\left(|V|^2|E|\right)$

Portanto, a complexidade do algoritmo assume o valor mais alto: $O\left(|V|^2|E|\right)$.

Estruturas dos dados

A estrutura do grafo foi implementada através de uma lista de arestas, já que o algoritmo busca arestas específicas.

O código fonte está disponível em:

- https://github.com/ignitz/dcc199_tp_final/blob/master/src/main.c

Testes

Foi criado grafos conexos aleatórios de tamanhos variados para efetuarmos testes de execução.

Os testes de encontram em:

- https://github.com/ignitz/dcc199_tp_final/tree/master/src/tests

Seguem as saídas no saida de fluxo máximo e tempo de execução em segundos.

$|V| = 10$

```
318
.001270531
```

```
571
.001553841
```

$|V| = 100$

```
322
.005501648
```

```
319
.063763368
```

```
91
.002576342
```

```
386
.055882479
```

$|V| = 200$

```
451
.505722545
```

```
581
.491276133
```

```
393
.575241899
```

```
243
.566083337
```

$|V| = 1000$

```
887
.355961284
```

```
167
.236621816
```

```
918
.405924430
```

```
721
71.945276677
```

$|V| = 2000$

```
953
587.119914111
```

$|V| = 3000$

```
255
3.961093684
```

Observe que nos dois últimos há tempos inversos do que esperado, a característica em que o grafo gerado pode influênciar na velocidade do algoritmo. Por exemplo, se o diâmetro entre a fonte e o sumidouro for pequeno.

Referências

- https://mitpress.mit.edu/books/introduction-algorithms