GRASP COM RECONEXÃO POR CAMINHOS PARA O PROBLEMA EIXO-RAIO COM ALOCAÇÃO SIMPLES

Alexandre Xavier Martins
xmartins@decea.ufop.br
Yasmine de Melo Costa
yasmine_melo@yahoo.com.br
Departamento de Engenharia de Produção
Universidade Federal de Ouro Preto
Campus de João Monlevade – MG – Brasil

Ricardo Saraiva de Camargo
rcamargo@dep.ufmg.br
Departamento de Engenharia de Produção
Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil

RESUMO

Este trabalho apresenta uma heurística eficiente desenvolvida para o projeto de redes eixo-raio com alocação simples. Este é um problema importante com muitas aplicações nos sistemas de transporte de carga e passageiros, e sistemas de telecomunicação. Um GRASP eficiente, combinado com três diferentes tipos de estruturas de vizinhança, é usado para resolver o problema. Além disso, um esquema baseado em reconexão por caminhos é apresentado a fim de aprimorar a qualidade das soluções do GRASP. O GRASP com reconexão por caminhos proposto claramente superou três conhecidas soluções heurísticas da literatura para as instâncias testadas. A abordagem por reconexão por caminhos desempenhou um papel importante na robustez do algoritmo proposto, tanto em termos de tempo computacional quanto na qualidade das soluções.
PALAVRAS CHAVE. GRASP. Reconexão por Caminhos. Desenho de Redes Eixo-Raio.

ABSTRACT

The uncapacitated single allocation hub location problem is addressed in this work. This is a very important problem with many applications in cargo, passenger and telecommunication systems. A very efficient GRASP combined with three different types of neighborhood structure is used to tackle the problem. Furthermore, a Path Relinking (PR) scheme is implemented in order to improve the quality of the GRASP's solutions. The proposed GRASP with PR clearly outperformed three well known heuristics of the literature for the selected instances. The PR approaches played a major role in the robustness of the proposed algorithm, both in terms of computational time and of solution quality.
KEYWORDS. GRASP. Path Relinking. Design of Hub-and-spoke networks.

1. Introdução

Redes eixo-raio (E-R) são utilizadas com grande frequência em sistemas de comunicação, de transporte e sistemas logísticos, como por exemplo: companhias aéreas de passageiros (Aykin

(1995)), indústrias de transporte por caminhões (Cunha e Silva (2007)), sistemas de telecomunicações (Klincewicz (1998)), entre outras. Devido às diversas aplicações, esse tipo de rede se tornou um importante campo de pesquisa na área de localização (Alumur e Kara (2008); Campbell et al. (2002)).

Nas redes E-R, a comunicação entre os nós não acontece de forma direta, mas através de nós concentradores que são responsáveis pela agregação, roteamento e distribuição do fluxo de demanda entre os diferentes pontos de origem/destino. De uma forma geral, o projeto de redes eixo-raio envolve a localização dos nós concentradores e a alocação dos nós não concentradores aos concentradores instalados (Campbell et al. (2002)).

A comunicação em uma rede E-R acontece da seguinte forma: o fluxo de demanda primeiramente é coletado do nó origem pelo concentrador em que o mesmo está alocado; após isso, o fluxo de demanda coletado é encaminhado para o concentrador em que o destino está atribuído; finalmente, o concentrador faz a distribuição desse fluxo ao respectivo destino. A agregação dos fluxos entre os nós concentradores permite a utilização de meios de transportes mais eficientes, e de maior capacidade de volume de tráfego nas conexões entre concentradores, resultando assim em menor custo de transporte por unidade (O'Kelly e Bryan (1998)). Dessa forma, economia de escala é obtida. Essa economia de escala geralmente é representada por um fator constante de desconto no custo de transporte entre os concentradores $(0 < \alpha < 1)$.

Diversas variantes desse problema são encontradas na literatura, cada uma contemplando diferentes características, tais como: alocação simples ou múltipla dos nós não concentradores aos concentradores; número de concentradores pré-determinado ou variável; permissão de ligação direta entre nós não concentradores; restrição de capacidade; efeitos de congestionamento. Uma revisão detalhada dessas diferentes variantes pode ser encontrada em Alumur e Kara (2008).

Neste artigo, uma heurística é proposta para resolução do projeto de redes com alocação simples. Uma heurística construtiva e diferentes procedimentos de busca local para o problema são apresentados. Além disso, também é apresentada uma estratégia baseada em reconexão por caminhos. Nos experimentos computacionais realizados, a abordagem proposta se mostrou superior à Busca Tabu de Silva e Cunha (2009), método considerado estado da arte na literatura para o projeto de redes E-R com alocação simples, tanto na qualidade das soluções obtidas quanto em tempo de processamento gasto. Além disso, o método proposto se mostrou eficiente na obtenção da solução ótima das instâncias consideradas. O artigo é organizado da seguinte maneira: a Seção 2 apresenta uma revisão bibliográfica. A heurística proposta é detalhada na Seção 3. A Seção 4 apresenta os testes computacionais e resultados obtidos. As conclusões são descritas na Seção 5.

2. Exame da literatura

A primeira formulação quadrática para o projeto de redes E-R foi produzida por O'Kelly (1987), que também propôs uma solução heurística para o problema. Já Campbell (1994) apresentou uma formulação linear e Abdinnour-Helm e Venkataramanan (1998) desenvolveram uma formulação em programação inteira quadrática baseada no fluxo de multi-produtos. Abdinnour-Helm e Venkataramanan (1998) ainda propuseram um método *branch-and-bound* que utiliza a estrutura de fluxo multi-produtos para obter limites inferiores e um algoritmo genético (AG) para encontrar bons limites superiores.

