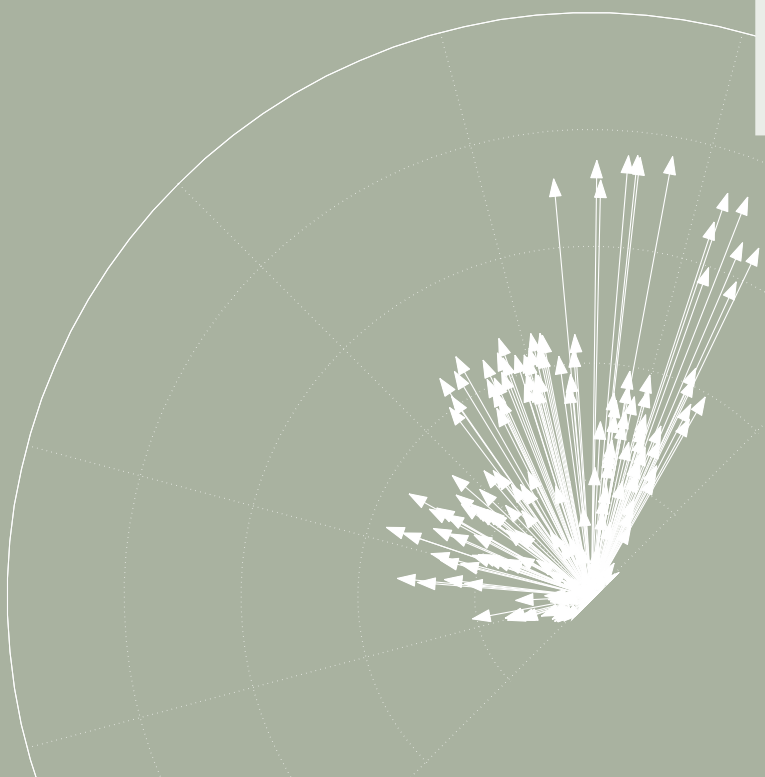
# MANUAL DE COMPUTAÇÃO EVOLUTIVA E METAHEURÍSTICA

ANTÓNIO GASPAR-CUNHA
RICARDO TAKAHASHI
CARLOS HENGGELER ANTUNES
COORDENADORES



IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDITORA ufmg

CAPÍTULO 10

# GRASP: Busca Gulosa Aleatorizada e Adaptativa

*Mauricio G. C. Resende* [*] *Geraldo R. Mateus* [**] *Ricardo M. A. Silva* [***]

[*] *Algorithms and Optimization Res. Dep.*
*AT&T Labs Research, USA*

[**] *Departamento de Ciência da Computação*
*Universidade Federal de Minas Gerais*

[***] *Departamento de Ciência da Computação*
*Universidade Federal de Lavras*

*Busca gulosa aleatorizada adaptativa*[1] é uma meta-heurística multi-partida que aplica o método de busca local repetidamente a partir de soluções construídas por um algoritmo guloso aleatório. Este capítulo apresenta os componentes da meta-heurística GRASP, juntamente com hibridizações com o método de religamento de caminhos[2].

[1] GRASP, do inglês *Greedy Randomized Adaptive Search Procedure*.
[2] Do inglês *path relinking*.

## 1. Introdução

Um problema de otimização combinatória pode ser definido por um conjunto finito $E = \{1, \ldots, n\}$, um conjunto de soluções viáveis $F \subseteq 2^E$ e uma função objetivo $f : 2^E \to \mathbb{R}$, todos definidos para cada problema específico. Neste capítulo consideraremos a versão de minimização do problema, na qual o objetivo consiste em encontrar uma solução ótima $S^* \in F$ tal que $f(S^*) \leq f(S)$, $\forall S \in F$. Dentre os diversos problemas reais e práticos de otimização combinatória podemos citar as mais distintas aplicações, tais como roteamento, escalonamento, planejamento da produção e inventário, localização de instalações, biologia computacional, entre outras.

Embora muito progresso venha sendo alcançado na tarefa de encontrar comprovadas soluções ótimas em problemas de otimização combinatória, empregando técnicas tais como *branch and bound*, planos de corte e programação dinâmica, assim como soluções aproximadas via algoritmos aproximativos, entretanto muitos problemas de otimização combinatória que ocorrem na prática se beneficiam de métodos heurísticos que rapidamente produzem soluções de boa qualidade sem necessariamente garantir a otimalidade. Neste sentido, muitas heurísticas modernas para otimização combinatória são baseadas nos *guidelines* de meta-heurísticas, tais como: algoritmos genéticos, recozimento simulado (*simulated annealing*), busca tabu (*tabu search*), pesquisa em vizinhança variável (*variable neighborhood search*), *scatter search*, religamento de caminhos (*path relinking*), pesquisa local iterativa (*iterated local search*), colônia de formigas (*ant colony optimization*), otimização por enxame de partículas (*particle swarm optimization*) e procedimento de busca guloso aleatorizado e adaptativo (GRASP).

