Capítulo 2

PEGUE COM RELINKING DE CAMINHO: AVANÇOS RECENTES E FORMULÁRIOS

Mauricio GC Resende [1] e Celso C. Ribeiro [2]

[1] Internet and Network Systems Research, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 EUA.
mgcr@research.att.com

[2] Departamento de Ciência da Computação, Universidade Católica do Rio de Janeiro, Rua Marqu Ruaês de SãSão Vicente, 225, Rio de Janeiro, RJ 22453-900 Brasil
celso@inf.puc-rio.br

Abstrato: A reconexão de caminhos é um grande aprimoramento do procedimento básico de busca adaptativa aleatória gananciosa (GRASP), levando a melhorias significativas no tempo e na qualidade da solução. A reconexão de caminhos adiciona um mecanismo de memória ao GRASP, fornecendo uma estratégia de intensificação que explora as trajetórias que conectam as soluções GRASP e as melhores soluções de elite produzidas anteriormente durante a pesquisa. Este artigo analisa os avanços e aplicações recentes do GRASP com reconexão de caminhos. Uma breve revisão do GRASP é fornecida. Isso é seguido por uma descrição da reconexão de caminhos e como ela é incorporada ao GRASP. Diversas aplicações recentes de GRASP com reconexão de caminhos são revisadas. O artigo conclui com uma discussão de extensões para esta estratégia, em particular, implementações paralelas e aplicações de reconexão de caminhos com outras metaheurísticas.

Palavras-chave: Metaheurísticas, GRASP, reconexão de caminhos.

2.1 INTRODUÇÃO

APERTO (Greedy Randomized Adaptive Search Procedure) é uma metaheurística para encontrar soluções aproximadas para problemas de otimização combinatória formulados como

$$\min f(x) \text{ sujeito a } x \in X,$$

Onde f (·) é uma função objetivo a ser minimizada e X é um conjunto discreto de soluções viáveis. Ele foi introduzido pela primeira vez por Feo e Resende [14] em um artigo que descreve uma heurística probabilística para cobertura de conjuntos. Desde então, GRASP tem experimentado um desenvolvimento contínuo [15, 43] e tem sido aplicado em uma ampla gama de áreas problemáticas [18].

2.1.1 Pesquisa local multi-start

GRASP pode ser pensado como um método de pesquisa que aplica repetidamente a pesquisa local de diferentes soluções iniciais em X. Em cada etapa da pesquisa local, a vizinhança N (x) da solução atual x é procurado por uma solução y ∈ N (x) de tal modo que f (y) <f (x). Se tal solução de melhoria for encontrada, torna-se a solução atual e outra etapa de busca local é realizada. Se nenhuma solução de melhoria for encontrada, o procedimento para com x como uma solução localmente ideal.

Uma solução inicial óbvia para pesquisa local é uma solução gerada por um algoritmo ganancioso. Algoritmos gananciosos constroem uma solução, um elemento de cada vez. Por exemplo, uma árvore é construída uma aresta de cada vez; uma programação é construída uma operação por vez; uma partição de vértice é construída um vértice de cada vez. Em cada etapa de um algoritmo ganancioso, um conjunto de elementos candidatos C contém todos os elementos que podem ser adicionados para estender a solução parcial. Algoritmos gananciosos fazem uso de uma função gananciosa g (e) que mede o custo incremental de adicionar elemento e ∈ C para a solução parcial atual. Para um problema de minimização, o elemento $e^* = \text{argmin}\{ g(e): e \in C\}$ é escolhido para ser adicionado à solução parcial. A adição de e^* à solução parcial geralmente restringe o conjunto de elementos candidatos, o que é refletido pela redução de sua cardinalidade. O procedimento termina quando uma solução completa é construída, ou seja, quando C = ∅.

A desvantagem de usar um algoritmo guloso como solução inicial para pesquisa local é que, se uma regra determinística for usada para quebrar laços, um algoritmo guloso produzirá uma única solução e, portanto, a pesquisa local só pode ser aplicada uma vez. Mesmo quando uma regra de desempate probabilística é usada, a diversidade de soluções puramente gananciosas geralmente é baixa.

O outro extremo é iniciar repetidamente a pesquisa local a partir de soluções geradas aleatoriamente. Embora essa abordagem produza um alto nível de diversidade nas soluções iniciais, a qualidade média dessas soluções aleatórias é geralmente muito pior do que a de uma solução gananciosa. Além disso, o tempo que a pesquisa local leva para convergir para uma solução ótima local é, em média, muito mais longo do que quando uma solução inicial gananciosa é usada.

GRASP combina construção gananciosa e aleatória usando ganância para construir uma lista de candidatos restrita (RCL) e aleatoriedade para selecionar

um elemento da lista, ou usando aleatoriedade para construir a lista e avidez para seleção. Elementos candidatos e ∈ C são classificados de acordo com seu valor de função gananciosa g (e). Em um RCL baseado em cardinalidade, o último é composto pelo k os elementos mais bem classificados. Em uma construção baseada em valor ção, o RCL consiste nos elementos do conjunto { e ∈ CG ∗ ≤ g (e) ≤ g ∗ + α · (g ∗ - g ∗)}, Onde g ∗ = min { g (e): e ∈ CG ∗ = max { g (e): e ∈ C}, e α é um parâmetro que satisfaz 0 ≤ α ≤ 1. Uma vez que o melhor valor para α é frequentemente difícil de determinar, geralmente é atribuído um valor aleatório para cada iteração GRASP.

O algoritmo 2.1 mostra o pseudocódigo para um procedimento de busca adaptativa aleatória gananciosa pura. O valor da melhor solução é armazenado em f ∗ e eu max As iterações GRASP são feitas. Cada iteração consiste em uma fase de construção aleatória gananciosa, seguida por uma fase de pesquisa local, começando com a solução aleatória gananciosa. Se a solução resultante da busca local melhorar a melhor solução até o momento, ela é armazenada em x ∗.

```
Dados : Número de iterações eu max
Resultado: Solução x ∗ ∈ X
f ∗ ← ∞;
para i = 1,. . . , eu max Faz
    x ← GreedyRandomizedConstruction ();
    x ← LocalSearch ( x);
    E se f (x) <f ∗ então
        f ∗ ← f (x);
        x ∗ ← x;
    fim
fim
```

Algoritmo 2.1: Um GRASP básico para minimização.

A Figura 2.1 exibe os resultados para uma instância do problema de cobertura máxima [36], mostrando a distribuição dos valores da função objetivo para a fase de construção e a fase de busca local de um algoritmo multi-start puramente aleatório (seguido por busca local) e um GRASP com o parâmetro α fixado em 0,85. Em ambos os gráficos, as iterações foram classificadas pelo valor da função objetivo da solução encontrada pela pesquisa local. Os gráficos mostram que a construção GRASP atinge uma diversidade razoável em termos de valores de solução, ao mesmo tempo em que produz soluções iniciais para busca local que têm valores de função objetivo muito melhores. Os valores da função objetivo situam-se em torno de 3,5 para a construção aleatória e 9,7 para a construção GRASP, enquanto os valores obtidos pela pesquisa local rondam os 9,9. Consequentemente, os tempos de busca local são muito menores para GRASP do que para o algoritmo multi-start puramente aleatório.

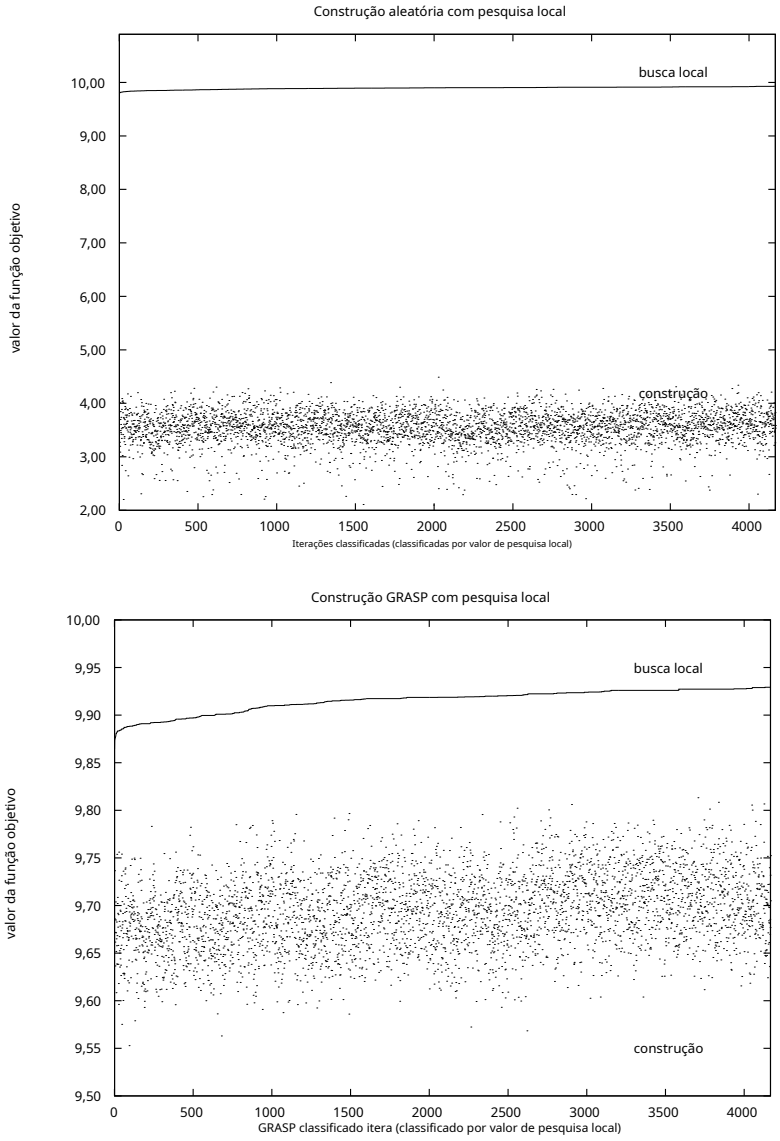

Figura 2.1. Multi-start aleatório vs. GRASP em uma instância de problema de cobertura máxima.

A Figura 2.2 mostra, com resultados para 100 execuções na mesma instância de um problema de máxima satisfação, o benefício de usar GRASP em vez de reiniciar repetidamente a pesquisa local com uma solução gerada aleatoriamente e uma solução gananciosa. Duas curvas comparam o valor da função objetivo (melhor e média ao longo das 100 execuções) para diferentes valores do RCL

parâmetro α. Duas outras curvas comparam os tempos de solução (tempo médio total e tempo médio de pesquisa local) para diferentes valores de α. Uma vez que este é um problema de maximização, α = 0 corresponde à construção aleatória, enquanto α = 1 corresponde à construção gananciosa. Enquanto a solução média melhora à medida que passamos de aleatório para ganancioso, a melhor solução (no que estamos realmente interessados) melhora à medida que nos afastamos da construção aleatória, mas atinge o máximo antes de atingir α = 1 e, em seguida, diminui depois disso. À medida que a solução média aumenta, a difusão das soluções diminui. A combinação do aumento no valor médio da solução e a presença de spread suficiente contribuem para produzir a melhor solução com α =. 8. O tempo de solução diminui à medida que se passa de aleatório para ganancioso, principalmente devido à diminuição do tempo para pesquisa local.

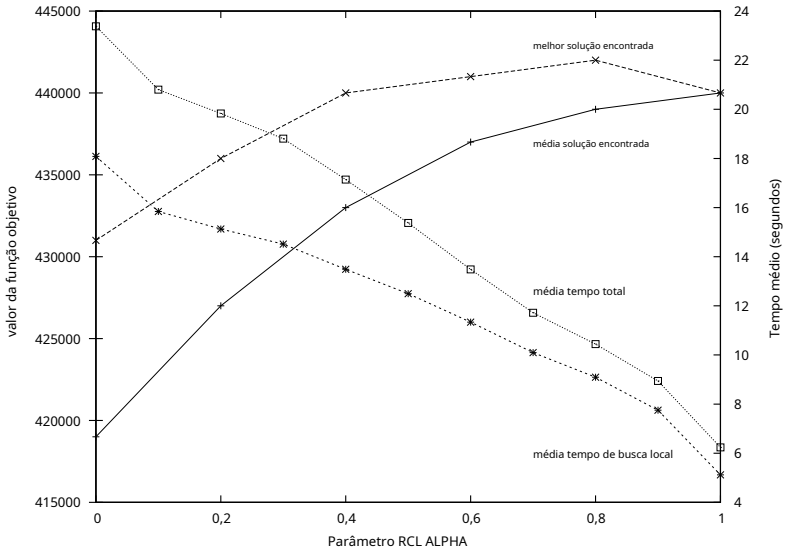

Figura 2.2. Média versus melhor solução encontrada e tempo total de execução versus tempo de pesquisa local como uma função do parâmetro RCL α em 100 é executado em uma instância de máxima satisfação.

