CAP 254

Otimização Combinatória

Professor: **Dr. L.A.N. Lorena**

Assunto: Metaheurísticas

Antonio Augusto Chaves

Conteúdo

C01 – Simulated Annealing (20/11/07).

C02 – Busca Tabu (22/11/07).

C03 – Colônia de Formigas (27/11/07).

C04 - GRASP e VNS (29/11/07).

C05 – Metaheurísticas Híbridas – CS (04/12/07).

Material baseado nas notas de aula do Prof. Dr. Marcone Jamilson Freitas Souza (UFOP)
http://www.decom.ufop.br/prof/marcone/
e em material do Prof. Maurício G. C. Resende (AT&T Labs)
http://www.research.att.com/~mgcr/

GRASP

GRASP

- Greedy Randomized Adaptive Search Procedure (GRASP)

  Proposta por **T.A. Feo e M.G.C. Resende**

  http://mauricio.resende.info/MiniCursoGRASP.pdf

Algumas Referências:

- Feo & Resende (1989): GRASP introduced for set covering
- Resende (1989): talk on GRASP at INFORMS NYC Meeting
- Feo & Resende (1991): tutorial at INFORMS Nashville Meeting
- Feo & Resende (1995): first survey
- Festa & Resende (2002): annotated bibliography
- Resende & Ribeiro (2003): most recent survey
- Resende & González Velarde (2003): survey in Spanish

GRASP

- Origens:

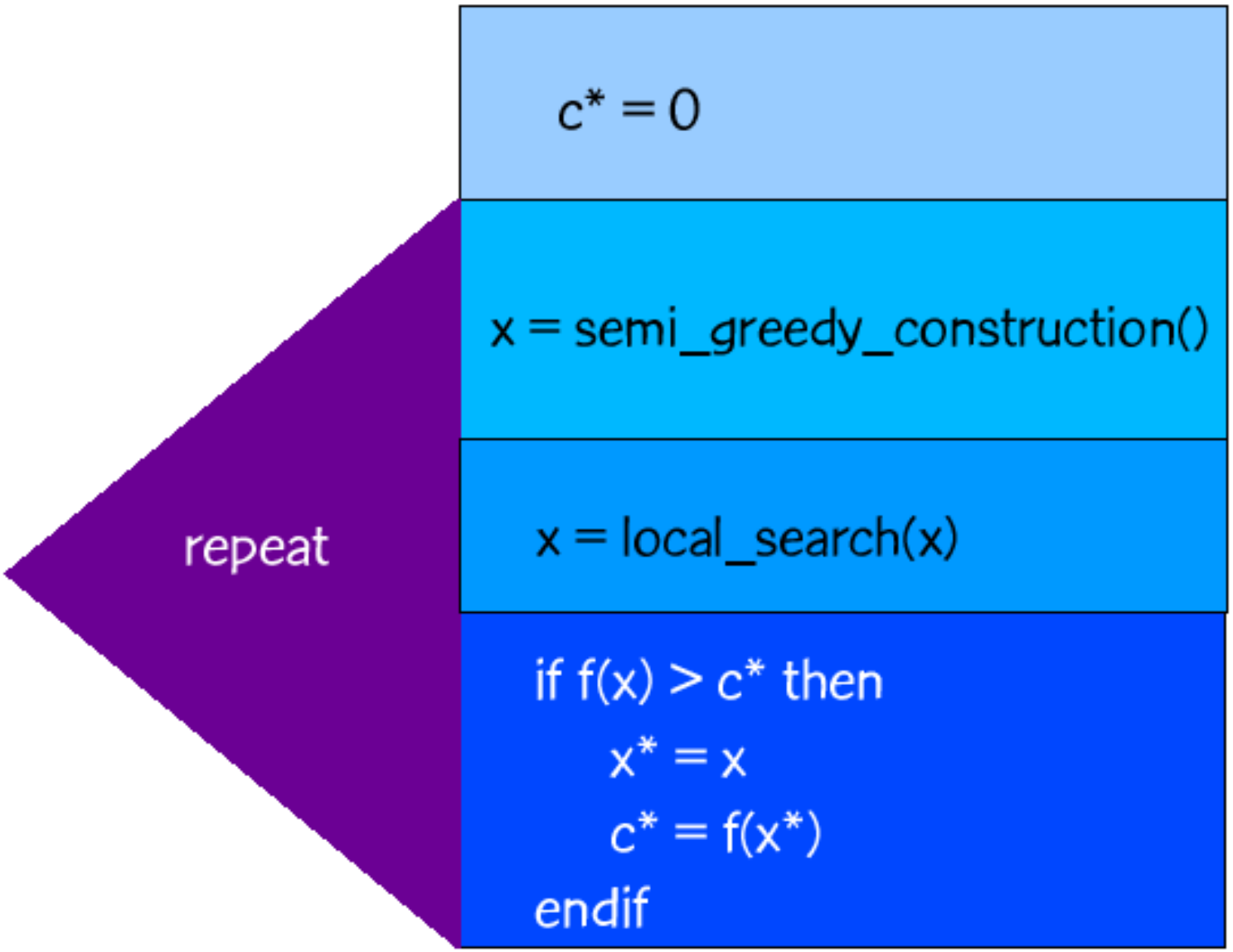
  Probabilistic algorithm (GRASP) for difficult set covering problems
  [Feo & Resende., 1989]

  As idéias do GRASP foram relatadas em trabalhos anteriores, por exemplo:

  random multistart local search [e.g. Lin & Kernighan, 1973]

  semi-greedy heuristics [e.g. Hart & Shogan, 1987]

GRASP: Algoritmo Básico



GRASP: Algoritmo Básico

- É um processo iterativo ou multi-start
- **Fase de Construção**: "gulosidade" + aleatoriedade
  - Construir uma solução viável combinando "gulosidade" e aleatoriedade
- **Fase de Busca Local**: pesquisar a vizinhança corrente até um ótimo local ser encontrado
  - Soluções geradas pelo procedimento de construção não são necessariamente ótimas:
    - Eficácia da busca local depende da: estrutura de vizinhança, estratégia de busca, e da rápida avaliação dos vizinhos, mas também do próprio procedimento de construção.

GRASP: Fase de Construção

Algoritmo Guloso (Greedy)

- Para definir uma heurística semi-gulosa, nós precisamos primeiro considerar um algoritmo guloso.

- Algoritmo Guloso: construir uma solução, um elemento por vez:

repetir até construir

  - Definir os elementos candidatos.
  - Aplicar uma função gulosa para cada elemento candidato.
  - "Rankear" os elementos de acordo com o valor da função gulosa.
  - Adicionar o melhor elemento do "rank" na solução.

