Busca Tabu, Colônia de Formigas e Recozimento Simulado
Otimização Combinatória
2024

André Luiz Brun[1]

[1]Colegiado de Ciência da Computação
Campus de Cascavel - UNIOESTE

Resumo. *Este documento consiste na especificação formal do segundo trabalho da disciplina de Otimização Combinatória (Csc3040) para o ano letivo de 2024. Aqui são descritas as atividades a serem desenvolvidas e como cada processo deverá ser realizado. Além disso, o documento contém as informações sobre a formação das equipes, o objeto de trabalho de cada uma e as datas de entrega dos relatórios.*

1. Regras

Para que o trabalho seja realizado, alguns pontos devem ser levados em conta durante o desenvolvimento. A seguir são listadas algumas regras que devem, obrigatoriamente, serem seguidas pela equipe ao desenvolver o trabalho.

➡ Cada equipe deve implementar o método de otimização seguindo as especificações que lhes foi definida. A construção de soluções que não sigam tal determinação serão consideradas erradas;

➡ Não devem ser utilizados métodos prontos para a construção do método de otimização. A adoção de códigos prontos (disponíveis em repositórios como Github, Bitbucket, ou mesmo em páginas) implicará na atribuição de nota zero ao trabalho;

➡ Métodos auxiliares disponíveis nas linguagens podem ser usados na implementação. Por exemplo, funções para ordenação, exibição, salvamento de arquivos, podem ser adotadas;

➡ As linguagens em que o trabalho pode ser desenvolvido são: C, C++, Python e Java. Não serão aceitos trabalhos desenvolvidos em outras linguagens;

➡ Os parâmetros empregados ao longo da execução devem ser definidos a partir dos arquivos de entrada disponíveis;

➡ Os critérios de parada podem variar de acordo com a implementação. Poderão ser utilizados como critério um número pré determinado de iterações ou através da convergência do algoritmo, definida a partir de uma quantidade de iterações sem melhora no desempenho. O número de iterações, quando adotado, deve ser grande suficiente para que os métodos possam escapar de mínimos ou máximos locais.

➡ A data de entrega é dia dd/mm/aaaa até as 23:39 por meio da tarefa disponível na turma da disciplina no Teams.

2. Caracterização dos arquivos de entrada e resultados esperados

2.1. Mochila Binária

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 Tabela 1.

No próprio nome do arquivo é apresentada a quantidade de itens presentes. Por exemplo, o arquivo "Mochila10.txt" (ilustrado na Tabela 1) apresenta a capacidade da mochila e as informações de custo e valor de dez itens. A primeira linha conterá a capacidade da mochila. Na segunda linha estarão contidos os benefícios dos n itens. Por fim, os custos de cada um dos n itens estarão dispostos na terceira linha do arquivo.

Tabela 1. Exemplo de arquivo de entrada para o problema da mochila binária

Capacidade da Mochila	105									
Benefício dos itens	3	42	5	48	42	13	3	20	12	37
Custo dos itens	2	35	13	29	9	25	2	14	4	17

Download: Arquivos de entrada disponíveis no link: download

Saída: Conjunto de itens que serão colocados na mochila e valor do benefício obtido ao se adicionar tais itens.

2.2. Problema do Caixeiro Viajante e de Roteamento

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 Tabela 2. Como é evidenciado a primeira linha contém o número (n) de vértices presentes na estrutura. As n linhas seguintes contêm os custos 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.

Apenas para critério de ilustração, o arquivo de entrada representado na Tabela 2 é apresentado na Figura 1. Na grafo é possível verificara que o vértice 1 está conectado ao vértice 5 e que o custo dessa aresta é de 6 unidades (parte inferior do grafo).

Tabela 2. Exemplificação da estrutura de um arquivo de entrada para o TSP e Roteamento

Número de Vértices	10									
Distâncias entre os vértices	0	0	0	9	6	9	0	0	0	8
	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

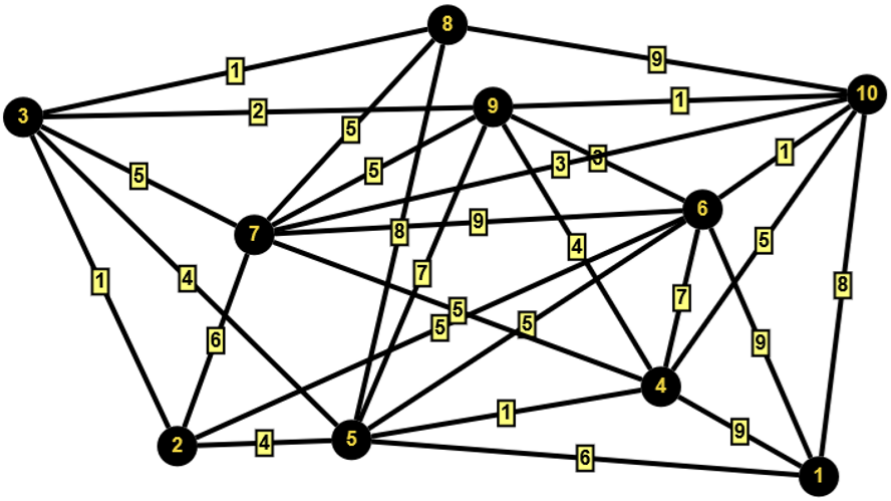

Figura 1. Grafo construído a partir das conexões presentes no arquivo de entrada da Tabela 2

