Variable Neighborhood Search (VNS)

Marcone Jamilson Freitas Souza[1,2,3]

Puca Huachi Vaz Penna[1]

[1] Departamento de Computação

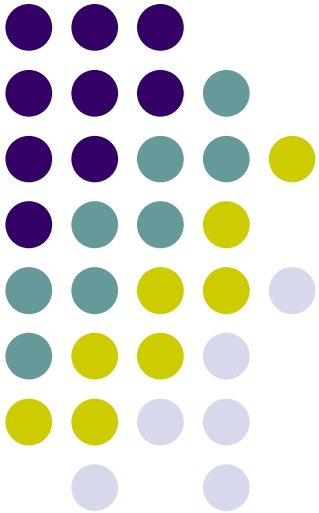
[1] Programa de Pós-Graduação em Ciência da Computação

Universidade Federal de Ouro Preto

[2] Programa de Pós-graduação em Modelagem Matemática e Computacional / CEFET-MG

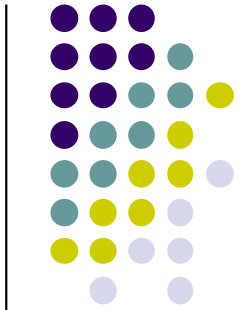
[3] Programa de Pós-graduação em Instrumentação, Controle e Automação de Processos de Mineração / ITV/UFOP

www.decom.ufop.br/prof/marcone, www.decom.ufop.br/puca
E-mail: {marcone,puca}@ufop.edu.br

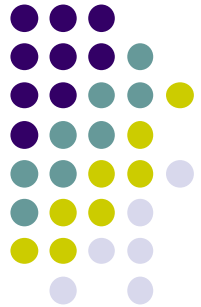


Souza, Marcone J. F. e Penna, Puca H. V. Variable Neighborhood Search. Notas de aula de Técnicas Metaheurísticas para Otimização Combinatória. Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, 2021. Disponível em http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/VNS.pptx

Variable Neighborhood Descent (VND)

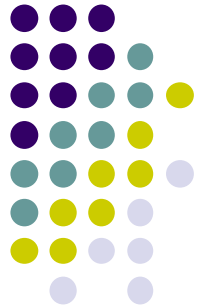


- Proposto por Nenad Mladenovic & Pierre Hansen em 1997
- Método de Descida em Vizinhança Variável
- Explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança
- Explora vizinhanças gradativamente mais “distantes”
- No VND básico, proposto em 1997, sempre que há melhora em uma certa vizinhança, retorna-se à vizinhança “menos distante”
- Posteriormente foram propostas outras variantes que alteram a forma de decidir qual será a próxima vizinhança explorada



Variable Neighborhood Descent (VND)

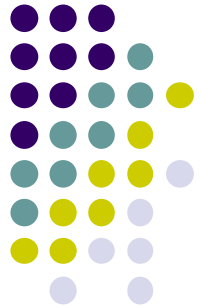
- Princípios básicos:
  - Um ótimo local com relação a uma vizinhança não necessariamente corresponde a um ótimo com relação a outra vizinhança
  - Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança
  - Para muitos problemas, ótimos locais com relação a uma vizinhança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



Procedimento VND básico

```
1  Seja s0 uma solução inicial e r o número de estruturas
   de vizinhança;
2  s ← s0;                          {Solução corrente}
3  k ← 1;                           {Tipo de estrutura de vizinhança}
4  enquanto (k ≤ r) faça
5        Encontre o melhor vizinho s' ∈ N(k)(s);
6        se ( f(s') < f(s) )
7              então s ← s'; k ← 1;
8              senão k ← k + 1;
9        fim-se;
10 fim-enquanto;
11 Retorne s;
fim VND;
```

Obs.: A linha 5 envolve a análise de toda a vizinhança da solução *s* e pode ser dispendiosa. Uma alternativa é fazer uma análise mais simples da vizinhança, seguindo uma estratégia *First Improvement*. Ou seja, havendo um vizinho de melhora, mova para ele. Assim, a linha 5 pode ser substituída por: *s*' ← PrimeiraMelhora(*s*, *k*);

