Abordagem Heurística para o Problema de Reconstrução de Redes de Transporte com Prazos de Recuperação

Jonatas Batista Costa das Chagas
Universidade Federal de Viçosa
Viçosa - MG - Brasil
jonatas.chagas@ufv.br

André Gustavo dos Santos
Universidade Federal de Viçosa
Viçosa - MG - Brasil
andre@dpi.ufv.br

RESUMO

O Problema de Reconstrução de Redes de Transporte com Prazos de Recuperação (PRRTPR) surge quando as conexões (arestas) de uma rede de transporte são destruídas ou danificadas por algum desastre. A cada vértice da rede é associado um prazo de recuperação (*deadline*). O problema tem como objetivo reconectar todos os vértices da rede, minimizando o maior atraso na recuperação dos vértices. A reconstrução é realizada por uma única equipe de trabalho, que inicialmente está localizada em algum vértice da rede. A velocidade de restauração é constante e a equipe de trabalho pode viajar dentro da parte já restaurada da rede com tempo considerado zero. Neste trabalho é proposta uma abordagem heurística baseada na metaheurística Simulated Annealing para resolver o problema. Os resultados obtidos são comparados com os de um método exato (branch-and-bound) existente na literatura, mostrando ser competitivo, principalmente, para as instâncias maiores, onde melhora alguns resultados existentes.

PALAVRAS CHAVE. Reconstrução de Redes, Scheduling, Metaheurística.

Tópicos: MH - Metaheurísticas, OC - Otimização Combinatória

ABSTRACT

The Transportation Network Reconstruction Problem with Deadlines of Recuperation (TNRPDR) arises when the connections (edges) of a transportation network are damaged or destroyed by a disaster. To each vertex of the network a value of deadline is associated. The objective of the problem is to reconnect all the vertices of the network, minimizing the longest lateness in the recovery of the vertices. The restoration is performed by a single team work, which initially is located at some vertex of the network. The restore speed is constant and the work team can travel within the already restored part of the network in zero time. We propose a heuristic approach based on the Simulated Annealing metaheuristic to solve the problem. The results are compared to those of an exact method (branch-and-bound) existing in the literature, showing to be competitive, mainly, for large instances, for which some existing results were improved.

KEYWORDS. Network Reconstruction, Scheduling, Metaheuristic.

Paper topics: MH - Metaheuristics, OC - Combinatorial Optimization

1. Introdução

Atentados terroristas e desastres naturais como enchentes, terremotos, furacões e tempestades causam enormes prejuízos ecológicos e econômicos, além de fazer milhares de vítimas todos os anos (Guha-Sapir et al. [2015]; Tavares [2004]; Abadie e Gardeazabal [2008]). Geralmente, o número de vítimas aumenta na fase pós-desastre devido à dificuldade de acessar as áreas afetadas, impossibilitando o atendimento às vítimas (PAHO [2000]).

Os danos causados pelos desastres nas estradas impedem que as equipes de resgate acessem as regiões afetadas. Decidir quais estradas e a ordem que serão reconstruídas é crucial para diminuir o impacto causado pelos desastres (Berktaş et al. [2016]).

A literatura apresenta diversos estudos sobre diferentes problemas de reconstrução de redes. Feng e Wang [2003] desenvolveram um modelo multi-objetivo envolvendo maximização do desempenho de reabilitação da rede, minimização do risco dos socorristas e maximização do número de vidas salvas, onde as estradas são reparadas por vários grupos de trabalho para maximizar a eficácia do resgate. Hu e Sheu [2013] também desenvolveram um modelo multi-objetivo que inclui três objetivos conflitantes envolvendo custos operacionais relacionados ao transporte e reciclagem de detritos, riscos ambientais e traumas psicológicos das vítimas dos desastres. Berktaş et al. [2016] propuseram modelos matématicos e algoritmos heurísticos para dois diferentes problemas, onde o objetivo do primeiro problema é minimizar o tempo total gasto para alcançar todos os vértices críticos da rede e o objetivo do segundo problema é minimizar a soma ponderada dos tempos gastos para alcançar os vértices, onde pesos indicam as prioridades de cada vértice da rede.

