Capítulo 1

GRASP: Procedimentos de Busca Gulosos, Aleatórios e Adaptativos

Mauricio Guilherme de Carvalho Resende e Ricardo Martins de Abreu Silva*

Resumo: Procedimento de busca gulosos, aleatórios e adaptativos (do inglês *Greedy Randomized Adaptive Search Procedure* – GRASP) é uma meta-heurística multi-partida para otimização combinatória 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, hibridizações com o método de religamento de caminhos e a versão do GRASP paralela. Por fim, exemplos de aplicações da meta-heurística GRASP em problemas de lógica e atribuição oriundas da literatura são apresentadas.

Palavras-chave: GRASP, Meta-heurística, Heurísticas híbridas, Religamento de caminhos, Busca local.

Abstract: *GRASP (Greedy Randomized Adaptive Search Procedures) is a multi-start metaheuristic for combinatorial optimization that repeatedly applies local search from solutions generated with a randomized greedy algorithm. This chapter presents the building blocks of GRASP, its hybridization with path-relinking, and its parallel implementation. The chapter concludes with examples, found in the literature, of the application of GRASP to logic and assignment problems.*

Keywords: *GRASP, Metaheuristics, Hybrid heuristics, Path relinking, Local search.*

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, será considerada 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 práticos de otimização combinatória pode-se 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, *simulated annealing*, busca tabu, *variable neighborhood search*, *scatter search*, religamento de caminhos, *iterated local search*, *ant colony optimization*, *swarm optimization* e procedimento de busca guloso, aleatório 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. As hibridizações de GRASP com religamento de caminhos são explanadas na Seção 5, enquanto o GRASP paralelo na Seção 6. Por fim, exemplos de aplicações da meta-heurística GRASP nas áreas de lógica e atribuição são apresentadas na Seção 7, seguidas pelas conclusões na Seção 8.

*Autor para contato: rmas@cin.ufpe.br

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*.

Dados: $s_0 \in S$
Resultado: ótimo local.
1 $t \leftarrow s_0$;
2 **enquanto** *existe* $s \in N(t)$ *tal que* $f(s) < f(t)$ **faça**
3 $\quad t \leftarrow s$;
4 **fim**
5 **retorna** t;

Algoritmo 1: Busca local padrão

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. Pode-se 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, *variable neighborhood descent* (Andrade & Resende, 2006; Hansen & Mladenović, 1998; Martins et al., 1999; Ribeiro & Souza, 2002; Ribeiro et al., 2002; Ribeiro & Vianna, 2005), *variable neighborhood search* (Beltrán et al., 2004; Canuto et al., 2001; Drummond et al., 2002; Festa et al., 2002; Hansen & Mladenović, 1998; Ochi et al., 2001), *busca tabu de memória curta* (Abdinnour-Helm & Hadley, 2000; de Souza et al., 2003; Delmaire et al., 1999; Glover, 1989, 1990; Laguna & González-Velarde, 1991; Li et al., 2004; Lim & Wang, 2004; Moura & Oliveira, 2005; Pu et al., 2006; Serra & Colomé, 2001; Souza et al., 2001), *simulated annealing* (de la Peña, 2004; Kirkpatrick et al., 1983; Liu et al., 2000), *iterated local search* (Baum, 1986, 1987; Baxter, 1981; Johnson, 1990; Martin & Otto, 1996; Martin et al., 1991; Ribeiro & Urrutia, 2007), e *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.

3. Procedimentos de Busca Gulosos, Aleatórios e 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 procedimentos 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 et al. (2007a), Duarte et al. (2007b), Nascimento et al. (2010) e Mateus et al. (2011).

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 será visto 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 & Resende (1989). Ver Feo & Resende (1995), Pitsoulis & Resende (2002), Resende & Ribeiro (2003), Resende et al. (2010b) e Resende (2008) para um revisão da meta-heurística GRASP e Festa & Resende (2002, 2009a,b) para uma bibliografia anotada do assunto.

Dados: critério de parada
Resultado: solução x^*.
1 $x^* \leftarrow \infty$;
2 **enquanto** *critério de parada não for satisfeito* **faça**
3 $\quad x \leftarrow$ `GulosoAleatorio`$(\cdot)$;
4 $\quad$ **se** x *não for viável* **então**
5 $\quad\quad x \leftarrow$ `reparo`(x);
6 $\quad$ **fim**
7 $\quad x \leftarrow$ `BuscaLocal`(x);
8 $\quad$ **se** $f(x) < f(x^*)$ **então**
9 $\quad\quad x^* \leftarrow x$;
10 $\quad$ **fim**
11 **fim**
12 **retorna** x^*;

Algoritmo 2: Pseudo-código de um GRASP básico.

4. Método de Construção Guloso Aleatório

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

4.1 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 & 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 α determina quão guloso ou aleatório 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.

Dados: $E = \{$ conjunto discreto finito $\}$.
Resultado: solução S.
$S \leftarrow \emptyset$; $C \leftarrow E$;
enquanto $|C| > 0$ **faça**
  Para todo $c \in C$ computar o valor da função gulosa $g(c)$;
  Definir $LRC(C) \leftarrow \{c \in C \mid g(c)$ tam um valor reduzido$\}$;
  Selecionar aleatoriamente $c^* \in LRC(C)$;
  Adicionar c^* à solução parcial: $S \leftarrow S \cup \{c^*\}$;
  Seja C o conjunto de elementos que
  podem ser adicionados à S;
fim
retorna S;

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

4.2 Construção por amostragem gulosa

Neste segundo esquema de construção, denominado *construção por amostragem gulosa* por Resende & 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. Este processo é repetido até quando não existir nenhum candidato.

Dados: $p, E = \{$ conjunto discreto finito $\}$.
Resultado: solução S.
$S \leftarrow \emptyset$; $C \leftarrow E$;
enquanto $|C| > 0$ **faça**
  Amostrar aleatoriamente $\min\{p, |C|\}$ elementos
  a partir de C e inserí-los em $LRC(C)$;
  Selecionar $c^* = \text{argmin}\{g(c) \mid c \in LRC(C)\}$;
  Adicionar c^* à solução parcial: $S \leftarrow S \cup \{c^*\}$;
  Seja C o conjunto de elementos
  que podem ser adicionados à S;
fim
retorna S;

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

4.3 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 & 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 Seção 4.2.

4.4 Construção gulosa proporcional

A cada iteração deste quarto esquema de construção, denominado *guloso proporcional* e introduzido pela primeira vez por Resende & 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 $g(c') - \min\{g(c)|c \in C\}$.

4.5 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 α pode ser selecionado de modo uniformemente aleatório a partir do intervalo $[0, 1]$, de modo que cada iteração GRASP use um valor diferente de α, resultando em um balanceamento diferente entre os enfoques guloso e aleatório.

Prais & Ribeiro (2000) mostraram que usando um valor fixo para o parâmetro α 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 α fosse utilizado. Sendo assim, eles propuseram uma extensão do procedimento GRASP básico, denominado *GRASP reativo*, na qual o parâmetro α 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 & Ribeiro (2000) definem $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ como o conjunto de valores possíveis para α. As probabilidades associadas com a escolha de cada valor são inicialmente iguais a $p_i = 1/m,\ i = 1, \ldots, m$. Além disto, 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 α. 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 membros 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.

4.6 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 & 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 é necessário 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 & 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 fortemenete 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 λ, por exemplo, inicialmente λ pode ser inicializado para um valor alto que decresce a medida que a diversifição é chamada.

4.7 Construção por amostragem viciada

O sétimo esquema de construção foi introduzido por 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 (1996), uma família de distribuições de probabilidade é introduzida. Elas são baseadas na colocação (*rank*) $r[\sigma]$ atribuída a cada elemento candidato σ, 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 respectivoso *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 $mathitLRC$ ou a partir do conjunto completo de candidatos, ou seja, $LRC = C$.

4.8 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 & 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 et al. (2002) mostram que este caso ocorre na heurística de caminho mínimo de Takahashi & 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 et al. (2001) para o problem de Steiner com prêmios (*prize-collecting Steiner tree problem*), o qual fez uso de um algoritmo de aproximação primal-dual de Goemans & 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 (1996) como uma estratégia de intensificação para explorar trajetórias conectando soluções elite obtidas pela busca tabu ou *scatter search* (Glover, 2000; Glover & 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. O pseudo-código do Algoritmo 5 ilustra o religamento de caminhos misto sobre o par de soluções x_s e x_t.

