PUSH-RELABEL

Bernardo Amorim, UFMG
Kaio Vieira, UFMG

Resumo
O algoritmo de Push-Relabel foi proposto por Andrew Goldberg e Robert Tarjan em 1988 [4] e se propõe a resolver o problema de fluxo máximo. Neste artigo apresentamos esse algoritmo e demonstramos sua corretude e complexidade.

1. Intuição

Os algoritmos de fluxo mais populares, como os algoritmos de Edmonds-Karp [3] e de Dinitz [2], são baseados em iterativamente encontrar um caminho aumentante da fonte para o sumidouro e aumentar o fluxo que passa por esse caminho, se aproveitando do seguinte lema:

Lema 1. *Um fluxo é máximo se e somente se não existe caminho aumentante da fonte até o sumidouro no grafo residual.*

Desta forma, estes algoritmos começam em uma configuração em que nenhuma unidade de fluxo sai da fonte e aumentam essa quantidade por meio de caminhos aumentantes até que não seja mais possível encontrá-los, mantendo a invariante de que o fluxo é válido a cada iteração do algoritmo.

O algoritmo de Push-Relabel, por outro lado, começa em uma configuração em que todas as arestas saindo da fonte foram saturadas, passando o máximo de fluxo possível da fonte para seus vizinhos. Entretanto, apesar dessa atribuição não representar um fluxo válido, não existe caminho da fonte até o sumidouro no grafo residual e essa invariante é mantida a cada iteração do algoritmo, através de operações que retornam unidades de fluxo excedente de volta para a fonte, terminando assim que o fluxo se tornar válido.

2. Definições

Utilizaremos a seguinte terminologia:

- $G(V, E)$ = grafo de entrada onde $n = |V|$, $m = |E|$
- $G_f(V_f, E_f)$ = grafo residual
- $f(u, v)$ = fluxo de u para v em G_f, $f(u, v) = -f(v, u)$
- $c(u, v)$ = capacidade da aresta
- $c_f(u, v)$ = capacidade da aresta no grafo residual
- s = fonte, t = sumidouro

A configuração da rede residual mantida a cada passo do algoritmo é formalizada pelo conceito a seguir.

Definição 1 (Pré-fluxo). *Um pré-fluxo é uma função f das arestas para fluxo tal que:*
- *$f(u, v) \leq c(u, v)$ — o pré-fluxo não excede a capacidade da aresta.*
- *Para todo vértice $v \neq s$:*

  fluxo entrando em $v \geq$ fluxo saindo de v

Desta forma, um pré-fluxo é basicamente um fluxo onde não é necessário que tudo que entre em um vértice saia, o que motiva a nossa próxima definição:

Definição 2 *(Excesso). O excesso $e(v)$ de um vértice v é a diferença entre o quanto de fluxo entra no vértice e o quanto sai. Note que o excesso de um vértice que não é a fonte é sempre não negativo.*

Além do que foi definido acima, a noção de uma altura definida para cada vértice é uma ferramenta importante para guiar as operações a serem feitas.

Definição 3 *(Altura). Uma altura inteira não negativa $h(v)$ é associada a cada vértice v e as seguintes invariantes serão respeitadas durante o algoritmo:*
1. $h(s) = n$
2. $h(t) = 0$
3. *Para toda aresta $(u, v) \in G_f$ tal que $c_f(u, v) > 0$, $h(u) \leq h(v) + 1$ i.e. arestas no grafo residual sobem, permanecem retas ou descem uma unidade.*

3. Pseudocódigo

O algoritmo pode ser descrito por duas simples operações, PUSH (1) que nos permite empurrar fluxo através de uma aresta, e RELABEL (2) que nos permite mudar a altura de um vértice para tentar passar fluxo por ele no futuro.

Algoritmo 1 PUSH

função PUSH(u, v)
  $\Delta \leftarrow \min(e(u), c_f(u, v))$
  Empurre Δ unidades de fluxo por (u, v)
  retorne

Algoritmo 2 RELABEL

função RELABEL(v)
  $h(v) \leftarrow h(v) + 1$
  retorne

Para garantir as invariantes (as da altura e da não existência de caminho $s - t$ em G_f), inicia-se o algoritmo com o procedimento de inicialização em Algoritmo 3.

