Iterated Local Search

Helena R. Lorenço, Olivier Martinz and THOMAS STUTZLE

Idéias

- Metaheurística deve ser simples, eficiente e mais genérica possível.
- Problema específico deve ser incorporado à metaheurística. Portanto deve ser composta por:
  - 1. função geral e;
  - 2. parte que varia de acordo com o problema.
- A heurística incorporada estará em um módulo (“caixa-preta”).
- A Busca Local Iterada: Constrói iterativamente uma seqüência de soluções geradas por uma heurística incorporada;
- Guia para melhores soluções é melhor que se usasse testes aleatórios repetidos.
- Heurística incorporada: Busca local

Algoritmo - Estrutura Geral

- Seja C uma função de custo a ser minimizada.
- Soluções candidatas são rotuladas s, e um conjunto por S.
- Busca local: determinística e sem memória
- Usa estrutura de vizinhança. Assim, é possível mover-se de uma solução s para uma melhor ainda por um caminho inteligente.
- Então, como reduzir o custo sem abrir a “caixa-preta”?

Random Restart

- A maneira mais simples de melhorar o custo é repetir a Busca Local de outro ponto inicial.
- Cada s* gerada será independente.
- Perde a utilidade a medida que o espaço de busca aumenta.
- Em instâncias muito genéricas levam o custo a: ter uma média que é uma taxa fixa que excede o custo ótimo;

Iterated Local Search (ILS)

- Pressuposto:
  - Os ótimos locais de um problema de otimização podem ser gerados a partir de perturbações na solução ótima local corrente
- A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções e fraca o suficiente para evitar um reinício aleatório

