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).

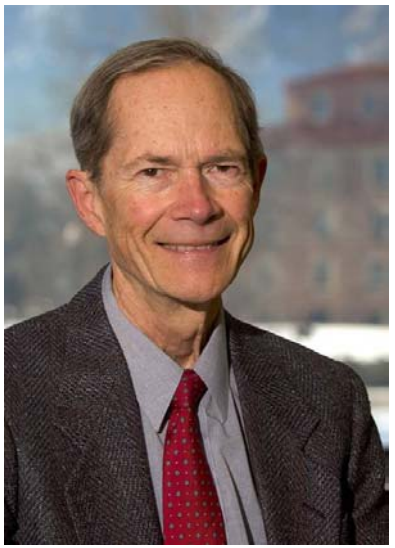
C03 – Colônia de Formigas (27/11/07).

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

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

Busca Tabu (Tabu Search)

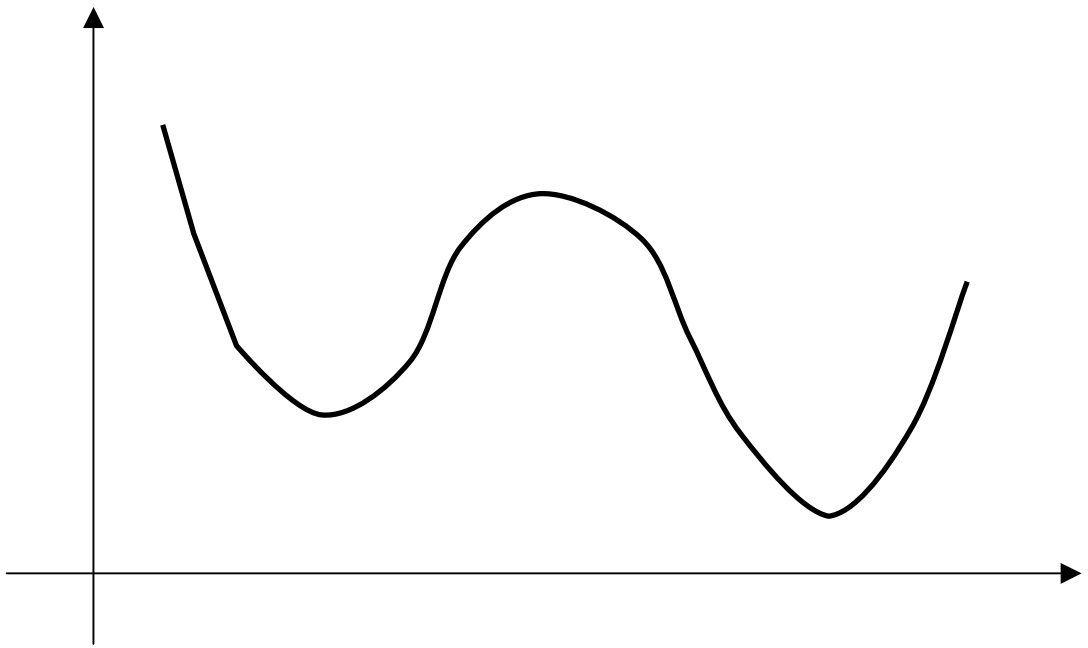
Busca Tabu



- Proposto por Fred Glover
  - "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research*, Vol. 13, No. 5, 533-549, 1986.

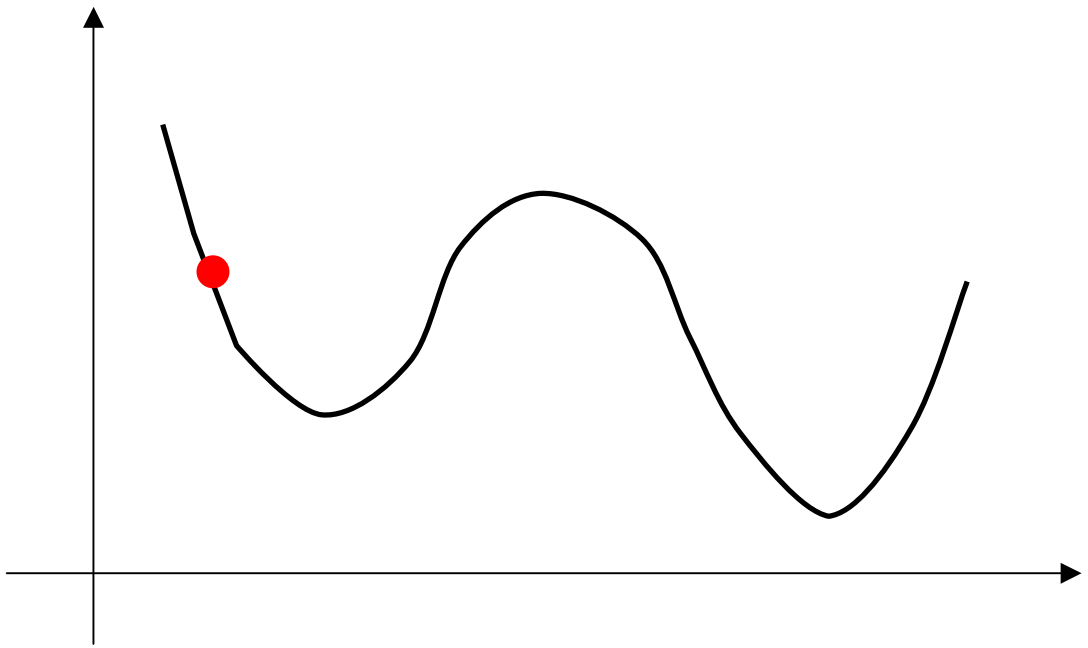
- Método de busca local
  - explorar o espaço de soluções movendo-se de uma solução para outra que seja seu melhor vizinho.
  - estrutura de memória para armazenar as soluções geradas (ou características dessas)
  - essas características possibilitam Busca Tabu escapar de ótimos locais

