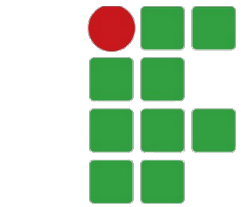
Metaheurísticas

Introdução

Felipe Augusto Lima Reis
felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

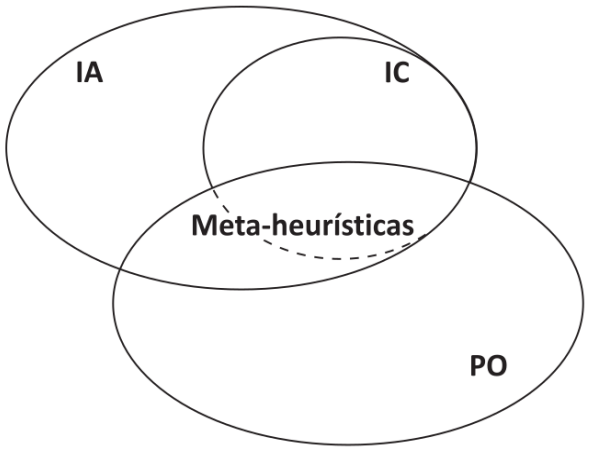
Orientações Iniciais

- Para entender os métodos que serão descritos nesta seção da disciplina, é necessário o entendimento, ao menos básico, dos seguintes conceitos;
  - Otimização e Pesquisa Operacional;
  - Complexidade de Algoritmos;
- Recomenda-se o conhecimento intermediário / avançado em programação.

Introdução

Contexto

- As heurísticas e metaheurísticas são um campo comum da Pesquisa Operacional, da IA e IC;
- Surgiram como alternativa aos métodos exatos para resolução de problemas de otimização de alta complexidade
  - Problemas que não podem ser solucionados em tempo polinomial (NP e NP-completos) [Belfiore and Fávero, 2013]



Fonte: Silva Neto e Becceneri (2009) apud [Belfiore and Fávero, 2013]

Conceitos - Otimização

- Otimização refere-se à escolha do melhor elemento em um conjunto de alternativas [Luzia and Rodrigues, 2009];
  - Um problema de otimização consiste em maximizar ou minimizar uma função linear de variáveis de decisão, sujeita a um conjunto de restrições [Belfiore and Fávero, 2013]
  - O resultado do problema de otimização é a produção de uma **solução ótima**
    - Corresponde ao melhor valor possível para um problema, segundo critérios estabelecidos previamente.

Conceitos - Pesquisa Operacional

- Segundo [Belfiore and Fávero, 2013], técnicas para solução de problemas de Pesquisa Operacional podem ser divididas em 3 grandes grupos:
  - Modelos Determinísticos;
    - Programação Linear, Não Linear, Dinâmica e Em Redes;
  - Modelos Estocásticos
    - Teoria das Filas e dos Jogos, Prog. Dinâmica Estocástica;
  - Outras técnicas (heurísticas e metaheurísticas)
    - Inteligência Artificial (IA), Inteligência Computacional;
- Segundo [Luke, 2013], heurísticas e metaheurísticas correspondem a um tipo de otimização estocástica.

Conceitos - Heurísticas

"Heurística pode ser definida como um procedimento de busca guiada pela intuição, por regras e ideias, visando encontrar uma boa solução." [Belfiore and Fávero, 2013]

Conceitos - Metaheurísticas

"Metaheurística é a combinação de procedimentos de busca com estratégias de mais alto nível, incluindo intensificação e diversificação, buscando escapar de ótimos locais e encontrar soluções muito próximas do ótimo global, porém sem garantia da otimalidade." [Belfiore and Fávero, 2013]

Metaheurísticas

Metaheurísticas

- Segundo [Luke, 2013] e [Luzia and Rodrigues, 2009], metaheurísticas são aplicadas em problemas de otimização sobre os quais há poucas informações:
  - Não se sabe previamente qual a solução ótima esperada;
  - Existem poucas ou nenhumas heurísticas disponíveis;
  - Força-bruta é inadequada, devido ao espaço de solução ser muito grande;
  - A solução candidata pode ser testada quanto a sua otimalidade;

Metaheurísticas

- Metaheurísticas são métodos de solução que coordenam procedimentos de buscas locais outras estratégias para criar um processo capaz de escapar de mínimos locais e pesquisar em todo o espaço de soluções;
- Metaheurísticas é utilizado como sinônimo de qualquer procedimento que empreguem estratégias para escapar de mínimos locais em espaços de busca de soluções complexas [Luzia and Rodrigues, 2009] [Luke, 2013].

