Meta-heurísticas

Computação Bioinspirada

Paulo Henrique Ribeiro Gabriel

Faculdade de Computação
Universidade Federal de Uberlândia

2023/1

Nesta aula

- Problemas de otimização
- Conceito de busca local
- Heurísticas e Meta-heurísticas

Otimização

- Um problema de otimização tem uma forma geral:

$$\text{minimizar ou maximizar } f(x)$$

$$\text{sujeito a } x \in S$$

Otimização

- Um problema de otimização tem uma forma geral:

$$\text{minimizar ou maximizar } f(x)$$

$$\text{sujeito a } x \in S$$

- Nesse caso, S é um conjunto em $\mathbb{R}^n$ e $f(x)$ é uma função de valor real definida sobre S

Otimização

- Um problema de otimização tem uma forma geral:

$$\text{minimizar ou maximizar } f(x)$$

$$\text{sujeito a } x \in S$$

- Nesse caso, S é um conjunto em $\mathbb{R}^n$ e $f(x)$ é uma função de valor real definida sobre S
  - S é chamado espaço de busca (domínio viável)

Otimização

- Um problema de otimização tem uma forma geral:

$$\text{minimizar ou maximizar } f(x)$$

$$\text{sujeito a } x \in S$$

- Nesse caso, S é um conjunto em $\mathbb{R}^n$ e $f(x)$ é uma função de valor real definida sobre S
  - S é chamado espaço de busca (domínio viável)
  - f é a função objetivo

Otimização Combinatória

Processo de encontrar a melhor solução (ou solução ótima) para problemas com um conjunto discreto de soluções viáveis

- Dados:

Otimização Combinatória

Processo de encontrar a melhor solução (ou solução ótima) para problemas com um conjunto discreto de soluções viáveis

- Dados:
  - Conjunto discreto de soluções S

Otimização Combinatória

Processo de encontrar a melhor solução (ou solução ótima) para problemas com um conjunto discreto de soluções viáveis

- Dados:
  - Conjunto discreto de soluções S
  - Função objetivo $f(x) : x \in S \to \mathbb{R}$

Otimização Combinatória

Processo de encontrar a melhor solução (ou solução ótima) para problemas com um conjunto discreto de soluções viáveis

- Dados:
  - Conjunto discreto de soluções S
  - Função objetivo $f(x) : x \in S \to \mathbb{R}$
- Objetivo:

Otimização Combinatória

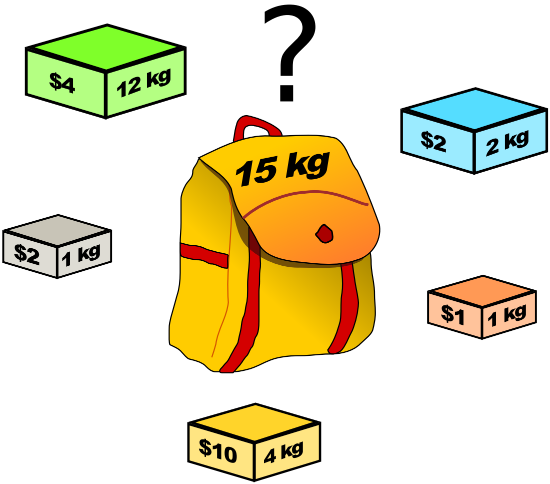
Processo de encontrar a melhor solução (ou solução ótima) para problemas com um conjunto discreto de soluções viáveis

- Dados:
  - Conjunto discreto de soluções S
  - Função objetivo $f(x) : x \in S \rightarrow \mathbb{R}$
- Objetivo:
  - Encontrar $x \in S : f(x) \leq f(y), \forall y \in S$

Otimização Combinatória: Aplicações

- Roteamento de veículos
- Sequenciamento de tarefas
- Empacotamento
- Gestão de estoque e produção
- Posicionamento
- Atribuição de recursos

Otimização Combinatória: Exemplo



Problema da Mochila Binária

Informalmente: Dado um conjunto de itens, cada um com um peso e um valor, determine quais itens incluir na coleção para que o peso total seja menor ou igual a um determinado limite e o valor total seja o maior possível

Problema da Mochila Binária

Informalmente: Dado um conjunto de itens, cada um com um peso e um valor, determine quais itens incluir na coleção para que o peso total seja menor ou igual a um determinado limite e o valor total seja o maior possível

Formalmente: Dado um conjunto de n itens numerados 1 a n, cada um com um peso w_i e um valor v_i, juntamento com uma capacidade de peso máximo W,

$$\text{maximizar } \sum_{i=1}^{n} v_i x_i$$

$$\text{sujeito a} \sum_{i=1}^{n} w_i x_i \leq W \text{ e } x_i \in \{0, 1\}$$

Problema da Mochila Binária

Problema de busca (*Maximum Knapsack*):

Instância: Conjunto finito U, um tamanho $s(u) \in Z^+$ e um valor $v(u) \in Z^+$ para cada $u \in U$ e um inteiro positivo $B \in Z^+$

Problema da Mochila Binária

Problema de busca (*Maximum Knapsack*):

Instância: Conjunto finito U, um tamanho $s(u) \in Z^+$ e um valor $v(u) \in Z^+$ para cada $u \in U$ e um inteiro positivo $B \in Z^+$

Pergunta: Existe um subconjunto $U' \subseteq U$ tal que $\sum_{u \in U'} s(u) \leq B$

Problema da Mochila Binária

- O problema *Maximum Knapsack*[1] é $\mathcal{NP}$-Completo mesmo quando $s(u) = v(u) \; \forall u \in U$
- O caso $s(u) = v(u)$ é chamado Soma Máxima de Subconjuntos (*Maximum Subset Sum*)
- Caso particular do problema do Particionamento[2]

[1] GAREY, Michael R. and JOHNSON, David S. *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY: W.H. Freeman, 1979.