O religamento de caminhos misto (Ribeiro & 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 & Martí (1999) e posteriormente seguida por várias extensões,

Dados: $x_s \in S$, $x_t \in S$.
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
    $t \leftarrow y$; $y \leftarrow x$; $x \leftarrow t$;
fim
$x^* \leftarrow$ `BuscaLocal`(x^*);
retorna x^*;

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

melhorias e aplicações de sucesso (Mateus et al., 2011; Aiex et al., 2005; Canuto et al., 2001; Festa et al., 2007; Oliveira et al., 2004a; Resende et al., 2010a; Resende & Ribeiro, 2003; Resende & Werneck, 2004, 2006; Ribeiro et al., 2002). Um resumo de GRASP com religamento de caminhos é apresentado em Resende & Ribeiro (2005a). Duas estratégias básicas são usadas. Na primeira, o religamento de caminhos é aplicada 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 & Werneck, 2004).

O conjunto elite inicialmente é vazio. Uma vez que se deseja manter este conjunto com soluções diversificadas 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 et al. (2011) 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.

Dados: $i_{\max}$.
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
  $x \leftarrow$ `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
    fim
  fim
fim
$P \leftarrow$ `PosOtimizacao`$\{P\}$;
$x^* \leftarrow \text{argmin}\{f(x), x \in P\}$;
retorna x^*;

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

6. GRASP Paralelo

Assim como qualquer heurística multi-partida (*multi-start*) para otimização combinatória, um GRASP pode ser implementado em paralelo dividindo-se k iterações independentes entre ρ processadores. Outro enfoque consiste em dar um valor alvo τ da função objetivo a cada processador e executar o algoritmo até que o primeiro processador encontre uma solução com valor pelo menos tão bom quanto τ, altura em que o restante dos processadores finalizão suas execuções. Cabe ressaltar que é necessário garantir que nenhuma iteração em paralelo inicie seu processamento com a mesma semente do gerador de números aleatórios (Pardalos et al., 1996). Estes são exemplos do paralelismo do caminho independente múltiplo (*multiple independent walk*) (Verhoeven & Aarts, 1995).

Diversas implementações paralelas de GRASP usando as estratégias acima têm sido relatadas na literatura, por exemplo (Martins et al., 2000, 1998; Murphey et al., 1998; Pardalos et al., 1995, 1996). Na maioria destes artigos, uma observação comum foi realizada. A aceleração nos tempos de respostas medidos eram proporcionais ao número de processadores. Um típico exemplo pode ser visto em Pardalos et al. (1996) onde, para uma implementação baseada em PVM de um GRASP paralelo para o problema do MAX-SAT, foi medida uma aceleração média quase idêntica ao número de processadores (ver Tabela 1).

Esta observação pode ser explicada se a variável aleatória *solution time to target* é exponencialmente distribuída, como indicada pela seguinte proposição (Verhoeven & Aarts, 1995).

Proposição 1 *: Seja $P_\rho(t)$ a probabilidade de uma dada solução alvo não ser encontrada em t unidades de tempo com ρ processos independentes. Se $P_1(t) = e^{-t/\lambda}$ com $\lambda \in \mathbb{R}^+$, i.e. P_1 corresponde a uma distribuição exponencial, então $P_\rho(t) = e^{-\rho t/\lambda}$.*

Tabela 1. Tempo de CPU (em segundos) e aceleração sobre problemas MAX-SAT. Aceleração média é mostrada para 5, 10, e 15 processadores.

problema	**1 processador**	**5 processadores**		**10 processadores**		**15 processadores**	
	tempo	tempo	aceleração	tempo	aceleração	tempo	aceleração
jnh201	310,4	62,8	4,9	30,5	10,2	22,2	14,0
jnh202	312,2	59,8	5,2	31,2	10,0	23,4	13,3
jnh203	351,2	72,3	4,9	35,2	10,0	23,2	15,1
jnh205	327,8	63,4	5,2	32,1	10,2	22,5	14,6
jnh207	304,7	56,7	5,4	29,6	10,3	19,8	15,4
jnh208	355,2	65,6	5,4	33,2	10,7	21,0	16,9
jnh209	339,0	60,5	5,6	33,6	10,1	21,6	15,7
jnh210	318,5	57,6	5,5	32,5	9,8	20,8	15,3
jnh301	414,5	85,3	4,9	45,2	9,2	28,3	14,6
jnh302	398,7	88,6	4,5	48,2	8,3	27,0	14,7
aceleração média:			5,2		9,9		15,0

A proposição 1 é oriunda da própria definição de distribuição exponencial. Isto implica que a probabilidade de uma solução com um dado valor ser encontrada em tempo ρt com um processo sequencial é igual a probabilidade de encontrar uma solução ao menos tão boa quanto o valor dado em tempo t com ρ processos paralelos independentes. Sendo assim, em tempo linear de aceleração é possível alcançar a solução alvo via múltiplos processos independentes.

Uma proposição análoga mediante uma distribuição exponencial bi-paramétrica deslocada (*shifted*).

Proposição 2 *: Seja $P_\rho(t)$ a probabilidade de uma dada solução alvo não ser encontrada em t unidades de tempo com ρ processos independentes. Se $P_1(t) = e^{-(t-\mu)/\lambda}$ com $\lambda \in \mathbb{R}^+$ e $\mu \in \mathbb{R}$, i.e. P_1 corresponde a uma distribuição exponencial bi-paramétrica, então $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$.*

Analogamente, a proposição 2 é oriunda da própria definição de distribuição exponencial bi-paramétrica. Isto implica que a probabilidade de uma solução com um dado valor ser encontrada em tempo ρt com um processo sequencial é igual a $1 - e^{-(\rho t-\mu)/\lambda}$, enquanto a probabilidade de encontrar uma solução ao menos tã boa quanto o valor dado em tempo t com ρ processos paralelos independentes é $1 - e^{-\rho(t-\mu)/\lambda}$. Note que se $\mu = 0$, então ambas probabilidades são iguais e correspondem a distribuição exponencial não deslocada (*non-shifted*). Além disto, se $\rho\mu \ll \lambda$, então as duas probabilidades são aproximadamente iguais. Sendo assim, em tempo linear de aceleração é possível alcançar a solução alvo via múltiplos processos independentes.

Aiex et al. (2002) estudaram as distribuições de probabilidade empíricas da variável aleatória tempo-para-solução-alvo (*time to target solution*) em cinco implementações GRASP. Eles mostraram que, dado um valor da solução alvo, o tempo que o GRASP leva para encontrar uma solução ao menos tão boa quanto a alvo se encaixa em uma distribuição exponencial biparamétrica. Por exemplo, a Figura 1 mostra um gráfico com distribuições teóricas empíricas e estimadas para a heurística GRASP aplicada a uma instância particular de um problema. Este gráfico foi gerado mediante 200 execuções independentes do GRASP, cuja metodologia encontra-se descrita em Aiex et al. (2002).

7. Aplicações GRASP

Desde 1980 a heurística GRASP tem sido aplicada a uma grande variedade de problemas de otimização industrial e de pesquisa operacional. Isto inclui problemas em escalonamento, roteamento, lógica, particionamento, localização e *layout*, teoria dos grafos, atribuição, manufatura, transporte, telecomunicações, desenho automático, sistema de potência elétrico, projeto VLSI, entre outros. Festa & Resende (2001, 2009a,b) apresentaram uma extensiva bibliografia anotada da literatura sobre GRASP. Nesta Seção será apresentada uma pequena parte destas aplicações, limitando-se a problemas de lógica e de atribuição.

7.1 Lógica: máxima satisfatibilidade ponderada (Festa et al., 2007).

A fórmula proposicional Φ sobre um conjunto de n variáveis booleanas $V = \{x_1, \ldots, x_n\}$ na forma normal conjuntiva (*conjunctive normal form* - CNF) é uma conjunção sobre um conjunto de m cláusulas

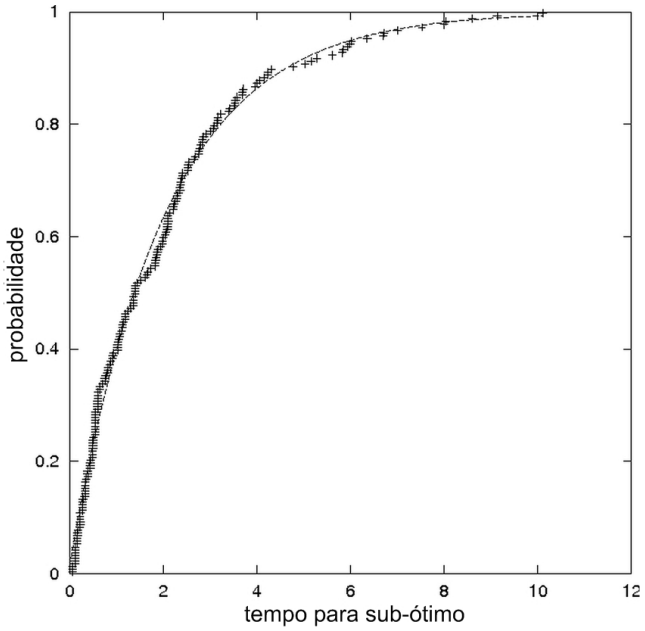

Figura 1. Distribuições teórica e empírica superpostas.

