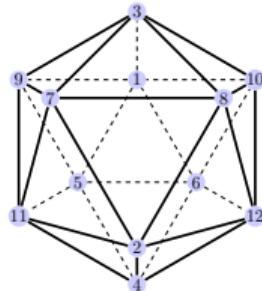


8 - Iterated Local Search (ILS) & Variable Neighborhood Search (VNS)

Alexandre Checoli Choueiri

16/04/2023



- ① Iterated Local Search (ILS) - A idéia geral
- ② ILS No *fitness landscape*
- ③ ILS - Pseudocódigo e design
- ④ Variable Neighborhood Search (VNS) - A ideia geral
- ⑤ VND No *fitness landscape*
- ⑥ VND - Pseudocódigo
- ⑦ Atividades

Iterated Local Search (ILS) - A idéia geral

ILS - A idéia geral

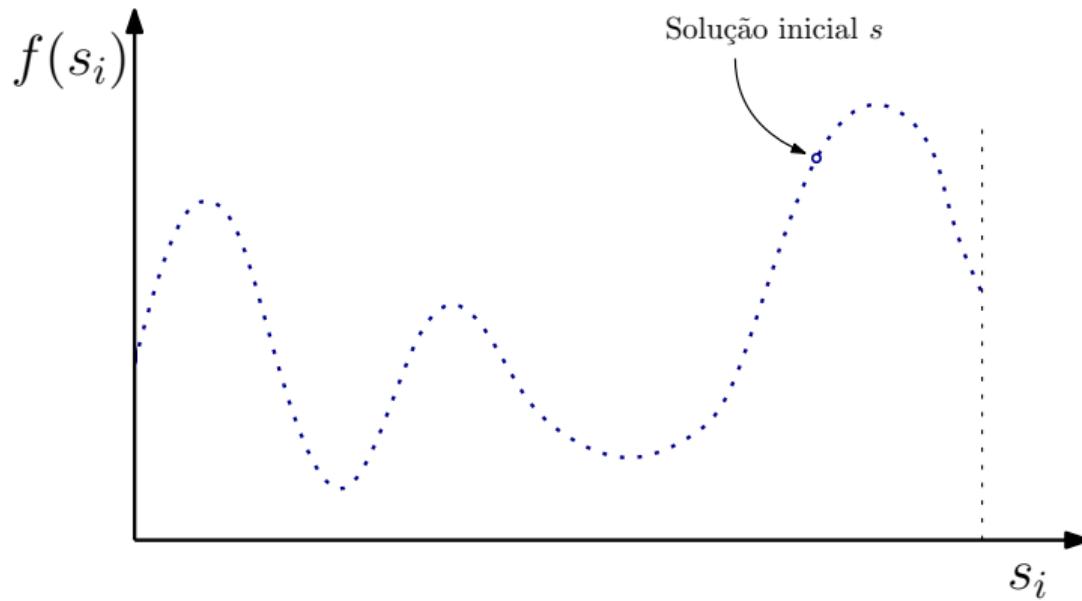
A qualidade do **ótimo local** obtido pela busca local é muito dependente da escolha da solução inicial, como já havíamos mencionado. A idéia do ILS (*Iterated Local Search* - busca local iterativa) é gerar **diversas soluções** iniciais com alta variabilidade, e aplicar um **componente de intensificação** (como a busca local) em cada uma delas.

ILS - A idéia geral

Na prática, **o ILS não gera soluções aleatórias iniciais**, mas sim aplica um componente de **perturbação** no ótimo local e reaplica a **intensificação**. Dessa forma busca-se manter alguma parte do ótimo local encontrado, porém "jogá-lo" para uma outra região do *fitness-landscape*.

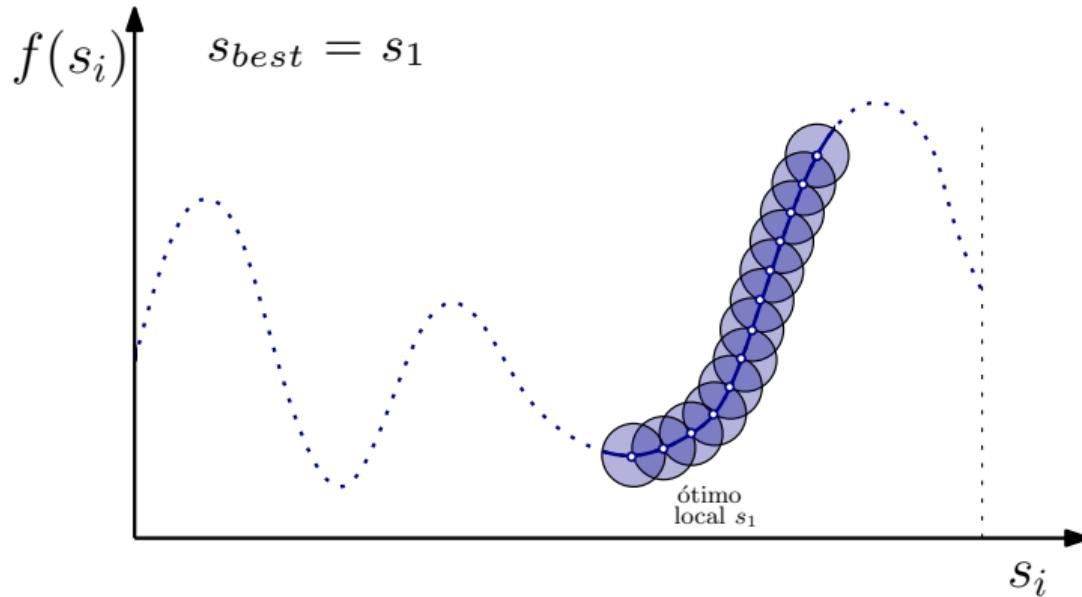
ILS No *fitness landscape*

No fitness landscape



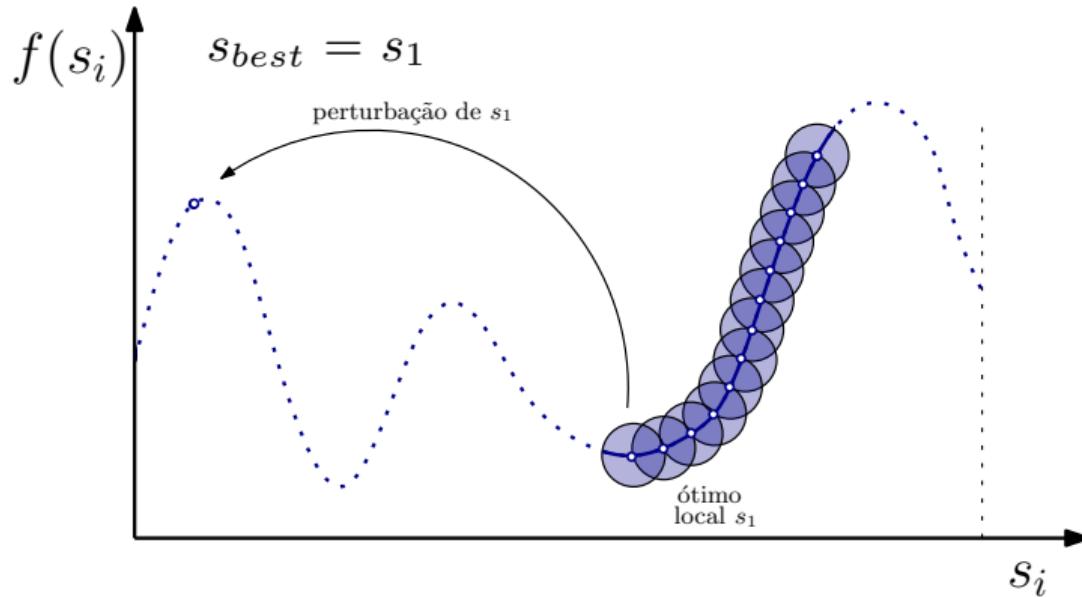
Inicialmente geramos uma solução inicial.