2.1.2 Mecanismos baseados em memória

Se iteração GRASP eu usa a semente do gerador de número aleatório $s_{eu,}$ então as iterações são sem memória, ou seja, elas produzem o mesmo resultado independendo da ordem em que são executados. No restante desta seção, revisamos como o uso da memória foi introduzido no GRASP.

A memória pode ser usada para evitar trabalho redundante. Por exemplo, pode-se armazenar em uma tabela hash todas as soluções construídas e usadas como

soluções para pesquisa local [30]. Cada vez que uma nova solução é construída, ela só será usada como uma solução inicial na fase de busca local se não estiver presente na tabela hash.

A filtragem de soluções construídas [16, 30, 34] evita a aplicação de pesquisa local a soluções de baixa qualidade, onde a pesquisa local provavelmente levará muito tempo para convergir para um ótimo local de baixa qualidade.

Fleurent e Glover [19] introduziram um mecanismo de memória de longo prazo na construção GRASP que faz uso de um conjunto de soluções de elite encontradas durante as iterações GRASP. Seu mecanismo favorece variáveis (fortemente determinadas) que não podem ser alteradas sem corroer o objetivo ou alterar significativamente outras variáveis e variáveis (consistentes) que recebem um determinado valor em grande parte das soluções de elite.

Prais e Ribeiro [34] introduziram outro mecanismo de aprendizagem para a construção GRASP, que chamaram reativo APERTO. Lembre-se de que em uma lista de candidatos restritos com base em valores, um parâmetro α determina o nível de aleatoriedade ou ganância usado para compor o RCL. Em vez de usar um valor fixo para α, GRASP reativo seleciona um valor, aleatoriamente, de um conjunto discreto de valores $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Cada valor α_i tem associado com isso uma probabilidade p_i que será selecionado ($\sum_{i=1}^{m} p_i = 1$). O A ideia no GRASP reativo é mudar essas probabilidades à medida que as iterações prosseguem, para favorecer os valores que levaram a melhores soluções nas iterações GRASP anteriores.

Laguna e Martˊi [28] introduziram outra estratégia para o uso de memória de longo prazo que consiste em um conjunto de soluções de elite. A cada iteração GRASP, esta estratégia combina a solução GRASP com uma solução elite selecionada aleatoriamente, usando path-relinking [23]. Este é o assunto da próxima seção.

2.2 PATH-RELINKING

Path-relinking foi originalmente proposto por Glover [23] como uma estratégia de intensificação explorando trajetórias conectando soluções de elite obtidas por tabu search ou scatter search [24-26]. Partindo de uma ou mais soluções de elite, caminhos no espaço de soluções que levam a outras soluções de elite são gerados e explorados na busca por melhores soluções. Para gerar caminhos, movimentos são selecionados para introduzir atributos na solução atual que estão presentes na solução de orientação de elite. A reconexão de caminhos pode ser vista como uma estratégia que busca incorporar atributos de soluções de alta qualidade, favorecendo esses atributos nas jogadas selecionadas.

O algoritmo 2.2 ilustra o pseudo-código do processo de religação de caminho dure aplicado a um par de soluções x_s (solução inicial) e x_t (solução-alvo).

```
Dados : Solução inicial x_s e solução alvo x_t
Resultado: Melhor solução x* no caminho de x_s para x_t
Calcular a diferença simétrica Δ(x_s, x_t);
f* ← min{f(x_s), f(x_t)};
x* ← argmin{f(x_s), f(x_t)};
x ← x_s;
enquanto Δ(x, x_t) 6= ∅ Faz
    m* ← argmin{f(x ⊕ milímetros ∈ Δ(x, x_t)};
    Δ(x ⊕ m*, x_t) ← Δ(x, x_t) \ {m*};
    x ← x ⊕ m*;
    E se f(x) <f* então
        f* ← f(x);
        x* ← x;
    fim
fim
```

Algoritmo 2.2: Revinculação de caminhos.

O procedimento começa calculando a diferença simétrica $\Delta(x_s, x_t)$ entre as duas soluções, ou seja, o conjunto de movimentos necessários para alcançar x_t (solução alvo) de x_s (solução inicial). Um caminho de soluções é gerado ligando x_s e x_t. A melhor solução x^* neste caminho é retornado pelo algoritmo. Em cada etapa, o procedimento examina todos os movimentos $m \in \Delta(x, x_t)$ da solução atual x e seleciona aquele que resulta no mínimo solução de custo, ou seja, aquela que minimiza $f(x \oplus m)$, Onde $x \oplus m$ é a solução resultante da aplicação de movimento m para solução x. A melhor jogada m^* é feito, produzindo solução $x \oplus m^*$. O conjunto de movimentos disponíveis é atualizado. Se necessário, a melhor solução x^* é atualizada. O procedimento termina quando x_t é alcançado, ou seja, quando $\Delta(x, x_t) = \emptyset$.

Notamos que a reconexão de caminhos também pode ser vista como uma restrição estratégia de busca local aplicada à solução inicial x_s, em que apenas um conjunto limitado de movimentos pode ser executado e onde os movimentos ascendentes são permitido. Várias alternativas foram consideradas e combinadas em implementações recentes de reconexão de caminhos [1-3, 7, 48, 50, 52]:

- revinculação periódica: a reconexão de caminhos não é aplicada sistematicamente, mas apenas periodicamente;
- revinculação direta: a reconexão de caminhos é aplicada usando o pior entre x_s e x_t como solução inicial e a outra como solução alvo;
- revinculação para trás: os papéis de x_s e x_t são trocados, path-relinking é aplicado usando o melhor entre x_s e x_t como solução inicial e a outra como solução alvo;

- para trás e para frente revinculando: duas trajetórias diferentes são exploradas, o primeiro usando x_s como a solução inicial e a segunda usando x_t nesta função;

- revinculação mista: dois caminhos são explorados simultaneamente, o primeiro emanando de x_s e o segundo de x_t, até que se encontrem em uma solução intermediária equidistante de x_s e x_t;

- religação aleatória: em vez de selecionar o melhor movimento ainda não selecionado, selecione aleatoriamente um dentre uma lista de candidatos com os movimentos mais promissores no caminho que está sendo investigado; e

- revinculação truncada: a trajetória completa entre x_s e x_t não é investigado, mas apenas parte dele.

Todas essas alternativas envolvem trocas entre o tempo de computação e a qualidade da solução. Ribeiro et al. [50] observou que explorar dois diferentes trajetórias para cada par (x_s, x_t) leva aproximadamente o dobro do tempo necessário para explorar apenas um deles, com melhorias muito marginais em qualidade da solução. Eles também observaram que se apenas uma trajetória deve ser investigada, melhores soluções são encontradas quando o procedimento de religação começa com o melhor entre x_s e x_t. Uma vez que a vizinhança da solução inicial é explorada com muito mais cuidado do que a do guia um, começar pelo melhor deles dá ao algoritmo uma chance melhor de investigar com mais detalhes a vizinhança da solução mais promissora. Pelo mesmo motivo, as melhores soluções geralmente encontram-se mais próximas da solução inicial do que da solução guia, permitindo a poda da trajetória de religação antes que esta seja alcançada.

2.3 SEGURAR COM RELINAÇÃO DE CAMINHO

A reconexão de caminhos é um grande aprimoramento do procedimento GRASP básico, levando a melhorias significativas no tempo e na qualidade da solução.

O uso do path-relinking dentro de um procedimento GRASP, como uma estratégia de intensificação aplicada a cada solução localmente ótima, foi proposto pela primeira vez por Laguna e Mart´ı [28]. Ele foi seguido por várias extensões, melhorias e aplicativos bem-sucedidos [3, 9, 43, 45, 50]. Duas estratégias básicas são usadas:

- A reconexão de caminhos é aplicada a todos os pares de soluções elite, seja periodicamente durante as iterações GRASP ou após todas as iterações GRASP terem sido realizadas como uma etapa pós-otimização; e

- A reconexão de caminhos é aplicada como uma estratégia de intensificação para cada ótimo local obtido após a fase de busca local.

Aplicar a reconexão de caminhos como uma estratégia de intensificação para cada ótimo local parece ser mais eficaz do que simplesmente usá-la apenas como uma etapa de pós-otimização. Em geral, combinar intensificação com pós-otimização resulta na melhor estratégia. No contexto de intensificação, a reconexão de caminhos é aplicada aos pares (x, y) de soluções, onde x é uma solução localmente ideal produzida por cada iteração GRASP após pesquisa local e y é uma das poucas soluções de elite escolhidas aleatoriamente em um pool com um número limitado Max Elite de soluções de elite encontradas ao longo da pesquisa. A seleção aleatória uniforme é uma estratégia simples de implementar. Uma vez que a diferença simétrica é uma medida do comprimento do caminho explorado durante a religação, uma estratégia tendenciosa para os elementos do pool y com alta diferença simétrica em relação a x geralmente é melhor do que usar seleção aleatória uniforme [45].

A piscina está originalmente vazia. Visto que desejamos manter um pool de soluções boas, mas diversas, cada solução localmente ótima obtida pela pesquisa local é considerada uma candidata a ser inserida no pool se for suficientemente diferente de todas as outras soluções atualmente no pool. Se a piscina já tiver Max Elite soluções e o candidato é melhor do que o pior deles, então uma estratégia simples é fazer com que o primeiro substitua o último. Outra estratégia, que tende a aumentar a diversidade do pool, é substituir o elemento do pool mais semelhante ao candidato entre todos os elementos do pool com custo pior que o do candidato. Se o pool não estiver cheio, o candidato é simplesmente inserido.

A pós-otimização é feita em uma série de pools. O pool inicial P_0 é a piscina P obtido no final das iterações GRASP. O valor que da melhor solução de P_0 é atribuído a f^*_0 e o balcão da piscina é inicializado k = 0. No k- a iteração, todos os pares de elementos no pool P_k são combinados usando revinculação de caminho. Cada resultado da reconexão de caminho é testado para associação no pool P_{k+1} seguindo os mesmos critérios usados durante as iterações GRASP. Se uma nova melhor solução for produzida, ou seja, $f^*_{k+1} < f^*_k$, então $k \leftarrow k + 1$ e uma nova iteração de pós-otimização é feito. Caso contrário, a pós-otimização para com $x^* = \operatorname{argmin}\{ f(x) \mid x \in P_{k+1}\}$ como resultado.

O algoritmo 2.3 ilustra tal procedimento. Cada iteração GRASP agora tem três etapas principais:

- Fase de construção: um procedimento ganancioso de construção aleatória é usado para construir uma solução viável;
- Fase de pesquisa local: a solução construída na primeira fase é progressivamente melhorada por uma estratégia de busca na vizinhança, até que um mínimo local seja encontrado; e

- Fase de reconexão do caminho: o algoritmo de reconexão de caminho usando qualquer uma das estratégias descritas na Seção 2.2 é aplicado à solução obtida por pesquisa local e a uma solução selecionada aleatoriamente do pool. A melhor solução encontrada ao longo dessa trajetória também é considerada como candidata a inserção na piscina e o titular é atualizado.