Como esse problema pertence à classe dos NP-difíceis (Campbell et al. (2002)), a maioria dos algoritmos propostos para sua resolução são baseados em heurísticas ou metaheurísticas. Uma heurística híbrida baseada em AG e busca tabu foi proposta por Abdinnour-Helm (1998). O AG determina o número e localização dos concentradores, enquanto a busca tabu trabalha na alocação dos nós não concentradores aos concentradores instalados. Os resultados encontrados por esse método superaram o AG de Abdinnour-Helm e Venkataramanan (1998). Outro método baseado em AG, que obteve melhor performance que Abdinnour-Helm (1998), foi proposto por Topcuoglu et al. (2005). O AG desenvolvido é responsável por determinar o número e localização dos concentradores, bem

como alocar os demais nós aos concentradores instalados.

Cunha e Silva (2007) desenvolveram uma combinação eficiente das heurísticas AG e *Simulated Annealing* (SA). Nesse método, todas as soluções passam por uma busca local após as fases de cruzamento e mutação, objetivando melhorar a alocação dos nós não concentradores aos concentradores instalados. Além disso, os autores fizeram um estudo de caso com uma empresa de transporte no Brasil. Chen (2007) apresentou também uma heurística híbrida baseada em SA, busca tabu e procedimentos de melhoria. Esse método híbrido obteve melhor performance que o AG de Topcuoglu et al. (2005).

Mais recentemente, Silva e Cunha (2009) desenvolveram 3 variantes da heurística busca tabu com *multi-start*, obtendo resultados satisfatórios que superam os demais métodos desenvolvidos até então. Finalmente, Filipovic et al. (2009) propuseram um AG com um novo esquema de codificação das soluções. Experimentos computacionais demonstraram que esse método possui boa performance quando comparado com Topcuoglu et al. (2005) e Chen (2007).

Abordagens exatas também são consideradas para o referido problema. A pesquisa feita por Labbé e Yaman (2004) trabalha com duas formulações e desigualdades válidas que podem ser separadas em tempo polinomial, mas os autores não apresentaram resultados computacionais. O método de decomposição de Benders foi aplicado por de Castro (2010) obtendo resultados satisfatórios quando comparados com o CPLEX.

3. Desenvolvimento

Nesta seção descrevemos como cada um dos módulos do algoritmo proposto foram implementados. Descrevemos inicialmente o método de geração de soluções, depois cada uma das buscas locais é descrita, por fim, apresentamos a reconexão por caminhos.

3.1. Método de Construção

A heurística construtiva implementada é baseada na fase de construção do GRASP (Feo e Resende (1989)). Nesse procedimento, as soluções são inicialmente configuradas com somente um nó concentrador. Em cada iteração um nó diferente é selecionado através de uma lista circular, sendo então melhorada pela adição de novos concentradores até que o custo total não seja reduzido.

Resumindo, calculsa-se o custo marginal de redução β_j de configurar um nó não concentrador j como concentrador da solução e realocar todos os outros nós não concentradores ao concentrador mais próximo. Quando $\beta_j \geq 0$, ou seja a mudança aumentou o custo total, o nó j é adicionado à lista NL, que contém os nós descartados como candidato a concentrador da solução em questão. Após calcular os custos marginais de redução para todos nós não concentradores, determina-se o custo marginal de redução máximo (β_{max}) e mínimo (β_{min}) entre os nós $j \notin NL$. Assim, construímos uma lista restrita de candidatos L, onde $L = \{j | \beta_j \leq \beta_{min} + \lambda(\beta_{max} - \beta_{min})\}$, e $\lambda \in [0; 1]$ representa um parâmetro que controla o nível de aleatoriedade no processo de construção e o tamanho de L. Um nó não concentrador é aleatoriamente escolhido de L, o restante dos nós não concentradores são realocados ao nó concentrador mais próximo.

Para cada solução, os nós não concentradores são selecionados por meio de um processo aleatório iterativo até que $L = \{\emptyset\}$. Cabe ressaltar que, em cada iteração, novos valores para β_j, β_{max}, β_{min}, NL e L são obtidos.

3.2. Busca Local

Dada a natureza do problema abordado, uma grande variedade de estratégias de busca local pode ser explorada: A busca local com realocação de não concentradores, busca local com inserção de novos concentradores, ou também, remoção de concentradores, a troca de um concentrador instalado por um nó não-concentrador. Neste trabalho, nós propomos três buscas locais, as quais chamamos "Realocação", "Troca de concentrador"e "Remoção de concentrador".

Para usar o benefício de cada busca local, uma possível abordagem é aplicar o método VND proposto por Mladenovic e Hansen (1997). A técnica VND consiste em explorar sistematicamente diferentes vizinhanças de uma solução através de uma busca local de maneira a reduzir o risco de se ficar preso num ótimo local. Durante a busca, somente movimentos de melhora são aceitos e sempre que uma nova solução de melhora é encontrada, o método reinicia da primeira estrutura de vizinha considerada.

3.2.1. Realocação

Nesta busca local somente movimentos de realocação de nós não concentradores são permitidos. Portanto, o número de concentradores e os concentradores instalados não são alterados nesse procedimento. Essa busca local objetiva melhorar os custos de transmissão em uma rede. Nesse procedimento, todas as possíveis realocações de cada nó não concentrador aos concentradores instalados são testadas. Após todos os movimentos, a melhor realocação é efetuada, se gerar diminuição nos custos. A Figura 1 apresenta todos os possíveis movimentos de realocação do nó não concentrador 3.