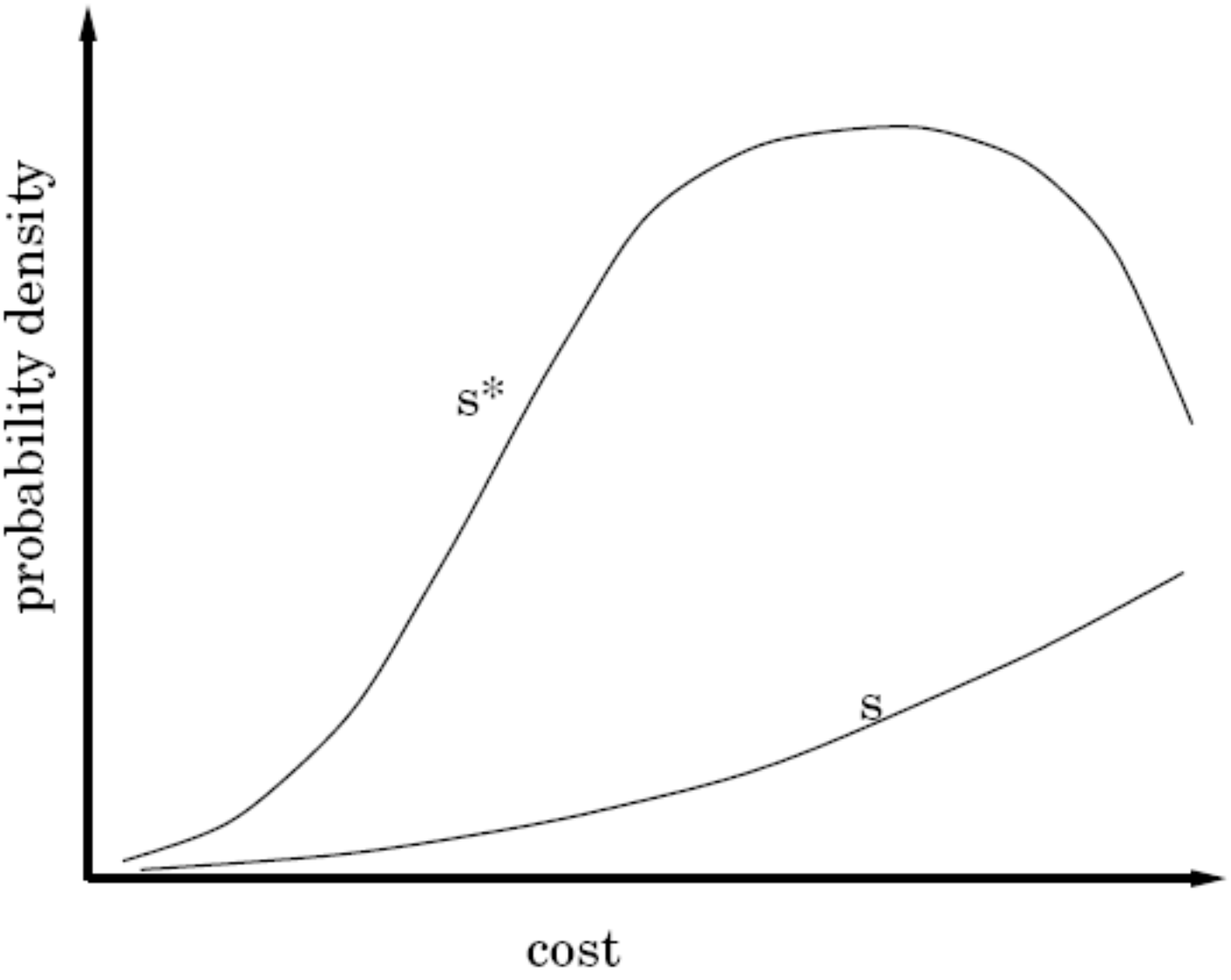


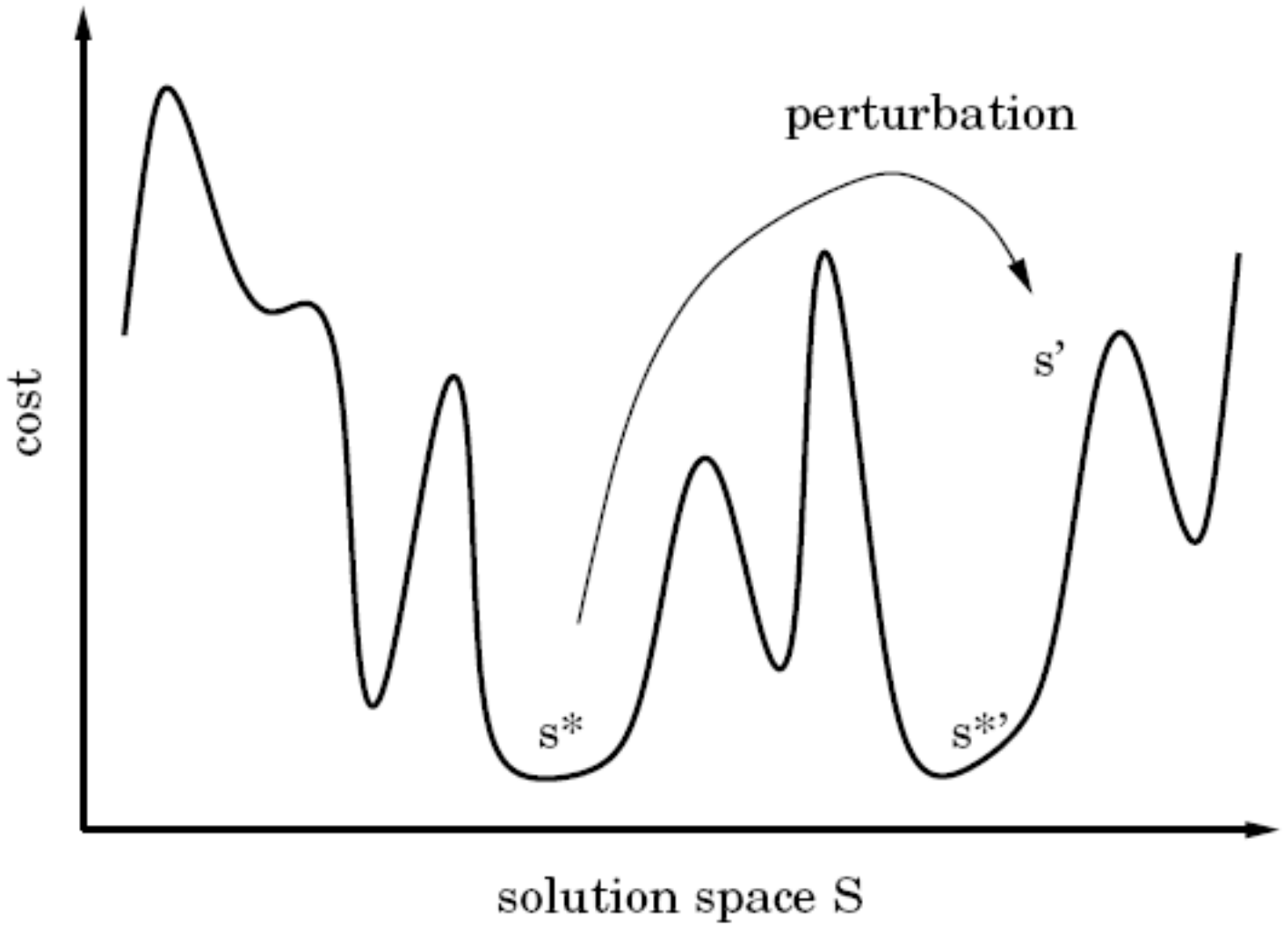
Figure 1: Probability densities of costs. The curve labeled s gives the cost density for all solutions, while the curve labeled s^* gives it for the solutions that are local optima.