Este capítulo tem por objetivo apresentar os componentes estruturais básicos do GRASP, incluindo diversos esquemas construtivos, métodos de busca local e hibridizações com religamento de caminhos. Os esquemas de busca local são apresentados na Seção 2. Em seguida, a meta-heurística GRASP é revista na Seção 3. Na Seção 4, oito variantes do método construtivo guloso aleatório são descritas. Por fim, as hibridizações de GRASP com religamento de caminhos são explanadas na Seção 5, seguidas pelas conclusões na Seção 6.

## 2. Esquemas de Busca Local

Considere o grafo $\mathcal{X}(S, M)$ correspondente ao *espaço de soluções vizinhas*, onde o conjunto de vértices $S$ representa todas as soluções viáveis de um problema de otimização combinatória e o conjunto de arestas $M$, os *movimentos* conectando soluções vizinhas. Uma solução $s \in S$ pertence à vizinhança $N(t)$ de uma solução $t \in S$ se $s$ pode ser obtida a partir de $t$ através de pequenas alterações pré-definidas empregadas em $t$. Se a solução $s \in N(t)$, então $t \in N(s)$, $(s, t) \in M$ e $(t, s) \in M$. Por último, diferentes estruturas de vizinhança podem ser definidas para cada problema de otimização combinatória.

Os esquemas de busca local partem de uma solução inicial, por sua vez tida como sendo a solução corrente, a procura de uma solução melhor na vizinhança da solução corrente. Se esta solução existir, automaticamente tornar-se-á a nova solução corrente e a busca local recursivamente será aplicada sobre esta solução. O procedimento termina quando nenhuma solução melhor do que a corrente existir na vizinhança desta última. Tal esquema de busca local exige que o tamanho da vizinhança seja tal que sua exploração possa ser feita eficientemente.

O Algoritmo 1 mostra o pseudo-código para o algoritmo de busca local padrão. A busca local começa em algum vértice $s_0 \in S$ tornando-o a solução atual $t$, ou seja, $t = s_0$. A cada iteração, o algoritmo procura uma solução vizinha que possua um valor da função objetivo melhor do que o valor correspondente à solução corrente, ou seja, uma solução $s \in N(t)$ tal que $f(s) < f(t)$. Se tal solução existir, tornar-se-á a solução corrente, ou seja, $t = s$. Estas iterações são repetidas até não existir nenhuma solução melhor do que a solução corrente $t$ na vizinhança $N(t)$. Neste caso, a solução $t$ é

denominada como um *ótimo local*.

A busca local pode ser vista como partindo de um vértice $s \in S$ e examinado os nós adjacentes no grafo $\mathcal{X}$ em busca de uma solução de melhoria. Na variante *first-improving*, a busca local se move para a primeira solução de melhoria encontrada na vizinhança, enquanto na variante *best-improving* todas as soluções da vizinhança são avaliadas de modo que a busca local se direcione para a melhor solução encontrada. Podemos considerar esta fase da busca como sendo a de *intensificação*, uma vez que apenas uma pequena parte do espaço de soluções está sendo considerada.

Vários enfoques foram propostos para estender a busca local acima citada, tais como descida em vizinhança variável (*variable neighborhood descent*) (Andrade e Resende, 2006; Hansen e Mladenović, 1999; Martins et al., 1999; Ribeiro e Souza, 2002; Ribeiro et al., 2002; Ribeiro e Vianna, 2005), busca em vizinhança variável (*variable neighborhood search*) (Beltrán et al., 2004; Canuto et al., 2001; Drummond et al., 2002; Festa et al., 2002; Hansen e Mladenović, 1999; Ochi et al., 2001), busca tabu de memória curta (Abdinnour-Helm e Hadley, 2000; de Souza et al., 2003; Delmaire et al., 1999; Glover, 1989, 1990; Laguna e González-Velarde, 1991; Li, Guo, Wang e Lim, 2004; Lim e Wang, 2004; Moura e Oliveira, 2005; Pu et al., 2006; Serra e Colomé, 2001; Souza et al., 2001), recozimento simulado (de la Penã, 2004; Kirkpatrick et al., 1983; Liu et al., 2000), pesquisa local iterativa (Baum, 1986a,b; Baxter, 1981; Johnson, 1990; Martin e Otto, 1996; Martin et al., 1991a; Ribeiro e Urrutia, 2007), e busca em vizinhança de muito larga escala(*very-large scale neighborhood search*) (Ahuja et al., 2001, 2002; Geng et al., 2005), os quais admitem, além da exploração da vizinhança da solução corrente, desde movimentos para soluções de custo maior do que a solução corrente, passando pelo cenário de múltiplas vizinhanças, até a exploração de vizinhanças bem maiores.