Algoritmo 3 INICIALIZAÇÃO

função INICIALIZAÇÃO
  # garantindo as 3 invariantes da altura
  $h(s) \leftarrow n$
  para $\forall v \neq s$ **faça**
    $h(v) \leftarrow 0$
    Empurre $c(s, v)$ unidades de fluxo por (s, v)
retorne

A altura de cada vértice é mantida com o intuito de proibir que operações de PUSH sejam feitas de maneira indiscriminada, para que não se mova fluxo em um ciclo. Por isso só faremos PUSH quando a aresta em questão "abaixar" uma unidade de altura. Ao não conseguir fazer PUSH partindo de um vértice com excesso, fazemos RELABEL neste. Desta forma, o pseudocódigo para o algoritmo completo segue em 4.

Algoritmo 4 PUSH–RELABEL

INICIALIZAÇÃO()
enquanto $\exists v \notin \{s, t\}$ tal que $e(v) > 0$ **faça**
  Escolha algum desses v's
  se $\exists (v, u) \in E_f$ tal que $c_f(v, u) > 0$ e $h(v) = h(u) + 1$
então
    PUSH(v, u)
  senão
    RELABEL(v)

A Figura 1 mostra um exemplo das operações realizadas pelo algoritmo. Após o último estado ilustrado, o algoritmo ainda realizaria três operações de RELABEL no vértice a, elevando sua altura de 1 para 4, e uma operação PUSH(a, s), devolvendo as duas unidades de excesso à fonte e encerrando sua execução.

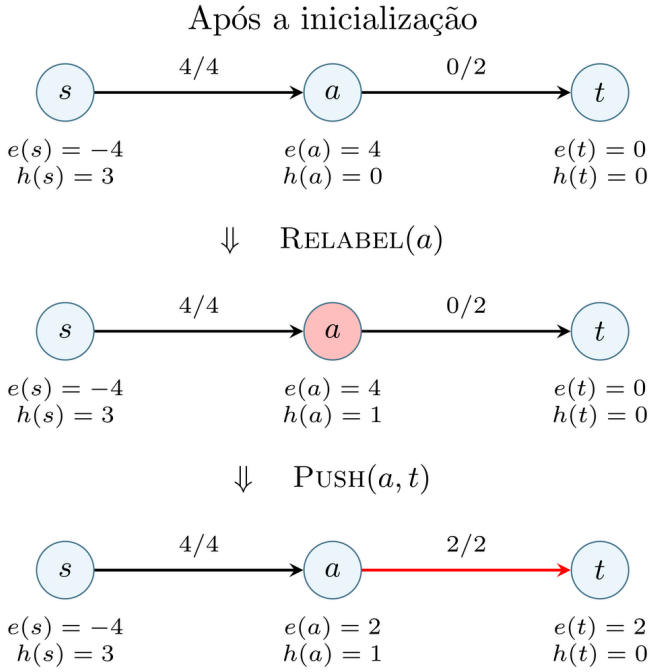

FIGURA 1: EXEMPLO DAS OPERAÇÕES RELABEL E PUSH. ABAIXO DE CADA VÉRTICE SÃO MOSTRADOS SEU EXCESSO E SUA ALTURA; EM CADA ARESTA, O RÓTULO INDICA FLUXO/CAPACIDADE.

4. Corretude

É fácil ver que as operações descritas mantêm as invariantes 1, 2 e 3. Agora demonstramos que a invariante principal é mantida:

Lema 2. *Não existe caminho $s - t$ em G_f em qualquer momento do algoritmo.*

Demonstração. Suponha a fim de contradição que esse caminho $s = v_0, v_1, \ldots, v_l = t$ existe. Então $l < |V|$. Como respeitamos a terceira invariante da altura, $h(v_i) \leq h(v_{i+1}) + 1$, desta forma temos $h(s) \leq h(t) + l < n$, o que contradiz que $h(s) = n$.

Note agora que o algoritmo só para se nenhum vértice de $V \setminus \{s, t\}$ possuir excesso. Perceba também que nessa configuração o pré-fluxo é um fluxo. Desta forma, como consequência direta dos Lemas 1 e 2, para provar que a resposta do algoritmo é ótima basta demonstrar que este termina.

5. Terminação e complexidade

Primeiro, demonstremos o seguinte lema auxiliar que, apesar de exigir alguma sofisticação técnica diz algo muito simples: se um vértice possui excesso, dado que esse excesso veio da fonte por algum caminho, deve haver alguma forma de devolvê-lo para a fonte.

Lema 3. Em qualquer momento do algoritmo, se v é tal que $e(v) > 0$, então existe caminho $v - s$ em G_f.

