Relatório - Trabalho 2

Otimização

Brendon Henrique
bhps17@inf.ufpr.br

GRR20170203

Universidade Federal do Paraná
Bacharelado em Ciência da Computação

1 Problema:

O problema resolvido foi o Elenco Representativo, onde dado um conjunto de grupos da sociedade, um conjunto de atores e determinado número de personagens, o algoritmo implementado deveria retornar um conjunto de atores de menor custo possível que representasse todos os conjuntos. A implementação necessariamente deveria usar o algoritmo de Branch and Bound para resolver este problema, dando duas opções de funções de bound, onde uma seria default(a que apresenta melhor desempenho) e outra seria utilizada mediante a indição da opção '-a' via linha de comando.

2 Modelagem:

A modelagem utilizada para resolver este problema foi bastante simples: A ideia basicamente era realizar uma busca em profundidade em árvore representada por uma fila começando com o nodo que contém somente o ator de menor custo até o nodo que contém maior nível possível que seria o número de personagens buscados para o elenco.

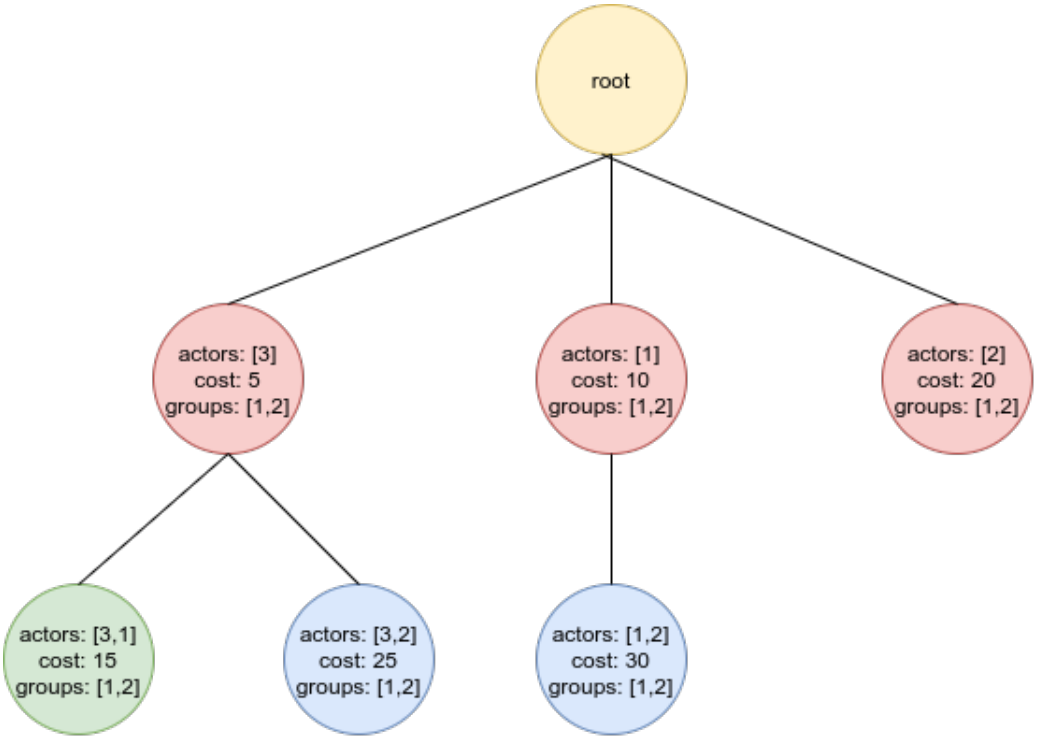
Ou seja, sempre trabalhamos com o conjunto de atores ordenado por custo de cada ator e em cada ramificação só checamos os próximos nodos com combinações que ainda não foram geradas e verificadas na árvore de execução. Trabalhando dessa forma temos a certeza de que a primeira solução que encontrarmos, ou seja, o primeiro nó que tenha o número de atores igual ao número de personagens buscados e os grupos que os atores do nó compõe são os mesmos grupos buscados por quem deseja montar o elenco será a solução, pois devido ao ordenamento por custo dos atores e a ordem de execução, está é solução de menor custo possível.

Exemplo de entrada:

```
2 3 2
10 2
1
2
20 1
2
5 2
1
2
```

Como é possível ver, se fosse gerada todas os nodos para este exemplo, teríamos encontrado três soluções (nodos azul + nodo verde) para o problema, entretanto como já esperado a primeira solução encontrada (nodo verde) é a que apresenta o menor custo possível dada a entrada de exemplo.

