UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

Análise de Algoritmos

Prof. Dr. Herbert Oliveira Rocha
herberthb12@gmail.com

O que será abordado neste curso?

Problemas Computacionais

seus Algoritmos

suas Complexidades

Descrição da Disciplina

- Disciplina: Análise de Algoritmos
- Carga horária: 60h

- Término: 21/06/2023
- Horário: **10h às 12h**
- Professor: Herbert Oliveira Rocha, Dr.

    e-mail: herbert.rocha@ufrr.br

    github: https://github.com/hbgit

    home page: https://hbgit.github.io/

Site da Disciplina: No SIGAA

Dados do professor

https://prismrr.github.io/

Plano de Ensino

SIGAA

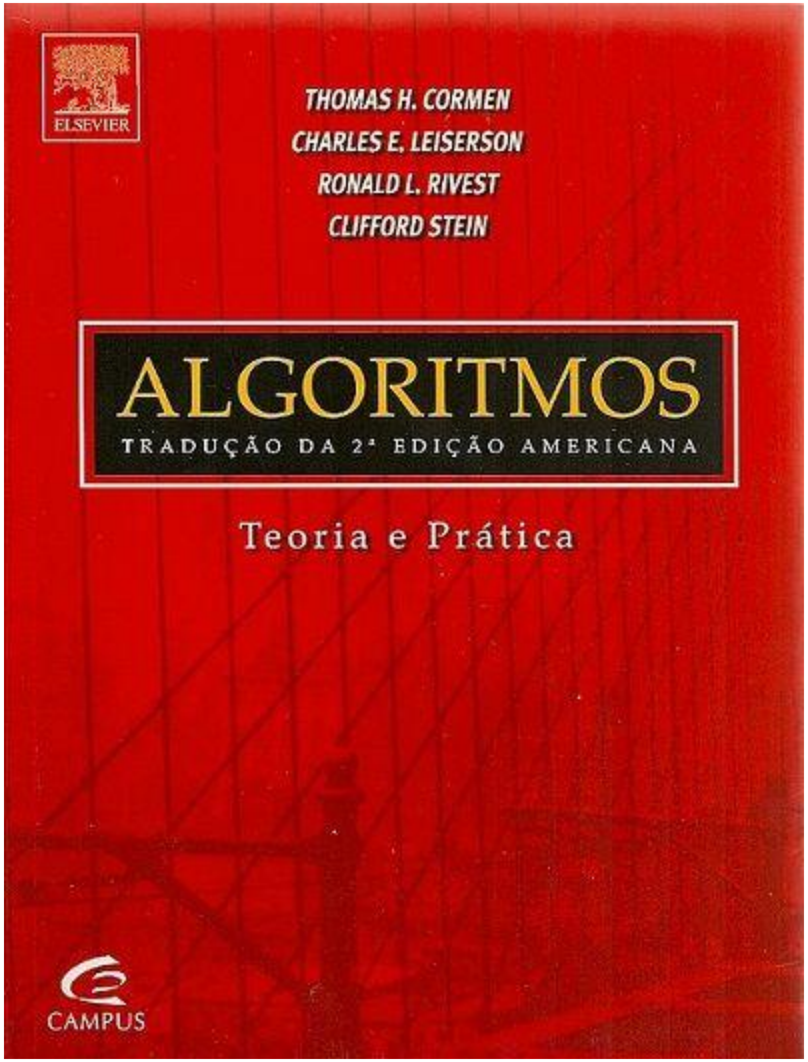
CODESIGNAL