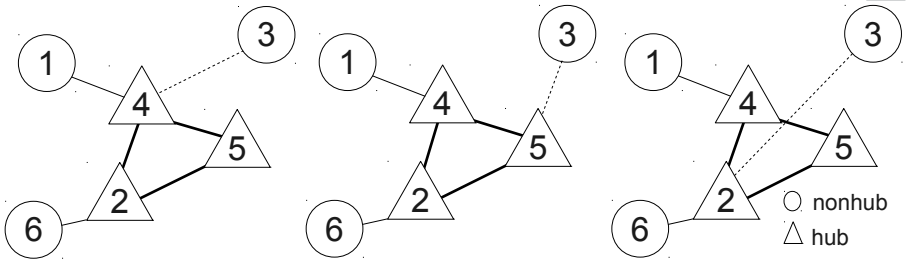

Figura 1: Exemplo de todas as possíveis realocações de um nó.

3.2.2. Troca de Concentrador

A ideia desta busca local é muito simples: Trocar um concentrador instalado por um nó não concentrador. Em outras palavras, um concentrador é selecionado para ser transformado em um nó não concentrador, e um nó não concentrador é escolhido para se tornar um concentrador. Ao contrário de outros procedimentos de busca local similares presentes na literatura (Ver Silva e Cunha (2009)), na busca local proposta, a substituição só ocorre entre um concentrador e um dos nós não concentradores a ele atribuídos.

A Figura 2 ilustra a ideia por trás do processo de troca de um concentrador por um nó não concentrador. Neste exemplo, a parte (a) mostra uma solução s com 6 nós, 3 concentradores e 3 não concentradores. O concentrador 4 tem dois nós não concentradores alocados a ele, os nós 1 e 3. Primeiro, o concentrador 4 é selecionado para ser transformado em um nó não concentrador e o nó não concentrador 1 é escolhido para se tornar um concentrador (veja a parte (b) da Figura 2). Todos os antigos nós não concentradores, exceto o nó 1, do antigo concentrador 4 são reatribuídos ao concentrador mais próximo. Observe que o nó 3 estava atribuído ao concentrador 4 e agora está alocado ao concentrador 5 e que o nó 4 agora está alocado ao concentrador 2. A parte (c) mostra a transformação do nó 1 em um concentrador. Finalmente, veja a parte (d) da Figura 2, todos nós não concentradores são realocados ao concentrador mais próximo, agora observe a presença do novo concentrador no nó 1. Se as reatribuições melhorarem a solução, estas são implementadas e a solução é atualizada.

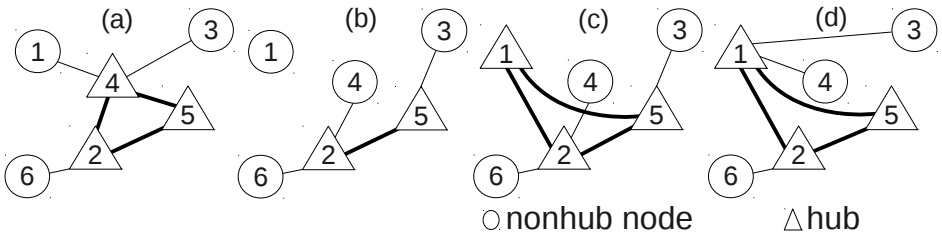

Figura 2: Trocando um concentrador por um não concentrador

3.2.3. Remoção de Concentrador

O procedimento de busca local que remove um concentrador busca alterar o número de concentradores da rede, explorando a vizinhança de uma solução através da retirada de concentradores da solução. Para isso, testa-se a remoção de cada um dos nós concentradores da solução, alocando os nós não concentradores ao concentrador mais próximo. Todas as remoções são testadas e a melhor remoção é feita caso haja melhora na função objetivo. A Figura 3 mostra a solução com 6 nós, 3 concentradores e 3 não concentradores, veja a parte (a). As partes (b) até (d) apresentam todas as possíveis configurações de concentradores removendo um concentrador a cada vez.

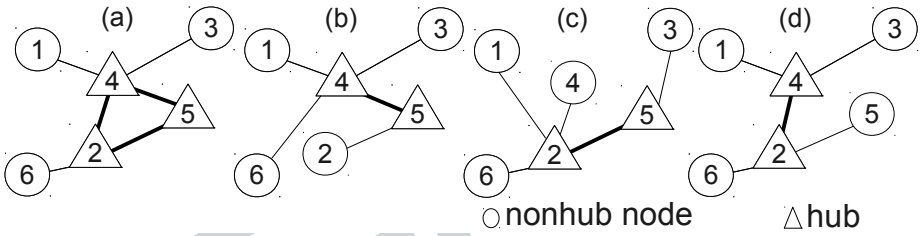

Figura 3: Removendo concentradores

3.3. Reconexão por Caminhos

Laguna e Martí (1999) combinaram com sucesso o método de reconexão por caminhos com o GRASP. Eles começaram com uma solução encontrada com o GRASP e uma solução guia selecionada aleatoriamente de um conjunto de três soluções elite, e então encontraram um caminho conectando as duas soluções por um processo de reconexão.