Metaheurísticas Construtivas

- Soluções construtivas são aquelas "construídas de forma iterativa através da adição de componentes a uma solução inicial nula, sem *backtracking*, até que uma solução completa seja encontrada [Luzia and Rodrigues, 2009]";
- Uma metaheurística construtiva estabelece estratégias para a construção de uma solução, de modo que em cada passo, ela adiciona um elemento à solução parcial, de forma a obter os melhores resultados [Sucupira, 2004].

Metaheurísticas baseadas em população

- Métodos baseados em população contém um conjunto de soluções candidatas ao invés de uma solução única;
- Ao contrário de outros métodos, quando implementados de forma paralela, os algoritmos populacionais possibilitam que soluções candidatas afetem umas às outras.
- Soluções boas são propagadas e soluções ruins são rejeitadas, fazendo com que o algoritmo tenha convergência em direção a melhores soluções [Luke, 2013].

Metaheurísticas baseadas em população

- Alguns métodos populacionais tem inspiração na biologia
  - Por utilizar conceitos de biologia, genética e evolução, essas técnicas são conhecidos como **Computação Evolucionária** (*Evolutionary Computation - EC*);
  - Algoritmos relacionados a essas técnicas são conhecidos como **Algoritmos Evolucionários (EA)**;
  - Segundo [Luke, 2013], algoritmos evolucionários podem ser classificados em dois tipos principais[1]:
    - Algoritmos geracionais: alteram uma população inteira por iteração
    - Algoritmos de estado estacionário: atualizam uma amostra de candidatos da solução a cada época.

[1] Existem outras classificações e sub-classificações de algoritmos populacionais.

Busca Local (Otimização)

- Busca Local (Otimização)
  - Método heurístico para solução computacional de problemas de otimização difícil;
  - Pode ser usado para encontrar uma solução que maximize um critério entre várias soluções candidatas;
  - A solução inicia-se com uma configuração inicial (aleatória) e é modificada de forma a mover-se pelo espaço de busca (espaço de soluções)
    - Pequenas modificações são feitas nas soluções potenciais até atingir um critério de parada [Coppin, 2004].

Busca Informada e Não Informada

- Segundo [Coppin, 2004], métodos de busca podem ser subdivididos em busca informada e não informada
  - Busca Informada: usa informações adicionais sobre nós ainda não explorados para decidir quais nós examinar a seguir;
    - Frequentemente utilizam heurísticas para examinar o espaço de busca de forma mais eficiente;
  - Busca Não Informada: métodos que não usam informações sobre nós já explorados (também conhecida como busca cega).
- De forma geral, métodos que utilizam heurísticas são considerados Busca Informada e aqueles que não utilizam são Busca Não Informada [Coppin, 2004].

Complexidade Computacional

Complexidade Computacional

"A teoria de complexidade computacional busca quantificar a quantidade de recursos computacionais necessários para solução de uma determinada tarefa" [Arora and Barak, 2009]

- A complexidade pode ser definida em termos de:
  - Tempo: número de operações realizadas;
  - Espaço: quantidade de memória utilizada.

Complexidade computacional

- Em relação ao tempo, conta-se o número de funções elementares executadas, como adição, subtração, multiplicação e divisão;
- O número de operações deve ser contato com base em uma única operação *(adição/substração ou mult/divisão)*;
  - Complexidade adição e substração: $\Theta(n)$
  - Complexidade multiplicação: $\mathcal{O}(n \log n \log \log n)$ [2]

[2] Segundo [Harvey and Van Der Hoeven, 2019], multiplicações podem ser executadas em tempo $\mathcal{O}(n \log n)$

Ordem de Grandeza

- Definição: Sejam funções $f : \mathbb{R}^+ \to \mathbb{R}^+$ e $g : \mathbb{R}^+ \to \mathbb{R}^+$. A função f possui a mesma ordem de grandeza de g, definida por $f = \Theta(g)$, se existirem constantes positivas x_0, c_1 e c_2 tal que $x \geq x_0$ e $c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)$ [Gersting, 2014]
  - Essa definição indica que a função $f(x)$ é dominada assintoticamente pelas funções $c_1 \cdot g(x)$ e $c_2 \cdot g(x)$.