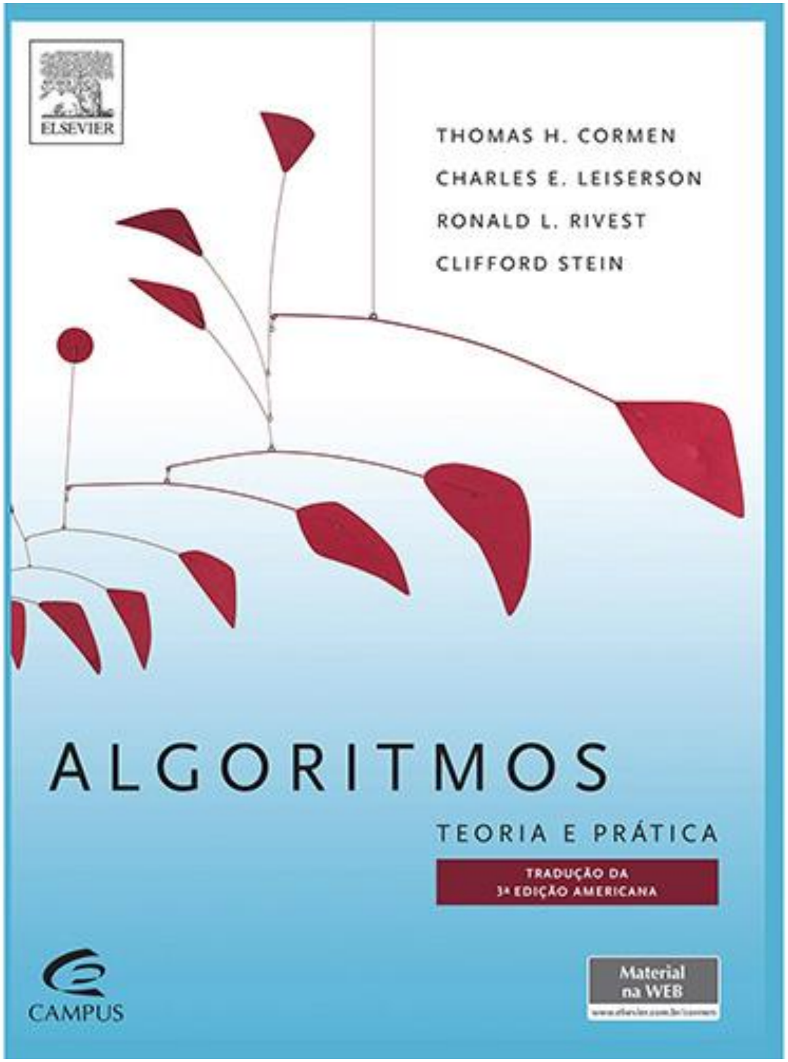
beecrowd

GitHub

plotly

Referências



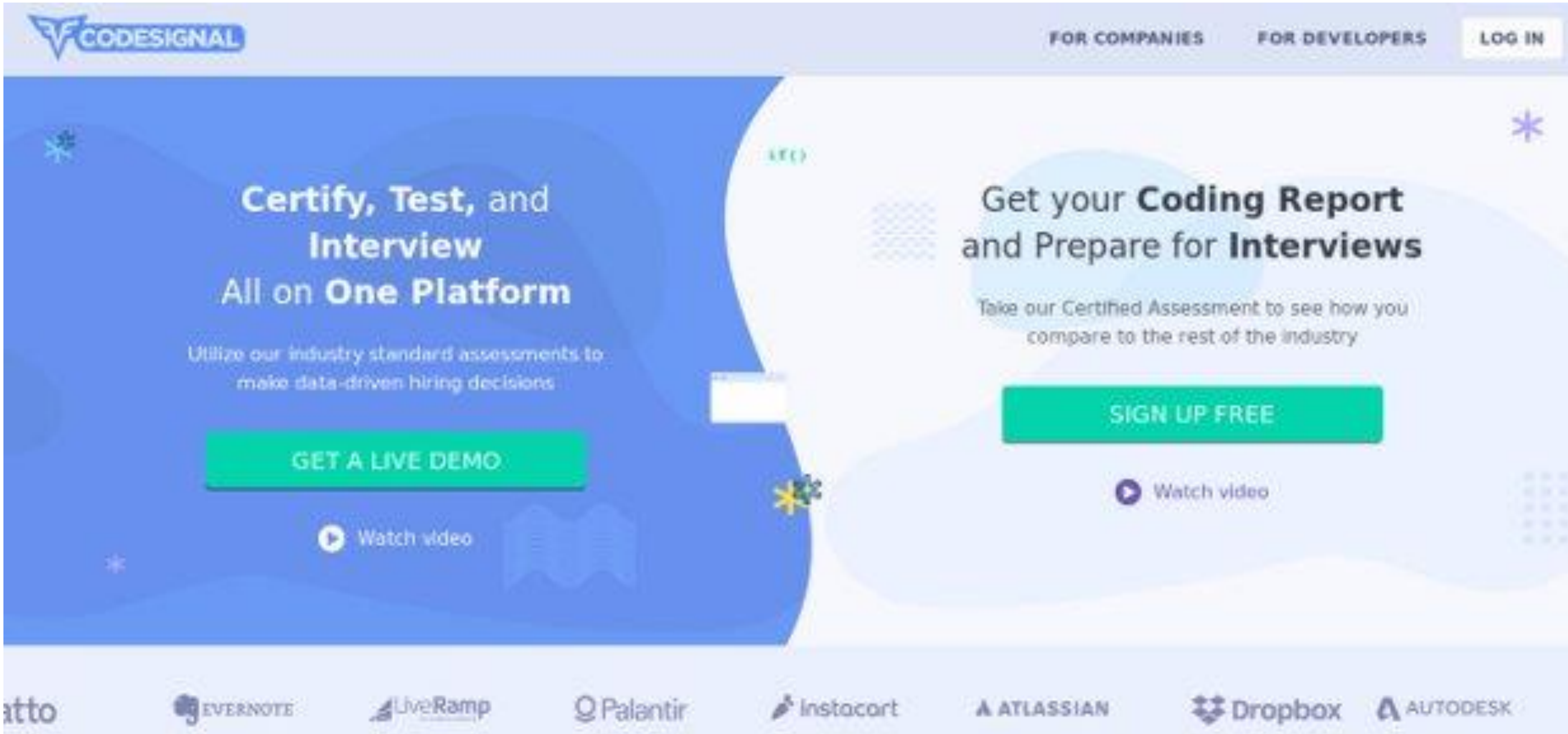


Projeto Final

https://github.com

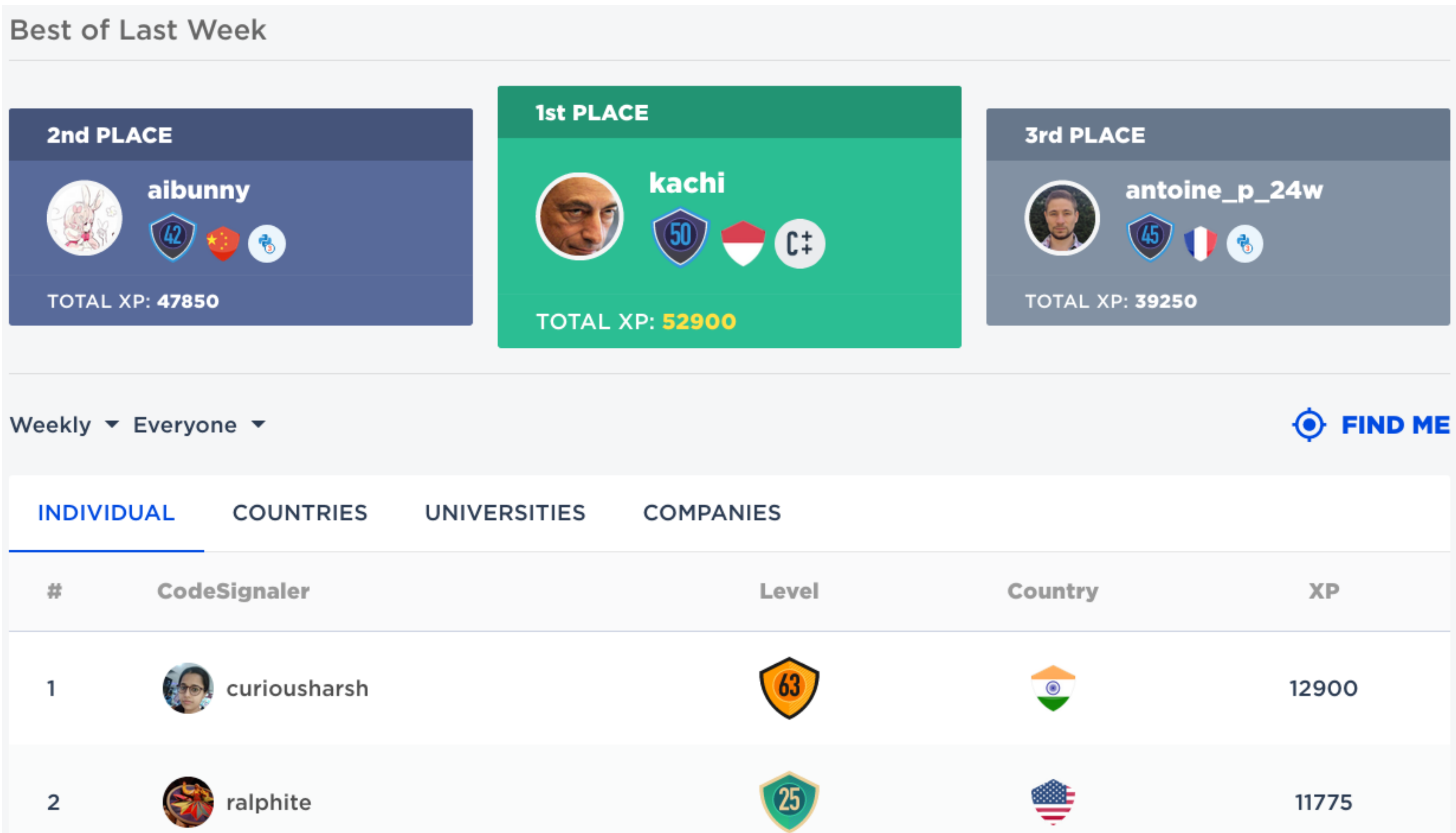
Pontos Extras

https://codesignal.com/developers/



Pontos Extras

https://app.codesignal.com/leaderboard/



E os Pontos Extras?

https://codesignal.com/developers/

Regras:

1. **Resolver o maior número no modo Arcade - Graphs**

1. **Resolver o maior número de desafios no Company Challenges**

1. **Somente os 5 primeiros com maior pontuação receberão pontos extras.**

1. **O histórico das batalhas deverá ser comprovado**

E os Pontos Extras?

https://codesignal.com/

Pontuação:

