

# VIZINHANÇA E ALGORITMOS DE BUSCA LOCAL

DCE529 - Algoritmos e Estruturas de Dados III

Atualizado em: 28 de outubro de 2024

Iago Carvalho

Departamento de Ciência da Computação



# ALGORITMO, HEURÍSTICA E META-HEURÍSTICA

Um algoritmo computa a resposta exata para um problema específico

Uma heurística computa uma solução aproximada para um problema específico

Uma meta-heurística é um *framework* para construção de heurísticas

- Não resolvem um problema específico
- Possibilita criar heurísticas para diversos problemas
- São extremamente generalizáveis

Problemas NP-Completo possuem um número exponencial de soluções

- Impraticável listar todas elas

Meta-heurísticas exploram um subconjunto destas soluções

Uma meta-heurística é um *framework*, um "guia", sobre como explorar esse subconjunto de soluções

- Quanto mais *eficaz* e mais *eficiente* for esta amostragem **melhor** é a heurística resultante

# CARACTERÍSTICAS DE META-HEURÍSTICAS

## Simplicidade

- São baseadas em princípios claros

## Generalidade

- Podem ser facilmente generalizadas para diversos problemas

## Eficácia

- Produzem soluções de boa qualidade

## Eficiência

- Baixo custo computacional

## Diversificação

- Como ela realiza a busca
- Ato de explorar uma grande área do espaço de buscas

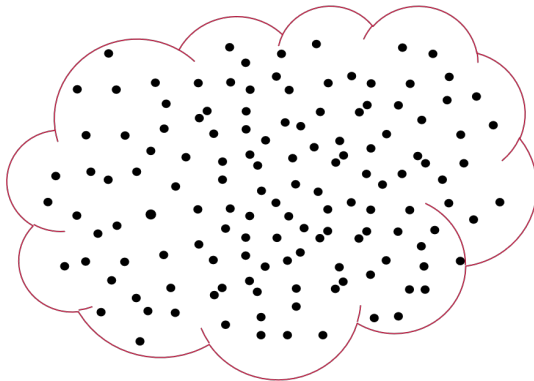
## Intensificação

- Buscas realizadas em soluções próximas a outras
- No geral, tende-se a intensificar a busca próximo a soluções de boa qualidade

# CONJUNTO DE SOLUÇÕES

Definimos o conjunto de soluções como  $\Gamma$

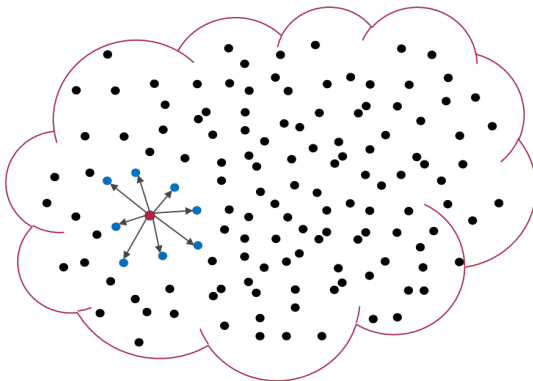
- Define-se uma solução como  $S \in \Gamma$
- Uma busca é realizada neste conjunto de soluções



# VIZINHANÇA

Considera-se soluções vizinhas como sendo soluções próximas umas das outras

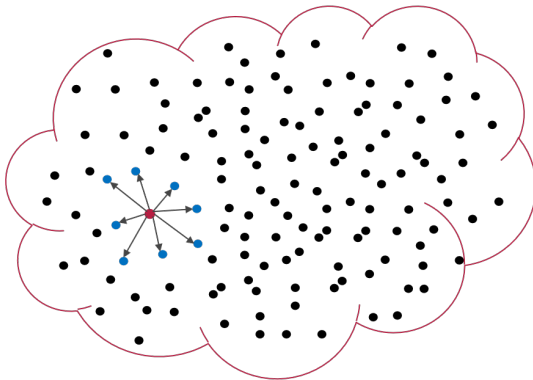
- Função de proximidade definida anteriormente
- Normalmente, relacionado a quantidade de diferentes elementos entre as soluções