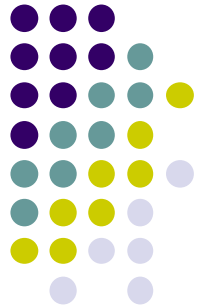


Framework genérico do VND

```
1  Seja s0 uma solução inicial e r o número de estruturas
   de vizinhança;
2  s ← s0;                          {Solução corrente}
3  k ← 1;                           {Tipo de estrutura de vizinhança}
4  enquanto (k ≤ r) faça
5        s' ← MelhorVizinho(s, k);
6        AltereVizinhança(s, s', k)
7  fim-enquanto;
8  Retorne s;
fim VND;
```

Obs.: A linha 5 pode ser substituída por: $s' \leftarrow$ PrimeiraMelhora(s, k);

Framework mais recente para o VND

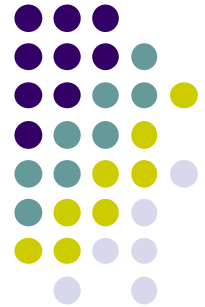


- Descrito em: Hansen, P.; Mladenovic, N.; Todosijevic, R. e Hanafi, S. Variable Neighborhood Search: basics and variants. Euro Journal of Computational Optimization, 5:423-454, 2017.
- O passo 9 do algoritmo abaixo pode ser Neighborhood_Change_Sequential, Neighborhood_Change_Pipe ou Neighborhood_Change_Cyclic

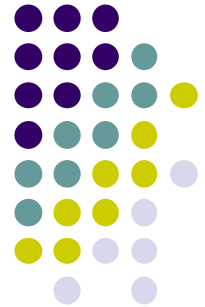
```
1    Seja s0 uma solução inicial e r o número de estruturas de vizinhança;
2    s ← s0;{Solução atual}
3    repita
4        k ← 1;              {Vizinhança corrente}
5        PARE ← false;
6        s' ← s;             {Cópia da solução atual, melhor solução até então}
7        repita
8            s'' ← MelhorVizinho(s, k);
9            AltereVizinhança(s, s'', k);
10       até k = r
11       se ( f(s) ≥ f(s') )
12          então PARE ← true;
13       fim-se;
14   até PARE = true;
15   Retorne s';
fim VND;
```

Obs.: A linha 8 pode ser substituída por: *s*'' ← PrimeiraMelhora(*s*, *k*);

Procedimentos de alteração de vizinhança



- *Sequential neighborhood change step*:
  - Se houver melhoramento em uma vizinhança, retorna-se à primeira delas; caso contrário, passa-se para a vizinhança seguinte
- *Pipe neighborhood change step*:
  - Se houver melhoramento em uma vizinhança, permanece-se nela; caso contrário, passa-se para a vizinhança seguinte
- *Cyclic neighborhood change step*:
  - Passa-se para a vizinhança seguinte, independentemente de haver ou não melhora na solução atual

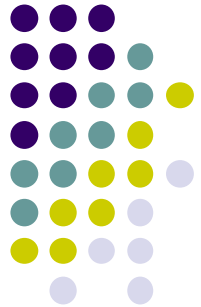


Sequential neighborhood change step

- Se houver melhoramento na solução atual, retorna-se à primeira vizinhança; caso contrário, passa-se para a vizinhança seguinte

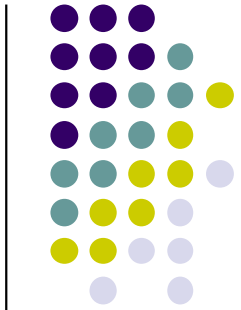
```
Neighborhood_Change_Sequential(s, s', k);
1   se ( f(s') < f(s) )
2         então
3             s ← s';
4             k ← 1;
5         senão
6             k ← k + 1;
7   fim-se;
8   Retorne s, k;
```

Pipe neighborhood change step



- Se houver melhoramento em uma vizinhança, permanece-se nela; caso contrário, passa-se para a vizinhança seguinte