1º Lugar = 5 pontos

2ª Lugar = 4 pontos

3ª Lugar = 3 pontos

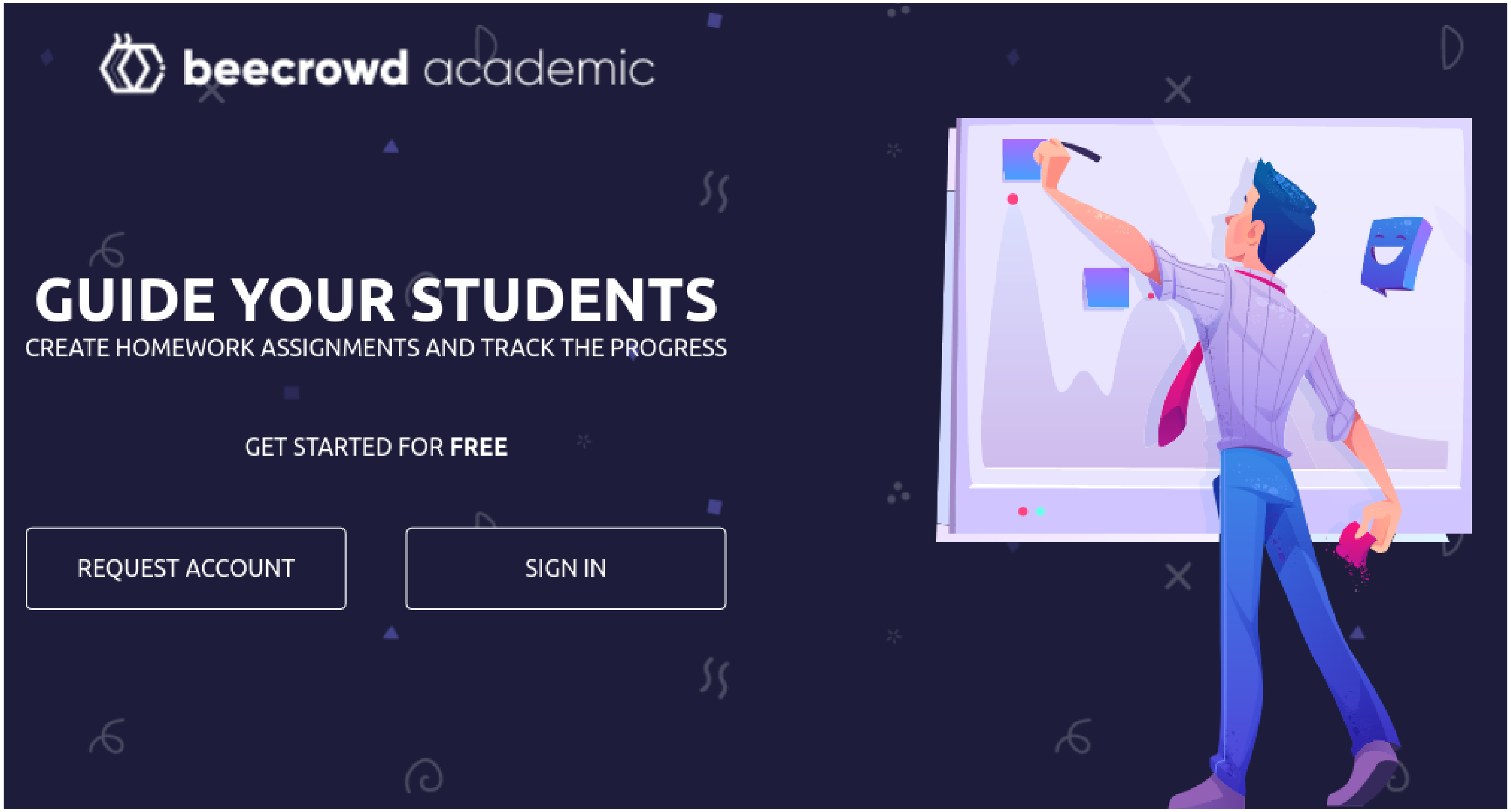
4ª Lugar = 2 pontos

5ª Lugar = 1 pontos

E os Pontos Extras?

https://www.beecrowd.com.br/academic/

5 pts – lab com 12 questões



UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

Problemas

Problemas

O problema da ida ao supermercado

Onde estacionar o carro ?

Como fluir entre as prateleiras ?

Problema 1

- Eu conheço algum problema similar? Se positivo, qual foi a solução dada?
    - **Torre de Hanoi**
    - **O dilema do Imperador**
    - **Jogo de Xadrez**
- Que estratégia posso usar para resolvê-lo?
    - Todas as possibilidades - MÉTODO EXAUSTIVO

Problema 2

- Que dados nós temos para começar a resolução do problema?
- Quais são as restrições envolvidas?
- O que é uma solução para o problema?
- Eu conheço algum problema similar? Se positivo, qual foi a solução dada?
- Que estratégia posso usar para resolvê-lo?
- *O que é o problema ?*

Problema 2

- ✓ Que estratégia posso usar para resolvê-lo?
  - ✓ Todas as possibilidades - MÉTODO EXAUSTIVO.
  - ✓ Tentar sempre a melhor alternativa - MÉTODO GULOSO.
  - ✓ Começar por qualquer lugar - MÉTODO HEURÍSTICO.
  - ✓ *Determinar* a melhor rota - ISTO É MATEMATICAMENTE COMPROVADO, MAS COMPUTACIONALMENTE DIFÍCIL DE SER OBTIDO.

Problemas

O PROBLEMA DO CAIXEIRO VIAJANTE e o PROBLEMA DA TORRE DE HANOI são considerados COMPUTACIONALMENTE DIFÍCEIS de se resolver.

Problemas

- **Um problema de manufatura**
  – Em que ordem produzir?
  – Como montar?
  – Comprar as peças de qual fornecedor?
- **Um problema de distribuição de mercadorias**
  – Qual caminhão?
  – Por onde chegar ao cliente?
  – Que clientes atender primeiro?
- **Um problema de alocação de antenas de TV, celular**
  – Quantas são necessárias para um melhor recobrimento da região?
  – Que locais?
  – Qual distância entre elas?

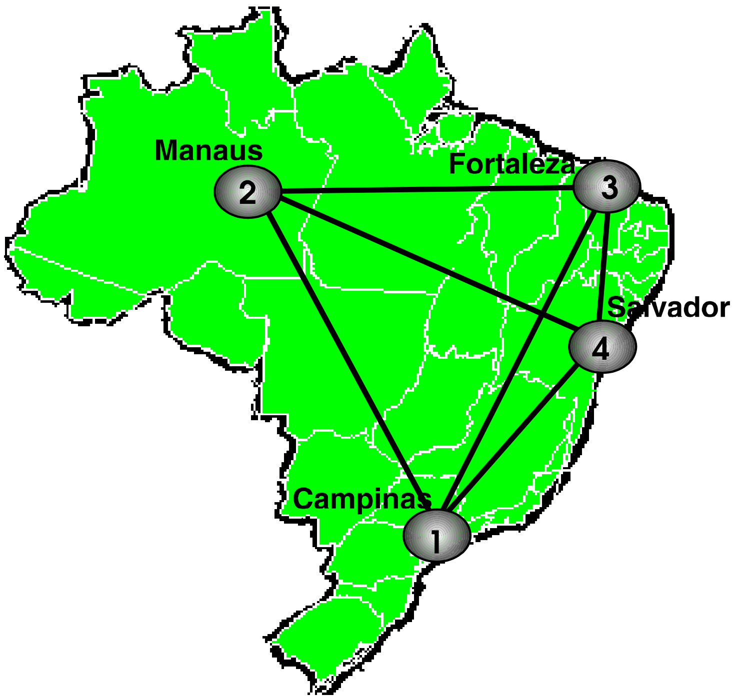
O Problema do Caixeiro Viajante

Traveling Salesman Problem ou **TSP**

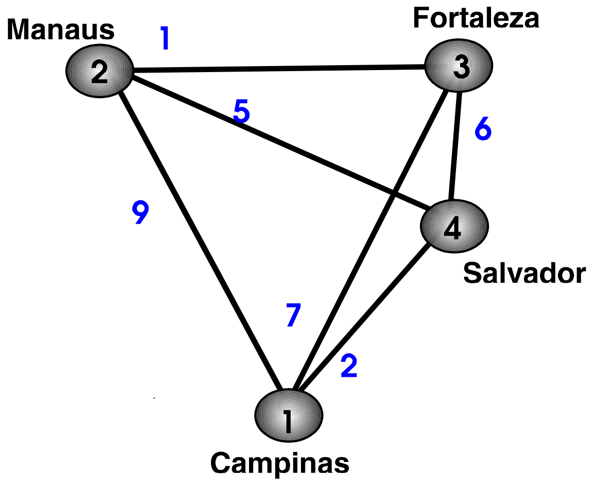
Um caixeiro viajante quer visitar todo um determinado conjunto de cidades, cada cidade exatamente uma única vez, terminando a visita na mesma cidade que começar. Ainda, o custo envolvido no percurso deve ser o menor possível.

- Problema clássico da Teoria da Complexidade Computacional (NP-difícil);
- Um dos mais estudados e utilizado para modelar problemas práticos e até outros modelos teóricos;
- A afirmativa "*tão difícil quanto o TSP*" é uma máxima;
- Versão simétrica: $(n\text{-}1)!/2$ trajetórias válidas, para n cidades;
- Exemplo prático: Problemas de Escalonamento.