**Algoritmo 1** Busca local padrão

1: Entrada: $s_0 \in S$
2: Resultado: Ótimo local
3: $t \leftarrow s_0$
4: **enquanto** existe $s \in N(t)$ tal que $f(s) < f(t)$ **faça**
5: $\quad t \leftarrow s$
6: **fim enquanto**
7: **retornar** $t$

## 3. Procedimentos de Busca Gulosos Aleatorizados Adaptativos

Um método de busca eficaz precisa viabilizar sua diversificação. Portanto, uma boa estratégia para a busca local não deve se ater a uma região particular do espaço de soluções, como por exemplo, em torno de uma solução construída por um algoritmo guloso. Uma alternativa consiste em iniciar uma busca local a partir de várias soluções geradas aleatoriamente, com a expectativa de que haja um caminho de melhoria de custo partindo de uma destas soluções até uma solução ótima ou próxima do ótimo.

Um procedimento de busca guloso, aleatório e adaptativo repetidamente aplica a busca local a partir de soluções construídas com um algoritmo guloso aleatório. O melhor ótimo local dentre todas as buscas locais é retornado como solução da heurística. O Algoritmo 2 apresenta o pseudo-código de um GRASP genérico. Ao fim da fase de construção gulosa aleatória, a solução obtida é viável para diversas instâncias da heurística. Entretanto, para alguns problemas, a solução pode ser inviável, logo requerendo a aplicação de um procedimento de reparo que restaure sua viabilidade. Exemplos de implementações GRASP munidas de procedimentos de reparos para restaurar a viabilidade de soluções podem ser encontrados em Duarte, Ribeiro e Urrutia (Duarte et al., 2007), Duarte et. al (Duarte et al., 2006), Nascimento, Resende e Toledo (Nascimento et al., 2010) e Mateus, Resende e

Silva (Mateus et al., 2009).

Uma característica especialmente atraente da meta-heurística GRASP é a facilidade com que pode ser implementada. Dado que poucos parâmetros precisam ser inicializados e ajustados, logo o desenvolvimento pode se ater à implementação dos algoritmos e estruturas de dados que garantam a eficiência do projeto. Como veremos a seguir, as implementações básicas do GRASP repousam exclusivamente em dois parâmetros. O primeiro controla o número de iterações dos métodos construtivo e de busca local a serem aplicados. O segundo controla a mistura dos enfoques aleatório e guloso do método construtivo. A despeito de sua simplicidade e facilidade de implementação, GRASP é uma meta-heurística muito eficaz e produz as melhores soluções conhecidas para muito problemas.

GRASP foi introduzido pela primeira vez por Feo e Resende (Feo e Resende, 1989). Ver Feo e Resende (Feo e Resende, 1995), Pitsoulis e Resende (Pitsoulis e Resende, 2002), Resende e Ribeiro (Resende e Ribeiro, 2003) e Resende (Resende, 2008) para um revisão da meta-heurística GRASP e Festa e Resende (Festa e Resende, 2002, 2009a,b) para uma bibliografia anotada do assunto.

**Algoritmo 2** Pseudo-código de um GRASP básico.

```
1: Entrada: Critério de parada
2: Resultado: Solução x*.
3: x* ← ∞
4: enquanto critério de parada não for satisfeito faça
5:    x ← GulosoAleatorio(·)
6:    se x não for viável então
7:       x ← reparo(x)
8:    fim se
9:    x ← BuscaLocal(x)
10:   se f(x) < f(x*) então
11:      x* ← x
12:   fim se
13: fim enquanto
14: retornar x*
```

## 4. Método de Construção Guloso Aleatório

Os métodos de construção gulosos aleatórios do GRASP visam produzir um conjunto diversificado de soluções iniciais de boa qualidade para a busca local. Este propósito é alcançado pela adição de aleatoriedade ao algoritmo guloso. São ilustrados aqui oito formas de realizar tal tarefa, nas quais as soluções são construídas pela adição de elementos a uma solução parcial.

### Construção semi-gulosa

O Algoritmo 3 mostra o pseudo-código para este método de construção. A cada iteração do primeiro esquema de construção, denominado *algoritmo semi-guloso* por Hart e Shogan (Hart e Shogan, 1987), seja $C$ o conjunto de todos os elementos que podem ser adicionados à solução parcial e seja $LRC(C)$ a lista restrita de candidatos composta por elementos candidatos de alta-qualidade. A qualidade de um elemento candidato é determinada por sua contribuição, naquele ponto, para o custo da solução sendo construída. A função gulosa $g(c)$ mede esta contribuição para cada candidato $c \in C$. Membros do conjunto $LRC(C)$ podem ser determinados pelo *rank* ou pela qualidade relativa aos outros candidatos. Membros segundo o *rank*, também denominados *baseados na cardinalidade*, são obtidos se o candidato é um dos $q$ candidatos com o menor valor da função gulosa, onde $q$ é um parâmetro de entrada que