No fitness landscape



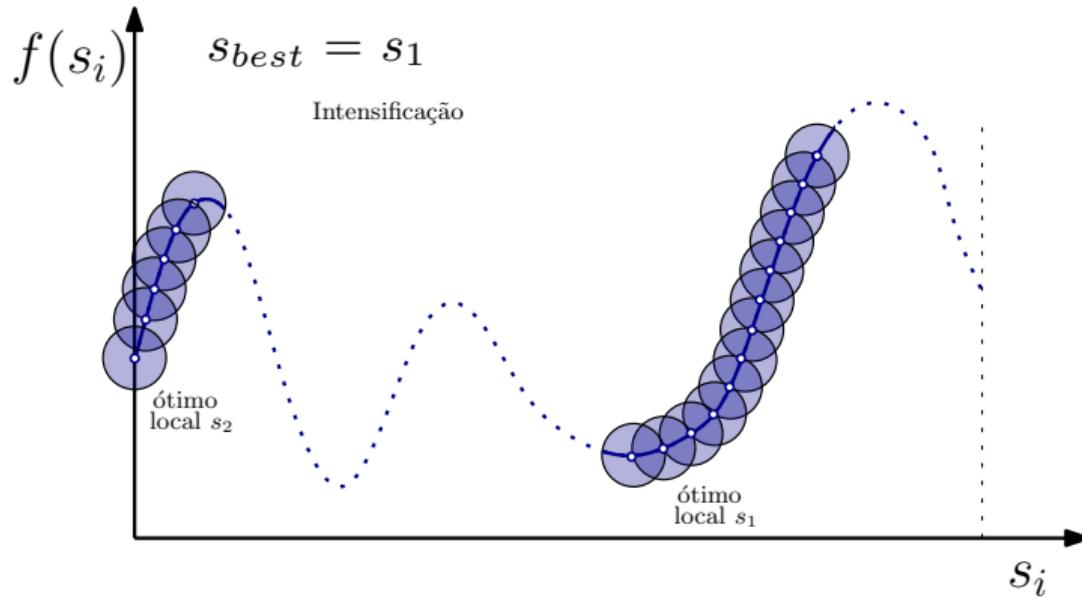
E aplicamos um componente de intensificação (como a **busca local**).

No *fitness landscape*



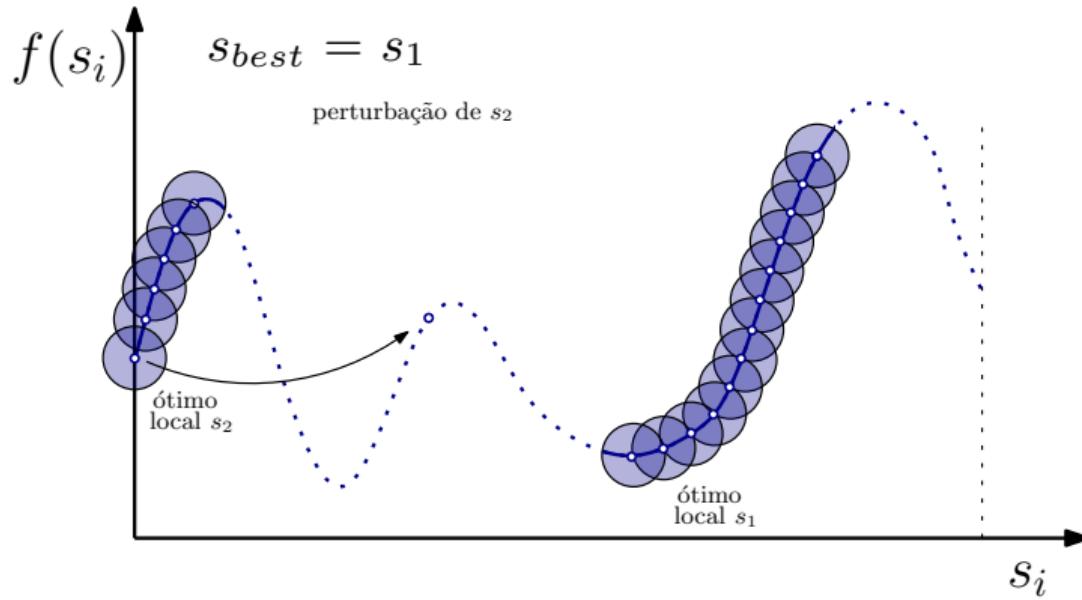
Em seguida perturbamos o ótimo local, mandando-o a outra região do *fitness landscape*.

No fitness landscape



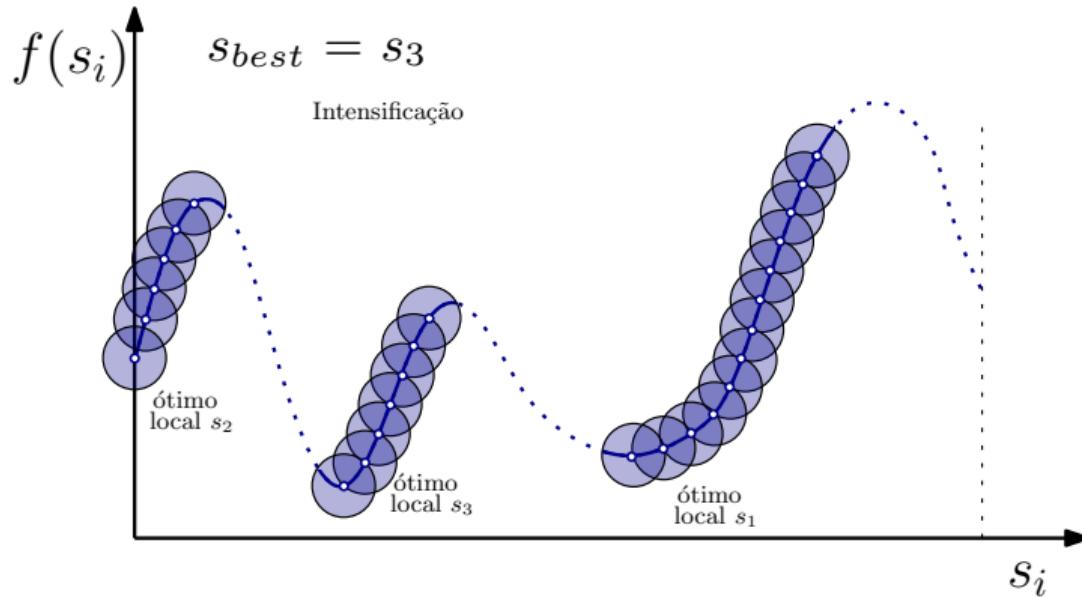
E o componente de intensificação é novamente aplicado.

No fitness landscape



O processo é repetido por um **número determinado** de iterações.

No fitness landscape



E sempre a melhor solução até o momento deve ser armazenada.