Exemplo de um TSP com 4-cidades

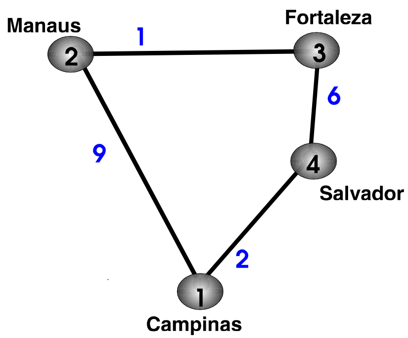


min



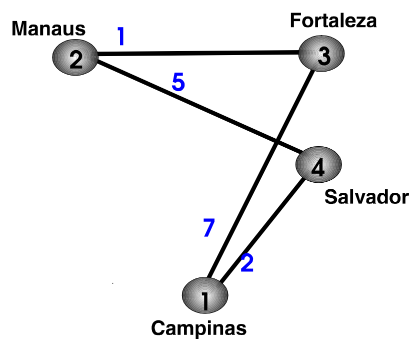
Exemplo de um TSP com 4-cidades

As três trajetórias possíveis ((4-1)!/2) são:



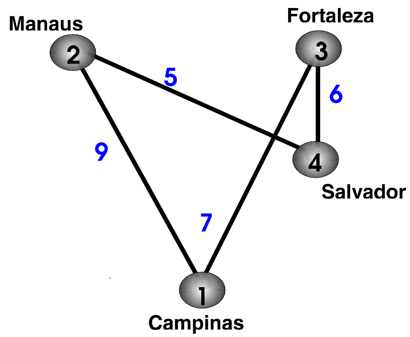
A

Comprimento: 18



B

Comprimento 15

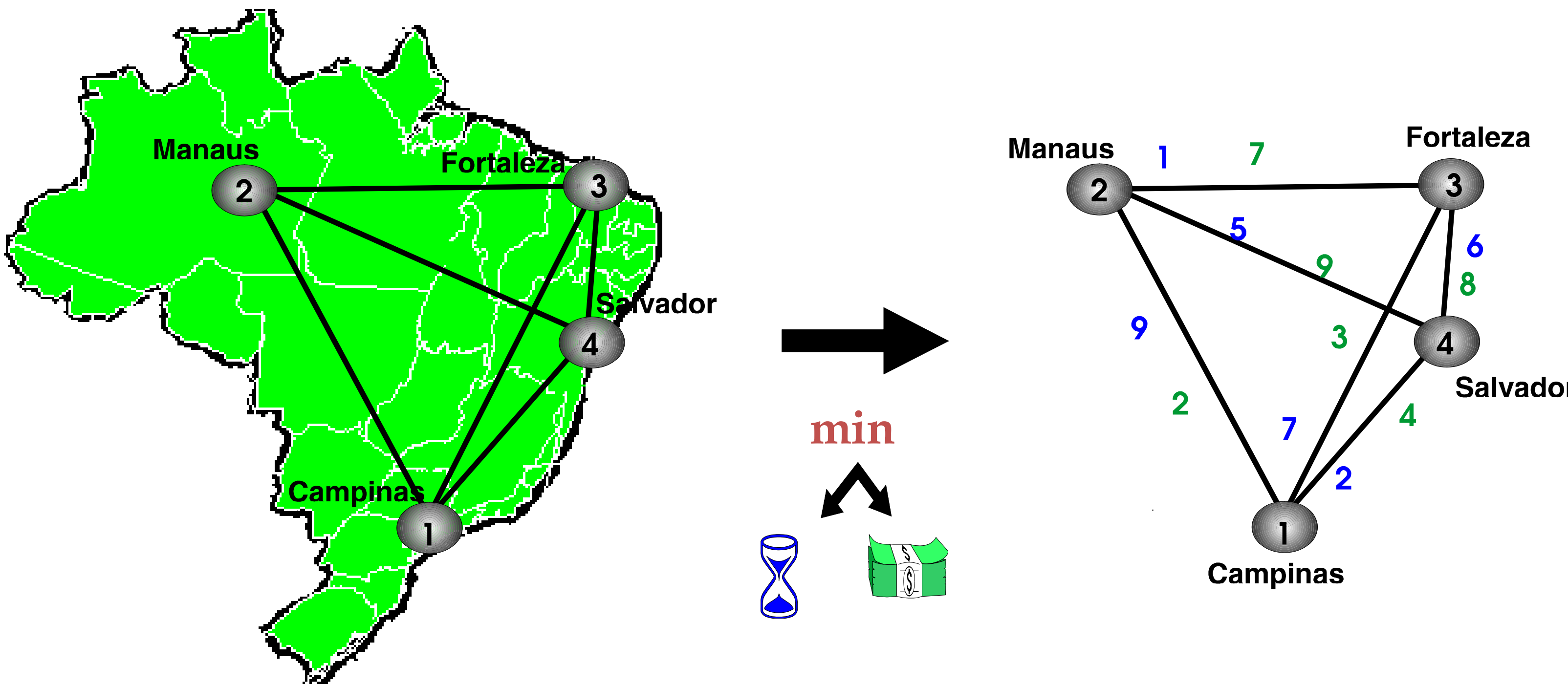


C

Comprimento: 27

A solução ótima é a trajetória B

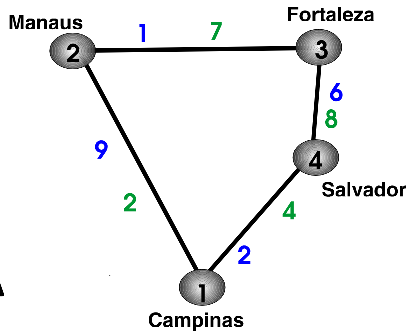
Exemplo de um TSP com 4-cidades e 2 objetivos



Exemplo de um TSP com 4-cidades e 2 objetivos

As três trajetórias possíveis ((4-1)!/2):

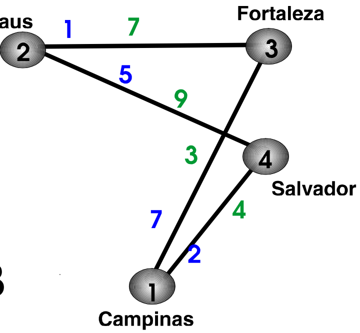
A



Comprimento (matriz 1): 18

Comprimento (matriz 2): 21

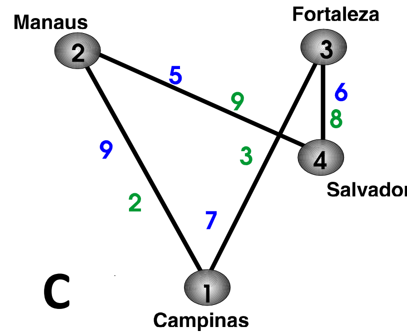
B



Comprimento (matriz 1): 15

Comprimento (matriz 2): 23

C



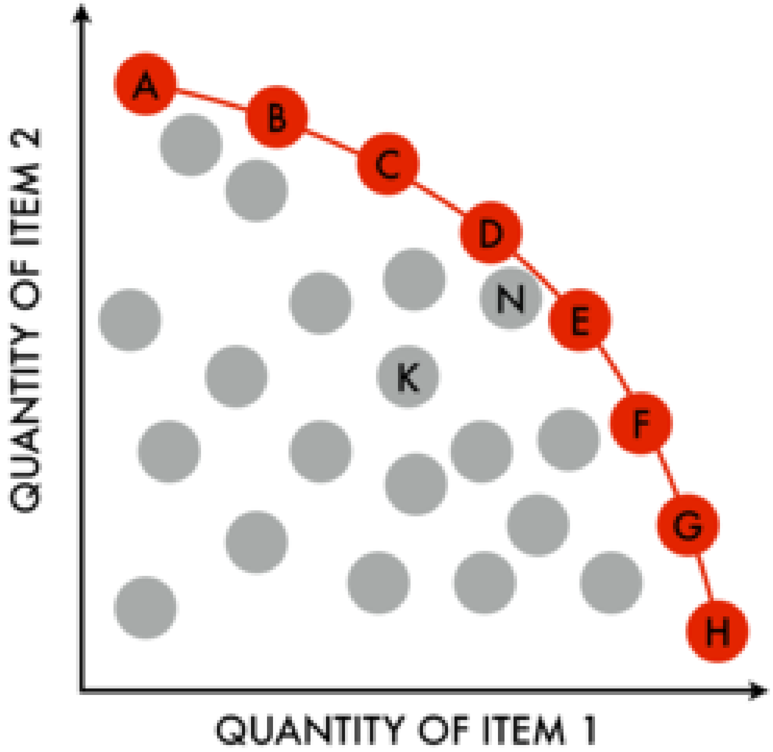
Comprimento (matriz 1): 27

Comprimento (matriz 2): 22

Pareto Ótimo:

