Internet Paths

Algoritmia e Desempenho em Redes de Computadores

Grupo 9

Diogo Alves (86980)
Luís Crespo (87057)
Pedro Martin (87094)

Prof. João Sobrinho

1 Introdução

Com o objectivo de aplicar generalizações de algoritmos estudados em aula para o estudo do Border Gateway Protocol (BGP), algoritmo de eleição de caminhos numa rede Internet composta por ASes, foi definido um set de atributos, uma ordem total e uma extensão. Verificou-se que a propriedade de isotonicidade à direita não era satisfeita – no caso de uma extensão de *peer* com *peer* e *provider* com *peer*, *peer* é preferido a *provider*, mas nas extensões não – pelo que os algoritmos criados utilizam isotonicidade à esquerda.

2 Algoritmo I

2.1 Explicação

Com o objectivo de determinar para uma dada rede que tipo de caminho é eleito pelo BGP entre uma dada fonte e um dado destino, foi utilizado uma versão modificada do algoritmo de Dijkstra. Recorrendo ao uso de 4 filas (uma para cada tipo de caminho), o algoritmo começa por analisar o nó destino, actualizando os seus vizinhos de acordo com a ordem de preferência e a cada iteração as respectivas filas são actualizadas com os vizinhos descobertos para cada tipo de caminho. O algoritmo termina quando todas as filas estiverem vazias, o que significa que, por esta ordem, todas as ligações do tipo *customer* (armazenadas na fila de *customers*) foram analisadas, todas as ligações do tipo *peer* (armazenadas na fila de *peers*) foram analisadas, e assim adiante até às ligações do tipo *invalid*. O tipo de caminho dos vizinhos do nó actual é actualizado se a ligação para o nó actual for preferida, respeitando as extensões possíveis.

No caso da rede ser comercialmente conexa e que, portanto, todos os nós têm caminho comercial para todos os nós, um nó que só tenha ligações de um tipo para os seus vizinhos há de eleger, independentemente do destino, sempre esse tipo de caminho. Esta propriedade é utilizada neste algoritmo, sendo verificado se a rede é comercialmente conexa. Uma vez que, no caso de existirem ciclos na rede a complexidade de verificar esta propriedade aumenta significativamente, assume-se para efeitos deste algoritmo que a rede não é comercialmente conexa.

Uma vez que, para qualquer caso, a inserção e remoção de um elemento de uma fila tem complexidade $O(1)$ e que é feita uma remoção por nó e uma inserção por ligação, existindo $|V|$ nós e $|E|$ ligações na rede, o algoritmo tem complexidade $O(|V| + |E|)$. Para calcular a função de densidade de probabilidade (PDF), é executado este algoritmo para todos os nós destinos possíveis do grafo com uma complexidade de $O(|V|(|V| + |E|))$.

2.2 Estatísticas

Na Figura 1 podemos ver o gráfico da função PDF que este algoritmo produziu para a rede do ficheiro *LargeNetwork.txt*.

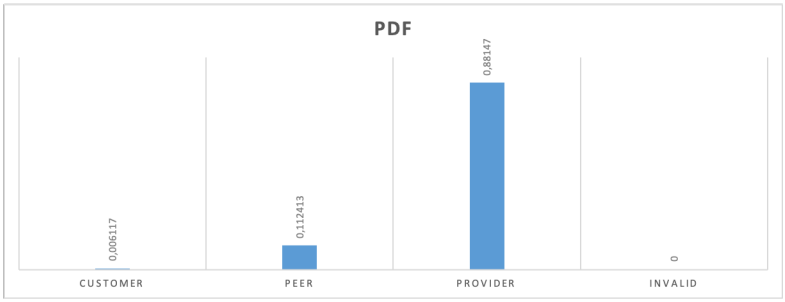

Figure 1: PDF do algoritmo I para o ficheiro "LargeNetwork.txt".