Averbakh [2012] e Averbakh e Pereira [2012] definiram diferentes problemas de reconstrução de redes. Em tais problemas supõe-se que existe uma rede de transporte cujas conexões entre pares de pontos tenham sido destruídas e que precisam ser restauradas, a fim de reconectar todos os pontos dessa rede, ou seja, restaurar a rede de forma que todos os pontos possam ser atingidos. Para reconstruir tais conexões são dispostas várias equipes de trabalho que, inicialmente, estão localizadas em alguns vértices da rede (depósitos). Todas começam a trabalhar simultaneamente no tempo 0 e cada uma é capaz de restaurar uma unidade de comprimento da rede por uma unidade de tempo. Quando k equipes trabalham no mesmo trecho, a velocidade de reparação daquele trecho aumenta k vezes. Considera-se que cada equipe pode viajar nas partes já restauradas da rede com uma velocidade infinita, isto é, em qualquer momento, podem se transferir para qualquer ponto dentro da rede já construída com tempo de deslocamento igual a zero. Isso se dá pelo fato que a velocidade de restauração é incomparavelmente mais lenta do que a velocidade de deslocamento em conexões já restauradas. Neste trabalho, assim como nos anteriores, considera-se apenas 1 equipe de trabalho.

O instante em que um vértice é alcançado pela primeira vez por alguma equipe de trabalho é chamado de tempo de recuperação daquele vértice. A todo vértice é associado um valor de prazo de recuperação (*deadline*), que informa o tempo máximo que as equipes de trabalho devem levar para construir uma conexão até aquele vértice. Esse *deadline* representa um limite de auto sustentabilidade do vértice, ou seja, um limite de tempo em que cada vértice pode permanecer isolado do resto da rede sem causar danos. Por exemplo, o valor do *deadline* associado a cada vértice pode representar uma estimativa de quanto tempo as pessoas localizadas na região representada pelo vértice ainda terão água potável.

O objetivo dos problemas é determinar um cronograma de construção (*scheduling*) que minimiza alguns objetivos de programação em função dos tempos de recuperação dos vértices.

Averbakh e Pereira [2015] definem três problemas com diferentes objetivos. Os problemas descritos definem a existência de apenas uma equipe de trabalho, que inicialmente é localizada em um vértice especifico da rede (depósito). A equipe de trabalho sempre trabalha na reconstrução de uma única conexão por vez. Tais problemas são definidos e nomeados como segue abaixo:

- **Problema L**: o objetivo é minimizar o maior atraso dos vértices (*Lateness*)
- **Problema T**: o objetivo é minimizar o número de vértices com atraso (*Tardiness*)

- **Problema F**: problema de decisão, onde é preciso dizer se é possível recuperar todos os vértices sem atraso ou não (*Feasibility*)

Averbakh e Pereira [2015] demostraram formalmente que todos os três problemas pertencem a classe de problemas NP-Hard, propuseram uma formulação matématica (MILP) para cada problema e também propuseram um algoritmo branch-and-bound (B&B) para resolvê-los.

Dada a alta complexidade dos problemas, este trabalho descreve uma abordagem heurística baseada na metaheurística Simulated Annealing (SA) com o objetivo de encontrar soluções de boa qualidade em tempo aceitável para o problema L acima. A heurística proposta pode ser usada para resolver os problemas T e F, com pequenas e simples alterações, mas os resultados não serão apresentados para esses problemas porque os resultados do B&B não foram disponibilizados pelos autores, impossibilitando a comparação dos diferentes métodos de resolução.

Este trabalho é organizado da seguinte forma: na seção 2 o Problema de Reconstrução de Redes de Transporte com Prazos de Recuperação (PRRTPR) é formalmente definido, é apresentado um exemplo ilustrativo e é descrito o modelo de programação linear inteira mista definido por Averbakh e Pereira [2015]. Na seção 3 é apresentada a heurística proposta, detalhando a estrutura de representação da solução, a função de avaliação e os passos da heurística. A seção 4 apresenta os resultados da heurística proposta, que são comparados com os resultados do B&B da literatura. Por fim, as conclusões obtidas são apresentadas na seção 5 e é comentado sobre um próximo possível trabalho.