Download: A entrada dos dados para a construção dos grafos deve ser feita com base nos arquivos texto disponíveis em no link: downlad.

Saída: A rota encontrada para visitar todas as cidades e retornar à origem, bem como o custo total para percorrer tal caminho.

2.3. Problema de Designação Generalizada

Entrada: Arquivos em formato txt, conforme ilustrado na Figura 2, 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.

Figura 2. Estrutura do arquivo de entrada para o problema da designação generalizada

Na Tabela 3 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.

Tabela 3. Custos dos módulos ao serem desenvolvidos por cada programador

		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

Na Tabela 4 são apresentadas quantas horas cada programador gasta para trabalhar em cada um dos oito módulos a serem desenvolvidos.

Tabela 4. Carga horária necessária para desenvolver cada módulo

		Módulos a serem desenvolvidos							
		1	2	3	4	5	6	7	8
Programadores	1	10	14	16	12	8	20	10	16
	2	13	12	15	13	9	18	9	18
	3	8	8	13	8	8	17	16	14
	4	15	17	18	15	12	15	16	18

Na Tabela 5 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.

Tabela 5. Número de horas que cada programador tem disponível

	Programadores			
	1	2	3	4
CH Disponível	30	25	20	40

Download: Os arquivos de entrada para testes da implementação podem ser acessados no endereço: download.

Saída: A designação de quais módulos foram atribuídos para quais programadores. Além disso, deve-se apresentar o custo total para a implementação dos módulos conforme a designação obtida.