Ordem de Grandeza

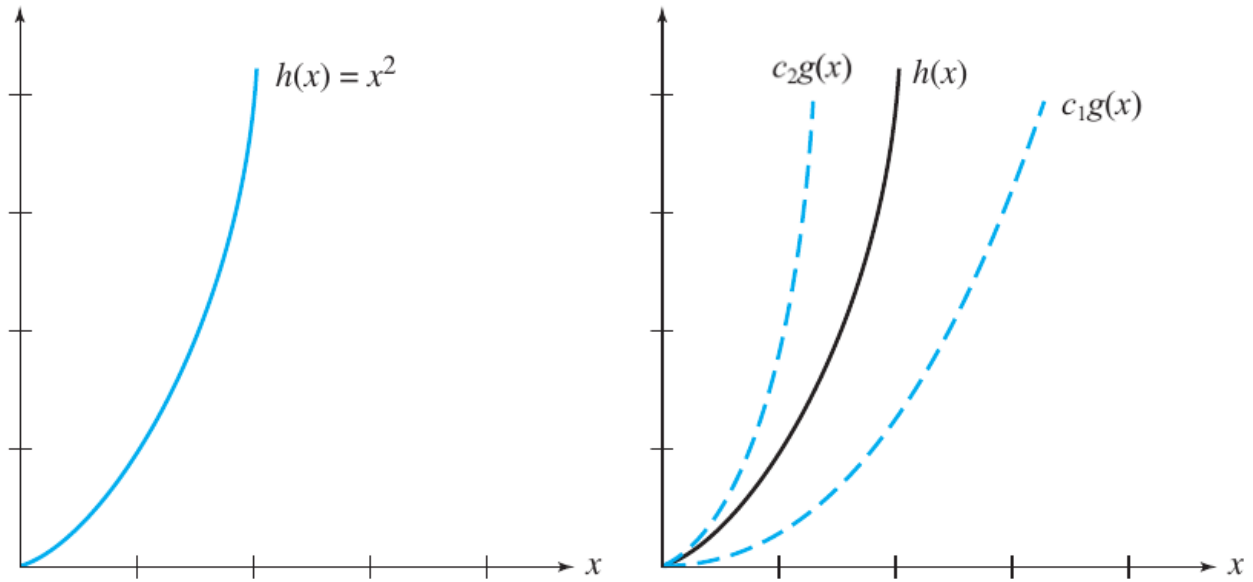
- Para a equação abaixo, com constantes positivas c_1 e c_2, podemos estabelecer as seguintes conclusões:

$$\boxed{c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)}$$

  - A função $f(x)$ irá se manter sempre dentro de um "envelope" das demais;
  - A expressão $c_1 \cdot g(x)$ será um limite inferior, enquanto a expressão $c_2 \cdot g(x)$ será um limite superior de $f(x)$;
  - A alteração do valor das constantes altera a largura do envelope, porém não altera sua forma;
  - Se f está contida, a partir de n_0, em um envelope definido por g, então f e g tem a mesma ordem de grandeza [da Silva, 2012].

Ordem de Grandeza

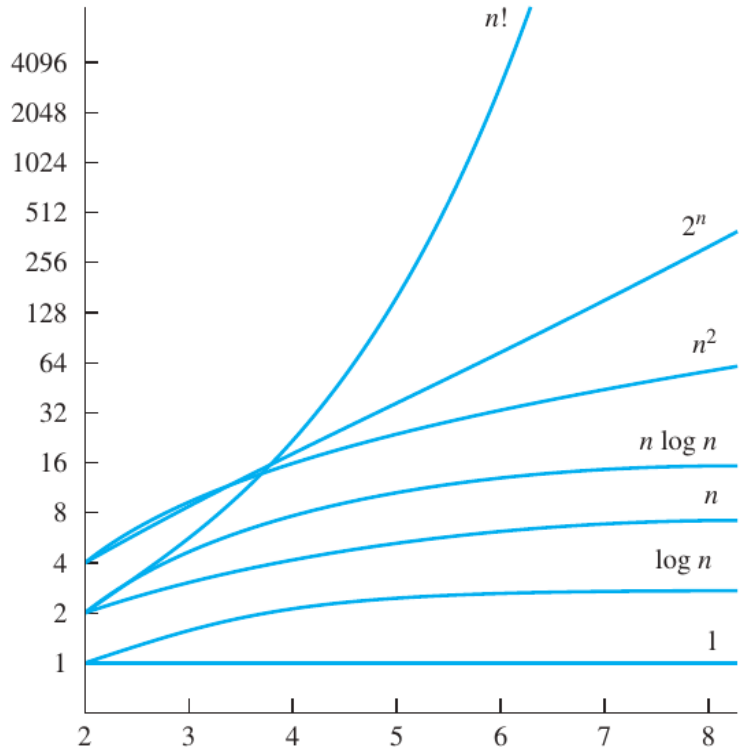
- A figura abaixo exibe o comportamento de uma função $h(x)$, dominada assintoticamente por $c_1 \cdot g(x)$ e $c_2 \cdot g(x)$.
  - Utilizam-se como limite, em geral, funções já conhecidas, para que seja possível inferir um comportamento da função avaliada.