# VIZINHANÇA - DEFINIÇÃO FORMAL

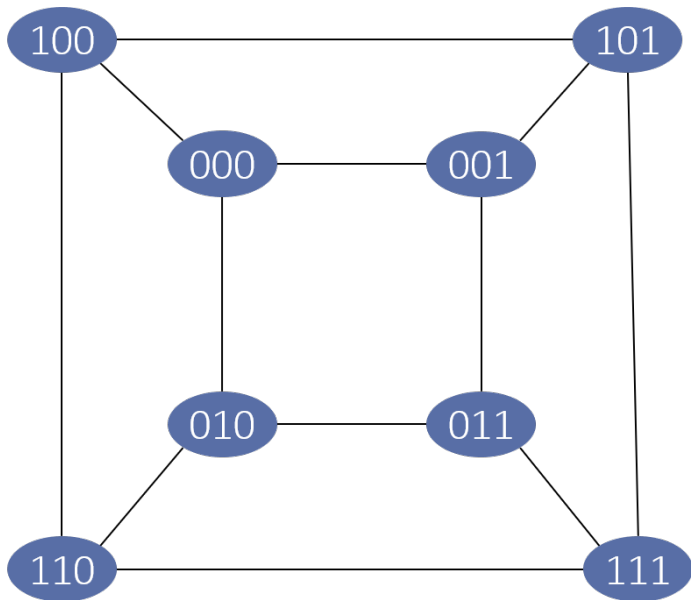
Uma vizinhança é uma função  $N : \Gamma \mapsto 2^\Gamma$

- Mapeia uma solução  $S \in \Gamma$  a um subconjunto  $N(S) \subseteq \Gamma$

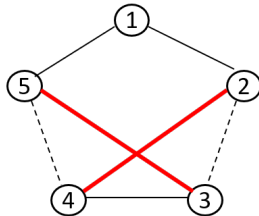
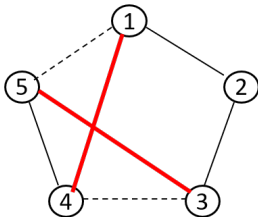
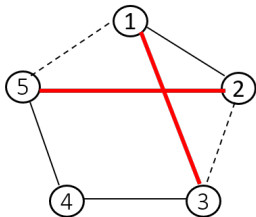
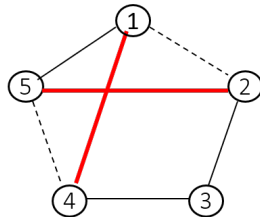
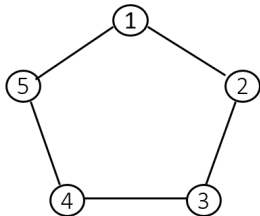
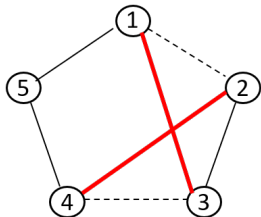




## VIZINHANÇA NO ESPAÇO $\mathbb{B}^3$



# VIZINHANÇA PARA O PROBLEMA DO CAIXEIRO VIAJANTE



Duas soluções em  $\Gamma$  podem ou não ser vizinhas

- Depende da função de vizinhança aplicada

Meta-heurísticas de busca local são algoritmos que possuem uma única solução

- Realizam buscas no espaço de soluções aplicando uma ou mais funções de vizinhança sobre uma solução inicial
  - Solução inicial muitas vezes criada com uma heurística construtiva

# UMA PRIMEIRA META-HEURÍSTICA DE BUSCA LOCAL

Random walk

Inicia de uma solução viável

- Criada a partir de uma heurística construtiva

Iteração

- Move-se para um vizinho aleatório

Critério de parada

- Tempo ou número de iterações sem melhora

# MELHORANDO UM POQUINHO...

Hill climbing

Inicia de uma solução viável

- Criada a partir de uma heurística construtiva

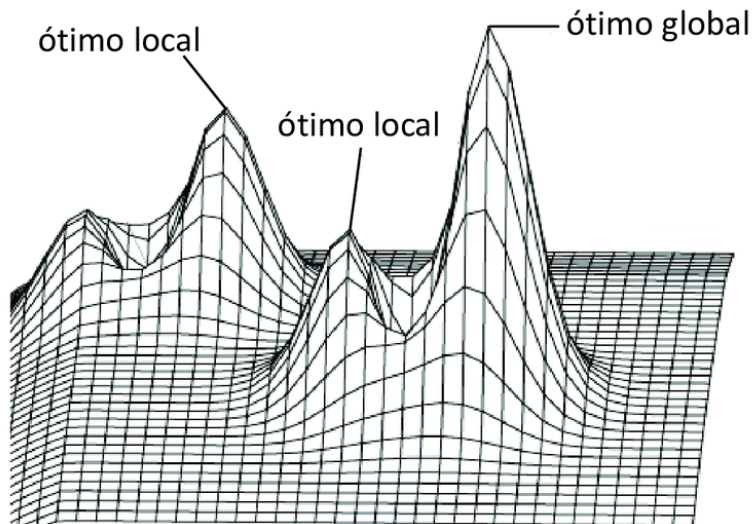
Iteração

- Move-se para um vizinho aprimorante
  - Vizinho de melhor qualidade

Critério de parada

- Não existe nenhum vizinho aprimorante
- Ótimo local

# ÓTIMO LOCAL E GLOBAL



VNS/VND [▶ Link](#)

