Sloan

1. Análise de complexidade do algoritmo de Sloan

O algoritmo para redução de *profile* proposto por Sloan [1] é composto de duas etapas. Na primeira etapa, são determinados dois vértices pseudoperiféricos. Um deles é utilizado como vétice inicial para a renumeração e o outro é utilizado na função de priorização que determina a ordem dos vértices renumerados na segunda etapa. Nessa primeira etapa, um vértice qualquer s do grafo a ser renumerado é escolhido. Então, gera-se a estrutura de nível enraizada $\mathscr{L}(s)$. Dado um vértice $v \in V$, a estrutura de nível $\mathscr{L}(v)$ enraizada no vértice v, com profundidade $\ell(v)$, é o particionamento $\mathscr{L}(v) = \{L_0(v), L_1(v), \ldots, L_{\ell(v)}(v)\}$, em que $L_0(v) = \{v\}$ and $L_i(v) = Adj(L_{i-1}(v)) - \bigcup_{j=0}^{i-1} L_j(v)$, para $i = 1, 2, 3, \ldots, \ell(v)$, and $Adj(U) = \{w \in V : (u \in U \subseteq V)\ \{u, w\} \in E\}$. Em particular, $\ell(v) = \max_{u \in V}[d(v, u)]$ denota a excentricidade do vértice v e a distância $d(v, u)$ é o menor número de arestas de v a u.

A partir da estrutura de nível enraizada $\mathscr{L}(s)$, define-se, então, o conjunto Q, que contém um vértice de cada grau do último nível $L_{\ell(s)}(s)$ da estrutura de nível enraizada $\mathscr{L}(s)$. Em seguida, para cada vértice $w \in Q$, verifica-se $l(w) > l(s)$. Caso a condição seja satisfeita para algum vértice w, então, esse vértice passa a ser o vértice s corrente e o processo é repetido. Caso contrário, a etapa termina e retorna o vértice s e o vértice com menor largura de nível e entre os vértices pertencentes a Q.

No algoritmo 1, é mostrado um pseudocódigo da segunda etapa do algoritmo de Sloan [1]. Nessa etapa, os vértices são numerados em ordem decrescente conforme a função de priorização definida na linha 4 desse pseudocódigo. Para essa função, o grau de cada vértice e a sua distância até e são considerados. Ainda, os pesos w_1 e w_2 são utilizados de modo a ponderar a influência dessas características no cálculo da prioridade. Sloan [1] recomendou a utilização de $w_1 = 1$ e $w_2 = 2$.

No algoritmo de Sloan [1], uma fila de prioridades é criada a partir dos vértices de entrada. Para a inserção de elementos na fila de prioridades, o método atribui estados a cada vértice. Um vértice já numerado recebe o estado pós-ativo (veja a linha 21) e os demais vértices terão estado ativo, pré-ativo ou inativo. Todo vértice ainda não numerado que seja adjacente a um vértice pós-ativo é definido como em estado ativo. Os vértices adjacentes a um vértice ativo são definidos como em estado pré-ativo. Já os vértices que não possuem estados ativo, pós-ativo ou pré-ativo, são definidos como vértices inativos. Dessa forma, a fila de prioridades contém apenas vértices com estado ativo ou pré-ativo. Esse processo termina após todos os vértices serem numerados e removidos da fila de prioridades.

Para a análise do pior caso, pode-se considerar um grafo em que o vértice inicial é adjacente a todos os demais vértices do grafo. Dessa forma, a atualização na fila de prioridades passa a ser mais dispendiosa; pois, todos os vértices são inseridos na fila de prioridades F.

