Um Método de Pós-Otimização para o Problema de Síntese de Redes a 2-caminhos

Glaubos Clímaco, Isabel Rosseti, Luidi Gelabert, Marcos Guerine
Instituto de Computação - Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 - Bloco E - CEP 24210-240 - Niterói/RJ - Brasil
{fglaubos,rosseti,luidi,mguerine}@ic.uff.br

RESUMO

Neste trabalho é proposto um processo de pós-otimização, que combina o estado-da-arte de uma heurística com mineração de dados existente com Programação Linear Inteira (PLI), para solucionar o *2-Path Network Design Problem* (2-PNDP). A ideia principal é obter informações úteis a partir da heurística original e inseri-las no modelo de PLI, a fim de melhorar a qualidade das soluções em um tempo computacional razoável. No intuito de validar esta proposta, uma comparação entre ambas as estratégias foi realizada e os resultados computacionais mostraram que nosso método de pós-otimização gerou melhores resultados que os obtidos com a estratégia estado-da-arte da literatura.

PALAVRAS CHAVE. 2-PNDP, PLI, heurística, Área de principal: Otimização Combinatória.

ABSTRACT

In this work is proposed a post optimization process, that combines a state-of-the-art of an existing data mining heuristic with Integer Linear Programming (ILP), to solve the 2-path network design problem (2-PNDP). The main idea of our approach is to obtain useful information from the original heuristic and insert it in the ILP model, such that it improves the solution quality in a reasonable computing time. Aiming to validate this proposal, a comparison between the strategies was made and computational results showed that our post optimization method generates better results than the obtained with the state-of-art strategy of the literature.

KEYWORDS. 2-PNDP, PLI, heuristic, Main area: Combinatorial optimization.

1. Introdução

Em otimização combinatória, soluções ótimas normalmente são alcançadas por métodos exatos, tais como programação linear (PL), *branch-and-bound*, *branch-and-cut* e *branch-and-price* (Wolsey e Nemhauser, 1988). Entretanto, esses métodos geralmente tornam-se impraticáveis quando aplicados a problemas de grande porte. Dessa forma, metaheurísticas representam uma classe importante de técnicas na obtenção de soluções em tempos computacionais razoáveis, para problemas difíceis. Destacamos alguns exemplos de metaheurísticas tais como, GRASP (*greedy randomized adaptive search procedure*) (Feo e Resende, 1995), Algoritmos Genéticos (Kelly, 1996), Busca Tabu (Glover *et al*., 2000), *Scatter Search* (Glover *et al*., 2000), que têm sido aplicados com sucesso a problemas de otimização (Boussaïd *et al*., 2013).

Neste trabalho, propomos uma estratégia que combina a heurística hibridizada com mineração de dados MDM-GRASP-RC (Barbalho *et al*., 2013) com programação linear

inteira (PLI), para solucionar o *2-Path Network Design Problem* (2-PNDP). Seja $G(V, E)$ um grafo conectado não direcionado, tal que V representa o conjunto de vértices e E o conjunto de arestas, cada uma com custos não-negativos. Dado um conjunto de demandas D contendo pares de vértices origem-destino, o 2-PNDP consiste em encontrar um subconjunto $E' \subseteq E$ de custo mínimo contendo um 2-caminho (um caminho com no máximo 2 arestas) entre cada par origem-destino pertencente a D. Algumas aplicações do 2-PNDP podem ser vistas em projetos de síntese de redes, no qual o projetista necessita assegurar confiabilidade dos dados e poucos atrasos. Dahl e Johannessen (Dahl e Johannessen, 2004) provaram que a versão de decisão do 2-PNDP é um problema NP-completo.

Uma formulação para o 2-PNDP, definida como modelo Caminho, foi proposta em (Dahl e Johannessen, 2004). Seja P_d o conjunto de todos os 2-caminhos para cada par origem-destino $d \in D$. Uma variável binária z_P é definida como $z_P = 1$ se um caminho P é escolhido a partir de um conjunto P_d, ou $z_P = 0$ caso contrário. Nesse sentido, o 2-PNDP pode ser formulado como segue:

$$min \sum_{e \in E} c_e x_e \tag{1}$$

Sujeito a:

$$\sum_{p \in P_d} z_P = 1, \forall d \in D \tag{2}$$

$$z_P \leq x_e, \forall d \in D, P \in P_d, e \in P \tag{3}$$

$$z_P \geq 0, \forall d \in D, P \in P_d \tag{4}$$

$$x_e, z_P \in \{0, 1\}, \forall d \in D, P \in P_d, e \in P \tag{5}$$

O número de restrições (2) é no máximo p (número total de 2-caminhos), e define que um 2-caminho é selecionado para cada par origem-destino $d \in D$. O problema representado pelo modelo Caminho possui k nós, e o número de restrições (3) é no máximo $p(2k - 3)$, pois com exceção do 2-caminho que une de maneira direta um par origem-destino, todos os outros 2-caminhos são formados por duas arestas. O número de restrições (4) é no máximo $p(k - 1)$, porque para cada par origem-destino, existem $(k - 1)$ 2-caminhos. Portanto, em um grafo completo esse modelo possui $O(k^2)$ restrições, o que nos levou a escolhê-lo como modelo de PLI para nossa proposta, uma vez que as restrições da outra formulação apresentada por Dahl e Johannessen (Dahl e Johannessen, 2004), crescem exponencialmente com k.