Algoritmo Semi-Guloso

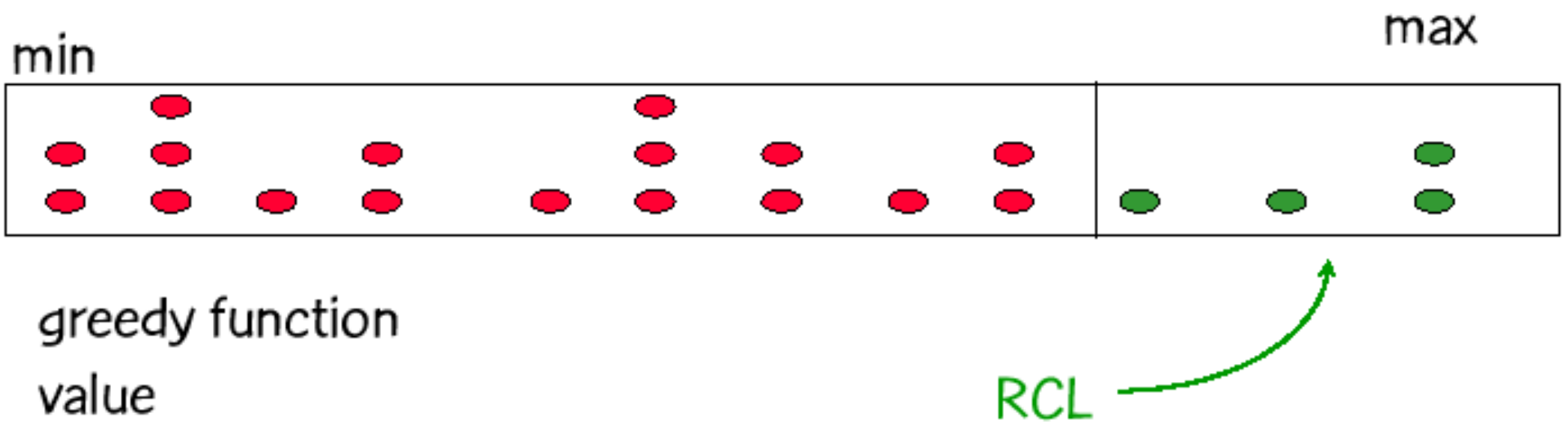
- Uma heurística semi-gulosa tenta ir ao redor da convergência para soluções que não sejam mínimos locais.

- Repetir até uma solução ser construída

  - Para cada elemento candidato
    - aplicar uma função gulosa no elemento
  - "Rankear" todos os elementos de acordo com seus valores da função gulosa
  - Colocar os elementos melhores "rankeados" na lista de candidatos restrita (LCR) ou restricted candidate list (RCL)
  - Selecionar um elemento da RCL aleatoriamente e adicionar este na solução

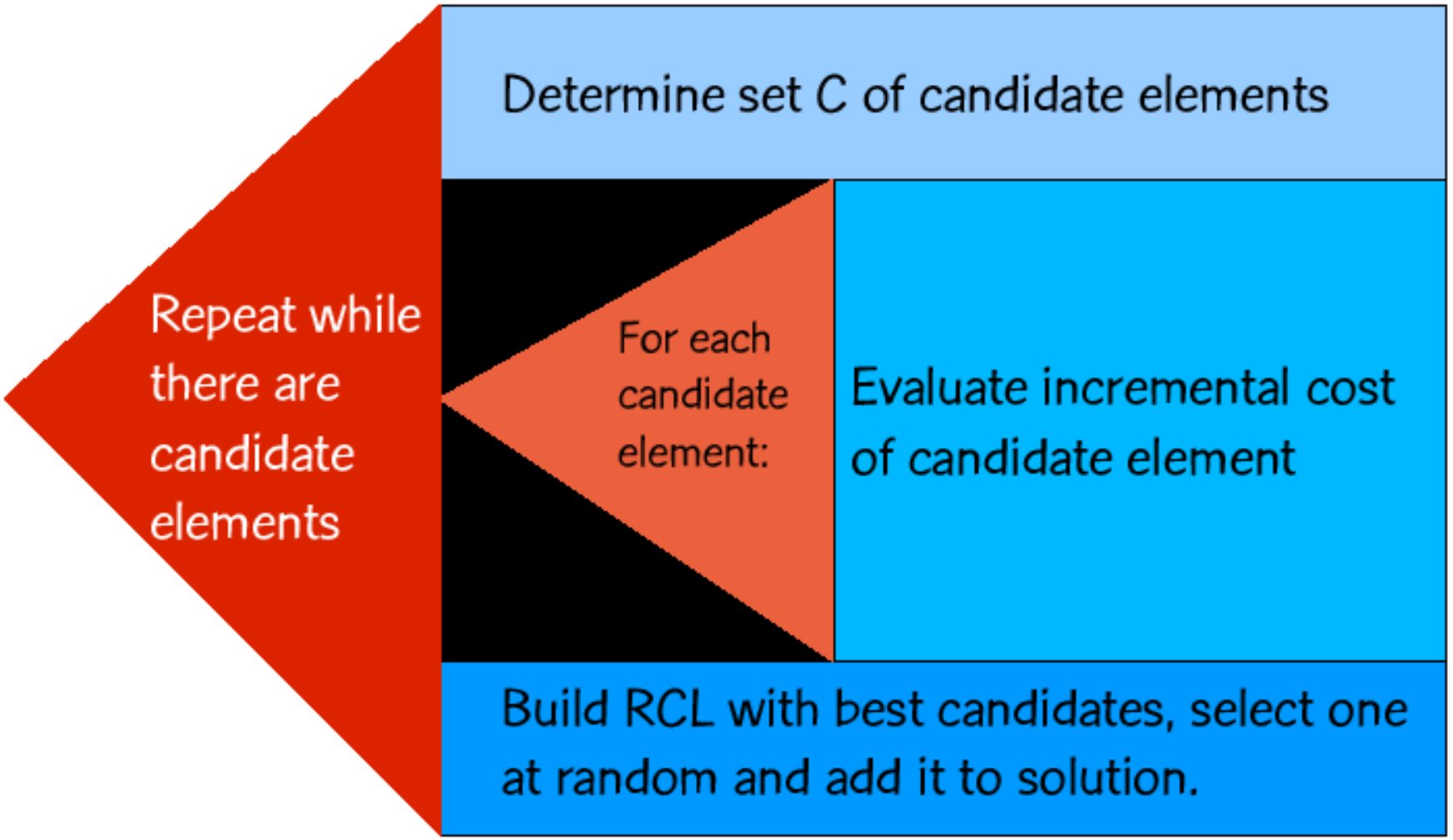
Algoritmo Semi-Guloso

- Elementos candidatos são “rankeados de acordo com o valor da função gulosa.



- RCL é um conjunto dos elementos candidatos melhores “rankeados”.

GRASP: Fase de Construção



GRASP: Fase de Construção

- Problema de Minimização

- Procedimento de construção básico:
  - Função gulosa c(e): custo associado com a incorporação do elemento e na solução corrente parcial
  - c^{min}: menor valor de custo
  - RCL é composta pelos elementos com os menores custos

GRASP: Fase de Construção

Como determinar o tamanho da RCL?