2. Formalização do Problema L

Seja $G = (V, E)$ uma rede com o conjunto de vértices $V = \{1, 2, ..., n\}$ e o conjunto de arestas E. Para cada aresta $(i, j) \in E$ é associado o valor C_{ij}, referente ao tamanho da aresta que conecta esses dois vértices. As arestas de E são as conexões candidatas à reconstrução. A equipe de trabalho que inicialmente está localizada no vértice 1 (depósito) é capaz de construir uma conexão com uma velocidade constante de 1 unidade de tamanho por uma unidade de tempo. A velocidade de viagem da equipe de trabalho, dentro da rede já construída, é infinita, ou seja, a qualquer momento, a equipe de trabalho pode se transferir para qualquer ponto dentro da rede já construída com tempo de percurso zero.

A equipe de trabalho começa a trabalhar no tempo 0. O instante de tempo c_j é o tempo que a equipe de trabalho recupera o vértice j, ou seja, o instante de tempo que a equipe de trabalho alcança, pela primeira vez, o vértice j. O tempo de recuperação do vértice inicial (depósito) é definido como 0, ou seja, $c_1 = 0$.

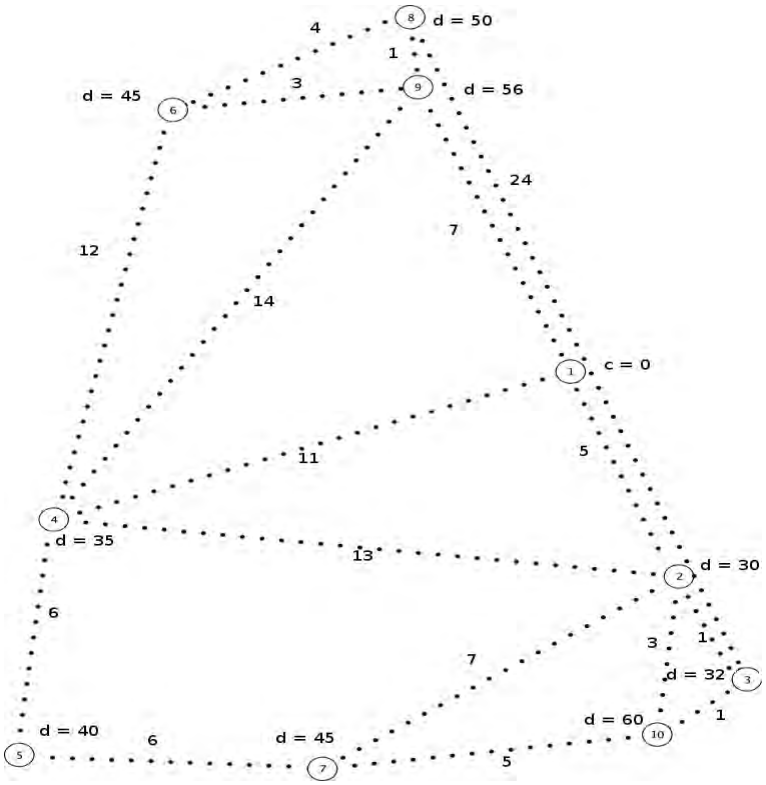
A cada vértice $j > 1$ é associado um valor d_j de *deadline* que informa o tempo máximo que a equipe de trabalho pode levar para construir uma conexão até aquele vértice. O vértice 1 não apresenta *deadline* já que é o ponto de localização inicial da equipe de trabalho. Os vértices são rotulados de acordo com a ordem não-crescente de *deadline*, isto é, $d_2 \leq d_3 ... \leq d_n$.

