10ª Jornada Científica e Tecnológica
7º Simpósio da Pós-Graduação
INSTITUTOS FEDERAIS 10 ANOS
INSTITUTO FEDERAL Sul de Minas Gerais
ISSN 2319-0124

ALGORITMOS PARA RESOLUÇÃO DO PROBLEMA DO FLUXO MÁXIMO EM GRAFOS

Samuel E. da SILVA[1]; Ricardo J. MARTINS[2] ; Rogério B. de PAIVA[3]

RESUMO

Sabe-se que, atualmente, o mundo está cada vez mais globalizado, o que gera um grande número de pessoas e dispositivos conectados a internet, e muitas vezes as topologias criadas não suportam o fluxo de informação que nelas passam. Um dos problemas que surgem, é o do Fluxo Máximo, que consiste em saber qual o fluxo máximo que uma rede consegue suportar. Existem alguns algoritmos que resolvem esse problema, porém existem diversas implementações para os mesmos. Este trabalho, portanto, pretende apresentar implementações eficientes e fáceis de serem utilizadas, para que cientistas e estudantes utilizem em seus trabalhos.

Palavras-chave: Grafos; Otimização; Fluxo Máximo; Redes; Modelagem Computacional

1. INTRODUÇÃO

O acesso à internet está em expansão. Segundo a *Internet Live Users* (2018), quase quatro bilhões de usuários utilizam Internet, o que representa aproximadamente 40% da população mundial. Estima-se também, segundo CISCO (2017), que em 2050, teremos 50 bilhões de dispositivos conectados à internet. Dessa forma, como muitas muitos equipamentos e pessoas estão conectados, há um grande número de informação e muitas vezes as topologias criadas para gerenciar esse fluxo podem ficar sobrecarregadas. O ideal é que essas informações não sejam perdidas, e assim faz-se necessário estudar medidas para prevenção e correção desses problemas.

Uma das maneiras de planejar como ocorre o transporte de informação é o fluxo máximo. Neste problema deseja-se conseguir sempre, em uma rede, o fluxo máximo de informações que ela pode conseguir suportar, para otimizar os processos.

Esse problema possui diversos exemplos que incluem, em roteamento de redes de computadores, como foi proposto por Schroeder (2006), em planos de evacuação para situações de emergência, como foi apresentado por Yamada (1996).

1 Orientado, IFSULDEMINAS – *Campus* Muzambinho. E-mail: 12151002618@ifsuldeminas.edu.br.
2 Orientador, IFSULDEMINAS – *Campus* Muzambinho. E-mail: ricardo.martins@ifsuldeminas.edu.br.
3 Coorientador, IFSULDEMINAS – *Campus* Muzambinho. E-mail: rogeiro.paiva@ifusuldeminas.edu.br

Assim, como o problema do fluxo máximo possui aplicações diversas, criar algoritmos eficientes e que economizem recursos é necessário.

Existem algoritmos que resolvem esse problema e diferentes formas de implementá-los. Este trabalho objetiva apresentar implementações eficientes e de fácil utilização, o que possibilita que estudantes e cientistas possam utilizá-los em seus trabalhos.

2. MATERIAL E MÉTODOS

Para a implementação dos algoritmos, foi necessário estudar a teoria do fluxo máximo, a partir do trabalho de Ford (1956). Os algoritmos de Ford Fulkerson (1956), Edmond Karp (1972) e Dinic (1970) foram implementados.

A linguagem de programação usada foi C++.

Para validação das implementações, foram utilizados alguns exercícios de programação que necessitam desses algoritmos. Esses exercícios foram retirados dos sites URI[4], UVa[5], e SPOJ br[6]. Basicamente esses exercícios propõem situações reais em que o problema do fluxo máximo é aplicável e necessita de um código eficiente que resolva o problema para ser aceito. Assim, pegando-se o tempo de execução dos três algoritmos em um dos exercícios, foi feito um comparativo para verificar qual o mais rápido.

3. RESULTADOS E DISCUSSÕES

Com esse trabalho, foram apresentados implementações eficientes de três algoritmos para resolução do problema do Fluxo Máximo. Durante o estudo, foi prezado deixar as implementações de forma fácil e prática de serem utilizadas.

Na tabela 1, podem ser observadas as funções principais dos três algoritmos. Na última linha da tabela há o tempo de execução de cada algoritmo para a resolução do exercício Hooligan do URI [7].

Neste exercício, eram dados como entradas grafos que tinham cerca de 40 vértices e até 100 arestas. O tempo calculado, foi a média entre o tempo de execução em todos os grafos da entrada.

