Otimização - Trabalho 2

Jorge Lucas Vicilli Jabczenski[1]
Vinicius Tikara Venturi Date[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)

Resumo. *Relatório sobre o segundo trabalho da disciplina de otimização, que tem como objetivo explicar, modelar e resolver o problema enunciado, o problema da mochila químida, utilizando a técnica de Branch & Bound.*

1. Descrição do problema

O problema é similar ao problema da mochila, isto é, consiste em: dada uma lista de itens com seus valores e pesos, selecionar quais são os mais valiosos e cabem na mochila ao mesmo tempo, tendo para todo item, a escolha de colocar ou não colocar na mochila, uma escolha binária. Entretanto há uma restrição a mais em relação ao problema original, certos pares de itens não podem ser selecionados ao mesmo tempo, pois reagem entre si, o que impossibilitaria a escolha de ambos ao mesmo tempo. Em resumo, é um problema da mochila com mais um tipo de restrição.

2. Modelagem

Como visto na seção anterior, o problema a ser resolvido é o problema da mochila, exceto que existe uma restrição a mais, pares de itens que não podem estar juntos numa mesma escolhas. Portanto, é possível utilizar as mesmas ideias do problema da mochila original, em especial, como é usado o *backtracking* para resolver este problema.

2.1. Cortes de galhos da árvore de escolhas

Podemos checar se o item a ser colocado inviabiliza a solução, e para tanto podemos checar duas situações:

Uma delas é checar se o peso ultrapassará a capacidade da mochila ao ser adicionado o item atual, mesma ideia do problema da mochila original.

O outro corte é baseado em checar se ocorre alguma reação do item a ser escolhido com os demais já presentes na mochila, pois se for o caso, o ladrão não tem porque tentar uma escolha dessas.

Esse dois cortes conseguem evitar as escolhas impossíveis, sem a necessidade de chegar nas folhas da árvore de decisões.

2.2. Função limitante

O objetivo da função limitante é gerar uma estimativa para a resolução do problema, com duas importantes características:

1. Ter um valor sempre maior que o valor do problema para uma dada entrada
2. Ter uma baixa complexidade de tempo do que a do problema, pois é necessário o uso extensivo dela durante a execução do programa.

Conseguimos reutilizar a função limitante do problema da mochila original, pois, tendo a mesma lista de itens do problema da mochila química, o problema da mochila original sempre retorna um valor maior, pois as restrições de pares nos impedem de escolher certos itens que, sem a restrição, seriam os melhores no momento. Portanto, como a mochila racional limita por cima a mochila original, ela deve limitar a mochila química da mesma forma.

O problema da mochila racional é igual ao problema da mochila, com uma pequena mudança, é possível escolher frações dos itens. Como explicado no livro de referência da disciplina, a mochila racional utiliza uma estratégia gulosa para resolver o problema. Por ser um algoritmo guloso, é também de fácil computação, pelo menos quando comparado com o *backtracking*, por fim, atendendo os dois requisitos para uma função limitante.

Com a função limitante nós podemos entender que ela realiza cortes da mesma forma que as checagens simples explicadas anteriormente fazem, entretanto agora os cortes são de possíveis ramos, mas que não trariam nenhum aumento no ótimo da função. Noutras palavras, a função limitante nos permite rejeitar escolhas sem necessariamente ter que chegar nas folhas da árvore de decisão.

2.3. Ordenação

A ordenação é feita a partir da função limitante aplicada às duas escolhas, colocar ou não colocar o item na mochila, e então ordenando o valor retornado por elas, de forma decrescente.

O propósito da ordenação é percorrer as escolhas de forma ótima, pois conseguimos decidir qual dos caminhos tomar primeiro, aquele de maior possível ganho. Descartamos galhos que não tem chance de serem ótimos, igual na seção anterior, mas de forma mais esperta, se a escolha maior das duas já tem um limite menor que o ótimo encontrado até o momento, ambas escolhas podem ser descartadas.

3. Implementação

Foram utilizadas variáveis globais para o controle do status da mochila entre chamadas da função de resolução do problema, como o controle de capacidade máxima, o valor ótimo até o momento e os itens escolhidos até o momento.