Busca Tabu



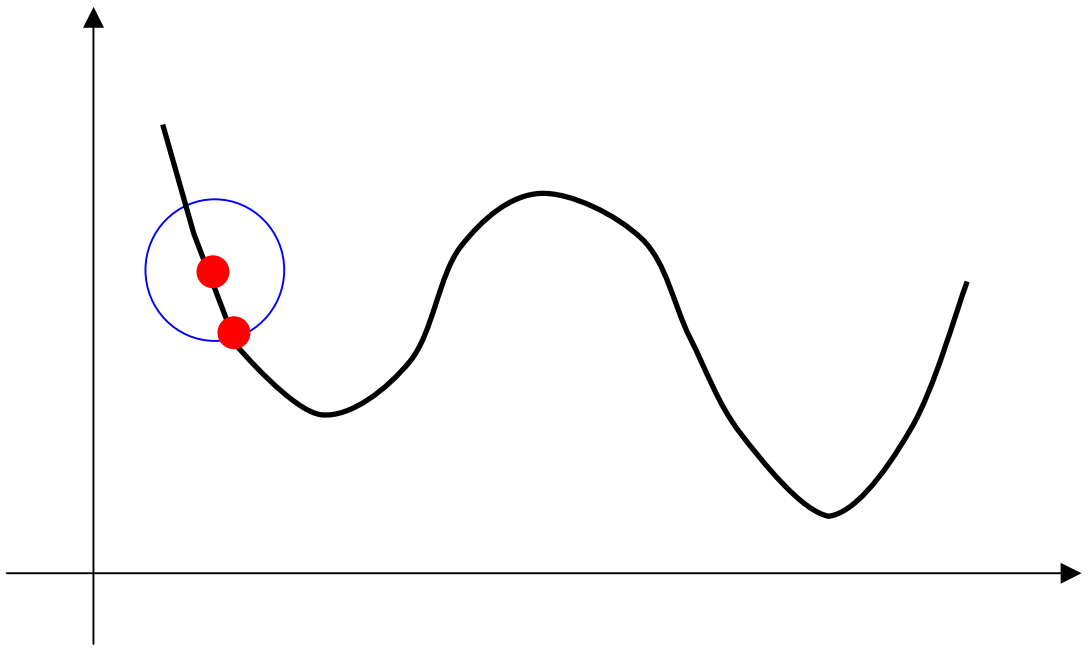
Busca Tabu

- 1ª Idéia: Utilizar Heurística de Descida



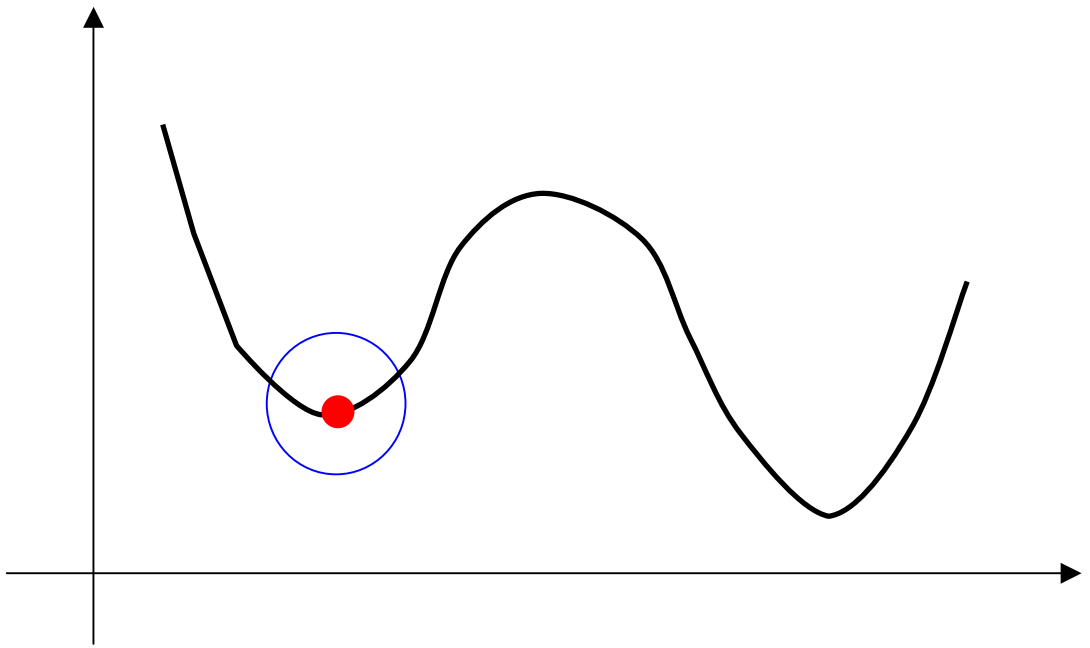
Busca Tabu

- 1ª Idéia: Utilizar Heurística de Descida



Busca Tabu

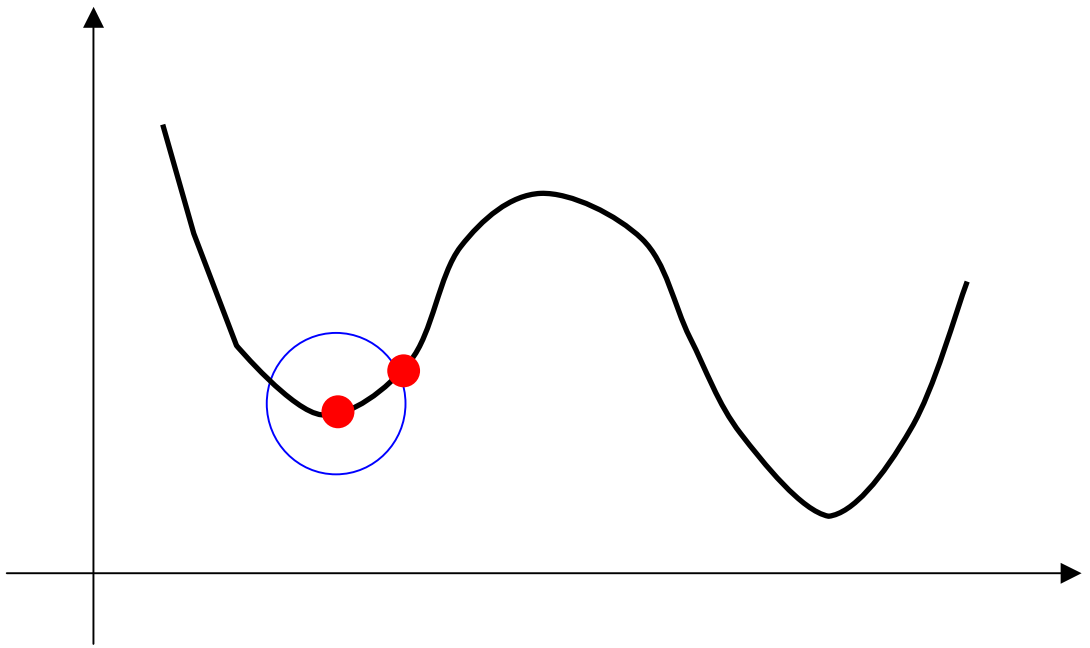
- 1ª Idéia: Utilizar Heurística de Descida



Problema: Fica-se preso no primeiro ótimo local

Busca Tabu

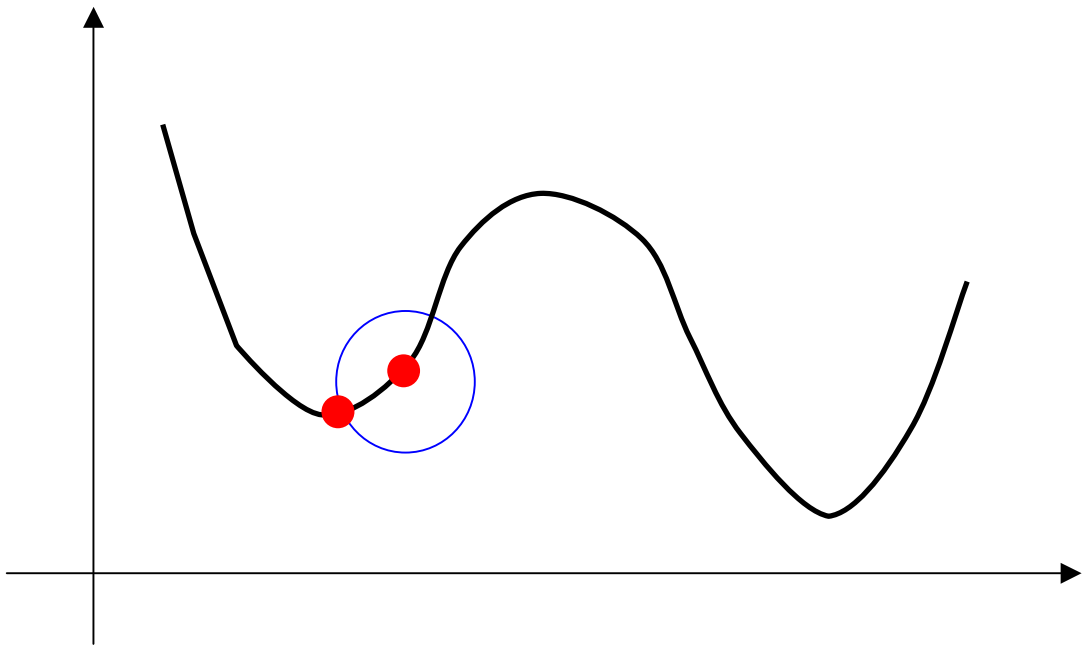
- 2ª Idéia: Mover para o Melhor Vizinho



O melhor vizinho pode ser de piora!

Busca Tabu

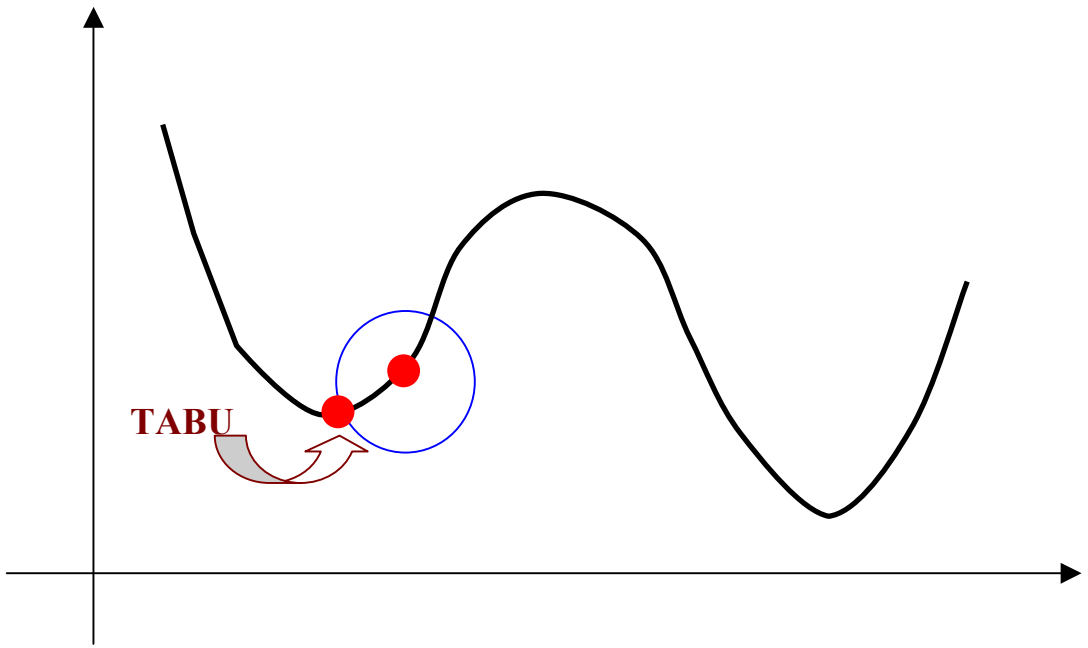
- 2ª Idéia: Mover para o Melhor Vizinho



Problema: Ciclagem

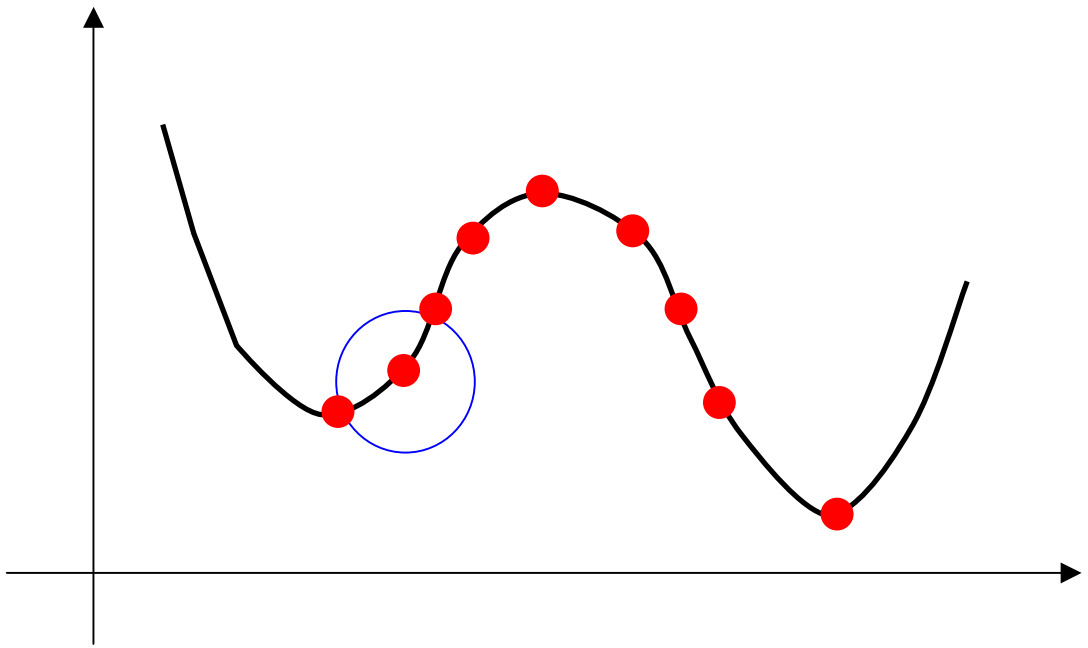
Busca Tabu

- 3ª Idéia: Criar uma Lista Tabu



Busca Tabu

- 3ª Idéia: Criar uma Lista Tabu



Problemas com uma Lista Tabu de soluções

- É computacionalmente inviável armazenar **todas** as soluções geradas!
  - Idéia: Armazenar apenas as últimas |T| soluções geradas
  - Observação: Uma lista com as |T| últimas soluções evita ciclos de até |T| iterações
  - Problema: Pode ser inviável armazenar |T| soluções e testar se uma solução está ou não na Lista Tabu
  - Idéia: Criar uma Lista Tabu de **movimentos reversos**
- Problema: Uma Lista Tabu de movimentos pode ser muito **restritiva** (impede o retorno a uma solução já gerada anteriormente e também a outras soluções ainda não geradas)

Busca Tabu

- 4ª Idéia: Critério de Aspiração

  - ✓ Retirar o *status* tabu de um movimento sob determinadas circunstâncias

  - ✓ Exemplo: aceitar um movimento, mesmo que tabu, se ele melhorar o valor da função objetivo global (Critério de aspiração por objetivo)

  - ✓ Aspiração por *default*: Realizar o movimento tabu mais antigo se todos os possíveis movimentos forem tabus.