Ao final das iterações GRASP, uma fase de pós-otimização combina as soluções de elite do pool na busca pelas melhores soluções.

```
Dados : Número de iterações eu max
Resultado: Solução x * ∈ X
P ← ∅;
f * ← ∞;
para i = 1,. . . , eu max Faz
    x ← GreedyRandomizedConstruction ();
    x ← LocalSearch ( x);
    E se eu ≥ 2 então
        Escolha, aleatoriamente, soluções de pool Y ⊆ P revincular
        com x;
        para y ∈ Y Faz
            Determine qual ( x ou y) é inicial x s e qual é o
            alvo x t;
            x p ← PathRelinking ( x s, x t);
            Atualize o conjunto de elite P com x p;
            E se f (x p) < f * então
                f * ← f (x p);
                x * ← x p;
            fim
        fim
    fim
fim
P = PostOptimize { P};
x * = argmin { f (x), x ∈ P};
```

Algoritmo 2.3: Um GRASP básico com heurística de revinculação de caminho para minimização.

Aiex [1] e Aiex et al. [4] demonstraram experimentalmente que os tempos de solução para encontrar um valor de solução alvo com uma heurística GRASP ajustam-se a uma distribuição exponencial de dois parâmetros. A Figura 2.3 ilustra esse resultado, mostrando as distribuições empíricas e teóricas sobrepostas observadas para um dos casos estudados nos experimentos computacionais relatados pelos autores, que envolveram 2.400 execuções de GRASP

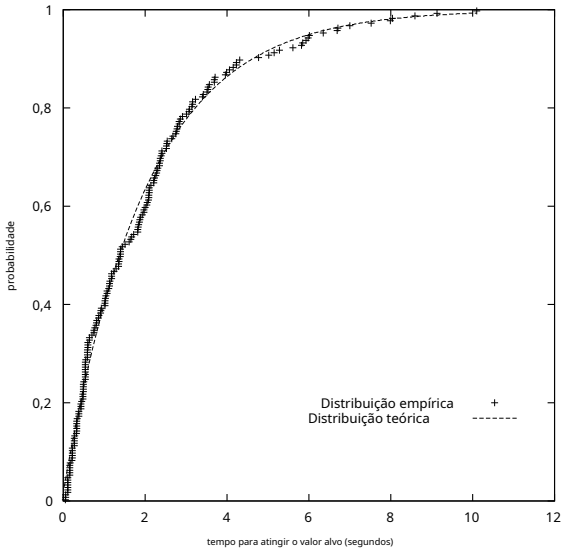

Figura 2.3. Distribuições empíricas e teóricas sobrepostas (tempos até os valores alvo medidos em segundos em um computador SGI Challenge com 28 processadores).

procedimentos para cada um dos cinco problemas diferentes: conjunto independente máximo [16, 37], atribuição quadrática [29, 38], planarização do gráfico [40, 47], satisfação máxima ponderada [39] e cobertura máxima [36]. O mesmo resultado ainda é válido quando GRASP é implementado em conjunto com um procedimento de religação de caminho [2].

2.4 APLICAÇÕES

O relink de caminho foi usado com sucesso junto com GRASP em uma variedade de aplicações, como o problema de atribuição de três índices [1, 3], o problema de roteamento de circuitos privados em redes de comunicação [41], o problema de projeto de rede de 2 caminhos [48], o p- o problema da mediana [44], o problema de Steiner nos gráficos [50], o problema de programação job-shop [1, 2], o problema da árvore de Steiner para coleta de prêmios [9], o problema de atribuição quadrática [32], o MAX-CUT problema [17], e o problema de spanning tree mínimo capacitado [53]. Alguns desses aplicativos serão analisados no restante desta seção.

Antes de revisarmos algumas dessas aplicações, primeiro descrevemos um gráfico usado em vários de nossos artigos para comparar experimentalmente diferentes algoritmos aleatórios ou diferentes versões do mesmo algoritmo aleatório [1, 4]. Este gráfico mostra as distribuições empíricas da variável aleatória tempo para atingir o valor da solução. Para traçar a distribuição empírica, fixamos um valor alvo da solução e executamos cada algoritmo T independente

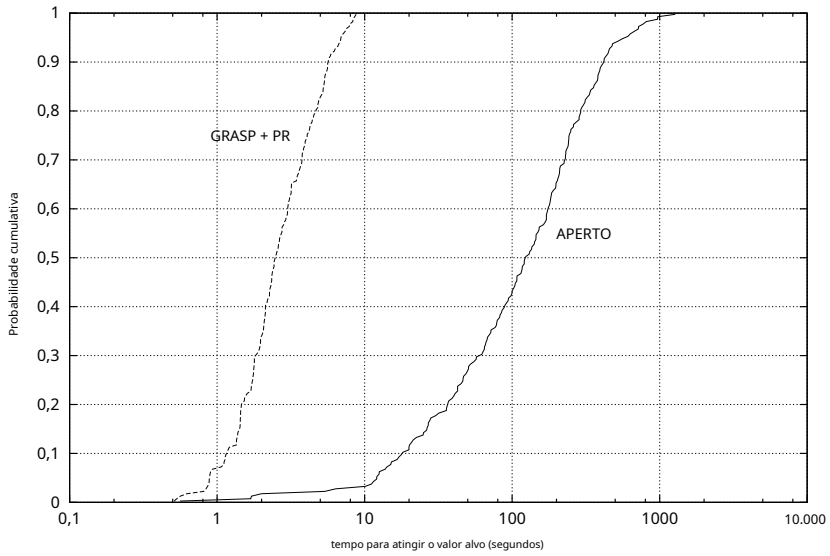

Figura 2.4. Distribuições empíricas das variáveis aleatórias hora de direcionar a solução para um GRASP puro e um GRASP com redirecionamento de caminho para a instância MAX-CUT G11 com valor alvo de 552 [17]. Duzentas execuções independentes de cada algoritmo foram usadas para fazer os gráficos.

vezes, registrando o tempo de execução quando uma solução com custo pelo menos tão bom quanto o valor-alvo é encontrada. Para cada algoritmo, associamos com o eu- o tempo de execução classificado (t eu) uma probabilidade p i = (eu - 1 2) / T, e traçar os pontos z i = (t eu, p eu), para i = 1,. . . , T. A Figura 2.4 mostra um gráfico comparando um GRASP puro com um GRASP com reconexão de caminho para Instância MAX-CUT G11 com valor de solução alvo de 552. A figura mostra claramente que GRASP com revinculação de caminho (GRASP + PR) é muito mais rápido do que GRASP puro para encontrar uma solução com peso 552 ou mais. Por exemplo, a probabilidade de encontrar tal solução em menos de 5 segundos é superior a 80% com GRASP com reconexão de caminho, enquanto é de cerca de 2% com GRASP puro. Da mesma forma, com probabilidade de 50% GRASP com path-relinking encontra tal solução alvo em menos de 2,5 segundos, enquanto para GRASP puro, com probabilidade de 50% uma solução é encontrada em menos de 122 segundos.

2.4.1 Roteamento de circuito virtual privado

Um serviço de frame relay oferece redes privadas virtuais aos clientes, fornecendo um conjunto de circuitos virtuais privados de longo prazo (PVCs) entre os terminais do cliente em uma grande rede de backbone. Durante o provisionamento de um PVC, as decisões de roteamento são feitas sem nenhum conhecimento de

pedidos futuros. Com o tempo, essas decisões podem causar ineficiências na rede e ocasionais redirecionamentos dos PVCs são necessários. Resende e Ribeiro [43] formulam o problema de roteamento de PVC o ffl ine como um problema de fluxo multi-commodity inteiro com restrições adicionais e com uma função objetivo que minimiza atrasos de propagação e / ou congestionamento da rede. Eles propõem variantes de um GRASP com heurística de reconexão de caminhos para esse problema. Resultados experimentais para problemas de tamanho realístico mostram que GRASP se beneficia muito da reconexão de caminhos e que as heurísticas propostas são capazes de melhorar as soluções encontradas com técnicas de roteamento padrão.

Deixar G = (V, E) ser um gráfico não direcionado que representa a rede frame relay. Denotado por V = { 1,. . . , n} o conjunto de nós de backbone onde os switches residem, enquanto E é um conjunto de troncos (ou bordas) que conectam os nós do backbone, com | E | = m. Troncos paralelos são permitidos. Desde a G é um gráfico não direcionado, flui através de cada tronco (eu j) ∈ E têm dois componentes a serem somados, um em cada direção. No entanto, para fins de modelagem, custos e capacidades sempre estarão associados apenas ao par pedido (eu j) satisfatório i <j. Para cada tronco (eu j) ∈ E, denotar por $b_{eu\,j}$ sua largura de banda máxima permitida (em kbits / segundo), enquanto $c_{eu\,j}$ denota o número máximo de PVCs que podem ser encaminhados através dele e $d_{eu\,j}$ é o atraso de propagação ou salto associado ao tronco. Cada mercadoria k ∈ K = { 1,. . . , p} é um PVC a ser roteado, associado com um par origem-destino e com um requisito de largura de banda (ou demanda, também conhecida como largura de banda efetiva) r_k. Este último leva em consideração a largura de banda real exigida pelo cliente no encaminhamento e direções reversas, bem como um fator de overbooking.

Deixar $x^k_{ij} = 1$ se e somente se borda (eu j) ∈ E é usado para encaminhar mercadoria k ∈ K. A função de custo $\varphi_{eu\,j}(x^1_{eu\,j}, \ldots, x^{p}_{eu\,j}, x^1_{ji}, \ldots, x^{p}_{ji})$ associado com cada tronco (eu j) ∈ E com eu <j é a combinação linear de um tronco componente de retardo de propagação e um componente de congestionamento de tronco. O componente de atraso de propagação é definido como

$$\varphi^{d}_{eu\,j}(x^1_{eu\,j}, \ldots, x^{p}_{eu\,j}, x^1_{ji}, \ldots, x^{p}_{ji}) = d_{eu\,j} \cdot \sum_{k \in K} \rho_k (x^k_{ij} + x^k_{ji}), \qquad (2.1)$$

onde coeficientes ρ_k são usados para modelar duas funções de atraso plausíveis:

- Se $\rho_k = 1$, então esse componente leva à minimização do número de saltos ponderados pelo atraso de propagação em cada tronco.
- Se $\rho_k = r_k$, então a minimização leva em consideração a largura de banda efetiva roteada através de cada tronco ponderada por sua propagação atraso.

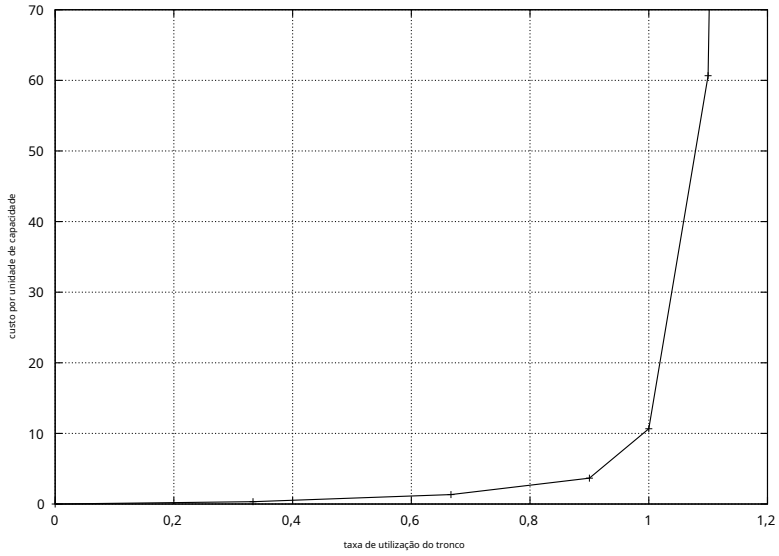

Figura 2.5. Componente de custo de congestionamento linear por partes associado a cada tronco.