Fonte: Adaptado de [Gersting, 2014]

Ordem de Grandeza

- Algumas funções genéricas com comportamento conhecido na literatura podem ser vistas na figura abaixo.



Fonte: [Rosen, 2019]

As funções são utilizadas como estimativa de comportamento. Gráfico em escala logarítmica.

Ordem de Grandeza

- São adotadas as seguintes terminologias para complexidade de algoritmos:

Complexity	*Terminology*
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n \log n)$	Linearithmic complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta(b^n)$, where $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity

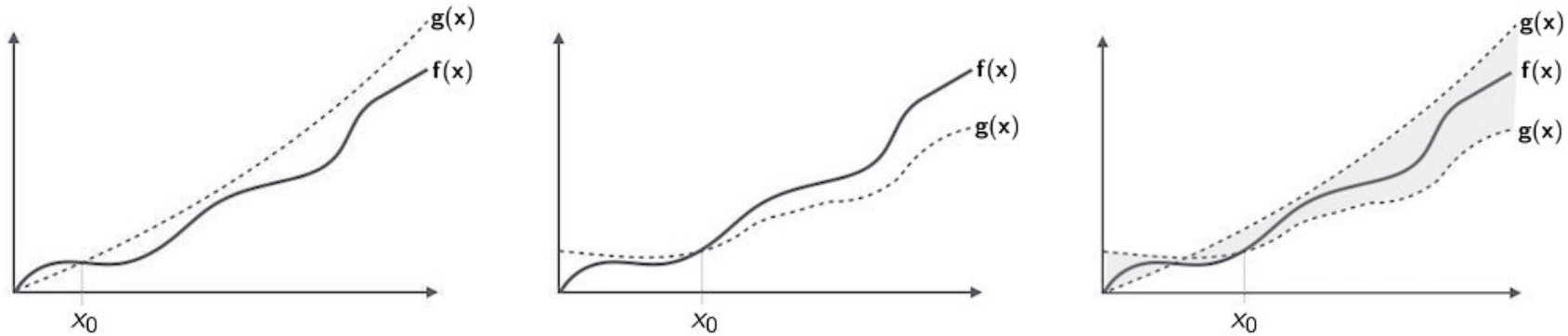
Fonte: [Rosen, 2019]

Notações Big-O, Big-Ω e Big-Θ

- A notação Big-O é utilizada como uma estimativa teórica do limite superior de execução de um algoritmo
  - Está associada à execução do algoritmo no pior caso, ou seja, ao tempo máximo (ou tamanho máximo em memória) para finalização da execução do algoritmo;
  - A notação é criada com base no crescimento de funções;
- Além da notação Big-O, mais utilizada, também existem as notações Big-Ω e Big-Θ
  - Big-Ω: expressa o limite inferior do algoritmo - associada ao melhor caso em complexidade de tempo ou espaço;
  - Big-Θ: expressa limites inferiores e superiores do algoritmo.

Notações Big-O, Big-Ω e Big-Θ

- O comportamento das notações Big-O, Big-Ω e Big-Θ podem ser vistas na figura abaixo.