GRASP [▶ Link](#)

ILS [▶ Link](#) [▶ Link](#)

Busca tabu [▶ Link](#)

...

```
procedimento  $GRASP(f(.), g(.), N(.), GRASP_{max}, s)$   
1   $f^* \leftarrow \infty$ ;  
2  para ( $Iter = 1, 2, \dots, GRASP_{max}$ ) faça  
3       $Construcao(g(.), \alpha, s)$ ;  
4       $BuscaLocal(f(.), N(.), s)$ ;  
5      se ( $f(s) < f^*$ ) então  
6           $s^* \leftarrow s$ ;  
7           $f^* \leftarrow f(s)$ ;  
8      fim-se;  
9  fim-para;  
10  $s \leftarrow s^*$ ;  
11 Retorne  $s$ ;  
fim  $GRASP$ 
```



# GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE

O algoritmo GRASP foi proposto por Fao e Resende (1995)

- Feo, T. A. e Resende, M. G. C. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, 6:109-133, 1995 [▶ Link](#)

O algoritmo é iterativo, e cada iteração é composta por duas fases

1. Construção
  - Constroi uma solução de forma parcialmente gulosa, semi-aleatória
2. Refinamento
  - Utiliza buscas locais para melhorar a solução inicial obtida

# GRASP - FASE DE CONSTRUÇÃO

A fase de construção é parcialmente gulosa e parcialmente aleatória

A parte gulosa é necessária para

- Gerar soluções de boa qualidade
- Acelerar a convergência a ótimos locais (e possivelmente globais)

Necessário inserir aleatoriedade devido a natureza iterativa do algoritmo

- A cada iteração, deve-se gerar uma solução diferente
- Desta forma, é possível explorar, de forma efetiva, o espaço de soluções

## GRASP - FASE DE REFINAMENTO

Aplica um algoritmo de busca local para refinar a solução obtida anteriormente

Inicialmente, realiza um *hill climbing* utilizando um único esquema de vizinhança

Entretanto, outras estratégias mais inteligentes podem ser utilizadas

- VNS
- ILS
- Guided local search
- ...

Veremos algumas destas estratégias mais a frente

```
procedimento  $GRASP(f(\cdot), g(\cdot), N(\cdot), GRASP_{max}, s)$   
1   $f^* \leftarrow \infty$ ;  
2  para ( $Iter = 1, 2, \dots, GRASP_{max}$ ) faça  
3       $Construcao(g(\cdot), \alpha, s)$ ;  
4       $BuscaLocal(f(\cdot), N(\cdot), s)$ ;  
5      se ( $f(s) < f^*$ ) então  
6           $s^* \leftarrow s$ ;  
7           $f^* \leftarrow f(s)$ ;  
8      fim-se;  
9  fim-para;  
10  $s \leftarrow s^*$ ;  
11 Retorne  $s$ ;  
fim  $GRASP$ 
```

O GRASP conta com dois parâmetros

1. *GRASP*<sub>max</sub>: Define o número máximo de iterações a ser realizada pelo algoritmo
2.  $\alpha$ : Fator de aleatoriedade da heurística construtiva

# VARIABLE NEIGHBORHOOD DESCENT

O algoritmo VND foi proposto por Mladenovic e Hansen (1997)

- Mladenović N, Hansen P. Variable neighborhood search. Computers & operations research. 1997 Nov 1;24(11):1097-100. [▶ Link](#)
- Uma melhor e mais completa referência pode ser encontrada em [▶ Link](#)

Explora diversas estruturas de vizinhança de forma iterativa

- Normalmente, explora vizinhanças gradativamente mais complexas
- Sempre que há uma melhora, retorna-se a vizinhança de menor complexidade