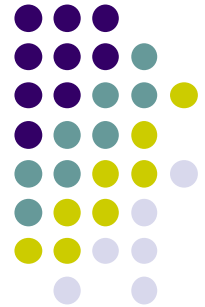
```
Neighborhood_Change_Pipe(s, s', k);
1   se ( f(s') < f(s) )
2         então
3             s ← s';
4         senão
5             k ← k + 1;
6   fim-se;
7   Retorne s, k;
```



Cyclic neighborhood change step

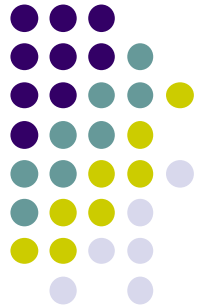
- Passa-se para a próxima vizinhança, independentemente de melhora na solução atual

```
Neighborhood_Change_Cyclic(s, s', k);
1    k ← k + 1;
2    se ( f(s') < f(s) )
3          então s ← s';
4    fim-se;
5    Retorne s, k;
```



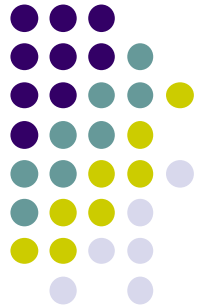
Variantes VND

- B-VND *(Basic VND)*:
  - Usa o procedimento *Sequential neighborhood change step* para decidir qual a próxima vizinhança a ser explorada
- P-VND (*Pipe* VND):
  - Usa o procedimento *Pipe neighborhood change step* para decidir qual a próxima vizinhança a ser explorada
- C-VND (*Cyclic VND*):
  - Usa o procedimento *Cyclic neighborhood change step* para decidir qual a próxima vizinhança a ser explorada
- U-VND (*Union* VND):
  - Consiste em uma busca local na união de TODAS as vizinhanças usadas para explorar o espaço de soluções do problema



Variantes VND

- RVND *(VND with Random neighborhood*):
  - Proposto simultaneamente em:
    - Souza, M. J. F.; Coelho, I. M.; Ribas, S.; Santos, H. G.; Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research, 207:1041-1051, 2010.
    - Subramanian, A.; Drummond, L. M. A.; Bentes, C.; Ochi, L. S.; FARIAS, R. A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. Computers & Operations Research, 37:1899-1911, 2010.
  - Nomeado RVND em Subramanian *et al.* (2010)



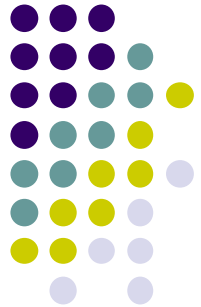
Variantes VND

- RVND *(VND with Random neighborhood)*:
  - Pressuposto para aplicação do RVND:
    - A melhor ordem de exploração das vizinhanças pode ser diferente por instância
  - Vantagens de aplicação do RVND:
    - Não é necessário calibrar a ordem das vizinhanças, como requerido no VND clássico
    - Assim como no VND, há garantia de que a solução retornada é um ótimo local com relação a todas as vizinhanças exploradas

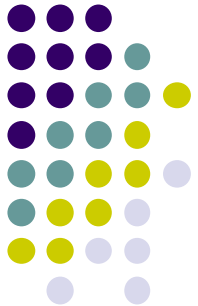
Variantes VND

- RVND *(VND with Random neighborhood)*:
  - Funcionamento: A **cada chamada** do método VND, as vizinhanças são ordenadas aleatoriamente
  - Segue os mesmos princípios do VND
  - Exemplo: Suponha que as vizinhanças de um dado problema são N = {N^1, N^2, N^3, N^4}
  - Ao aplicar o RVND, essas vizinhanças são colocadas em uma ordem aleatória, por exemplo: N^3, N^1, N^4, N^2
  - Neste caso, a primeira vizinhança a ser explorada será N^3; a segunda, N^1; a terceira, N^4 ; e a última será N^2

Variable Neighborhood Search (VNS)