[4] Link para o URI Online Judge - https://www.urionlinejudge.com.br/judge/pt
[5] Link para o UVa Online Judge - https://www.urionlinejudge.com.br/judge/pt
[6] Link para o Sphere Online Judge - http://www.spoj.com/
[7] Link do exercício: https://www.urionlinejudge.com.br/judge/pt/problems/view/1394

Tabela 1 - Trecho da implementação dos algoritmos

Ford Fulkerson	Edmond Karp	Dinic

`int ford_fulkerson(int _s, int _t){`
`  int maxflow = 0;`
`  while(1){`
`  memset(p, -1, sizeof p);`
`  dfs(_s);`
`  if(p[_t] == -1)break;`
`  int id = _t;`
`  int f = 0x3f3f3f3f;`
`  while(id != _s)`
`  {`
`  f = min(flow[p[id]][id],f);`
`  id = p[id];`
`  }`
`  id = _t;`
`  while(id != _s){`
`  flow[p[id]][id] -= f;`
`  flow[id][p[id]] += f;`
`  id = p[id];`
`  }`
`  maxflow += f;`
`  }`
`  return maxflow;`
`}`
	`int edmonds_karp(int _s, int _t)`
	`{`
	`  int fluxoMaximo = 0;`
	`  while(bfs(_s,_t)){`
	`  int menorAresta = oo;`
	`  int id = _t;`
	`  while(id != _s)`
	`  {`
	`  menorAresta = min(menorAresta, fluxo[p[id]][id]);`
	`  id = pai[id];`
	`  }`
	`  id = _t;`
	`  while(id != _s){`
	`  fluxo[p[id]][id] -= menorAresta;`
	`  fluxo[id][p[id]] += menorAresta;`
	`  id = pai[id];`
	`  }`
	`  fluxoMaximo += menorAresta;`
	`  }`
	`  return fluxoMaximo;`
	`}`
		`///Função principal do algoritmo`
		`void dinic(){`
		`  for (lim = (1 << 30); lim >= 1;)`
		`  {`
		`  if (!bfs()){`
		`  lim >>= 1;`
		`  continue;`
		`  }`
		`  for (int i = s; i <= t; i++)`
		`  pt[i] = 0;`
		`  int pushed;`
		`  while (pushed = dfs(s, lim)){`
		`  flow = flow + lim;`
		`  }`
		`  cout << flow << endl;`
		`  }`
		`  }`

mais de 1 segundo	0.120 segundos	0.120 segundos

FONTE: do autor.

Como pode-se notar, o algoritmo de Ford Fulkerson teve um desempenho inferior em termos de tempo, se comparado aos demais algoritmos. Isso pode ser justificado, em partes, por ser mais antigo e que sua complexidade computacional superior. Já os outros dois tiveram desempenho igual, pelo grafo do exercício se comportar bem em ambos.

As três implementações desenvolvidas no trabalho estão disponíveis no site GitHub[8].

4. CONCLUSÕES

Podemos concluir que o algoritmo de Ford-Fulkerson foi o mais custoso computacionalmente. Já os outros algoritmos de Edmond Karp e Dinic apresentaram custo computacional equivalente. É necessário avaliar o trabalho para outras entradas para que haja uma definição de qual algoritmo possui o melhor desempenho. Espera-se com isso que estudantes e

[8] Link para as implementações resultadas do trabalho - https://github.com/SamuelEduardoSilva/algoritmos/tree/master/tcc%20-%20implementa%C3%A7%C3%B5es

profissionais que necessitarem aplicar o fluxo máximo em seus projetos, possam utilizar essas implementações.

AGRADECIMENTOS

Primeiramente agradecer ao meu orientador do trabalho de conclusão de curso que auxiliou na escolha de um tema. Agradecer também ao IFSULDEMINAS por ter proporcionado uma excelente infraestrutura para estudo.

REFERÊNCIAS

CISCO. **Segundo projeção, até 2020 cerca de 50 bilhões de dispositivos estarão conectados à internet.** Disponível em: <https://goo.gl/2vQWXk> . Acesso em: 29 mar. 2018.

DINITS, E. A. Algorithms for solution of a problem of maximum flow in a network with power estimation. **Soviet Math. Cokl**., v. 11, p. 1277-1280, 1970.

EDMONDS, Jack; KARP, Richard M. Theoretical improvements in algorithmic efficiency for network flow problems. **Journal of the ACM (JACM)**, v. 19, n. 2, p. 248-264, 1972.

FORD JR, Lester R. **Network flow theory.** RAND CORP SANTA MONICA CA, 1956.

INTERNETLIVESTATS.COM. **Internet Users**. Disponível em: <http://www.internetlivestats.com/internet-users> . Acesso em: 29 mar. 2018.