Segundo Trabalho Prático (CI1238)

Caio Henrique Ramos Rufino

23 de julho de 2024

1 Explicação do Problema

Comissão Representativa[1]: Um órgão do governo quer montar uma comissão para tratar de um certo assunto. Mas este órgão, preocupado com a representatividade da comissão, quer que todos os grupos (previamente elencados) da sociedade sejam representados. Uma pessoa pode fazer parte de mais de um destes grupos, portanto podemos ter menos representantes que grupos. Dados um conjunto S de grupos, um conjunto C de candidatos, e um subconjunto $S_c \subseteq S$, para cada candidato c, indicando os grupos dos quais c faz parte, devemos encontrar um conjunto $X \subseteq C$ tal que $\bigcup_{c \in X} S_c = S$ e $|X|$ seja mínimo.

2 Modelagem da função limitante

A nova função limitante foi modelada com base em uma estratégia gulosa bastante simples. A função retorna o tamanho do conjunto de candidatos escolhidos até o momento, $|E|$, mais a quantidade de conjuntos necessários para cobrir S, considerando apenas a quantidade de grupos que cada candidato cobre e não a cobertura dos grupos.

Para calcular essa função limitante, é necessário realizar os seguintes passos:

1. Criar um conjunto onde cada elemento representa o tamanho do conjunto de grupos coberto por cada candidato.
2. Ordenar esse conjunto de forma decrescente.
3. Selecionar iterativamente o elemento de maior valor, removê-lo do conjunto, e subtrair esse valor da quantidade de grupos que ainda falta cobrir.

Em resumo, o método seleciona os candidatos que cobrem o maior número de grupos, até que o conjunto dos grupos a serem cobertos seja completamente coberto, considerando apenas o número de grupos e não a cobertura em si.

3 Comparação entre as funções limitantes

A tabela a seguir contém os resultados de diversos casos de teste com entradas aleatórias para comparar o desempenho entre a função limitante dada, e a nova:

		bound do professor		bound nova		bound do professor / bound nova	
grupos	candidatos	Nós	tempo(ms)	Nós	tempo(ms)	Nós	tempo(ms)
10	20	52177	9.1	1137	1.1	45.8	7.8
10	20	55645	11.4	619	0.6	89.9	18.1
10	20	14323	3.3	239	0.2	59.8	12.5
15	30	1542267	398.6	13261	10.3	116.2	38.6
15	30	1438145	381.4	35295	24.7	40.8	15.4
15	30	776757	207.4	14921	11.5	52.1	18.1
20	40	1550887	508.4	19617	22.4	79.1	22.7
20	40	1702443	582.6	86579	94.1	19.7	6.2
20	40	9217847	3185.3	280657	352.1	32.8	9.1
25	50	19628151	8030.0	299885	510.0	65.4	15.7
25	50	15074145	6566.0	806351	1290.7	18.7	5.1
25	50	40687881	16674.9	1358093	2013.1	30.0	8.3
30	60	113833807	56339.2	3132231	5630.0	36.4	10.0
30	60	340998931	170258.7	13641287	25552.9	25.0	6.7
30	60	885309535	444682.7	13206579	29449.0	67.0	15.1

É visível que a nova função limitante desempenha significativamente melhor, isso porque fornece uma estimativa mais precisa de $P(X)$[1], que é o valor máximo de uma solução descendente de X. Enquanto a função limitante dada verifica apenas se todos os grupos estão cobertos, retornando 0 ou $|E| + 1$, a nova função calcula a quantidade mínima de candidatos adicionais necessários para cobrir todos os grupos restantes. Isso aproxima melhor $P(X)$ e permite uma poda mais expressiva.

4 Detalhes da implementação

4.1 Modo de uso

Para gerar o executável chamado `comissao`, basta rodar o comando `make` no diretório corrente.

O formato de entrada e saída segue a especificação do trabalho. A entrada é recebida pela `stdin`, e a saída impressa tanto na `stdout` (Conjunto de menor tamanho dos candi-

datos que soluciona o problema, ou `Inviavel`, se não houver solulção) quanto na `stderr` (Nós percorridos e tempo de sexecução em segundos).

4.2 Modelagem computacional

Para percorrer a árvore de soluções possíveis se utilizou de uma função recursiva, que mantém um vetor de candidatos escolhidos, onde cada posição representa um candidato e recebe o valor 1 se este foi escolhido, e 0 caso contrário.

4.3 Arquivos e estrutura do diretório

Os arquivos de código fonte foram separados entre dois diretórios, `src`, para arquivos `.c` e `include` para arquivos `.h`.

Além disso, existem três diretórios utilizados para testes. Um deles contém os arquivos de entrada e um script em Python para gerar entradas, chamado `entradas_teste`. Outro diretório, `scripts`, contém dois scripts que geraram os arquivos de saída dos testes, que estão no diretório `log`. Há dois scripts e dois arquivos `.log` diferentes: um se refere à função de bound nova e o outro à fornecida na especificação do trabalho.

Referências

1 KREHER, D. L.; STINSON, D. R. **Combinatorial Algorithms; Generation, Enumerantion, and Search**. [S.l.]: CRC Press, 1999.

2 GUEDES, A. **Segundo Trabalho Prático**. [s.n.], 2024. Disponível em: <https://www.inf.ufpr.br/andre/Disciplinas/CI1238-2024-1/trabalho2.pdf>.