O valor $c_j - d_j$ é o tempo de atraso (*lateness*) do vértice j. Caso a equipe de trabalho recupere o vértice j antes do prazo de recuperação associado a tal vértice, ou seja, $(c_j < d_j)$, o valor $c_j - d_j$ será negativo, sendo possível que a solução apresente valor negativo.

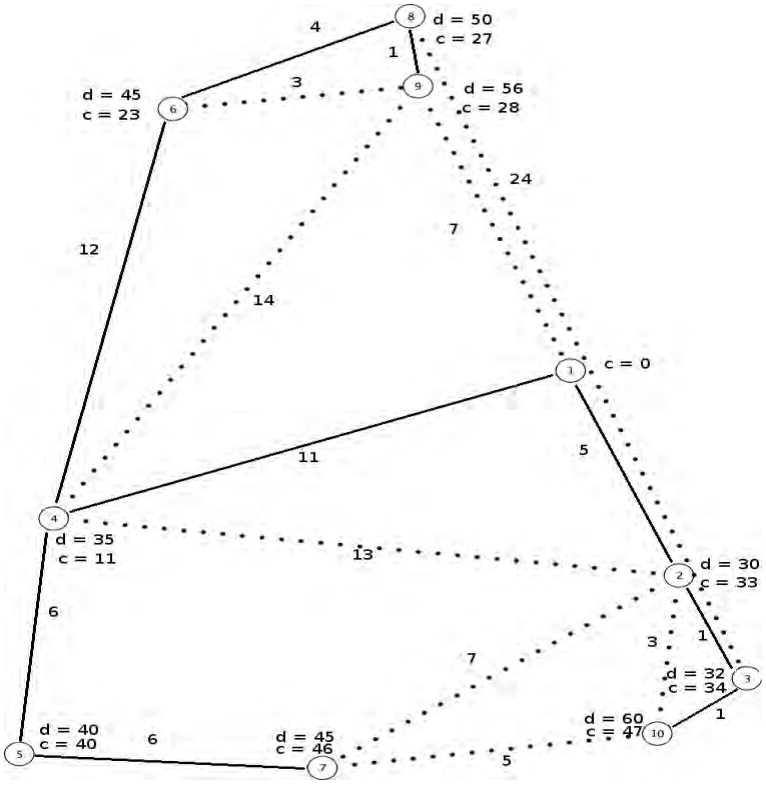
O objetivo do problema é determinar um agendamento de construções de forma que o maior *lateness* seja o mínimo possível, isto é, $\min\{max_{j=2}^{n}(c_j - d_j)\}$.

2.1. Exemplo ilustrativo

A figura 1 apresenta um exemplo de rede, bem como uma possível solução para tal exemplo. A rede da figura 1(a) contém 10 vértices e as arestas tracejadas indicam as conexões candidatas à reconstrução. Os valores ao lado das arestas representam seus comprimentos. O valor de *deadline* de cada vértice está indicado pela letra d ao lado dos vértices. A solução representada na figura 1(b) mostra as arestas (linhas contínuas) que são as conexões reconstruídas. O valor de recuperação de cada vértice está indicado na figura pela letra c. A ordem de construção das conexões foi: $(1 \rightarrow 4), (4 \rightarrow 6), (6 \rightarrow 8), (8 \rightarrow 9), (1 \rightarrow 2), (2 \rightarrow 3), (4 \rightarrow 5), (5 \rightarrow 7), (3 \rightarrow 10)$. O maior atraso que foi gerado por essa solução foi com o valor 3, no vértice 2.