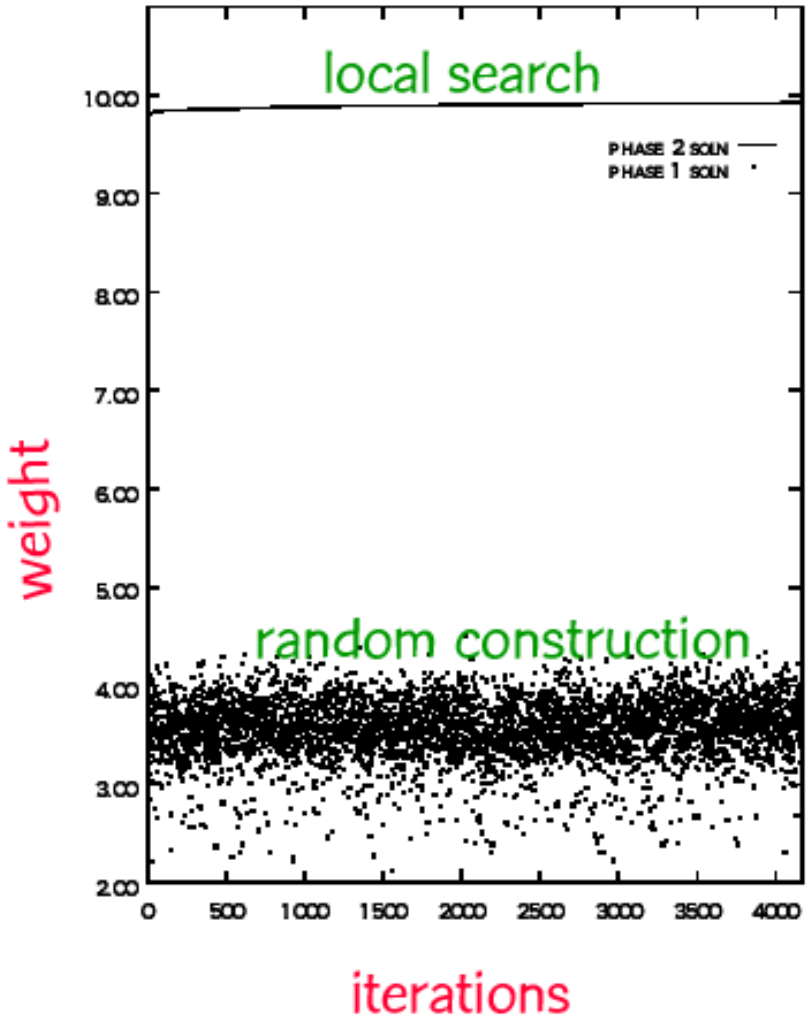
- Construção baseada em Cardinalidade:
  - p elementos com os menores custos
- Construção baseada em Qualidade:
  - Parâmetro α define a qualidade dos elementos na RCL
  - RCL contém elementos com custo

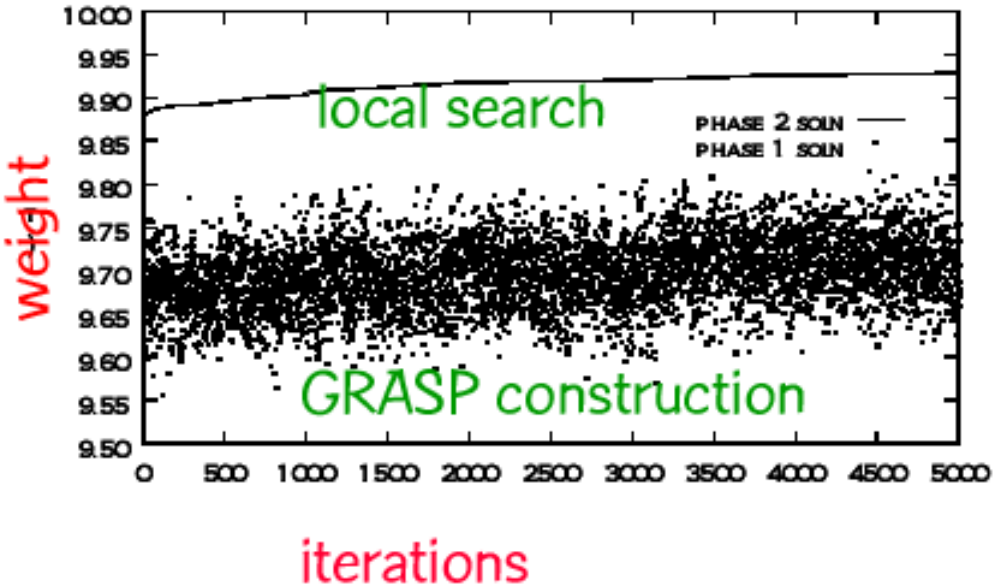
    $c^{min} \leq c(e) \leq c^{min} + \alpha\ (c^{max} - c^{min})$

    - $\alpha = 0$: construção puramente gulosa
    - $\alpha = 1$: construção puramente aleatória
- Selecionar elementos aleatoriamente para a RCL utilizando probabilidades com distribuição uniforme