O restante deste artigo é organizado da seguinte maneira. Na Seção 2, são apresentadas estratégias anteriores para a solução do 2-PNDP, inclusive o estado-da-arte MDM-GRASP-RC abordado em (Barbalho *et al.*, 2013). Em seguida, a Seção 3 descreve como

foi desenvolvido nosso método de pós-otimização MPO. A Seção 4 apresenta uma análise de seu desempenho e uma comparação com o MDM-GRASP-RC. Finalmente, na Seção 5 é apresentada a conclusão deste trabalho e são propostos trabalhos futuros.

2. Revisão bibliográfica

A heurística estado-da-arte para solução do 2-PNDP, apresentada em (Barbalho *et al*., 2013), é a incorporação de um processo de mineração de dados à heurística do GRASP com reconexão por caminhos (RC) abordada em (Rosseti, 2003). Ambos métodos obtiveram sucesso utilizando o a metaheurística GRASP como base para suas estratégias, na qual tem sido aplicada com êxito na solução de vários outros problemas de otimização, em diversas áreas como problemas de escalonamento, roteamento, particionamento, localização e atribuição (Festa e Resende, 2009).

GRASP é um procedimento multipartida, em que cada iteração consiste em duas fases: construção e busca local. Basicamente, uma solução viável é gerada na fase de construção, e então sua vizinhança é explorada pela busca local com o objetivo de melhorar a solução obtida pela fase construtiva. Esse processo iterativo é repetido até que o critério de parada seja atingido, e então a melhor solução encontrada ao longo de todas as iterações é definida como o resultado final do processo.

Na fase de construção, componentes da solução são selecionadas individualmente e agrupadas a uma solução parcial, até que essa seja completamente construída. Componentes que ainda não estão na solução são classificadas de acordo com uma função gulosa, e as melhores classificadas formam uma Lista Restrita de Candidatos (LRC). Em seguida, uma componente é escolhida aleatoriamente a partir desta lista e inserida na solução atual.

Após passar por uma fase de construção, não há garantia que a solução construída seja localmente ótima (Feo e Resende, 1995). Nesse sentido, uma busca local é aplicada. Busca local é um processo que explora a vizinhança de uma solução, buscando por soluções de melhor qualidade. A vizinhança de uma solução é definida por uma função que relaciona a similaridade entre essa solução e as demais soluções presentes no espaço de busca. Durante esse processo, se uma solução melhor x' é alcançada, uma nova busca local é realizada a partir de x', caso contrário, a busca local termina.

A metaheurística GRASP é uma estratégia sem memória, pois suas iterações são independentes e nenhuma informação é passada para iterações futuras. O objetivo de inserir reconexão por caminhos a um algoritmo GRASP puro é armazenar soluções boas encontradas anteriormente, e utilizá-las como guias na busca por melhorias.

Reconexão por caminhos é uma técnica proposta por Glover (Glover *et al*., 1977) para explorar possíveis trajetórias conectando soluções de alta qualidade, obtidas por heurísticas como Busca Tabu e *Scatter Search* (Glover *et al*., 2000). Seu primeiro uso integrado ao GRASP foi realizado por Laguna e Martí (Laguna e Martí, 1999), e atualmente, várias extensões, melhorias e aplicações de sucesso dessa técnica podem ser encontradas na literatura (Chaovalitwongse *et al*., 2011) (Resende e Ribeiro, 2005).

De acordo com Rosseti (Rosseti, 2003), essa técnica é aplicada a um par de soluções $\{s_i, s_g\}$, no qual s_g é uma solução guia localmente ótima obtida após uma busca local, e s_i é uma solução selecionada aleatoriamente de um conjunto elite do GRASP, L, de tamanho $MaxElite$. Inicialmente, L está vazio. Toda solução obtida pela busca local é candidata a pertencer a L, desde que essa seja diferente de todas as outras soluções desse

conjunto. Se $MaxElite$ é atingido e a candidata é melhor que a pior solução de L, a primeira substitui a segunda. Se o conjunto ainda não estiver completo, a candidata é inserida automaticamente.

O RC inicia calculando a diferença simétrica $\triangle(s_i, s_g)$ entre s_i e s_g, cujo resultado é o conjunto de movimentos que s_i deve fazer para alcançar s_g. Partindo de s_i, realiza-se o melhor movimento ainda não executado até o momento em que s_g é atingida. Após esses movimentos, a melhor solução encontrada durante a trajetória é considerada candidata a pertencer ao conjunto elite, e a melhor solução global é atualizada.