2.4. Problema do Empacotamento Unidimensional

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

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

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

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 3):

```
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 3. Pseudocódigo do algoritmo guloso para encontrar uma primeira solução válida para o problema do empacotamento

Download: Os arquivos de entrada para os testes podem ser acessados no link: download.

Saída: Deve retornar a distribuição dos itens nos recipientes bem como a quantidade mínima de recipientes usada em tal distribuição.

2.5. Problema da Conexão de Circuitos

Entrada: Arquivos em formato txt (conforme detalhado na Tabela 7) 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.

Além da quantidade máxima de conexões presentes no circuito há ainda a limitação de mínimo e máximo de cada ponto (componente). Cada um deles deve conter pelo menos uma conexão. Por outro lado, na terceira linha da Tabela 7, 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.

Tabela 7. Estrutura do arquivo de entrada para o problema de otimização de circuitos

Número de pontos a serem conectados	5				
Número total de conexões no circuito	8				
Número máximo de conexões que um componente pode realizar	4				
Posição (em X) de todos os componentes	12,1	14,7	9	4,3	1,2
Posição (em Y) de todos os componentes	15	10	2,4	2,1	0,8

A função de vizinhança deve remover uma conexão aleatória e trocar por outra nova. Por exemplo, considere que há uma conexão entre os componentes 1 e 5 e que

esta foi selecionada para ser substituída. Em seguida serão então sorteados dois novos componentes para gerar uma nova conexão.

Nesse ponto dois fatores devem ser considerados:

- Para gerar a nova conexão pelo menos um dos novos componentes deve ser diferente dos dois anteriores;
- Se um dos componentes tem apenas uma conexão, ela não pode ser removida. Caso contrário seu número de conexões será zero. Caso isso ocorra outra conexão deve ser sorteada.

Download: Os arquivos de entrada para os testes podem ser acessados no endereço: download.

Saída: 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. A saída deve apresentar o conjunto de conexões efetuadas e a soma total das distâncias das conexões.

2.6. Problema das n rainhas

Entrada: arquivo em formato txt contendo o número de rainhas presentes no problema especificando a posição inicial de cada rainha.

A estrutura do arquivo de entrada está representada na Tabela 8. Neste cenário o problema é composto por quatro rainhas. A segunda linha do arquivo indica as posições ocupadas pela rainhas da seguinte forma: a primeira rainha está posicionada na linha 2, a segunda e terceira rainhas estão posicionadas na linha 4 enquanto a quarta rainha está posicionada na linha 2, como pode ser visualizado na Figura 4.

Tabela 8. Estrutura do arquivo de entrada para o problema das n rainhas

Número de rainhas (n)	4			
Tamanho de cada um dos itens	2	4	4	2

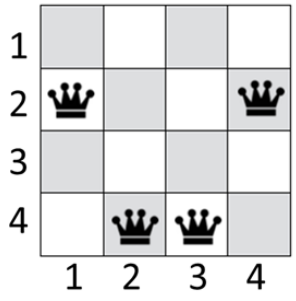

Figura 4. Cenário do arquivo de entrada representado na Tabela 8

Saída: A solução final indicando a posição de cada uma das n rainhas, de forma que nenhuma ataque as demais.

Download: os arquivos de entrada podem ser obtidos através do link: download.

3. Formação das Equipes e Definição dos objetivos

Equipe 1	Igor Correa Domingues de Almeida Guilherme Altmeyer Soares
Aplicação: Problema do Caixeiro Viajante usando vizinhança do tipo substituição Técnica: Busca Tabu	
Equipe 2	Fábio Mendes Crepaldi Hermes Soares de Jesus
Aplicação: Problema do Caixeiro Viajante Técnica: Otimização por Colônia de Formigas	
Equipe 3	Rafael dos Santos André Gustavo Franco
Aplicação: Problema de Roteamento Técnica: Otimização por Colônia de Formigas	
Equipe 4	Arthur Angelo Cenci Silva Lucas Ivanov Costa
Aplicação: Problema da Designação Generalizada Técnica: Busca Tabu	
Equipe 5	Weberson Morelli Leite Junior João Gabriel Fazio Pauli
Aplicação: Problema do Caixeiro Viajante usando vizinhança do tipo substituição Técnica: Simulated Annealing	
Equipe 6	Paulo Roberto Bomfim Rosa Bruno Felipe Lovato
Aplicação: Problema do Caixeiro Viajante usando vizinhança do tipo substituição Técnica: Simulated Annealing	
Equipe 7	Ellen Cristini Schreiber Brzozoski Maria Eduarda Amorim Quevedo
Aplicação: Problema do Caixeiro Viajante usando vizinhança do tipo inserção Técnica: Busca Tabu	
Equipe 8	Matheus Barros Camille de Fatima Gonçalves Simão
Aplicação: Problema das n rainhas Técnica: Busca Tabu	
Equipe 9	João Vitor da Silva João Leonardo Livero Lavaqui
Aplicação: Problema do Caixeiro Viajante usando vizinhança do tipo inserção Técnica: Simulated Annealing	

Equipe 10	Kauan Carlos Campos Matheus Felipe Pereira Lopes
Aplicação: Problema do Empacotamento Unidimensional Solução Inicial Baseada na Estratégia Gulosa Técnica: Simulated Annealing	
Equipe 11	Eduardo Dias Machado Lucas Gabriel Hadada
Aplicação: Problema da Designação Generalizada Técnica: Busca Tabu	
Equipe 12	Jonathan Santos Tadei Gabriel Merline Cavalcante Brasil
Aplicação: Problema da Mochila Binária Técnica: Busca Tabu	
Equipe 13	Jean Lucas Ribeiro de Oliveira José Raul Ramírez Pinzón
Aplicação: Problema da Designação Generalizada Técnica: Simulated Annealing	
Equipe 14	Matheus Henrique Gaspar Samuel Vitrio Ferreira
Aplicação: Problema da Designação Generalizada Técnica: Busca Tabu	
Equipe 15	Monique Barros Ryan Hideki Inoue Matsunaga Pereira
Aplicação: Problema das n rainhas Técnica: Busca Tabu	
Equipe 16	Alan Felipe Muller Pedro Lucas Moraes
Aplicação: Problema do Empacotamento Unidimensional Solução Inicial baseada no pior casa (um item por recipiente) Técnica: Simulated Annealing	
Equipe 17	Carlos Eduardo Pagani Grosso Matheus Henrique Gallert
Aplicação: Problema de Roteamento Técnica: Otimização por Colônia de Formigas	
Equipe 18	Thalita Wiederkehr Pereira Vinícius Mattos Marcos
Aplicação: Problema da Mochila Binária Técnica: Simulated Annealing	

Equipe 19	Felipe Kravec Zanatta Roberto da Silva Ferreira
Aplicação: Problema da Mochila Binária Técnica: Simulated Annealing	
Equipe 20	Caio Hideki Gomes Shimohiro Matheus Bueno Pereira
Aplicação: Otimização de Circuitos Técnica: Simulated Annealing	
Equipe 21	Fabio Kenji Sato Pedro Henrique de Oliveira Berti
Aplicação: Otimização de Circuitos Técnica: Simulates Annealing	
Equipe 22	Silvestre Chiari Netto Victor Negri Bergamo Lopes da Silva
Aplicação: Problema do Caixeiro Viajante Técnica: Otimização por Colônia de Formigas	