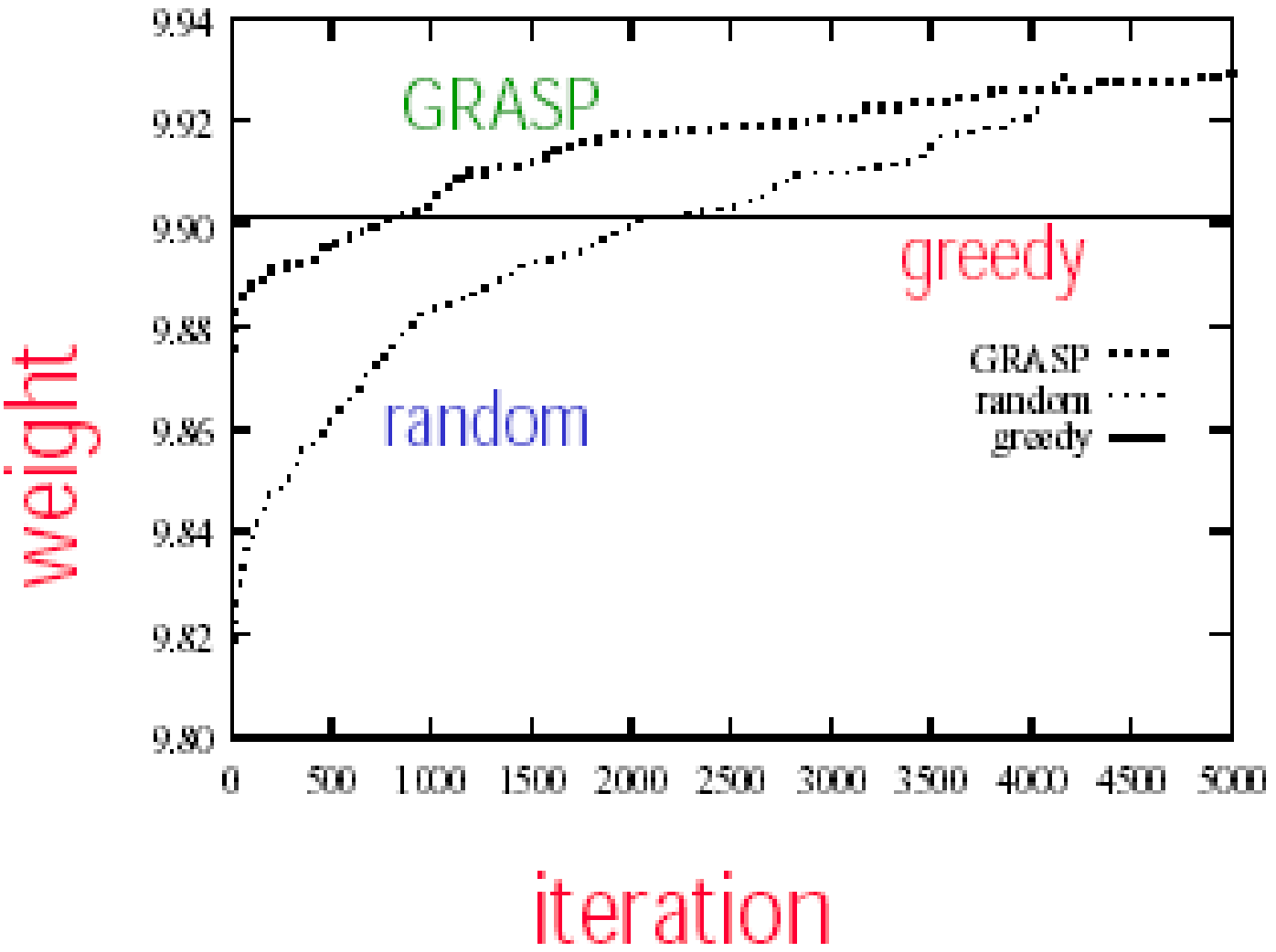
GRASP: Fase de Construção

- Eficácia da construção semi-gulosa vs. aleatória





Processo de Convergência do GRASP



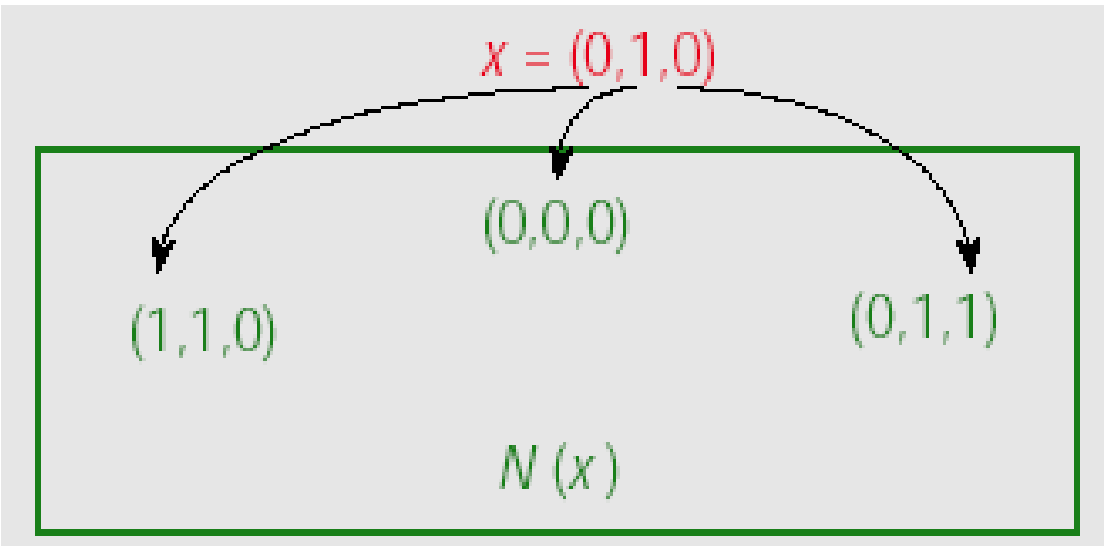
GRASP: Fase de Busca Local

Busca Local

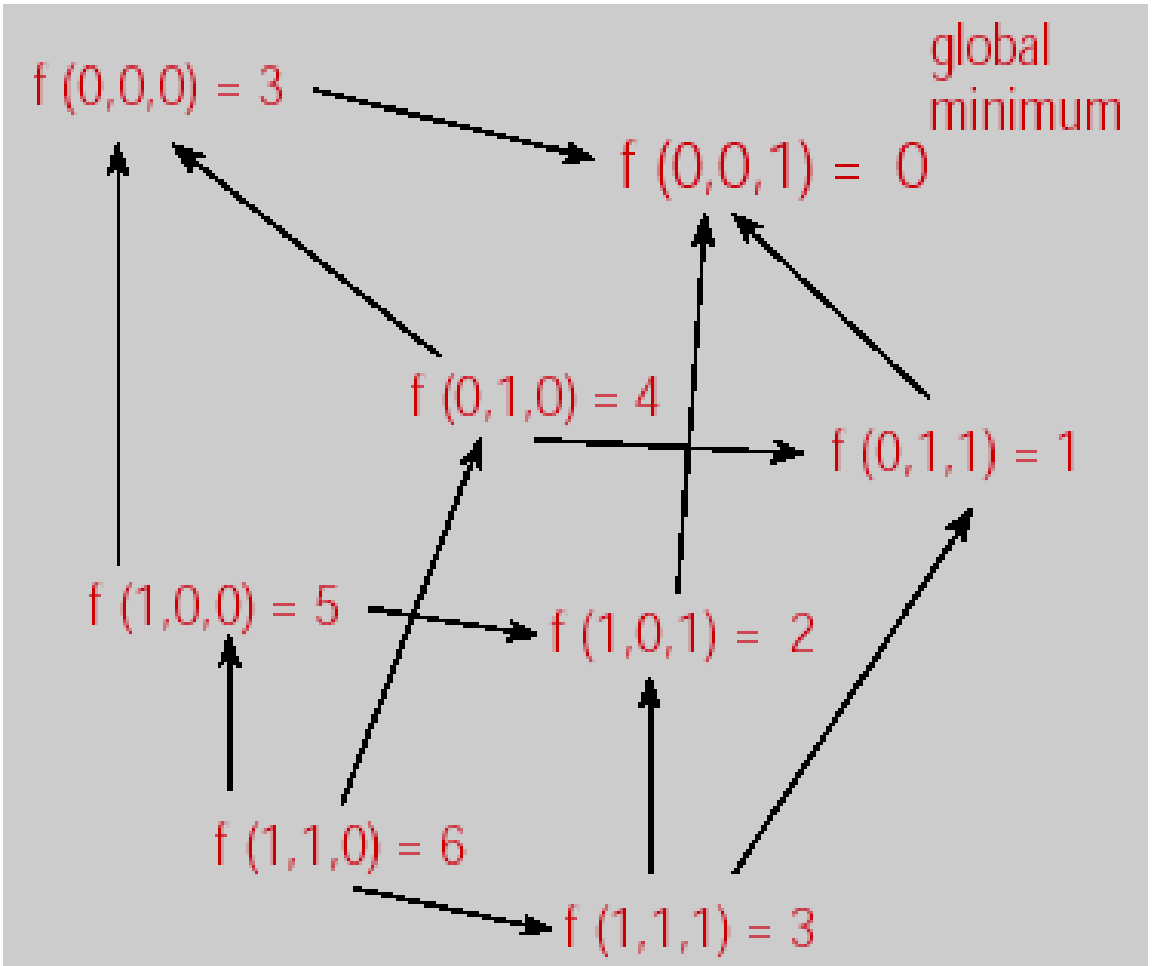
- Para definir busca local, primeiro precisamos especificar uma estrutura de vizinhança local

- 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 (geralmente chamada de movimento) em x.