[2] KARP, R. M. Reducibility among Combinatorial Problems. *Complexity of Computer Computations* Springer US, 1972. Disponível em: http://dx.doi.org/10.1007/978-1-4684-2001-2_9.

Problema da Mochila Binária

Exemplo

Problema da Mochila Binária

Exemplo

- Pesos: $w = \{3, 2, 4, 1\}$

Problema da Mochila Binária

Exemplo

- Pesos: $w = \{3, 2, 4, 1\}$
- Valores: $v = \{8, 3, 9, 6\}$

Problema da Mochila Binária

Exemplo

- Pesos: $w = \{3, 2, 4, 1\}$
- Valores: $v = \{8, 3, 9, 6\}$
- $W = 5$

Problema da Mochila Binária

Exemplo

- Pesos: $w = \{3, 2, 4, 1\}$
- Valores: $v = \{8, 3, 9, 6\}$
- $W = 5$

- Valor ótimo: 15 (itens de peso 4 e 1)

Programação Dinâmica

- Para cada item, podemos escolher entre "empacotá-lo ou descartá-lo"

Programação Dinâmica

- Para cada item, podemos escolher entre "empacotá-lo ou descartá-lo"
- Seja $F(i, j)$ uma função que nos dá o valor ótimo para os primeiros i itens e um limite de peso j

Programação Dinâmica

- Para cada item, podemos escolher entre “empacotá-lo ou descartá-lo”
- Seja $F(i, j)$ uma função que nos dá o valor ótimo para os primeiros i itens e um limite de peso j
- Essa função é dada pela relação de recorrência:

Programação Dinâmica

- Para cada item, podemos escolher entre "empacotá-lo ou descartá-lo"
- Seja $F(i, j)$ uma função que nos dá o valor ótimo para os primeiros i itens e um limite de peso j
- Essa função é dada pela relação de recorrência:

$$F(i,j) = \begin{cases} F(i-1,j), \text{ se } w_i > j \\ \max\{F(i-1,j), F(i-1,j-w_i)+v_i\}, \text{ se } w_i \leq j \end{cases}$$

Problema da Mochila Binária

```
MOCHILA(n, w, v, W)
for i ← 0 to W
    F[0, i] ← 0
    for j ← 1 to n
        x ← F[j − 1, i]
        if i − w_j ≥ 0
            y ← F[j − 1, i − w_j] + v_j
            if x < y
                x ← y
        F[j, i] ← x
return F[n, W]
```

phrg@ufu.br

Problema da Mochila Binária

- Má noticia: algoritmo pseudo-polinomial

Problema da Mochila Binária

- Má noticia: algoritmo pseudo-polinomial
- Tempo de execução: $\mathcal{O}(nW)$

Problema da Mochila Binária

- Má noticia: algoritmo pseudo-polinomial
- Tempo de execução: $\mathcal{O}(nW)$
- Ou seja, o algoritmo é muito sensível às variações de W

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)
  - Programação dinâmica

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)
  - Programação dinâmica
  - Planos de corte

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)
  - Programação dinâmica
  - Planos de corte
  - *branch and cut* (ramificação e corte)

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)
  - Programação dinâmica
  - Planos de corte
  - *branch and cut* (ramificação e corte)
- Muitos problemas difíceis de otimização combinatória ainda não são resolvidos de maneira exata e requerem bons métodos heurísticos

Otimização Combinatória

- Muito progresso nos últimos anos para encontrar a solução exata (provavelmente ótima)
  - Programação dinâmica
  - Planos de corte
  - *branch and cut* (ramificação e corte)
- Muitos problemas difíceis de otimização combinatória ainda não são resolvidos de maneira exata e requerem bons métodos heurísticos
- O objetivo dos métodos heurísticos é produzir rapidamente soluções de boa qualidade, sem necessariamente fornecer qualquer garantia de qualidade da solução

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas
  - Randomização controlada

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas
  - Randomização controlada
  - Estratégias de aprendizado

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas
  - Randomização controlada
  - Estratégias de aprendizado
  - Decomposição induzida

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas
  - Randomização controlada
  - Estratégias de aprendizado
  - Decomposição induzida
  - Meta-heurística

Métodos para Problemas de Otimização Combinatória

- Algoritmos exatos: programação dinâmica, planos de corte, ramificação, etc.
- Algoritmos de aproximação
- Heurísticas
  - Randomização controlada
  - Estratégias de aprendizado
  - Decomposição induzida
  - Meta-heurística
  - Busca local