determina quão gulosa ou aleatória será a construção. Membros *baseados na qualidade* relativa a outros candidatos determinam um valor de corte para a função gulosa e somente consideram candidatos com uma valor guloso que não seja maior do que este valor de corte. Neste caso, a implementação utiliza um parâmetro real $\alpha \in [0,1]$. Seja $\acute{g} = \max\{g(c) \mid c \in C\}$ e $\grave{g} = \min\{g(c) \mid c \in C\}$. Um candidato $c \in C$ é posto na $LRC$ se e somente se $\grave{g} \leq g(c) \leq \grave{g} + \alpha \cdot (\acute{g} - \grave{g})$. O parâmetro de entrada $\alpha$ determina quão gulosa ou aleatória será a construção. Um dentre os candidatos em $LRC(C)$ é selecionado aleatoriamente e inserido na solução parcial. A construção é repetida até não haver mais candidatos.

**Algoritmo 3** Método construtivo da meta-heurística GRASP: aleatorização do algoritmo guloso ou semi-guloso.

1: Entrada: $E = \{$ conjunto discreto finito $\}$.
2: Resultado: Solução $S$.
3: $S \leftarrow \emptyset$; $C \leftarrow E$
4: **enquanto** $|C| > 0$ **faça**
5: Para todo $c \in C$ computar o valor da função gulosa $g(c)$
6: Definir $LRC(C) \leftarrow \{c \in C \mid g(c)$ tam um valor reduzido$\}$
7: Selecionar aleatoriamente $c^* \in LRC(C)$
8: Adicionar $c^*$ à solução parcial: $S \leftarrow S \cup \{c^*\}$
9: Seja $C$ o conjunto de elementos que podem ser adicionados à $S$
10: **fim enquanto**
11: **retornar** $S$

### Construção por amostragem gulosa

Neste segundo esquema de construção, denominado *construção por amostragem gulosa* por Resende e Werneck (Resende e Werneck, 2004), um algoritmo guloso é aplicado a cada amostragem aleatória de candidatos, ao invés da aleatorização do algoritmo guloso. O Algoritmo 4 mostra o pseudo-código para este procedimento construtivo. A cada passo um subconjunto de tamanho fixo de candidatos em $C$ é amostrada e a contribuição incremental ao custo da solução parcial é computada para cada elemento amostrado. Um elemento com a melhor contribuição incremental é selecionado e adicionado à solução parcial. Esse processo é repetido até quando não existir nenhum candidato.

**Algoritmo 4** Construção gulosa aleatória da meta-heurística GRASP: amostragem gulosa

1: Entrada: $p, E = \{$ conjunto discreto finito $\}$.
2: Resultado: Solução $S$.
3: $S \leftarrow \emptyset$; $C \leftarrow E$
4: **enquanto** $|C| > 0$ **faça**
5: Amostrar aleatoriamente $\min\{p, |C|\}$ elementos
6: a partir de $C$ e inserí-los em $LRC(C)$
7: Selecionar $c^* = \text{argmin}\{g(c) \mid c \in LRC(C)\}$
8: Adicionar $c^*$ à solução parcial: $S \leftarrow S \cup \{c^*\}$
9: Seja $C$ o conjunto de elementos que podem ser adicionados à $S$
10: **fim enquanto**
11: **retornar** $S$;

### Construção aleatória-gulosa

Uma possível restrição da construção gulosa aleatória baseada em uma lista restrita de candidatos é sua complexidade. A cada passo do método construtivo, cada elemento candidato ainda não selecionado tem que ser avaliado pela função gulosa. Nos casos onde a diferença entre o número de elementos do conjunto $E$ e o número de elementos que compõem uma solução do problema é alto, o método construtivo pode não ser muito eficiente. Como ocorre no caso do esquema de construção por amostragem gulosa, este terceiro esquema introduzido por Resende e Werneck (Resende e Werneck, 2004), denominado *construção aleatório guloso*, também atende esta restrição.

Neste esquema, uma parte da solução é construída pela escolha aleatória de $p$ elementos candidatos e o restante da solução é completado de forma gulosa. A solução resultante é gulosa e aleatória. O valor de $p$ determina quão guloso ou aleatório é o método construtivo. Valores reduzidos de $p$ levam à soluções mais gulosas, enquanto valores maiores levam à soluções mais aleatórios. A amostragem aleatória foi apresentada na Subseção 4.