2.1. GRASP com reconexão por caminhos para o 2-PNDP

No intuito de promover uma colaboração entre GRASP e RC, ambos métodos foram adaptados para o 2-PNDP. No GRASP com RC, a fase construtiva gulosa adaptada computa um 2-caminho por vez. Seja w_e o peso original de cada aresta. Inicialmente, w'_e recebe os pesos originais w_e e a solução s é inicializada como vazia. A cada iteração dessa fase de construção, um par origem-destino $(a, b) \in D$ é aleatoriamente selecionado e o menor caminho P entre (a, b) é calculado pesos w'_e. Em seguida, para cada aresta $(i, j) \in P$, seus pesos w'_{ij} são definidos com o valor zero. Esse par origem-destino (a, b) é removido do conjunto D e P é anexado a solução atual s. A construção termina quando D se torna um conjunto vazio.

Como mencionado anteriormente, a fase de construção do GRASP costuma não encontrar uma solução localmente ótima, recomendando-se a execução de uma busca local. Nesse sentido, uma busca local adaptada é realizada. Cada solução s pode ser vista com uma coleção de $|D|$ 2-caminhos. Dada qualquer solução s, sua solução vizinha s' pode ser obtida pela substituição de qualquer 2-caminho em s por outro 2-caminho entre o mesmo par origem-destino. A busca local tenta aprimorar as soluções obtidas de maneira gulosa na fase de construção adaptada, e é interrompida quando $|D|$ iterações são executadas sem melhorias.

Para o 2-PNDP, a técnica de reconexão por caminhos é aplicada na solução obtida pela busca local (s), e para uma solução selecionada aleatoriamente de L (y). Esse procedimento é realizado duas vezes, uma utilizando s como solução inicial e y como guia (*Forward Relinking*), e outra definindo esta última como solução inicial e a primeira como guia (*Backward Relinking*) (Resende e Ribeiro, 2005).

2.2. Mineração de dados aplicada ao GRASP com RC

Nesta seção são apresentados os métodos DM-GRASP-RC (*Data Mining* GRASP com reconexão por caminhos) e seu aprimoramento MDM-GRASP-RC (*Multi-Data Mining* GRASP com reconexão por caminhos), ambos desenvolvidos por Barbalho *et al* (Barbalho *et al*., 2013). Tais métodos são processos de mineração de dados integrados ao GRASP com o RC, desenvolvido por Rosseti (Rosseti, 2003).

No GRASP original as iterações são realizadas independentemente, e como consequência disso, o conhecimento adquirido em iterações anteriores não é utilizado nas subsequentes. A ideia básica de incorporação da mineração de dados é encontrar padrões em soluções de alta qualidade que conduzam e aprimorem o processo de busca.

O DM-GRASP-RC é composto de duas fases. A primeira é chamada de fase de geração do conjunto elite, que consiste em n execuções do GRASP com RC para obter

um conjunto de diferentes soluções. As d melhores soluções desse conjunto formam o conjunto elite.

Após essa primeira fase, a mineração de dados é aplicada ao conjunto elite com o intuito de extrair padrões, ou seja, conjuntos de elementos que apareçam frequentemente nas soluções do conjunto elite. É importante ressaltar que um conjunto de itens frequentes com suporte $s\%$ representa um conjunto de elementos que ocorrem em $s\%$ das soluções elite.

Posteriormente, a segunda fase do DM-GRASP-RC é realizada executando novamente *n* iterações, entretanto, com algumas diferenças. Nessa nova fase, um padrão é selecionado a partir de um conjunto de padrões minerados, e uma fase de construção adaptada gerará uma solução completa a partir desse padrão, ou seja, todas as soluções obtidas por essa construção conterão os elementos do padrão selecionado.

Na proposta DM-GRASP-RC, o procedimento de mineração de dados é executado apenas uma vez, no meio do processo inteiro. Apesar dos resultados satisfatórios, Barbalho et al. (Barbalho *et al*., 2013) acreditaram na hipótese de melhoria do método executando a mineração de dados mais de uma vez, assim que o conjunto elite se tornar estável e sempre que esse conjunto for alterado e novamente se tornar estável.

A partir da hipótese acima, foi desenvolvido o método MDM-GRASP-RC (*multi data-mining* GRASP-RC)(Barbalho *et al*., 2013). A principal ideia dessa proposta, é executar mineração de dados sempre que o conjunto elite se tornar estável, ou seja, não ocorrer mudança nesse conjunto dado um número de iterações, e também sempre que este conjunto for alterado e novamente ficar estável. O resultado da implementação dessa hipótese foi uma exploração mais gradual do conjunto elite, permitindo a extração de padrões mais promissores.