Eficiência ou **óptimo de Pareto** é um conceito desenvolvido pelo italiano Vilfredo Pareto, que define um estado de alocação de recursos em que é impossível realocá-los tal que a situação de qualquer participante seja melhorada sem piorar a situação individual de outro participante

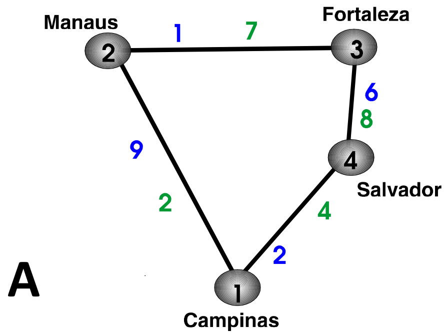
Exemplo de um TSP com 4-cidades e 2 objetivos

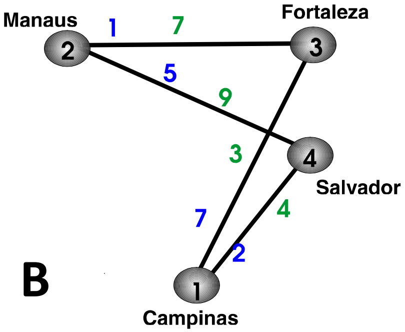


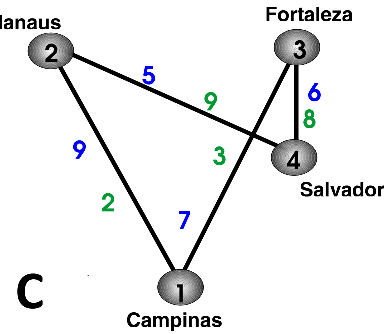
Os pontos vermelhos indicam a escolha ótima em termos de eficiência dado o conjunto de opções, formando uma *fronteira de Pareto* sob o espaço de escolhas restante (abaixo e à esquerda, em cinza)

Exemplo de um TSP com 4-cidades e 2 objetivos

As três trajetórias possíveis ((4-1)!/2):







Comprimento (matriz 1): 18

Comprimento (matriz 2): 21

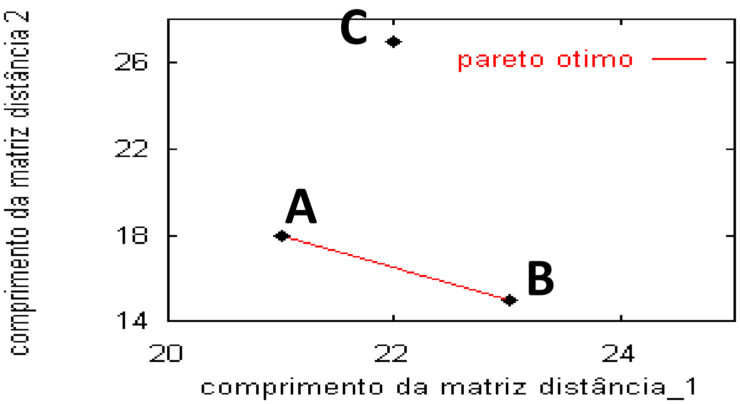
Comprimento (matriz 1): 15

Comprimento (matriz 2): 23

Comprimento (matriz 1): 27

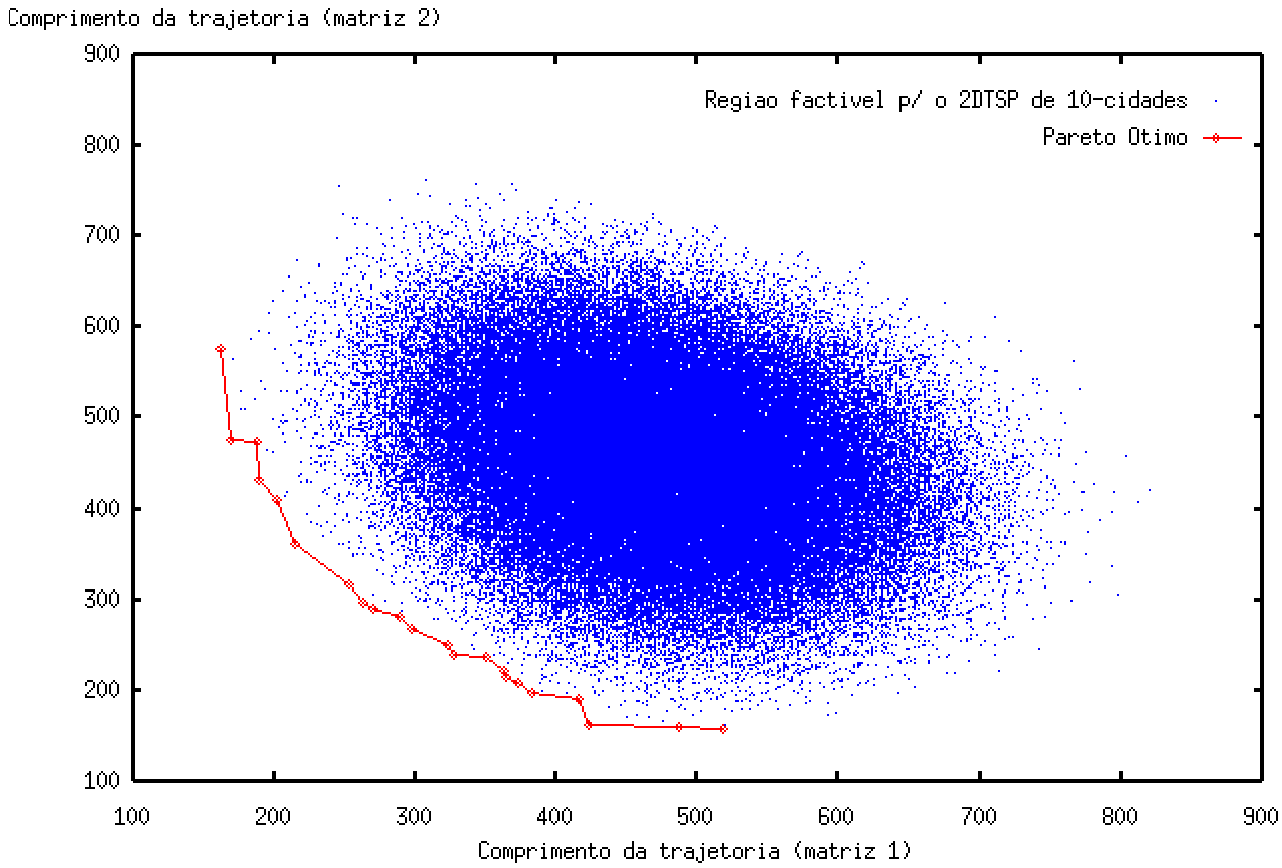
Comprimento (matriz 2): 22

Pareto Ótimo:



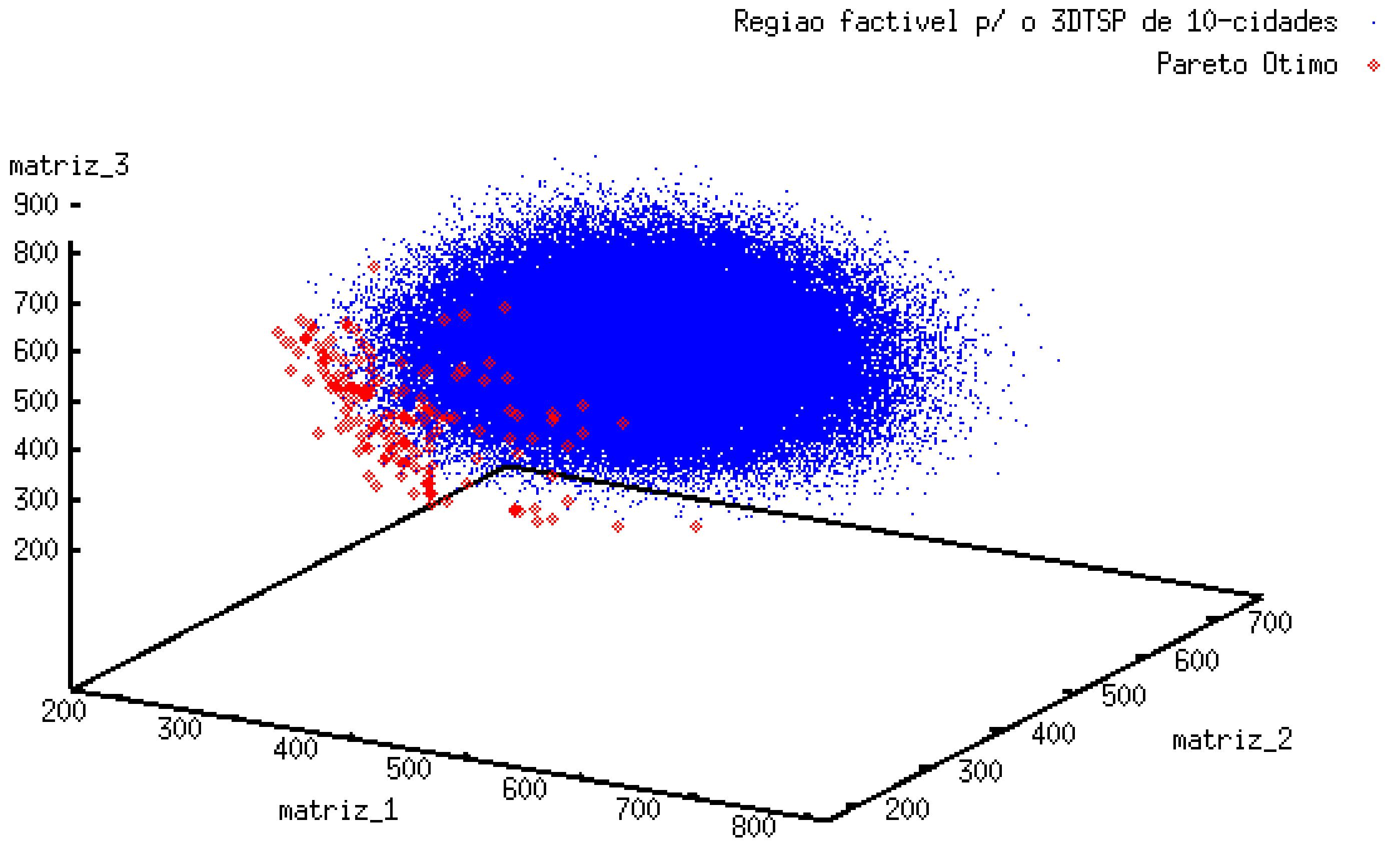
Exemplo de um TSP com 10-cidades e 2 objetivos

- 181.440 trajetórias válidas;
- 22 soluções no Pareto Ótimo (2 matrizes de distância).

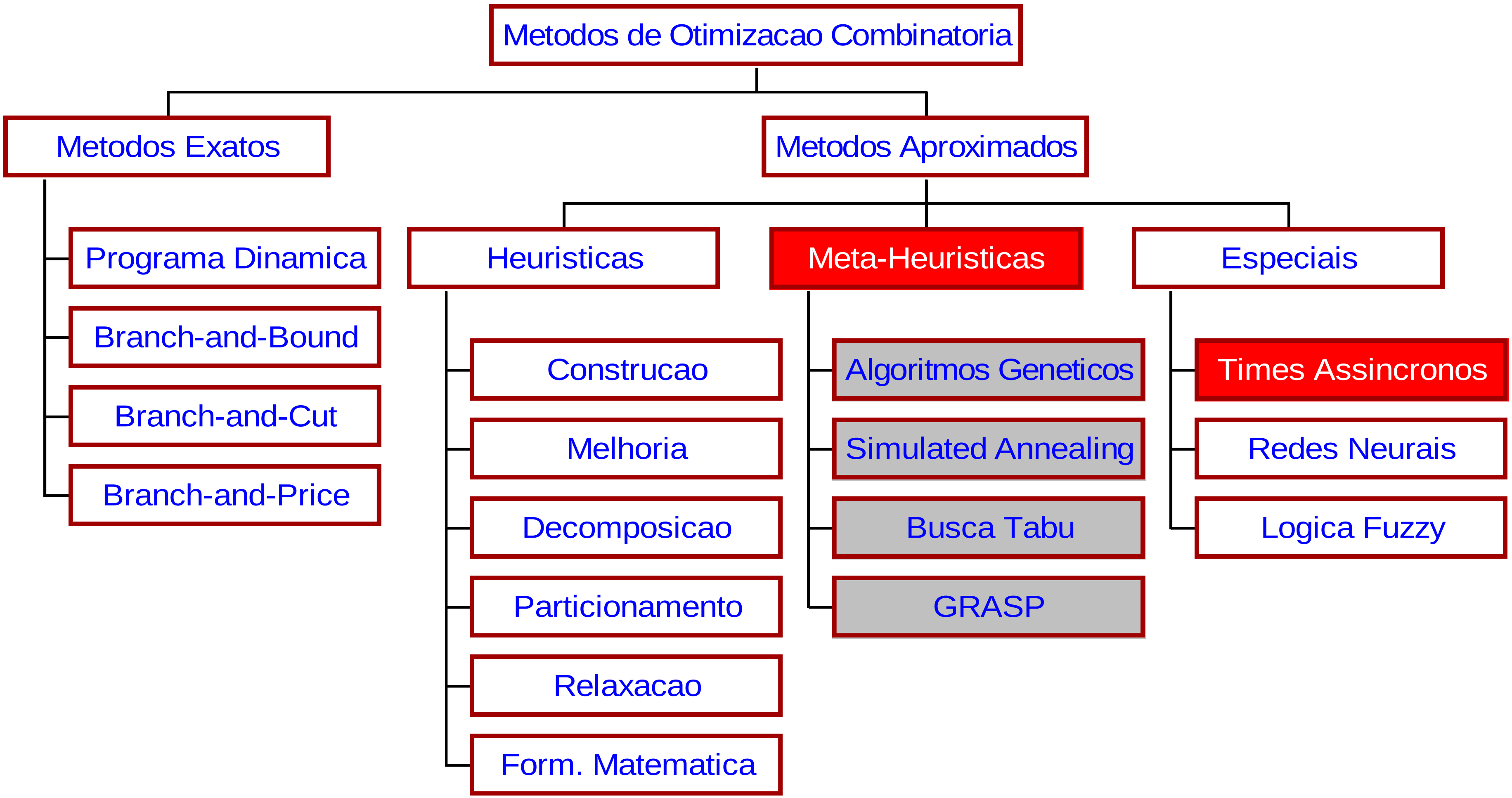


Exemplo de um TSP com 10-cidades e 3 objetivos

- 181.440 trajetórias válidas;
- 162 soluções no Pareto Ótimo (3 matrizes de distância).