(a) Big-O (b) Big-Ω (c) Big-Θ

Fonte: Adaptado de [Tutorials Point, 2021]

Problemas $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-Difíceis e $\mathcal{NP}$-Completos

Tratabilidade

- Definição 1: Um problema que é solucionado por um algoritmo com complexidade inferior ou igual à polinomial para o pior caso é denominado tratável [Rosen, 2019].
  - Espera-se que o algoritmo produza uma solução para o problema em tempo razoável;
  - Na prática, caso o grau do polinômio seja alto (por exemplo, 100), o problema poderá se tornar inviável.

- Definição 2: Um problema que é solucionado por um algoritmo com complexidade superior à polinomial para o pior caso é denominado intratável [Rosen, 2019].

Problemas Solúveis e Insolúveis

- Definição: Um problema é denominado insolúvel caso não exista um algoritmo para resolvê-lo
  - Os problemas que são resolvidos por algoritmos são denominados solúveis;
  - Dentre os problemas insolúveis mais conhecidos, destaca-se o Problema da Parada, proposto por Alan Turing.

Problemas $\mathcal{P}$ e $\mathcal{NP}$

- Definição 1: Problemas tratáveis são pertencentes à classe de problemas polinomiais $(\mathcal{P})$ [Rosen, 2019].

- Definição 2: Problemas que não podem ser solucionados em tempo polinomial pertencem à classe $\mathcal{NP}$ [Rosen, 2019]
  - A abreviação $\mathcal{NP}$ corresponde à "Tempo Polinomial Não Determinístico" (*Nondeterministic Polynomial Time*).
  - Alguns problemas $\mathcal{NP}$ são classificados ainda em $\mathcal{NP}$-Completos e $\mathcal{NP}$-Difíceis.

Problemas $\mathcal{NP}$-Difíceis e $\mathcal{NP}$-Completos

INSTITUTO FEDERAL
Minas Gerais

- Um problema $\mathcal{C}$ é $\mathcal{NP}$-Completo quando:
  1. O problema for $\mathcal{NP}$;
  2. Todo problema $\mathcal{NP}$ é redutível ao problema $\mathcal{C}$ por uma transformação em tempo polinomial.

- Um problema $\mathcal{C}$ é $\mathcal{NP}$-Difícil se respeitar a condição 2, independentemente de respeitar a condição 1
  - Um problema é $\mathcal{NP}$-Difícil quando é considerado tão difícil de ser solucionado quanto um problema $\mathcal{NP}$-Completo [Szajda, 2014].

Problemas $\mathcal{NP}$-Difíceis e $\mathcal{NP}$-Completos

- $\mathcal{NP}$-Completos são um conjunto restrito de problemas que possuem correlações entre si;
  - Um problema $\mathcal{NP}$-Completo pode ser mapeado (ou transformado) em outro problema $\mathcal{NP}$-Completo;
  - Com isso, caso um problema $\mathcal{NP}$-Completo seja resolvido em tempo polinomial por um algoritmo, então todos os demais problemas $\mathcal{NP}$-Completos também o serão;
  - O primeiro problema $\mathcal{NP}$-Completo é o problema da Satisfabilidade (SAT);
  - Existem mais de 3000 problemas $\mathcal{NP}$-Completos, que podem ser mapeados no SAT [Rosen, 2019]
    - Dentre eles, destacam-se os 21 Problemas de Karp, listados por Richard Karp em 1972.

Problemas $\mathcal{NP}$-Completos

- Problema da Satisfabilidade (SAT)
  - O problema consiste em verificar se a valoração de variáveis de uma determinada fórmula booleana satisfaz à fórmula em questão, tornando-a verdadeira;
  - O problema deve estar na Forma Normal Conjuntiva (FNC), correspondente à "expressão booleana formada por conjunções de cláusulas, cada qual formada por disjunção de variáveis booleanas" [da Silva, 2017]
    - Exemplo 1: $E_1 = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee x_2) \wedge (x_3 \vee x_2)$
    - Exemplo 2: $E_2 = (x_1 \vee x_4) \wedge (\neg x_1 \vee \neg x_3) \wedge (x_3 \vee x_2) \wedge (\neg x_3)$

Problemas $\mathcal{NP}$-Completos

- Problema da Satisfabilidade (SAT)
  - O SAT consiste em avaliar uma expressão na FNC, de modo a verificar se existe uma atribuição de valores lógico booleanos às variáveis $x_i, i = 1, 2, ..., n$ que torne a expressão verdadeira
    - Exemplo: A expressão abaixo é satisfatível se $x_1 = V$, $x_2 = F$ e $x_3 = V$.

$$\boxed{E_1 = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3 \vee x_2) \wedge (x_3 \vee x_2)}$$

  - Melhor algoritmo: $O(2^n)$.