Deixar $y_{ij} = \sum_{k \in K} r_k (x_{kij} + x_{k ji})$ ser o fluxo total através do tronco $(i, j) \in E$ com $i < j$. O componente de congestionamento de tronco depende da utilização cotações $você_{ij} = y_{i,j} / b_{i,j}$ de cada tronco $(i, j) \in E$ com $i < j$. É considerada a função linear por partes proposta por Fortz e Thorup [20] e representado na Figura 2.5, que penaliza cada vez mais os fl uxos que se aproximam ou violam os limites de capacidade:

$$\varphi_{b_{i,j}}(x_{1,i,j}, \ldots, x_{p,i,j}, x_{j,i}, \ldots, x_{p\,ji}) = \begin{cases} você_{i,j}, & você_{i,j} \in [0, 1/3) \\ 3 \cdot você_{i,j} - 2/3, & você_{i,j} \in [1/3, 2/3), \\ 10 \cdot você_{i,j} - 16/3, & você_{i,j} \in [2/3, 9/10), \\ 70 \cdot você_{i,j} - 178/3, & você_{i,j} \in [9/10, 1), \\ 500 \cdot você_{i,j} - 1468/3, & você_{i,j} \in [1, 11/10), \\ 5000 \cdot você_{i,j} - 16318/3, & você_{i,j} \in [11/10, \infty). \end{cases} \quad (2.2)$$

Para o roteamento do PVC, Resende e Ribeiro utilizaram a função custo

$$\varphi_{i,j}(x_{1,i,j}, \ldots, x_{p,i,j}, x_{j,i}, \ldots, x_{p\,ji}) = (1 - \delta) \cdot \varphi_{d_{i,j}}(x_{1,i,j}, \ldots, x_{p,i,j}, x_{j,i}, \ldots, x_{p\,ji}) + \delta \cdot \varphi_{b_{i,j}}(x_{1,i,j}, \ldots, x_{p,i,j}, x_{j,i}, \ldots, x_{p\,ji}) \quad (2,3)$$

associado a cada tronco (eu j) ∈ E com i <j, onde pesos (1 - δ) e δ correspondem, respectivamente, ao atraso de propagação e aos componentes de congestionamento da rede, com δ ∈ [0, 1].

Na fase de construção do GRASP, as rotas são definidas uma de cada vez. Um novo PVC é selecionado para ser roteado em cada iteração. Para reduzir os tempos de computação, usamos uma combinação das estratégias usualmente empregadas pelo GRASP e amostragem estocástica com viés heurístico. Criamos uma lista de candidatos restrita com um número fixo de elementos n_c. A cada iteração, ele é formado pelo n_c pares de PVC não roteados com as maiores demandas. Um elemento ℓ é selecionado aleatoriamente desta lista com probabilidade $\pi(\ell) = r_\ell / \sum_{k \in RCL} r_k$.

Uma vez que um PVC $\ell \in K$ é selecionado, ele é roteado em um caminho mais curto de sua origem até seu destino. As restrições de capacidade de largura de banda são relaxadas e tratadas por meio da função de penalidade introduzida pelo componente de congestionamento de tronco (2.2) dos pesos de borda. As restrições ao limite de PVCs roteados através de cada tronco são explicitamente levadas em consideração ao proibir o roteamento através de troncos que já usam seu máximo número de PVCs. O peso $\Delta\varphi_{eu\,j}$ de cada borda (eu j) ∈ E é dada pelo incremento do valor da função de custo $\varphi_{eu\,j}(x_{1\,eu\,j}, \ldots, x_{p_{eu\,j}\,ji}, \ldots, x_{p\,ji})$, associado a roteamento r_ℓ unidades adicionais de demanda por meio de borda (eu j).

Mais precisamente, vamos $\underline{K} \subseteq K$ ser o conjunto de PVCs previamente roteados e $\underline{K}_{eu\,j} \subseteq \underline{K}$ ser o subconjunto de PVCs que são roteados através do tronco (eu j) ∈ E. Li da mesma forma, deixe $\overline{K} = \underline{K} \cup \{\ell\} \subseteq K$ ser o novo conjunto de PVCs roteados e $\overline{K}_{ij} = \underline{K}_{eu\,j} \cup \{\ell\} \subseteq K$ ser o novo subconjunto de PVCs que são roteados por meio de tronco (eu j). Então, defina $\underline{x}^h_{ij} = 1$ se PVC $h \in \underline{K}$ é encaminhado através do tronco (eu j) ∈ E a partir de eu para j, $\underline{x}^h_{ij} = 0$ caso contrário. Da mesma forma, definir $\overline{x}^h_{ij} = 1$ se PVC $h \in \overline{K}$ é encaminhado através do tronco (eu j) ∈ E a partir de eu para j, $\overline{x}^h_{ij} = 0$ caso contrário. De acordo com (2.3), o custo associado a cada aresta (eu j) ∈ E no a solução atual é dada por $\varphi_{eu\,j}(\underline{x}_{1\,eu\,j}, \ldots, \underline{x}_{p_{eu\,j}\,ji}, \ldots, \underline{x}_{p\,ji})$. No mesmo forma, o custo associado a cada borda (eu j) ∈ E após o roteamento de PVC ℓ vai ser $\varphi_{eu\,j}(\overline{x}_{1\,eu\,j}, \ldots, \overline{x}_{p_{eu\,j}\,ji}, \ldots, \overline{x}_{p\,ji})$. Então, o peso da borda incremental $\Delta\varphi_{eu\,j}$ associado ao roteamento de PVC $\ell \in K$ através da borda (eu j) ∈ E, usado nos cálculos de caminho mais curto, é dado por

$$\Delta\varphi_{ij} = \varphi_{eu\,j}(\overline{x}_{1\,eu\,j}, \ldots, \overline{x}_{p_{eu\,j}\,ji}, \ldots, \overline{x}_{p\,ji}) - \varphi_{eu\,j}(\underline{x}_{1\,eu\,j}, \ldots, \underline{x}_{p_{eu\,j}\,ji}, \ldots, \underline{x}_{p\,ji}). \quad (2,4)$$

A aplicação das restrições que limitam o número de PVCs roteados através de cada tronco pode levar a pares de demanda não roteáveis. Nesse caso, a solução atual é descartada e uma nova fase de construção é iniciada.

Cada solução construída na primeira fase pode ser vista como um conjunto de rotas, uma para cada PVC. O procedimento de busca local busca aprimorar cada

rota na solução atual. Para cada PVC $k \in K$, comece removendo r_k unidades de fl uxo de cada borda em sua rota atual. Em seguida, calcule increpesos de borda mental $\Delta \varphi_{eu\,j}$ associado ao roteamento desta demanda através de cada tronco $(eu\,j) \in E$ de acordo com (2.4). Um novo caminho mais curto provisório a rota é calculada usando os pesos de borda incrementais. Se a nova rota melhorar a solução, ela substituirá a rota atual do PVC k. Isso é continuado até que nenhuma rota de melhoria possa ser encontrada.

Na estratégia de reconexão de caminhos proposta, o conjunto de movimentos corresponde com a diferença simétrica $\Delta(x_1, x_2)$ entre qualquer par $\{x_1, x_2\}$ de soluções é o subconjunto $K_{x_1, x} \subseteq K$ de PVCs encaminhados através de diferentes rotas em x_1 e x_2 Sem perda de generalidade, suponha que o pathrelinking comece a partir de qualquer solução de elite z na piscina e usa o local solução ótima y como solução orientadora.

A melhor solução y ao longo do novo caminho a ser construído é inicializado com z. Para cada PVC $k \in K_{y, z,}$ os mesmos cálculos de caminho mais curto descritos para construção e pesquisa local são usados para avaliar o custo da nova solução obtida por meio do redirecionamento da demanda associada ao PVC k através da rota usada na solução de orientação y em vez daquele usado na solução atual originado de z. O melhor movimento é selecionado e removido de $K_{y, z.}$ A nova solução obtida por reencaminhamento do PVC selecionado acima é calculada, o titular y é atualizado, e um novo a iteração é retomada. Essas etapas são repetidas, até a solução norteadora y é atingido. O titular y é retornado como a melhor solução encontrada pela reconexão de caminho e inserida no pool se for melhor do que a pior solução atualmente no pool.

A Figura 2.6 ilustra a comparação dos quatro algoritmos: GRASP puro (APERTO), GRASP com redirecionamento de caminho direto (GRASP + PRf, em que uma solução localmente ideal é usada como a solução inicial), GRASP com religação de caminho inverso (GRASP + PRb, em que uma solução de elite é usada como a solução inicial), e GRASP com revinculação de caminho para trás e para frente (GRASP + PRfb, em que a reconexão de caminho é realizada em ambas as direções) na instância de roteamento de PVC fr750a (60 nós, 498 arcos e 750 mercadorias). Para um determinado tempo de cálculo, a probabilidade de encontrar uma solução pelo menos tão boa quanto o valor alvo aumenta a partir de APERTO para GRASP + PRf, a partir de GRASP + PRf para GRASP + PRfb, e de GRASP + PRfb para GRASP + PRb. Por exemplo, há 9,25% de probabilidade para GRASP + PRfb para encontrar uma solução alvo em menos de 100 segundos, enquanto essa probabilidade aumenta para 28,75% para GRASP + PRb. Para APERTO, existe um 8,33% de probabilidade de encontrar uma solução alvo em 2.000 segundos, enquanto para GRASP + PRf essa probabilidade aumenta para 65,25%. GRASP + PRb encontra uma solução alvo em no máximo 129 segundos com 50% de probabilidade. Para o

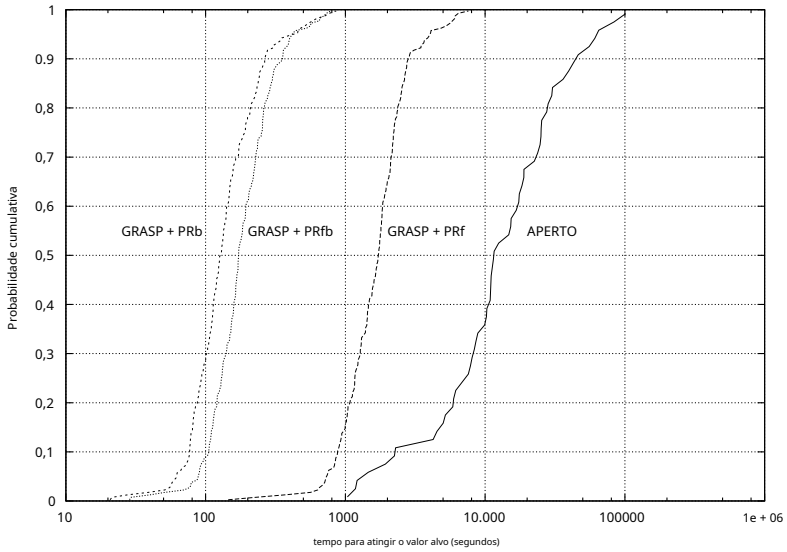

Figura 2.6. Distribuições empíricas de tempo para solução de destino para GRASP, GRASP com revinculação de caminho para frente, GRASP com revinculação de caminho para trás e GRASP com revinculação de caminho para trás e para frente para instância de roteamento de circuito virtual privado fr750a.
Duzentas execuções independentes foram feitas para cada algoritmo. O valor da solução alvo usado foi 479000.

mesma probabilidade, este tempo aumenta para 172, 1727 e 10933 segundos, respectivamente, para as variantes GRASP + PRfb, GRASP + PRf, e APERTO.

Esses resultados sugerem que a variante GRASP + PRb, que executa o pathrelinking para trás de uma solução de elite para uma solução ideal localmente, é o mais eficaz.

Outro experimento comparando as quatro variantes foi feito na instância de roteamento de PVC att (90 nós, 274 troncos, 272 commodities). Dez execuções independentes de cada algoritmo foram feitas por 100 segundos em um computador SGI Challenge de 196 MHz. A Tabela 2.1 resume esses resultados. Para cada variante, esta tabela lista os valores de solução melhores e médios encontrados após 10 segundos e após 100 segundos. Os resultados apontam para GRASP + PRb e GRASP + PRfb como as duas melhores heurísticas. É interessante notar que mesmo se dados 100 segundos, APERTO encontra soluções de pior qualidade do que aquelas encontradas por GRASP + PRb e GRASP + PRfb em apenas 10 segundos.

2.4.2 Projeto de rede de 2 caminhos

Deixar G = (V, E) ser um gráfico não direcionado conectado, onde V é o conjunto de nós e E é o conjunto de arestas. UMA k- caminho entre nós s, t ∈ V é uma sequência de no máximo k bordas conectando-os. Dado um não negativo

Tabela 2.1. Comparação de GRASP, GRASP com revinculação de caminho para frente, GRASP com relink de caminho para trás e GRASP com revinculação de caminho para trás e para frente para instância de roteamento de circuito virtual privado att. Dez execuções independentes de 100 segundos foram feitas para cada algoritmo.