### Construção gulosa proporcional

A cada iteração deste quarto esquema de construção, denominado *guloso proporcional* e introduzido pela primeira vez por Resende e Werneck (Resende e Werneck, 2004), a função gulosa $g(c)$ para todo elemento candidato $c \in C$ é computada e em seguida um candidato é selecionado aleatoriamente, porém de um modo viciado: a probabilidade de um dado candidato $c' \in C$ ser selecionado é inversamente proporcional a $g(c') - \min\{g(c)|c \in C\}$.

### Construção GRASP reativa

O quinto esquema construtivo é denominado *GRASP reativo*. Nos procedimentos de construção guloso aleatório, o projetista deve decidir como balancear os enfoques guloso e aleatório. Uma simples alternativa seria balanceá-los aleatoriamente. Por exemplo, na construção semi-gulosa, com membros em $LRC$ segundo a qualidade relativa aos outros candidatos, o parâmetro $\alpha$ pode ser selecionado de modo uniformemente aleatório a partir do intervalo $[0, 1]$, de modo que cada iteração GRASP use um valor diferente de $\alpha$, resultando em um balanceamento diferente entre os enfoques guloso e aleatório.

Prais e Ribeiro (Prais e Ribeiro, 2000) mostraram que usando um valor fixo para o parâmetro $\alpha$ no esquema semi-guloso baseado na qualidade frequentemente retarda o encontro de soluções de alta qualidade, as quais eventualmente seriam encontradas caso outro valor para $\alpha$ fosse utilizado. Sendo assim, eles propuseram uma extensão do procedimento GRASP básico, denominado *GRASP reativo*, na qual o parâmetro $\alpha$ é selecionado aleatoriamente a cada iteração a partir de um conjunto discreto de possíveis valores. Cabe ressaltar que valores das soluções encontrados ao longo das iterações anteriores servem de guia para o processo de seleção.

Prais e Ribeiro (Prais e Ribeiro, 2000) definem $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ como o conjunto de valores possíveis para $\alpha$. As probabilidades associadas com a escolha de cada valor são inicialmente iguais a $p_i = 1/m,\ i = 1, \ldots, m$. Além disso, seja $z^*$ a solução titular (a melhor solução encontrada até aquele ponto) e $A_i$ o valor médio de todas soluções encontradas usando $\alpha = \alpha_i,\ i = 1, \ldots, m$. As probabilidades de seleção são periodicamente reavaliadas tomando $p_i = q_i / \sum_{j=1}^{m} q_j$, com $q_i = z^*/A_i$ para $i = 1, \ldots, m$. O valor de $q_i$ será maior para os valores de $\alpha = \alpha_i$ associados às melhores soluções em média. Logo, valores maiores de $q_i$ correspondem a valores mais adequados para o parâmetro $\alpha$. As probabilidades mais apropriadas a estes valores irão então aumentar quando forem reavaliadas. Cabe observar, que esta estratégia reativa não é limitada aos procedimentos semi-gulosos com elementos em $LRC$ baseados na qualidade relativa. Ou seja, ela pode ser estendida a outros esquemas de construção gulosos aleatório, todos os quais necessitam balancear o aspecto guloso com o aspecto aleatório.

### Construção com memória de longo prazo

O sexto esquema de construção, denominado *construção com memória de longo prazo*, introduz uma estrutura de memória de longo prazo. Embora o GRASP reativo também utilize memória de longo prazo (informações coletadas em iterações anteriores) para ajustar o balanceamento entre os aspectos gulosos e aleatórios, Fleurent e Glover (Fleurent e Glover, 1999) observaram que isto não acontecia no GRASP básico. Sendo assim, eles propuseram um esquema de longo-prazo para atender esta questão em heurísticas *multi-partida*. Memória de longo prazo é um dos fundamentos nos quais se apoia por exemplo a busca tabu (Glover, 1989, 1990). A idéia básica consiste em manter um conjunto de soluções elite a ser usado na fase de construção. Mais detalhes a respeito deste enfoque serão fornecidos no decorrer deste capítulo Por enquanto, tudo o que precisamos saber consiste em como uma solução tornar-se uma solução elite: esta deve ser melhor do que o melhor membro do conjunto elite ou melhor do que seu pior membro, contanto que seja suficientemente diferente das outras soluções do conjunto.

