Universidade Federal da Paraíba
Centro de Informática
Disciplina: Análise e Projeto de Algoritmos
Professor: Bruno Bruck

Projeto Final

Descrição do problema

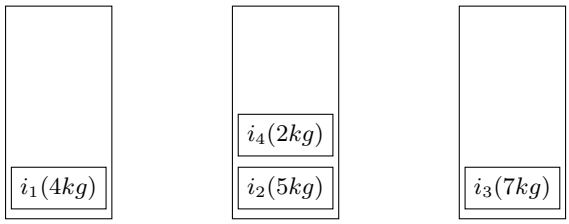
Estamos nos aproximando da época do Natal e o Papai Noel está muito atarefado preparando todos os presentes. Além de gerenciar a fabricação, ele ainda tem que tomar uma série de decisões logísticas para garantir que todas as crianças recebam seu presente. Visto que ele já tem muito o que fazer, são seus ajudantes que ficam encarregados por realizar as entregas utilizando uma frota de trenós. Como os trenós são extremamente rápidos o maior problema não é a rota de entrega, mas sim decidir quais presentes serão carregados em cada um dos trenós. O problema fica mais complexo pelo fato de que existem presentes que não podem ser carregados juntos por questões variadas (como, por exemplo, fragilidade ou regiões muito diferentes de entrega).

Neste ano, o Papai Noel encarregou um de seus ajudantes de pesquisar alguma forma de resolver o problema de forma mais otimizada e automatizada. Depois de algum tempo pesquisando, o ajudante ficou sabendo da existência de algoritmos de otimização e sugeriu ao Papai Noel que isso poderia ser uma solução interessante. Após contratar um consultor especializado, o problema foi formalizado da seguinte forma:

> " Seja I o conjunto de presentes que devem ser empacotados e k o número de trenós disponíveis. Cada presente $i \in I$ possui um determinado peso p_i (em kg) e cada trenó possui uma capacidade máxima Q (em kg). Devido ao fato de que alguns presentes não podem ser empacotados no mesmo trenó, é fornecida uma lista L que contem todos os pares de presentes (i, j), tal que $i, j \in I$, que são incompatíveis entre si e que devem ser carregados em trenós diferentes. O objetivo do problema é empacotar todos os presentes de forma a minimizar o número de trenós utilizados ao mesmo tempo em que são respeitadas as restrições de capacidade e incompatibilidade. "

Exemplo de instância e solução

Para exemplificar o problema, considere uma instância (cenário) onde existem um total de 4 presentes. (ou seja, $I = \{i_1, i_2, i_3, i_4\}$), 2 trenós, $p = [4, 5, 7, 2]$, $Q = 10$ e $L = \{(i_1, i_2), (i_1, i_4)\}$. Sendo assim, de acordo com a lista L, os presentes i_1 e i_2 não podem ser colocados no mesmo trenó e nem os presentes i_1 e i_4. Nesse cenário, uma possível solução seria como a seguir:



Note que, se levássemos em conta somente a restrição de capacidade, seria possível utilizar 2 trenós. Entretando, por conta das restrições de incompatibilidade fomos obrigados a utilizar um total de 3 trenós.

Instruções

O projeto deve ser realizado em grupo de **3 integrantes** e vale 10 pontos, relativos à terceira nota da disciplina. Cada grupo deve desenvolver um algoritmo eficiente de busca local (ou meta-heurística) para o problema de otimização descrito acima. O código-fonte deve ser **obrigatoriamente** escrito na linguagem **C/C++**.

Note que o seu programa deve ser capaz de ler um arquivo contendo os dados de uma instância (cenário) do problema e utilizar tais dados como entrada para o algoritmo. O formato de arquivo a ser utilizado é o seguinte:

```
numero_presentes
k
Q
numero_elementos_em_L

array p

lista L
```

A instância utilizada na seção anterior, por exemplo, poderia ser representada pelo seguinte arquivo:

```
4
2
10
2

4 5 7 2

1 2
1 4
```

Etapas e prazos

Este projeto contém os seguintes entregáveis:

- Implementação de **ao menos uma heurística de construção**, que nada mais é do que um **algoritmo guloso** para geração de uma solução viável.
- Implementação de **pelo menos 2 movimentos de vizinhança**.
- Implementação do algoritmo de busca local chamado VND (Variable Neighborhood Descent).
- Implementação de uma meta-heurística (OPCIONAL). Sugestões: GRASP ou ILS.
- Resultados computacionais: **criar uma tabela** que contenha os resultados obtidos pela(s) heurística(s) construtiva(s) e pelo VND, e que compare tais resultados com a solução ótima de cada instância. Essa tabela deverá conter os seguintes dados para cada heurística construtiva e para o VND:
  - Média do valor da solução (em no **mínimo 10 execuções** para cada instância caso exista algum fator aleatório no algoritmo)
  - Melhor solução encontrada
  - Média do tempo gasto pelo respectivo algoritmo
  - GAP para a solução ótima

Caso decida implementar a meta-heurística, é necessário adicionar uma coluna de resultados para ela na tabela.

O prazo de entrega do projeto é até às **23:59 do dia 11 de Dezembro de 2022**. Devem ser enviados via SIGAA, o código-fonte do projeto e um relatório em *pdf* contendo o nome dos integrantes do grupo e a tabela de resultados computacionais.

Avaliação

Cada grupo deverá apresentar presencialmente o projeto em data a ser agendada pelo professor. A nota do projeto é individual e leva em consideração diversos critérios, como demonstração de entendimento do código na apresentação, qualidade do código, eficiência dos algoritmos implementados, qualidade dos resultados obtidos, dentre outros. Não apresentar o projeto implica em nota zero.

Dicas

Como calcular o valor da medida GAP: Suponha que desejamos calcular o valor GAP para o resultado da heurística construtiva para a instância chamada *nome_instancia*. Supondo que o valor encontrado pela heurística para essa instância é dado por $valor_{heuristica}$ e o valor ótimo para essa instância é $valor_{otimo}$, o cálculo do GAP é realizado da seguinte forma:

$$gap = \left(\frac{valor_{heuristica} - valor_{otimo}}{valor_{otimo}} \right) \times 100$$

Note que o valor do gap é dado em percentagem (%) e indica a "distância" da solução, no caso, da heurística construtiva para o valor ótimo.

Para calcular o GAP dos resultados obtidos pelo VND basta substituir $valor_{heuristica}$ pelo valor encontrado pela VND.

Exemplo de tabela de resultados:

		Heurística construtiva			VND		
	ótimo	valor solução	tempo	gap	valor solução	tempo	gap
instancia1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia4	0.0	0.0	0.0	0.0	0.0	0.0	0.0