Busca Tabu

- Busca Tabu começa a partir de uma solução inicial s_0 qualquer
- Um algoritmo Busca Tabu explora, a cada iteração, um subconjunto *V* da vizinhança *N*(*s*) da solução corrente *s*.
- O membro *s'* de *V* com **melhor valor** nessa região segundo a função *f*(.) torna-se a nova solução corrente mesmo que *s'* seja **pior** que *s*, isto é, que *f*(*s'*) > *f*(*s*) para um problema de minimização.
- O critério de escolha do melhor vizinho é utilizado para escapar de um ótimo local.
- Porém, esta estratégia pode fazer com que o algoritmo **cicle**, isto é, que retorne a uma solução já gerada anteriormente.

Busca Tabu

- Para evitar que isto ocorra, existe uma lista tabu *T*, a qual é uma lista de movimentos proibidos.

- A lista tabu clássica contém os movimentos reversos aos últimos |*T* | movimentos realizados (onde |*T* | é um parâmetro do método) e funciona como uma fila de tamanho fixo, isto é, quando um novo movimento é adicionado à lista, o mais antigo sai.

- Assim, na exploração do subconjunto *V* da vizinhança *N*(*s*) da solução corrente *s*, ficam excluídos da busca os vizinhos *s'* que são obtidos de *s* por movimentos *m* que constam na lista tabu.

Busca Tabu

- A lista tabu reduz o risco de ciclagem
  - garantindo o não retorno, por $|T|$ iterações, a uma solução já visitada anteriormente;
  - Mas, também pode proibir movimentos para soluções que ainda não foram visitadas.
    - *função de aspiração*, que é um mecanismo que retira, sob certas circunstâncias, o *status* tabu de um movimento.

- Para cada possível valor v da função objetivo existe um nível de aspiração $A(v)$: uma solução s' em V pode ser gerada se $f(s') < A(f(s))$, mesmo que o movimento m esteja na lista tabu.

- A função de aspiração A é tal que, para cada valor v da função objetivo, retorna outro valor $A(v)$, que representa o valor que o algoritmo aspira ao chegar de v.

Busca Tabu

- Um exemplo simples de aplicação desta idéia de aspiração é considerar $A(f(s)) = f(s^*)$ onde s^* é a melhor solução encontrada até então. Neste caso, aceita-se um movimento tabu somente se ele conduzir a um vizinho melhor que s^* (aspiração por objetivo).

- Esse critério se fundamenta no fato de que soluções melhores que a solução s^* corrente, ainda que geradas por movimentos tabu, não foram visitadas anteriormente, evidenciando que a lista de movimentos tabu pode impedir não somente o retorno a uma solução já gerada anteriormente mas também a outras soluções ainda não geradas.

Busca Tabu

- Duas regras são normalmente utilizadas de forma a interromper o procedimento.
  - Pela primeira, pára-se quando é atingido um certo número máximo de iterações sem melhora no valor da melhor solução.
  - Pela segunda, quando o valor da melhor solução chega a um limite inferior conhecido (ou próximo dele). Esse segundo critério evita a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é julgada suficientemente boa.

- Os parâmetros principais de controle do método de Busca Tabu são
  - a cardinalidade $|T|$ da lista tabu
  - a função de aspiração A
  - a cardinalidade do conjunto V de soluções vizinhas testadas em cada iteração
  - $BTmax$, o número máximo de iterações sem melhora no valor da melhor solução.

Procedimento Busca Tabu

procedimento *BT*
1. Seja s_0 solução inicial;
2. $s^* \leftarrow s_0$; {Melhor solução obtida até então}
3. Iter $\leftarrow$ 0; {Contador do número de iterações}
4. MelhorIter $\leftarrow$ 0; {Iteração mais recente que forneceu s^*}
5. Seja BTmax o número máximo de iterações sem melhora em s^*;
6. T $\leftarrow \varnothing$; {Lista Tabu}
7. Inicialize a função de aspiração A;
8. **enquanto** (Iter – MelhorIter $\leq$ BTmax) **faça**
9. Iter $\leftarrow$ Iter + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de V $\subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin$ T) ou s' atenda a condição de aspiração ($f(s') < A(f(s))$);
11. Atualize a Lista Tabu T;
12. $s \leftarrow s'$;
13. **se** $f(s) < f(s^*)$ **então**
14. $s^* \leftarrow s$;
15. MelhorIter $\leftarrow$ Iter ;
16. **fim-se**;
17. Atualize a função de aspiração A;
18. **fim-enquanto**;
19. **Retorne** s^*;

fim *BT*;

Busca Tabu aplicada ao Problema da Mochila

Seja uma mochila de capacidade b = 23

Objeto (j)	1	2	3	4	5
Peso (w_j)	4	5	7	9	6
Benefício (p_j)	2	2	3	4	4

Representação de uma solução: $s = (s_1, s_2, ..., s_5)$, onde $s_j \in \{0,1\}$

Movimento *m* = troca no valor de um bit

Lista tabu = {<posição do bit alterado>}

|T| = 1;

BTmax = 1;

Aspiração por objetivo.

Busca Tabu aplicada ao Problema da Mochila

- Função de Avaliação:

$$f(s) = \sum_{j=1}^{n} p_j s_j - \alpha \times \max\{0, \sum_{j=1}^{n} w_j s_j - b\}$$

- **Passo 0**: Seja uma solução inicial qualquer, por exemplo:

  s = (01010)

  $f(s)$ = 6

  Peso corrente da mochila = 14

  Lista tabu = $T = \emptyset$;

  Melhor solução até então: s^* = (01010) e $f(s^*)$ = 6

  Iter = 0; *MelhorIter* = 0;

Passo 1: *Devemos, agora, analisar todos os vizinhos de* s *e calcular a função de avaliação deles*

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}1010)^t$	18	8	8
$(0\mathbf{0}010)^t$	9	4	4
$(01\mathbf{1}10)^t$	21	9	9
$(0100\mathbf{0})^t$	5	2	2
$(0101\mathbf{1})^t$	20	10	10

Melhor vizinho: *s'* = (01011), com *f*(*s'*) = 10

Como *s'* é o melhor vizinho de *s*, então *s* ← *s'*, isto é, a nova solução corrente passa a ser: *s* = (01011)

Lista tabu = *T* = *{5}* (indicando que o bit da quinta posição não pode ser modificado, a não ser que o critério de aspiração seja satisfeito)

Melhor solução até então: *s** = (01011) e *f*(*s**) = 10 (pois *f*(*s'*) > *f*(*s**))

Iter = 1; *MelhorIter* = 1;

Como (*Iter* – *MelhorIter*) = (1 – 1) = 0 ≤ *BTmax* = 1, então o procedimento de exploração do espaço de soluções deve continuar.

Passo 2: *Determinemos, agora, o melhor vizinho de* s = *(01011):*

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}1011)^t$	24	12	-3
$(0\mathbf{0}011)^t$	15	8	8
$(01\mathbf{1}11)^t$	27	13	-47
$(010\mathbf{0}1)^t$	11	6	6
$(0101\mathbf{0})^t$	14	6	6

Melhor vizinho: *s'* = (00011), com *f*(*s'*) = 8

Como *s'* é o melhor vizinho de *s*, então *s* ← *s'* (mesmo sendo *f*(*s'*) pior que *f*(*s*)), isto é, a nova solução corrente passa a ser: *s* = (00011)

Lista tabu = *T* = {2} (observa-se que, como a cardinalidade da lista tabu foi fixada em um, então o movimento proibido anterior sai e entra o novo movimento proibido, isto é, o bit da segunda posição não pode ser modificado, a não ser que o critério de aspiração seja satisfeito)

Melhor solução até então: *s** = (01011) e *f*(*s**) = 10

Iter = 2; *MelhorIter* = 1;

Como (*Iter* – *MelhorIter)* = (2 – 1) = 1 ≤ *BTmax* = 1, então o BT continua.

***Passo 3**: Determinemos, agora, o melhor vizinho de* s = *(00011)*

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}0011)^t$	19	10	10
$(0\mathbf{1}011)^t$	20	10	10
$(00\mathbf{1}11)^t$	22	11	11
$(000\mathbf{0}1)^t$	6	4	4
$(0001\mathbf{0})^t$	9	4	4

Melhor vizinho: *s'* = (00111), com *f*(*s'*) = 11

Como *s'* é o melhor vizinho de *s*, então *s* ← *s'*, isto é, a nova solução corrente passa a ser: *s* = (00111)

Lista tabu = *T* = {3} (indicando que o bit da terceira posição não pode ser modificado, a não ser que o critério de aspiração seja satisfeito)

Melhor solução até então: *s** = (00111) e *f*(*s**) = 11 (pois *f*(*s'*) > *f*(*s**))

Iter = 3; *MelhorIter* = 3;

Como (*Iter* – *MelhorIter*) = (3 – 3) = 0 ≤ *BTmax* = 1, então o procedimento de exploração do espaço de soluções continua.