Fleurent e Glover (Fleurent e Glover, 1999) definem uma *variável fortemente determinada* como sendo aquela que não pode ser alterada sem comprometer o objetivo ou mudar significantemente outras variáveis; e uma *variável consistente* como a que recebe o mesmo valor em grande parte do conjunto elite. Seja $I(e)$ uma medida das características consistentes e fortemente determinadas do elemento $e$ da solução a partir do conjunto $E$. Como $I(e)$ torna-se maior na medida em que $e$ aparece mais frequentemente no conjunto de soluções elite, ele é usado na fase de construção da seguinte maneira. Seja $g(e)$ o valor da função gulosa para o candidato $e \in C$, ou seja o custo incremental associado com a incorporação do elemento $e \in C$ na solução em construção. Seja $K(e) = F(g(e), I(e))$ uma função associada aos aspectos gulosos e de intensificação, como por exemplo, $K(e) = \lambda g(e) + I(e)$. O esquema de intensificação que favoresse a seleção dos elementos $e \in C$ com um alto valor para $K(e)$ devem atualizar a probabilidade de seleção para o seguinte valor: $p(e) = K(e)/\sum_{s \in \mathrm{LRC}} K(s)$. Cabe observar que a função $K(e)$ pode variar com o tempo pela mudança dos valores $\lambda$, por exemplo, inicialmente $\lambda$ pode ser inicializado para um valor alto que decresce a medida que a diversifição é chamada.

### Construção por amostragem viciada

O sétimo esquema de construção foi introduzido por Bresina (Bresina, 1996) e denominado de *construção por amostragem viciada*. Consiste em outra forma de partir de uma seleção uniforme de elementos candidatos na construção de uma solução gulosa aleatória. Ao invés de escolher de modo uniformemente aleatório o próximo elemento candidato a ser adicionado à solução parcial, a construção por amostragem viciada sugere que qualquer distribuição de probabilidade possa ser usada para favorecer alguns tipos de candidatos em particular.

No mecanismo de construção proposto por Bresina (Bresina, 1996), uma família de distribuições de probabilidade é introduzida. Elas são baseadas na colocação (*rank*) $r[\sigma]$ atribuída a cada elemento candidato $\sigma$, de acordo com os valores de suas funções gulosas. O elemento com o menor valor da função gulosa tem *rank* 1, o segundo menor tem *rank* 2, e assim por diante. Várias funções viciadas $b(\cdot)$ foram introduzidas por Bresina, tais como, *vício aleatório* onde $b(r) = 1$, *vício linear* onde $b(r) = 1/r$, *vício logarítmico* onde $b(r) = \log^{-1}(r+1)$, *vício exponencial* onde $b(r) = e^{-r}$ e *vício polinomial* de ordem $n$ onde $b(r) = r^{-n}$. Após todos os elementos em $LRC$ receberem seus respectivos *ranks*, a probabilidade $\pi(\sigma)$ para selecionar o elemento $\sigma \in LRC$ pode ser computada da seguinte forma: $\pi(\sigma) = b(r[\sigma])/(\sum_{\sigma' \in LRC} b(r[\sigma']))$. Cabe observar que de acordo com este esquema, candidatos podem ser selecionados a partir do conjunto $LRC$ ou a partir do conjunto completo de candidatos, ou seja, $LRC = C$.

### Construção com pertubação de custos

No oitavo esquema de construção, denominado *construção com pertubação de custos*, pertubações aleatórias são introduzidas no problema original como uma forma de obter aleatoriedade. A idéia de introduzir ruído no custo original com o intuito de aleatorizar o processo construtivo é similar ao *noising method* de Charon e Hudry (Charon e Hudry, 1993, 2002).

Em circunstâncias onde os algoritmos construtivos são insensíveis às estratégias aleatórias padrão, tal como selecionar um elemento aleatoriamente a partir de uma lista restrita de candidatos, a construção com pertubação de custo pode ser mais eficaz do que a estratégia gulosa aleatória do GRASP básico. Ribeiro, Uchoa e Werneck (Ribeiro et al., 2002) mostram que este caso ocorre na heurística de caminho mínimo de Takahashi e Matsuyama (Takahashi e Matsuyama, 1980), usado como um dos principais componentes da fase construtiva de um GRASP híbrido proposto para um problema de Steiner em grafos.

Outra situação onde esta estratégia de pertubação dos custos pode ser eficiente ocorre quando não há um algoritmo guloso a ser aleatorizado, como foi o caso do GRASP híbrido desenvolvido por Canuto, Resende, e Ribeiro (Canuto et al., 2001) para o problema de Steiner com prêmios *prize-collecting Steiner tree problem*, o qual fez uso de um algoritmo de aproximação primal-dual de Goemans e Williamson (Goemans e Williamson, 1996) para construir soluções iniciais usando custos pertubados.

## 5. Hibridizações com Religamento de Caminhos

A heurítica de religamento de caminhos foi originalmente proposta por Glover (Glover, 1996) como uma estratégia de intensificação para explorar trajetórias conectando soluções elite obtidas pela busca tabu ou *scatter search* (Glover, 2000; Glover e Laguna, 1997; Glover et al., 2000). Partindo de uma ou mais soluções elite na busca por melhores soluções, caminhos que levam a outras soluções elite são gerados e explorados no grafo do espaço de soluções. Para gerar estes caminhos, movimentos são selecionados para introduzir atributos presentes na solução destino na solução corrente. Desta forma, o religamento de caminhos pode ser visto como uma estratégia que procura incorporar atributos de soluções de alta qualidade.

