Universidade Estadual do Oeste do Paraná – UNIOESTE
Campus de Cascavel
Colegiado de Ciência da Computação

Otimização Combinatória

Trabalho 2
Busca Tabu, Simulated Anealling e Colônia de Formigas

Conforme combinado, cada grupo ficará responsável pelo desenvolvimento de uma abordagem de otimização aplicada a um cenário específico.

Sobre a implementação

1. Não podem ser usadas soluções prontas. Cada equipe deve implementar sua própria resolução para a técnica.
2. Deve-se exibir na interface da implementação as soluções parciais do método em execução. Através da visualização deve ser possível acompanhar a evolução da técnica.
3. Para implementar a interface pode-se empregar métodos e modelos prontos.
4. Linguagens que podem ser usadas: C, C++, Java e Pyhton
5. Deve-se enviar o projeto construído contento todos os códigos fonte construídos.
6. A entrega do trabalho deve ser feita via Teams.
7. Cada equipe deverá submeter um arquivo zipado chamado “Grupo n” contendo todos os arquivos especificados anteriormente.

DATA DE ENTREGA: XX/XX/2024 até as 23:59

Atentem-se pois não serão aceitos trabalhos entregues após o prazo!

Membros	Grupo	Técnica	Problema
Gabriel Alves Mazucco Gabriel Jared de Barros Amorim Gabriel Pereira	4	Busca Tabu	Problema do Caixeiro Viajante
Eduardo Cozer Geandro Ribeiro da Silva Vinicius Messaggi de Lima Ribeiro	1	Simulated Anealling	Problema da Mochila Binária
Guilherme Vier Jackson Renato Faquineti Rampazzo	7	Busca Tabu	Problema da Designação Generalizada
Eduardo Dias Machado Matheus Henrique Gallert Lucas Ivanov Costa	6	Simulated Anealling	Problema do Empacotamento Unidimensional começando com pior solução
João Gabriel Fazio Pauli Fernando Seiji Onoda Inomata Lucas Batista Deinzer Duarte	13	Busca Tabu	Problema da Mochila Binária
Michel Caesar Koval de Oliveira Matheus Centenaro	14	Simulated Anealling	Problema do Caixeiro Viajante
Weberson Morelli Leite Junior Kawan de Oliveira Gabriel Cezimbra	2	Busca Tabu	Problema do Empacotamento Unidimensional começando com pior solução
Renan Valduga Kafer Erik Felipe Olinek de Castilho Victor Negri Bergamo Lopes da Silva	8	Simulated Anealling	Problema da Designação Generalizada
Eric Klaus Brenner Melo e Santos Ruan Rubino de Carvalho Matheus Rogério Pesarini	5	Simulated Anealling	Problema do Empacotamento Unidimensional começando com solução gulosa
Carlos Eduardo Pagani Grosso Rafael Pascoali Czerniej Vitor Mayorca Camargo	12	Colônia de Formigas	Problema do Caixeiro Viajante
Matheus Henrique Gaspar Gabriel Yudi Leite Higuchi	10	Busca Tabu	Problema do Empacotamento Unidimensional começando com solução gulosa
Bruno Henrique de David Glixinski Monique Barros Ryan Hideki Inoue Matsunaga Pereira	3	Simulated Anealling	Problema do Caixeiro Viajante
André Henrique dos Santos da Silva Nathan Luiz Silva Oliboni Vinicius Eduardo Moraes de Oliveira	11	Colônia de Formigas	Problema de Roteamento
Guilherme Augusto Deitos Alves Vinicius Vieira Viana Felipe Kravec Zanatta	9	Simulated Anealling	Problema da Designação Generalizada

Mochila Binária
(Grupos 1 e 13)

Entrada: Arquivos em formato txt contendo a capacidade da mochila e uma lista de itens contendo o custo e o benefício de cada um. O formato do arquivo é apresentado na Figura 1.

105	Capacidade da Mochila
3 42 5 48 42 13 3 20 12 37	Benefícios dos Itens
2 35 13 29 9 25 2 14 4 17	Custos dos Itens

Figura 1. Configuração do arquivo de entrada para o problema da mochila binária

No próprio nome do arquivo é apresentada a quantidade de itens presentes. Por exemplo, o arquivo "Mochila10.txt" apresenta a capacidade da mochila e as informações de custo e valor de dez itens.

Arquivos de entrada disponíveis em:
https://drive.google.com/file/d/1hHgX7rv2EivKS2SHHFuZhJFufEidWQmK/view?usp=sharing

Problema do Caixeiro Viajante e de Roteamento
(Grupos 4, 14, 12, 3 e 11)

Entrada: Arquivos em formato txt contendo o número de vértices que compõe o grafo e a sua devida lista de adjacências, contendo o peso das arestas entre cada par de vértices.