***Passo 4**: Determinemos, agora, o melhor vizinho de s = (00111)*

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}0111)^t$	24	13	-2
$(0\mathbf{1}111)^t$	25	13	-17
$(00\mathbf{0}11)^t$	15	8	8
$(001\mathbf{0}1)^t$	13	7	7
$(0011\mathbf{0})^t$	16	7	7

Observe que o vizinho com o melhor valor para a função de avaliação é *s'* = (00011), com *f*(*s'*) = 8, mas esta solução é tabu, uma vez que o bit da terceira posição está na lista tabu. Como o critério de aspiração desta solução não é satisfeito, pois *f*(*s'*) = 8 ≤ *f*(*s*¤) = 11, esta solução não é aceita. Desta forma, considera-se o **melhor vizinho não tabu**, a saber:

Melhor vizinho: *s'* = (00101), com *f*(*s'*) = 7 (Critério de Desempate)

Como *s'* é o melhor vizinho de *s* (mesmo sendo de piora), então *s* ← *s'*, isto é, a nova solução corrente passa a ser: *s* = (00101)

Lista tabu = *T* = {4} (indicando que o bit da quarta posição não pode ser modificado, a não ser que o critério de aspiração seja satisfeito)

Melhor solução até então: *s** = (00111) e *f*(*s**) = 11

Iter = 4; *MelhorIter* = 3;

Como (*Iter – MelhorIter)* = (4 – 3) = 1 ≤ *BTmax* = 1, então prossegue a busca.

Passo 5: *Determinemos, agora, o melhor vizinho de s = (00101)*

Vizinhos de s	Peso dos vizinhos de s	Benefício dos vizinhos de s	$f(s')$
$(\mathbf{1}0101)^t$	17	9	9
$(0\mathbf{1}101)^t$	18	9	9
$(00\mathbf{0}01)^t$	6	4	4
$(001\mathbf{1}1)^t$	23	11	11
$(0010\mathbf{0})^t$	7	3	3

Observe que o vizinho com o melhor valor para a função de avaliação é s' = (00111), com $f(s')$ = 11. Entretanto, esta solução é tabu, uma vez que o bit da quarta posição está na lista tabu. Como o critério de aspiração desta solução não é satisfeito, pois $f(s')$ = 11 ≤ $f(s^*)$ = 11, esta solução não é aceita. Desta forma, considera-se o melhor vizinho não tabu, a saber (já aplicado um critério de desempate):

Melhor vizinho: s' = (10101), com $f(s')$ = 9

Desta forma, a nova solução corrente passa a ser: s = (10101), com $f(s)$ = 9

Lista tabu = T = {1} (indicando que o bit da primeira posição não pode ser modificado, a não ser que o critério de aspiração seja satisfeito)

Melhor solução até então: s^* = (00111) e $f(s^*)$ = 11

Iter = 5; *MelhorIter* = 3;

Como (*Iter* – *MelhorIter*) = (5 – 3) = 2 > *BTmax* = 1, então **PARE**. O método de Busca Tabu retorna, então, ***s** = (00111)*** como solução final, com valor ***f(s*) = 11***.

Prescrições especiais para a Busca Tabu

- Lista tabu dinâmica:
  - Tamanho variável no intervalo [t_{min}, t_{max}]
  - Tamanho deve ser mudado periodicamente (p.ex., a cada $2t_{max}$ iterações)
  - Objetivo: Se há ciclagem com um determinado tamanho, mudando-se o tamanho, muda-se a quantidade de movimentos tabu e possivelmente a seqüência de soluções geradas e conseqüentemente, diminui-se a probabilidade de ciclagem

- Passagem por regiões planas
  - Aumentar o tamanho da lista enquanto estiver na região plana
  - Retornar ao tamanho original quando houver mudança no valor da função de avaliação

Intensificação em Busca Tabu

- É comum em métodos de Busca Tabu incluir estratégias de intensificação, as quais têm por objetivo concentrar a pesquisa em determinadas regiões consideradas promissoras.

- Uma estratégia típica é retornar à uma solução já visitada para explorar sua vizinhança de forma mais efetiva.

- Outra estratégia consiste em incorporar atributos das melhores soluções já encontradas durante o progresso da pesquisa e estimular componentes dessas soluções a tornar parte da solução corrente.
  - são consideradas livres no procedimento de busca local apenas as componentes não associadas às boas soluções, permanecendo as demais componentes fixas.
  - um procedimento de intensificação com estas características é o Path-Relinking (Reconexão por Caminhos).

Diversificação em Busca Tabu

- Métodos baseados em Busca Tabu incluem, também, estratégias de diversificação. O objetivo dessas estratégias, que tipicamente utilizam uma memória de longo prazo, é redirecionar a pesquisa para regiões ainda não suficientemente exploradas do espaço de soluções.

- Estas estratégias procuram, ao contrário das estratégias de intensificação, gerar soluções que têm atributos significativamente diferentes daqueles encontrados nas melhores soluções obtidas.

- A diversificação, em geral, é utilizada somente em determinadas situações, como, por exemplo, quando dada uma solução *s*, não existem movimentos *m* de melhora para ela, indicando que o algoritmo já exauriu a análise naquela região.

Path-Relinking

- Estratégia de intensificação para explorar trajetórias que conectavam soluções elite (Glover, 1996)

- Originalmente proposta no contexto de Busca Tabu ou *Scatter Search*.

- Essa busca por soluções de melhor qualidade consiste em gerar e explorar caminhos no espaço de soluções partindo de uma ou mais soluções elite e levando a outras soluções elite.
  - são selecionados movimentos que introduzem atributos das soluções guia na solução corrente.
  - estratégia que objetiva incorporar atributos de soluções de boa qualidade, favorecendo a seleção de movimentos que as contenham.

Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

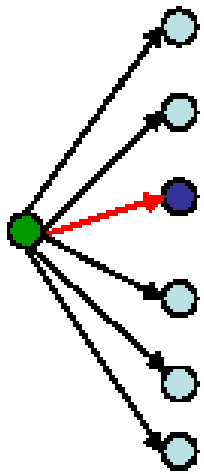
- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:

Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:

solução inicial



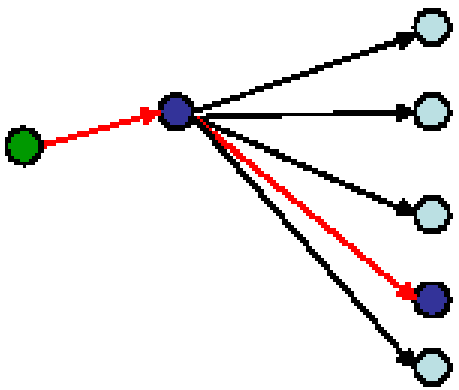
solução guia

Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:

solução inicial

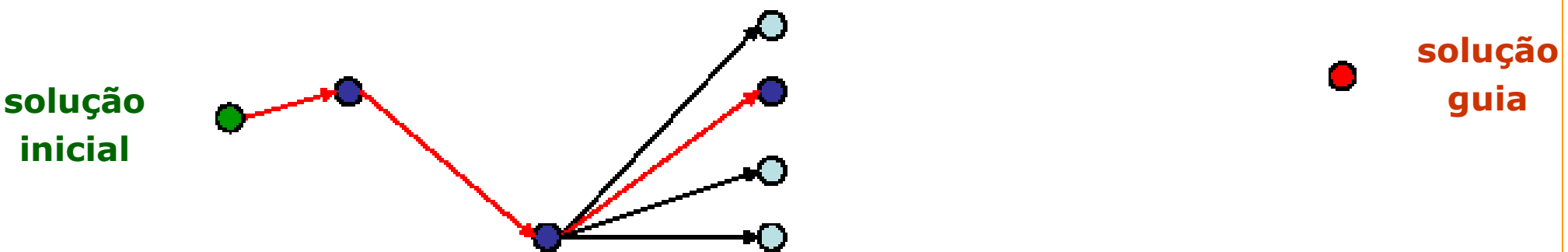


solução guia

Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

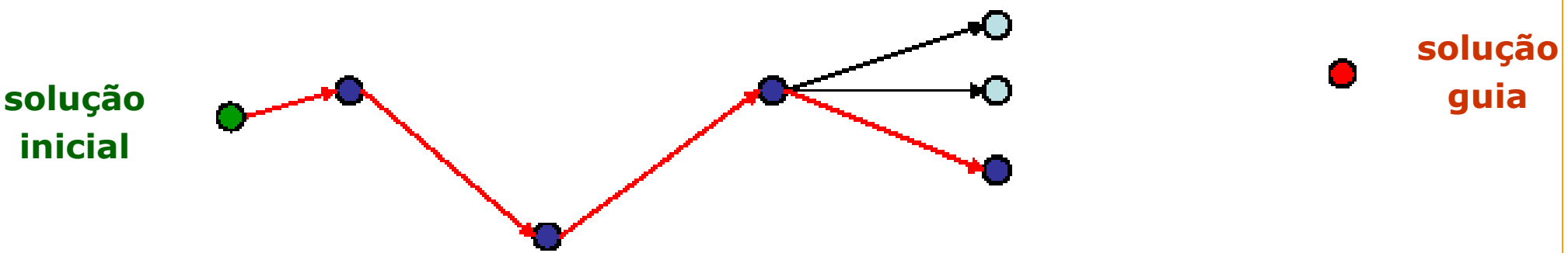
- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:



Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

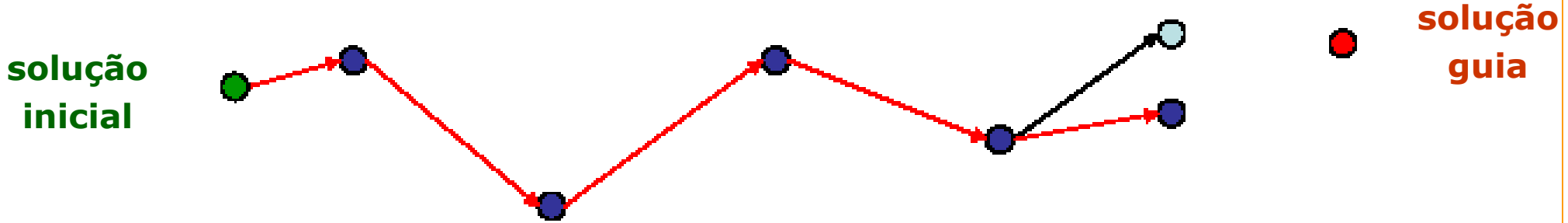
- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:



Path-Relinking

- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

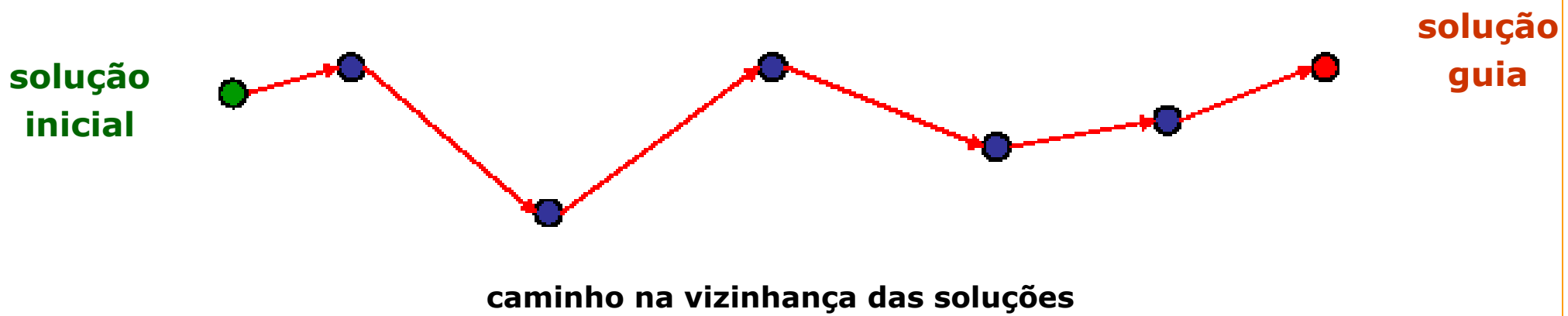
- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:



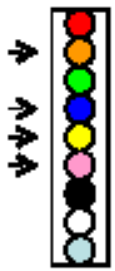
Path-Relinking

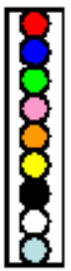
- Caminho é gerado selecionando movimentos que introduzam na solução inicial atributos da solução guia.

- A cada passo, todos os movimentos que incorporam atributos da solução guia são avaliados e o **melhor** movimento é selecionado:



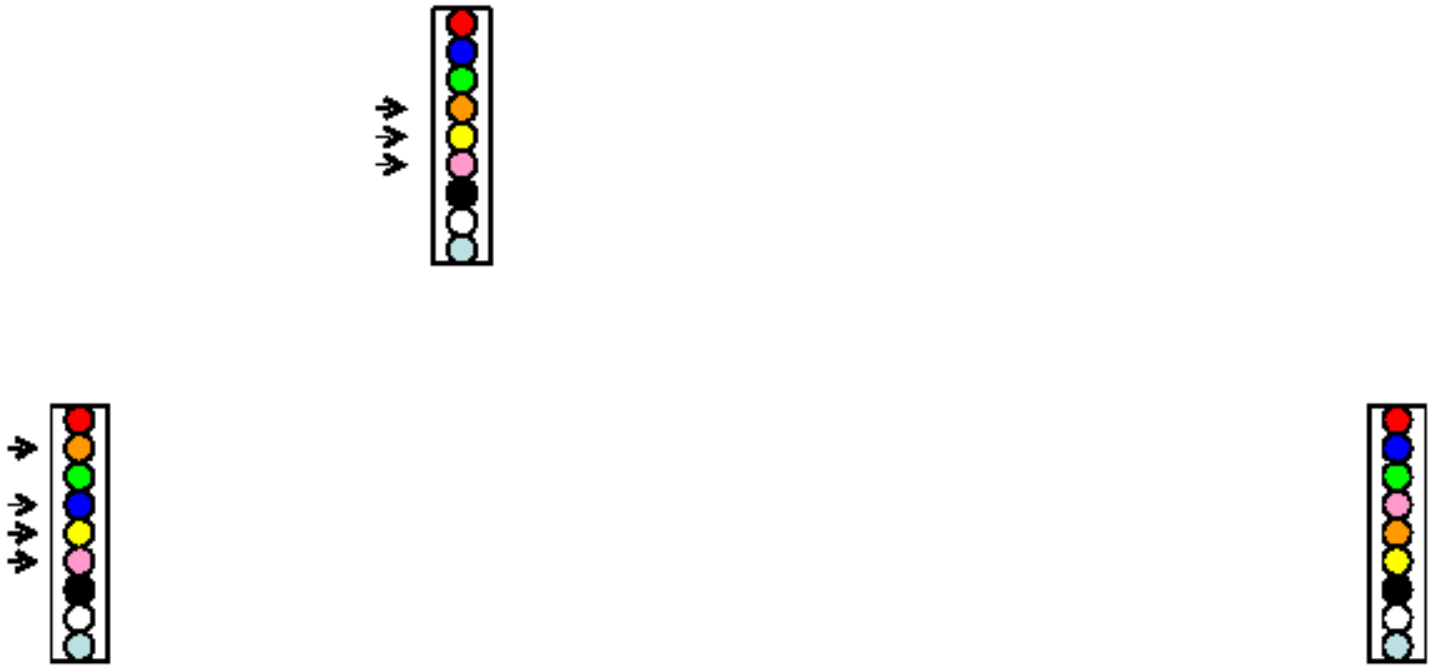
Path-Relinking



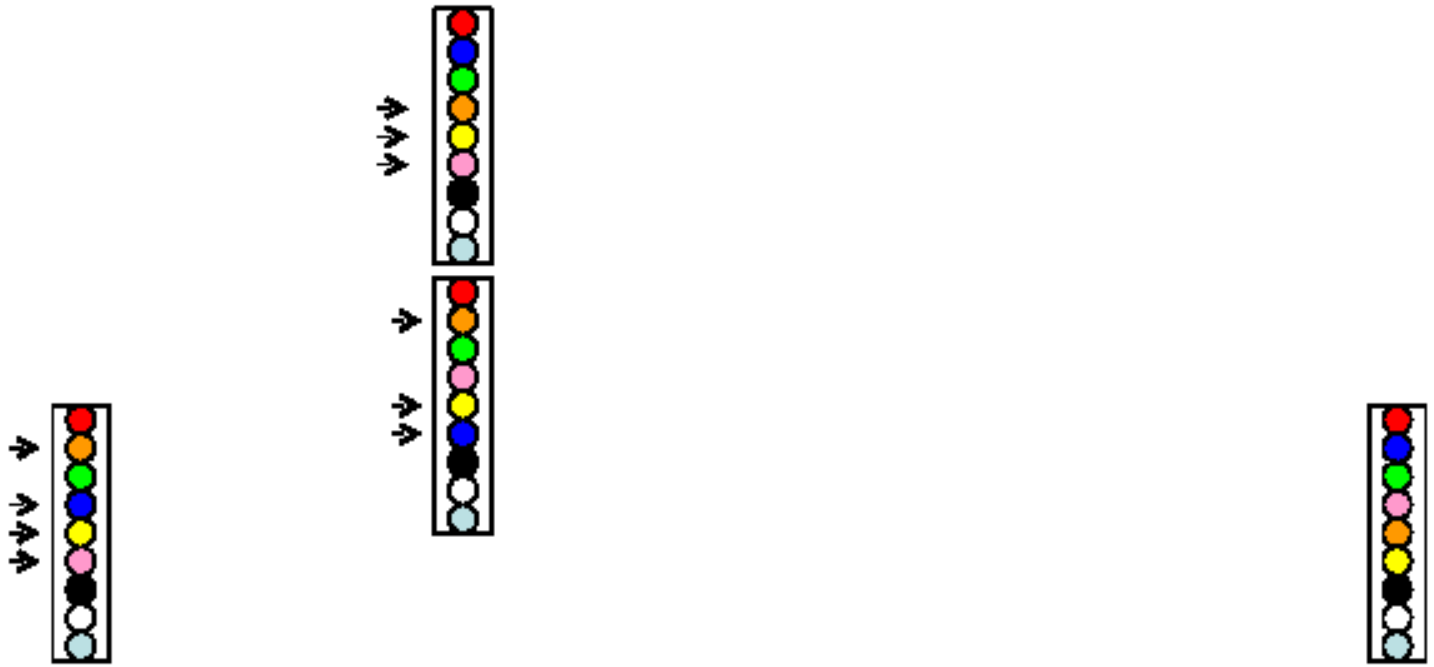


Exemplo retirado de uma palestra do prof. Dr. Maurício G. C. Resende
XXXVI SBPO, São João Del Rey – MG, 23 a 26 de novembro de 2004