10 corridas	10 segundos		100 segundos	
Variante	melhor	média	melhor	média
APERTO	126603	126695	126228	126558
GRASP + PRf	126301	126578	126083	126229
GRASP + PRb	125960	126281	125666	125883
GRASP + PRfb	125961	126307	125646	125850

função de peso nós → R + associado com as bordas de G e um conjunto D de pares de nós origem-destino, o Problema de projeto de rede de 2 caminhos (2PNDP) consiste em encontrar um subconjunto mínimo ponderado de arestas $E' \subseteq E$ contendo 2 caminhos entre cada par origem-destino. As aplicações podem ser encontradas no projeto de redes de comunicação, em que caminhos com poucas bordas são procurados para garantir alta confiabilidade e pequenos atrasos. Dahl e Johannessen [13] provaram que a versão de decisão do 2PNDP é NP-completa.

Rosseti [52] e Ribeiro e Rosseti [48] descreveram implementações sequenciais e paralelas de GRASP com religamento de caminho para o projeto de rede de 2 caminhos. A construção de uma nova solução começa com a inicialização dos pesos das arestas modi fi cados com os pesos das arestas originais. Cada iteração da fase de construção começa pela seleção aleatória de um par origem-destino ainda em D. Um 2-caminho mais curto entre as extremidades deste par é calculado, usando os pesos de aresta modi fi cados. Os pesos das arestas neste 2-caminho são definidos como zero até o final do procedimento de construção, o par origem-destino é removido D,
e uma nova iteração é retomada. A fase de construção para quando 2 caminhos foram calculados para todos os pares origem-destino.

A fase de busca local busca aprimorar cada solução construída na fase de construção. Cada solução pode ser vista como um conjunto de 2 caminhos, um para cada par origem-destino em D. Para introduzir alguma diversidade, conduzindo diferentes aplicações da busca local para diferentes ótimos locais, os pares origem-destino são investigados em cada iteração GRASP em uma ordem circular definida por uma permutação aleatória diferente de seus índices originais. Cada 2 caminhos na solução atual é provisoriamente eliminado. Os pesos das arestas usadas por outros 2 caminhos são temporariamente definidos como zero, enquanto aqueles que não são usados por outros 2 caminhos na solução atual são restaurados aos seus valores originais. Um novo 2-caminho mais curto será

entre as extremidades do par origem-destino sob investigação é calculado, usando os pesos modificados. Se o novo 2-path melhora a solução atual, então a última é modificada; caso contrário, o 2-path anterior é restaurado. A pesquisa para se a solução atual não foi melhorada após uma sequência de | D | iterações ao longo das quais todos os 2 caminhos foram investigados. Caso contrário, o próximo 2-path na solução atual é investigado para substituição e uma nova iteração é retomada.

Cada iteração GRASP executa uma fase de intensificação usando pathrelinking, na qual a solução recém-gerada obtida no final da fase de busca local é combinada com uma solução selecionada aleatoriamente do pool de soluções de elite. A reconexão de caminhos começa determinando todos os pares origem-destino cujos 2 caminhos associados são diferentes nas duas soluções. Esses cálculos equivalem a determinar o conjunto de movimentos que devem ser aplicados à solução inicial para alcançar a guia. Cada movimento é caracterizado por um par de 2 caminhos, um a ser inserido e o outro a ser eliminado da solução atual. A cada iteração de pathrelinking, o melhor movimento ainda não selecionado é aplicado à solução atual e a melhor solução encontrada ao longo do caminho que conecta as duas soluções é atualizada.

Várias estratégias para a implementação da etapa de reconexão de caminho foram investigadas em [49, 52]: GRASP puro (APERTO), GRASP com redirecionamento de caminho direto (GRASP + PRf, em que uma solução localmente ideal é usada como a solução inicial), GRASP com religação de caminho inverso (GRASP + PRb, em que uma solução elite é usada como a solução inicial), GRASP com retrocesso e redirecionamento de caminho para frente (GRASP + PRfb, em que a reconexão de caminho é realizada duas vezes, uma em cada direção), e GRASP com a reconexão de caminho mista (GRASP + PRm, em que dois caminhos em direções opostas são explorados simultaneamente).

Os resultados exibidos na Tabela 2.2 ilustram o comportamento dessas cinco variantes em instâncias geradas aleatoriamente [52] em gráficos completos com 100, 200, 300, 400 e 500 nós. Para cada instância, fornecemos os valores de solução melhores e médios encontrados em dez execuções independentes de cada algoritmo. Para cada tamanho de problema, o tempo de processamento é limitado ao observado para 200 iterações do procedimento GRASP puro na primeira instância do grupo. Algoritmos GRASP + PRfb e GRASP + PRm tiveram um desempenho melhor do que as outras variantes, na medida em que juntas encontraram as melhores soluções e as melhores soluções médias para todas as instâncias da tabela. GRASP com reconexão de caminho para trás geralmente tem um desempenho melhor do que a variante de reconexão de caminho para frente, devido ao fato de que ele começa

Tabela 2.2. Resultados de dez execuções de cada algoritmo em instâncias geradas aleatoriamente de problemas de projeto de rede de 2 caminhos com tempo de processamento limitado.

	APERTO		GRASP + PRf		GRASP + PRb		GRASP + PRfb		GRASP + PRm	
| V |	melhor	média	melhor	média	melhor	média	melhor	média	melhor	média
100	779	784,3	760	772,8	763	769,3	749	762,7	755	765,3
	762	769,6	730	749,4	735	746,0	729	741,7	736	745,7
	773	779,2	762	769,3	756	766,1	757	763,6	754	765,1
	746	752,0	732	738,4	723	736,7	719	730,4	717	732,2
	756	762,3	742	749,7	739	746,5	737	742,9	728	743,7
200	1606	1614,7	1571	1584,4	1540	1568,0	1526	1562,0	1538	1564,3
	1601	1608,8	1557	1572,8	1559	1567,9	1537	1558,9	1545	1563,3
	1564	1578,2	1523	1541,9	1516	1531,9	1508	1519,9	1509	1528,7
	1578	1585,6	1531	1553,3	1518	1538,1	1510	1532,2	1513	1534,7
	1577	1599,6	1567	1575,4	1543	1563,5	1529	1556,3	1531	1556,1
300	2459	2481,9	2408	2425,0	2377	2401,3	2355	2399,2	2366	2393,6
	2520	2527,7	2453	2469,7	2419	2449,1	2413	2438,9	2405	2439,4
	2448	2463,5	2381	2403,1	2339	2373,8	2356	2375,3	2338	2370,3
	2462	2482,1	2413	2436,2	2373	2409,3	2369	2.400,9	2350	2401,0
	2450	2458,8	2364	2402,5	2328	2368,6	2347	2373,9	2322	2365,4
400	3355	3363,8	3267	3285,5	3238	3257,0	3221	3239,4	3231	3252,2
	3393	3417,5	3324	3338,2	3283	3306,8	3220	3292,2	3271	3301,4
	3388	3394,4	3311	3322,4	3268	3291,9	3227	3275,1	3257	3273,2
	3396	3406,0	3316	3326,5	3249	3292,0	3256	3284,8	3246	3287,9
	3416	3429,3	3335	3365,5	3267	3327,7	3270	3313,9	3259	3323,5
500	4338	4350,1	4209	4247,1	4176	4207,6	4152	4196,1	4175	4206,2
	4353	4369,6	4261	4278,6	4180	4233,7	4166	4219,6	4175	4226,3
	4347	4360,7	4239	4257,8	4187	4224,8	4170	4201,9	4187	4217,9
	4317	4333,8	4222	4238,6	4157	4197,4	4156	4182,2	4159	4197,1
	4362	4370,4	4263	4292,0	4203	4294,0	4211	4236,8	4200	4240,2

de uma solução de elite que muitas vezes é melhor do que o ótimo local atual, explorando totalmente a vizinhança do primeiro.

Os resultados observados para a variante GRASP + PRm são muito encorajadores: este algoritmo encontrou soluções melhores do que as outras variantes para 40% das instâncias.

Para ilustrar e comparar ainda mais essas cinco variantes, exibimos na Figura 2.7 um gráfico da distribuição de probabilidade empírica do tempo para atingir o valor da solução de cada algoritmo, calculado a partir de 200 execuções independentes. Esses gráficos mostram que a probabilidade de encontrar uma solução pelo menos tão boa quanto um valor alvo aumenta de APERTO para GRASP + PRf para GRASP + PRb para GRASP + PRfb, e finalmente para GRASP + PRm. Esses resultados confirmam uma observação feita inicialmente por Ribeiro et al. [50] e posteriormente por Resende e Ribeiro [42], sugerindo que a estratégia para trás executa

um papel importante nas implementações bem-sucedidas de reconexão de caminhos. Além disso, eles também indicam que a estratégia mista de revinculação de caminhos proposta por Rosseti [52] é muito eficaz.

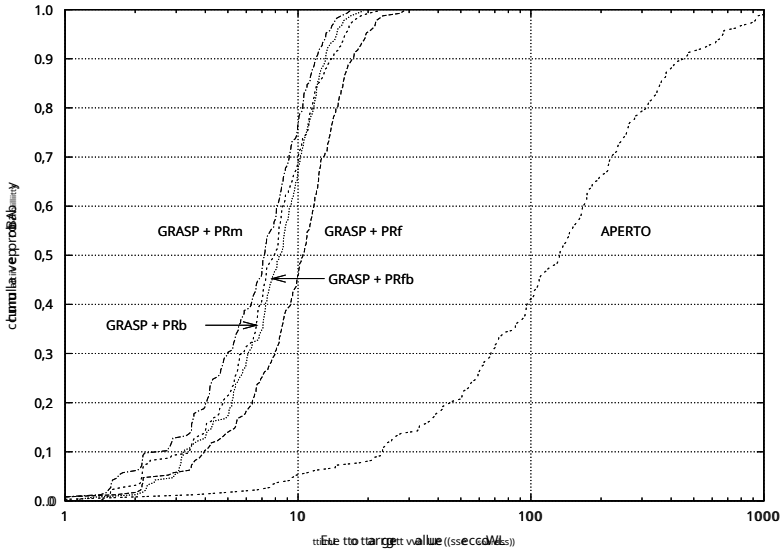

Figura 2.7. Distribuições empíricas de tempo para solução de destino para GRASP, GRASP com religação de caminho para frente, GRASP com religação de caminho para trás, GRASP com religação de caminho para trás e para frente e GRASP com religação de caminho misto para uma instância de design de rede de 2 caminhos com 80 nós. Duzentas execuções independentes foram feitas para cada algoritmo. O valor da solução alvo usado foi 588.

2.4.3 p- problema mediano

No p- problema mediano, recebemos um conjunto F de m instalações potenciais, um conjunto você de n usuários (ou clientes), uma função à distância d: U × F → R, e uma constante p ≤ m, e quer determinar qual p instalações a serem abertas de forma a minimizar a soma das distâncias de cada usuário até a instalação aberta mais próxima. É um problema NP-difícil bem conhecido [27].

Resende e Werneck [45] descrevem um GRASP com path-relinking para o p- problema mediano. Resultados empíricos em instâncias da literatura mostram que o algoritmo é robusto e que executa pelo menos tão bem quanto outros métodos, e muitas vezes melhor em termos de tempo de execução e qualidade da solução. Em todos os casos, as soluções obtidas pelo GRASP com reconexão de caminho estavam dentro de 0,1% dos limites superiores mais conhecidos. Para um grande número de instâncias, novas soluções mais conhecidas foram produzidas pelo novo algoritmo.

O algoritmo ganancioso padrão para o p- O problema mediano [10, 55] começa com uma solução vazia e adiciona instalações uma de cada vez, escolhendo a mais lucrativa em cada iteração (aquela cuja inserção causa a maior queda no custo da solução). O procedimento de construção proposto em [45] é semelhante ao algoritmo guloso, mas ao invés de selecionar a melhor entre todas as opções possíveis, ele apenas considera q <m possíveis inserções (escolhidas uniformemente ao acaso) em cada iteração. O mais lucrativo entre eles é selecionado. O tempo de execução do algoritmo é O (m + pqn). A ideia é fazer q pequeno o suficiente para reduzir significativamente o tempo de execução do algoritmo (quando comparado ao guloso puro) e para garantir um grau razoável de randomização. Em testes, o valor
q = d registro $_{2}$ (m / p) e foi determinado ser adequado.