Diferentes vizinhanças induzem a diferentes ótimos locais

- Explorar diversas vizinhanças pode levar a ótimos locais de melhor qualidade

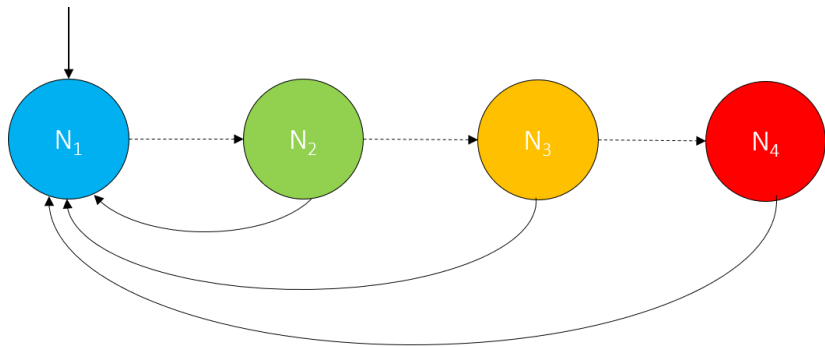
Preocupação com o tempo de execução

- Vizinhanças menos complexas são exploradas primeiro
- Vizinhanças complexas são pouco exploradas
  - Utilizadas como *último recurso* para sair de um ótimo local

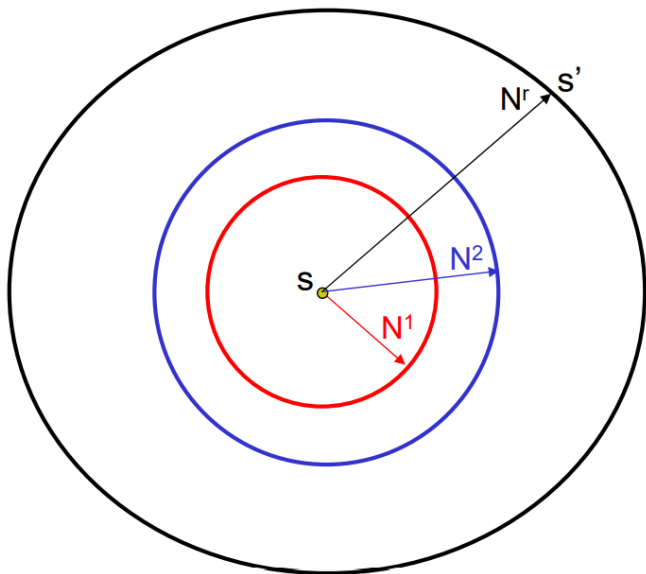
```
1  Seja  $s_0$  uma solução inicial e  $r$  o número de estruturas  
   de vizinhança;  
2   $s \leftarrow s_0$ ;                                {Solução corrente}  
3   $k \leftarrow 1$ ;                                {Tipo de estrutura de vizinhança}  
4  enquanto ( $k \leq r$ ) faça  
5       $s' \leftarrow \text{MelhorVizinho}(s, k)$ ;  
6       $\text{AltereVizinhança}(s, s', k)$   
7  fim-enquanto;  
8  Retorne  $s$ ;  
fim VND;
```



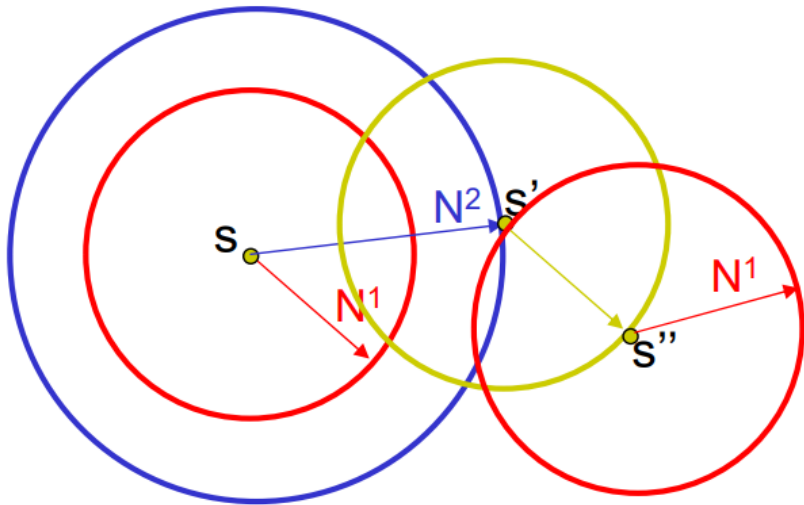
## VND - TROCA DE VIZINHANÇAS



## VND - VIZINHANÇAS MAIS COMPLEXAS



## VND - TROCA DE VIZINHANÇAS



O VND possui um único parâmetro

Conjunto  $r$  de esquemas de vizinhança

## VND - PSEUDO-CÓDIGO MELHORADO

```
1  Seja  $s_0$  uma solução inicial e  $r$  o número de estruturas de vizinhança;
2   $s \leftarrow s_0$ ; {Solução atual}
3  repita
4       $k \leftarrow 1$ ;           {Vizinhança corrente}
5      PARE  $\leftarrow$  false;
6       $s' \leftarrow s$ ;       {Cópia da solução atual, melhor solução até então}
7      repita
8           $s'' \leftarrow \text{MelhorVizinho}(s, k)$ ;
9           $\text{AltereVizinhança}(s, s'', k)$ ;
10     até  $k = r$ 
11     se (  $f(s) \geq f(s')$  )
12         então PARE  $\leftarrow$  true;
13     fim-se;
14 até PARE = true;
15 Retorne  $s'$ ;
fim VND;
```