O pseudo-código do Algoritmo 5 ilustra o religamento de caminhos misto aplicado ao par de soluções $x_s$ e $x_t$. O religamento de caminhos misto (Ribeiro e Rosseti, 2002) alterna os papéis de solução origem e destino a cada movimento.

O religamento de caminhos misto começa computando as diferenças simétricas $\Delta(x_s, x_t)$ entre a solução origem $x_s$ e a destino $x_t$. Ou seja, o conjunto de movimentos necessários para alcançar $x_t$ a partir de $x_s$, e vice-versa. Portanto, dois caminhos de soluções são gerados, um partindo de $x_s$ e outro de $x_t$. Estes caminhos crescem até se encontrarem para formar um único caminho entre $x_t$ e $x_s$. A busca local é aplicada na melhor solução $x^*$ neste caminho e o mínimo local é retornado pelo algoritmo. Inicialmente, $x$ e $y$ são inicializados como $x_s$ e $x_t$, respectivamente. A cada passo, o procedimento examina todos os movimentos $m \in \Delta(x, y)$ partindo da solução corrente $x$ em direção às soluções que contêm um atributo adicional de $y$ e seleciona aquele que minimiza $f(x \oplus m)$, com $x \oplus m$ sendo a solução resultante da aplicação do movimento $m$ sobre a solução $x$. Em seguida, o melhor movimento $m^*$ é realizado produzindo a solução $x \oplus m^*$. Depois, se necessário, a melhor solução $x^*$ é atualizada. Por fim, os conjuntos de movimentos disponíveis são atualizados e os papéis de $x$ e $y$ invertidos. O procedimento termina quando $|\Delta(x, y)| = 1$.

Religamento de caminhos consiste em um grande avanço no procedimento GRASP padrão, levando a melhorias significantes de desempenho e qualidade das soluções. A hibridização entre religamento de caminhos e GRASP foi primeiro proposta por Laguna e Martí (Laguna e Martí, 1999) e posteriormente seguida por várias extensões, melhorias e aplicações de sucesso (Aiex et al., 2005; Canuto et al., 2001; Festa et al., 2006; Oliveira et al., 2004; Resende et al., 2010; Resende e Ribeiro, 2003; Resende e

**Algoritmo 5** Religamento de caminhos misto entre as soluções $x_s$ e $x_t$.

Entrada: $x_s \in S$, $x_t \in S$.
Resultado: Solução $x^*$.
Calcule as diferenças simétricas $\Delta(x_s, x_t)$ e $\Delta(x_t, x_s)$
$f^* \leftarrow \min\{f(x_s), f(x_t)\}$
$x^* \leftarrow \text{argmin}\{f(x_s), f(x_t)\}$
$x \leftarrow x_s$; $y \leftarrow x_t$
**enquanto** $|\Delta(x, y)| > 1$ **faça**
  $m^* \leftarrow \text{argmin}\{f(x \oplus m) : m \in \Delta(x, y)\}$
  $x \leftarrow x \oplus m^*$
  Atualize $\Delta(x, y)$ e $\Delta(y, x)$
  **se** $f(x) < f^*$ **então**
    $f^* \leftarrow f(x)$
    $x^* \leftarrow x$
  **fim se**
  $t \leftarrow y$; $y \leftarrow x$; $x \leftarrow t$
**fim enquanto**
$x^* \leftarrow$ `BuscaLocal`$(x^*)$
**retornar** $x^*$;

**Algoritmo 6** Uma heurística GRASP-PR básica para minimização.

Entrada: $i_{\max}$.
Resultado: Solução $x^*$.
$P \leftarrow \emptyset$
$f^* \leftarrow \infty$
**para** $i = 1, \ldots, i_{\max}$ **faça**
  $x \leftarrow$ `ConstrucaoGulosaAleatoria`()
  **se** $x$ não for viável **então**
    $x \leftarrow$ `reparo`$(x)$
  **fim se**
  $x \leftarrow$ `BuscaLocal`$(x)$
  **se** $i \geq 2$ **então**
    Escolha aleatoriamente as soluções elite $\mathcal{Y} \subseteq P$ para religar com $x$
    **para** $y \in \mathcal{Y}$ **faça**
      $x_p \leftarrow$ `ReligamentoDeCaminhos`$(x, y)$
      Atualize o conjunto elite $P$ com $x_p$
      **se** $f(x_p) < f^*$ **então**
        $f^* \leftarrow f(x_p)$
        $x^* \leftarrow x_p$
      **fim se**
    **fim para**
  **fim se**
**fim para**
$P \leftarrow$ `PosOtimizacao`$\{P\}$
$x^* \leftarrow \text{argmin}\{f(x), x \in P\}$
**retornar** $x^*$

Werneck, 2004, 2006; Ribeiro et al., 2002). Um resumo de GRASP com religamento de caminhos é

apresentado em Resende e Ribeiro (Resende e Ribeiro, 2005a). Duas estratégias básicas são usadas. Na primeira, o religamento de caminhos é aplicado a todos os pares de soluções elite, seja periodicamente durante as iterações GRASP ou como uma estapa de pós-otimização, após todas as iterações GRASP terem sido realizadas. Na segunda, o religamento de caminhos é aplicado como uma estratégia de intensificação a cada ótimo local obtido após a busca local.