ILS - Pseudocódigo e design

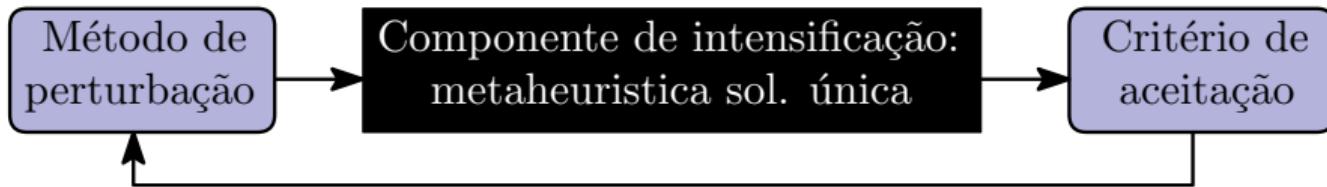
Pseudocódigo

Abaixo segue um pseudocódigo template do ILS.

Algorithm 1 Template genérico ILS

```
s0 = GeraSolInicial()                                ▷ Gera uma solução inicial
s* = Intensificacao(s0)                          ▷ Busca local
repeat
    s' = PerturbaSol(s*)
    s'_* = Intensificacao(s')
    s_* = CriterioDeAceite(s*,s'_*)                ▷ Aceita s'_* ou não
until Criterio de parada
return melhor s.                                     ▷ Melhor s deve ser armazenada
```

Componentes de *design*



De forma geral o ILS possui 3 componentes de *design*:

1. **O método de perturbação:** é o componente de exploração do algoritmo.
2. **Intensificação:** é o componente de intensificação, pode ser uma caixa preta, ou seja, qualquer metaheuristica de solução única pode ser usada.
3. **Aceitação:** critério que decide se a nova solução intensificada será usada na próxima iteração.

Componentes de *design*

A premissa do ILS é que a componente de **perturbação** deve ser mais eficiente do que um **reinicio aleatório**, onde uma solução independente aleatória seria gerada em toda iteração. Perturbando **uma parte** da solução garante que estaremos explorando o espaço de busca, mas em uma região não muito afastada do ótimo local.

Vale ressaltar que essa abordagem depende do *fitness landscape*, pode ser que um algoritmo com reinicio aleatório seja mais eficiente. Dessa forma, um primeiro exercício no design de um ILS é **executar uma perturbação enviesada e comparar com o reinicio totalmente aleatório**.

OBSERVAÇÃO

Podemos usar as próprias vizinhanças para o componente de perturbação. Por exemplo, para o TSP podemos perturbar uma solução com diversos movimentos SWAPS aleatórios. Uma boa decisão de design é usar uma vizinhança para a perturbação **diferente daquela usada na componente de intensificação**. Dessa forma a probabilidade de **ciclagem** das soluções é reduzida.

Variable Neighborhood Search (VNS) - A ideia geral

VNS - A ideia geral

Como no ILS, o VNS (variable neighborhood search - busca em vizinhança variável) tenta escapar de ótimos locais, porém parte das seguintes premissas:

1. A escolha de uma vizinhança \mathcal{N}_1 define um *fitness landscape* \mathcal{F}_1 .
2. Todo *fitness landscape* possui um ótimo local.
3. O ótimo global é um ótimo local, referente a alguma vizinhança \mathcal{N}_i .

VNS - A ideia geral

Dessa forma, a ideia do **VNS** é explorar sucessivamente um conjunto pré-definido de vizinhanças. A busca é feita de forma **aleatória** ou **sistemática** no conjunto de vizinhanças (dando origem a variantes), obtendo diferentes ótimos locais. A forma mais comum de VNS é chamada de **VND** (*variable neighborhood descent*).

O VND é uma variante determinística do VNS. O VND usa sucessivas vizinhanças de forma descendente em buscas locais.

VND - A ideia geral

1. Inicialmente é necessário definir um conjunto $\mathcal{N}_l (l = 1, \dots, l_{max})$ de vizinhanças. Seja \mathcal{N}_1 a primeira vizinhança a ser usada, e x uma solução inicial.

VND - A ideia geral

1. Inicialmente é necessário definir um conjunto $\mathcal{N}_I (I = 1, \dots, I_{max})$ de vizinhanças. Seja \mathcal{N}_1 a primeira vizinhança a ser usada, e x uma solução inicial.
2. Se uma melhoria na solução x na vizinhança atual $\mathcal{N}_I(x)$ não é possível, a estrutura de vizinhança é alterada de \mathcal{N}_I para \mathcal{N}_{I+1} .

VND - A ideia geral

1. Inicialmente é necessário definir um conjunto $\mathcal{N}_I (I = 1, \dots, I_{max})$ de vizinhanças. Seja \mathcal{N}_1 a primeira vizinhança a ser usada, e x uma solução inicial.
2. Se uma melhoria na solução x na vizinhança atual $\mathcal{N}_I(x)$ não é possível, a estrutura de vizinhança é alterada de \mathcal{N}_I para \mathcal{N}_{I+1} .
3. Se uma melhoria na solução atual x é encontrada, retorna-se a estrutura de vizinhança $\mathcal{N}_1(x)$, e a busca é reiniciada.

VND - A ideia geral

1. Inicialmente é necessário definir um conjunto $\mathcal{N}_I (I = 1, \dots, I_{max})$ de vizinhanças. Seja \mathcal{N}_1 a primeira vizinhança a ser usada, e x uma solução inicial.
2. Se uma melhoria na solução x na vizinhança atual $\mathcal{N}_I(x)$ não é possível, a estrutura de vizinhança é alterada de \mathcal{N}_I para \mathcal{N}_{I+1} .
3. Se uma melhoria na solução atual x é encontrada, retorna-se a estrutura de vizinhança $\mathcal{N}_1(x)$, e a busca é reiniciada.
4. O algoritmo termina quando nenhuma vizinhança \mathcal{N}_I melhorar a solução atual.

VND - A ideia geral

1. Inicialmente é necessário definir um conjunto $\mathcal{N}_I (I = 1, \dots, I_{max})$ de vizinhanças. Seja \mathcal{N}_1 a primeira vizinhança a ser usada, e x uma solução inicial.
2. Se uma melhoria na solução x na vizinhança atual $\mathcal{N}_I(x)$ não é possível, a estrutura de vizinhança é alterada de \mathcal{N}_I para \mathcal{N}_{I+1} .
3. Se uma melhoria na solução atual x é encontrada, retorna-se a estrutura de vizinhança $\mathcal{N}_1(x)$, e a busca é reiniciada.
4. O algoritmo termina quando nenhuma vizinhança \mathcal{N}_I melhorar a solução atual.

OBS: A estratégia do VND vai ser **mais efetiva** se as diferentes estruturas de vizinhanças usadas forem **complementares** entre si, no sentido de que o ótimo local para uma vizinhança \mathcal{N}_i não será um ótimo local para a vizinhança \mathcal{N}_j .

VND - A ideia geral

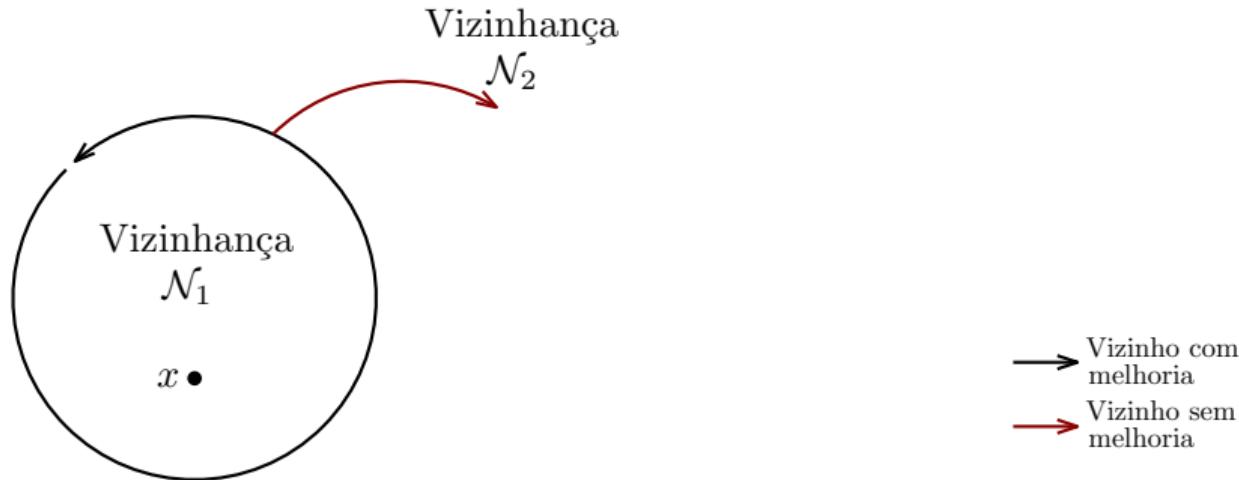
Podemos visualizar esse processo mais facilmente por meio de um fluxograma.

