TP1

Henrique Daniel de Sousa
Matrícula: 2021031912

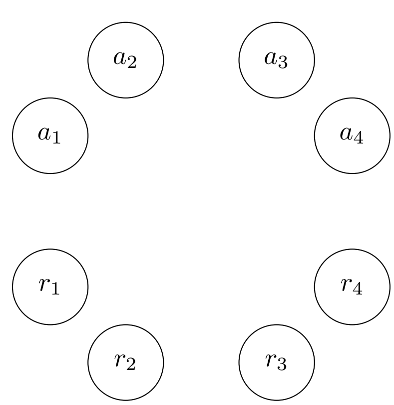
1 Introdução

O problema em questão se trata da verificação da existência de um possivel conjunto de leis que não criem uma contradição entre suas aprovações e rejeições e que possa atender a todas votações dos seguidores. Logo, pensando na atribuição de valores para as propostas, sendo que 0 significa a rejeição e 1 a aprovação, é possível organizar os votos como um produto de somas da forma booleana. Então, consequentemente, podemos interpretar e resolver este problema usando o algoritmo de resolução para o problema **2-SAT**.

2 Modelagem

2.1 Modelagem do problema em grafos

Pensando na modelagem do problema como um problema **2-SAT**, podemos criar um grafo $G = (V, E)$, tal que $|V| = 2P$, sendo P o número de propostas e para cada seguidor fossem adicionadas cláusulas ao produto de somas booleano. Assim, cada vértice desse grafo corresponde a uma proposta, de modo que para toda proposta exista um vértice de aprovação e um vértice de rejeição.
Assim, o grafo de uma campanha com 4 propostas pode ser visto da seguinte forma:



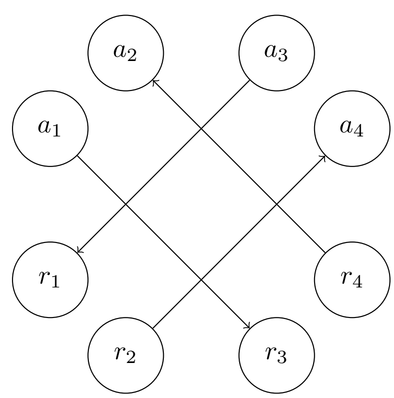
Assim, é possível modelar os votos da seguinte forma:

Digamos que um seguidor votou a favor de 1 e 3 e contra 2 e 4. Desse modo, podemos adicionar duas cláusulas à equação:

$$(a1 + a3)(r2 + r4)$$

Portanto, o problema pode ser resolvido se a equação booleana que descreve a a junção de todas cláusulas tiver solução. Assim, devemos adicionar arestas que liguem os vértices para verificar essa condição. Isso pode ser feito ligando, para cada cláusula $(x + y)$, $-x \rightarrow y$ e $-y \rightarrow x$.

Então, o grafo para representar a entrada de exemplo ficaria da seguinte forma:



Nessa implementação do trabalho, o grafo foi representado como um vetor com listas de adjacência.

Então, para cada clausula $(x + y)$, devemos verificar se existe, ao mesmo tempo:

1. caminho de x até $-x$
2. caminho de $-x$ até x
3. caminho de y até $-y$
4. caminho de $-y$ até y

Caso ambos existam, então a equação não pode ser satisfeita.

2.2 Algoritmo para resolução do problema

Esse problema pode ser resolvido aplicando o algoritmo de Kosaraju, que consiste nos seguintes passos:

1. Aplique DFS no grafo e guarde a ordem topológica inversa numa pilha;
2. Crie um grafo G^t, que inverte as direções de todas arestas;
3. Processe um DFS no grafo G^t, sendo que os vértices iniciais devem ser retirados do topo da pilha.

Desse modo, é possivel obter os componentes fortemente conexos (SCC) do grafo. Portanto, para verificar se há caminhos $x \to -x$ e $-x \to x$ ao mesmo tempo, basta verificar se x e $-x$ estão no mesmo SCC.

Analisando a complexidade de tempo deste algoritmo, observa-se que:

(i) A complexidade do DFS é $O(|V| + |E|)$

(ii) A complexidade para gerar o grafo transposto é $O(|V| + |E|)$

Portanto, a complexidade total do algoritmo é $O(|V| + |E|)$, ou seja, linear.