O pseudocódigo do MDM-GRASP-RC é apresentado no Algoritmo 1. O bloco de instruções das linhas 5 a 23, corresponde à primeira fase de geração do conjunto elite, no qual iterações do GRASP com RC são executadas até o momento que M esteja pronto para ser minerado, ou o número máximo de iterações for alcançado. A seguir, das linhas 24 a 44, sempre que M estiver pronto, o processo de mineração de dados é executado na linha 26. Na linha 28, o próximo maior padrão é selecionado e então, na linha 29, a construção adaptada é executada. Em seguida, uma busca local é realizada na linha 30, e das linhas 32 a 35, duas chamadas do RC são executadas. Se uma solução com menor custo é encontrada, a melhor solução é atualizada na linha 40. Após todas as iterações, a melhor solução é retornada na linha 45.

3. Processo de pós-otimização proposto

Um processo de pós-otimização MPO é proposto para coletar informações úteis da heurística híbrida de mineração de dados MDM-GRASP-RC (Barbalho *et al*., 2013) e combiná-las com o modelo Caminho apresentado na Seção 1.

Antes do desenvolvimento desta proposta, optamos por resolver as instâncias contidas em (Barbalho *et al*., 2013) utilizando apenas a PLI pura e o modelo original (Dahl e Johannessen, 2004). Contudo, não foi possível resolvê-las em tempo computacional aceitável, pois apenas para resolver sua relaxação linear, foram utilizadas mais de três horas na tentativa de resolver uma instância de menor tamanho, contendo 100 vértices e 1000 demandas.

Algoritmo 1: **MDM-GRASP-RC**($maxIteracoes, d, t$)

$L \leftarrow \emptyset$;
$f^* \leftarrow \infty$;
$M \leftarrow \emptyset$;
$k \leftarrow 0$;
Enquanto $M\_nao\_esta\_pronto$ **e** $k < masIteracoes$ **faça**
  $s \leftarrow$ Construcao2-Caminho();
  $s \leftarrow$ BuscaLocal2-Caminho(s);
  Atualizar o conjunto das soluções elite L com s;
  Se $|L| \geq 2$ **então**
    Seleciona aleatoriamente uma solução elite y de L;
    $s_1 \leftarrow$ RC (s, y);
    Atualizar o conjunto das soluções elite L com s_1;
    $s_2 \leftarrow$ RC (y, s);
    Atualizar o conjunto das soluções elite L com s_2;
    $s \leftarrow$ argmin $\{f(s), f(s_1), f(s_2)\}$;
  Fim-se
  atualizarElite(M, s, d);
  Se $f(s) < f^*$ **então**
    $s^* \leftarrow s$;
    $f^* \leftarrow f(s)$;
  Fim-se
  $k \leftarrow k + 1$;
Fim-enquanto
Enquanto $k < masIteracoes$ **faça**
  Se M_esta_pronto **então**
    $conjunto\_padroes \leftarrow$ minerar (M, t);
  Fim-se
  $padrao \leftarrow$ selecionarProsimoMaiorPadrao($conjunto\_padroes$) ;
  $s \leftarrow$ Construcao2-CaminhoAdaptado($padrao$);
  $s \leftarrow$ BuscaLocal2-Caminho(s);
  Atualizar o conjunto das soluções elite L com s;
  Seleciona aleatoriamente uma solução elite y de L;
  $s_1 \leftarrow$ RC (s, y);
  Atualizar o conjunto das soluções elite L com s_1;
  $s_2 \leftarrow$ RC (y, s);
  Atualizar o conjunto das soluções elite L com s_2;
  $s \leftarrow$ argmin $\{f(s), f(s_1), f(s_2)\}$;
  atualizarElite(M, s, d);
  Se $f(s) < f^*$ **então**
    $s^* \leftarrow s$;
    $f^* \leftarrow f(s)$;
  Fim-se
  $k \leftarrow k + 1$;
Fim-enquanto
Retorne s^*;

Visto a ineficiência de se utilizar PLI pura para resolver essas instâncias, realizou-se uma análise detalhada acerca do comportamento das componentes durante a execução de ambas estratégias originais, GRASP-RC e MDM-GRASP-RC. Dessa foma, percebeu-se que durante as execuções da busca local, várias arestas (referidas como E^0) nunca são escolhidas para pertencerem a uma solução ótima local. Para todas as instâncias abordadas em (Barbalho *et al*., 2013), E^0 representa 70% de E, no pior caso. Nesse sentido, removeu-se E^0 do grafo original, reduzindo o espaço de busca para $G(V, E \setminus E^0)$. Consequentemente, o resolvedor tratou um problema de tamanho menor, diminuindo seu tempo de execução.

Para uma ideia mais detalhada sobre o benefício da remoção do conjunto E^0, na Tabela 1 é apresentada uma instância de cada tamanho, mostrando a média das arestas removidas em dez execuções com sementes diferentes, e a porcentagem que elas representam no conjunto E.