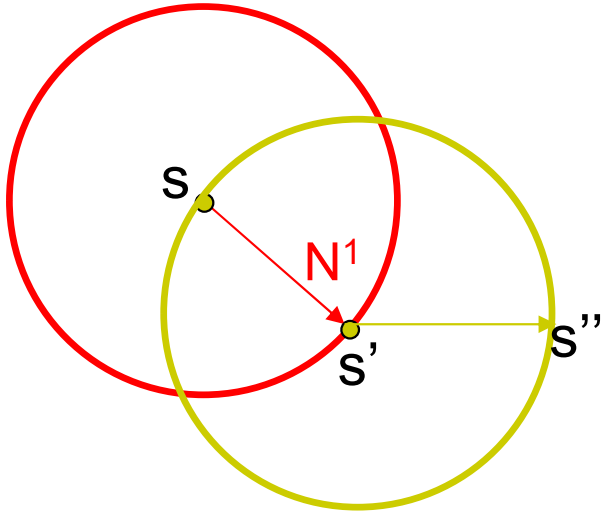
- Proposto por Nenad Mladenovic & Pierre Hansen em 1997
- Metaheurística de busca local que explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança
- Explora vizinhanças gradativamente mais “distantes”



Variable Neighborhood Search (VNS)

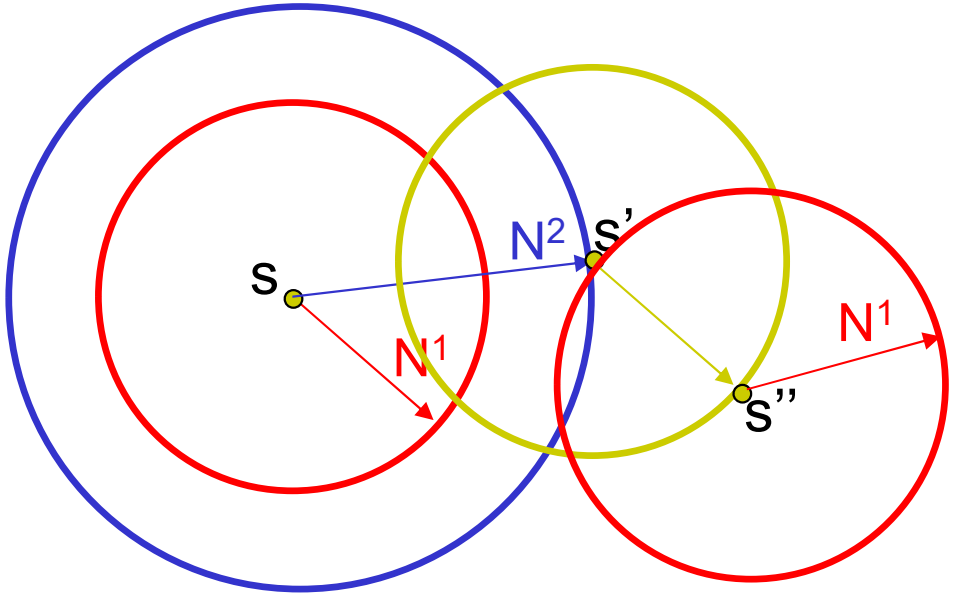
- Possui quatro componentes principais:
  - Gerador de solução inicial
  - Procedimento de perturbação (*shaking*):
    - gera uma perturbação na k-ésima vizinhança da solução atual
  - Procedimento de busca local, que pode ser o VND:
    - refina a solução atual
  - Procedimento de troca da vizinhança:
    - define qual a próxima vizinhança a ser explorada

Variable Neighborhood Search (VNS)