- Valores iguais a zero indicam a ausência de conexão entre os vértices;
- Os grafos disponíveis no link são ponderados;
- Os grafos disponíveis no link são não direcionados.

O formato de cada arquivo é apresentado na Figura 2. Na ilustração cada linha representa o custo de cada aresta a partir do vértice de origem até o destino. Por exemplo, na primeira linha temos o valor 6. Ele indica que o custo para ir do vértice 1 até o quinto vértice é igual a seis.

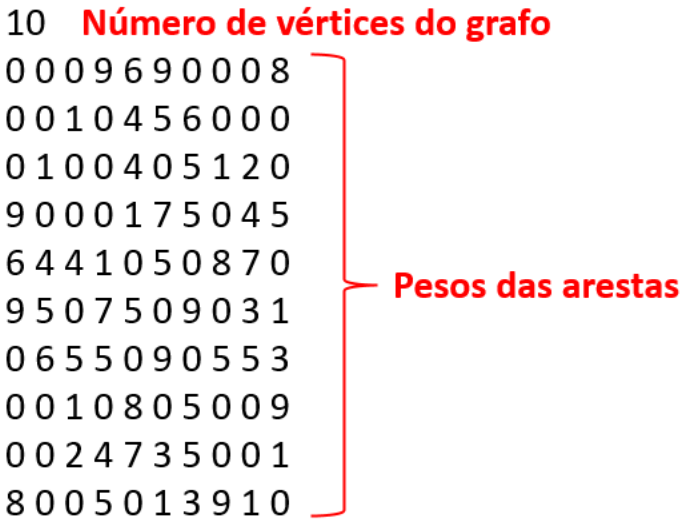

Figura 2. Estrutura dos arquivos de entrada para os problemas TSP e de Roteamento

A matriz indicada na Figura 2 refere-se ao grafo apresentado a seguir.

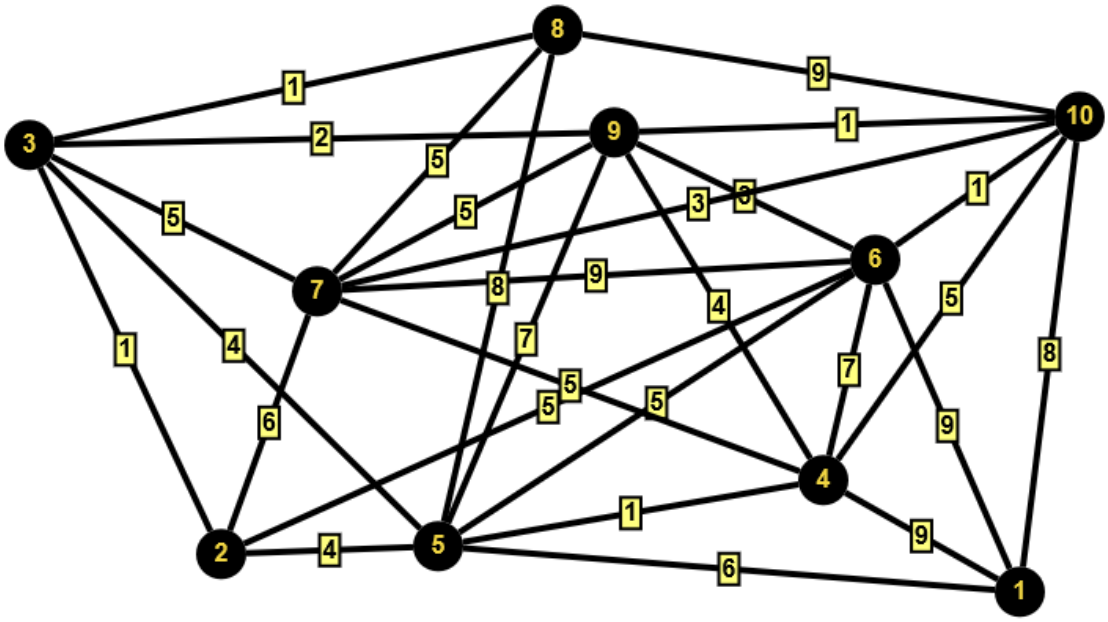

Figura 3. Grafo construído a partir do arquivo de entrada apresentado na Figura 1

A entrada dos dados para a construção dos grafos deve ser feita com base nos arquivos texto disponíveis em: https://drive.google.com/file/d/1EMjUKBrOclz3MUTcU-ifyVP5Vuu-ogZE/view?usp=sharing

Problema de Designação Generalizada
(Grupos 7, 8 e 9)

Entrada: Arquivos em formato txt contendo na primeira linha o número (NP) de programadores disponíveis para o trabalho. Na linha seguinte é apresentada a quantidade de módulos (NM) que precisam ser desenvolvidos. Nas NP linhas seguintes são apresentados os custos de cada programador para cada módulo. Cada coluno refere-se a um dos NM módulos. Em seguida são apresentadas NP linhas com as cargas horárias gastas por cada programador para desenvolver cada um dos NM módulos. Na última linha do arquivo temos NP colunas que se referem à cada horária que cada programador tem disponível para a tarefa.