Aplicar religamento de caminhos como uma estratégia de intensificação sobre cada ótimo local geralmente é mais eficaz do que simplesmente usá-lo como uma etapa de pós-otimização. Em geral, combinar intensificação com pós-otimização resulta em uma melhor estratégia. No contexto de intensificação, religamento de caminhos é aplicado em pares $(x, y)$ de soluções, onde $x$ é uma solução ótima local produzida a cada iteração GRASP, após a aplicação da busca local, e $y$ é uma das soluções elite aleatoriamente escolhidas a partir de um conjunto elite com um número máximo `Max_Elite` de elementos obtidos no decorrer da busca. A seleção aleatória uniforme é uma simples estratégia a ser implementada. Desde que a diferença simétrica é uma medida do tamanho do caminho a ser explorado durante o religamento, uma estratégia viciada que privilegia as soluções $y$ do conjunto elite com alta diferença simétrica em relação a $x$ é usualmente melhor do que aquela usando a seleção uniformemente aleatória (Resende e Werneck, 2004).

O conjunto elite inicialmente é vazio. Desde que nós desejamos manter este conjunto com diversificadas soluções de alta qualidade, cada ótimo local obtido pela busca local é considerado como um candidato a ser inserido no conjunto elite, caso seja suficientemente diferente de todas as outras soluções atualmente contidas no mesmo. Se o conjunto já possui `Max_Elite` soluções e o candidato é melhor do que o pior entre eles, então uma simples estratégia consiste em substituir a segunda pela primeira. Outra estratégia, a qual tende a aumentar a diversidade do conjunto elite, consiste em substituir o elemento mais similar à solução candidata dentre todos os elementos do conjunto elite com custo pior do que o custo da solução candidata. Caso o conjunto elite não esteja cheio, o candidato é simplesmente adicionado ao mesmo.

A pós-otimização é realizada sobre uma série de conjuntos. O conjunto inicial $P_0$ consiste no conjunto $P$ obtido ao final das iterações GRASP. O valor da melhor solução de $P_0$ é atribuída a $f_0^*$ e um contador de iterações $k$ é inicializado para 0. Na $k$-ésima iteração, todos os pares de elementos do conjunto $P_k$ são combinados através do religamento de caminhos. Cada resultado do religamento de caminhos é testado para inclusão no conjunto $P_{k+1}$ seguindo os mesmos critérios usados nas iterações GRASP. Se uma nova melhor solução é produzida, ou seja, $f_{k+1}^* < f_k^*$, então $k \leftarrow k + 1$ e uma nova iteração da pós-otimização é realizada. Caso contrário, a pós-otimização é encerrada com $x^* \in \text{argmin}\{f(x) \mid x \in P_{k+1}\}$ como resultado.

O pseudo-código do Algoritmo 6 ilustra este procedimento. Agora, cada iteração GRASP possui três passos principais. Na fase de construção, um procedimento de busca gulosa e aleatória é usado para construir uma solução viável. Na fase de busca local a solução construída na primeira fase é progressivamente melhorada por uma estratégia de busca na vizinhança, até que um mínimo local seja encontrado. Na fase de religamento de caminhos o algoritmo de religamento de caminhos é aplicado sobre a solução obtida pela busca local e uma solução aleatoriamente selecionada a partir do conjunto elite. A melhor solução encontrada ao longo desta trajetória é também considerada como uma candidata a inserção no conjunto elite. Ao final das iterações GRASP, a fase de pós-otimização combina as soluções do conjunto elite na busca por melhores soluções. Mateus, Resende e Silva (Mateus et al., 2009) propõem uma nova versão de GRASP com religamento de caminhos, motivado pelo fato de que em alguns problemas um movimento guiado pela solução destino não garante a viabilidade da nova solução construída.

## 6. Conclusões

Este capítulo considerou os componentes estruturais básicos necessários para projetar heurísticas baseadas nos princípios que regem a meta-heurística GRASP. Estes componentes incluem esquemas de construção aleatórios, procedimentos de busca local e a introdução de estruturas de memória através do religamento de caminhos. Um tópico importante não abordado neste capítulo é o GRASP paralelo. O leitor interessado é direcionado ao artigo de Resende e Ribeiro (Resende e Ribeiro, 2005b).