Esta função (linha 9 no pseudo-código do slide anterior) pode ser implementada de diversas formas

1. *Sequential neighborhood change*

- Esquema descrito anteriormente
- Se houver melhora em uma vizinhança, retorna-se a primeira delas
- Caso contrário, utiliza a vizinhança seguinte

2. *Pipe Neighborhood change*

- Se houver melhora em uma vizinhança, não realiza a troca
- Caso contrário, utiliza a vizinhança seguinte

3. *Cyclic neighborhood change*

- Utiliza a vizinhança seguinte independente do resultado de melhora ou não

4. *Random neighborhood change*

- Variação das três anteriores
- Entretanto, não há uma ordem definida para troca
- Trocas realizadas de forma aleatória

O algoritmo VNS foi proposto junto do VND

- Mesmas referências

A principal diferença do VND para o VND é o *shake*

- Mudança aleatória da solução corrente
- Similar a um operador de mutação em algoritmos evolutivos

```
1  Seja  $s_0$  uma solução inicial,  $r$  o número de estruturas de
   vizinhança;
2   $s \leftarrow s_0$ ;    {Solução atual}
3  enquanto (Critério de parada não satisfeito) faça
4       $k \leftarrow 1$ ; {Vizinhança corrente}
5      enquanto ( $k \leq r$ ) faça
6           $s' \leftarrow \text{Shake}(s, k)$ ;
7           $s'' \leftarrow \text{BuscaLocal}(s')$ ;
8           $\text{Altere\_vizinhança}(s, s'', k)$ 
9      fim-enquanto;
10 fim-enquanto;
11 Retorne  $s$ ;
fim VNS;
```



# VNS - VARIAÇÕES

## Reduced VNS (RVNS)

- Não existe busca local (linha 7 do slide anterior)
- Escolhe um vizinho de forma aleatória na vizinhança corrente
  - Caso exista melhora, move-se para este vizinho e continua o processo; Caso contrário, troca-se de vizinhança
- Critério de parada idêntico ao do VNS

## Skewed VNS (SVNS)

- Aceita soluções de pior custo durante o processo de busca local
- O objetivo é explorar o espaço de buscas de forma mais ampla

## Smart VNS

- Aumenta o tamanho do *shake* iterativamente
- Inicialmente, favorece a intensificação
- Após, favorece a exploração do espaço de buscas
- Caso haja melhora, o tamanho do *shake* é novamente reduzido

# ITERATED LOCAL SEARCH

O algoritmo ILS foi proposto na tese de doutorado de Thomas Stützle

- Thomas Stützle. Local Search Algorithms for Combinatorial Problems Analysis, Improvements, and New Applications. PhD thesis, Darmstadt University of Technology, Department of Computer Science, Darmstadt, Germany, 1998.
- Uma boa e mais sucinta referência pode ser encontrada em

▶ [Link](#)

Baseado em perturbações (*shake*) e buscas locais

- Uma única estrutura de vizinhança
- Perturbações progressivamente mais "fortes"

```
s0 ← SolucaoInicial();  
s ← BuscaLocal(s0);  
iter ← 0; MelhorIter ← Iter; nivel ← 1;  
enquanto ( iter - melhorIter < ILSmax ) faça  
    iter ← iter + 1;  
    s' ← perturbacao(s, nivel);  
    s'' ← BuscaLocal(s');  
    se ( f(s'') < f(s) ) então  
        s ← s'';  
        melhorIter ← iter;  
        nivel ← 1;  
    senão  
        nivel ← nivel + 1;  
    fim-se  
fim-enquanto
```

```

procedimento perturbação(s, nível)
    s' ← s;
    nmodificacoes ← nível + 1;
    cont ← 1;
    enquanto ( cont ≤ nmodificacoes ) faça
        Aplique movimento aleatório em s';
        cont ← cont + 1;
    fim-enquanto
retorne s'
    
```

É possível combinar as metaheurísticas acima descritas

No geral, altera-se a fase de busca local das heurísticas pelo algoritmo VND

Assim, temos algoritmos como o

- ILS-VND
- GRAPS-VND
- VNS-VND

Diversas outras alterações são possíveis!