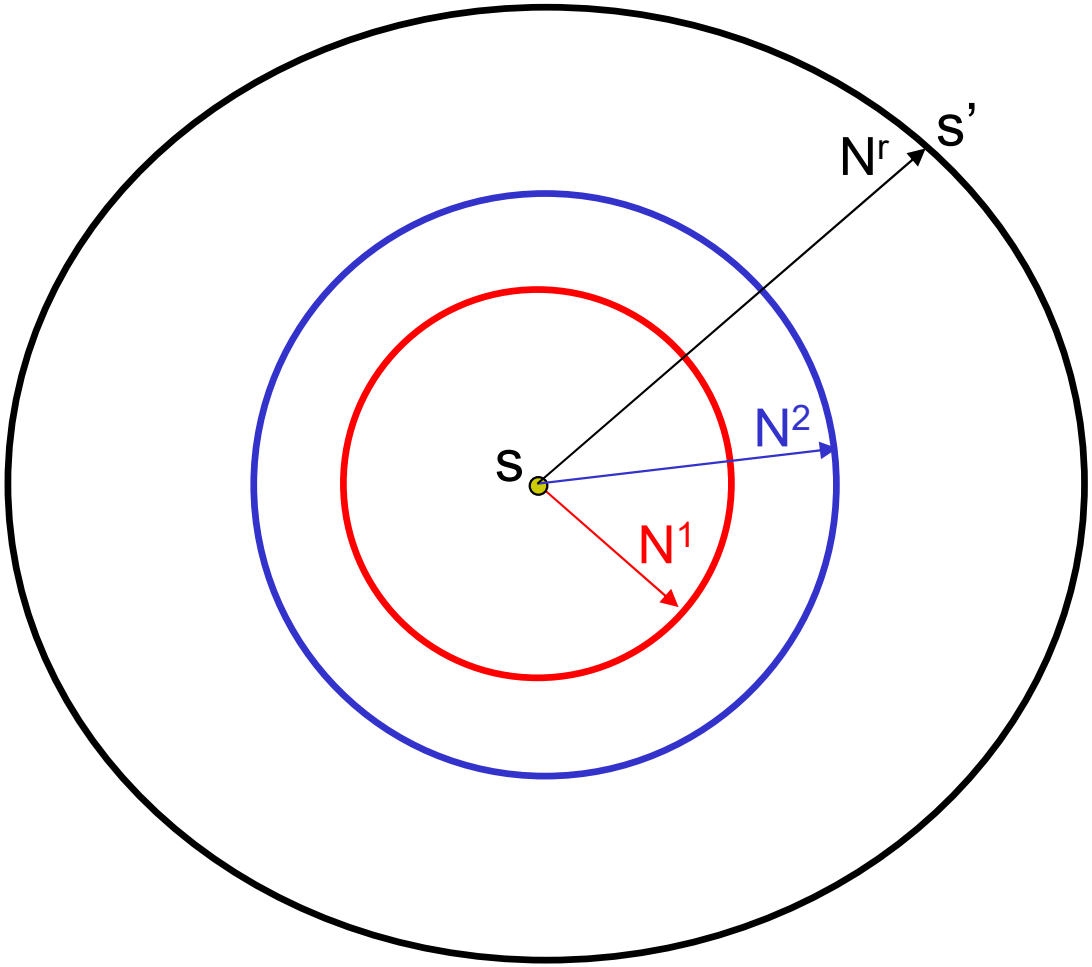


s'' aceito se f(s'') < f(s)

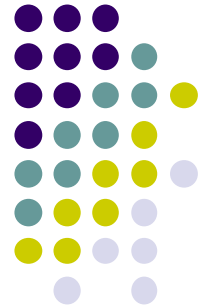
Variable Neighborhood Search (VNS)



Variable Neighborhood Search (VNS)

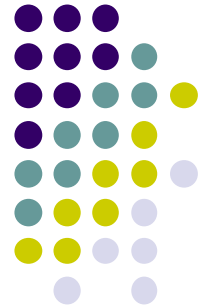


Procedimento VNS básico (Hansen e Mladenovic, 1997)



```
1   Seja s_0 uma solução inicial e r o número de estruturas de
    vizinhança;
2   s ← s_0;                        {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;
```