Abordagens de Resolução de Problemas



Análise de Problemas e Algoritmos

Analisar problemas e algoritmos do ponto de vista computacional, significa determinar como os mesmos se comportam para pequenas e, principalmente, grandes instâncias e para as casos mais comuns, os melhores casos e, principalmente, os casos mais difíceis de ocorrerem.

Medidas de Análise

- **Tempo de Execução.**
- **Espaço Ocupado de Memória.**

subsequence sum algorithms

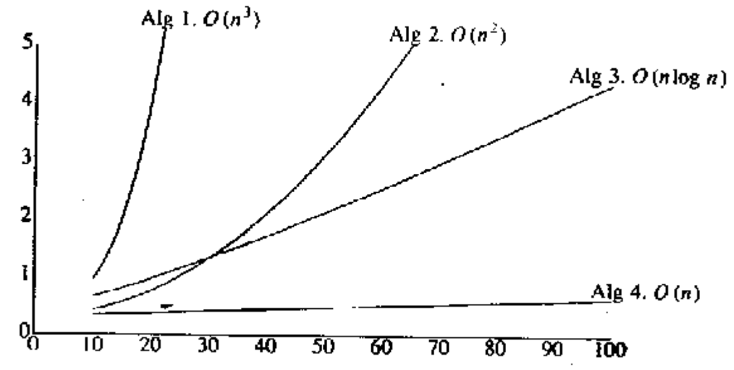
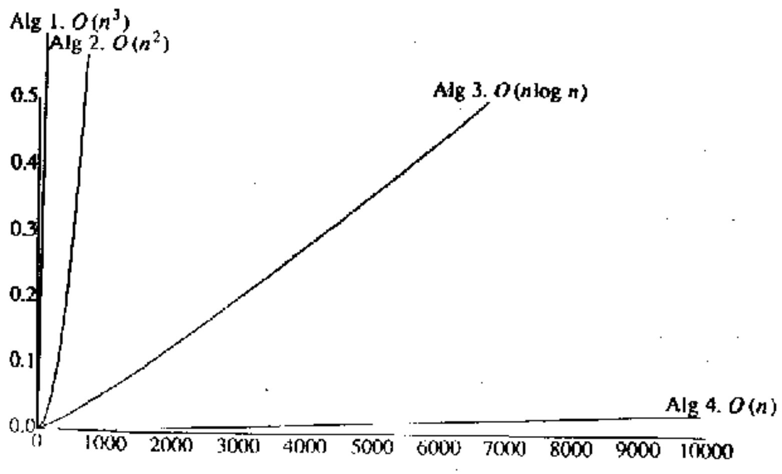


Figure 2.4 Plot (n vs. seconds) of various maximum subsequence sum algorithms



CUSTO DE UTILIZAÇÃO DE UM ALGORITMO

0 custo de um algoritmo pode ser medido de várias formas:

- Através da **execução do programa** em um computador real (tempo de execução medido diretamente).

Através do uso de um **modelo matemático**.

(computador MIX proposto por Knuth D.E.,1968).

Análise de Problemas e Algoritmos

Na Prática:

- Ignorar o custo de algumas das operações envolvidas.

  Ex.: ignorar operações aritméticas, atribuição e manipulações de índices.

- Considerar apenas as operações mais significativas.

  Ex.: na ordenação considerar somente o número de comparações entre os elementos.

Medidas de Tempo Obtidas Através da Execução do Programa em um Computador Real

PROBLEMAS:

- Resultados dependentes do **compilador** que pode favorecer, algumas construções em, detrimento de outras;
- Resultados dependem do **hardware**;
- Quando **grandes quantidades de memória** são utilizadas, as medidas de tempo podem depender deste aspecto.

COMO MEDIR 0 CUSTO DE EXECUÇÃO DE UM ALGORITMO

Função de Custo ou Função de Complexidade onde:

f(n) = medida de custo necessário para executar um algoritmo para um problema de tamanho n,

Se f(n) é uma medida, da quantidade de tempo necessário para executar um algoritmo em um problema de tamanho n, então *f* é chamada função de complexidade de tempo.

Se *f(n)* é uma medida da quantidade de memória necessária para executar um algoritmo de tamanho n, então *f* é chamada função de complexidade de espaço.

COMO MEDIR O CUSTO DE EXECUÇÃO DE UM ALGORITMO

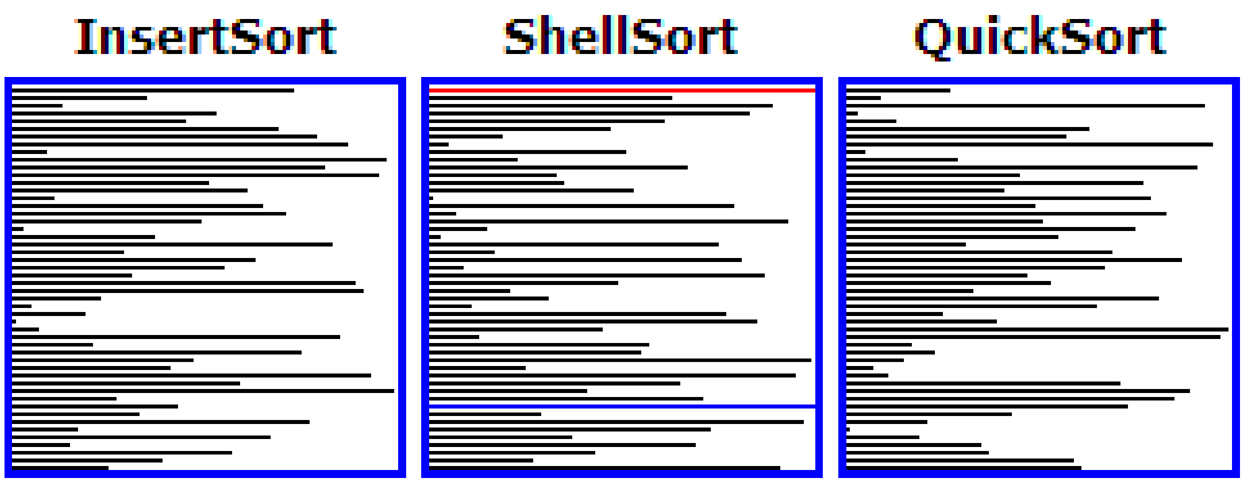
IMPORTANTE.. tempo não é tempo REAL!

É importante ressaltar que a complexidade de tempo na realidade não representa tempo diretamente, mas o número de vezes que determinada operação considerada importante é executada.

ANÁLISE DE ALGORITMOS:

1. Análise de um algoritmo particular

2. Análise de uma classe de algoritmos.



1. **Análise de um algoritmo particular**

Qual é o custo de usar um dado algoritmo para resolver um problema específico?

- Análise do número de vezes que cada parte do algoritmo deve ser executada;
- Estudo da quantidade de memória necessária.