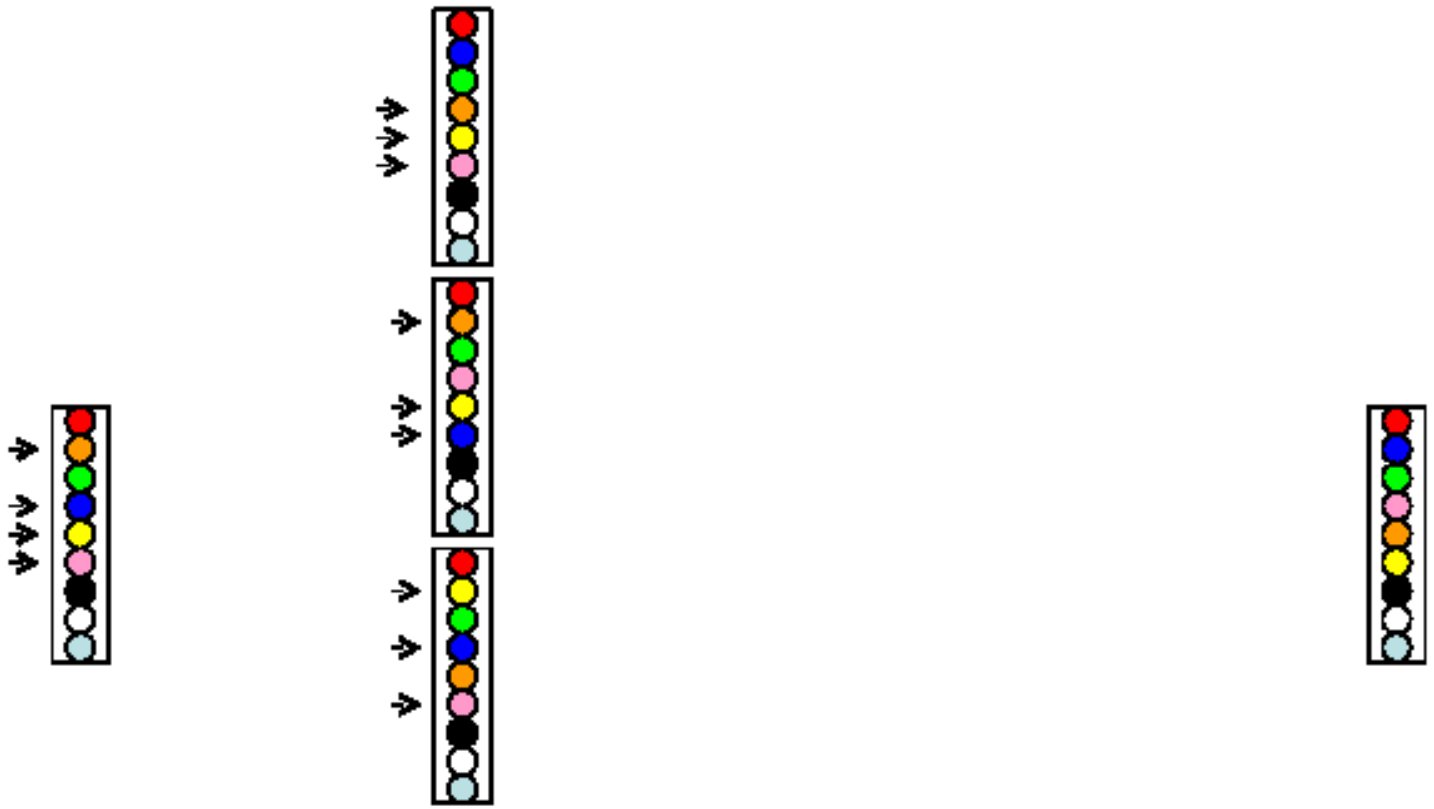
Path-Relinking



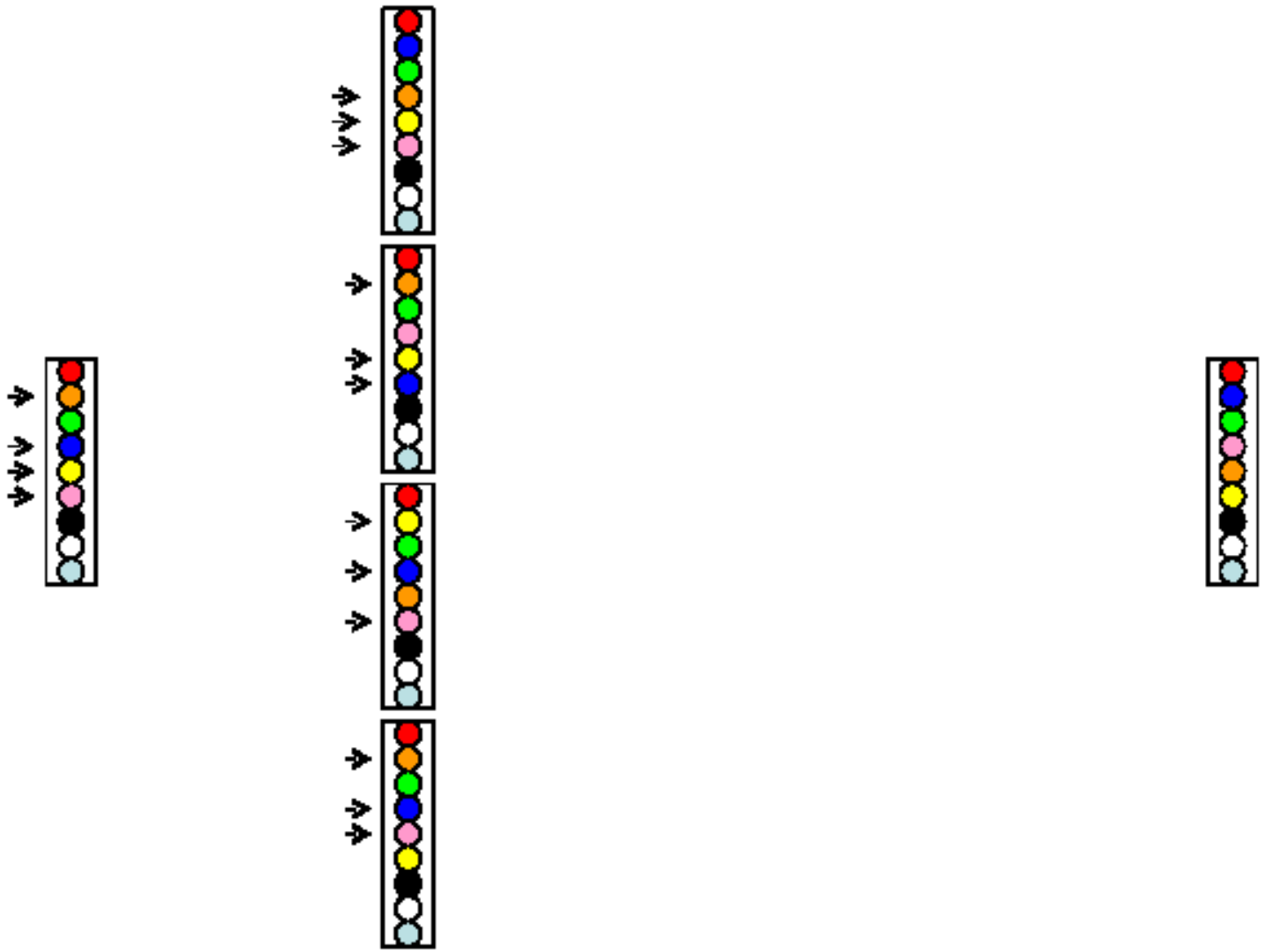
Path-Relinking



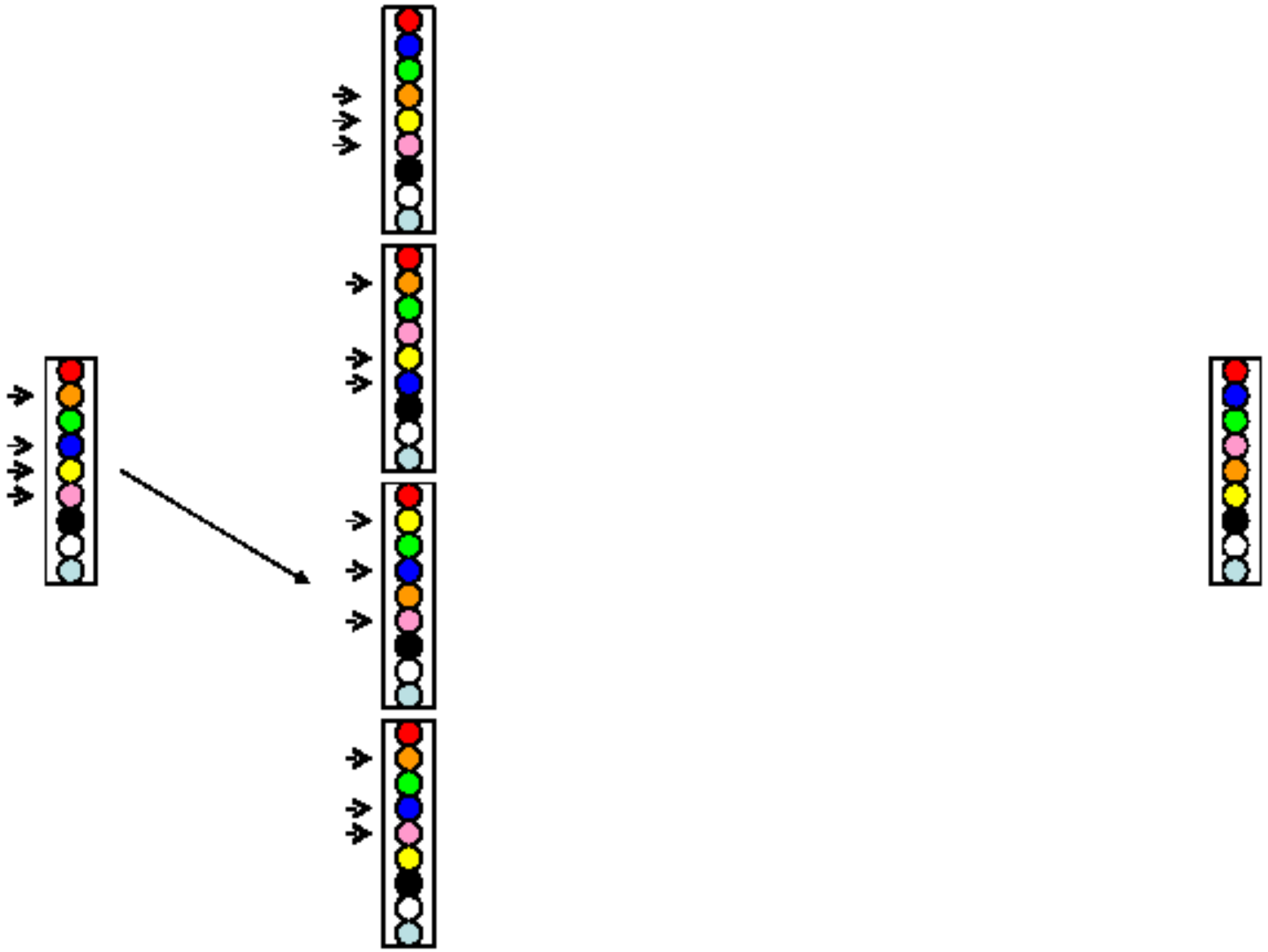
Path-Relinking