2.3 Discussão

Uma das ideias estudada e implementada para optimizar este algoritmo consiste em utilizar a informação acerca da análise feita no projecto anterior sobre se o grafo é ou não comercialmente conexo. Porém, esta informação não é usada na implementação da função que determina a PDF de uma dada rede. Porém, de forma a diminuir o tempo de execução dessa estatística, poderia-se remover os nós triviais que foram mencionados

anteriormente na explicação do algoritmo e correr o algoritmo modificado de Dijkstra para a nova rede. No caso de existirem componentes do grafo que tenham ficado desconexas da componente que contem o nó destino, os nós pertencentes a essas componentes elegem o tipo de caminho *provider*. É possível remover os nós triviais uma vez que i) se forem do tipo *provider* ou *peer*, então não poderão haver caminhos que passem por eles e sejam comerciais ao mesmo tempo e ii) se forem do tipo *customer*, então são necessariamente Tier-1 (uma vez que não têm *providers*) o que implica necessariamente que, se uma componente da rede estiver ligada a esse nó e ficar separada da componente que contem o nó destino, todos os caminho comercias dessa componente passam por esse nó Tier-1 não podendo eleger caminhos de outros tipos sem esses caminhos deixarem de ser comerciais.

3 Algoritmo II

3.1 Explicação

Foi utilizada uma versão modificada do algoritmo de Dijkstra de forma a calcular a distância do caminho eleito pelo BGP. Para as estatísticas é invocado este algoritmo para cada destino da rede. O algoritmo desenvolvido recebe o nó de destino e retorna um vetor com as distancias de todos os nós para esse destino. O algoritmo começa com um acervo com $|V|$ nós e distâncias infinitas para todos eles, excepto para o destino, que tem distância 0. A cada iteração é extraído do acervo o nó u cujo caminho é preferível com distância mínima e são actualizadas, para os vizinhos desse nó (v), os tipos de caminhos e as distâncias. Na verdade, é utilizado apenas um vetor no acervo para estes dois parâmetros: $d + t \times n$. Em que d é a distância do nó ao destino, t o tipo de caminho e n o número de nós. Desta forma não é necessário utilizar dois vetores (um para o tipo de caminho e outro para as distâncias) e a prioridade continua a ser o tipo de caminho e o critério de desempate a distância.

Este algoritmo tem a complexidade do algoritmo de Dijkstra na situação em que é utilizado um acervo para guardar as distâncias dos nós, ou seja, $O((|V| + |E|)log(|V|)$. Para calcular a função distribuição acumulada complementar (CCDF), é executado este algoritmo para todos os nós d do grafo, com uma complexidade de $O(|V|(|V| + |E|)log(|V|)$.

3.2 Estatísticas

Na Figura 2 podemos ver o gráfico da função CCDF que este algoritmo produziu para a rede do ficheiro *LargeNetwork.txt*.

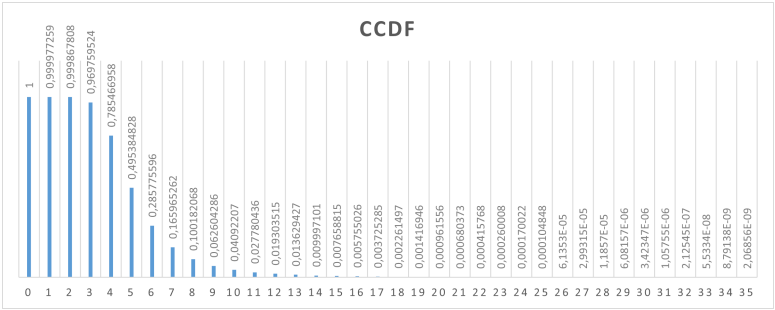

Figure 2: CCDF do algoritmo II para o ficheiro "LargeNetwork.txt".

3.3 Discussão

Para obter as distâncias mínimas para o tipo de caminho eleito pelo BGP é necessário utilizar uma heap de forma a otimizar o Dijkstra e não pode ser utilizada a solução do algoritmo I (4 filas) porque as prioridades não se resumem a três tipos, envolve distâncias até n nós. Utilizando a codificação referida para as distâncias ($d + t \times n$), o problema torna-se o típico Dijkstra de cálculo de caminhos mais curtos mas com condições de relaxamento mais apertadas.

4 Algoritmo III

4.1 Explicação

Com este algoritmo pretende-se, dada uma origem e um destino num grafo, obter o tamanho do caminho de menor comprimento, que seja do mesmo tipo que o caminho BGP calculado para aquela origem e destino.