- Considere x = (0,1,0) e a vizinhança 1-flip de um vetor 0/1

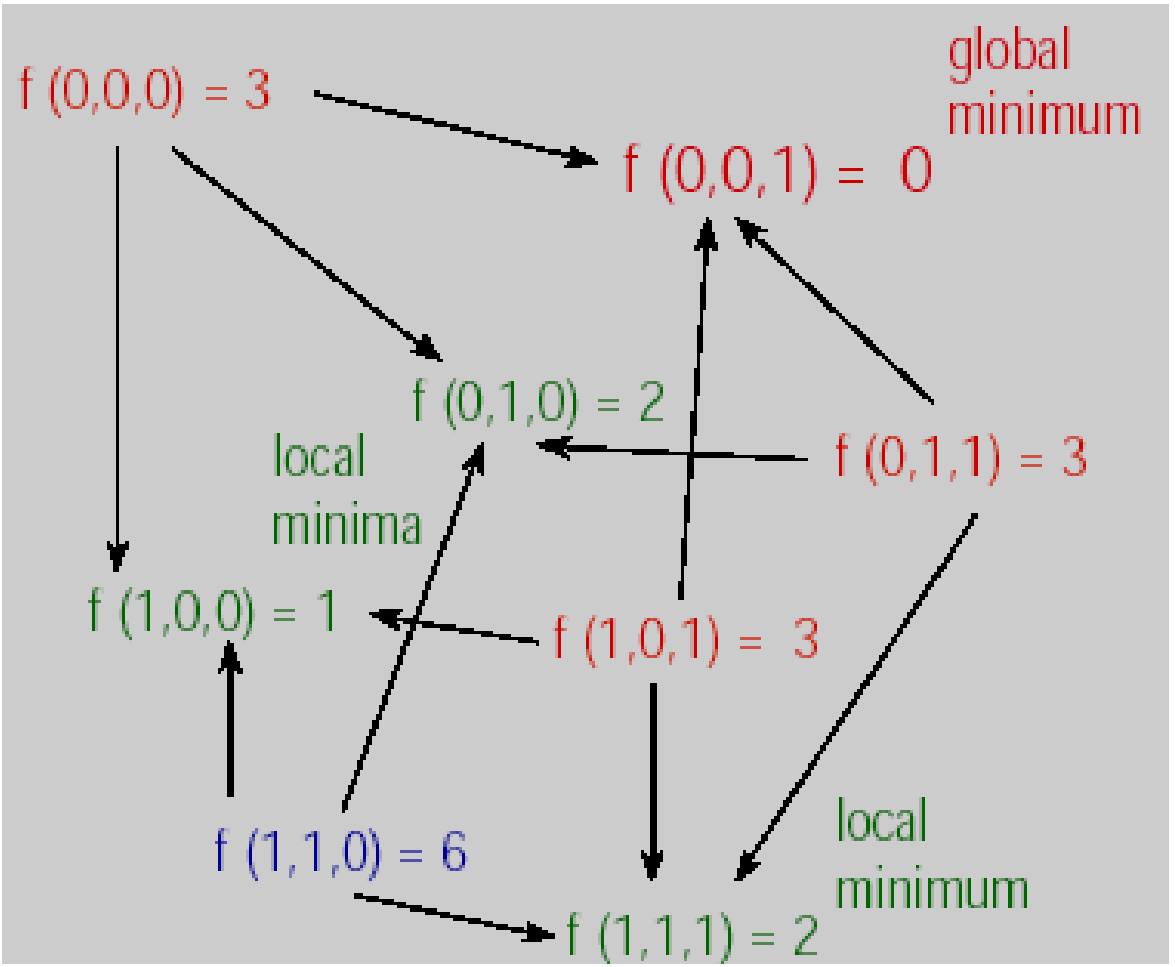


Busca Local (condição ideal)



Com qualquer solução inicial a Busca Local encontra o ótimo global

Busca Local (situação mais real)



Mas algumas soluções iniciais levam a Busca Local para um mínimo local

Busca Local com GRASP

- First improving vs. best improving:
  - First improving é geralmente mais rápida.
  - Convergência prematura para um ótimo local de baixa qualidade é mais comum de ocorrer com best improving.

- Filtragem: evitar a aplicação de busca local em soluções de baixa qualidade, somente soluções promissoras não visitadas são investigadas

- Variable Neighborhood Descent (VND): acelerar a busca e superar o ótimo encontrando com uma única vizinhança

Busca Local com GRASP

- GRASP com Path-Relinking

  Após cada iteração GRASP (construção e busca local)

  – Utilizar a solução GRASP como solução inicial

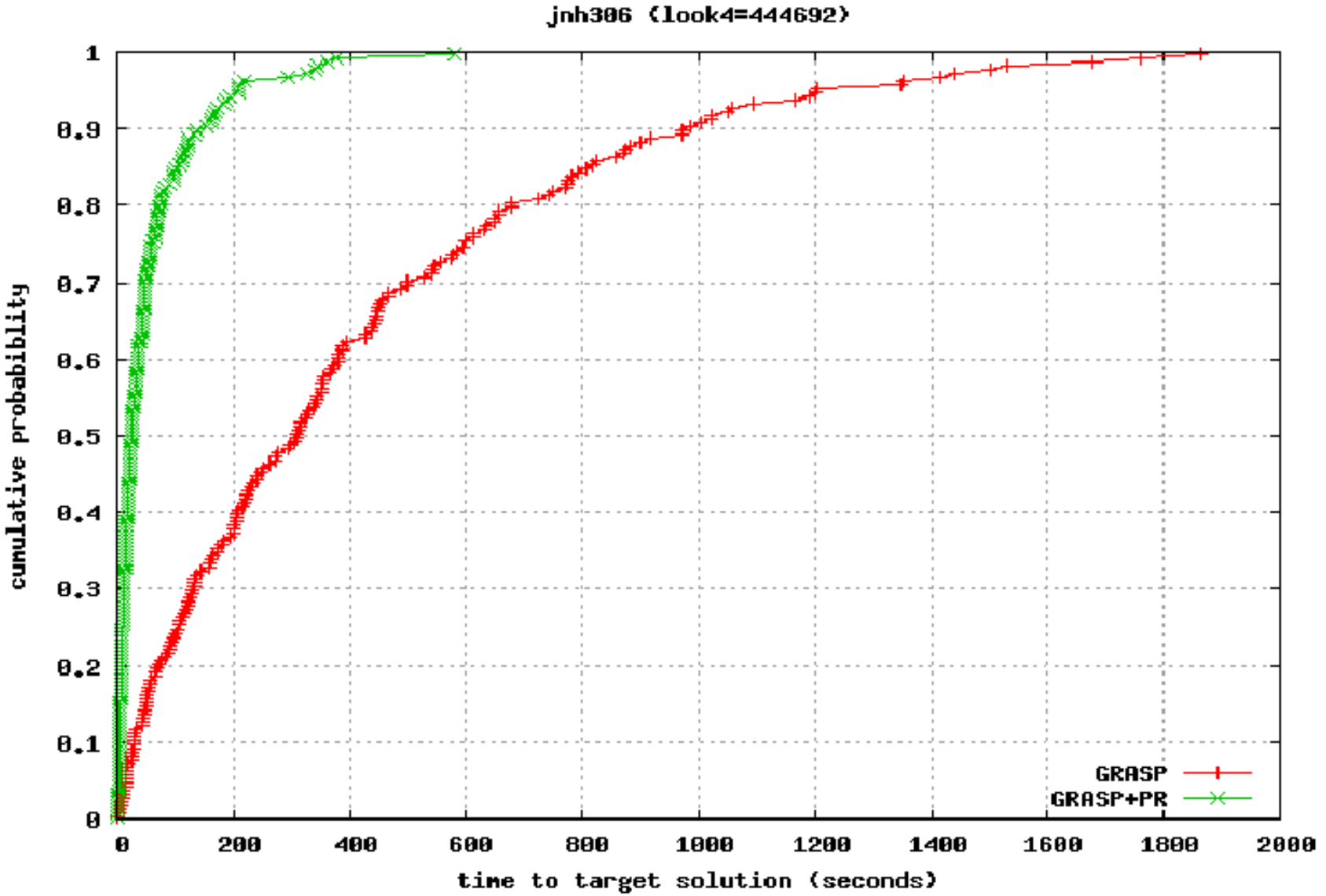
  – Selecionar uma solução elite: solução guia