Instância	**Arestas removidas**	**Porcentagem de E**
a100-1	3485.9	70.42%
a200-1	15496.7	77.87%
a300-1	36172.9	80.65%
a400-1	66082.2	82.81%
a500-1	104238.4	83.56%

Tabela 1. Arestas removidas do problema original.

Após alguns experimentos, também descobriu-se que soluções obtidas por ambos métodos possuem algumas arestas em comum. Assim, no intuito de evitar um trabalho já realizado pelo MDM-GRASP-RC, selecionou-se, a partir da solução encontrada por essa heurística original, as arestas mais promissoras (referidas como E^*_{final}). Suas respectivas variáveis foram fixadas no modelo de PLI com o valor igual a um, forçando o resolvedor a encontrar apenas soluções que contenham essas arestas.

O pseudocódigo da proposta de pós-otimização é apresentado no Algoritmo 2. É válido ressaltar que antes da chamada do Algoritmo 2, o método MDM-GRASP-RC é executado e ambos os conjuntos de arestas E^* e E^0 são construídos e passados como parâmetro de entrada para o MPO. O conjunto E^* é construído obtendo-se todas as arestas presentes na solução do MDM-GRASP-RC, e E^0 é gerado obtendo-se todas as arestas não utilizadas em nenhuma busca local desse último método. É passado como parâmetro de entrada mais uma variável, $numFix$, na qual define a porcentagem da solução final do MDM-GRASP-RC que pertencerá a solução final da pós-otimização, determinando um limite inferior de similaridade entre as soluções obtidas pelo MDM-GRASP-RC e pelo método de pós-otimização. Na linha 1, é chamada uma função na qual ordenará todas as arestas pertencentes a E^*, de acordo com os critérios $incidencia$ (correspondente ao custo de inserção de uma aresta na solução), e num_vezes (o número de vezes que uma aresta é utilizada no atendimento de diferentes demandas).

Uma vez que as melhores arestas se encontram ordenadas na lista $arestas\_ordenadas$, na linha 2, a variável n é inicializada com o número de arestas que serão utilizadas do conjunto E^*. Nas linhas 4 a 7, as primeiras n arestas são inseridas no conjunto E^*_{final}, correspondendo à porcentagem de num_fix das melhores arestas em

E^*. Na linha 8, ocorre a redução do espaço de soluções do problema utilizando o conjunto E^0. Na linha 9, as variáveis correspondentes às arestas do conjunto E^*_{final} são fixadas com o valor igual a um no modelo Caminho, forçando o resolvedor a buscar soluções que contenham arestas presentes em E^*_{final}. E, por fim, o resolvedor é chamado na linha 10, recebendo como parâmetro de entrada o modelo modificado.

Algoritmo 2: **MPO**($E^*, E^0, numFix$)

1: ordenar_decrescente($E^*, arestas\_ordenadas, num\_vezes, incidencia$);
2: $n \leftarrow |arestas\_ordenadas| \times num\_fix$;
3: $i \leftarrow 0$;
4: **Enquanto** $i < n$ **faça**
5: $\quad E^*_{final} \leftarrow E^*_{final} \cup arestas\_ordenadas(i)$;
6: $\quad i \leftarrow i + 1$;
7: **Fim-enquanto**
8: reduzir($modelo, E^0$);
9: fixar($modelo, E^*_{final}$);
10: solver($modelo$);

4. Experimentos Computacionais

Nesta seção, os resultados computacionais da heurística MDM-GRASP-RC (Barbalho *et al*., 2013) e da nossa proposta de pós-otimização são comparados. As duas estratégias foram testadas a partir das instâncias apresentadas em (Barbalho *et al*., 2013). Essas instâncias são grafos completos com $|V| \in \{100, 200, 300, 400, 500\}$, nas quais os custos das arestas foram gerados aleatoriamente a partir de uma distribuição uniforme no intervalo $[1, 10]$, e $10 \times |V|$ pares origem-destino são escolhidos aleatoriamente.

Ambos métodos foram implementados na linguagem de programação C, utilizando o compilador gcc versão 4.8.3, e todos os testes foram executados em um computador pessoal Intel®Core™ i5 CPU 650 @ 3.20 GHz com 4GB RAM, utilizando o sistema operacional Linux. Para a execução do modelo de PLI, utilizamos o IBM ILOG CPLEX Optimizer (IBM) e a capacidade de paralelização do processador não foi utilizada.

Como mencionado anteriormente na Seção 3, a porcentagem de arestas $numFix$ de E^* escolhida da solução original obtida pela abordagem MDM-GRASP-RC é um parâmetro que indica um mínimo de similaridade entre a solução encontrada pela execução do modelo de PLI e a solução original. Dessa forma, a fim de encontrar a melhor configuração, escolhemos cinco instâncias e, para cada uma, variamos o parâmetro $numFix$ em 70%, 75% e 80%. Para cada instância, dez execuções foram realizadas com limite de tempo de 3600 segundos e os resultados são apresentados na Tabela 2. A primeira coluna corresponde ao identificador de instância $ax - y$, na qual $x = |V|$ e y é a semente usada como parâmetro para gerar uma instância aleatória. A segunda coluna corresponde a melhor solução conhecida (MSC) na literatura (Barbalho *et al*., 2013). As colunas seguintes representam o melhor custo, a média de custo e a média de tempo extra utilizado pelo resolvedor, para cada configuração diferente de $numFix$.

