TP 01- O Plano de Campanha

Leandro Freitas de Souza - 2021037902

Introdução:

Esse trabalho se trata de um problema típico de satisfabilidade em grafos: um político, com o intuito de aumentar a coerência de suas propostas, realiza uma enquete em uma rede social em que cada eleitor escolhe duas propostas para serem mantidas e duas propostas para serem retiradas do plano de governo.

A partir desses dados, o algoritmo implementado deve calcular se é possível ou não satisfazer todos os seguidores. Cada seguidor é satisfeito se, e somente se ao menos uma das propostas votadas a favor é mantida no plano e ao menos uma das propostas votadas contra é retirada dele.

Modelagem:

Sabemos que o problema pode ser tratado como um problema de lógica booleana, em que:

$$S = (P_1 + P_2)(\neg P_3 + \neg P_4)...(P_i + P_j)(\neg P_k + \neg P_l)...$$

Essa equação característica do problema gera algumas implicações lógicas:

$\neg P_1 \rightarrow P_2$ $\quad\quad$ $\neg P_i \rightarrow P_j$

$\neg P_2 \rightarrow P_1$ $\quad\quad$ $\neg P_j \rightarrow P_i$

$P_3 \rightarrow \neg P_4$ $\quad\quad$ $P_k \rightarrow \neg P_l$

$P_4 \rightarrow \neg P_3$ $\quad\quad$ $P_l \rightarrow \neg P_k$

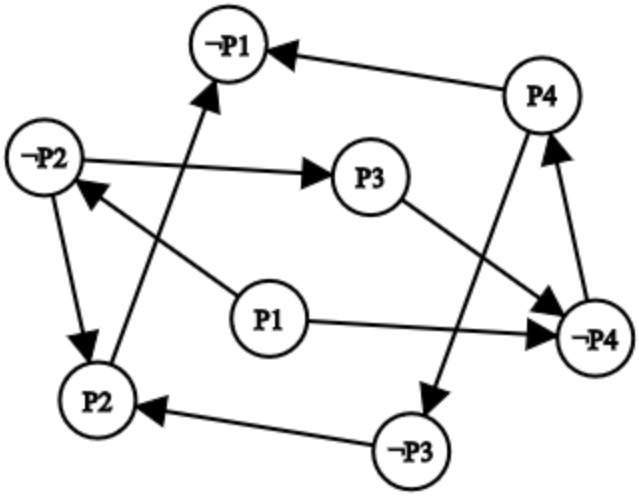
Dessa forma, a partir dessas implicações, foi modelado um grafo em que cada vértice representa um estado para cada proposta. Em outras palavras, cada proposta possui

dois vértices correspondentes: um que representa o estado de aceitação da proposta (em que ela será mantida), e um que representa o seu estado de negação (em que ela será retirada). Assim, a partir desses estados e das implicações obtidas a partir do problema, podemos traçar as arestas do grafo.

Um caso importante que deve ser observado é o caso em que uma pessoa realiza apenas uma escolha e, desse modo, essa escolha deve ser satisfeita. Assim, temos que a não realização dessa escolha deve resultar em uma contradição. Para descrever essa contradição, em um caso em que apenas uma proposta Pi foi escolhida, foi utilizada a implicação

$$\neg P_1 \rightarrow P_1$$

Assim, com todas as implicações e os vértices do grafo definidos, podemos definir as arestas do grafo exatamente como as implicações obtidas.



Grafo gerado pelo primeiro caso dado no enunciado

Uma vez que o grafo foi obtido, temos que se é possível, a partir de um vértice P_i, chegar no vértice $\neg P_i$ e chegar em a partir de $\neg P_i$, independente da escolha para a

proposta $\neg P_i$, será gerada uma contradição. Dessa maneira, é possivel satisfazer todos os seguidores se, e somente se para todo vértice P, P não pertence ao mesmo componente fortemente conexo que ¬P.

Algorítmo de Kosaraju

Uma vez modelado o grafo e definido o problema, foi utilizado o algorítmo de Kosaraju para definir os componentes fortemente conectados do grafo.

Esse algorítmo consiste em três etapas:

1. Executar um DFS no grafo correspondente ao problema para guardar o tempo de finalização em cada nó. Nessa etapa, ao final de cada execução do DFS, o vértice correspondente é adicionado em uma pilha, de modo a guardar a ordem de finalização.
2. Obter o grafo transposto do grafo correspondente ao problema. Esse grafo foi obtido já na definição do primeiro grafo, no início do código.
3. No grafo transposto, executar o DFS novamente. Cada componente obtido no algorítmo será fortemente conexo.

Assim, uma vez que os componentes fortemente conexos foram obtidos, basta, para cada vértice, verificar se ele não pertence ao mesmo componente que seu complemento. Para isso, na execução do DFS, foi implementado um vetor que atribuia um componente para cada vértice, o que possibilita essa verificação em tempo $\mathcal{O}(P)$, sendo P o número de pessoas.

Complexidade

A complexidade de tempo da realização de um DFS é $\mathcal{O}(V + E)$. Uma vez que $V = 2P$, sendo P o número de propostas, e $E = 4S$, sendo S o número de seguidores que responderam a enquete. Assim, temos que a complexidade do DFS é $\mathcal{O}(2P + 4S) = \mathcal{O}(P + S)$.

O código executa, em sequência, dois DFS e depois verifica os componentes conexos de cada vértice. Assim, sua complexidade é:

$$\mathcal{O}(P + S) + \mathcal{O}(P + S) + \mathcal{O}(P) = \mathcal{O}(P + S)$$