$\mathbb{C} = \{C_1, \ldots, C_m\}$. Cada cláusula C_i é uma disjunção de $|C_i|$ literais, onde cada literal l_{ij} é uma variável x_j ou sua negação $\neg x_j$. Formalmente, escreve-se

$$\Phi = \bigwedge_{i=1}^{m} C_i = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{|C_i|} l_{ij} \right).$$

Uma cláusula é satisfeita se ao menos um dos seus literais é igual a 1 (verdadeiro). O que significa que alguma variável Booleana não negada é igual a 1 (um), ou alguma negada é igual a 0 (zero). A fórmula proposicional é dita estar satisfeita se todas as suas cláusulas são satisfeitas. No problema de satistafibilidade (*satisfiability problem* - SAT), deve-se decidir se existe uma atribuição de valores-verdade às variáveis tal que uma dada fórmula proposicional é satisfeita. SAT foi o primeiro problema a ser provado como NP-completo (Cook, 1971). O problema da máxima satisfatibilidade (*maximum Satisfiability problem* – MAX-SAT) é uma generalização do problema SAT onde, dada uma fórmula proposicional, deseja-se encontrar uma atribuição de valores-verdade às variáveis de modo a maximizar o número de cláusulas satisfeitas. Generalizando ainda mais, se for introduzido um peso positivo w_i a cada cláusula C_i, então o problema MAX-SAT ponderado consiste em encontrar uma atribuição de valores-verdade às variáveis tal que a soma de pesos das cáusulas satisfeitas seja maximizada. O problema MAX-SAT tem diversas aplicações práticas e teóricas em áreas tais como teoria da complexidade, otimização combinatória e inteligência artificial (Battiti & Protasi, 1998). É um problema intratável no sentido de que não existe um algoritmo que resolva o problema em tempo polinomial a menos que P = NP (Garey & Johnson, 1979).

7.1.1 GRASP: fase construtiva

Na fase de construção do algoritmo, descrita no Algoritmo 7, uma lista restrita de candidatos (LRC) é mantida contendo elementos correspondentes às atribuições de valores 1 (`verdadeiro`) ou 0 (`falso`) para variáveis ainda não atribuídas. Escolher um elemento a ser adicionado a uma solução parcial a partir da lista LRC corresponde a atualizar o respectivo valor verdade de uma dada variável. Dada uma solução parcial, a qual corresponde a um conjunto de cláusulas satisfeitas, deseja-se que o próximo elemento a ser adicionado a solução maximize o peso total das cláusulas não satisfeitas que passam a ser satisfeitas após a atribuição. Seja $N = \{1, 2, \ldots, n\}$ e $M = \{1, 2, \ldots, m\}$ os conjuntos de índices associados aos conjuntos de variáveis e cláusulas, respectivamente. Além disto, para $i \in N$, seja Γ_i^+ o conjunto de cláusulas não satisfeitas que se tornariam satisfeitas se a variável x_i fosse atualizada para `verdadeiro`, e Γ_i^- o conjunto de cláusulas não satisfeitas que se tornariam satisfeitas se a variável x_i fosse atualizada para `falso`. Considere γ_j^+ e γ_j^- o ganho no valor da função objetivo, se for atualizado o valor da variável não atribuída x_j para 1 e 0, respectivamente. Especificamente

$$\gamma_i^+ = \sum_{j \in \Gamma_i^+} w_j \text{ e } \gamma_i^- = \sum_{j \in \Gamma_i^-} w_j.$$

Se $X \subseteq V$ é o conjunto das variáveis já atribuídas, computa-se o melhor ganho

$$\gamma^* := \max\{\gamma_j^+, \gamma_j^- : j \text{ tal que } x_j \in V \setminus X\}$$

e mantém-se no LRC apenas aquelas atribuições com γ_j^+ e γ_j^- maiores ou iguais a $\alpha \cdot \gamma^*$ onde $0 \leq \alpha \leq 1$ é um parâmetro. Uma escolha aleatória a partir de LRC corresponde a uma nova atribuição $x_s = 1$ ($x_s = 0$), a qual é adicionada a solução parcial $X = X \cup \{x_s\}$. Após cada uma destas adições à solução parcial, os conjuntos Γ_i^+, Γ_i^-, assim como os ganhos γ_j^+ e γ_j^- são atualizados, processo o qual é ilustrado no Algoritmo 7, onde s é o índice da variável já adicionada à solução e L_q o conjunto dos índices das variáveis ainda não atribuídas pertencentes à cláusula C_q. Pode-se reconhecer dois casos possíveis. Se a variável já adicionada foi atualizada para `verdadeiro` então Γ^+, Γ^-, γ^+ e γ^- são atualizadas nas linhas 5, 8, 12, e 13, enquanto se a variável já atribuída foi atualizada para `falso` então Γ^+, Γ^-, γ^+ e γ^- são atualizada nas linhas 19, 22, 26 e 27. O processo é repetido até $|X| = n$. Por fim, o parâmetro α reflete a razão de aleatoriedade versus gulosidade no processo de construção, onde $\alpha = 1$ corresponde a pura seleção gulosa e $\alpha = 0$ a uma pura seleção aleatória.

Dados: índice s.
Resultado: solução para MAX-SAT ponderado.

```
1  se x_s = 1 então
2  |  para j ∈ Γ_s^+ faça
3  |  |  para k ∈ L_j (k ≠ s) faça
4  |  |  |  se x_k não é negada na cláusula j então
5  |  |  |  |  Γ_k^+ = Γ_k^+ − {j}; γ_k^+ = γ_k^+ − w_j;
6  |  |  |  fim
7  |  |  |  se x_k é negada na cláusula j então
8  |  |  |  |  Γ_k^− = Γ_k^− − {j}; γ_k^− = γ_k^− − w_j;
9  |  |  |  fim
10 |  |  fim
11 |  fim
12 |  Γ_s^+ = ∅; Γ_s^− = ∅;
13 |  γ_s^+ = 0; γ_s^− = 0;
14 fim
15 se x_s = 0 então
16 |  para j ∈ Γ_{−s}^− faça
17 |  |  para k ∈ L_j (k ≠ s) faça
18 |  |  |  se x_k não é negada na cláusula j então
19 |  |  |  |  Γ_k^+ = Γ_k^+ − {j}; γ_k^+ = γ_k^+ − w_j;
20 |  |  |  fim
21 |  |  |  se x_k é negada na cláusula j então
22 |  |  |  |  Γ_k^− = Γ_k^− − {j}; γ_k^− = γ_k^− − w_j;
23 |  |  |  fim
24 |  |  fim
25 |  fim
26 |  Γ_{−s}^+ = ∅; Γ_{−s}^+ = ∅;
27 |  γ_{−s}^+ = 0; γ_{−s}^− = 0;
28 fim
29 retorna
```

Algoritmo 7: Pseudo-código da função gulosa da fase construtiva do GRASP-PR aplicada ao problema MAX-SAT ponderado.

7.1.2 GRASP: busca Local

A vizinhança 1-*flip* é usada na busca-local, a qual é definida como segue:

$$N_1(\mathbf{x}) := \{\mathbf{y} \in \{0,1\}^n : d(\mathbf{x}, \mathbf{y}) = 1\}. \tag{1}$$

onde $\mathbf{x}$ é um máximo local se e somente se $c(\mathbf{x}) \geq c(\mathbf{y})$ para todo $\mathbf{y} \in N_1(\mathbf{x})$. Uma implementação direta do procedimento da busca local requereria $|N_1| = n$ avaliações de função para encontrar a melhor solução em

uma dada vizinhança, onde cada avaliação computaria a soma dos pesos das cláusulas satisfeitas. Se deseja-se encontrar um máximo local, pode-se precisar de um número exponencial de passos computacionais (Johnson et al., 1988; Krentel, 1988). Entretanto, pode-se explorar a estrutura da vizinhança para reduzir o esforço computacional.