Instância	MSC	70%			75%			80%		
		melhor	média	tempo extra	melhor	média	tempo extra	melhor	média	tempo extra
a100-1	676	664	673.2	1209.6	671	676.9	50.4	675	678.8	0.9
a200-1	1374	1367	1372.1	236.1	1367	1373.1	17.9	1367	1373.3	46.9
a300-1	2082	2065	2081.1	2080.8	2072	2082.1	1547.6	2069	2086.1	81.4
a400-1	2779	2792	2801.6	1713.2	2773	2783.0	1413.5	2774	2782.1	27.1
a500-1	3554	3567	3680.3	3603.4	3541	3555.2	2003.7	3548	3561.7	22.0

Tabela 2. Variação da porcentagem de arestas fixadas.

Na Tabela 2, na medida que o nível de fixação aumenta, o custo da solução piora e o tempo de processamento do resolvedor diminui em relação a níveis menores de fixação. Contudo, fixando 80% das arestas, o MPO já alcança as melhores soluções conhecidas para essas cinco instâncias, requerendo apenas um pequeno tempo adicional de execução em relação às outras configurações. Portanto, escolhemos 80% como nossa configuração padrão de fixação.

Após o experimento para escolher o grau de fixação, testamos nossa abordagem para todas as instâncias. Como apresentado na Seção 3, nosso método de pós-otimização é realizado ao final da heurística MDM-GRASP-RC e, portanto, seu custo computacional já inicia a partir do tempo despendido por esse último método. Dessa forma, para uma comparação justa, a nossa implementação (ou seja, a abordagem original acrescida do método de pós-otimização) foi executada dez vezes para cada instância, com dez sementes distintas. O tempo despendido em cada uma de suas execuções, foi fornecido ao MDM-GRASP-RC, modificando, assim, o critério de parada da versão original (isto é, a abordagem original agora interrompe a sua execução quando é alcançado o tempo utilizado pela nossa heurística proposta).

Os resultados dessa comparação são apresentados na Tabela 3. Cada estratégia foi executada dez vezes para cada instância, com sementes diferentes. A Tabela 3 apresenta o custo da melhor solução obtida, o custo médio referente às dez execuções, o tempo computacional médio utilizado nos dois métodos e os desvios percentuais de ambas estratégias em relação às melhores soluções e às médias. Na comparação entre as duas abordagens, os valores em negrito representam os melhores resultados alcançados.

Observando a Tabela 3, comprovou-se que realmente o MPO possui um desempenho melhor ou igual ao MDM-GRASP-RC para a maioria das instâncias, usando-se a mesma quantidade de tempo de execução em ambas as abordagens.

A redução no custo das soluções obtida pelo MPO deve-se ao fato deste método sempre convergir para um ótimo global do problema modificado em função das informações adquiridas. O mesmo muitas vezes não acontece com o MDM-GRASP-RC, que em determinado momento de sua execução possui essas mesmas informações das execuções já realizadas e apenas atinge um ótimo local.