(a)



(b)

Figura 1: a) Exemplo de uma rede destruída/danificada; b) Exemplo de uma possível solução para tal rede. (Adaptada de Averbakh e Pereira [2015])

2.2. Formulação MILP

Abaixo é descrito o modelo de programação linear inteira mista (MILP) proposto por Averbakh e Pereira [2015] para o PRRTPR versão L:

Dados:

- G: grafo que representa a rede de conexão
- V: conjunto de vértices a serem recuperados
- E: conjunto de arestas
- E': conjunto de arestas $(i, j) \in E$ tal que $j \neq 1$
- C_{ij}: comprimento da aresta (i, j)
- n: número de vértices
- d_j: valor de *deadline* do vértice j

Variáveis de decisão:

- x_{ij}^k: 1 se a aresta (i, j) é a k-ésima aresta construída; 0 caso contrário
- t_k: tempo em que o k-ésimo vértice foi recuperado
- c_j: tempo em que o vértice j foi recuperado
- z: valor do maior *lateness*

Modelo:

$$\min z \tag{1}$$

sujeito a

$$z \geq c_j - d_j \qquad \forall j \in V \setminus \{1\} \tag{2}$$

$$\sum_{k \in [1:n-1]} t_k = \sum_{j \in V \setminus \{1\}} c_j \tag{3}$$

$$t_k = \sum_{k' \in [1:k]} \sum_{(i,j) \in E'} C_{ij} x_{ij}^{k'} \qquad \forall k \in [1 : n-1] \tag{4}$$

$$c_j \geq t_k - M \sum_{k' \in [1:k-1]} \sum_{i:(i,j) \in E'} x_{ij}^{k'} \qquad \forall k \in [1 : n-1], j \in V \setminus \{1\} \tag{5}$$

$$\sum_{i:(1,i) \in E'} x_{1i}^1 = 1 \tag{6}$$

$$\sum_{(i,j) \in E'} x_{ij}^k = 1 \qquad \forall k \in [2 : n-1] \tag{7}$$

$$\sum_{k \in [1:n-1]} \sum_{i:(i,j) \in E'} x_{ij}^k = 1 \qquad \forall j \in V \setminus \{1\} \tag{8}$$

$$x_{ij}^k \leq \sum_{k' \in [1:k-1]} \sum_{i':(i',i) \in E'} x_{i'i}^{k'} \qquad \forall (i, j) \in E', i \neq 1, k \in [1 : n-1] \tag{9}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall (i, j) \in E', k \in [1 : n-1] \tag{10}$$

$$t_k \geq 0 \qquad \forall k \in [1 : n-1] \tag{11}$$

$$c_j \geq 0 \qquad \forall j \in V \setminus \{1\} \tag{12}$$

$$z \in \mathbb{R} \tag{13}$$

A função objetivo que minimiza o maior atraso dos vértices é definida pelas equações (1) e (2). A restrição (3) é desnecessária, mas assegura que os valores das variáveis t_k, de fato representam os tempos de recuperação dos vértices. A restrição (4) calcula o tempo de recuperação do k-ésimo vértice. A restrição (5) calcula o tempo de recuperação do vértice j. A restrição (6) garante que a equipe de trabalho começa a trabalhar no vértice 1 (depósito). A restrição (7) assegura que apenas uma conexão é reconstruída por vez. A restrição (8) impede a formação de ciclos e garante que todos os vértices sejam recuperados. A restrição (9) garante que a ordem de construção das conexões seja viável. As restrições (10), (11), (12) e (13) definem os domínios das variáveis de decisão.

A constante M na restrição (5) deve ser um valor maior ou igual ao tempo de recuperação de todos os vértices. O valor de M foi definido por Averbakh e Pereira [2015] como sendo o valor do número de vértices vezes a maior aresta da rede, ou seja, $M = n \cdot (\max_{\{i,j\} \in E} C_{ij})$

3. Abordagem heurística

Esta seção descreve a heurística proposta, detalhando a forma de representação e avaliação da solução, bem como seus passos e demais componentes.

3.1. Representação da solução

A solução é representada por um vetor que armazena a ordem em que os vértices são recuperados. A figura 2 mostra como a solução da figura 1(a) é representada. O primeiro vértice a ser recuperado é o vértice 4, seguido pelos vértices 6, 8, 9, 2, 3, 5, 7 e 10.

1	2	3	4	5	6	7	8	9
4	6	8	9	2	3	5	7	10

Figura 2: Representação da solução ilustrada na figura 1(b).