Para tal, foi utilizada uma versão modificada do algoritmo de Dijkstra. O algoritmo desenvolvido recebe um vetor com os tipos de caminhos BGP de todos os nós para um destino d e produz como output os tamanhos dos caminhos mínimos de todos os nós para o nó d, cujo tipo é o mesmo do caminho BGP. Durante a execução do algoritmo existem quatro variáveis de estado para cada nó s:

- A distância mínima entre s e t por um caminho do tipo *Provider* ($distP_s$)
- A distância mínima entre s e t por um caminho do tipo *Peer* ($distR_s$)
- A distância mínima entre s e t por um caminho do tipo *Customer* ($distC_s$)
- A distância mínima entre s e t por um caminho do tipo igual ao tipo de caminho BGP ($dist_s$)

Estas variáveis vão sendo atualizadas para cada nó durante a execução do programa e, evidentemente, assumindo que existe um caminho comercial entre s e d, o valor de $dist_s$ vai ter um valor igual a uma das outras três variáveis.

O algoritmo começa com um acervo com $|V|$ nós e distâncias ($dist_s$) infinitas para todos eles, excepto para o nó d, que tem distância 0. A cada iteração do algoritmo de Dijkstra, é extraído do acervo o nó u cuja distância é mínima e são atualizadas, para os vizinhos desse nó (v), as distâncias $distP_v$, $distR_v$ e $distC_v$, com base na nova informação, da seguinte forma:

- $distC_v := MIN(distC_v, distC_u + 1)$
- $distR_v := MIN(distR_v, distC_u + 1)$
- $distP_v := MIN(distP_v, distP_u + 1, distR_u + 1, distC_u + 1)$

Esta atualização é feita tendo em conta as regras para caminhos comerciais (isto é, os caminhos comerciais são da forma PRC, com $P \geq 0, R = \{0, 1\}, C \geq 0$). É também atualizada a variável $dist_v$, conforme o caminho BGP do nó v para o nó d (por exemplo, se o caminho BGP for do tipo *Provider*, $dist_v := distP_v$ e é atualizada a distância de v no acervo com este valor.

Este algoritmo tem a complexidade do algoritmo de Dijkstra na situação em que é utilizado um acervo (Binary Heap) para guardar as distâncias dos nós, ou seja, $O((|V| + |E|)log(|V|)$. Para calcular a função distribuição acumulada complementar (CCDF), é executado este algoritmo para todos os nós d do grafo, com uma complexidade de $O(|V|(|V| + |E|)log(|V|)$.

4.2 Estatísticas

Na Figura 3 podemos ver o gráfico da função CCDF que este algoritmo produziu para a rede do ficheiro *LargeNetwork.txt*. Comparando este gráfico com o produzido pelo algoritmo anterior, verificamos que este gráfico assume sempre valores menores ou iguais, como já seria de esperar, uma vez que os caminhos encontrados ou têm igual comprimento ou são menores.

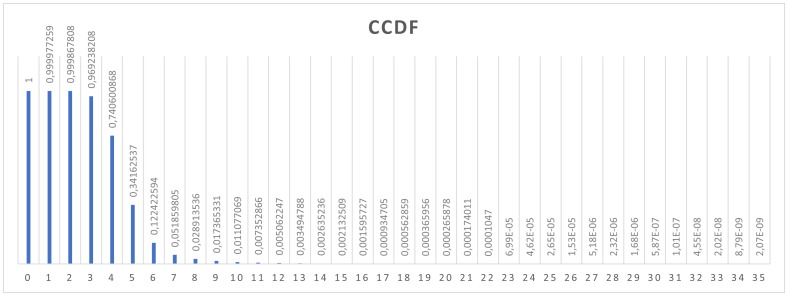

Figure 3: CCDF do algoritmo III para o ficheiro "LargeNetwork.txt".

4.3 Discussão

Quando nos deparámos com este problema, o nosso primeiro instinto foi utilizar o algoritmo BFS, partindo da origem por ligações do tipo do caminho BGP, uma vez que em grafos com ligações de peso unitário, a BFS é capaz de calcular caminhos mínimos com uma complexidade inferior ao Dijkstra. No entanto, apercebemo-nos que este método não funcionaria, porque: (1) havia situações em que a BFS poderia não chegar ao destino; (2) a não ser que se construísse uma matriz com todos os tipos de ligações entre pares (o que é impraticável - esta matriz ocuparia cerca de 16GB), a complexidade para calcular a CCDF seria muito superior.