				MDM-GRASP-RC				MPO			
Instância	Melhor Solução	Melhor Média	Tempo Médio	Melhor Solução	Média Solução	DP Melhor	DP Média	Melhor Solução	Média Solução	DP Melhor	DP Média
a100-1	675	679.2	9.87	676	680.9	0.001	0.002	**675**	**679.2**	0.000	0.000
a100-10	655	665.1	9.37	658	666.8	0.004	0.002	**655**	**665.1**	0.000	0.000
a100-100	661	667.2	10.27	662	669.0	0.001	0.002	**661**	**667.2**	0.000	0.000
a100-1000	641	646.3	9.54	642	648.4	0.001	0.003	**641**	**646.3**	0.000	0.000
a100-10000	658	663.5	9.71	660	665.9	0.003	0.003	**658**	**663.5**	0.000	0.000
a200-1	1367	1373.5	55.12	1374	1379.4	0.005	0.004	**1367**	**1373.5**	0.000	0.000
a200-10	1362	1363.0	55.41	**1362**	1371.0	0.000	0.005	**1362**	**1363.0**	0.000	0.000
a200-100	1339	1348.8	59.99	1344	1352.0	0.003	0.002	**1339**	**1348.8**	0.000	0.000
a200-1000	1349	1356.2	67.81	1354	1362.5	0.003	0.004	**1349**	**1356.2**	0.000	0.000
a200-10000	1357	1365.6	50.89	1360	1370.7	0.002	0.003	**1357**	**1365.6**	0.000	0.000
a300-1	2068	2084.8	164.93	**2068**	2088.9	0.000	0.001	**2068**	**2084.8**	0.000	0.000
a300-10	2113	2119.9	134.77	2119	2124.9	0.002	0.002	**2113**	**2119.9**	0.000	0.000
a300-100	2062	2067.0	124.37	2065	2072.8	0.001	0.002	**2062**	**2067.0**	0.000	0.000
a300-1000	2067	2082.1	276.23	**2067**	**2082.1**	0.000	0.000	2073	2082.9	0.002	0.000
a300-10000	2053	2064.9	133.02	2054	2069.8	0.000	0.002	**2053**	**2064.9**	0.000	0.000
a400-1	2774	2781.8	223.80	2778	2785.3	0.001	0.001	**2774**	**2781.8**	0.000	0.000
a400-10	2826	2838.7	234.13	2832	**2838.7**	0.002	0.000	**2826**	2933.8	0.000	0.030
a400-100	2791	2797.1	232.24	2796	2801.4	0.001	0.001	**2791**	**2797.1**	0.000	0.000
a400-1000	2789	2801.3	241.43	2792	2804.4	0.001	0.001	**2789**	**2801.3**	0.000	0.000
a400-10000	2821	2839.3	243.51	2827	2845.1	0.002	0.002	**2821**	**2839.3**	0.000	0.000
a500-1	3548	3561.6	403.76	3552	3566.2	0.001	0.001	**3548**	**3561.6**	0.000	0.000
a500-10	3553	3562.7	395.94	3.556	3565.7	0.000	0.000	**3553**	**3562.7**	0.000	0.000
a500-100	3556	3569.4	424,58	3564	3572.8	0.000	0.000	**3556**	**3569.4**	0.000	0.000
a500-1000	3527	3546.8	420.90	3531	3551.5	0.001	0.001	**3527**	**3546.8**	0.000	0.000
a500-10000	3550	3580.2	414.83	3556	3584.7	0.001	0.001	**3550**	**3580.2**	0.000	0.000

Tabela 3. Resultados Computacionais comparando MDM-GRASP-RC e MPO utilizando o mesmo tempo de execução

4.1. Significância estatística

A fim de verificar se as diferenças de valores médios, mostrados na Tabela 3, obtidos pelas estratégias avaliadas são estatisticamente significantes, foi realizado o teste de *Friedman* (Conover e Iman, 1981) com *p*-valor igual a 0.03.

Estratégia	Grupo de instâncias				
	a100	a200	a300	a400	a500
MDM-GRASP-RC	0 (0)	0 (0)	1 (0)	1 (1)	0 (0)
MPO	5(5)	5(4)	4(4)	4(4)	5(5)

Tabela 4. Análise de significância estatística

A Tabela 4.1 apresenta, para cada par de estratégia e para cada grupo de instâncias, composto por instâncias com o mesmo número de nós, o número de instâncias com melhores médias encontradas por cada estratégia e, entre parênteses o número de vezes em que *p*-valor foi menor que 0.03, significando que a probabilidade da diferença de desempenho ser devido à aleatoriedade é menor que 3%.

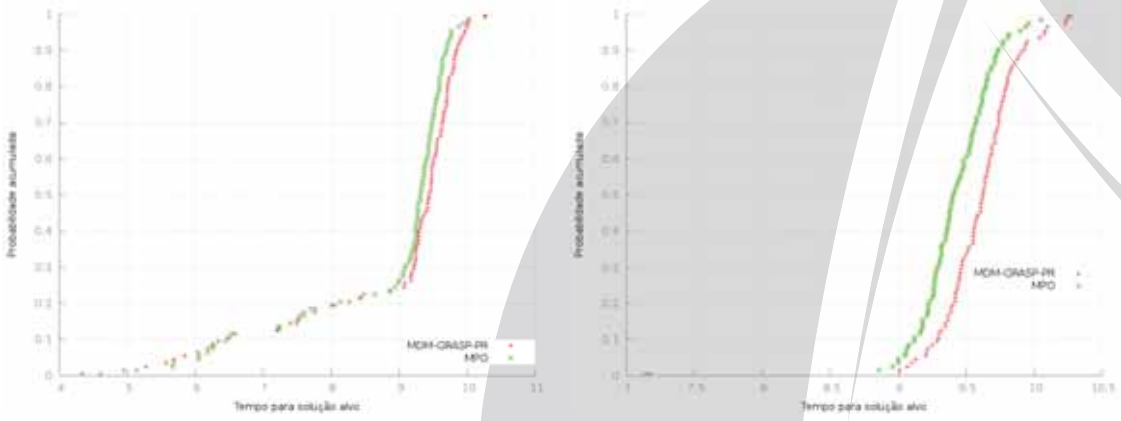
Pode-se perceber que quase todos os resultados de desempenho possuem signifi-

cância estatística, com exceção do grupo a200, no qual um resultado de desempenho melhor para o MPO (em relação à estratégia original) não possui significância estatística. Já para uma instância de 300 nós, apesar da estratégia de mineração original ser melhor que a MPO, esse resultado não possui significância estatística.