O procedimento padrão de pesquisa local para o p- problema mediano, orig- proposto por Teitz e Bart [54], é baseado em facilidades de troca. Dada uma solução inicial S, o procedimento determina, para cada instalação f 6 ∈ S, qual facilidade g ∈ S (se houver) melhoraria mais a solução se f e g foram trocados (ou seja, se f foram abertos e g fechado). Se houver um movimento de melhoria, f e g são trocados. O procedimento continua até que nenhuma troca de melhoria possa ser feita, caso em que um mínimo local terá sido encontrado. A complexidade desta pesquisa local baseada em troca é O (pmn) por iteração. Whitaker [55] propôs uma implementação eficiente deste método, que ele chamou intercâmbio rápido,
para o qual o limite no tempo de execução de cada iteração é reduzido para O (mn). Resende e Werneck [35] propuseram recentemente uma implementação alternativa. Embora tenha a mesma complexidade de pior caso que o de Whitaker, pode ser substancialmente mais rápido na prática. A aceleração (de até três ordens de magnitude) resulta do uso de informações coletadas nas primeiras iterações do algoritmo para reduzir a quantidade de computação realizada em estágios posteriores. Embora essa implementação possa exigir uma quantidade maior de memória, com o uso de algumas técnicas de programação (por exemplo, representação de matriz esparsa e cache), os requisitos adicionais de memória podem ser minimizados.

A intensificação (via redirecionamento de caminho) ocorre em dois estágios diferentes. Primeiro, cada iteração GRASP contém uma etapa de intensificação, na qual a solução recém-gerada é combinada com uma solução do pool. Em seguida, na fase de pós-otimização, as soluções do pool são combinadas entre si.

Deixar S $_{1}$ e S $_{2}$ ser duas soluções válidas, interpretadas como conjuntos de instalações (abertas). O procedimento de reconexão de caminho começa com uma das soluções (dizer, S $_{1)}$ e gradualmente o transforma no outro (S $_{2)}$ trocando elementos de S $_{2}$ \ S $_{1}$ e trocar elementos de S $_{1}$ \ S $_{2}$ O número total de trocas realizadas é | S $_{2}$ \ S $_{1}$ |, que é igual a | S $_{1}$ \ S $_{2}$ | A escolha

de qual troca fazer em cada estágio é gananciosa: a jogada mais lucrativa (ou menos custosa) é feita.

O resultado da reconexão do caminho é o melhor mínimo local no caminho. Um mínimo local neste contexto é uma solução que é bem-sucedida (imediatamente) e precedida (imediatamente ou por meio de uma série de soluções de mesmo valor) no caminho por soluções estritamente piores. Se o caminho
não tem mínimos locais, uma das soluções originais (S_1 ou $S_{2)}$ é retornado com igual probabilidade. Quando há uma solução de melhoria no caminho,
este critério corresponde exatamente ao tradicional: ele simplesmente retorna o melhor elemento no caminho. É diferente apenas quando o pathrelinking padrão é malsucedido, caso em que tenta aumentar a diversidade selecionando uma solução diferente dos extremos do caminho.

Observe que a reconexão de caminhos é muito semelhante ao procedimento de pesquisa local descrito anteriormente, com duas diferenças principais. Primeiro, o número de todos movimentos reduzidos são restritos: apenas elementos em $S_2 \setminus S_1$ pode ser inserido, e apenas aqueles em $S_1 \setminus S_2$ pode ser removido. Em segundo lugar, movimentos que não melhoram são permitidos. Essas diferenças são facilmente incorporadas às implementações básicas mentação do procedimento de pesquisa local.

O procedimento de intensificação é aumentado pela realização de uma pesquisa local completa na solução produzida pela reconexão de caminhos. Como essa solução costuma estar muito próxima de um ótimo local, essa aplicação tende a ser muito mais rápida do que em uma solução gerada pelo algoritmo construtivo aleatório. Um efeito colateral da aplicação da pesquisa local neste ponto é o aumento da diversidade, uma vez que é possível usar instalações que não pertenciam a nenhuma das soluções originais.

Os gráficos na Figura 2.8 comparam GRASP com reconexão de caminho e GRASP puro na instância TSPLIB de 1400 instalações e 1400 usuários fl1400. O gráfico à esquerda mostra a qualidade da melhor solução encontrada como uma fração do valor médio da primeira solução para GRASP com reconexão de caminho e GRASP puro para p = 500. Os tempos são dados como múltiplos do tempo médio necessário para realizar uma iteração com várias partidas. Valores menores são melhores. O gráfico à direita mostra as razões entre as soluções parciais encontradas com e sem reconexão de caminho para diferentes valores de p. Razões menores que 1.000 favorecem o uso de reconexão de caminhos. Os gráficos mostram que GRASP se beneficia da reconexão de caminhos, em particular para grandes valores de p.

2.4.4 Problema de atribuição de três índices

O problema de atribuição de três índices (AP3) foi introduzido por Pierskalla [33] como uma extensão da atribuição bidimensional clássica. problema. Considere um gráfico tripartido completo $K_{n, n, n = (}$ eu ∪ J ∪ K, (eu × J) ∪ (eu × K) ∪ (J × K)), Onde EU J, e K são conjuntos separados de tamanho n.

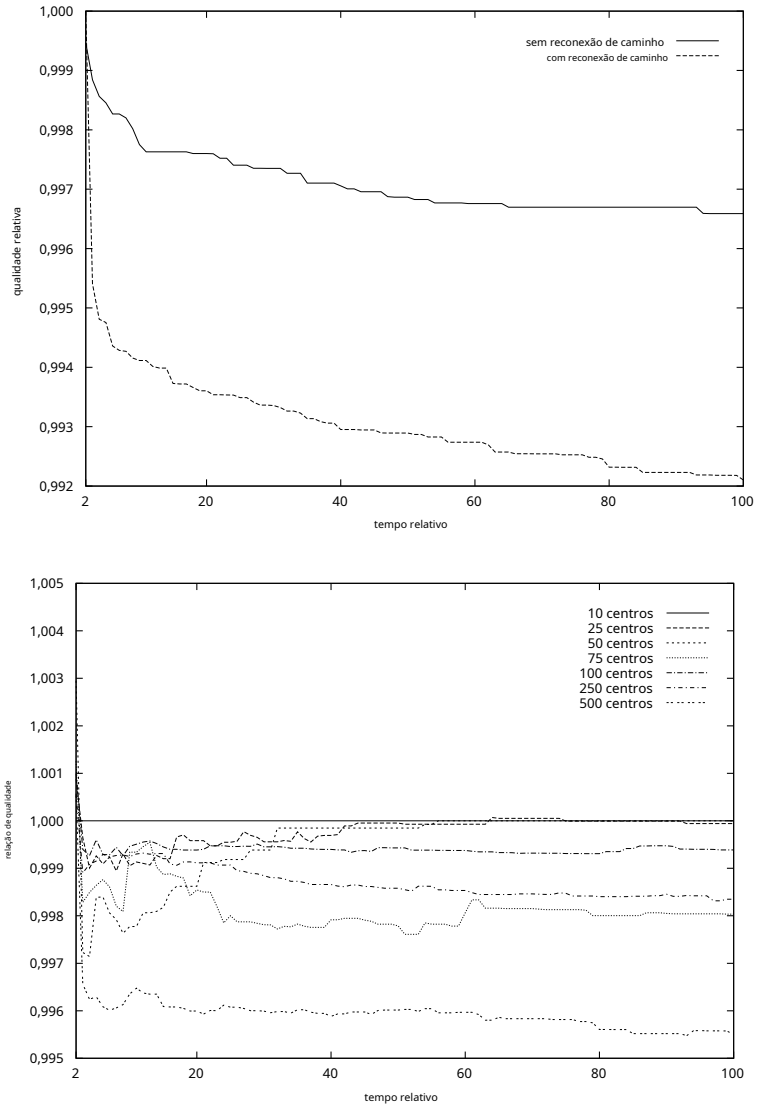

Figura 2.8. GRASP com reconexão de caminho versus GRASP puro na instância TSPLIB fl1400.

Se um custo $c_{i, j, k}$ está associado a cada trigêmeo (i, j, k) ∈ eu × J × K, então o AP3 consiste em encontrar um subconjunto UMA ∈ eu × J × K de n trigêmeos como
que cada elemento de eu ∪ J ∪ K ocorre em exatamente um tripleto de UMA, e a soma dos custos dos trigêmeos escolhidos é minimizada. O AP3 é

NP-difícil [21, 22]. Uma formulação baseada em permutação para AP3 é

$$\min_{p,\, q \in \pi_N} \sum_{i=1}^{n} c_{ip(i)q(i)},$$

Onde π_N denota o conjunto de todas as permutações do conjunto de inteiros $N = \{1, 2, \ldots, n\}$.

Aiex [1] e Aiex et al. [3] descreve um GRASP com redirecionamento de caminho para AP3. Os resultados computacionais mostram claramente que este GRASP para AP3 se beneficia da reconexão de caminhos e se compara bem com as heurísticas propostas anteriormente para este problema. GRASP com path-relinking foi capaz de melhorar a qualidade da solução das heurísticas propostas por Balas e Saltzman [6], Burkard et al. [8], e Crama e Spieksma [11] em todas as instâncias propostas nesses documentos.

A fase de construção GRASP constrói uma solução viável S selecionando n trigêmeos, um de cada vez. A solução S está inicialmente vazio e o conjunto C de trigêmeos candidatos é inicialmente o conjunto de todos os trigêmeos. Para selecionar o p o trigêmeo (p = 1,. . . , n - 1) a ser adicionado à solução, uma lista de candidatos restrita C' é definido para incluir todos os trigêmeos (i, j, k) no candidato definir C tendo custo $c_{ijk} \leq \underline{c} + \alpha (c - \underline{c})$, Onde

$\underline{c} = \min \{ c_{ijk} \parallel (i, j, k) \in C\}$ e $c = \max \{ c_{ijk} \parallel (i, j, k) \in C\}$.

Trigêmeo $(e_{u_p}, j_p, k_p) \in C'$ é escolhido aleatoriamente e adicionado à solução, ie $S = S \cup \{(e_{u_p}, j_p, k_p)\}$. Uma vez (e_{u_p}, j_p, k_p) for selecionado, o conjunto de trigêmeos candidatos deve ser ajustado para levar em consideração que (e_{u_p}, j_p, k_p) é parte de a solução. Qualquer trigêmeo (i, j, k) de tal modo que $i = i_p$ ou $j = j_p$ ou $k = k_p$ é removido do conjunto atual de trigêmeos candidatos. Após n - 1 trigêmeo foram selecionados, o conjunto C de trigêmeos candidatos contém um último trio que é adicionado a S, completando assim a fase de construção.

No procedimento de busca local, a solução atual é aprimorada pesquisando-se sua vizinhança por uma solução melhor. A solução do AP3 pode ser representada por um par de permutações (p, q). Para uma solução $p, q \in \pi_N$, a vizinhança de 2 trocas é $N_2(p, q) = \{p', q' \mid d(p, p') + d(q, q') = 2\}$, onde $d(s, s') = |\{eu \mid si) 6 = s'(i)\}|$.

Na busca local, cada custo de uma solução de vizinhança é comparado com o custo da solução atual. Se o custo do vizinho for menor, a solução é atualizada, a busca é interrompida e uma busca na nova vizinhança é inicializada. A busca local termina quando nenhum vizinho da solução atual tem um custo menor do que a solução atual.