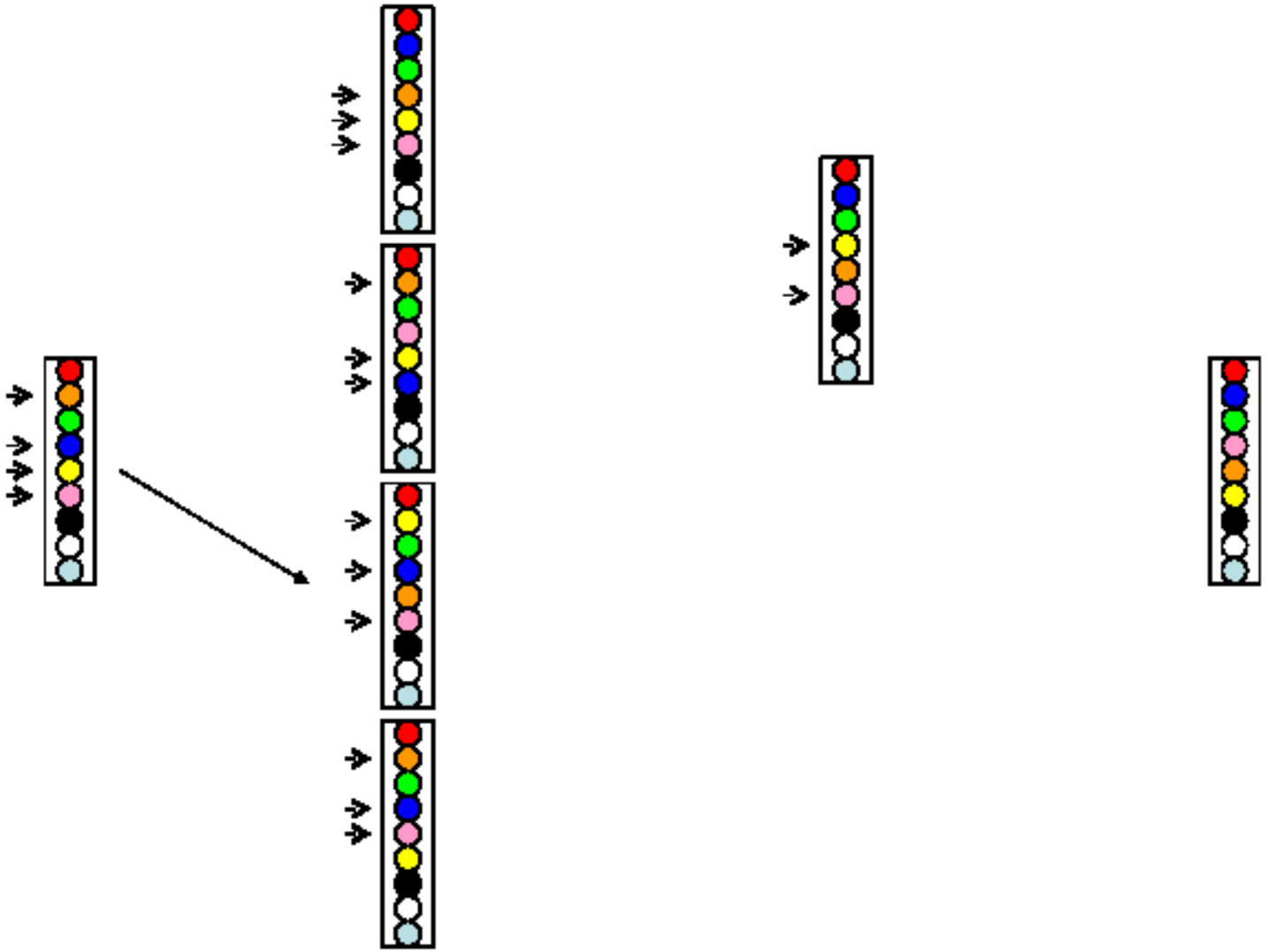
Path-Relinking



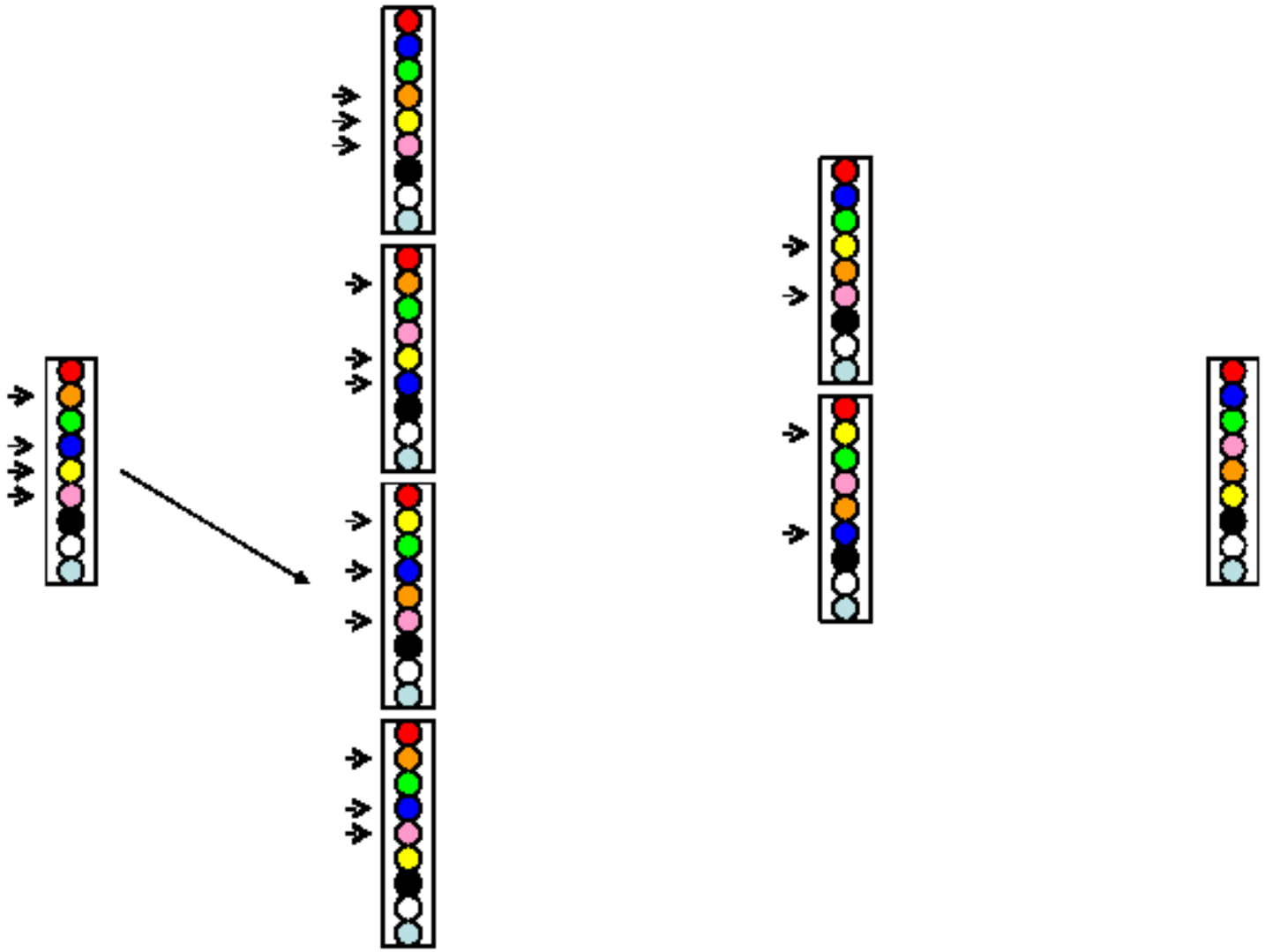
Path-Relinking



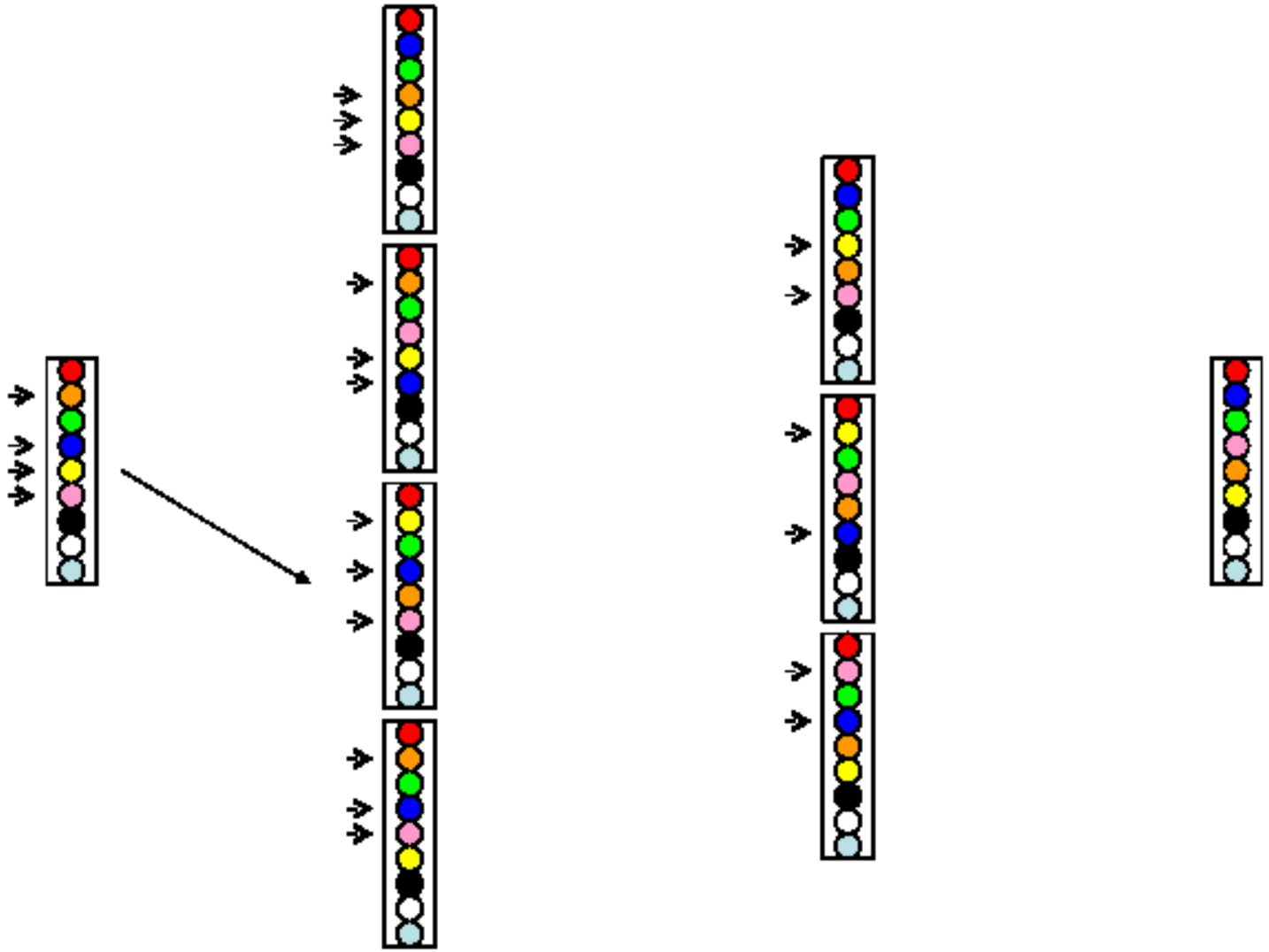
Path-Relinking



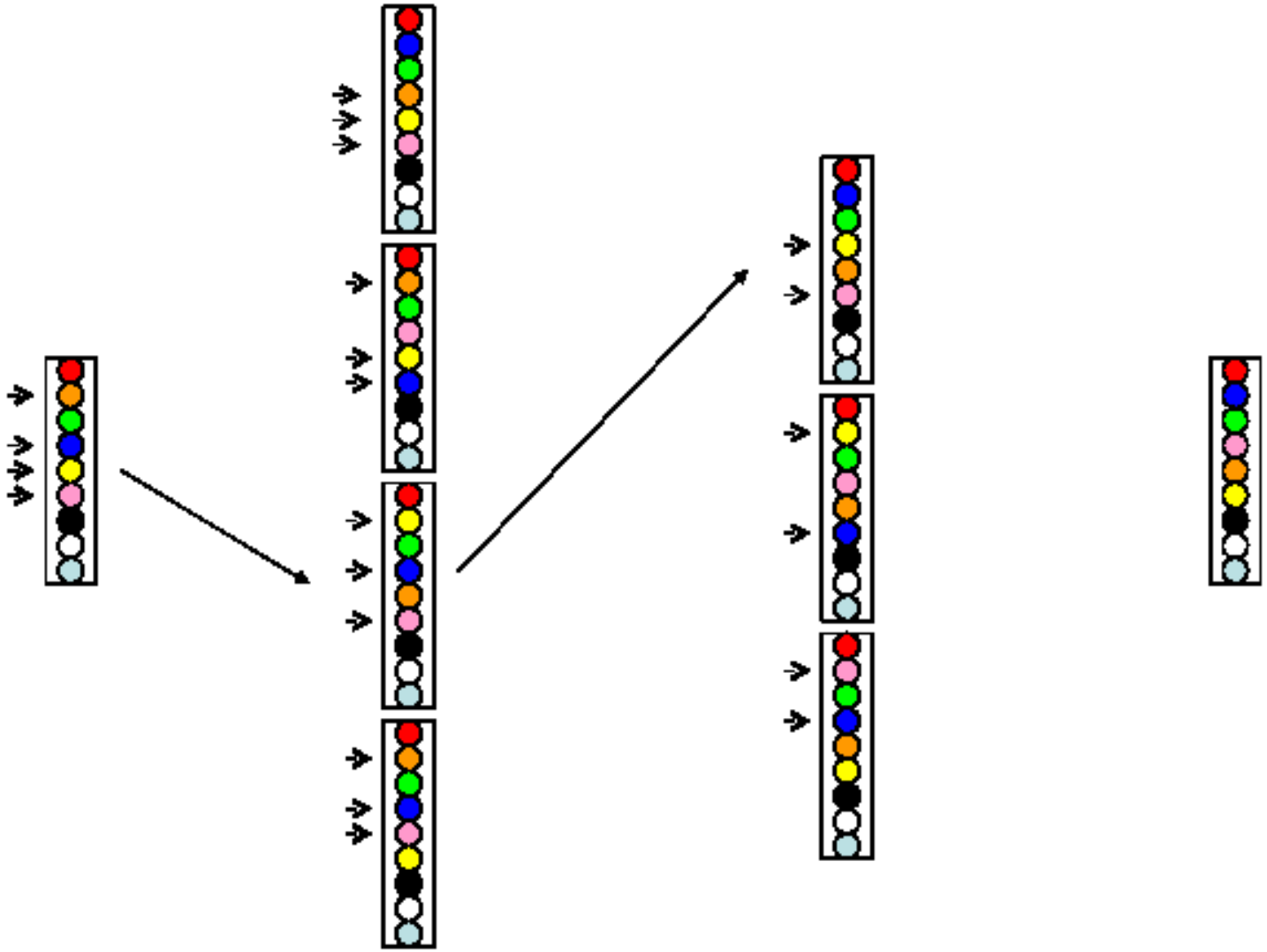