Buscando em S*

- Evita grandes espaços de busca
- É feito de forma recursiva.
- Gera uma hierarquia de busca locais aninhadas
- Como formular a busca local no nível mais baixo da hierarquia?
- Busca local requer uma estrutura de vizinhança;
- O maior problema é como definir os vizinhos de ***S**** para que sejam numerados e acessados de forma eficiente.
- Aplicar busca estocástica em S*

Algoritmo

- Para explorar S^* sem a noção de vizinhança, aplica-se uma perturbação que leva s^* a um estado intermediário s',
- aplica-se a busca local em s', gerando s''^*, se este passar pelo teste de aceitação ele passar a ser o próximo elemento do caminho em S^*.
- Memória: a maioria ainda não utiliza, mas se precisar de s^* previamente:
  - Então podem ser usados: diversificação, intensificação, tabu,
  - perturbações adaptativas, critério de aceitação etc.
- O desempenho da ILS depende muito da perturbação e do critério de aceitação escolhido.


perturbation
cost
s'
s*
s*'
solution space S

Componentes do ILS:

- GeraSolucaoInicial:
- BuscaLocal:
  - Retorna uma solução melhorada
- Perturbacao:
  - Modifica a solução corrente guiando a uma solução intermediária
- CriterioAceitacao:
  - Decide de qual solução a próxima perturbação será aplicada

ILS

procedimento ILS

$s_0 \leftarrow$ *SolucaoInicial*

$s \leftarrow$ *BuscaLocal*(s_0)

iter $\leftarrow$ 0

enquanto (*iter* < $iter_{max}$)

iter $\leftarrow$ *iter* + 1

s' $\leftarrow$ *perturbação*(*s*, *histórico*)

s" $\leftarrow$ *BuscaLocal*(*s'*)

s $\leftarrow$ *CriterioAceitacao*(*s*, *s'*, *s''*)

fim-enquanto

retorne *s*

se (*f*(*s"*) < *f*(*s*)) **faç**

s $\leftarrow$ *s"*

fim-se

Algoritmo – Melhorando a performance

- **Solução Inicial**
- Começar com uma boa solução pode ser importante se soluções de alta qualidade estão sendo buscadas o mais rápido possível.
- Há dois tipo de inicialização: Aleatória ou Gulosa.
  - Vantagens da gulosa:
    - Combinada com a busca local resulta em soluções s0* de melhor qualidade
    - Uma busca local a partir de uma solução gulosa, requer menos tempo de CPU.