VND - Fluxograma



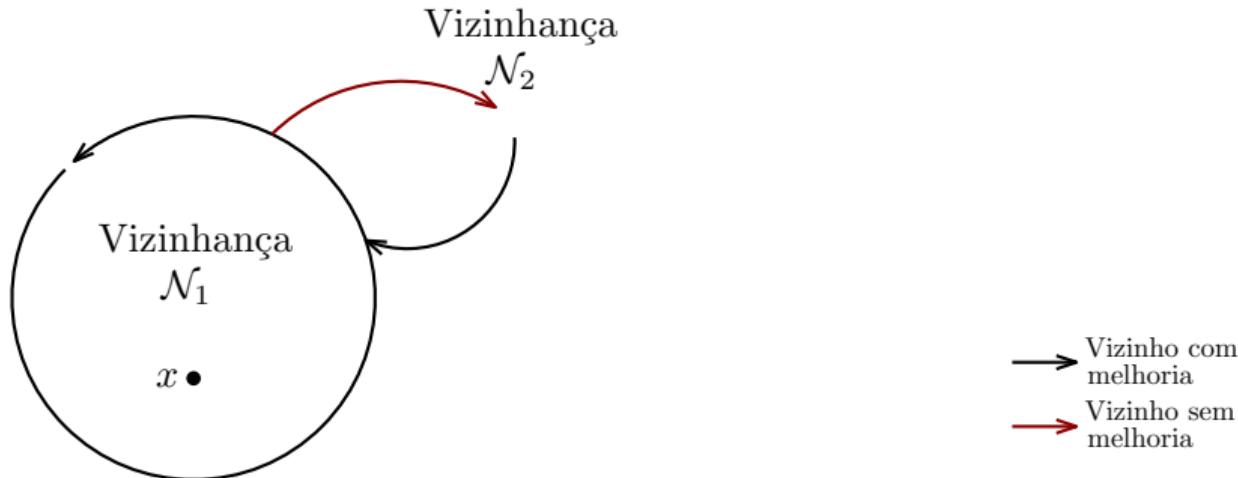
A busca local é aplicada usando a vizinhança \mathcal{N}_1

VND - Dinâmica



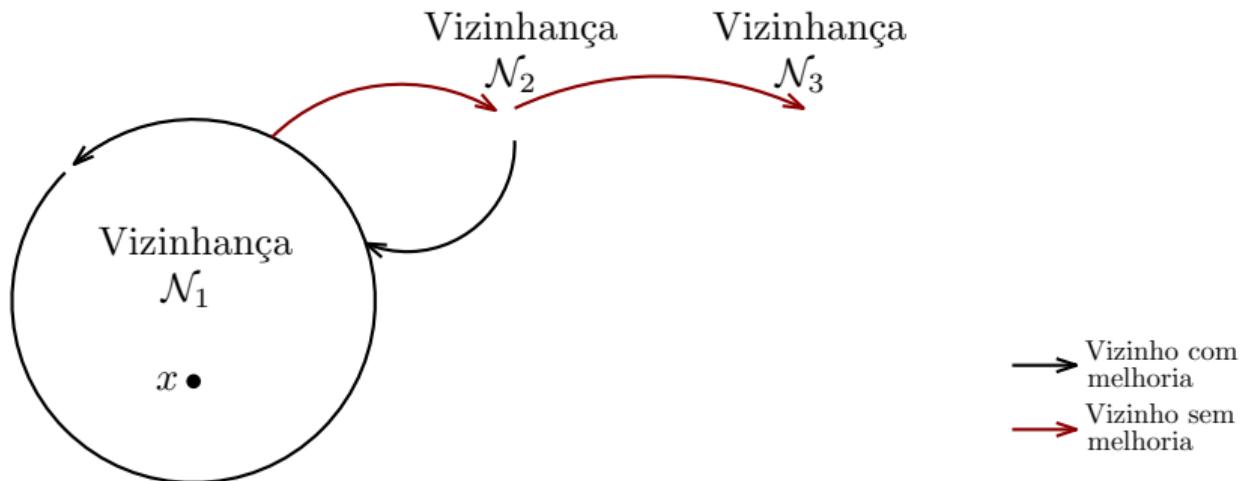
Quando a solução não pode mais ser melhorada (ótimo local), troca-se para \mathcal{N}_2 .

VND - Dinâmica



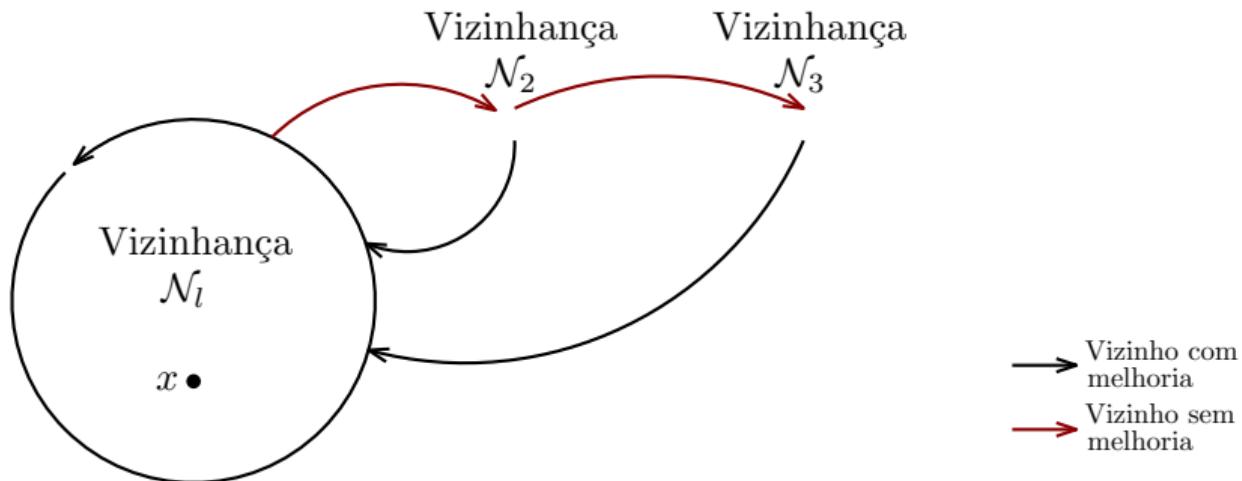
Novamente, uma busca local é executada em \mathcal{N}_2 . Se a solução foi melhorada, volta-se para \mathcal{N}_1 .