Em nossa implementação a reconexão por caminhos tem também duas soluções: Uma solução guia e uma solução guiada. A cada iteração a inserção de concentradores que está na solução guia, mas estão ausentes na solução guiada, e a exclusão de concentradores que estão na solução guiada, mas não estão na solução guia, são avaliadas. A cada verificação, os nós não concentradores são primeiramente alocados ao concentrador mais próximo e então uma busca local por realocação é executada. No final de cada iteração, a melhor mudança é implementada, mesmo que a solução guiada não seja melhorada. Quando o processo encontra uma solução melhor que a melhor solução global esta solução é atualizada. O método termina quando a solução guiada e a solução guia têm a mesma estrutura de concentradores.

A Figura 4 exemplifica a ideia da reconexão por caminhos adotada. Usando uma solução inicial (Letra (a) da Figura 4), diferentes estruturas de concentradores podem ser geradas inserindo ou removendo concentradores guiados pela solução guia representada na coluna (d). Por exemplo, na coluna (b) da Figura 4, a primeira estrutura de concentradores é encontrada definindo o nó 1 como um concentrador, enquanto a segunda e a terceira são obtidas removendo os concentradores 2

e 5, respectivamente. Assumindo que a melhor mudança desta iteração é aquela que gera a terceira estrutura, a segunda iteração começa desta solução. Duas diferentes mudanças são possíveis: A inserção do concentrador 1 e a remoção do concentrador 2, respectivamente, a primeira e segunda estrutura de concentrador da coluna (c). Novamente depois de assumir que a melhor mudança é definir o nó 1 como um concentrador, somente uma mudança é possível: Remover o concentrador 2. Fazendo isto, o método para, já que a solução atual tem agora a mesma estrutura de concentradores da solução guia. O trajeto do método é representado por setas em negrito.

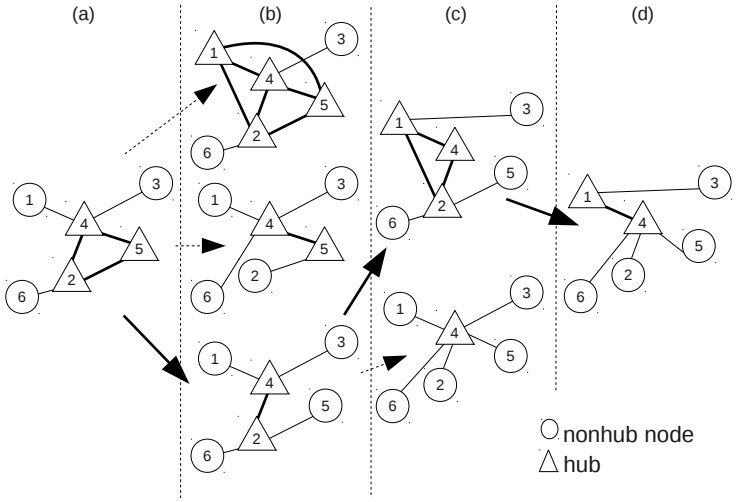

Figura 4: Reconexão entre duas soluções

De maneira a preencher ξ (conjunto de soluções elite), as primeiras θ diferentes soluções geradas são ordenadas da melhor (s^1) para a pior (s^θ) e inseridas no conjunto elite, onde $\theta = |\xi|$. Então, ao longo das iterações do GRASP, cada solução s' gerada é avaliada observando critérios de qualidade (se $fo(s') \leq fo(s^1)$ for verdadeiro) e diversidade (se $fo(s') < fo(s^\theta)$ e $\varphi(s', \xi) \geq \theta$ forem verdadeiros). O critério $\varphi(s', \xi) \geq \theta$, se atendido, garante que s' é suficientemente diferente das outras soluções no conjunto elite, onde φ é uma função que computa a "distância"entre a solução s' e as soluções presentes no conjunto elite. Para cada solução diferente em termos de nós concentradores a função $\varphi(s', \xi)$ soma uma unidade.

Se um dos critérios supracitados é observado, então s' é inserido em ξ. Além disso, como o valor de θ é constante, consequentemente, a cada nova inserção, a pior solução s^θ é removida. Vale a pena notar que o conjunto elite é atualizado dinamicamente, uma vez que a solução gerada pela reconexão por caminhos é testada para a inclusão em ξ.

4. Resultados Computacionais

A fim de avaliar o desempenho do GRASP proposto, quatro conjuntos de experimentos computacionais são projetados. Todos os testes computacionais foram realizados num computador com processador Dual Core com 2,53 Mhz e 4 Gb de memória RAM, funcionando no sistema operacional MS Windows Vista. Além disso, todos os algoritmos foram implementados em C++.

Uma base de dados padrão da literatura, *Australian Post* (AP), foi utilizada nos experimentos. Essa base de dados foi introduzida por Ernst e Krishnamoorthy (1996) e possui instâncias de $N = \{10; 20; 30; \ldots; 100; 130; 150; 170; 200\}$. As economias de escala foram configuradas como $\alpha = \{0.2; 0.4; 0.6; 0.8\}$. Assim, geraram-se 56 problemas teste representados por APn-α, onde n representa o número de nós de demanda e α é a economia de escala. É importante ressaltar que, como as instâncias da base de dados AP possuem custos fixos somente para os 50 primeiros nós, e todos nós são candidatos à concentrador, custos fixos foram aleatoriamente gerados para todos os

nós como em Camargo et al. (2008).

Em testes preliminares definimos que o parâmetro λ do GRASP é igual a 0.05. A ordem de execução das buscas locais é a mesma ordem apresentada neste trabalho, ou seja, Realocação, Troca de concentrador e remoção de concentrador. O tamanho do conjunto elite da reconexão por caminhos $|\xi| = 5$.