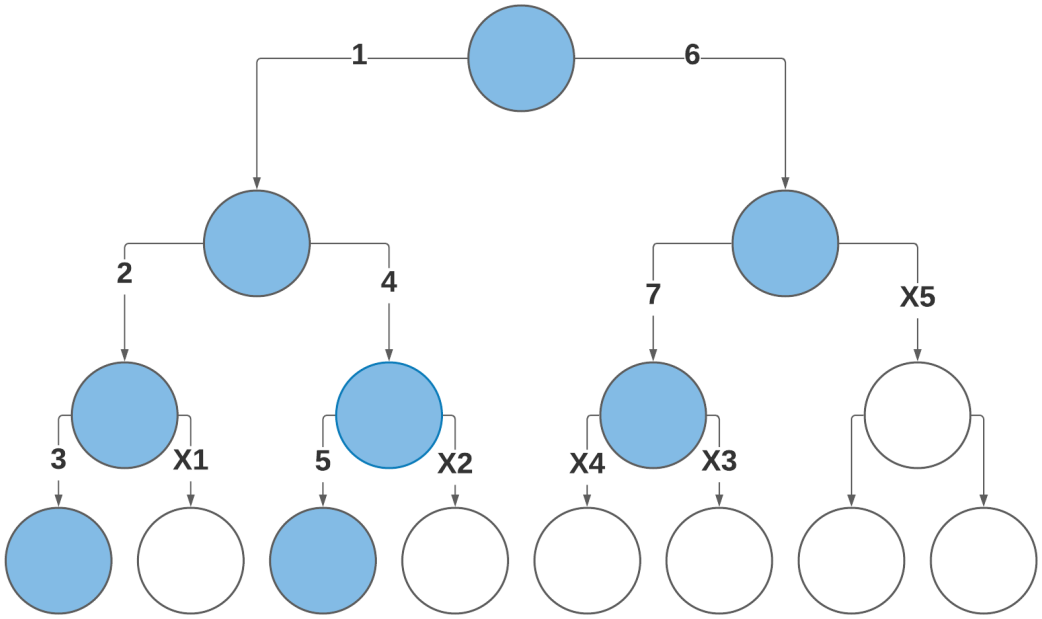
Para os pares nós criamos uma lista que guarda a informação de itens que interagem com entre si, par a par. Essa lista é percorrida quando precisamos checar se um dado item pode ser colocado na mochila, a lista é percorrida tendo o índice em mãos, e, se ele faz par com algum item da lista, não é possível sua inclusão nesta escolha.

A função limitante não teve mudanças, ela é implementada similarmente àquela descrita no livro, a função da mochila racional, como descrita anteriormente.

A função de resolução segue o esquema de *backtracking* com *Branch & Bound* visto no livro, com algumas mudanças, visto que só existe duas escolhas para serem feitas, e uma delas não adiciona nada à lista que está sendo montada.

Foi feito também um programa simples que gera entradas para serem utilizadas pelo programa principal, para o auxílio nos testes do programa principal.

3.1. Exemplo de execução do primeiro exemplo



No exemplo, as arestas estão numeradas na ordem em que o algoritmo faz as chamadas recursivas.

O algoritmo decide não colocar tanto o primeiro item[1] quanto o segundo[2], pois a função limitante utiliza o espaço da mochila para colocar frações do terceiro item, testando essa possibilidade antes. Ao chegar no nó após a segunda escolha[2], observa-se que não é possível inserir o terceiro item na mochila[X3], pois ele ultrapassa a capacidade máxima, terminando então na folha de nenhum item na mochila[3], com um lucro de 0, pois não temos lucro ótimo maior que zero ainda.

Agora, volta na árvore de decisões, e é escolhido inserir o segundo item na mochila[4], pois este item tem um valor de limite maior que o lucro ótimo do momento, cabe na mochila e não reage com nada da mochila até o momento. De forma semelhante, o terceiro item não pode ser inserido na mochila por ultrapassar o limite máximo[X2]. Resta então escolher não selecionar o terceiro item[5], obtendo uma nova escolha ótima, apenas o item 2 com lucro de 10.

Por fim, volta-se até o começo da árvore de decisões, e agora vemos o que acontece se colocarmos o primeiro item na mochila[6]. De forma semelhante, é escolhido não tomar o item 2 neste momento pois a função limitante desta escolha é maior[7], por causa da possibilidade do item 3. E, como acontecia anteriormente, não é possível inserir o terceiro item por causa do peso[X3], e também não é possível não escolhe-lo[X4], pois o retorno da função limitante neste caso (1) é menor do que o lucro ótimo até o momento (10).