Dada uma solução inicial $\mathbf{x}$ defina G_i como o ganho em peso total resultante da variável x_i em $\mathbf{x}$, para todo i. Seja k tal que $G_k = \max\{G_i \mid i \in N\}$. Se $G_k = 0$ então $\mathbf{x}$ é o máximo local e a busca local termina. Caso contrário, a atribuição verdade resultante de x_k em $\mathbf{x}$ é um máximo local e, portanto, é preciso apenas atualizar os valores G_i tal que a variável x_i ocorra na cáusula na qual x_k ocorra, pois os valores restantes G_i não mudam na nova atribuição verdade.

Após a atualização dos valores G_i, repete-se o mesmo processo até $G_k = 0$ onde o procedimento da busca local é terminado. O procedimento é descrito no pseudo-código do Algoritmo 8. Dada uma atribuição verdade $\mathbf{x}$ e um índice k correspondente a variável x_k, o procedimento `GerarGanhos` é usado para atualizar os valores G_i retornados em um array G. Note que na linha 2, $k = 0$ é passado à procedure desde que inicialmente todos os valores de G_i sejam gerados (por convenção a variável x_0 ocorre em todas as cláusulas). Nas linhas 4 até 8, o procedimento encontra um máximo local, cujo valor é salvo na linha 9.

Dados: $\mathbf{x}$,`MelhorSoluçãoEncontrada`.
Resultado: `MelhorSoluçãoEncontrada`.
1 `MelhorSoluçãoEncontrada` $= c(\mathbf{x})$;
2 `GerarGanhos`($\mathbf{x}$,G,0);
3 $G_k = \max\{G_i \mid i = 1, \ldots, n\}$;
4 **enquanto** $G_k \neq 0$ **faça**
5 | Troque o valor de x_k;
6 | `GerarGanhos`($\mathbf{x}$,G,k);
7 | $G_k = \max\{G_i \mid i = 1, \ldots, n\}$;
8 **fim**
9 `MelhorSoluçãoEncontrada` $= c(\mathbf{x})$;
10 **retorna** `MelhorSoluçãoEncontrada`

Algoritmo 8: Pseudo-código do procedimento da Busca Local do GRASP-PR aplicado ao problema MAX-SAT ponderado

7.1.3 Religamento de caminhos

Seja $\mathbf{x}$ uma *solução inicial*, $\mathbf{y}$ a *solução alvo*, $\mathcal{E}$ o conjunto elite cujo tamanho não exceda `MaxElite`. Denota-se o conjunto de soluções gerado pelos elementos comuns de $\mathbf{x}$ e $\mathbf{y}$ como:

$$S(\mathbf{x}, \mathbf{y}) := \{\mathbf{w} \in \{0,1\}^n : w_i = x_i = y_i, i \notin \Delta(\mathbf{x}, \mathbf{y})\} \setminus \{\mathbf{x}, \mathbf{y}\}, \tag{2}$$

onde é evidente que $|S(\mathbf{x}, \mathbf{y})| = 2^{d(\mathbf{x},\mathbf{y})} - 2$. Levando-se em consideração que o tamanho deste espaço é exponencialmente grande, empregar-se-à uma busca gulosa onde o caminho de soluções

$$\mathbf{x} = \mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{d(\mathbf{x},\mathbf{y})}, \mathbf{w}_{d(\mathbf{x},\mathbf{y})+1} = \mathbf{y},$$

é construído, tal que $d(\mathbf{w}_i, \mathbf{w}_{i+1}) = 1, \quad i = 0, \ldots, d(\mathbf{x}, \mathbf{y})$, e a melhor solução oriunda deste caminho é escolhida. Desde que ambas $\mathbf{x}, \mathbf{y}$ são máximos locais em alguma vizinhança N_1 oriundos da fase construtiva, com o intuito de $S(\mathbf{x}, \mathbf{y})$ possuir soluções não contidas nas vizinhanças de $\mathbf{x}$ ou $\mathbf{y}$ nós deve-se ter $d(\mathbf{x}, \mathbf{y}) > 3$. Portanto, não é preciso aplicar o religamento de caminhos entre duas soluções que não sejam suficientemente distantes entre si, desde que uma nova solução melhor que ambas $\mathbf{x}$ e $\mathbf{y}$ não será encontrada.

O pseudo-código do religamentos dos caminhos para o problema da máxima satisfatibilidade ponderada é mostrado no Algoritmo 9. Considerando-se que a solução inicial sempre será uma solução do conjunto elite, enquanto a alvo é oriunda da iteração GRASP, então uma maior liberdade na busca sobre a vizinhança ao redor da solução elite. Na linha 1, uma solução inicial $\mathbf{x}$ é selecionada aleatoriamente dentre os elementos do conjunto elite, a qual suficientemente difere da solução alvo $\mathbf{y}$. Na linha 2, a solução inicial é atualizada como $\mathbf{w}_0$, e na linha 3 $\mathbf{x}$ é salva como a melhor solução. O loop da linha 4 a 17 computa um caminho de soluções $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{d(\mathbf{x},\mathbf{y})-2}$, e a solução com o melhor valor da função objetivo é retornado na linha 18. Isto é obtido avançando uma solução por vez de modo guloso, como ilustrado da linha 6 a 12, enquanto a operação `flip`$(\mathbf{w}_k, i)$ tem o efeito de negar a variável w_i na solução $\mathbf{w}_k$. Cabe observar que o caminho nunca entra na vizinhança de $\mathbf{y}$.

Dados: $p, \mathcal{E} = \{$ conjunto discreto finito $\}$.
Resultado: máximo local $\mathbf{w}^*$.
1 Aleatoriamente selecione uma solução $\mathbf{x} \in \{\mathbf{z} \in \mathcal{E} : d(\mathbf{y}, \mathbf{z}) > 4\}$;
2 $\mathbf{w}_0 := \mathbf{x}$;
3 $\mathbf{w}^* := \mathbf{x}$;
4 **para** $k = 0, \ldots, d(\mathbf{x}, \mathbf{y}) - 2$ **faça**
5 max:= 0
6 **para** $i \in \Delta(\mathbf{w}_k, \mathbf{y})$ **faça**
7 $\mathbf{w}$:=flip$(\mathbf{w}_k, i)$;
8 **se** $c(\mathbf{w}) > max$ **então**
9 $i^* := i$;
10 max:= $c(\mathbf{w})$;
11 **fim**
12 **fim**
13 $\mathbf{w}_{k+1}$:=flip$(\mathbf{w}_k, i^*)$;
14 **se** $c(\mathbf{w}_{k+1}) > c(\mathbf{w}^*)$ **então**
15 $\mathbf{w}^* := \mathbf{w}_{k+1}$;
16 **fim**
17 **fim**
18 **retorna** $\mathbf{w}^*$

Algoritmo 9: Pseudo-código do religamento de caminhos do GRASP-PR aplicado ao problema MAX-SAT ponderado.

Na integração do procedimento de religamento de caminhos com GRASP, o conjunto elite de soluções é inicializado vazio e até ele alcançar sua capacidade máxima nenhum religamento de caminhos é realizado. Após uma solução $\mathbf{y}$ ser encontrada pelo GRASP, ela é passada para o procedimento do religamento de caminhos para gerar outra solução. Cabe observar que a mesma solução $\mathbf{y}$ pode ser obtida após o religamento de caminhos. Uma solução $\mathbf{y}$ é adicionada ao conjunto elite se alguma das seguintes condições for satisfeita:

1. $c(\mathbf{y}) > \max\{c(\mathbf{w}) : \mathbf{w} \in \mathcal{E}\}$,
2. $c(\mathbf{y}) > \min\{c(\mathbf{w}) : \mathbf{w} \in \mathcal{E}\}$ e $d(\mathbf{y}, \mathbf{w}) > \beta n, \quad \forall \mathbf{w} \in \mathcal{E}$, onde β é um parâmetro entre 0 e 1, com n sendo o número de variáveis.

Se $\mathbf{y}$ satisfaz uma das condições acima, ela então repõe uma solução elite $\mathbf{z}$ de peso não maior que $c(\mathbf{y})$ e mais similar a $\mathbf{y}$, i.e. $\mathbf{z} = \text{argmin}\{d(\mathbf{y}, \mathbf{w}) : \mathbf{w} \in \mathcal{E} \text{ such that } c(\mathbf{w}) \leq c(\mathbf{y})\}$.