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)
  - Algoritmos genéticos

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)
  - Algoritmos genéticos
  - Busca tabu (*tabu search*)

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)
  - Algoritmos genéticos
  - Busca tabu (*tabu search*)
  - Otimização de colônia de formigas

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)
  - Algoritmos genéticos
  - Busca tabu (*tabu search*)
  - Otimização de colônia de formigas
  - Pesquisa de vizinhança variável (*variable neighborhood search*, VNS)

Otimização Combinatória

- Meta-heurísticas são procedimentos de alto nível que coordenam heurísticas simples, como busca local, para encontrar soluções de melhor qualidade do que aquelas encontradas apenas pela heurística simples
- Exemplos:
  - Recozimento simulado (*simulated annealing*)
  - Algoritmos genéticos
  - Busca tabu (*tabu search*)
  - Otimização de colônia de formigas
  - Pesquisa de vizinhança variável (*variable neighborhood search*, VNS)
  - GRASP

Quando devemos usar meta-heurísticas?

- Em circunstâncias onde a complexidade do problema (ou o tempo disponível) para solução não permitem solução exata

- Incerteza nos dados do problema

Busca Local

- Para definir a busca local, é preciso especificar uma estrutura de vizinhança local
- Dada uma solução x, os elementos da vizinhança $N(x)$ de x são aquelas soluções y que podem ser obtidas pela aplicação de uma modificação elementar a x
  - Essas modificações são, comumente, chamadas movimentos

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa
- Possíveis movimentações:

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa
- Possíveis movimentações:
  1. $s' = (0, 0, 0, 1)$

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa
- Possíveis movimentações:
  1. $s' = (0, 0, 0, 1)$
  2. $s' = (0, 1, 1, 1)$

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa
- Possíveis movimentações:
  1. $s' = (0, 0, 0, 1)$
  2. $s' = (0, 1, 1, 1)$
  3. $s' = (0, 1, 0, 0)$

Vizinhança no problema da mochila

- Seja $s = (0, 1, 0, 1)$ uma solução para o problema da mochila
- Vamos modificar alguma variável s_j de 1 para 0 ou vice-versa
- Possíveis movimentações:
  1. $s' = (0, 0, 0, 1)$
  2. $s' = (0, 1, 1, 1)$
  3. $s' = (0, 1, 0, 0)$
  4. $s' = (1, 1, 0, 1)$

Busca Local

- Dada uma solução inicial x_0, uma vizinhança $N(x)$ e uma função $f(x)$ a ser minimizada:

  Busca Local(x_0)

  $x \leftarrow x_0$
  while $\exists y \in N(x) : f(y) < f(x)$
      $x \leftarrow y$
      return x

- No final, x é um mínimo local de $f(x)$

phrg@ufu.br

Busca Local

Ótimo Local

$$x \in N(x) \subset S : f(x) \leq f(y), \forall y \in N(x)$$

Busca Local

- A eficácia da busca local depende de vários fatores:
  - Estrutura da vizinhança
  - Função a ser minimizada
  - Solução inicial

Um algoritmo guloso

Construa uma solução, um elemento de cada vez:

- Defina os elementos candidatos

Um algoritmo guloso

Construa uma solução, um elemento de cada vez:

- Defina os elementos candidatos
- Aplique uma função gulosa a cada elemento candidato

Um algoritmo guloso

Construa uma solução, um elemento de cada vez:

- Defina os elementos candidatos
- Aplique uma função gulosa a cada elemento candidato
- Classifique os elementos de acordo com o valor da função gulosa

Um algoritmo guloso

Construa uma solução, um elemento de cada vez:

- Defina os elementos candidatos
- Aplique uma função gulosa a cada elemento candidato
- Classifique os elementos de acordo com o valor da função gulosa
- Adicione o melhor elemento classificado à solução

GRASP

Greedy randomized adaptive search procedure[3]:

1. Construa uma solução aleatória gulosa
2. Use a busca local para melhorar a solução construída
3. Atualize a melhor solução encontrada

[3] FEO, T. A.; RESENDE, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, v. 8, n. 2, p. 67–71, 1989.

Primeira fase: Construção

- Constrói uma solução viável
- Use a "gula" para construir uma lista restrita de candidatos e aplique a aleatoriedade para selecionar um elemento da lista
- Use aleatoriedade para construir uma lista restrita de candidatos e aplique a gula para selecionar um elemento da lista

Segunda fase: Busca local

- Busca na vizinhança atual até que um ótimo local seja encontrado
- As soluções geradas pelo procedimento de construção não são necessariamente ótimas
  - Questões já discutidas. . .

GRASP para Mochila Binária

E a Computação Bioinspirada?

- Diversas meta-heurísticas são métodos populacionais
- Ou seja, lidam com um conjunto de soluções simultaneamente
- Maior cobertura do espaço de busca (conjunto S)
- Mais diversidade

Agradecimentos

Esse material foi baseados em notas escritas pelos professores Dra. Aurora Pozo (UFPR) e Dr. Cláudio Meneses (UFABC)