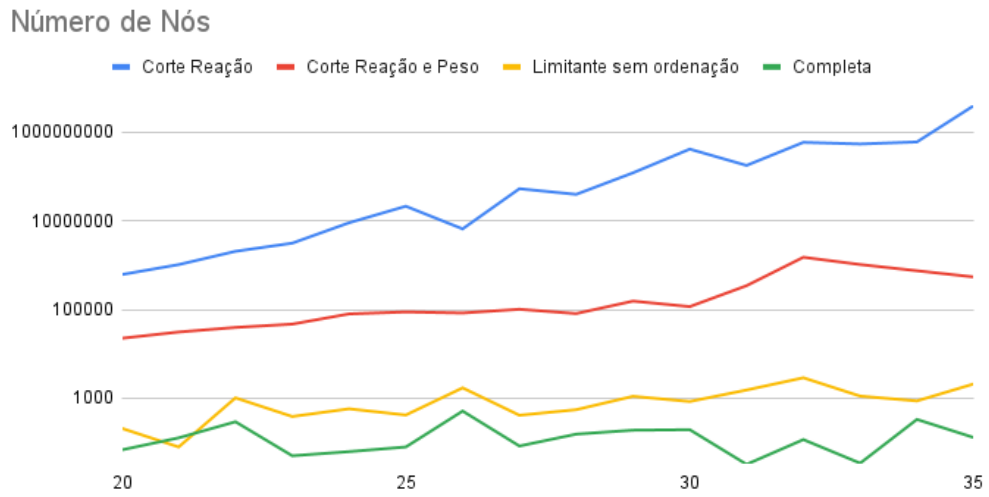
Finalizando temos uma última possível escolha, colocar o segundo item na mochila, entretanto ele reage com o primeiro item, impossibilitando esse caminho[X5].

Neste exemplo temos todos conceitos explicados na modelagem - cortes, função limitante e ordenação - evitando as chamadas fúteis da função

4. Análise da efetividade do Branch & Bound

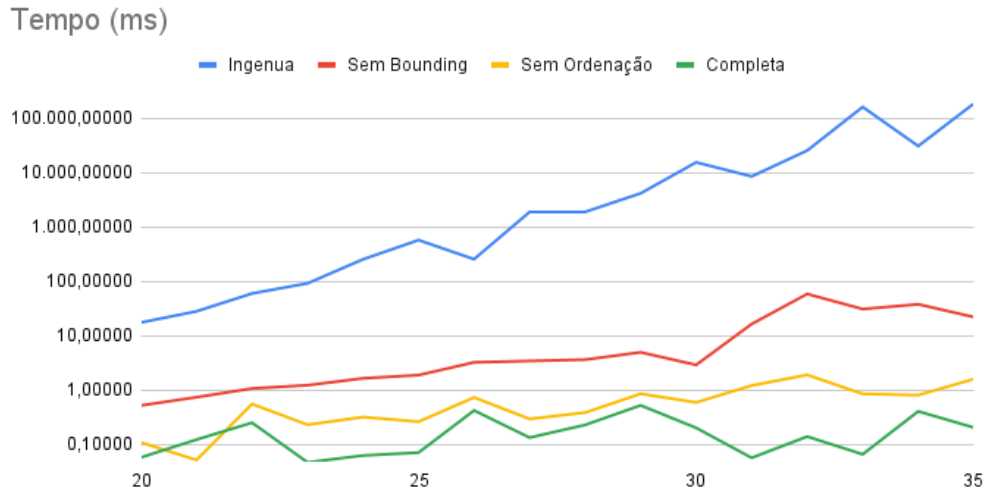
São comparadas as funções nas seguintes formas:

1. Apenas checando a reação dos itens a cada escolha
2. Com ambos cortes, de peso e reação
3. Com os cortes e função limitante
4. Com todos conceitos anteriores e ordenação, o *Branch & Bound* de fato.



A primeira vista, é possível ver que os cortes simples já nos trazem alguns ganhos em cima da implementação ingênua, entretanto o ganho maior é observado ao utilizar a função limitante, ela por si já diminui algumas ordens de grandeza dos nós da árvore de escolhas.

A perceber, a ordenação não nos traz tanta diferença de desempenho quando comparado à diferença entre cortes e limitante, mas ainda sim ocorre diferença de perto de uma ordem de grandeza, mostrando sua efetividade por direcionar as escolhas mais propensas ao sucesso em sua lógica.



O tempo segue em função do número de nós, uma vez que cada nó é uma chamada nova da função, tendo, portanto, um formato similar ao gráfico anterior.

5. Referências

Todo conteúdo foi baseado no livro referência da disciplina, *Combinatorial Algorithms Generation Enumeration and Search*.