VND - Dinâmica



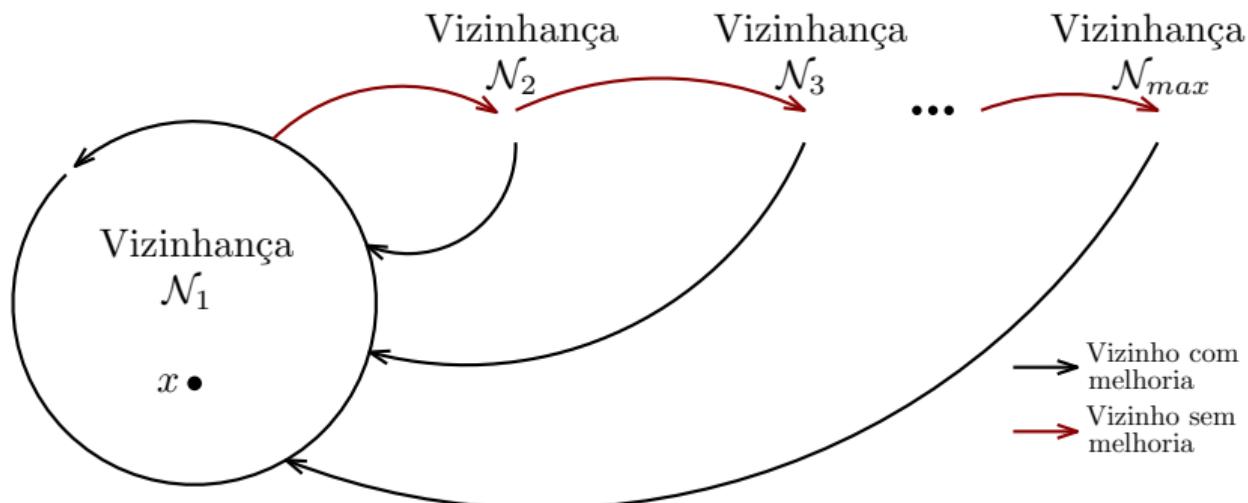
Quando tanto \mathcal{N}_1 quanto \mathcal{N}_2 não melhorarem x , ativa-se \mathcal{N}_3 .

VND - Dinâmica



A busca segue dessa forma, até o momento em que nenhuma vizinhança conseguir melhorar a solução x .

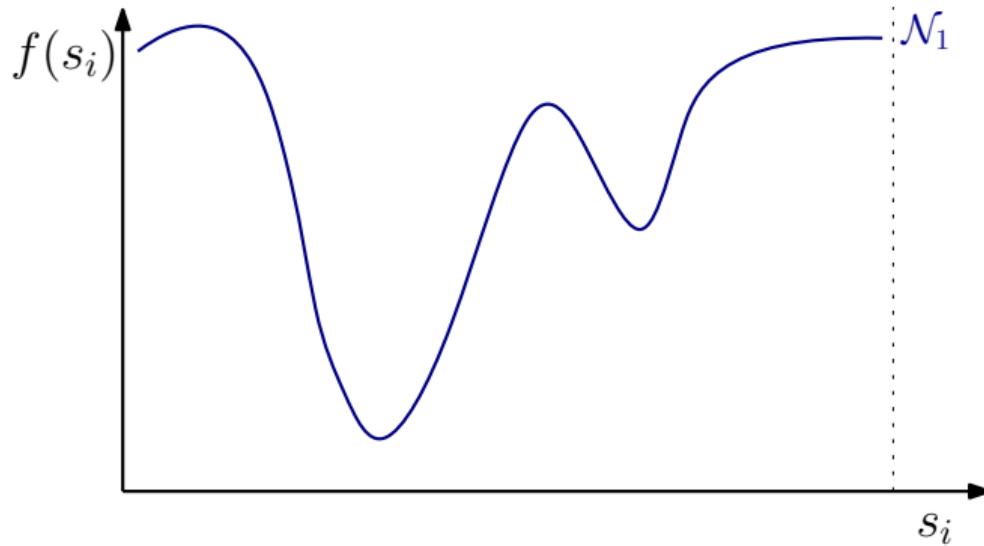
VND - Dinâmica



A busca segue dessa forma, até o momento em que nenhuma vizinhança conseguir melhorar a solução x .

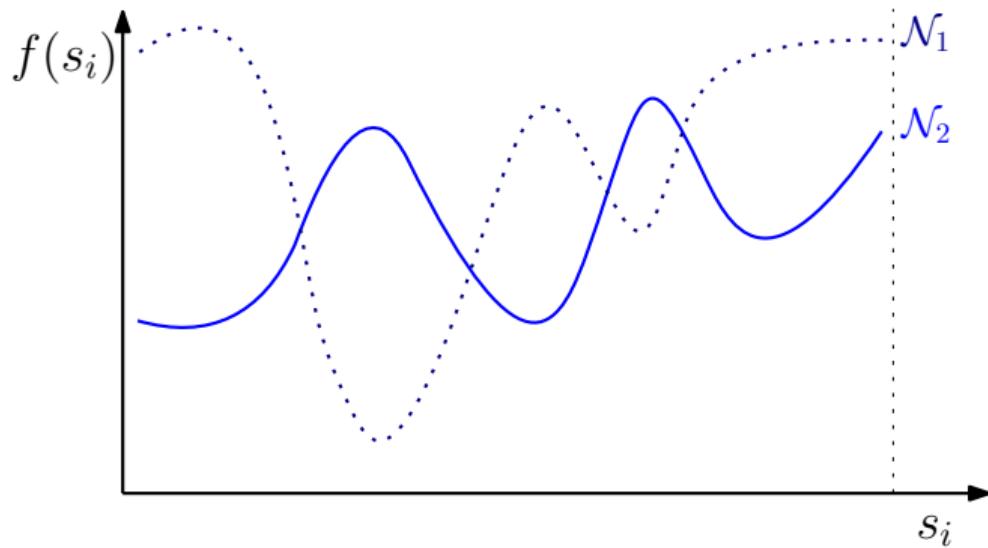
VND No *fitness landscape*

VND - Fitness landscape



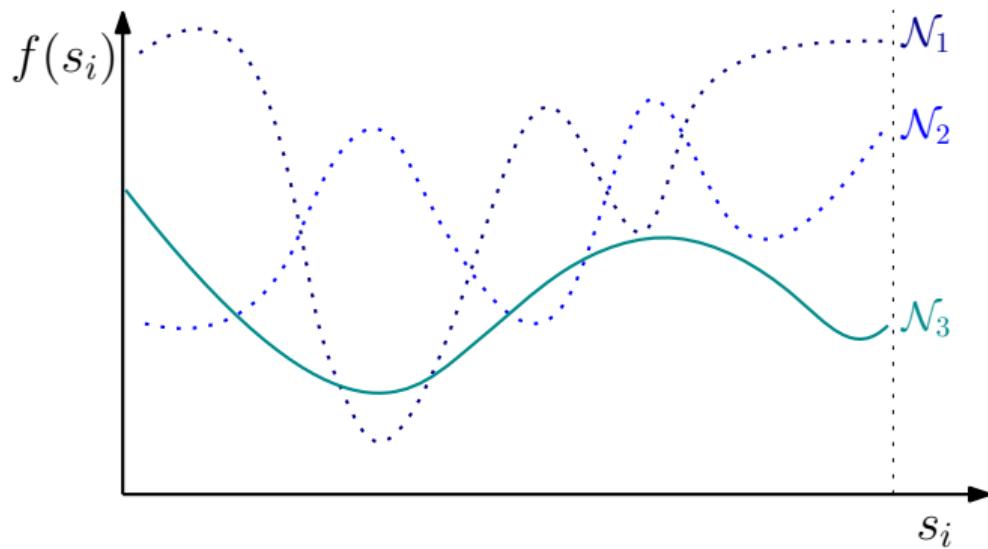
Seja a vizinhança \mathcal{N}_1 , \mathcal{N}_2 e \mathcal{N}_3 .

