



## Otimização Combinatória

### **Trabalho 2** **Busca Tabu, Simulated Annealing 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 Python
5. Deve-se enviar o projeto construído contendo 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: 14/02/2023 até as 23:59**

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

Membros	Grupo	Técnica	Problema
Augusto Barella Dal Pra Gustavo Portela Rauntenberg João Vitor Biederman	1	Busca Tabu	Problema do Caixeiro Viajante
Gabriel Lenser Gabriel Tadioto Oliveira Gustavo Alberto Ohse Hanke	2	Simulated Annealing	Problema da Mochila Binária
Gabriel Andrade de Araújo Roberval Requião Junior	3	Busca Tabu	Problema da Designação Generalizada
Fabio Novack da Silva Eduardo Pimentel dos Santos Gustavo Magalhães Faino	4	Simulated Annealing	Problema do Empacotamento Unidimensional começando com pior solução
Leonardo Bednarczuk Balan de Oliveira Gabriel Santos da Silva David Antonio Brocardo	5	Busca Tabu	Problema da Mochila Binária
Luiz Fernando Becher de Araujo Vinicius Sendoski de Andrade	6	Simulated Annealing	Problema do Caixeiro Viajante
Luiz Felipe Fonseca Rosa Leonardo Calsavara Pedro Henrique Ferreira Zoz	7	Busca Tabu	Problema do Empacotamento Unidimensional começando com pior solução
Isadora Coelho Araújo Maria Eduarda Crema Carlos Pedro Lucas Moraes	8	Simulated Annealing	Problema da Designação Generalizada
Erik Felipe Olinek de Castilho Jonathan Santos Tadei	9	Busca Tabu	Otimização de Circuitos (Conectividade)
Pedro Hernesto Pissetti Venzke João Luiz Reolon Gabriel Belinski Salomão	10	Simulated Annealing	Problema do Empacotamento Unidimensional começando com solução gulosa
Marlon Fabichacki Pereira Ronaldo Drecksler Farias Pachico Vinicius Visconsini Diniz	11	Colônia de Formigas	Problema do Caixeiro Viajante
Fernando Schumaker Fiedler Fabricio Marcos Cordeiro Luiz Eduardo Garzon	12	Busca Tabu	Problema do Empacotamento Unidimensional começando com solução gulosa
Bruno Stafuzza Maion Rafael Roberto Hoffmann Lucca Abbado Neres	13	Simulated Annealing	Problema do Caixeiro Viajante
Gabriel Neneve dos Santos Gabriel Rodrigues dos Santos Juliano Augusto da Silva	14	Simulated Annealing	Otimização de Circuitos (Conectividade)
Matheus Centenaro Pedro Henrique de Oliveira Berti Renan Valduga Kafer	15	Simulated Annealing	Problema da Designação Generalizada

## Mochila Binária (Grupos 2 e 5)

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.

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

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 1, 6, 11 e 13))

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.

10	Número de vértices do grafo
0 0 0 9 6 9 0 0 0 8	<div style="color: red; font-weight: bold;">Pesos das arestas</div>
0 0 1 0 4 5 6 0 0 0	
0 1 0 0 4 0 5 1 2 0	
9 0 0 0 1 7 5 0 4 5	
6 4 4 1 0 5 0 8 7 0	
9 5 0 7 5 0 9 0 3 1	
0 6 5 5 0 9 0 5 5 3	
0 0 1 0 8 0 5 0 0 9	
0 0 2 4 7 3 5 0 0 1	
8 0 0 5 0 1 3 9 1 0	

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.