Perturbação

- A ILS aplica perturbações para sair dos ótimos locais.
- O Grau de perturbação: número de vezes que os componentes da solução forem modificados.
- Pode-se obter melhores soluções se as características do problema são considerados.
- Perturbação Alta – ILS comporta-se como random restart e há pouca possibilidade de se encontrar boas soluções.
- Perturbação Fraca – a busca local cai frequentemente em um ótimo local já visitado e a diversificação torna-se muito limitada.
- Em problemas simples como TSP, pode se obter resultados satisfatórios usando perturbação fixa (movimento double-bridge)

Iterated Local Search (ILS)

- Combina intensificação com diversificação
- Intensificação:
  - É obtida fazendo-se “pequenas” perturbações na solução ótima corrente
- Diversificação:
  - É obtida aceitando-se quaisquer soluções s” e aplicando-se “grandes” perturbações na solução ótima corrente

Perturbações adaptativas

- Não existe um melhor valor para o grau de perturbação
- Assim, se torna melhor que seja adaptado durante a execução
- Pode ser feito de duas formas:
  - 1. Guardando o histórico da busca
  - 2. Mudando deterministicamente o grau durante a busca, utilizando
- oscilações estratégicas.

Velocidade

- Empiricamente ILS é mais rápido em busca locais que random restart

instance	$\#LS_{RR}$	$\#LS_{1\text{-}DB}$	$\#LS_{5\text{-}DB}$
kroA100	17507	56186	34451
d198	7715	36849	16454
lin318	4271	25540	9430
pcb442	4394	40509	12880
rat783	1340	21937	4631
pr1002	910	17894	3345
pcb1173	712	18999	3229
d1291	835	23842	4312
fl1577	742	22438	3915
pr2392	216	15324	1777
pcb3038	121	13323	1232
fl3795	134	14478	1773
rl5915	34	8820	556

Critério de Aceitação

- A perturbação junto com a busca local definem as possíveis transações entre a solução atual s* e uma solução vizinha s*’;
- O critério de aceitação determina quando s*' é aceito ou não;
- Pode ser usado para controlar o balanço entre intensificação e diversificação da busca.

Critérios de aceitação

- Better: forte intensificação e só aceita as melhores soluções.

$$\texttt{Better}(s^*, s^{*\prime}, history) = \begin{cases} s^{*\prime} & \text{if } C(s^{*\prime}) < C(s^*) \\ s^* & \text{otherwise} \end{cases}$$

Critérios de aceitação

- RW: sem considerar o custo, aplica a perturbação ao ótimo local visitado mais recentemente;
- favorece a diversificação sobre a intensificação

$$\mathrm{RW}(s^*, s^{*\prime}, history) = s^{*\prime}$$

Critérios de aceitação

- Restart: quando a intensificação parece ineficiente o algoritmo deve ser reinicializado.
- Por exemplo, quando não se obtém melhoras em um determinado número de iterações.

$$\texttt{Restart}(s^*, s^{*\prime}, history) = \begin{cases} s^{*\prime} & \text{if } C(s^{*\prime}) < C(s^*) \\ s & \text{if } C(s^{*\prime}) \geq C(s^*) \text{ and } i - i_{last} > i_r \\ s^* & \text{otherwise.} \end{cases}$$

Conclusões

- **ILS** é
  - simples
  - fácil de implementar
  - robusto
  - altamente eficiente.
- **Idéia essencial:**
  - Focar a busca no pequeno subespaço definido pelas soluções que
  - são localmente ótimas.
  - O sucesso da ILS depende principalmente da escolha da busca
  - local, das perturbações e do critério de aceitação.