VND - Fitness landscape



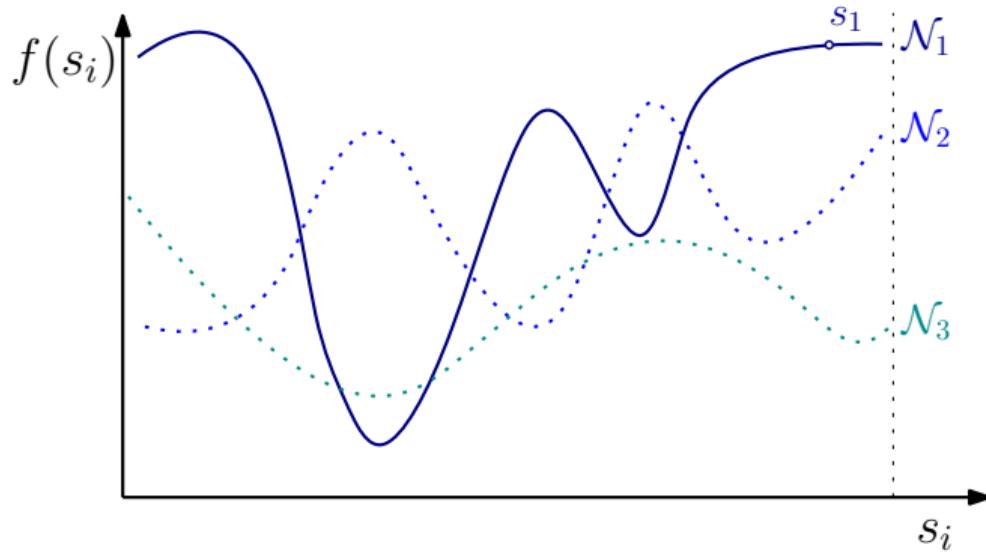
Seja a vizinhança \mathcal{N}_1 , \mathcal{N}_2 e \mathcal{N}_3 .

VND - Fitness landscape



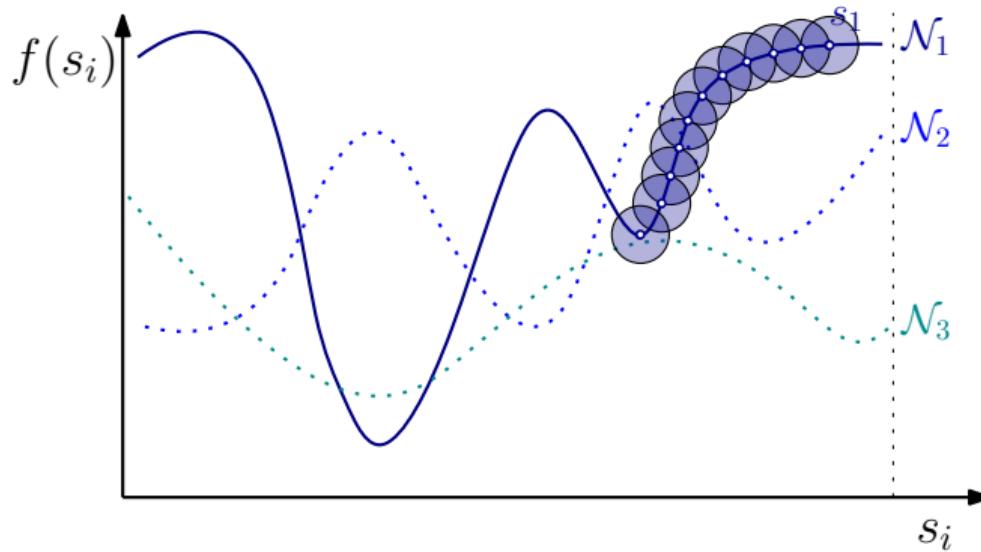
Seja a vizinhança \mathcal{N}_1 , \mathcal{N}_2 e \mathcal{N}_3 .

VND - Fitness landscape



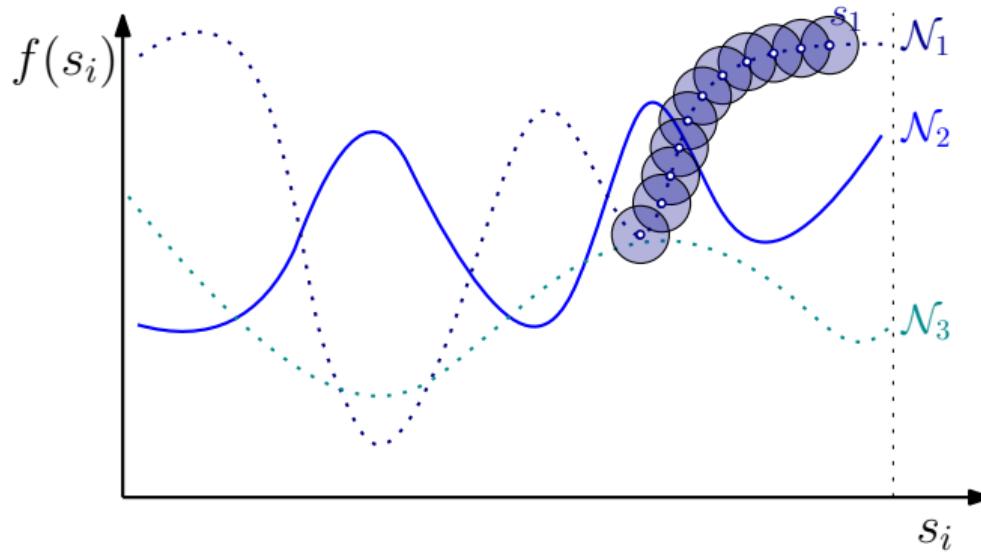
Considerando a solução inicial s_1 .

VND - Fitness landscape



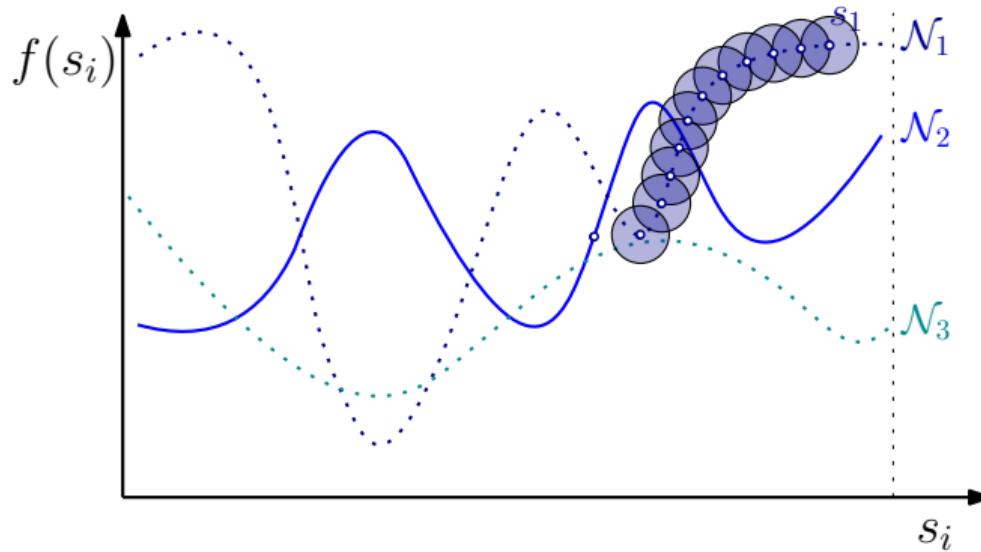
Uma busca local é executada em s_1 com a vizinhança \mathcal{N}_1 .