Path-Relinking



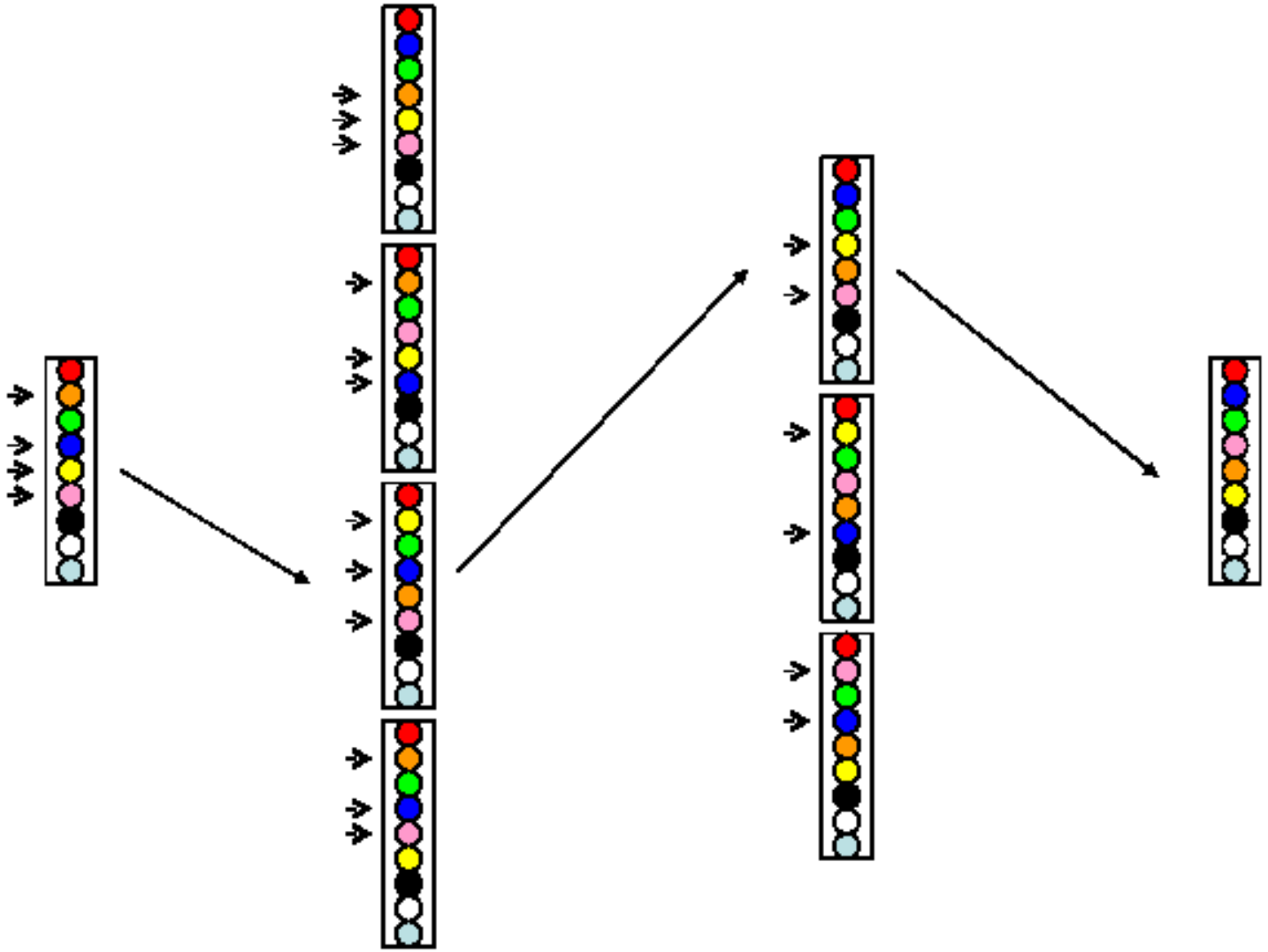
Path-Relinking



Path-Relinking



Path-Relinking



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).