Demonstração. Seja S o conjunto de vértices alcançáveis por v em G_f e suponha por contradição que $s \notin S$. A escolha de S implica que, para todo u, w onde $u \in \bar{S}$, $w \in S$ temos $f(u, w) \leq 0$. Logo:

$$\begin{aligned}\sum_{w \in S} e(w) &\leq \sum_{u \in V, w \in S} f(u, w) \\ &= \sum_{u \in \overline{S}, w \in S} f(u, w) + \sum_{u, w \in S} f(u, w) \\ &= \sum_{u \in \overline{S}, w \in S} f(u, w) \leq 0 \qquad (\text{antissimetria})\end{aligned}$$

Como o excesso é não-negativo, temos que $e(v) = 0$, contradição.

Agora limitaremos o número de RELABELS que podem ser realizados durante o algoritmo.

Lema 4 (Número de Relabels). *A todo momento do algoritmo, para todo vértice v vale $h(v) \leq 2n - 1$, logo o número de* RELABELS *do algoritmo é da ordem de $O(n^2)$.*

Demonstração. O lema é trivial para s e t. Considere $v \in V \setminus \{s, t\}$. Como o algoritmo só muda as alturas de vértices ativos ($e > 0$), é suficiente provar o Lema apenas para esses vértices ativos. Se v é ativo então pelo Lema 3 existe caminho $v = v_0, v_1, \ldots, v_l = s$ em G_f onde $l \leq n - 1$. Pela terceira invariante da altura temos que $h(v_i) \leq h(v_{i+1}) + 1$. Logo temos $h(v) \leq h(s) + l \leq h(s) + (n - 1) = 2n - 1$.

Para analisar o número de PUSHES os dividiremos em duas categorias segundo a seguinte definição:

Definição 4. *Uma operação* PUSH(u, v) *é dita ser saturante se a quantidade de fluxo enviada satura a aresta* (u, v).

Lema 5. *O número de* PUSHES *saturantes do algoritmo é da ordem de $O(nm)$.*

*Demonstração***.** Entre dois PUSHES saturantes na mesma aresta (u, v) é preciso que um dos vértices sofra RELABEL pelo menos duas vezes pela condição da altura imposta no PUSH. Como cada vértice só pode ser parte de $O(n)$ RELABELS de acordo com o Lema 4, cada aresta só sofre $O(n)$ PUSHES saturantes.

Lema 6. *O número de* PUSHES *não saturantes do algoritmo é da ordem de $O(n^2m)$.*

Demonstração. Tome $\Phi = \sum_{v \text{ ativo}} h(v)$. Todo PUSH não saturante diminui Φ por pelo menos 1, uma vez que ele desativa o vértice em questão. Já uma operação de PUSH saturante faz com que Φ cresça por no máximo $2n - 1$, uma vez que ativa no máximo um vértice e cuja altura máxima é $2n - 1$ pelo Lema 4. Uma operação de relabel aumenta Φ em exatamente 1. Como Φ se inicia em zero, é fácil ver que o número de PUSHES não saturantes é limitado por $O(n^2m)$ como consequência dos Lemas 4 e 5.

Teorema 1. *A complexidade do algoritmo de Push-Relabel é da ordem de $O(n^2m)$.*

Demonstração. A complexidade do algoritmo pode ser limitada pelo número de PUSHES e RELABELS, que são da ordem de $O(n^2m)$ como visto nos Lemas 4, 5 e 6.

6. Considerações finais

Podemos observar que o algoritmo 4 não impõe nenhuma restrição sobre a ordem em que os vértices com excesso devem ser escolhidos. O estudo desses critérios é importante para que o algoritmo seja eficiente na prática. Dentre as estratégias mais comuns, escolher o vértice com excesso positivo que possuir a maior altura a cada iteração permite melhorar a complexidade assintótica para $O(n^2\sqrt{m})$ [1].

Em programação competitiva, implementações de Push-Relabel que seguem essa estratégia podem ser eficientes e enxutas, constituindo uma opção interessante para um algoritmo de fluxo máximo em materiais de referência; a implementação da KACTL é um exemplo dessa abordagem [5].

Referências

[1] Joseph Cheriyan and S. N. Maheshwari. Analysis of preflow push algorithms for maximum network flow. SIAM Journal on Computing, 18(6):1057–1086, 1989.

[2] Yefim A. Dinitz. Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Mathematics Doklady, 11:1277–1280, 1970.

[3] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the Association for Computing Machinery, 19(2):248–264, 1972.

[4] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. Journal of the Association for Computing Machinery, 35(4):921–940, 1988.

[5] Simon Lindholm. Push-relabel. https://github.com/kth-competitive-programming/kactl/blob/main/content/graph/PushRelabel.h, 2015. KTH Challenge Algorithm Template Library (KACTL). Acesso em: 31 jul. 2026.