A reconexão do caminho é feita entre uma solução inicial

$$S = \{(1, j_{S1}, k_{S1}), (2, j_{S2}, k_{S2}), \ldots, (n, j_{Sn}, k_{Sn})\}$$

e uma solução orientadora

$$T = \{(1, j_{T1}, k_{T1}), (2, j_{T2}, k_{T2}), \ldots, (n, j_{Tn}, k_{Tn})\}.$$

Deixe a diferença simétrica entre S e T ser definido pelos seguintes dois conjuntos de índices:

$$\delta J = \{i = 1, \ldots, n \parallel j_{Si} \neq j_{Teu}\}$$

e

$$\delta K = \{i = 1, \ldots, n \parallel k_{Si} \neq k_{Teu}\}.$$

Uma solução intermediária do caminho é visitada em cada etapa do pathrelinking. Dois tipos elementares de movimentos podem ser executados. Em um movimento tipo um, trigêmeos

$$\{(eu_1, j_1, k_1), (eu_2, j_2, k_2)\}$$

são substituídos por trigêmeos

$$\{(eu_1, j_2, k_1), (eu_2, j_1, k_2)\}.$$

Em um movimento do tipo dois, trigêmeos

$$\{(eu_1, j_1, k_1), (eu_2, j_2, k_2)\}$$

são substituídos por

$$\{(eu_1, j_1, k_2), (eu_2, j_2, k_1)\}.$$

Definir δJ orienta movimentos do tipo um, enquanto δK orienta movimentos do tipo dois. Para tudo eu ∈ δJ, deixar q ser tal que $j_{Tq} = j_{Seu}$. Um movimento do tipo um substitui trigêmeos

$$\{(eu, j_{Seu}, k_{Seu}), (q, j_{Sq}, k_{Sq})\}$$

de

$$\{(eu, j_{Seu}, k_{Seu}), (q, j_{Seu}, k_{Sq})\}.$$

Para todos eu ∈ δK, deixar q ser tal que $k_{Tq} = k_{Seu}$. Um movimento do tipo dois substitui trigêmeos

$$\{(eu, j_{Seu}, k_{Seu}), (q, j_{Sq}, k_{Sq})\}$$

de

$$\{(eu, j_{Seu}, k_{Sq}), (q, j_{Sq}, k_{Seu})\}.$$

Em cada etapa, o movimento que produz a solução menos custosa é selecionado e o índice correspondente é excluído de qualquer δJ ou δK. Esta o processo continua até que haja apenas dois índices de movimento restantes em um dos conjuntos δJ ou δK. Nesta fase, qualquer um desses dois movimentos resulta no

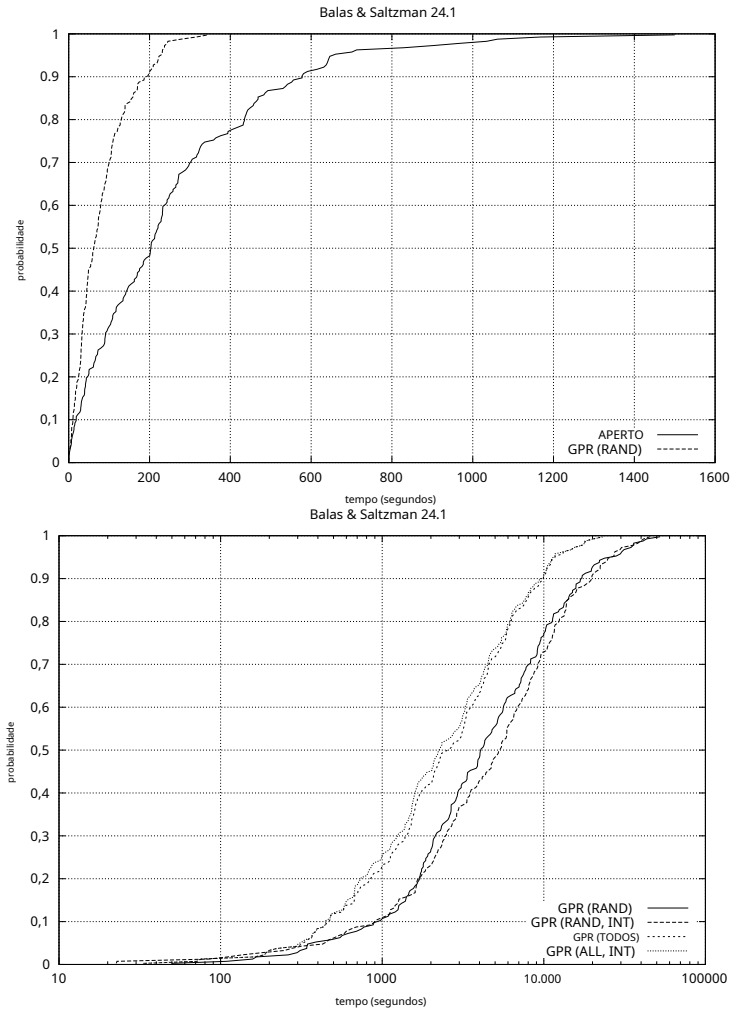

Figura 2.9. O gráfico da esquerda mostra as distribuições de probabilidade empírica de tempo para o valor alvo para GRASP e GRASP com revinculação de caminho (seleção aleatória de uma solução guia do conjunto de elite). O gráfico à direita mostra as distribuições de probabilidade empírica de tempo para o valor alvo para diferentes variantes de GRASP com reconexão de caminho.

solução orientadora e, portanto, não são realizadas. A melhor solução encontrada no caminho é retornada pelo procedimento.

Os gráficos na Figura 2.9 ilustram como GRASP com reconexão de caminho se compara com GRASP puro e como as diferentes variantes de GRASP com reconexão de caminho se comparam. As variantes de GRASP com reconexão de caminho testadas foram: seleção aleatória de uma solução guia [GPR (RAND)]; seleção aleatória de uma solução guia e religação periódica de todos os elementos no pool [GPR (RAND, INT)]; seleção de todos os elementos da piscina como guia

soluções de integração [GPR (ALL)]; e seleção de todos os elementos do pool como soluções de orientação com religação periódica de todos os elementos do pool [GPR (ALL, INT)]. Os algoritmos foram executados 200 vezes (usando diferentes sementes iniciais para o gerador de números aleatórios) na instância 24.1 de Balas e Saltzman [6], parando quando um valor de solução melhor que um determinado valor alvo foi encontrado. O experimento comparando GRASP puro com GRASP com revinculação de caminho usou um valor alvo de 17, enquanto aquele comparando as diferentes variantes de GRASP com revinculação de caminho usou um valor alvo mais difícil de 7. O gráfico à esquerda mostra o benefício de usar pathrelinking em GRASP. O gráfico à direita mostra que as variantes que usam a reconexão de caminho com todas as soluções de elite têm uma probabilidade maior de encontrar uma solução alvo em um determinado período de tempo do que as variantes que usam a reconexão de caminho com uma única solução de elite selecionada aleatoriamente. O uso de intensificação periódica não parece influenciar tanto as distribuições.

2.5 CONCLUSÕES E EXTENSÕES

Este artigo revisou os avanços e aplicações recentes do GRASP com reconexão de caminhos. Fornecendo uma breve discussão de cada componente do GRASP com religação de caminho e mostrando exemplos de como tais heurísticas podem ser implementadas para problemas de otimização combinatória, como roteamento de PVC, design de rede de 2 caminhos, atribuição de 3 índices e p- mediana, esperamos que este artigo sirva como um guia para o leitor reunir outro GRASP com heurísticas de reconexão de caminhos.

A reconexão de caminhos é um grande aprimoramento do procedimento básico de busca adaptativa aleatória gananciosa (GRASP), levando a melhorias significativas no tempo e na qualidade da solução. Ele adiciona um mecanismo de memória eficaz ao GRASP, fornecendo uma estratégia de intensificação que explora as trajetórias que conectam as soluções GRASP e as melhores soluções de elite produzidas anteriormente durante a pesquisa. Os resultados numéricos resumidos para os quatro problemas listados acima ilustram claramente os benefícios obtidos pela combinação de GRASP com reconexão de caminho.

Dentro revinculação do caminho evolutivo usado na fase de intensificação pós-otimização, uma nova geração de soluções de elite é gerada a partir da população atual no pool de soluções de elite, aplicando pathrelinking entre todos os pares de soluções nesta população. As soluções obtidas por cada operação de reconexão de caminho são testadas para inclusão na população da próxima geração, seguindo as regras usuais usadas no gerenciamento de pool. Essa estratégia foi usada com sucesso para o problema de Steiner em gráficos por Ribeiro et al. [50], para o p- problema mediano por Resende

e Werneck [44], e para o problema de localização de instalação não capacitada por Resende e Werneck [46].

A reconexão de caminhos também pode ser usada como um extrator de solução para métodos de população. Em particular, a reconexão de caminhos foi recentemente aplicada com sucesso como uma estratégia de crossover generalizada para gerar derivações otimizadas no contexto de um algoritmo genético para o problema de filogenia [51].

O fato de que o tempo de cálculo para encontrar um valor de solução alvo usando GRASP com redirecionamento de caminho ajusta-se a uma distribuição exponencial de dois parâmetros (cf. Seção 2.3, ver [1, 2, 4]) tem uma consequência importante em implementações paralelas de GRASP com reconexão de caminho: speedups lineares proporcionais ao número de processadores podem ser facilmente observados em estratégias paralelas independentes. Além disso, a reconexão de caminhos oferece um mecanismo muito eficaz para a implementação de estratégias cooperativas paralelas [12]. Nesse caso, a cooperação entre processadores é reforçada por um processador mestre que armazena e lida com um conjunto comum de soluções de elite que é compartilhado por todos os processadores escravos que executam GRASP com redirecionamento de caminho. Implementações cuidadosas fazendo uso apropriado dos recursos do computador podem levar a acelerações ainda maiores e a algoritmos paralelos muito robustos, veja por exemplo [31, 48, 49, 52]. Os resultados obtidos para o problema de projeto de rede de 2 caminhos são ilustrados na Figura 2.10, mostrando a aceleração obtida pela estratégia cooperativa em relação à independente em um cluster de oito processadores. Melhorias muito maiores podem ser obtidas com mais processadores.

Finalmente, notamos que a reconexão de caminhos também pode ser usada com sucesso em conjunto com implementações de outras metaheurísticas, como VNS e colônias de formigas, como relatado recentemente, por exemplo, em [5, 17].

AGRADECIMENTOS

A maior parte deste trabalho faz parte de suas dissertações e foi realizado em conjunto com os seguintes atuais e ex-mestres e Ph.D. alunos da Pontifícia Universidade Católica do Rio de Janeiro, Brasil, aos quais agradecemos todos: RM Aiex, SA Canuto, SL Martins, M. Prais, I. Rosseti, MC Souza, E. Uchoa, DS Vianna e RF Werneck.

REFERÊNCIAS

[1] RM Aiex. Uma investigaç¸ã É experimental da distribuiçãoDe probabilidade de tempo de soluç¸ã˜o em heur´ısticas GRASP e sua aplicaç¸ã ~o na análise de implementaç¸õ˜es paralelas. Tese de doutorado, Departamento

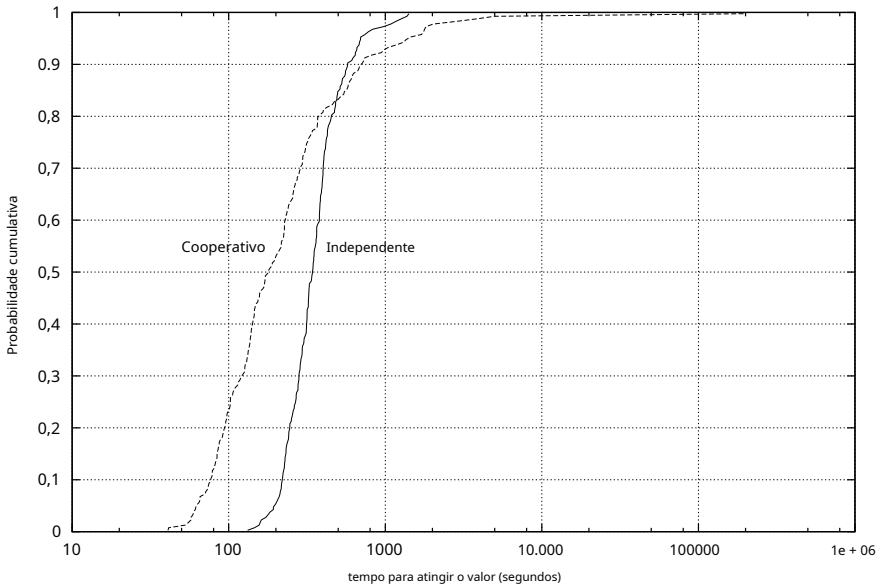

Figura 2.10. Distribuições de probabilidade de valor de tempo para atingir o objetivo em uma instância do problema de design de rede de 2 caminhos para implementações paralelas cooperativas e independentes de GRASP com revinculação de caminho em um cluster Linux com oito processadores.