VND - Fitness landscape



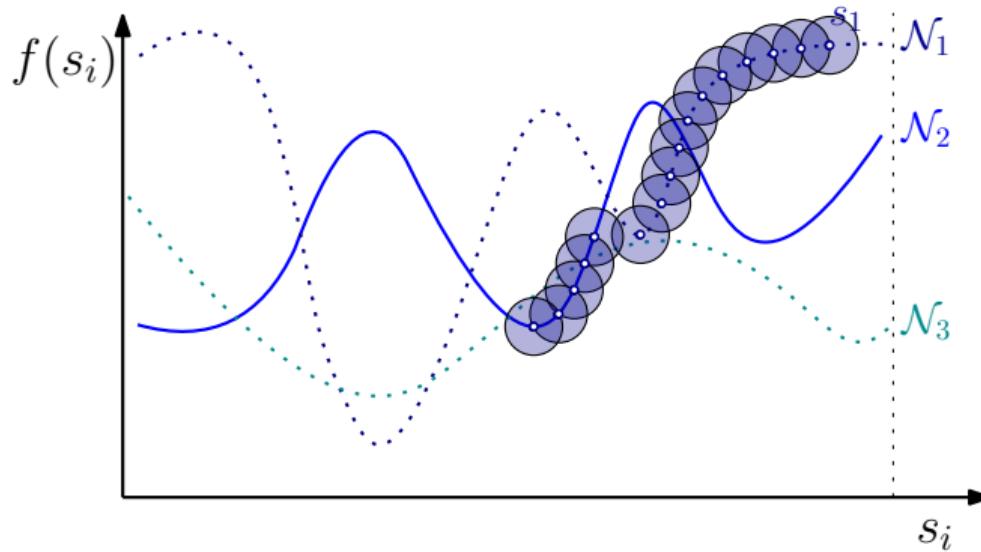
Quando o ótimo local é encontrado, ativamos a próxima vizinhança \mathcal{N}_2 .

VND - Fitness landscape



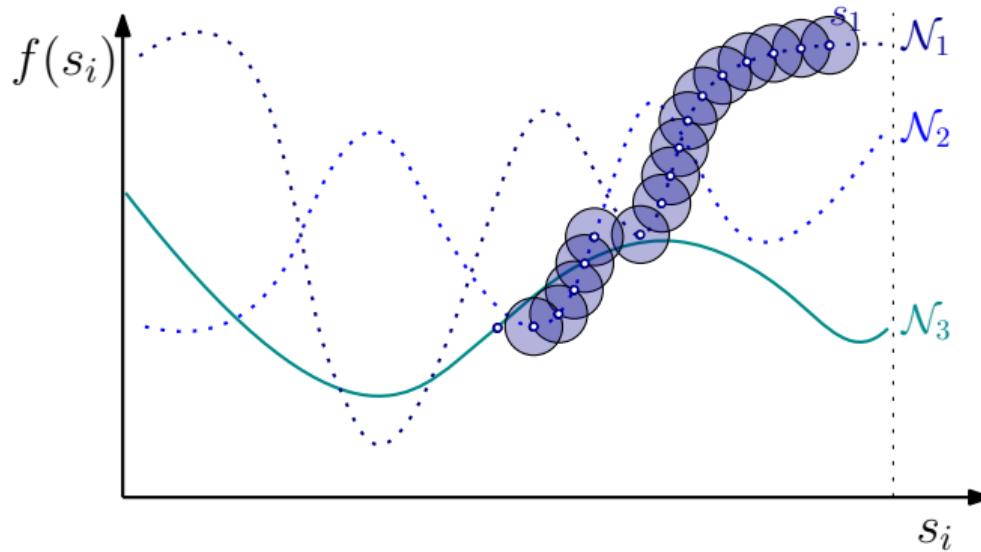
Novamente, uma busca local em \mathcal{N}_2 é realizada.

VND - Fitness landscape



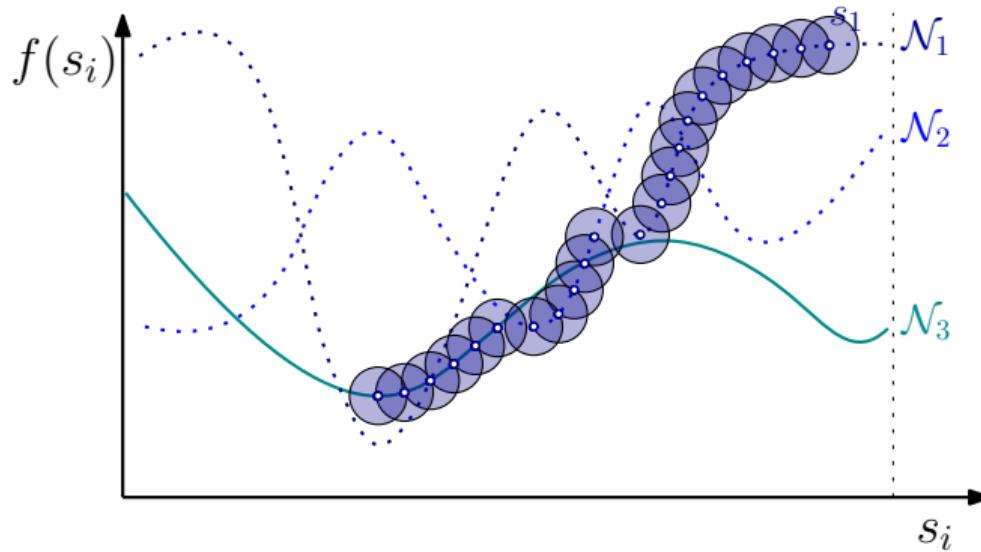
Até o momento em que o ótimo local em \mathcal{N}_2 for encontrado.

VND - Fitness landscape



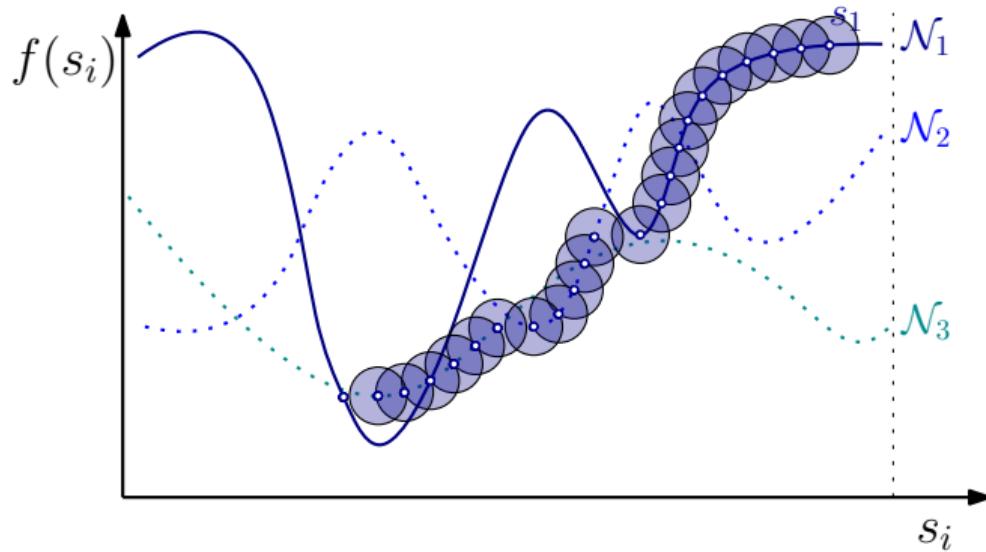
Quando o ótimo local for encontrado, a vizinhança \mathcal{N}_3 é ativada.