Conclusões

- Sugestões para aplicação da ILS:
  - problemas onde a maior parte das metaheurísticas falham;
  - problemas com multi objetivos;
  - problemas dinâmicos ou de tempo-real onde os dados variam durante a resolução do problema.
- Espera-se significantes melhoras com o uso de memória, estratégias explícitas de intensificação e diversificação.

Variable Neighborhood Descent (VND) e Variable Neighborhood Search(VNS)

Variable Neighborhood Descent (VND)

- Proposto por Nenad Mladenovic & Pierre Hansen em 1997
- Método de Descida em Vizinhança Variável
- Explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança
- Explora vizinhanças gradativamente mais “distantes”
- Sempre que há melhora em uma certa vizinhança, retorna-se à vizinhança “menos distante”

Variable Neighborhood Descent

- Princípios básicos:
  - Um ótimo local com relação a uma vizinhança não necessariamente corresponde a um ótimo com relação a outra vizinhança
  - Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança
  - Para muitos problemas, ótimos locais com relação a uma vizinhança são relativamente próximos

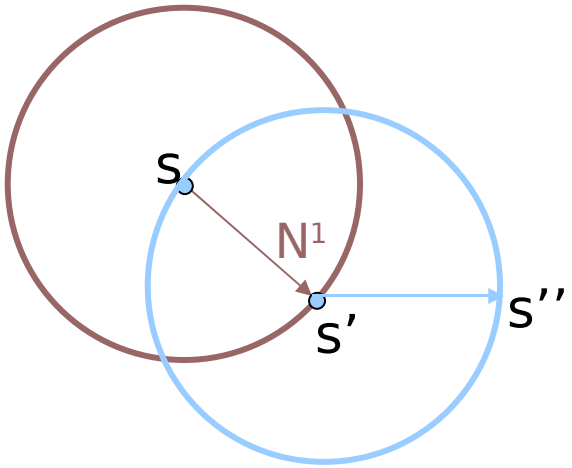
Procedimento VND

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 Encontre o melhor vizinho s' $\in N^{(k)}(s)$;
6 se ($f(s') < f(s)$)
7 então $s \leftarrow s'$; $k \leftarrow 1$;
8 senão $k \leftarrow k + 1$;
9 fim-se;
10 fim-enquanto;
11 Retorne s;
fim VND;

Variable Neighborhood Search

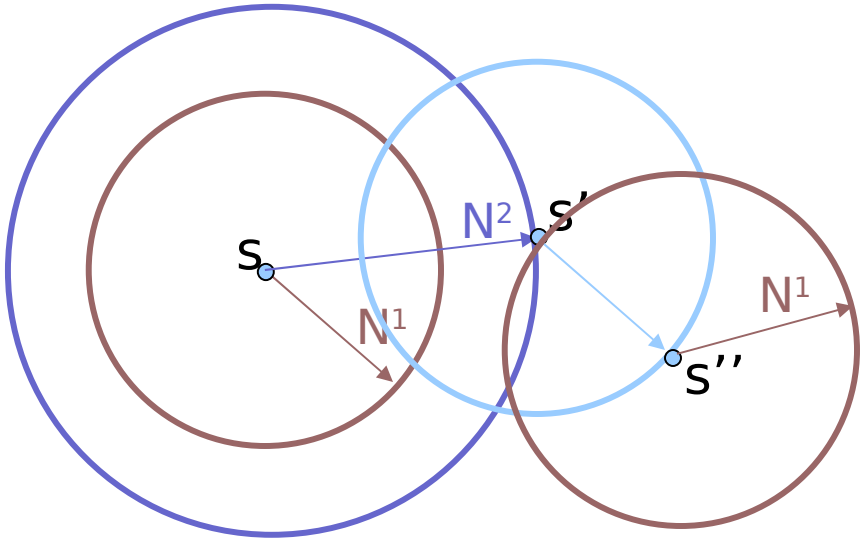
- Proposto por Nenad Mladenovic & Pierre Hansen em 1997
- Método de busca local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança
- Explora vizinhanças gradativamente mais “distantes”
- Focaliza a busca em torno de uma nova solução somente se um movimento de melhora é realizado