EXEMPLO 1: Máximo de um conjunto vet[1: n], n > 1

```
(int)  função Max (var vet: Vetor);
var i, Temp:  int;
INICIO
      Temp =  vet[1];
      PARA i  := 2 TO n
      SE Temp < vet[i] ENTÃO
         Temp = vet[i];
         Max = Temp;
       FIM-SE
FIM
```

EXEMPLO 1: Máximo de um conjunto vet[1: n], n > 1

Seja f a **função de complexidade tal que *f(n)* é o número de comparações entre os elementos de vet se vet tiver *n* elementos. Neste caso**

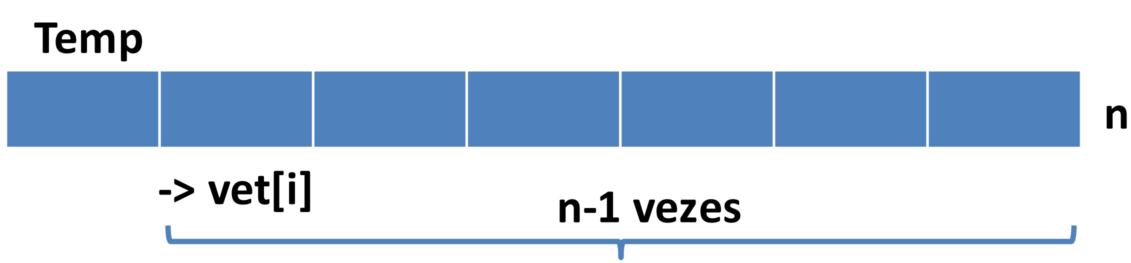
$f(1) = 0$

$f(n) = n - 1$, para $n > 1$

Análise de Problemas e Algoritmos

EXEMPLO 1: Máximo de um conjunto vet[1: n], n > 1

```
(int)  função Max (var vet: Vetor);
var i, Temp:  int;
INICIO
      Temp =  vet[1];
      PARA i := 2 TO n
      SE Temp < vet[i] ENTÃO
         Temp = vet[i];
         Max = Temp;
       FIM-SE
FIM
```



Problema de Encontrar o Maior Elemento de um Conjunto

MELHOR CASO = CASO MÉDIO = PIOR CASO

n-1 comparações

Complexidade - O(n)

EXEMPLO 2: Pesquisa Sequencial (ou Linear)

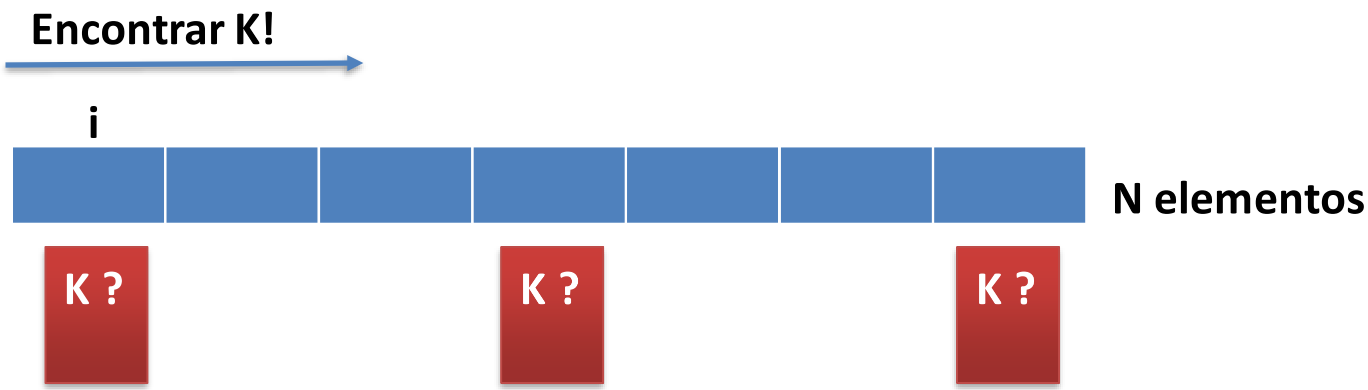
Problema

- **Cada registro contém uma chave única que é utilizada para recuperar registros do arquivo.**
- **Dada uma chave qualquer o problema consiste em localizar o registro que contenha esta chave.**

EXEMPLO: Pesquisa Sequencial

Solução

- **O algoritmo de pesquisa mais simples que existe é o que faz uma pesquisa sequencial.**



MELHOR CASO

Ocorre quando o registro procurado é o primeiro consultado.

Apenas 1 comparação (custo 1)

PIOR CASO

Ocorre quando o registro procurado é o último a ser consultado, ou então não está presente no arquivo.

n comparações ou n+1 comparações

ANÁLISE DO CASO MÉDIO

Considere que toda pesquisa recupera um registro.

Não existe pesquisa sem sucesso.

ANÁLISE DO CASO MÉDIO

Se *pi* for a probabilidade de que o *i*-ésimo registro seja procurado, e considerando que para recuperar o *i*-ésimo registro sejam necessárias *i* comparações, então:

$$f(n) = 1p1, + 2\ p2 + 3p3 + \ldots + npn$$

Para calcular *f(n)* basta conhecer a distribuição de probabilidades *pi*.

Se cada registro tiver a mesma probabilidade de ser acessado que todos os outros, então $pi = 1/n$, $1 \leq i \leq n$. Neste caso:

$$f(n) = \frac{1}{n}(1 + 2 + 3 + \cdots + n) \ = \ ?$$

ANÁLISE DO CASO MÉDIO

Se pi for a probabilidade de que o i-ésimo registro seja procurado, e considerando que para recuperar o i-ésimo registro sejam necessárias i comparações, então:

$$f(n) = 1p1, + 2\ p2 + 3p3 + \ldots + npn$$

Para calcular $f(n)$ basta conhecer a distribuição de probabilidades pi.

Se cada registro tiver a mesma probabilidade de ser acessado que todos os outros, então $pi = 1/n$, $1 \leq i \leq n$. Neste caso:

$$f(n) = \frac{1}{n}(1 + 2 + 3 + \cdots + n) = \frac{1}{n}\left(\frac{n(n+1)}{2}\right) = \frac{n+1}{2}$$

MELHOR CASO, PIOR CASO, CASO MÉDIO

- **Melhor**

  Corresponde ao menor tempo de execução sobre todas as possíveis entradas de tamanho n.

- **Pior Caso**

  Corresponde ao maior tempo de execução sobre todas as entradas de tamanho n.

  Se f é uma função de complexidade baseada na análise de pior caso então o custo de aplicar o algoritmo nunca é maior do que $f(n)$.

MELHOR CASO, PIOR CASO, CASO MÉDIO

- **Caso Médio (ou caso esperado)**

Corresponde à média dos tempos de execução de todas as entradas de tamanho n.

Na análise do caso esperado, uma distribuição de probabilidades sobre o conjunto de entradas de tamanho n é suposta, e o custo médio é obtido com base nesta distribuição.

Por esta razão, a análise do caso médio é geralmente muito mais difícil de obter do que as análises do melhor e do pior caso.

EXEMPLO: Pesquisa sequencial

Seja f uma função de complexidade tal que *f(n)* é o número de vezes que a chave de consulta é comparada com a chave de cada registro.

Neste Caso:

melhor caso : $f(n) = 1$

pior caso : $f(n) = n$

caso médio : $f(n) = (n+1)/2$

COMPLEXIDADE: $O(n)$

Análise de Problemas e Algoritmos

- **ALGORITMO PARA ENCONTRAR O MAIOR E O MENOR ELEMENTO DE UM CONJUNTO**
- **ALGORITMO PARA ORDENAR UM CONJUNTO DE ELEMENTOS**

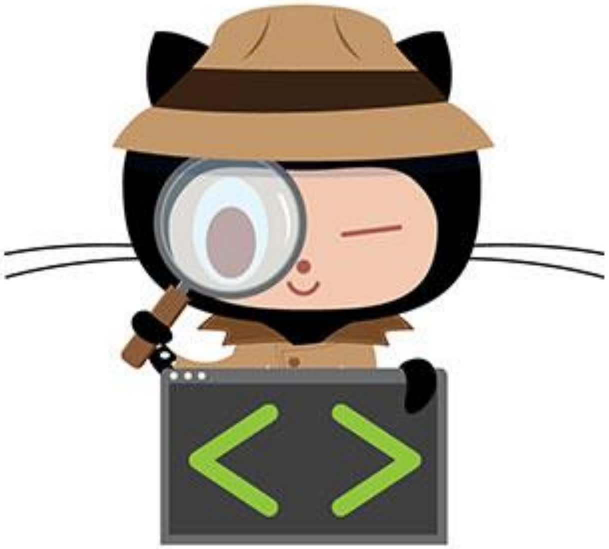
  Buble Sort

  Insertion Sort

  Selection Sort

(Analisem o melhor caso, pior caso e o caso médio)

See you



Perguntas?