No primeiro conjunto de experimentos, Tabela 1, o melhor valor global (BestValue) é obtido para cada instância, depois de cinco execuções de cada algoritmo considerado. Para cada método, o desvio percentual relativo entre a melhor solução encontrada pelo algoritmo e o BestValue obtido para cada instância é calculado. O valor avaliado deste desvio assim como o valor do desvio mínimo para cada algoritmo no experimento são apresentados como Dev($\%$) e MinDev($\%$), respectivamente.

Além disso a linha #Best representa o número de instâncias nas quais o valor da melhor solução encontrada pelo método é igual ao BestValue. Além disso, para cada instância, o Score de cada método é considerado como o número de métodos que obtiveram uma melhor solução que ele. Se há empates, os métodos recebem um valor de Score igual ao número de métodos estritamente melhores que ele. A soma desta pontuação sobre todas as instâncias para cada método é então reportada como Score na Tabela 1. Uma pontuação mais baixa implica num algoritmo melhor. Mais detalhes sobre estas metodologias de comparação podem ser encontradas em Ribeiro et al. (2002). Apresentamos na Tabela 1 os resultados para duas variações do GRASP, as quais chamamos de GVND, tendo somente o método de construção e as buscas locais, e GRC, onde a reconexão por caminhos é também utilizada. As duas versões do GRASP são comparadas com dois algoritmos genéticos apresentados na literatura por Topcuoglu et al. (2005) (GA1) e Cunha e Silva (2007) (GA2). O critério de parada foi escolhido para ser o tempo de processamento, o qual utilizamos $0.2 \times |n|$ segundos para cada método.

Tabela 1: Comparação dos métodos GRASP com GA1 e GA2

Method	GVND	GRC	GA1	GA2
Score	10	2	100	161
#Best	51	55	9	2
Dev (%)	0.049	0.020	6.052	43.069
MinDev(%)	0.022	0.005	4.318	34.637

Observando a Tabela 1 vemos que as duas versões do GRASP se mostram com um desempenho superior aos dois algoritmos genéticos utilizados na comparação. Além disso vemos que entre as versões do GRASP a versão GRC é a que encontra os melhores resultados.

No segundo experimento cada instância é resolvida 10 vezes para cada versão do GRASP com um tempo limite de $3 \times |n|$ segundos como critério de parada. A Tabela 2 mostra os resultados obtidos resumidos como o mínimo (MinDev) e médio (Dev) desvio percentual da solução ótima, e o tempo médio para alcalçá-lo (T [S]) para as duas versões do GRASP. As soluções ótimas foram obtidas por meio da decomposição de Benders apresentada em de Castro (2010). Além disso, o tempo em segundos requerido pela decomposição de Benders apresentada em de Castro (2010) para resolver cada caso é reportado na coluna BD.

O método GVND não conseguiu encontrar a solução ótima em 6 das 56 instâncias, mais especificamente: AP90-6, AP100-6, AP150-6, AP150-8, AP170-6 e AP200-6. Observe que todos, exceto um dos casos, tem o fator de desconto igual a 0.6. Para a versão GRC, a solução ótima foi atingida pelo menos uma vez para todas as 56 instâncias. Observe que os registros na coluna MinDev são todos zero para este método. Além disso, para 3 dos 56 casos (AP150-8, AP170-6, e AP200-6), ele falhou pelo menos uma vez para encontrar a solução ótima dentro do tempo limite. Mais uma vez, todos exceto um desses casos tem fator de desconto igual a 0.6.

Numa terceira bateria de testes os mesmos algoritmos foram executados 200 vezes para

