Corte Mínimo de Grafos Pesados

Relatório do 2º Projecto - Análise e Síntese de Algoritmos

Baltasar Dinis, 89416 e Afonso Ribeiro, 86752

***Resumo*— Redes de distribuição devem ser desenhadas de forma a suportar o tráfego a que são submetidas. No entanto, a elevada densidade de interconexões entre os produtores, centros de distribuição e o destino final torna a avaliação das mesmas não óbvia. Neste relatório apresentamos uma solução para este problema, permitindo avaliar a capacidade da rede e que estações de abastecimento e ligações devem ser aumentadas para aumentar essa mesma capacidade. Este trabalho foi realizado no contexto da Unidade Curricular de Análise e Síntese de Algoritmos, no ano lectivo de 2018-2019.**

I. INTRODUÇÃO

Consideramos que há 3 categorias de vértices: 1) Os produtores, que têm um valor de produção p_i associado; 2) As estações de abastecimento, com capacidade para tratar uma determinada quantidade de bens; 3) uma estação de destino. Adicionalmente, cada ligação entre vértices têm um valor máximo que conseguem suportar.

O objetivo é calcular a capacidade da rede, e obter o fluxo máximo F de mercadorias, dos produtores para a estação de destino. Se $F < \sum_i p_i$, então a rede não é adequada, sendo necessário aumentar a capacidade das estações de abastecimento, bem como a capacidade das ligações.

O relatório está estruturado da seguinte forma: em II é apresentada a modelação do problema, o algoritmo e possíveis optimizações; em III é feita uma análise da complexidade da solução; em avalia-se experimentalmente a solução.

II. DESCRIÇÃO DA SOLUÇÃO

A. Modelação do Problema

Representamos o problema com um grafo dirigido pesado, no qual calculamos o corte mínimo. Consideramos os produtores como vértices, existindo um nó fantasma, que funciona como fonte, que se liga aos mesmos. A aresta da fonte s para o produtor i tem peso p_i. Assim conseguimos simular o produtor. Cada estação de abastecimento expande-se em dois vértices, ligados com uma aresta cuja capacidade é a da estação. O sentido da aresta é dos produtores para o destino.

Todas as ligações que precisam de ser aumentadas traduzem-se em arestas saturadas(pois limitam o fluxo). Só procuramos as estações e caminhos a aumentar mais próximas do sumidouro, precisamos de calcular o corte mínimo mais próximo do mesmo. Como pretendos fazê-lo a partir das alturas obtidas pelo algoritmo Push Relabel que da o corte mínimo mais próximo da fonte, è preciso executá-lo na rede transposta.

B. Cálculo do Corte Mínimo

Calculamos o corte mínimos de acordo com o método *Push-Relabel*[1], escolhendo os vértices com uma *FIFO*. Saturamos as arestas que partem da fonte e, enquanto os vértices estiverem ativos, é aplicada a operação de *discharge*.

Um vértice está ativo se tem excesso, sendo que quando lhe é inserido excesso (através de uma operação de *push*), é adicionado a uma fila. A operação de *discharge*, feita a partir do vértice no topo da fila. Envia todo o fluxo que consegue

para os nós que têm a altura uma unidade mais baixa da sua. Se, depois de fazer isto, continua com excesso, é feita uma operação de *relabel*, que aumenta a sua altura.

C. Optimizações

Há diversas optimizações que podem ser aplicadas, apresentamos aqui algumas: 1) inicialização de alturas mínimas; 2) cálculo periódico de alturas mínimas; 3) heurística de espaços . O objetivo é comum: diminuir o número de operações relabel desnecessárias.

Na inicialização pode ser feita uma procura em largura a partir do sumidouro no grafo transposto. Isto permite que se inicialize as alturas com as distâncias da procura, porque o fluxo tem de chegar à fonte, e para tal acontecer os nós de um nível da árvore de procura têm de estar mais baixos que os do nível abaixo (porque estão no caminho mais curto).

Pode ser refeita esta procura em profundidade, novamente a partir do sumidouro, mas desta vez no transposto do grafo residual, para recalcular as alturas mínimas necessárias. Como a procura em profundidade é $O(V+E)$, esta optimização seria feita com regularidade, mas não em demasia. O valor óptimo pode ser ajustado experimentalmente.

A terceira optimização é a heurística de espaços. Quando há um espaço nas alturas dos vértices — um valor $g <\mid V \mid$: $\nexists u \in V : h[u] = g$ — então podemos elevar todos os elementos com altura superior ao espaço para $\mid V \mid +1$, pois estes vértices não conseguem atingir o sumidouro.[2]

Na nossa implementação, optámos por não empregar estas heurísticas, pois não sentimos que fosse necessário e verificamos que a criação do grafo era um problema maior para a performance, devido ao uso da unidade de gestão de memória.