  – Executar Path-Relinking entre estas duas soluções

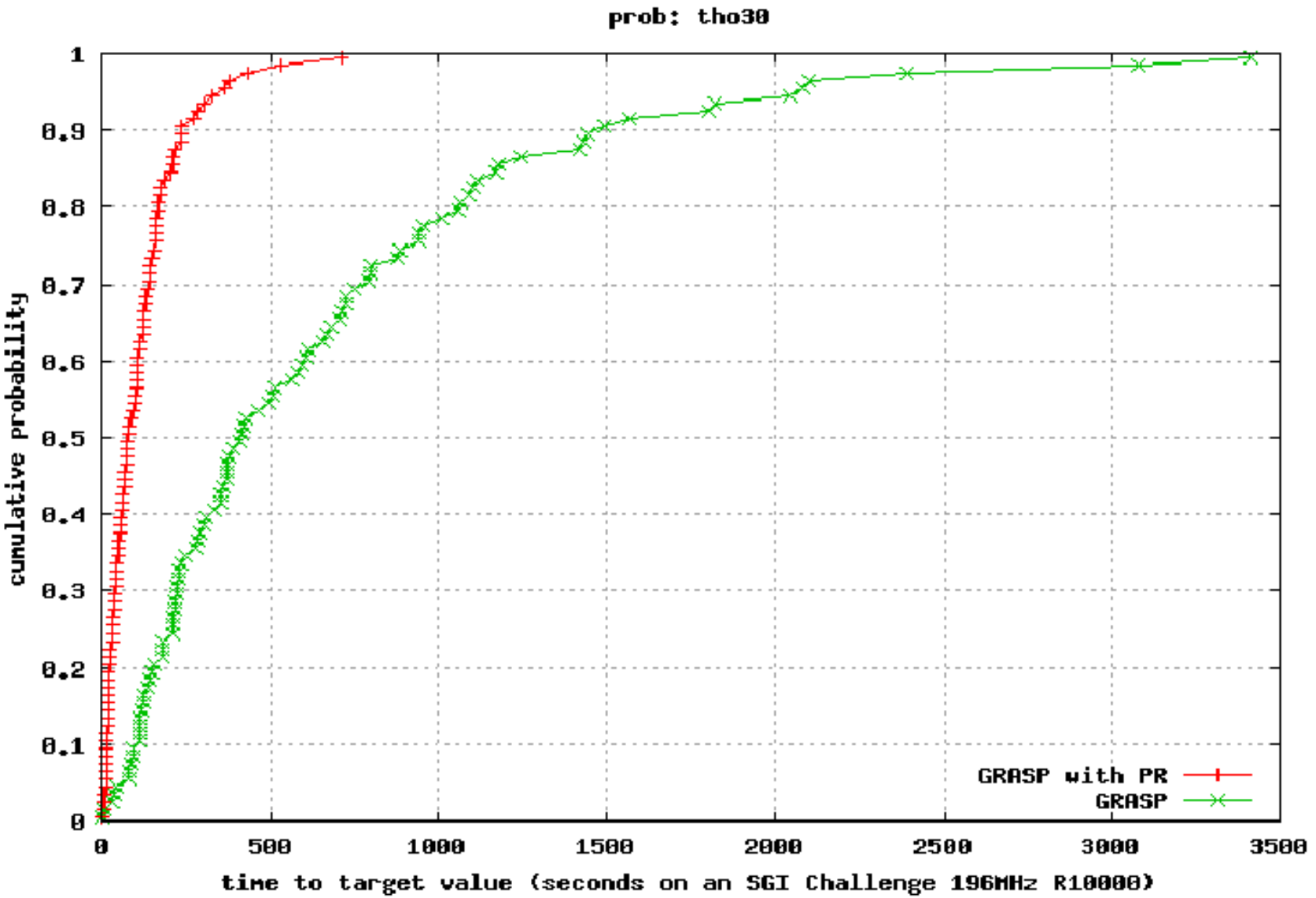
- Observação através de estudos experimentais:

  GRASP com Path-Relinking possui uma performance melhor que GRASP puro.

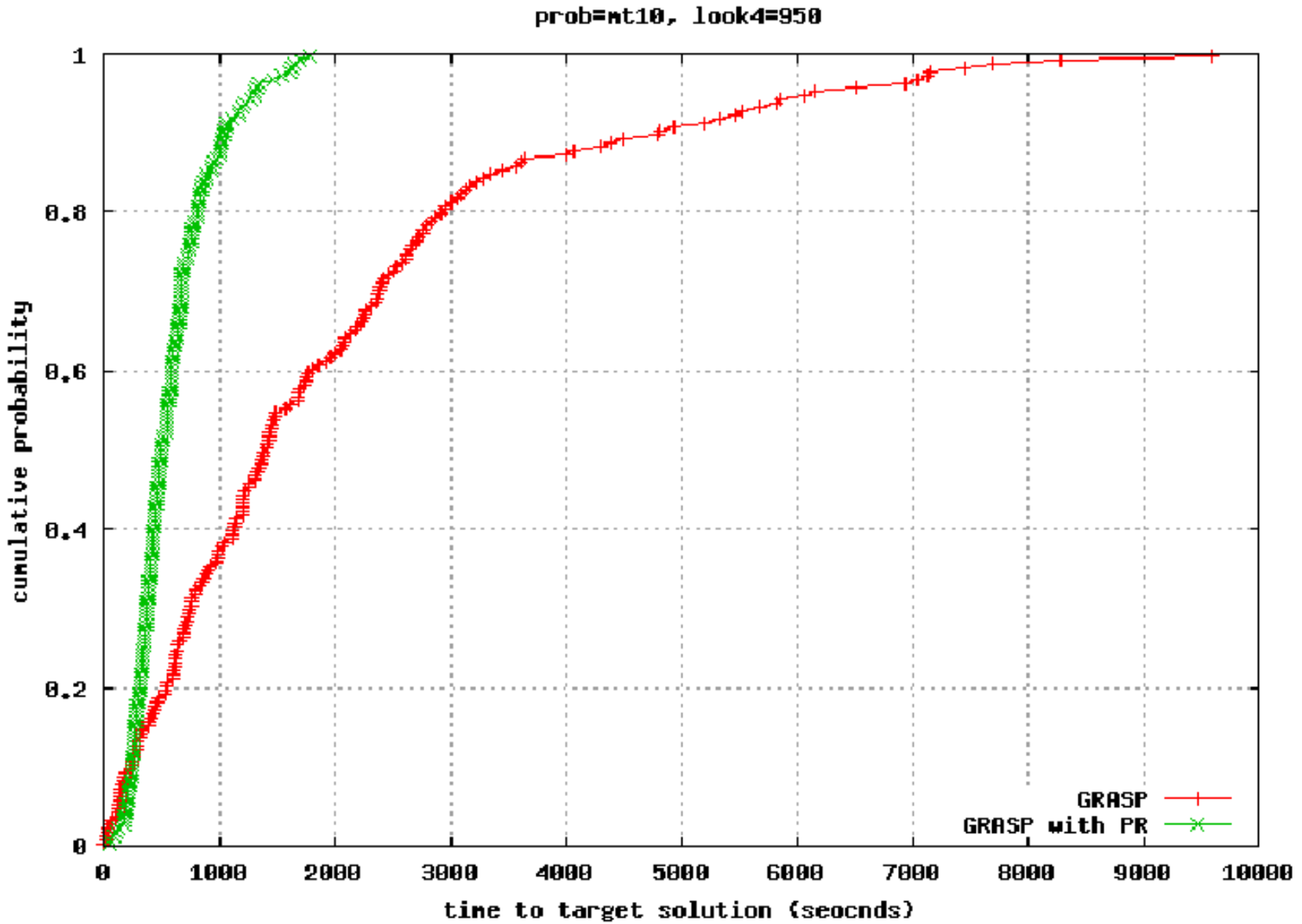
MAX-SAT
(Festa, Pardalos, Pitsoulis, e Resende, 2006)