7.2 Problemas de atribuição: atribuição quadrática generalizada (Mateus et al., 2011).

O problema da atribuição quadrática generalizada (*generalized quadratic assignment problem* – GQAP) (Savelsbergh, 1997; Ross & Soland, 1975; Elloumi et al., 2003; Lee & Ma, 2005; Cordeau et al., 2006; Hahn et al., 2008) é uma generalização do problema NP-Difícil da atribuição quadrática (*quadratic assignment problem* - QAP) (Oliveira et al., 2004b; Pardalos et al., 1994; Li et al., 1994) em que múltiplas instalações (*facilities*) podem ser atribuídas a uma mesma localidade, na medida em que a capacidade destas localidades permitam.

Seja $N = \{1, \ldots, n\}$ um conjunto de instalações (*facilities*) e $M = \{1, \ldots, m\}$ um conjunto de localidades. Além disto, denote por:

- $A_{n \times n} = (a_{ii'})$, o fluxo entre as instalações $i \in N$ e $i' \in N$, tal que $a_{ii'} \in \Re^+$, se $i \neq i'$, e $a_{ii'} = 0$, em caso contrário;
- $B_{m \times m} = (b_{jj'})$, a distância entre as localidades $j \in M$ e $j' \in M$, tal que $b_{jj'} \in \Re^+$, se $j \neq j'$, e $b_{jj'} = 0$, em caso contrário;
- $C_{n \times m} = (c_{ij})$, o custo de atribuir a instalação $i \in N$ à localidade $j \in M$, tal que $c_{ij} \in \Re^+$;
- $z \in \Re^+$, um fator de escala denominado custo por unidade de tráfego;
- $q_i \in \Re^+$, a capacidade demandada pela instalação $i \in N$ e
- $Q_j \in \Re^+$, a capacidade da localização $j \in M$.

O problema da atribuição quadrática generalizada (Savelsbergh, 1997; Ross & Soland, 1975; Elloumi et al., 2003; Lee & Ma, 2005; Cordeau et al., 2006; Hahn et al., 2008) consiste em encontrar $X_{n\times m} = (x_{ij})$, com $x_{ij} = \{0, 1\}$, onde a instalação $i \in N$ é atribuída a localidade $j \in M$ se e somente se $x_{ij} = 1$, de modo que as restrições

$$\sum_{j\in M} x_{ij} = 1, \forall i \in N, \tag{3}$$

$$\sum_{i\in N} q_i x_{ij} \leq Q_j, \forall j \in M, \tag{4}$$

$$x_{ij} \in \{0, 1\}, \forall i \in N, \forall j \in M \tag{5}$$

sejam satisfeitas e a função objetivo

$$\sum_{i\in N}\sum_{j\in M} c_{ij}x_{ij} + z\sum_{i\in N}\sum_{j\in M}\sum_{i'\in N, i'\neq i}\sum_{j'\in M} a_{ii'}b_{jj'}x_{ij}x_{i'j'} \tag{6}$$

minimizada. A restrição (3) garante que cada instalação seja atribuída a exatamente uma localidade, enquanto a restrição (4) garante que a capacidade das localidades não sejam violadas.

7.2.1 GRASP: fase construtiva

Suponha uma solução parcial do problema da atribuição quadrática generalizada, ou seja, que um número de atribuições já tenham sido realizadas. Para realizar a próxima atribuição, o procedimento precisa selecionar uma nova instalação e uma localidade. Localidades são disponibilizadas uma de cada vez. O procedimento aleatoriamente determina se é para usar uma nova localidade ou uma localidade previamente escolhida, favorecendo a nova localidade quanto mais as previamente selecionadas tenham capacidades disponíveis praticamente insuficientes. Se o procedimento determina que uma localidade previamente selecionada deva ser escolhida, então selecionam-se as instalações que podem ser atribuídas a localidade de máxima capacidade disponível, para em seguida escolher aleatoriamente uma instalação dentre estas. Das localidades que podem acomodar esta instalação, uma é selecionada aleatoriamente e a atribuição é realizada. Por outro lado, se não existem localidades, dentre as já selecionadas, com capacidade suficiente para acomodar uma instalação, uma nova localidade é selecionada aleatoriamente a partir do conjunto das localidades ainda não selecionadas. Como o todo o procedimento acima descrito não garante a produção de uma solução viável, o mesmo é repetido um número máximo de vezes até terminar com uma atribuição válida, ou com uma solução inviável.

A Figura 2 ilustra este procedimento. Na figura, o conjunto N de instalações é particionado entre o conjunto CF de instalações atribuídas e o conjunto F das instalações não atribuídas. Da mesma forma, o conjunto M das localidades é particionado entre o conjunto CL das localidades previamente selecionadas e o conjunto L das localidades não selecionadas. As instalações em CF devem ser atribuídas as localidades em CL. O subconjunto de instalações T consiste de todas as instalações não atribuídas com demanda menor ou igual à máxima capacidade disponível dentre as localidades em CL. Após uma instalação a partir de T ser aleatoriamente selecionada, o conjunto R consiste de todas as localidades previamente selecionadas capazes de acomodá-lo. Em seguida, uma localidade aleatoriamente selecionada a partir de R recebe esta instalação.

7.2.2 GRASP: busca local aproximada

O procedimento construtivo da Seção 7.2.1 produz uma solução viável p que não é garantida ser localmente ótima. Um procedimento de busca local é aplicado de início a p com o intuito de encontrar um mínimo local aproximado. O procedimento de busca local faz uso de duas estruturas de vizinhança denominadas *1-move* e *2-move*. Uma solução na vizinhança *1-move* de p é obtida pela mudança de uma atribuição instalação-localidade em p. De modo semelhante, uma solução pertencente à vizinhança *2-move* de p se dá pela mudança simultânea de duas atribuições instalação-localidade em p.

Uma maneira de realizar uma busca local nestas vizinhanças consiste em avaliar movimentos na vizinhança *1-move* até encontrar o primeiro movimento que melhore a solução atual. Caso não exista nenhum movimento de melhoria na vizinhança *1-move*, soluções pertencentes à vizinhança *2-move* são avaliadas até também encontrar o primeiro movimento que melhore a solução atual, caso exista. Outra maneira de realizar uma busca local é avaliar todas as soluções na vizinhança *1-move* e *2-move* a fim de se mover para a melhor solução de melhoria, caso exista. Em ambas variantes, a busca é repetida até que nenhuma solução de melhoria na vizinhança exista. Neste trabalho, um enfoque intermediário para busca local foi proposto.

Ao invés de avaliar todas as soluções presentes nas vizinhanças *1-move* e *2-move*, estas são amostradas a fim de povoar uma lista de candidatos com soluções de melhoria. A seguir, uma solução desta lista é selecionada aleatoriamente e um movimento é realizado para aquela solução. A busca é repetida até que não

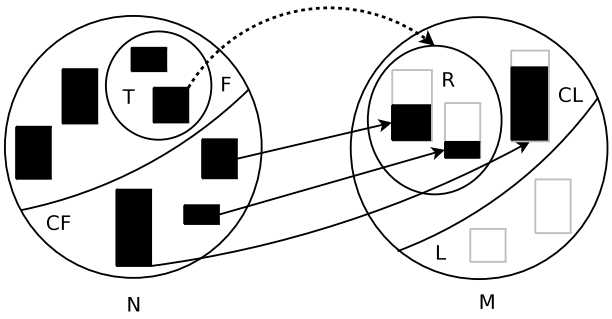

Figura 2. Ilustração de uma etapa da fase construtiva do método GRASP para o problema da atribuição quadrática generalizada.

exista nenhuma solução de melhoria amostrada. Dado que as soluções são amostradas, nem todos os vizinhos podem ser avaliados. Consequentemente, a melhor solução encontrada pode não ser um mínimo local. Por este motivo esta solução é denominada de *mínimo local aproximado*.

O pseudo-código para a busca local aproximada é descrito no Algoritmo 10. O procedimento toma como entrada a solução inicial π e dois parâmetros, *MaxCLS* e *MaxItr*, responsáveis por controlar o processo de amostragem.