Tabela 2: Variações do GRASP para encontrar soluções alvo nas instâncias AP

Métodos	GVND			GRC			BD
	MinDev	Dev	T	MinDev	Dev	T	T
Instance	(%)	(%)	[s]	(%)	(%)	[s]	[s]
AP10-2	0.000	0.000	0.0016	0.000	0.000	0.0001	0.1
AP10-4	0.000	0.000	0.0032	0.000	0.000	0.0001	0.07
AP10-6	0.000	0.000	0.0001	0.000	0.000	0.0001	0.03
AP10-8	0.000	0.000	0.0001	0.000	0.000	0.0016	0.02
AP20-2	0.000	0.000	0.0015	0.000	0.000	0.0016	1.92
AP20-4	0.000	0.000	0.045	0.000	0.000	0.0016	0.9
AP20-6	0.000	0.000	0.003	0.000	0.000	0.0015	0.31
AP20-8	0.000	0.000	0.0001	0.000	0.000	0.0048	0.18
AP30-2	0.000	0.000	0.0155	0.000	0.000	0.0156	5.37
AP30-4	0.000	0.000	1.5398	0.000	0.000	0.7347	12.32
AP30-6	0.000	0.000	0.003	0.000	0.000	0.0052	6.31
AP30-8	0.000	0.000	0.0156	0.000	0.000	0.0171	7.75
AP40-2	0.000	0.000	0.0232	0.000	0.000	0.0187	18.62
AP40-4	0.000	0.000	0.0218	0.000	0.000	0.0265	26.17
AP40-6	0.000	0.000	0.0124	0.000	0.000	0.0123	42.66
AP40-8	0.000	0.000	0.0156	0.000	0.000	0.0172	82.77
AP50-2	0.000	0.000	0.0421	0.000	0.000	0.0421	40.31
AP50-4	0.000	0.000	0.6132	0.000	0.000	0.9313	35.85
AP50-6	0.000	0.000	5.2977	0.000	0.000	3.1044	162.5
AP50-8	0.000	0.000	0.0216	0.000	0.000	0.022	177.24
AP60-2	0.000	0.000	0.117	0.000	0.000	0.1204	60.14
AP60-4	0.000	0.000	0.0934	0.000	0.000	0.0842	71.65
AP60-6	0.000	0.000	2.7582	0.000	0.000	2.3197	95.05
AP60-8	0.000	0.000	0.1575	0.000	0.000	0.1545	133.27
AP70-2	0.000	0.000	0.2917	0.000	0.000	0.2576	96.77
AP70-4	0.000	0.000	3.4538	0.000	0.000	1.1715	138.52
AP70-6	0.000	0.000	37.9442	0.000	0.000	5.184	261.23
AP70-8	0.000	0.000	12.2228	0.000	0.000	7.6893	1526.35
AP80-2	0.000	0.000	0.0811	0.000	0.000	0.0856	182.28
AP80-4	0.000	0.000	0.114	0.000	0.000	0.1203	268.36
AP80-6	0.000	0.000	8.2355	0.000	0.000	5.9141	320.44
AP80-8	0.000	0.000	49.5427	0.000	0.000	2.2058	935.51
AP90-2	0.000	0.000	0.3589	0.000	0.000	0.3638	532.04
AP90-4	0.000	0.000	10.2306	0.000	0.000	1.4134	542
AP90-6	0.283	0.283	270	0.000	0.000	18.3707	838.59
AP90-8	0.000	0.000	8.3632	0.000	0.000	6.0246	2474.3
AP100-2	0.000	0.000	3.273	0.000	0.000	2.8955	973.61
AP100-4	0.000	0.000	0.9672	0.000	0.000	0.6708	1532.42
AP100-6	0.071	0.071	300	0.000	0.000	24.8961	1992.98
AP100-8	0.000	0.000	17.8232	0.000	0.000	12.5439	5967.89
AP130-2	0.000	0.000	1.2665	0.000	0.000	1.2793	2026.9
AP130-4	0.000	0.001	170.5245	0.000	0.000	57.5673	4575.08
AP130-6	0.000	0.019	176.4653	0.000	0.000	198.6814	5664.12
AP130-8	0.000	0.011	146.1934	0.000	0.000	80.367	10841.97
AP150-2	0.000	0.000	4.3181	0.000	0.000	3.8719	5393.25
AP150-4	0.000	0.020	344.3395	0.000	0.000	30.3064	7639.01
AP150-6	0.156	0.191	450	0.000	0.000	271.7365	9430.92
AP150-8	0.509	0.524	450	0.000	0.026	352.7089	18870.88
AP170-2	0.000	0.000	1.58	0.000	0.000	1.5738	9007.6
AP170-4	0.000	0.000	10.19	0.000	0.000	10.5768	18020.87
AP170-6	0.498	0.499	510	0.000	0.105	394.8635	18831.92
AP170-8	0.000	0.000	1.19	0.000	0.000	1.1888	18184.37
AP200-2	0.000	0.000	2.73	0.000	0.000	2.7707	18155.09
AP200-4	0.000	0.000	2.58	0.000	0.000	2.5819	20345.74
AP200-6	0.146	0.306	600	0.000	0.026	492.0335	18060.72
AP200-8	0.000	0.000	274.88	0.000	0.000	292.6375	19144.56

três instâncias selecionadas (AP50-6, AP70-8, AP90-4). Os algoritmos foram programados para terminarem sua execução quando encontrassem a solução ótima de cada instância. Nas Figuras 5 a 7, cada ponto da curva associada a cada algoritmo representa um par (t_i, p_i), para cada execução de $i = 1, \ldots, 200$, onde t_i é o tempo da i-ésima execução mais rápida deste algoritmo e $p_i = (i - 0,5)/200$. De acordo com a metodologia descrita por Resende e Ribeiro (2003), cada um destes gráficos descreve uma aproximação da distribuição de probabilidade da variável aleatória tempo para alcançar uma solução tão boa quanto o valor alvo.

As Figuras 5 a 7 mostram os gráficos gerados para os três casos. Como pode ser visto, para a instância AP50-6, o GRC tem uma probabilidade mais alta de encontrar a solução ótima em um tempo menor que o GVND. O mesmo se repete para as instâncias AP70-8 e AP90-4 como podemos observar nas Figuras 6 e 7 respectivamente.

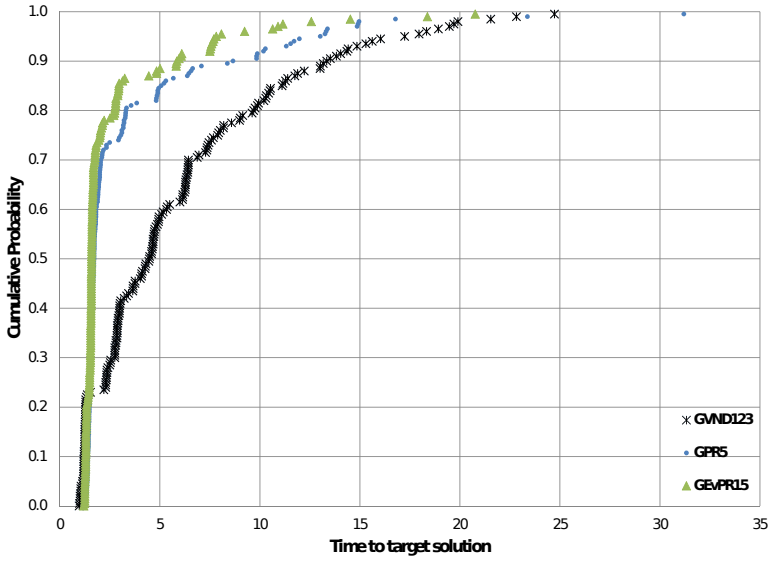

Figura 5: Gráfico para AP50-6.