QAP
(Oliveira, Pardalos e Resende, 2004)



Job shop scheduling (Aiex, Binato e Resende, 2003)



Pseudo-código GRASP

```
procedure GRASP(Max_Iterations, Seed)
1     Read_Input();
2     for k = 1,...,Max_Iterations do
3           Solution ← Greedy_Randomized_Construction(Seed);
4           Solution ← Local_Search(Solution);
5           Update_Solution(Solution, Best_Solution);
6     end;
7     return Best_Solution;
end GRASP.
```

GRASP: Fase de Construção

```
procedure Greedy_Randomized_Construction(Seed)
1     Solution ← ∅;
2     Evaluate the incremental costs of the candidate elements;
3     while Solution is not a complete solution do
4           Build the restricted candidate list (RCL);
5           Select an element s from the RCL at random;
6           Solution ← Solution ∪ {s};
7           Reevaluate the incremental costs;
8     end;
9     return Solution;
end Greedy_Randomized_Construction.
```

GRASP: Fase de Busca Local

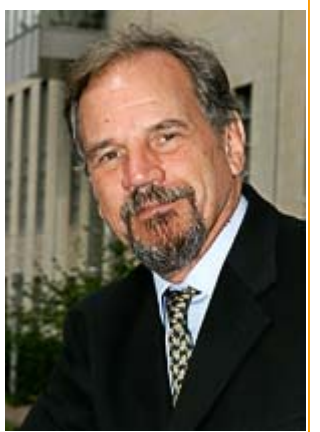
```
procedure Local_Search(Solution)
1     while Solution is not locally optimal do
2           Find s' ∈ N(Solution) with f(s') < f(Solution);
3           Solution ← s';
4     end;
5     return Solution;
end Local_Search.
```

VNS/VND

Variable Neighborhood Search (VNS)

- Nenad Mladenovic & Pierre Hansen em 1997

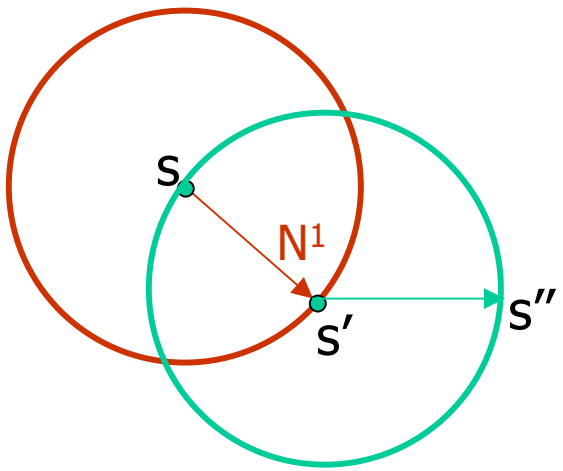
- Método de busca local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança



- Explora vizinhanças gradativamente mais "distantes"

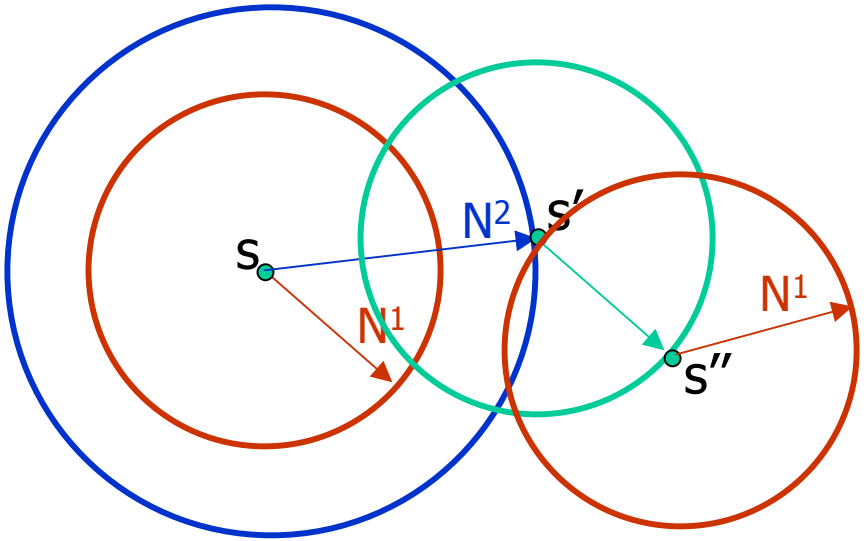
- Focaliza a busca em torno de uma nova solução somente se um movimento de melhora é realizado

Variable Neighborhood Search (VNS)

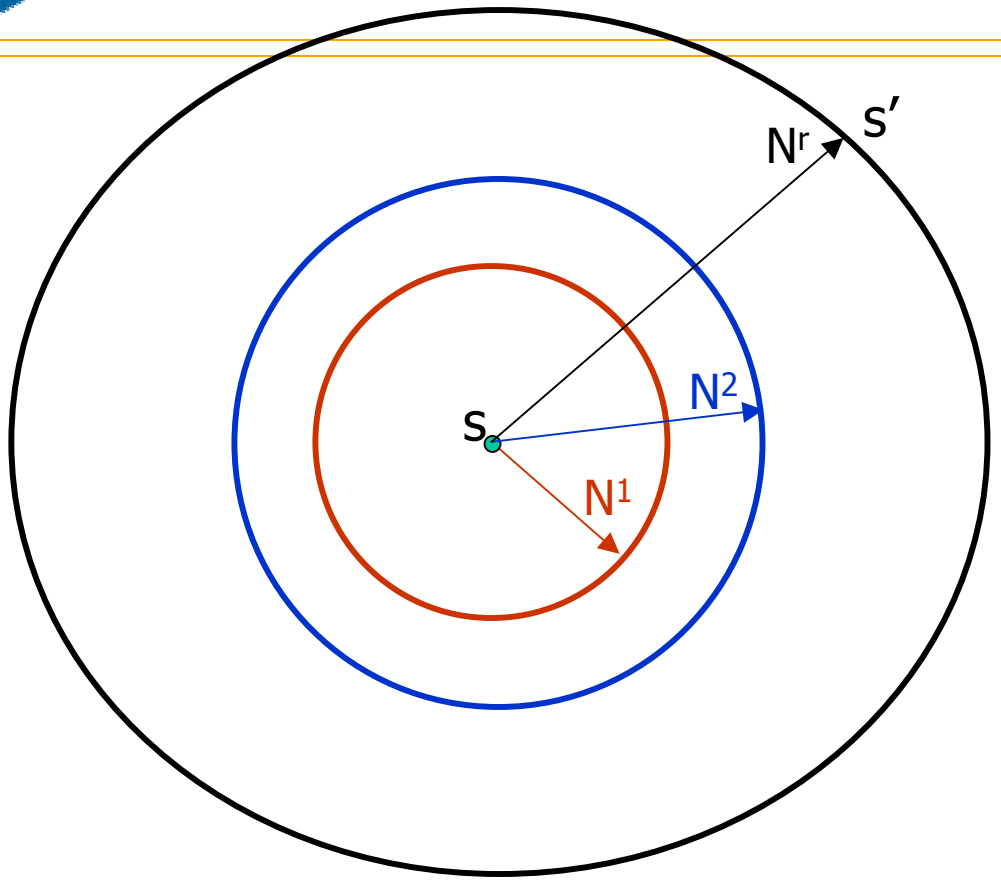


s″ aceito se f(s″) < f(s)