Variable Neighborhood Search (VNS)

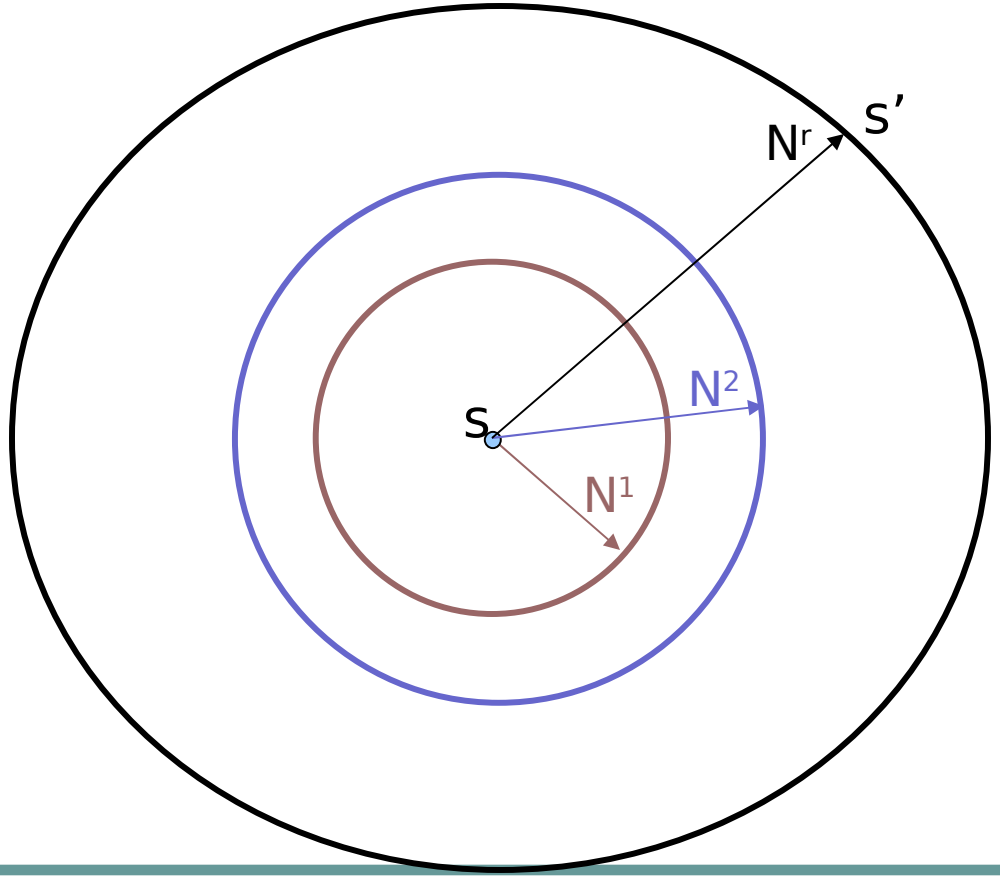


s'' aceito se f(s'') < f(s)

Variable Neighborhood Search (VNS)



Variable Neighborhood Search (VNS)



Procedimento VNS

```
1  Seja s0 uma solução inicial e r o número de estruturas de
   vizinhança;
2  s ← s0;                          {Solução corrente}
3  enquanto (Critério de parada não satisfeito) faça
4        k ← 1;                     {Tipo de estrutura de vizinhança}
5        enquanto (k ≤ r) faça
6              Gere um vizinho qualquer s' ∈ N(k)(s);
7              s'' ← BuscaLocal(s');
8              se ( f(s'') < f(s) )
9                    então s ← s''; k ← 1;
10                   senão k ← k + 1;
11             fim-se;
12       fim-enquanto;
13 fim-enquanto;
14 Retorne s;
fim VNS;
```