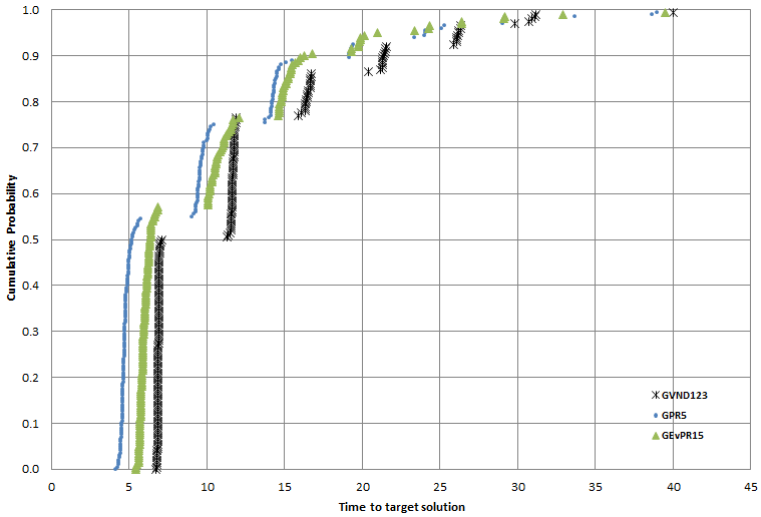

Figura 6: Gráfico para AP70-8.

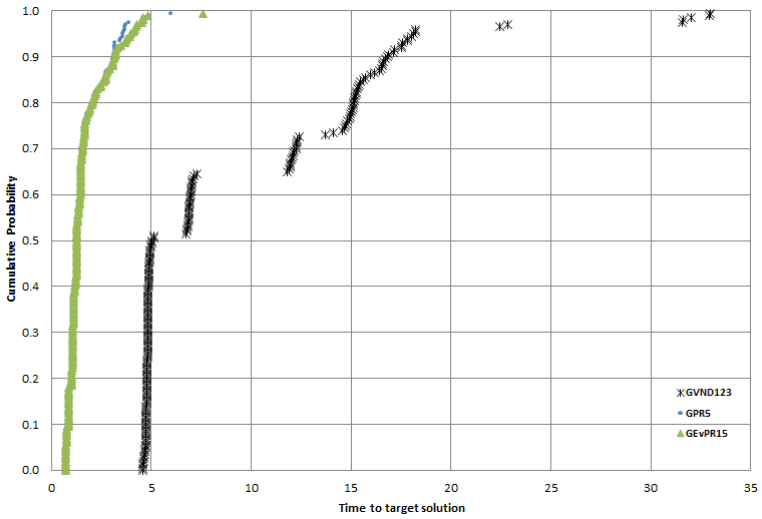

Figura 7: Gráfico para AP90-4

Finalmente, no último conjunto de experimentos, o GRC é comparado com a Busca Tabu (HUBTS) de Silva e Cunha (2009). Como o algoritmo HUBTS não tem um fator de aleatoriedade, a comparação é feita obtendo a melhor solução e o tempo necessário para encontrá-la através do HUBTS para cada instância. Então o método GRC é executado 50 vezes para cada instância, mas tendo a melhor solução obtida pelo HUBTS como alvo. A porcentagem de vezes que o GRC obtém uma solução melhor ou igual à solução do HUBTS num menor tempo é registrado. Os resultados são apresentados na Tabelas 3.

O método GRC é capaz de encontrar um valor melhor ou igual num tempo mais curto que o do algoritmo HUBTS na maioria das 50 execuções de cada instância. Em 4 das 56 instâncias (AP10-6, AP10-8, AP20-8, AP30-8) o GRC não realiza isto em 100% das 50 execuções, sendo o pior caso o exemplo AP30-4, onde falha em atingir o alvo num tempo menor em 14% das vezes. Isto somente acontece em instâncias de tamanho pequeno.

5. Conclusões

O objetivo deste estudo foi propor um GRASP competitivo em relação ao tempo computacional e à qualidade da solução, quando comparados com heurísticas conhecidas na literatura para o projeto de redes Eixo-Raio. Este é um problema importante com muitas aplicações em sistemas de transporte de carga, passageiros e telecomunicação.

A fase de construção do GRASP implementado foi combinada com três variações de estruturas de vizinhança, além disso, combinamos uma estratégia baseada em reconexão por caminhos para intensificar a busca.

Apresentamos duas versões para o método GRASP, uma somente com as buscas locais (GVND) e outra usando também a reconexão por caminhos (GRC). Estas variações claramente superam dois conhecidos algoritmos genéticos da literatura. Além disso, verificamos que a versão com reconexão por caminhos é capaz de encontrar a solução ótima para todas as instâncias testadas ao contrário da versão sem a reconexão. Os resultados também indicam que o GRC supera a Busca Tabu da literatura para os testes realizados. Como trabalhos futuros, podemos investigar a adaptação da fase de construção para outras metaheurísticas, como Algoritimos Genéticos, ou ampliar para outros problemas correlatos.