Doutor em Ciência da Computação, Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2002.

[2] RM Aiex, S. Binato e MGC Resende. GRASP paralelo com reconexão de caminhos para programação de job shop. Computação Paralela, 29: 393-430, 2003.

[3] RM Aiex, MGC Resende, PM Pardalos e G. Toraldo. GRASP com revinculação de caminho para o problema de atribuição de três índices. Relatório técnico, AT&T Labs Research, Florham Park, NJ 07733, 2000. Para aparecer em INFORMS J. on Computing.

[4] RM Aiex, MGC Resende e CC Ribeiro. Distribuição de probabilidade função de tempo de solução em GRASP: Uma investigação experimental. Journal of Heuristics, 8: 343–373, 2002.

[5] DJ Aloise, D. Aloise, CTM Rocha, CC Ribeiro, José C. Ribeiro Filho e Luiz SS Moura. Programação de plataformas de workover para produção de petróleo em terra. Matemática Aplicada Discreta, aparecer.

[6] E. Balas e MJ Saltzman. Um algoritmo para os três índices problema de atribuição. Oper. Res., 39: 150–161,191.

[7] S. Binato, H. Faria Jr. e MGC Resende. Greedy randomizado revinculação de caminho adaptável. Em JP Sousa, editor, Anais da IV Conferência Internacional Metaheurística, páginas 393–397, 2001.

[8] RE Burkard, R. Rudolf e GJ Woeginger. Tridimensional problemas de atribuição axial com coeficientes de custo decomponíveis. Matemática Aplicada Discreta, 65: 123–139, 1996.

[9] SA Canuto, MGC Resende e CC Ribeiro. Pesquisa local com perturbação para os problemas da árvore de Steiner de coleta de prêmios em gráficos. Redes, 38: 50–58, 2001.

[10] G. Cornuejols, ML Fisher e GL Nemhauser. Localização de contas bancárias para otimizar o fl utuador: Um estudo analítico de algoritmos exatos e aproximados. Ciência da Administração, 23: 789–810,1977.

[11] Y. Crama e FCR Spieksma. Algoritmos de aproximação para problemas de atribuição tridimensional com desigualdades triangulares. European Journal of Operational Research, 60: 273–279,1992.

[12] V.-D. Cung, SL Martins, CC Ribeiro e C. Roucairol. Estratogies para a implementação paralela de metaheurísticas. Em CC Ribeiro e P. Hansen, editores, Ensaios e pesquisas em metaheurísticas, páginas 263–308. Kluwer Academic Publishers, 2002.

[13] G. Dahl e B. Johannessen. O problema da rede de 2 caminhos. Internettrabalho, 43: 190–199, 2004.

[14] TA Feo e MGC Resende. Uma heurística probabilística para um com problema de cobertura de conjunto supostamente difícil. Pesquisa Operacional Letras, 8: 67–71,1989.

[15] TA Feo e MGC Resende. Busca adaptativa aleatória gananciosa procedimentos. Journal of Global Optimization, 6: 109–133, 1995.

[16] TA Feo, MGC Resende e SH Smith. Um ganancioso randomizado procedimento de pesquisa adaptativa para conjunto independente máximo. Pesquisa Operacional, 42: 860–878,1994.

[17] P. Festa, PM Pardalos, MGC Resende e CC Ribeiro. Correuheurísticas dominadas para o problema de corte máximo. Métodos e software de otimização, 7: 1033–1058, 2002.

[18] P. Festa e MGC Resende. GRASP: Uma bibliografia anotadaphy. Em CC Ribeiro e P. Hansen, editores, Ensaios e pesquisas sobre metaheurísticas, páginas 325–367. Kluwer Academic Publishers, 2002.

[19] C. Fleurent e F. Glover. Estratégia construtiva multistart aprimorada gies para o problema de atribuição quadrática usando memória adaptativa. INFORMS Journal on Computing, 11: 198–204, 1999.

[20] B. Fortz e M. Thorup. Engenharia de tráfego da Internet por meio da otimização pesos ospf. Dentro Proc. IEEE INFOCOM 2000 - The Conference on Computer Communications, páginas 519–528, 2000.

[21] AM Frieze. Complexidade de um problema de atribuição tridimensional. European Journal of Operational Research, 13: 161-164,1983.

[22] MR Garey e DS Johnson. Computadores e intratabilidade - A guia para a teoria da NP-completude. WH Freeman and Company, 1979.

[23] F. Glover. Pesquisa tabu e programação de memória adaptativa - Adavanços, aplicações e desafios. Em RS Barr, RV Helgason e JL Kennington, editores, Interfaces em Ciência da Computação e Pesquisa Operacional, páginas 1–75. Kluwer, 1996.

[24] F. Glover. Métodos multi-start e de oscilação estratégica - Principles para explorar a memória adaptativa. Em M. Laguna e JL Gonz´álesVelarde, editores, Ferramentas de computação para modelagem, otimização e simulação: interfaces em ciência da computação e pesquisa operacional, páginas 1–24. Kluwer, 2000.

[25] F. Glover e M. Laguna. Pesquisa Tabu. Kluwer, 1997.

[26] F. Glover, M. Laguna e R. Mart˝ı. Fundamentos da pesquisa dispersa e reconexão de caminho. Controle e cibernética, 39: 653–684, 2000.

[27] O. Kariv e L. Hakimi. Uma abordagem algorítmica para locais de rede problemas de instalação, Parte II: O p- medianas. SIAM Journal of Applied Mathematics, 37: 539-560, 1979.

[28] M. Laguna e R. Mart˝ı. GRASP e revinculação de caminho para 2 camadas minimização do cruzamento em linha reta. INFORMS Journal on Computing, 11: 44–52,1999.

[29] Y. Li, PM Pardalos e MGC Resende. Um ganancioso randomizado procedimento de pesquisa adaptativa para o problema de atribuição quadrática. Dentro PM Pardalos e H. Wolkowicz, editores, Atribuição quadrática e problemas relacionados, volume 16 de Série DIMACS em Matemática Discreta e Ciência da Computação Teórica, páginas 237–261. American Mathematical Society, 1994.

[30] SL Martins, PM Pardalos, MGC Resende e CC Ribeiro. Procedimentos de busca adaptativa aleatória gananciosa para o problema de Steiner em gráficos. Em PM Pardalos, S. Rajasekaran e J. Rolim, editores, Métodos de randomização em design algorítmico, volume 43 de Série DIMACS em Matemática Discreta e Ciência da Computação Teórica, páginas 133–145. American Mathematical Society, 1999.

[31] SL Martins, CC Ribeiro e I. Rosseti. Aplicativos e paralelos implementações de metaheurísticas em design e roteamento de redes. Notas de aula em ciência da computação, 3285: 205–213, 2004.

[32] CA Oliveira, PM Pardalos e MGC Resende. GRASP com revinculação de caminho para o QAP. Em Toshihide Ibaraki e Yasunari Yoshitomi, editores, Anais da Quinta Conferência Internacional Metaheurística, páginas 57–1 - 57–6, 2003.

[33] WP Pierskalla. O método de tri-substituição para os três problema de atribuição multidimensional. CORS J., 5: 71–81,1967.

[34] M. Prais e CC Ribeiro. GRASP reativo: um aplicativo para um problema de decomposição de matrizes na atribuição de tráfego TDMA. INFORMS Journal on Computing, 12: 164–176, 2000.

[35] MGC Resende e RF Werneck. Sobre a implementação de um procedimento de pesquisa local baseado em troca para o p- problema mediano. Dentro RE Ladner, editor, Anais do Quinto Workshop sobre Engenharia de Algoritmos e Experimentos (ALENEX'03), páginas 119–127. SIAM, 2003.

[36] MGC Resende. Computando soluções aproximadas do maximãe cobrindo o problema com o GRASP. Journal of Heuristics, 4: 161-171, 1998.

[37] MGC Resende, TA Feo e SH Smith. Algoritmo 787: Forsub-rotinas trans para solução aproximada de problemas de conjuntos independentes máximos usando GRASP. Transações ACM em software matemático, 24: 386–394, 1998.

[38] MGC Resende, PM Pardalos e Y. Li. Algoritmo 754: Fortran sub-rotinas para solução aproximada de problemas de atribuição quadrática densa usando GRASP. Transações ACM em software matemático, 22: 104-118, 1996.

[39] MGC Resende, LS Pitsoulis e PM Pardalos. Fortran subroutines para calcular soluções aproximadas de problemas MAX-SAT usando GRASP. Matemática Aplicada Discreta, 100: 95-113, 2000.

[40] MGC Resende e CC Ribeiro. A GRASP para gráfico planarização Redes, 29: 173–189, 1997.

[41] MGC Resende e CC Ribeiro. Um GRASP com reconexão de caminho para roteamento de circuito virtual privado. Redes, 41: 104-114, 2003.

[42] MGC Resende e CC Ribeiro. GRASP e reconexão de caminho: Avanços e aplicações recentes. Em Toshihide Ibaraki e Yasunari Yoshitomi, editores, Anais da Quinta Conferência Internacional Metaheurística, páginas T6–1 - T6–6, 2003.

[43] MGC Resende e CC Ribeiro. Adaptativo aleatório ganancioso procedimentos de pesquisa. Em F. Glover e G. Kochenberger, editores, Manual de metaheurísticas, páginas 219–249. Kluwer Academic Publishers, 2003.

[44] MGC Resende e RF Werneck. Um GRASP com reconexão de caminho para o problema da p-mediana. Relatório técnico TD-5E53XL, AT&T Labs Research, 2002.

[45] MGC Resende e RF Werneck. Uma heurística híbrida para o p-problema mediano. Journal of Heuristics, 10: 59–88, 2004.

[46] MGC Resende e RF Werneck. Uma heurística híbrida multistart para o problema de localização de instalação não capacitada. European Journal of Operational Research, aparecer.

[47] CC Ribeiro e MGC Resende. Algoritmo 797: Fortran subrou-tines para solução aproximada de problemas de planarização de grafos usando GRASP. Transações ACM em software matemático, 25: 341-352, 1999.

[48] CC Ribeiro e I. Rosseti. Um GRASP paralelo para o 2-path problema de projeto de rede. Notas de aula em ciência da computação, 2004: 922–926, 2002.

[49] CC Ribeiro e I. Rosseti. Implementos cooperativos paralelos eficientes tações das heurísticas GRASP. Relatório técnico, Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2005.

[50] CC Ribeiro, E. Uchoa e RF Werneck. Um GRASP híbrido com perturbações para o problema de Steiner em gráficos. INFORMS Journal on Computing, 14: 228–246, 2002.

[51] CC Ribeiro e DS Vianna. Um algoritmo genético para o phy-Problema de logeny usando uma estratégia de crossover otimizada baseada na reconexão de caminhos. Revista Tecnologia da Informaç,ã ˜, 3 (2): 67–70, 2003.

[52] I. Rosseti. Heur´ısticas para o problema de s´íntese de redes a 2-caminhos. Tese de doutorado, Departamento de Ciência da Computação, Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, julho de 2003.

[53] MC Souza, C. Duhamel e CC Ribeiro. Um GRASP com caminho heurística de religação para o problema de spanning tree mínimo capacitado. Em MGC Resende e J. Souza, editores, Metaheurísticas: Tomada de Decisão no Computador, páginas 627–658. Kluwer Academic Publishers, 2003.

[54] MB Teitz e P. Bart. Métodos heurísticos para estimar o gen-mediana eralizada do vértice de um gráfico ponderado. Pesquisa Operacional, 16: 955–961,1968.

[55] R. Whitaker. Um algoritmo rápido para o intercâmbio ganancioso de grandes agrupamento em escala e prolems de localização mediana. INFOR, 21: 95–108, 1983.