No algoritmo 1, tem-se um laço de repetição da linha 2 até a linha 5. Nesse trecho do algoritmo, o *status* inativo é atribuído a cada vértice do grafo, bem como as prioridades dos vértices são inicializadas. Esse laço é executado $|V|$ vezes. Para a determinação das prioridades iniciais de cada vértice do grafo, são executadas outras duas subrotinas: *Dist*, para a determinação da distância do vértice v ao vértice pseudoperiférico e; e *Grau*, para a verificação do grau do vértice v. Para se determinar a distância entre os vértices v e e, pode ser utilizada uma estrutura de nível enraizada no vértice e e calcula-se o nível em que o vértice v consta nessa estrutura. Dessa forma, obtém-se custo $O(|E| + |V|)$ nessa operação. Já para a verificação do grau de cada vértice, tem-se custo $O(|E|)$; logo, tem-se custo $O(|E| + |V|)$ nas $|V|$ iterações desse laço de repetição. Dessa forma, a linha 4, referente à inicialização das prioridades dos vértices, é dominante no trecho de código, com custo $O(|V| + |E|)$.

Nas linhas 6 e 7, o vértice pseudoperiférico s é inserido na fila de prioridades F e é atribuído o *status* pré-ativo para esse vértice. Para essas linhas, assim como para a linha 8, tem-se custo $O(1)$. No laço de repetição das linhas 9 a 38, os vértices pertencentes à fila de prioridades F são renumerados em ordem decrescente de prioridade. Os vértices do grafo serão numerados, um a cada iteração no pior caso em análise aqui; portanto, o laço é executado $|V|$ vezes.

Na linha 10, o vértice v de maior prioridade é removido da fila por meio da subrotina *RemoveVertive*. O custo para remover um vértice de uma fila de prioridades é $O(1)$; porém, ao se retirar um vértice da fila de prioridades, é necessário reformatar a *heap* máxima. Logo, para $|V|$ iterações, tem-se custo $O(|V| \cdot lg(|V|))$ nessa operação. Se o mesmo vértice possuir *status* pré-ativo (linha 11), então, para cada vértice u adjacente ao vértice v (linha 12), é realizada a atualização da prioridade do vértice u (linha 13). Se o *status* do vértice u for inativo (linha 14), atribui-se o *status* pré-ativo a esse vértice (linha 15) e o insere na fila de prioridades F (linha 16).

Ao se considerar o pior caso analisado aqui, na linha 12, são percorridos os vértices adjacentes apenas do primeiro vértice, pois os demais vértices não serão pré-ativos; todos os demais vértices já terão o *status* ativo ao serem verificados na condição da linha 11. Assim, como no pior caso o vértice inicial é adjacente a todos os demais vértices do grafo, tem-se custo $O(|V|)$ para a linha 12. A condição da linha 14 também será verificada em custo $O(|V|)$, o mesmo custo que a linha 13,

Algorithm 1: Etapa de numeração de vértices do algoritmo de Sloan [1].

```
   Entrada: grafo conexo G = (V, A), vértices
            pseudoperiféricos s e e, e pesos w1(=1) e
            w2(=2).
   Saída: renumeração S = {s(1), s(2), ..., s(|V|)}.
1  início
2    para cada ( v ∈ V(G) ) faça
3      v.status ← inativo;
4      v.prioridade ←
         w1.Dist(v,e) − w2.(Grau(v) + 1);
5    fim-para-cada
6    F ← {s};
7    s.status ← pré-ativo;
8    i ← 1;
9    enquanto ( F ≠ ∅ ) faça
10     v ← RemoveVertice(F);
11     se ( v.status = pré-ativo ) então
12       para cada ( u ∈ Adj(v) ) faça
13         u.prioridade ← u.prioridade + w2;
14         se ( u.status = inativo ) então
15           u.status ← pré-ativo;
16           F ← F ∪ {u};
17         fim-se
18       fim-para-cada
19     fim-se
20     s(i) ← v;
21     v.status ← pós-ativo;
22     i ← i + 1;
23     para cada ( t ∈ Adj(v) ) faça
24       se ( status = pré-ativo ) então
25         t.status ← ativo;
26         t.prioridade ← t.prioridade + w2;
27         para cada ( u ∈ Adj(t) ) faça
28           se ( u.status ≠ pós-ativo ) então
29             u.prioridade ←
               u.prioridade + w2;
30           fim-se
31           se ( u.status = inativo ) então
32             u.status ← pré-ativo;
33             F ← F ∪ {u};
34           fim-se
35         fim-para-cada
36       fim-se
37     fim-para-cada
38   fim-enquanto
39   retorna S;
40 fim
```