O real desafio do trabalho é a partir deste ponto mencionado acima (branch), reduzir o número de geração de nodos para que o desempenho desta busca seja otimizada (bound).



3 Funções de Bound

Quando temos um nodo retirado da fila e que não é solução para o problema a ideia é vamos enfileirar o próximo nodo a ser visitado combinando os atores já presentes no nodo com o ator de menor custo que ainda pode se juntar aos grupo de atores do nodo atual.

Foram utilizadas de acordo com o pedido pelo enunciado do trabalho duas funções de bound com objetivos diferentes.

A primeira função de bound e escolhida como default para a solução é a seguinte:

Se o nível desse nodo novo na árvore é igual ao número de personagens, ou seja, é um nodo folha para a árvore e se os grupos do nodo pai combinado com os grupos do ator de menor custo que pode entrar no novo nodo é igual aos grupos buscados pelo problema, então essa pode ser uma solução, caso contrário, nem chegamos a visitar esse nodo.

A segunda função de bound é mais simples e simplesmente se limita a cortar os futuros nodos e sequentes ramificações da árvore que apresentam um custo maior que o custo mínimo atual, gerando menos nodos a serem visitados na fila de execução que representa a árvore.

É importante lembrar que a segunda função de bound poderia ser removida utilizando uma estratégia de early return com a primeira solução encontrada que satisfaz as condições de conter todos os grupos procurados e o número de personagens ser igual ao número de atores no nodo. Esse early return foi retirado devido a dificultar a obtenção de funções de bound, pois não seria possível remover nodos ou soluções que previamente se sabe que não seria uma resposta válida para o problema haja visto que a primeira solução encontrada que seria utilizada para ser uma referência na comparação com as próximas soluções já é a ótima.

Em relação ao desempenho das funções de bound é possível observar que para exemplos menores não se tem uma grande diferença de desempenho entre ambas, o ponto é que para exemplos maiores onde mais nodos são visitados a primeira função de bound escolhida como default apresenta um resultado significativamente melhor que a segunda, mostrando que a estratégia de cortar as ramificações da árvore pelos grupos de cada nodo é uma estratégia interessante.

4 Implementação:

Para resolver o problema proposto foi utilizado a linguagem python na v3.8 sem a utilização de bibliotecas para lidar com o algoritmo do Branch and Bound.

O código apresenta 4 classes:
classe Actor - Responsável por abstrair os atores do problema, armazenando informações referentes ao index, custo e grupos de cada ator.

classe Node - Responsável por abstrair um nodo, contendo uma lista de atores, uma lista de grupos com todos os grupos de todos os atores contidos no nodo e um custo total.

classe Problem - Responsável por abstrair o problema instanciado pela entrada e se auto resolver, gerando ou a solução ótima ou uma mensagem informando que a resolução do problema é inviável.

A lógica por trás do método resolve() desta classe é que como o próprio nome sugere ela é a responsável por resolver o problema, sua lógica é bem simples, se trata de busca em profundidade em cada ramificação da árvore partindo de um nodo raiz (fictício) para os nodos que contém somente um ator ordenados por custo de forma ascendente. A cada novo nodo retirado da fila que ainda não foi visitado é checado se este nodo é uma possível solução do problema e é melhor (caso alguma tenha sido encontrada) do que a solução atual.

classe Reader - Responsável por lidar com a entrada do problema e abstrair o input via stdin para grupos, atores e número de personagens.

O arquivo main.py contém a lógica orquestradora do problema.

Utilizar o comando 'make' no diretório raiz do trabalho para gerar o executável 'elenco'. O programa funciona lendo os dados via stdin e com a opção '-a' durante a utilização do executável a bound function alternativa é utilizada.

No diretório examples/test/ encontra-se diversos exemplos utilizados durante a fase de testes com as respectivas respostas esperadas para cada um.

5 Conclusão:

O objetivo do trabalho era apresentar uma solução otimizada para o problema do Elenco Representativo utilizando o algortimo de Branch and Bround fazendo o uso de duas funções de bound distintas. Dado este objetivo, acredito que foi obtido com exito uma versão final com desempenho relevante, principalmente a considerar a questão de nodos visitados e tempo de execução, claro isso se deve a ordenação dos atores disponíveis a cada nodo por custo dos atores, o que leva sempre a acharmos a melhor solução possível já na primeira solução