Dados: π, *MaxCLS*,*MaxItr*
Resultado: Mínimo local aproximado π
1 **repita**
2 $\quad$ $count \leftarrow 0$; $CLS \leftarrow \emptyset$;
3 $\quad$ **repita**
4 $\quad\quad$ $\pi' \leftarrow$ `Move`(π);
5 $\quad\quad$ **se** π' *é viável* **e** $cost(\pi') < cost(\pi)$ **então**
6 $\quad\quad\quad$ $CLS \leftarrow CLS \cup \{\pi'\}$;
7 $\quad\quad$ **fim**
8 $\quad\quad$ $count \leftarrow count + 1$;
9 $\quad$ **até** $|CLS| \geq MaxCLS$ **or** $count \geq MaxItr$;
10 $\quad$ **se** $CLS \neq \emptyset$ **então**
11 $\quad\quad$ Selecionar aleatoriamente uma solução $\pi \in CLS$;
12 $\quad$ **fim**
13 **até** $CLS = \emptyset$;
14 **retorna** π;

Algoritmo 10: Pseudo-código da busca local aproximada do GRASP para o problema da atribuição quadrática generalizada.

O *loop* entre as linhas 1 e 13 é repetido até um mínimo local aproximado ser produzido. Na linha 2, o contador da amostragem *count* e a lista de candidatos *CLS* são inicializadas. A cada iteração do loop interno situado entre as linhas 3 e 9, as vizinhanças *1-move* e *2-move* de π são amostradas sem reposição pelo procedimento `Move`(π) da linha 4. Se este vizinho é uma solução viável de melhoria, ele é inserido em *CLS* como descrito na linha 6. Este procedimento é repetido até que a lista de candidatos esteja cheia, ou até que um número máximo de vizinhos tenha sido amostrado. Entre as linhas 10 e 12, se a lista de candidato não estiver vazia, uma atribuição $\pi \in CLS$ é aleatoriamente escolhida. Se o conjunto *CLS* estiver vazio após o processo de amostragem, então o procedimento termina retornando π como um mínimo local aproximado (ver linha 14). Caso contrário, o procedimento se move para outra solução em *CLS*, repetindo o loop mais externo.

7.2.3 Religamento de caminhos

Motivado pelo fato de que um simples movimento de uma solução x na direção da solução alvo x_t não garante a viabilidade da nova solução construída, uma nova variante de religamento de caminhos foi proposta por Mateus et al. (2011).

Suponha que dentre as diferenças entre x e x_t encontra-se a localidade atribuída à instalação f. Em outras palavras, enquanto a localidade atribuída a f em x_t é l, a localidade atribuída a f em x é u, com $l \neq u$. Neste

caso, não é necessariamente viável realizar um movimento em x que atribua f a l. Se a capacidade Q_l não é violada, então a nova solução é viável. Caso contrário, um procedimento de reparo deve ser aplicado para tentar tornar a solução viável, como descrito na Figura 3.

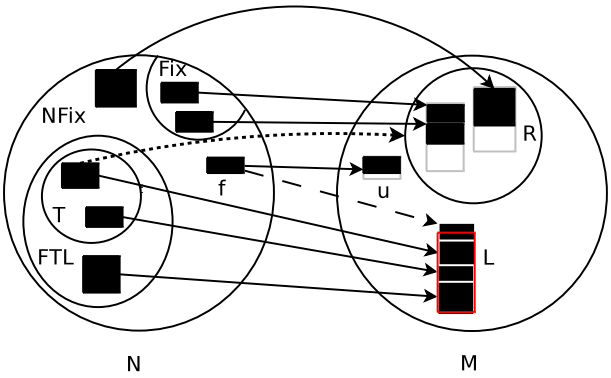

Figura 3. Reparo no religamento de caminhos do GRASP para o problema da atribuição quadrática generalizada.

Neste procedimento de reparo, um conjunto FTL de instalações é criado com todas as instalações ainda não fixadas atribuídas a localidade l cuja capacidade foi violada. Em seguida, o conjunto $T \subseteq FTL$ é construído com todas as instalações em FTL cujas demandas são menores ou iguais a capacidade máxima disponível dentre as localidades em M. Após uma instalação de T ser aleatoriamente selecionada, o conjunto R consiste das localidades em M capazes de acomodar esta instalação. Depois uma localidade é selecionada a partir do conjunto R e a instalação é atribuída a ela. Este processo é repetido até que a capacidade da localidade l tenha uma folga não negativa.

Seja π uma solução intermediária obtida durante o religamento de caminhos de π_s a π_t, como ilustrado na Figura 3. O conjunto de instalações N é particionado em dois conjuntos, Fix com as instalações fixadas e $nonFix$ com as instalações restantes. Seja $f \in nonFix$ uma instalação cuja localidade em π_t é l e em π é $u \neq l$, onde $l, u \in M$.

De acordo com a Figura 3, realizar um movimento em π que atribui a instalação f na localidade l é inviável, desde que a capacidade Q_l da localidade l é insuficiente para acomodar esta instalação. Neste caso, o conjunto de instalações $FTL \subseteq nonFix$ é criado com todas as instalações não fixadas atribuídas a l em π. Em seguida, o conjunto $T \subseteq FTL$ é construído com todas as instalações em FTL com demandas no máximo igual à máxima capacidade disponível entre as localidades em M. Seja i a instalação aleatoriamente selecionada a partir de T e R o conjunto com as localidades de M capazes de acomodar a instalação i. Depois, uma localidade j é selecionada aleatoriamente a partir do conjunto R e a instalação i é atribuída a ela. Este processo é repetido até que a capacidade da localidade l tenha uma folga não negativa.

Sabe-se que o processo de religamento de caminhos é uma sequência de passos de x_s para x_t. A cada passo, um movimento é realizado da solução atual x com ou sem reparo. Depois, uma instalação i é selecionada aleatoriamente a partir do conjunto composto de todas as instalações ainda não fixadas corrigidas no passo. Uma instalação é dita *corrigida* quando a localidade a qual foi atribuída torna-se a mesma atribuída a ela na solução alvo x_t. Após a instalação i ser fixada, o próximo passo é iniciado. Este processo continua até a solução alvo x_t ser alcançada ou quando nenhuma solução viável é obtida de x.

Este religamento de caminhos é diferente da variante pardão porque dado as soluções x_s e x_t, seus elementos comuns não são mantidos fixos a priori, de modo a uma pequena porção do espaço de soluções gerado pelos elementos restantes ser explorada. Nesta nova variante, uma instalação é fixada por vez a cada passo do religamento de caminhos.

8. 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 abordado neste capítulo é o GRASP paralelo, além de exemplos de aplicações da meta-heurística GRASP nas áreas de lógica e atribuição. O leitor interessado é direcionado ao artigo de Resende & Ribeiro (2005b).

Agradecimentos

A pesquisa de Ricardo M.A Silva foi parcialmente realizada durante seu pós-doutoramento na AT&T Labs Research em Florham Park, New Jersey, EUA; e parcialmente financiada pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), e pela Pró-reitoria de Pesquisa da Universidade Federal de Pernambuco (PROPESQ).

Referências

Abdinnour-Helm, S. & Hadley, S., Tabu search based heuristics for multi-floor facility layout. *International Journal of Production Research*, 38:365–383, 2000.

Ahuja, R.K.; Ergun, O.; Orlin, J.B. & Punnen, A.P., A survey of very large-scale neighborhood search techniques. *Discrete and Applied Mathematics*, 123:75–102, 2002.

Ahuja, R.K.; Orlin, J.B. & Sharma, D., Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming, Ser A*, 91:71–97, 2001.

Aiex, R.; Pardalos, P.; Resende, M. & Toraldo, G., GRASP with path-relinking for three-index assignment. *INFORMS Journal on Computing*, 17:224–247, 2005.

Aiex, R.M.; Resende, M.G.C. & Ribeiro, C.C., Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373, 2002.

Andrade, D. & Resende, M., A GRASP for PBX telephone migration scheduling. In: *Proceedings of The Eighth INFORMS Telecommunications Conference*. 2006.

Battiti, R. & Protasi, M., Approximate algorithms and heuristics for the MAX-SAT. In: Du, D. & Pardalos, P. (Eds.), *Handbook of Combinatorial Optimization*. Boston, USA: Kluwer Academic Publishers, v. 1, p. 77–148, 1998.

Baum, E.B., *Iterated descent: A better algorithm for local search in combinatorial optimization problems*. Technical Report, California Institute of Technology, Pasadena, USA, 1986.

Baum, E.B., Towards practical 'neural' computation for combinatorial optimization problems. In: *AIP Conference Proceedings 151 on Neural Networks for Computing*. Woodbury, USA: American Institute of Physics Inc., p. 53–58, 1987.