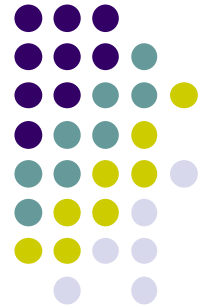
Framework genérico do VNS (Hansen et al., 2017)



```
1  Seja s0 uma solução inicial, r o número de estruturas de
   vizinhança;
2  s ← s0;        {Solução atual}
3  enquanto (Critério de parada não satisfeito) faça
4          k ← 1;  {Vizinhança corrente}
5          enquanto (k ≤ r) faça
6                  s' ← Shake(s, k);
7                  s'' ← BuscaLocal(s');
8                  Altere_vizinhança(s, s'', k)
9          fim-enquanto;
10 fim-enquanto;
11 Retorne s;
fim VNS;
```

Hansen, P.; Mladenovic, N.; Todosijevic, R. e Hanafi, S. Variable Neighborhood Search: basics and variants. Euro Journal of Computational Optimization, 5:423-454, 2017.

Algumas variantes do VNS



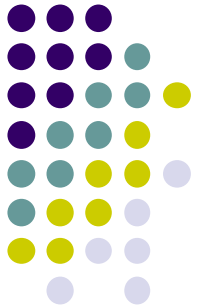
- *Basic Variable Neighborhood Search* (VNS)
  - Busca local do VNS feita por um método de busca local convencional.
- *General Variable Neighborhood Search* (GVNS)
  - Busca local do VNS feita pelo VND.
- *Reduced Variable Neighborhood Search* (RVNS)
  - Não há busca local.
  - Consiste em selecionar um ponto aleatório do espaço de busca da *k*-ésima vizinhança, iniciando da primeira. Se houver melhora, move-se para esse ponto, e retorna-se à primeira vizinhança; caso contrário, passa-se para próxima vizinhança e repete-se o processo. O RVNS termina quando se alcança a última vizinhança e o critério de parada for atingido.
- *Skewed Variable Neighborhood Search* (SVNS)
  - Aceita soluções de piora que distam da solução atual por um determinado valor.

Reduced VNS

```
1   Seja s_0 uma solução inicial e r o número de estruturas de
    vizinhança;
2   s ← s_0;                        {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               se ( f(s') < f(s) )
8                  então s ← s'; k ← 1;
9                  senão k ← k + 1;
10              fim-se;
11        fim-enquanto;
12  fim-enquanto;
13  Retorne s;
fim RVNS;
```

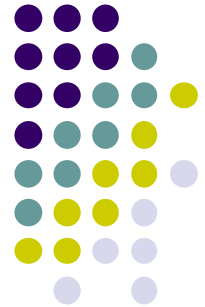
Critério de parada: Número máximo de iterações sem melhora, tempo de processamento, etc.

General Variable Neighborhood Search (GVNS)



```
1   Seja s0 uma solução inicial e r o número de estruturas de
    vizinhança;
2   s ← s0;                          {Solução atual}
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'' ← VND(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 GVNS;
```

Skewed Variable Neighborhood Search (SVNS)

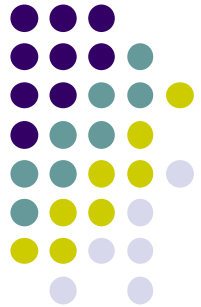


- Explora vales distantes da solução atual
- Uma vez que a melhor solução de uma região foi explorada é necessário explorar outras regiões do espaço de soluções para se conseguir melhorias
- No SVNS, essa exploração de outras regiões é feita aceitando-se soluções de piora que distam do último ótimo local por uma determinada distância controlada pelo algoritmo
- Difere das outras variantes VNS com relação ao procedimento de alteração da vizinhança (*NeighborhoodChangeS*)

Skewed VNS (SVNS)

1 Seja s_0 uma solução inicial e r o número de estruturas de vizinhança;
2 $s \leftarrow s_0$; {Solução corrente}
3 $s^* \leftarrow s$; {Melhor solução}
4 enquanto (Critério de parada não satisfeito) faça
5 $k \leftarrow 1$; {Tipo de estrutura de vizinhança}
6 enquanto ($k \leq r$) faça
7 Gere um vizinho qualquer s' $\in N^{(k)}(s)$;
8 $s'' \leftarrow$ BuscaLocal(s');
9 se ($f(s'') < f(s^*)$) então $s^* \leftarrow s''$;
10 Neighborhood_Change_Skewed(s, s'', k, α);
11 fim-enquanto;
12 fim-enquanto;
13 Retorne s;
fim SVNS;