Problemas $\mathcal{NP}$-Completos

- Problema da Caixeiro-Viajante[3] (TSP)
  - Dado um conjunto de cidades, deve-se determinar a menor rota que passa por todas as cidades uma única vez e retorna à cidade de origem.
- Problema do Clique[4]
  - Dado um grafo, o problema busca encontrar subgrafos completos ("cliques") em um grafo.
  - Um clique corresponde a um conjunto de nós em que todos os elementos estão conectados entre si.

[3] *Traveling Salesman Problem.*

[4] Também chamado de Problema do Clique Máximo.

Problemas $\mathcal{NP}$-Completos

- Problema da Mochila (Knapsack Problem)
  - Dado um conjunto de itens, que possuem pesos e valores, determinar o número de itens a serem colocados na mochila de modo a maximizar o valor total, respeitando a restrição de peso máximo da mochila.
  - Modelagem Matemática [da Silva, 2017]:

$$max\ z = \sum_{j=1}^{n} c_j x_j \qquad \text{(maximização de valor)}$$

$$\text{suj. a} \sum_{j=1}^{n} w_j x_j \leq K \qquad \text{(restrição de capacidade)}$$

Heurísticas e Metaheurísticas

- Devido ao custo computacional de solução de alguns problemas, são usados algoritmos aproximados
  - Esses algoritmos, principalmente aqueles com estratégias de mais alto nível, que buscam escapar de ótimos locais - as Metaheurísticas - serão o foco da disciplina.
- Dentre os algoritmos a serem estudados, destacam-se:
  - Hill-Climbing;
  - Simulated Annealing;
  - Algoritmos Genéticos;
  - GRASP e Path-Relinking;
  - Busca Tabu;
  - Ant Colony Optimization;
  - VND e VNS.

Referências I


Arora, S. and Barak, B. (2009).
Computational Complexity: A Modern Approach.
Cambridge University Press.


Belfiore, P. and Fávero, L. P. (2013).
Pesquisa operacional para cursos de engenharia.
Elsevier, 1 edition.


Coppin, B. (2004).
Artificial intelligence illuminated.
Jones and Bartlett illuminated series. Jones and Bartlett Publishers, 1 edition.


da Silva, D. M. (2012).
Matemática discreta - slides de aula.


da Silva, D. M. (2017).
Métodos heurísticos - fundamentos - slides de aula.


Gersting, J. L. (2014).
Mathematical Structures for Computer Science.
W. H. Freeman and Company, 7 edition.


Harvey, D. and Van Der Hoeven, J. (2019).
Integer multiplication in time O(n log n).
working paper or preprint.

Referências II

Luke, S. (2013).
Essentials of Metaheuristics (Second Edition).
lulu.com, 2 edition.
[Online]; Disponível em: https://cs.gmu.edu/~sean/book/metaheuristics/.

Luzia, L. F. and Rodrigues, M. C. (2009).
Estudo sobre as metaheurísticas.
[Online]; acessado em 22 de Setembro de 2020. Disponível em:
https://www.ime.usp.br/~gold/cursos/2009/mac5758/LeandroMauricioHeuristica.pdf.

Rosen, K. H. (2019).
Discrete Mathematics and Its Applications.
McGraw-Hill, 8 edition.

Sucupira, I. R. (2004).
Métodos heurísticos genéricos: metaheurísticas e hiper-heurísticas.
PhD thesis, Universidade de SÃ£o Paulo.
Acessado em 28 de Setembro de 2020. Disponível em:
https://www.ime.usp.br/~igorrs/monografias/metahiper.pdf.

Szajda, D. (2014).
Np-completeness.
[Online]; acessado em 29 de Abril de 2021. Disponível em: http:
//www.mathcs.richmond.edu/~dszajda/classes/cs315/Spring_2014/lectures/NP-Completeness.pdf.

Referências III

Tutorials Point (2021).
Data structures - asymptotic analysis.
[Online]; acessado em 17 de Março de 2021. Disponível em:
https://www.tutorialspoint.com/data_structures_algorithms/asymptotic_analysis.htm.