Baxter, J., Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32:815–819, 1981.

Beltrán, J.D.; Calderón, J.E.; Cabrera, R.J.; Pérez, J.A.M. & Moreno-Vega, J.M., GRASP/VNS hybrid for the strip packing problem. In: *Proceedings of Hybrid Metaheuristics*. p. 79–90, 2004.

Bresina, J.L., Heuristic-biased stochastic sampling. In: *Proceedings of the AAAI-96*. p. 271–278, 1996.

Canuto, S.; Resende, M. & Ribeiro, C., Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

Charon, I. & Hudry, O., The noising method: A new method for combinatorial optimization. *Operations Research Letters*, 14:133–137, 1993.

Charon, I. & Hudry, O., The noising methods: A survey. In: Ribeiro, C.C. & Hansen, P. (Eds.), *Essays and Surveys in Metaheuristics*. Boston, USA: Kluwer Academic Publishers, p. 245–261, 2002.

Cook, S.A., The complexity of theorem-proving procedures. In: *Proceedings of the Third annual ACM Symposium on Theory of Computing*. p. 151–158, 1971.

Cordeau, J.F.; Gaudioso, M.; Laporte, G. & Moccia, L., A memetic heuristic for the generalized quadratic assignment problem. *INFORMS Journal on Computing*, 18:433–443, 2006.

de la Peña, M.G.B., Heuristics and metaheuristics approaches used to solve the rural postman problem: A comparative case study. In: *Proceedings of the Fourth International ICSC Symposium on Engineering of Intelligent Systems*. 2004.

Delmaire, H.; Díaz, J.A.; Fernández, E. & Ortega, M., Reactive GRASP and Tabu Search based heuristics for the single source capacitated plant location problem. *INFOR*, 37(3):194–225, 1999.

Drummond, L.; Vianna, L.S.; Silva, M.B. & Ochi, L.S., Distributed parallel metaheuristics based on GRASP and VNS for Solving the traveling purchaser problem. In: *Proceedings of the Ninth International Conference on Parallel and Distributed Systems*. p. 257–266, 2002.

Duarte, A.; Ribeiro, C.C. & Urrutia, S., A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. In: Bartz-Bielstein, T.; Aguilera, M.J.B.; Blum, C.; Naujoks, B.; Role, A.; Rudolph, G. & Sampels, M. (Eds.), *Hybrid Metaheuristics*. v. 4771 de *LNCS*, p. 82–95, 2007a.

Duarte, A.R.; Ribeiro, C.C.; Urrutia, S. & Haeusler, E.H., Referee assignment in sports leagues. In: Burke, E.K. & Rudova, H. (Eds.), *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling*. v. 3867 de *LNCS*, p. 158–173, 2007b.

Elloumi, S.; Roupin, F. & Soutif, E., *Comparison of Different Lower Bounds for the Constrained Module Allocation Problem*. Technical Report CEDRIC-03-473, CNAM, CEDRIC Laboratory, Paris, France, 2003.

Feo, T.A. & Resende, M.G.C., A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

Feo, T.A. & Resende, M.G.C., Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

Festa, P.; Pardalos, P.; Resende, M. & Ribeiro, C., Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

Festa, P.; Pardalos, P.M.; Pitsoulis, L.S. & Resende, M.G.C., GRASP with path-relinking for the weighted MAXSAT problem. *Journal of Experimental Algorithmics*, 11, 2007Article 2.4: 1-16.

Festa, P. & Resende, M., GRASP: An annotated bibliography. In: Ribeiro, C. & Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, p. 325–367, 2002.

Festa, P. & Resende, M., An annotated bibliography of GRASP, Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24, 2009a.

Festa, P. & Resende, M., An annotated bibliography of GRASP, Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009b.

Festa, P. & Resende, M.G.C., GRASP: An annotated bibliography. In: Hansen, P. & Ribeiro, C.C. (Eds.), *Essays and Surveys on Metaheuristics*. Boston, USA: Kluwer Academic Publishers, 2001.

Fleurent, C. & Glover, F., Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11:198–204, 1999.

Garey, M.R. & Johnson, D.S., *Computers and intractability: A guide to the theory of NP-completeness*. New York, USA: W.H. Freeman and Company, 1979.

Geng, Y.; Li, Y. & Lim, A., A very large-scale neighborhood search approach to capacitated warehouse routing problem. In: *17th IEEE International Conference on Tools with Artificial Intelligence*. IEEE Computer Society, p. 8, 2005.

Glover, F., Tabu search – Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

Glover, F., Tabu search – Part II. *ORSA Journal on Computing*, 2:4–32, 1990.

Glover, F., Tabu search and adaptive memory programing – Advances, applications and challenges. In: Barr, R.S.; Helgason, R.V. & Kennington, J.L. (Eds.), *Interfaces in Computer Science and Operations Research*. Kluwer Academic, p. 1–75, 1996.

Glover, F., Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In: Laguna, M. & Gonzáles-Velarde, J.L. (Eds.), *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*. Kluwer Academic, p. 1–24, 2000.

Glover, F. & Laguna, M., *Tabu Search*. Kluwer Academic, 1997.

Glover, F.; Laguna, M. & Martí, R., Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684, 2000.

Goemans, M.X. & Williamson, D.P., The primal dual method for approximation algorithms and its application to network design problems. In: Hochbaum, D. (Ed.), *Approximation algorithms for NP-hard problems*. Boston, USA: PWS Publishing Co., p. 144–191, 1996.

Hahn, P.M.; Kim, B.J.; Guignard, M.; MacGregor Smith, J. & Zhu, Y.R., An algorithm for the generalized quadratic assignment problem. *Computational Optimization and Applications*, 40(3):351–372, 2008.

Hansen, P. & Mladenović, N., An introduction to variable neighborhood search. In: Voss, S.; Martello, S.; Osman, I.H. & Roucairol, C. (Eds.), *Meta-heuristics, Advances and trends in local search paradigms for optimization*. Dordrecht, The Netherlands: Kluwer Academic Publishers, p. 433–458, 1998.

Hart, J.P. & Shogan, A.W., Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

Johnson, D.S., Local optimization and the traveling salesman problem. In: *Proceedings of the 17th Colloquium on Automata*. Berlin, Germany: Springer-Verlag, v. 443 de *LNCS*, p. 446–461, 1990.

Johnson, D.S.; Papadimitriou, C.H. & Yannakakis, M., How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

Kirkpatrick, S.; Gelatt, Jr., C. & Vecchi, M., Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Krentel, M., The complexity of optimization problems. *Journal of Computer and System Sciences*, 36, 1988.

Laguna, M. & González-Velarde, J.L., A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing*, 2:253–260, 1991.

Laguna, M. & Martí, R., GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

Lee, C.G. & Ma, Z., *The generalized quadratic assignment problem*. Technical Report MIEOR TR2005-01, Department of Mechanical and Industrial Engineering at the University of Toronto, 2005.

Li, Y.; Pardalos, P. & Resende, M., A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos, P. & Wolkowicz, H. (Eds.), *Quadratic assignment and related problems*. New Providence, USA: American Mathemaical Society, v. 16 de *DIMACS Series in Discrete Matehematics and Theoretical Computer Science*, p. 237–261, 1994.

Li, Z.; Guo, S.; Wang, F. & Lim, A., Improved GRASP with tabu search for vehicle routing with both time window and limited number of vehicles. In: Orchard, B.; Yang, C. & Ali, M. (Eds.), *Innovations in Applied Artificial Intelligence – Proceedings of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Heidelberg, Germany: Springer-Verlag, v. 3029 de *LNCS*, p. 552–561, 2004.

Lim, A. & Wang, F., A smoothed dynamic tabu search embedded GRASP for m-VRPTW. In: *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*. Boca Raton, USA, p. 704–708, 2004.

Liu, X.; Pardalos, P.; Rajasekaran, S. & Resende, M., A GRASP for frequency assignment in mobile radio networks. In: Badrinath, B.; Hsu, F.; Pardalos, P. & Rajasejaran, S. (Eds.), Mobile Networks and Computing. American Mathematical Society, v. 52 de *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, p. 195–201, 2000.

Martin, O. & Otto, S., Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.

Martin, O.; Otto, S. & Felten, E., Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.

Martins, S.; Resende, M.; Ribeiro, C. & Pardalos, P., A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 17:267–283, 2000.