Variable Neighborhood Search (VNS)



Variable Neighborhood Search (VNS)



Variable Neighborhood Search (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               s'' ← BuscaLocal(s');
8               se ( f(s'') < f(s) )
9                     então s ← s''; k ← 1;
10              senão k ← k + 1;
11        fim-enquanto;
12  fim-enquanto;
13  Retorne s;
fim VNS;
```

Variable Neighborhood Descent (VND)

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

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

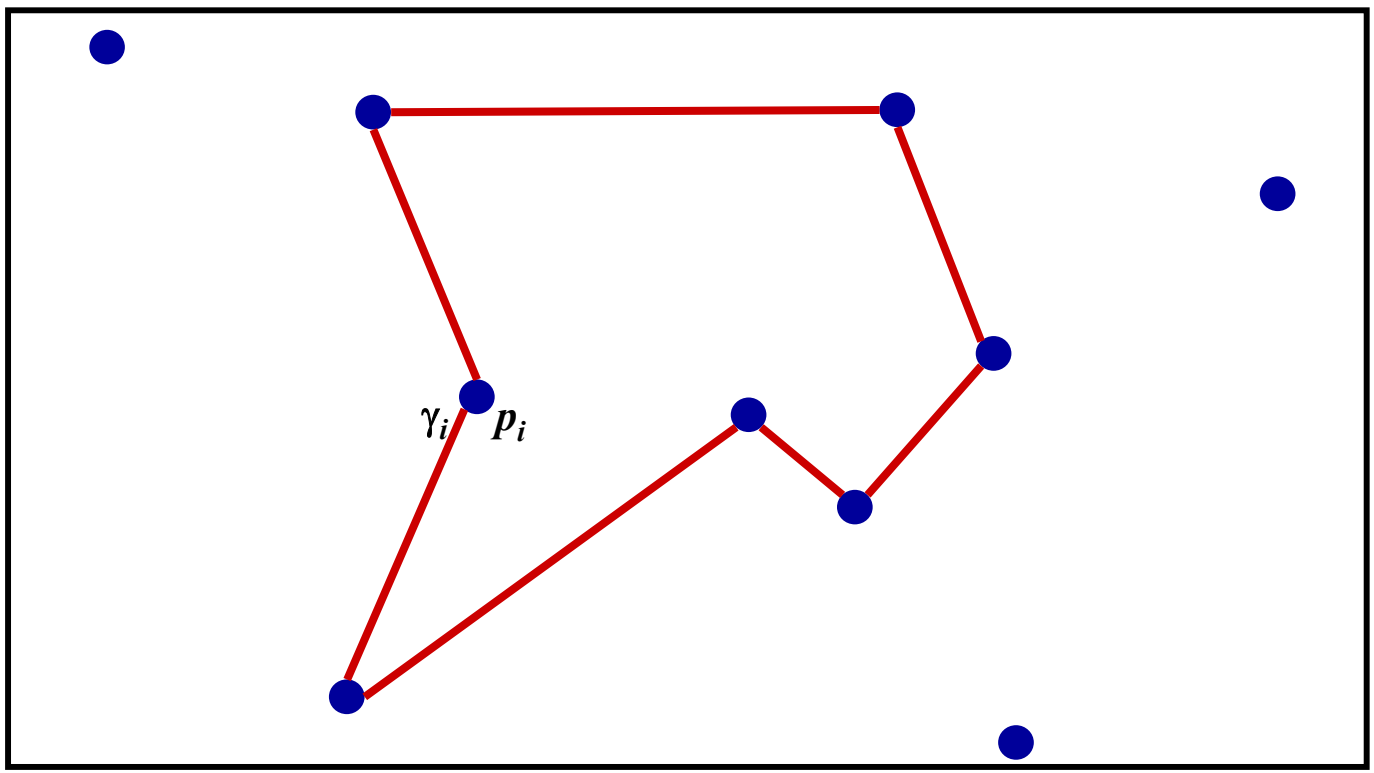
Variable Neighborhood Descent (VND)

```
1   Seja s_0 uma solução inicial e r o número de estruturas de vizinhança;
2   s ← s_0;                          {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;
10  fim-enquanto;
11  Retorne s;
fim VND;
```

GRASP/VNS aplicado ao PCTSP

PCV com Coleta de Prêmios (Prize-Collecting TSP – PCTSP)

$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} + \sum_{v_i \in V} \gamma_{v_i} (1 - y_{v_i}), \text{ sujeito à } \sum_{v_i \in V} p_{v_i} y_{v_i} \geq p_{\min}$$



GRASP/VNS aplicado ao PCTSP

- Fase de construção do GRASP
  - Lista de elementos candidatos (C)
    - Função Gulosa: $v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$
  - A cada iteração um vértice é selecionado aleatoriamente a partir da lista de candidatos restrita (LCR), e a lista C é atualizada.
  - Parâmetro $\alpha \in [0,1]$ determina o tamanho da LCR.
  - Termina quando não houver candidatos com custo de inserção positivo e o prêmio mínimo tenha sido coletado.

GRASP/VNS aplicado ao PCTSP

- Fase de busca local: VNS
  - As estruturas de vizinhança são definidas através de movimentos aleatórios:
    - m_1: Inserir 1 vértice que não está sendo visitado;
    - m_2: Retirar 1 vértice que está sendo visitado;
    - m_3: Trocar 2 vértices que estão sendo visitados de posição;
    - m_4: Inserir 2 vértices que não estão sendo visitados;
    - m_5: Retirar 2 vértices que estão sendo visitados;
    - m_6: Trocar 4 vértices que estão sendo visitados de posição;
    - m_7: Inserir 3 vértices que não estão sendo visitados;
    - m_8: Retirar 3 vértices que estão sendo visitados;
    - m_9: Trocar 6 vértices que estão sendo visitados de posição;

GRASP/VNS aplicado ao PCTSP

- A cada iteração seleciona-se aleatoriamente um vizinho, sendo este submetido a uma busca local.
- Se o ótimo local for melhor a que a solução corrente, a busca continua desta solução recomeçando da primeira estrutura de vizinhança. Caso contrário, a busca continua a partir da próxima vizinhança.

- A busca local aplicada no vizinho é o VND
  - SeqDrop
  - Add-Drop
  - SeqAdd

GRASP/VNS aplicado ao PCTSP

procedimento GRASP/VNS
para (número de iterações não satisfeito) **faça**

Fase de Construção:

$s = \varnothing$
enquanto (solução não construída) **faça**
produzir Lista de Candidatos (C) $\longrightarrow v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$
LCR = C * α
e = selecionar elemento aleatoriamente(LCR)
$s = s \cup \{e\}$
atualizar lista de Candidatos(C)
fim-enquanto

Fase de Busca Local (VNS):

k_{max} = número de vizinhanças
enquanto (critério de parada não satisfeito) **faça**
$k \leftarrow 1$
enquanto ($k \leq k_{max}$)
gerar um vizinho s‘ qualquer na vizinhança $N^k(s)$
s'' = Aplicar VND em s'
se (s'' for melhor que s) **faça**
$s \leftarrow s$''
$k \leftarrow 1$
senão
$k \leftarrow k + 1$
fim-enquanto
fim-enquanto
fim-para
fim-GRASP/VNS

Conteúdo do Curso

C01 – Simulated Annealing (20/11/07).

C02 – Busca Tabu (22/11/07).

C03 – Colônia de Formigas (27/11/07).

C04 - GRASP e VNS (29/11/07).

C05 – Metaheurísticas Híbridas – CS (04/12/07).