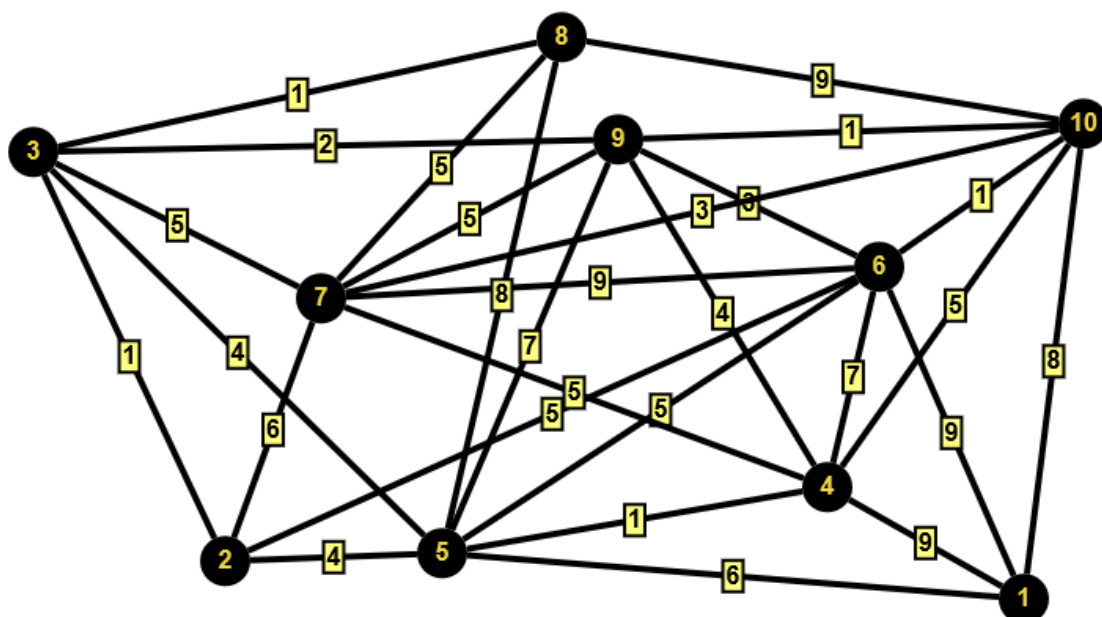


Figura 3. Grafo construído a partir do arquivo de entrada apresentado na Figura 1

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 3, 8 e 15)

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 hora que cada programador tem disponível para a tarefa.

4	<b>Número de programadores</b>						
8	<b>Número de módulos a serem desenvolvidos</b>						
7	7	10	8	16	16	0	17
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
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	<b>CH disponível por programador</b>			

} **Custo de Execução por cada programador por módulo**  
 } **CH gasta por cada programador Para cada módulo**

Figura 4. Formato do arquivo de entrada 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.

		<b>Módulos a serem desenvolvidos</b>							
		<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
<b>Programadores</b>	<b>1</b>	7	7	10	8	16	16	0	17
	<b>2</b>	10	5	9	9	14	4	16	11
	<b>3</b>	11	8	7	5	1	11	20	12
	<b>4</b>	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.

		Módulos a serem desenvolvidos							
		1	2	3	4	5	6	7	8
Programadores	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 da Conexão de Circuitos (Grupos 9 e 14)

Entrada (Figura 5): Arquivos em formato txt contendo na primeira linha o número de componentes que devem ser conectados. Na segunda linha é identificado o número exato de conexões que devem ser estabelecidas no circuito. Na terceira linha é apresentado o número máximo de conexões que cada componente pode realizar.

Nas linhas 4 e 5 são apresentadas as coordenadas de cada componente. Na quarta linha são apresentadas as posições no eixo X enquanto na última linha são detalhadas as posições sobre o eixo Y.

Sabendo-se cada componente deve ter no mínimo uma conexão, encontre a melhor combinação de conexão entre os elementos presentes no arquivo de entrada de forma que a soma dos cabos para estabelecer tais conexões seja o menor possível.

05	→	Número de componentes a serem conectados
08	→	Número de conexões que deve ser estabelecido em todo o circuito
04	→	Número máximo de conexões que um componente pode realizar
12,1 14,7 9,0 4,3 1,2	→	Posição no eixo X de todos os componentes
15,0 10,0 2,4 2,1 0,8	→	Posição no eixo Y de todos os componentes

Figura 5. Estrutura do arquivo de entrada para o problema de otimização de circuitos

Os arquivos de entrada para os testes podem ser acessados no endereço:  
<https://drive.google.com/file/d/1qazkugQ3v9sxENY1DILGIFMHjjO2RC02/view?usp=sharing>



## Problema do Empacotamento Unidimensional (Grupos 4, 7, 10 e 12)

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=s\\_haring](https://drive.google.com/file/d/1B76Lx2bhhFckTIpUnz069ljCmf3kxQN/view?usp=s_haring)