visto que os vértices que têm sua prioridade atualizada na linha 13 ainda não estão na fila de prioridades F. O custo da operação de se atualizar e de se inserir um vértice na fila de prioridades implementada como uma *heap* máxima é $O(lg(|V|))$, pois serão inseridos, no máximo, $|V|$ vértices na fila de prioridades. Desse modo, tem-se custo $O(|V| \cdot lg(|V|))$ para a linha 16 e a linha 15 apresenta custo $O(|V|)$.

Em seguida, o vértice v é renumerado e recebe o *status* pós-ativo (linhas 20 e 21). A variável i é incrementada na linha 22. O custo para essas operações é $O(1)$, mas como são executadas $|V|$ vezes, tem-se custo $O(|V|)$ para essas linhas.

Para cada vértice t adjacente ao vértice v (linha 23) que possui o *status* pré-ativo (linha 24), atualiza-se a prioridade e o *status* do vértice t para ativo (linhas 25 e 26). A linha 24 é executada $|E|$ vezes; porém, essa condição é satisfeita $|V|$ vezes, já que cada vértice deixa de ter *status* pré-ativo e passa a ter *status* ativo apenas uma vez. Desse modo, a linha 25 é executada $|V|$ vezes. O custo para se atualizar a prioridade dos vértices na linha 26 é $O(|V| \cdot lg(|V|))$.

Por fim, na linha 27, são percorridos os vértices adjacentes de cada vértice que satisfaz a condição da linha 24. Isso faz com que as linhas 27 e 28 sejam executadas $|E|$ vezes. O custo para atualizar a prioridade dos vértices adjacentes na linha 29 tem custo $O(|E| \cdot lg(|V|))$. A condição da linha 31 também é verificada $|E|$ vezes; porém, no pior caso, essa condição nunca é satisfeita. Desse modo, as linhas 32 e 33 não são executadas no pior cenário avaliado aqui.

Na Tabela I, são mostrados os custos referentes a cada linha do algoritmo 1. Assim, no pior caso, tem-se custo $O((|E| + |V|) \cdot lg(|V|))$ para essa implementação do algoritmo de Sloan [1].

Tabela I
COMPLEXIDADE NO PIOR CASO DO ALGORITMO DE SLOAN [1].

Linha	Custo
1	—
2	$O(\|V\|)$
3	
4	$O(\|V\| + \|E\|)$
5	$O(\|V\|)$
6	$O(1)$
7	
8	
9	$O(\|V\|)$
10	
11	
12	
13	
14	
15	
16	$O(\|V\| \cdot lg(\|V\|))$
17	—
18	$O(\|E\|)$
19	—
20	$O(\|V\|)$

Linha	Custo
21	$O(\|V\|)$
22	
23	$O(\|E\|)$
24	$O(\|V\|)$
25	$O(\|V\|)$
26	$O(\|V\| \cdot lg(\|V\|))$
27	$O(\|E\|)$
28	
29	$O(\|E\| \cdot lg(\|V\|))$
30	—
31	$O(\|E\|)$
32	—
33	
34	
35	$O(\|E\|)$
36	—
37	$O(\|E\|)$
38	$O(\|V\|)$
39	$O(1)$
40	—

REFERÊNCIAS

[1] S. W. Sloan. A Fortran program for profile and wavefront reduction. *International Journal for Numerical Methods in Engineering*, 28(11):2651-2679, 1989.