4.2. Análise do comportamento das estratégias

Nesta seção, apresentamos uma análise adicional dos experimentos computacionais realizada para ilustrar o comportamento das estratégias. Fizemos uma comparação entre os dois métodos baseada em *Time-to-target plots* (TTT-plots) (Aiex *et al*., 2007), usado para analisar o comportamento de algoritmos aleatórios. Basicamente, esses gráficos mostram a probabilidade acumulada de um algoritmo encontrar uma solução melhor ou igual a uma solução alvo prefixada, em um certo tempo de execução.

Os gráficos mostrados nas Figuras 1(a) e 1(b) foram gerados pela execução do MDM-GRASP-RC e MPO para a execução da instância a100-1, utilizando um alvo com dificuldade intermediária (679) e outro mais difícil (675). Para o alvo intermediário, a definição da melhor estratégia oscila até 9 segundos de execução, e após esse tempo, o MPO possui desempenho melhor até o momento de 10.3 segundos, tempo no qual ambos possuem probabilidade de 100% de atingir o alvo. Para um alvo intermediário, a Figura 1(a) ilustra um leve desempenho melhor do MPO. Contudo, para o alvo difícil, o MPO apresenta um comportamento superior ao MDM-GRASP-RC. Podemos observar na Figura 1(b) que o MPO atinge o alvo em 9.73 segundos com 90% de probabilidade, enquanto que o MDM-GRASP-RC possui probabilidade de apenas 60%.



(a) Alvo intermediário (b) Alvo difícil

Figura 1. *Time-to-target plot* para a instância a100-1

5. Conclusões e trabalhos futuros

Neste trabalho foi apresentada uma abordagem que incorpora um processo de pós-otimização a uma heurística com mineração de dados já conhecida e bem sucedida na resolução do 2-PNDP. Para verificar a relevância desta nova proposta, foram realizados experimentos computacionais as 25 instâncias utilizadas em (Barbalho *et al*., 2013).

Como resultado dos experimentos, notamos a importância dos dados obtidos durante a execução da heurística MDM-GRASP-RC. Por meio da análise desses dados, foi-se

possível reduzir o problema original e utilizar PLI para solucionar o problema reduzido, gerando melhores resultados em quase todas as instâncias. O MPO superou o MDM-GRASP-RC em média em 23 instâncias. Além disso, o MPO superou os melhores resultados em 22 instâncias, empatou em duas e perdeu em apenas uma.

Como trabalhos futuros, podemos utilizar o resolvedor de PLI inserido na heurística MDM-GRASP-RC, transformando o processo de pós-otimização em uma estratégia híbrida de mineração de dados com métodos exatos. Por exemplo, em vez de finalizar a heurística após a execução do resolvedor, a solução obtida por este último método exato poderia ser retornada para a heurística original, e ser candidata a pertencer ao conjunto elite da mineração de dados ou ao conjunto da reconexão por caminhos. Dessa forma, a heurística original poderá também obter informações úteis do resolvedor, acelerando a convergência no encontro de melhores soluções.

Referências

Aiex, R. M., Resende, M. G. C., e Ribeiro, C. C. (2007). TTT plots: a perl program to create time-to-target plots. *Optimization Letters*, 1:355–366.

Barbalho, H., Rosseti, I., Martins, S. L., e Plastino, A. (2013). A hybrid data mining GRASP with path-relinking. *Computers & Operations Research*, 40:3159–3173.

Boussaïd, I., Lepagnot, J., e Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82 – 117.

Chaovalitwongse, W. A., Oliveira, C. A., Chiarini, B., Pardalos, P. M., e Resende, M. G. C. (2011). Revised GRASP with path-relinking for the linear ordering problem. *Journal of Combinatorial Optimization*, 22:572–593.

Conover, W. J. e Iman, R. L. (1981). Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35:124–129.

Dahl, G. e Johannessen, B. (2004). The 2-path network problem. *Networks*, 43:190–199.

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

Festa, P. e Resende, M. G. C. (2009). An annotated bibliography of GRASP–Part I: Algorithms. *International Transactions in Operational Research*, 16:1–24.

Glover, F., Laguna, M., e Martí, R. (1977). Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics*, 19:653–684.

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

IBM. ILOG CPLEX Optimizer. Último acesso em 2/10/2014, disponível em: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

Kelly, J. P. (1996). *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, USA.

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

Resende, M. G. C. e Ribeiro, C. C. (2005). GRASP with path-relinking: Recent advances and applications. Em *Metaheuristics: progress as real problem solvers*, páginas 29–63. Springer.

Rosseti, I. C. (2003). *Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos*. Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

Wolsey, L. A. e Nemhauser, G. L. (1988). *Integer and combinatorial optimization*. Wiley.