4	**Número de programadores**							
8	**Número de módulos a serem desenvolvidos**							
7	7	10	8	16	16	0	17	**Custo de Execução por cada programador por módulo**
10	5	9	9	14	4	16	11	
11	8	7	5	1	11	20	12	
5	7	6	8	16	7	15	17	
10	14	16	12	8	20	10	16	**CH gasta por cada programador Para cada módulo**
10	14	16	12	8	20	10	16	
10	14	16	12	8	20	10	16	
10	14	16	12	8	20	10	16	
30	25	20	40	**CH disponível por programador**				

Figura 4. Formato do arquivo de entrara para o Problema de Designação Generalizada

Na Tabela 1 estão representados os custos para execução de cada módulo por cada desenvolvedor. O objetivo será escolher quem deve desenvolver cada módulo de forma que seja gasto o menor valor possível.

		Módulos a serem desenvolvidos							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Programadores	**1**	7	7	10	8	16	16	0	17
	2	10	5	9	9	14	4	16	11
	3	11	8	7	5	1	11	20	12
	4	5	7	6	8	16	7	15	17

Tabela 1. Custos dos módulos ao serem desenvolvidos por cada programador

Na Tabela 2 são apresentados quantas horas cada programador gasta para trabalhar em cada um dos oito módulos a serem desenvolvidos.

Programadores	Módulos a serem desenvolvidos							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	10	14	16	12	8	20	10	16
2	10	14	16	12	8	20	10	16
3	10	14	16	12	8	20	10	16
4	10	14	16	12	8	20	10	16

Tabela 2. Carga horária necessária para desenvolver cada módulo

Na Tabela 3 são apresentadas as quantidades de carga horária que cada programador dispõe para desenvolver os módulos requisitados. A solução final não deve permitir que nenhum dos quatro programadores trabalhe além da carga horária apresentada na tabela.

Programadores			
1	**2**	**3**	**4**
30	25	20	40

Tabela 3. Número de horas que cada programador tem disponível

Os arquivos de entrada para testes da implementação podem ser acessados no link: https://drive.google.com/file/d/1inILq2pWgxelLXZ6nJLdjFlbgR3WrTwB/view?usp=sharing

Problema do Empacotamento Unidimensional
(Grupos 6, 2, 5 e 10)

Entrada: Arquivos em formato txt (conforme apresentado na Figura 6) contendo na primeira linha a capacidade que cada recipiente é capaz de armazenar. Na segunda linha é identificado o número de itens que devem ser alocados na menor quantidade possível de recipientes. Na terceira linha são apresentados os tamanhos dos itens que devem ser alocados.

```
10    → Capacidade dos recipientes
8     → Número de itens a serem alocados
1 3 2 4 8 5 7 6    → Tamanho de cada um dos itens presentes
```

Figura 6. Estrutura do arquivo de entrada para o problema do empacotamento.

A tarefa consiste em, dada a capacidade de cada recipiente e as características dos itens, identificar qual o menor número de recipientes necessário para guardar todos os itens presentes.

No cenário em que o método começa com a **pior solução** possível cada item é acomodado em um recipiente. Dessa forma, a solução inicial demanda a maior quantidade de recipientes possível.

No cenário em que o método começa com uma **solução gulosa** para se obter a primeira solução válida deve-se rodar o um algoritmo guloso com o seguinte comportamento (Figura 7):

```
Enquanto ainda houver itens para alocar
{
        Retira de forma aleatória um item do conjunto
        Enquanto houver recipiente para testar
        {
                Se o tamanho do item for menor que o espaço vago no recipiente
                {
                        Coloca o item naquele recipiente
                        Sai do enquanto
                }
                Avança para o próximo recipiente disponível
        }
        Se o item não pôde ser guardado
        {
                Alocar um novo recipiente
                Atribuir o item em questão para o novo recipiente
        }
}
```

Figura 7. Pseudocódigo do algoritmo guloso para encontrar uma primeira solução válida para o problema do empacotamento.

Os arquivos de entrada para os testes podem ser acessados no endereço: https://drive.google.com/file/d/1B76Lx2bhhFckTIpUnz069ljCmf3kxQN_/view?usp=sharing