VND - Fitness landscape



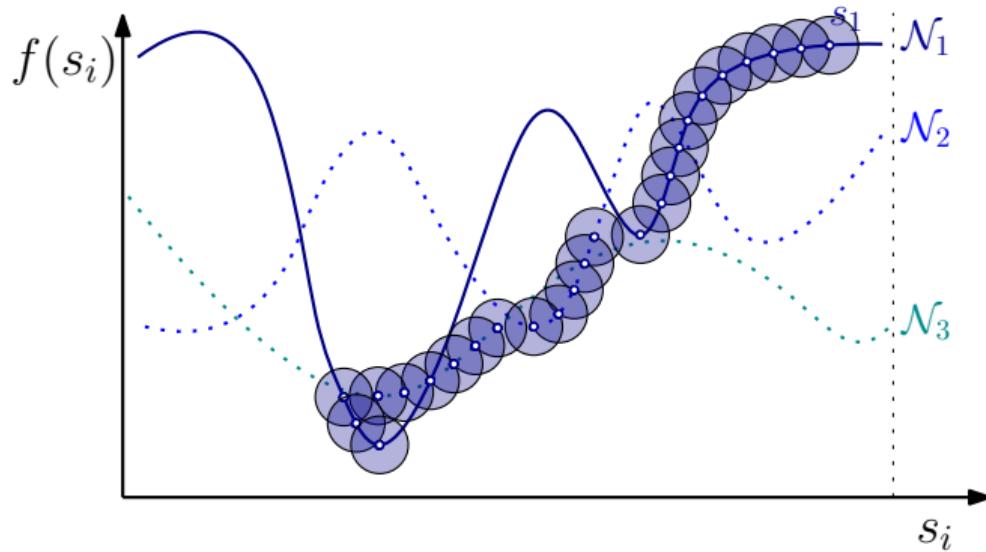
E uma busca local em \mathcal{N}_3 é executada.

VND - Fitness landscape



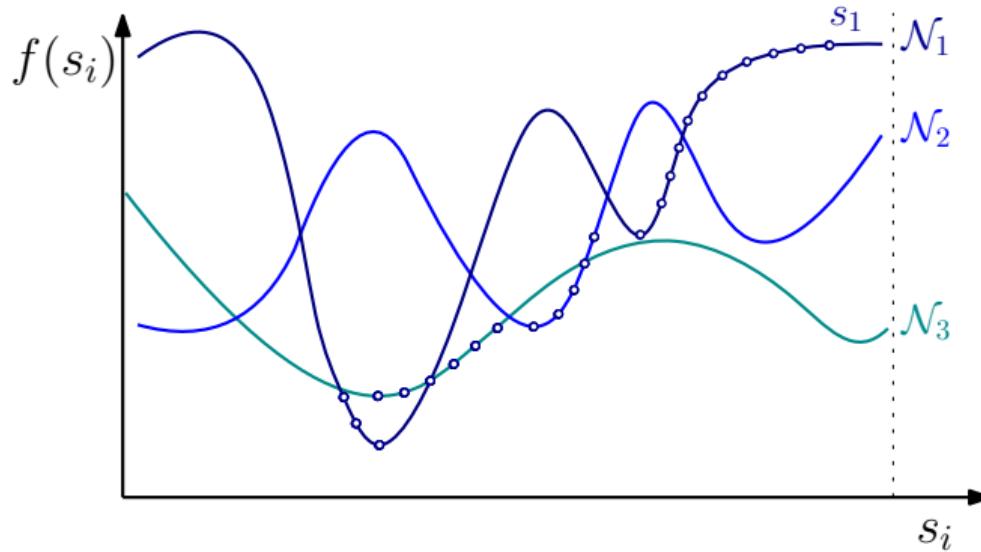
Após buscar em \mathcal{N}_3 , ativamos novamente \mathcal{N}_1 .

VND - Fitness landscape



E a busca é realizada até o novo ótimo local de \mathcal{N}_1 .

VND - Fitness landscape



O caminho todo percorrido em todas as vizinhanças é mostrado acima.

VND - Pseudocódigo

Pseudocódigo

Abaixo segue um pseudocódigo template do VND.

Algorithm 2 Template genérico VND

```
s0 = GeraSoluçãoInicial()                                ▷ Gera uma solução inicial
I = 1
while I < Imax do
    s' = BuscaLocal( NI, s0)
    if f(s') < f(s0) then
        s0 = s'
        I = 1
    else
        I = I + 1
    end if
end while
return melhor s.                                     ▷ Melhor s deve ser armazenada
```

Atividades

Exemplo - TSP

EXEMPLO: Implemente um algoritmo ILS para a resolução do TSP com as seguintes características:

1. Intensificação: busca local com 2-opt
2. Perturbação: SWAP e reinício aleatório (2 algoritmos diferentes)

EXEMPLO: Implemente um algoritmo VND para a resolução do TSP com as seguintes estruturas de vizinhanças: 2-OPT, SWAP e 1-SHIFT (leia no artigo) [1993 - Further results on the probabilistic traveling salesman problem- Bertsimas](#)

Use as instâncias do TSPlib para verificar a sua eficácia. Lembre de rodar diversos testes para calibrar os parâmetros do algoritmo.

Atividade I

- 1 Qual o principal componente do **ILS** que o permite sanar a deficiência da busca local?
- 2 Quais são os parâmetros do **ILS** que devem ser calibrados?
- 3 Qual o principal componente do **VND** que o permite sanar a deficiência da busca local?
- 4 Considerando o artigo [2000 - A study of exponential neighborhoods for the Traveling Salesman Problem - Daineko, Woeginger](#), entenda como funciona a vizinhança **ASSIGN**. Qual é a relação da vizinhança com o problema da designação? Faça um esboço de como isso funciona.
- 5 Considerando os pseudocódigos do ILS e do VND. Pense em um design de algoritmo que **combine características** dos dois algoritmos.

Atividade I

Algorithm 3 Template genérico VND

```
 $s_0 = \text{GeraSolIncial}()$  ;  $l = 1$      $\triangleright$  Gera uma solução inicial
while  $l < l_{max}$  do
     $s' = \text{BuscaLocal}(\mathcal{N}_l, s_0)$ 
    if  $f(s') < f(s_0)$  then
         $s_0 = s'$  ;
         $l = 1$ 
    else
         $l = l + 1$ 
    end if
end while
return melhor  $s$ .       $\triangleright$  Melhor  $s$  deve ser armazenada
```

Algorithm 4 Template genérico ILS

```
 $s_0 = \text{GeraSolIncial}()$   $\triangleright$  Gera uma solução inicial
 $s_* = \text{Intensificacao}(s_0)$        $\triangleright$  Busca local
repeat
     $s' = \text{PerturbaSol}(s_*)$ 
     $s'_* = \text{Intensificacao}(s')$ 
     $s_* = \text{CriterioDeAceite}(s_*, s'_*)$   $\triangleright$  Aceita  $s'_*$  ou não
until Criterio de parada
return melhor  $s$ .       $\triangleright$  Melhor  $s$  deve ser armazenada
```

Atividade II

- 1 Considerando o seu problema escolhido para a disciplina. Encontre 2 artigos científicos que usam ILS ou VND (ou usam ILS/VND como uma parte de outro algoritmo) para resolver o problema.