Há outras formas de calcular o fluxo máximo, que potencialmente seriam mais óptimas — nomeadamente utilizando a estrutura de árvores dinâmicas descrita em [1], que não consideramos para a implementação, em nome de evitar optimizações prematuras.

III. ANÁLISE TEÓRICA

Nesta secção fazemos uma breve análise da complexidade do algoritmo. Assumimos que o leitor está familiarizado com as operações de *push* e *relabel*, descritas em [1].

Definição 1 (Push não-saturante)**.** Um push diz-se não saturante quando não satura a aresta.

Lema 1. Uma operação de discharge tem no máximo um push não saturante. A prova é óbvia: se é feito um push não saturante não há mais excesso no vértice para enviar pela areta.

Definição 2 (Passagem)**.** Operações de discharge aplicadas nos vértices adicionados na passagem anterior.

Lema 2. Uma passagem custa $O(V)$, pois é feito no máximo um push não saturante por vértice durante a mesma.

Lema 3. São feitas $4n^2$ passagens na fila

Demonstração. Seja H a altura máxima dos vértices na fila. Quando é feita uma passagem podem acontecer 3 coisas: 1) H decresce, o que significa que um nó com altura máxima saiu da fila; 2) H mantém-se, o que significa que pelo menos um vértice (que tinha $h < H$) aumentou de altura; 3) H aumenta, o que significa que um vértice aumentou de altura, tal que $\Delta h[v] \geq \Delta H$. Como a altura máxima dos vértices é de $2 \mid V \mid < 1$, então as passagens onde H aumenta ou mantém-se são $O(V^2)$ (cada vértice só pode aumentar $O(V)$ vezes). Como $H_0 = 0$ e $H_f = 0$, então as passagens onde H diminui são também $O(V^2)$.

Teorema 1. *O algoritmo tem complexidade* $O(V^3)$.

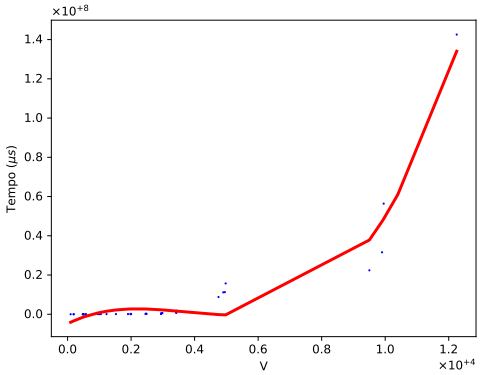

Fig. 1. Resultados experimentais: Latência em função do número de vértices

Demonstração. Corolário dos Lemas 2 e 3.

IV. AVALIAÇÃO EXPERIMENTAL

Geramos testes tendo em conta o número de vértices que gerariam (pois é a variável que dita a complexidade do algoritmo).

Na Figura 1 está exposta a relação entre o número de vértices e a complexidade temporal . O polinómio de regressão encontrado é cúbico tem um coeficiente de determinação $R^2 = 0.9473$, pelo que concluímos que o desempenho da nossa solução varia cubicamente com o número de vértices do grafo. Concluímos assim que estamos de acordo com o resultado teórico.

Note-se que no início, a curva parece constante. Isto deve-se ao facto de a própria construção do grafo ser pesada (devido à densidade e ao recurso à unidade de gestão de memória). Adicionalmente, os espaços entre os pontos deve-se à duplicação de arestas, o que dificultou a geração de um conjunto de dados que variasse linearmente no número de vértices.

Gostaríamos de ter gerado mais pontos, mas o facto do algoritmo ser cúbico dificultou isso mesmo.

Os testes foram efectuados numa arquitectura Intel® Core™i5-6200U.

V. CONCLUSÃO

Exposemos, analisamos e avaliamos experimentalmente uma solução ao problema proposto, mapeando-o para um problema de grafos onde calculamos o fluxo máximo de acordo com um algoritmo *push-relabel*, de complexidade $O(V^3)$.

REFERÊNCIAS

[1] A V Goldberg and R E Tarjan. 1986. “A new approach to the maximum flow problem.” In Proceedings of the eighteenth annual ACM symposium on Theory of computing (STOC ’86). ACM

[2] B V Cherkassky and A V Goldberg. 1994. “On implementing the push-relabel method for the maximum flow problem.” Integer Programming and Combinatorial Optimization. IPCO