Skewed VNS (SVNS)



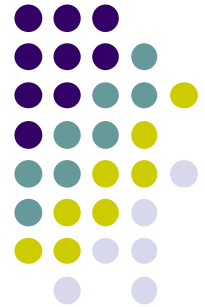
- Aplica o procedimento *Neighborhood_Change_Skewed* para alterar a vizinhança corrente
- Usa uma função ρ para medir a distância entre a solução atual s e o ótimo local encontrado s''
- O valor de α serve para definir o quanto dessa distância será permitida para aceitar uma solução de piora

```
Neighborhood_Change_Skewed(s, s'', k, α);
1   se ( f(s'') < f(s) + α.ρ(s, s'')  )
2         então s ← s''; k ← 1;
3         senão k ← k + 1;
4   fim-se;
5   Retorne s, k;
```

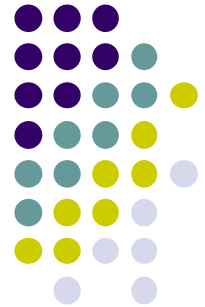
Skewed VNS (SVNS)

```
1   Seja s_0 uma solução inicial e r o número de estruturas de
    vizinhança;
2   s ← s_0;                          {Solução corrente}
3   s* ← s;                           {Melhor solução}
4   enquanto (Critério de parada não satisfeito) faça
5         k ← 1;                      {Tipo de estrutura de vizinhança}
6         enquanto (k ≤ r) faça
7               Gere um vizinho qualquer s' ∈ N^(k)(s);
8               s'' ← BuscaLocal(s');
9               se ( f(s'') < f(s*) ) então s* ← s'';
10              se ( f(s'') < f(s) + α.ρ(s, s'') )
11                 então s ← s''; k ← 1;
12                 senão k ← k + 1;
13              fim-se;
14        fim-enquanto;
15  fim-enquanto;
16  Retorne s*;
fim SVNS;
```

Skewed VNS (SVNS)



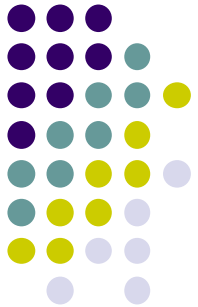
- Se o valor de $\rho(s, s'')$ for pequeno, o parâmetro α deve ser grande para possibilitar saltos maiores
- Um bom valor de α pode ser encontrado experimentalmente ou por um processo de aprendizagem
- Uma implementação trivial para este procedimento consiste em aceitar soluções piores que a solução atual em até α%, isto é, uma solução s'' é aceita se:
  - $f(s'') < (1 + \alpha)f(s)$,
  - sendo s a solução atual e s'' o ótimo local resultante da busca local



Smart Variable Neighborhood Search (*Smart* VNS)

- Origem no trabalho:
  - Souza, M. J. F.; Coelho, I. M.; Ribas, S.; Santos, H. G.; Merschmann, L. H. C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European Journal of Operational Research, 207:1041-1051, 2010.
- Segue as mesmas ideias do *Smart Iterated Local Search* (SILS)

Smart VNS



```
1   Seja s0 uma solução inicial, r o número de estruturas de vizinhança e
    pMax o número máximo de perturbações na mesma vizinhança;
2   s ← s0;      {Solução atual}
3   enquanto (Critério de parada não satisfeito) faça
4           k ← 1;  {Vizinhança corrente}
5           p ← 1;  {#perturbações na mesma vizinhança}
6           enquanto (k ≤ r) faça
7                   s' ← Shake(s, k, p);
8                   s'' ← BuscaLocal(s');
9                   se ( f(s'') < f(s) )
10                      então s ← s''; k ← 1; p ← 1;
11                      senão
12                          se p ≥ pMax
13                             então p ← 1; k ← k + 1;
14                             senão p ← p + 1;
15                          fim-se;
16                  fim-se;
17          fim-enquanto;
18  fim-enquanto;
19  Retorne s;
fim Smart VNS;
```