Tabela 3: GRC vs. HUBTS

Método	HUBTS		GRC
Instância	Target	T [s]	%
AP10-2	90963539.48	0.078	100
AP10-4	95079629.91	0.031	100
AP10-6	95161467.58	0.016	98
AP10-8	95161467.58	0.016	96
AP20-2	96172697.41	0.765	100
AP20-4	105291620.05	0.062	100
AP20-6	108418231.69	0.062	100
AP20-8	108486025.61	0.015	86
AP30-2	83576754.71	6.349	100
AP30-4	91209293.10	1.638	76
AP30-6	95053569.42	0.125	100
AP30-8	98520500.58	0.078	100
AP40-2	80537210.15	18.034	100
AP40-4	88042046.09	9.158	100
AP40-6	94523086.45	1.841	100
AP40-8	99180263.10	1.638	100
AP50-2	71261044.61	58.719	100
AP50-4	80325464.90	57.143	100
AP50-6	89389885.19	80.761	100
AP50-8	95205946.96	1.653	100
AP60-2	64790967.04	153.255	100
AP60-4	73074656.58	126.174	100
AP60-6	80673823.66	80.98	100
AP60-8	87492278.89	3.448	100
AP70-2	74451085.60	90.106	100
AP70-4	83272519.63	99.903	100
AP70-6	91741977.20	108.826	100
AP70-8	97177453.52	8.861	100
AP80-2	70713485.99	251.536	100
AP80-4	79704787.95	176.765	100
AP80-6	88418089.75	150.214	100
AP80-8	96119362.03	11.669	100
AP90-2	69223173.92	944.383	100
AP90-4	78931357.81	872.623	100
AP90-6	87012179.95	158.684	100
AP90-8	92780846.25	51.824	100
AP100-2	67584119.76	947.406	100
AP100-4	77545112.32	770.375	100
AP100-6	86436601.66	465.891	100
AP100-8	93193283.43	58	100
AP130-2	60901562.49	6650.61	100
AP130-4	71560028.43	13967.094	100
AP130-6	81407896.96	7658.953	100
AP130-8	90376670.44	1312.375	100
AP150-2	57397001.45	23407.609	100
AP150-4	68104243.16	44837.75	100
AP150-6	78077582.66	19317.672	100
AP150-8	86577496.58	7484.828	100
AP170-2	63888877.91	7122.375	100
AP170-4	73409009.56	12595.766	100
AP170-6	82043332.23	7675.843	100
AP170-8	88840659.23	606.125	100
AP200-2	62774425.09	29476.688	100
AP200-4	72405312.22	28364.796	100
AP200-6	81580789.69	24549	100
AP200-8	89530676.39	2922.484	100

Agradecimentos

Os autores agradecem ao CNPq e à FAPEMIG pelo apoio ao desenvolvimento deste trabalho.

Referências

Abdinnour-Helm, S. (1998). A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 2–3(106):489–499.

Abdinnour-Helm, S. e Venkataramanan, M. (1998). Solution approaches to hub location problems. *Annals of Operations Research*, 78:31–50.

Alumur, S. e Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, 190:1–21.

Aykin, T. (1995). Network policies for hub-and-spoke systems with applications to the air transportation system. *Transportation Science*, 29:201–221.

Camargo, R. S.; Miranda Jr, G. e Luna, H. P. (2008). Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers and Operations Research*, 35:1047–1064.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72:387–405.

Campbell, J. F.; Ernst, A. T. e Krishnamoorthy, M. (2002). Hub location problems. In Drezner, Z. e Hamacher, H. W., Editores, *Facility Location: Applications and Theory*, capítulo 12, pp. 373–407. Springer, 1^a Edição.

Chen, J. (2007). A hybrid heuristic for the uncapacitated single allocation hub location problem. *Omega*, 35:211–220.

Cunha, C. e Silva, M. (2007). A genetic algorithm for the problem of configuring a hub-and-spoke network for a LTL trucking company in Brazil. *European Journal of Operational Research*, 179:747–758.

de Castro, R. R. M. (2010). Sistemas eixo-raio de alocação simples: Modelos e algoritmos. Master's thesis, Department of Industrial Engineering, Federal University of Minas Gerais.

Ernst, A. T. e Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science*, 4:139–154.

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

Filipovic, V.; Kratica, J.; Tosic, D. e Dugosija, D. (2009). Ga inspired heuristic for uncapacitated single allocation hub location problem. *Applications of Soft Computing, Advances in Intelligent and Soft Computing*, 58:149–158.

Klincewicz, J. G. (1998). Hub location in backbone/tributary network design: a review. *Location Science*, 6:307–335.

Labbé, M. e Yaman, H. (2004). Projecting flow variables for hub location problems. *Networks*, 44(2):84–93.

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

Mladenovic, N. e Hansen, P. (1997). A variable neighborhood search. *Computers and Operations Research*, 24:1097–1100.

O'Kelly, M. E. (1987). A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32:393–404.

O'Kelly, M. E. e Bryan, D. L. (1998). Hub location with flow economies of scale. *Transportation Research Part B*, 32(8):605–616.

Resende, M. G. C. e Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In Glover, F. e Kochenberger, G., Editores, *Handbook of metaheuristics*, pp. 219–250. Kluwer Academic Publishers.

Ribeiro, C.; Uchoa, E. e Werneck, R. F. (2002). A hybrid grasp with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–46.

Silva, M. R. e Cunha, C. B. (2009). New simple and efficient heuristics for the uncapacitated single allocation hub location problem. *Computers and Operations Research*, 36(12):3152–3165.

Topcuoglu, H.; Corut, F.; Ermis, M. e Yilmaz, G. (2005). Solving the uncapacitated hub location problem using genetic algorithms. *Computers and Operations Research*, 32(4):967–984.