Martins, S.L.; Pardalos, P.M.; Resende, M.G.C. & Ribeiro, C.C., Greedy randomized adaptive search procedures for the steiner problem in graphs. In: Pardalos, P.M.; Rajasejaran, S. & Rolim, J. (Eds.), *Randomization Methods in Algorithmic Design*. American Mathematical Society, v. 43 de *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, p. 133–145, 1999.

Martins, S.L.; Ribeiro, C.C. & Souza, M.C., A parallel GRASP for the Steiner problem in graphs. In: Ferreira, A. & Rolim, J. (Eds.), *Proceedings of 5th International Symposium on Solving Irregularly Structured Problems in Parallel*. Heidelberg, Germany: Springer-Verlag, v. 1457 de *LNCS*, p. 285–297, 1998.

Mateus, G.; Resende, M. & Silva, R., GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics*, 17:527–565, 2011.

Moura, A. & Oliveira, J., A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 20:50–57, 2005.

Murphey, R.A.; Pardalos, P.M. & Pitsoulis, L.S., A parallel GRASP for the data association multidimensional assignment problem. In: Pardalos, P.M. (Ed.), *Parallel Processing of Discrete Problems*. New York, USA: Springer-Verlag, v. 106 de *The IMA Volumes in Mathematics and Its Applications*, p. 159–180, 1998.

Nascimento, M.C.V.; Resende, M.G.C. & Toledo, F.M.B., GRASP with path-relinking for the multi-plant capacitated plot sizing problem. *European Journal of Operational Research*, 200(3):747 – 754, 2010.

Ochi, L.S.; Silva, M.B. & Drummond, L., GRASP and VNS for solving traveling purchaser problem. In: *Proceedings of The Fourth Metaheuristics International Conference*. Porto, Portugal, p. 489–494, 2001.

Oliveira, C.A.; Pardalos, P.M. & Resende, M.G.C., GRASP with path-relinking for the quadratic assignment problem. In: Ribeiro, C.C. & Martins, S.L. (Eds.), *Proceedings of III Workshop on Efficient and Experimental Algorithms*. Heidelberg, Germany: Springer-Verlag, v. 3059 de *LNCS*, p. 356–368, 2004a.

Oliveira, C.A.S.; Pardalos, P.M. & Resende, M.G.C., GRASP with path-relinking for the quadratic assignment problem. In: Ribeiro, C.C. & Martins, S.L. (Eds.), *Efficient and Experimental Algorithms*. Heidelberg, Germany: Springer-Verlag, v. 3059 de *LNCS*, p. 356–368, 2004b.

Pardalos, P.M.; Pitsoulis, L.S. & Resende, M.G.C., A parallel GRASP implementation for the quadratic assignment problem. In: Ferreira, A. & Rolim, J. (Eds.), *Parallel Algorithms for Irregularly Structured Problems*. Boston, USA: Kluwer Academic Publishers, p. 115–130, 1995.

Pardalos, P.M.; Pitsoulis, L.S. & Resende, M.G.C., A parallel GRASP for MAX-SAT problems. In: Wasniewski, J.; Dongarra, J.; Madsen, K. & Olesen, D. (Eds.), *Applied Parallel Computing Industrial Computation and Optimization*. Heidelberg, Germany: Springer-Verlag, v. 1184 de *LNCS*, p. 575–585, 1996.

Pardalos, P.M.; Rendl, F. & Wolkowicz, H., The quadratic assignment problem: A survey and recent development. In: Pardalos, P.M. & Wolkowicz, H. (Eds.), *The Quadratic Assignment and Related Problems*. New Providence, USA: American Mathemaical Society, v. 16 de *DIMACS Series in Discrete Matehematics and Theoretical Computer Science*, p. 1–42, 1994.

Pitsoulis, L.S. & Resende, M.G.C., Greedy randomized adaptive search procedures. In: Pardalos, P.M. & Resende, M.G.C. (Eds.), *Handbook of Applied Optimization*. Oxford, UK: Oxford University Press, p. 168–183, 2002.

Prais, M. & Ribeiro, C., Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176, 2000.

Pu, G.G.; Chong, Z.; Qiu, Z.Y.; Lin, Z.Q. & He, J.F., A hybrid heuristic algorithm for HW-SW partitioning within timed automata. In: *Proceedings of Knowledge-based Intelligent Information and Engineering Systems*. Heidelberg, Germany: Springer-Verlag, v. 4251 de *LNAI*, p. 459–466, 2006.

Resende, M.G.C., Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures. In: Chen, Z.L. & Raghavan, S. (Eds.), *Tutorials in Operations Research*. INFORMS, p. 295–319, 2008.

Resende, M.G.C.; Martí, R.; Gallego, M. & Duarte, A., GRASP and path relinking for the max-min diversity problem. *Computers and Operations Research*, 37(3):498 – 508, 2010a.

Resende, M.G.C. & Ribeiro, C.C., Greedy randomized adaptive search procedures. In: Glover, F. & Kochenberger, G. (Eds.), *Handbook of Metaheuristics*. Kluwer Academic Publishers, p. 219–249, 2003.

Resende, M.G.C. & Ribeiro, C.C., GRASP with path-relinking: Recent advances and applications. In: Ibaraki, T.; Nonobe, K. & Yagiura, M. (Eds.), *Metaheuristics: Progress as Real Problem Solvers*. Berlin, Germany: Springer, p. 29–63, 2005a.

Resende, M.G.C. & Ribeiro, C.C., Parallel Greedy Randomized Adaptive Search Procedures. In: Alba, E. (Ed.), *Parallel Metaheuristics: A new class of algorithms*. New York, USA: John Wiley & Sons, p. 315–346, 2005b.

Resende, M.G.C.; Ribeiro, C.C.; Glover, F. & Marti, R., Scatter search and path-relinking: Fundamentals, advances, and applications. In: Gendreau, M. & Potvin, J.Y. (Eds.), *Handbook of Metaheuristics*. New York, USA: Springer, v. 146 de *Operations Research & Management Science*, p. 87–107, 2010b.

Resende, M.G.C. & Werneck, R.F., A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10:59–88, 2004.

Resende, M.G.C. & Werneck, R.F., A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174:54–68, 2006.

Ribeiro, C. & Souza, M., Variable neighborhood search for the degree constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 118:43–54, 2002.

Ribeiro, C. & Urrutia, S., Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 127:775–787, 2007.

Ribeiro, C. & Vianna, D., A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:325–338, 2005.

Ribeiro, C.C. & Rosseti, I., A parallel GRASP for the 2-path network design problem. In: Monien, B. & Feldmann, R. (Eds.), *Proceedings of the 8th International Euro-Par Conference on Parallel Processing*. Heidelberg, Germany: Springer-Verlag, v. 2004 de *LNCS*, p. 922–926, 2002.

Ribeiro, C.C.; Uchoa, E. & Werneck, R.F., A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–246, 2002.

Ross, G.T. & Soland, R.M., A branch-and-bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8:91–103, 1975.

Savelsbergh, M., A branch-and-price algorithm for the generalized assignment problem. *Operational Research*, 45:831–841, 1997.

Serra, D. & Colomé, R., Consumer choice and optimal location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.

de Souza, M.C.; Duhamel, C. & Ribeiro, C.C., A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In: Resende, M.G.C. & de Sousa, J.P. (Eds.), *Metaheuristics: Computer Decision-making*. Kluwer Academic Publishers, p. 627–658, 2003.

Souza, M.J.F.; Maculan, N. & Ochi, L.S., A GRASP-tabu search algorithm to solve a school timetabling problem. In: *Proceedings of The Fourth Metaheuristics International Conference*. Porto, Portugal, p. 53–58, 2001.

Takahashi, H. & Matsuyama, A., An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

Verhoeven, M.G.A. & Aarts, E.H.L., Parallel local search. *Journal of Heuristics*, 1:43–66, 1995.

Notas Biográficas

Mauricio Guilherme de Carvalho Resende é graduado em Engenharia Elétrica (PUC-Rio, 1978), mestre em Pesquisa Operacional (Georgia Institute of Technology, 1979) e doutor em Pesquisa Operacional (University of California, Berkeley, 1987). Atualmente está no AT & T Labs Research, Algorithms and Optimization Research Department. E-mail: mgcr@research.att.com.

Ricardo Martins de Abreu Silva é graduado, mestre e doutor em Ciência da Computação (Universidade Federal de Pernambuco, 1995, 1998 e 2003, respectivamente). Atualmente é professor do Centro de Informática da Universidade Federal de Pernambuco .