3.2. Avaliação da solução

A representação da solução informa apenas qual é a ordem em que os vértices serão recuperados, mas não informa quais arestas serão reconstruídas para recuperar tais vértices. Dada uma ordem de recuperação dos vértices, a decisão de quais arestas reconstruir pode ser feita da melhor forma possível, bastando que a equipe de trabalho sempre construa o caminho mais curto entre algum ponto da rede já construída até o vértice a ser recuperado. Logo, dada uma ordem de recuperação dos vértices, a avaliação de tal ordem é feita usando o Algoritmo 1. Note que na construção do caminho mínimo até o vertice a se recuperar, os vértices deste caminho também são recuperados.

Algoritmo 1: Função de avaliação *f(.)*

```
1  fitness ← ∞
2  verticesRecuperados ← {1} /** Depósito **/
3  for i ← 1 to n − 1 do
4  |  P ← caminho mínimo entre o i-ésimo vértice a ser recuperado e algum vértice que já foi recuperado
5  |  foreach v ∈ P do
6  |  |  if v ∉ verticeRecuperados then
7  |  |  |  verticesRecuperados ← verticesRecuperados ∪ {v}
8  |  |  |  c_v ← tempo gasto pela equipe de trabalho na recuperação da rede até o momento
9  |  |  |  fitness ← max(fitness, c_v − d_v)
10 |  |  end
11 |  end
12 end
13 return fitness
```

3.3. Estrutura de vizinhança Swap

Na estrutura de vizinhança utilizada para explorar o espaço de soluções um vizinho é obtido trocando-se dois vértices (swap) na ordem estabelecida de recuperação. A figura 3 ilustra um exemplo da geração de um vizinho utilizando a estrutura de vizinhança Swap.

Figura 3: Exemplo de vizinho na vizinhança Swap.

3.4. Metaheurística Simulated Annealing

Proposta por Kirkpatrick [1984] e Černỳ [1985], a metaheurística Simulated Annealing (SA) trata-se de uma técnica de busca local probabilística, que se basea em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como recozimento. Diferente de uma busca local, o Simulated Annealing permite que a solução corrente piore. A decisão de escolher uma pior solução é baseada em uma probabilidade que é em função do valor da solução corrente e da temperatura: quanto maior a temperatura e menor o valor da solução, maior é a chance de aceitar uma solução pior. O algoritmo termina quando é atingida uma temperatura baixa, ou seja, quando é atingido um equilíbrio. O Algoritmo 2 descreve a heurística proposta para o PRRTPR, baseada na metaheurística Simulated Annealing.

Algoritmo 2: Simulated Annealing (SA)

```
1  melhor_solucao ← {}
2  repeat
3      s ← solução inicial gerada aleatoriamente
4      T ← t_0
5      while T > t_min do
6          it ← 0
7          while it < num_it do
8              s' ← selecione aleatorimente uma solução vizinha à solução s utilizando a estrutura de vizinha Swap
9              Δ ← f(s') − f(s)
10             if Δ < 0 or random[0, 1] < e^{−Δ/t} then
11                 s ← s'
12                 if f(s') < f(melhor_solucao) then
13                     melhor_solucao ← s'
14                 end
15             end
16             it ← it + 1
17         end
18         T ← T × (1 − α)
19     end
20 until critérios de parada serem satisfeitos;
21 return melhor_solucao
```

A solução inicial do algoritmo é gerada de forma aleatória, escolhendo casualmente uma ordem de recuperação dos vértices. Enquanto os critérios de parada não são satisfeitos, o algoritmo seleciona aleatoriamente uma solução vizinha à solução atual e a aceita de acordo com uma probabilidade determinada em função da qualidade da solução e também da temperatura vigente. Ao fim do algoritmo, a melhor solução encontrada é retornada.

Em relação aos parâmetros utilizados no SA: o laço mais externo do algoritmo (critério de parada) é executado 100 vezes, a temperatura inicial foi definida como 1000 ($t_0 = 1000$), o coeficiente de resfriamento utilizado foi 0.05 ($\alpha = 0.05$), o valor da temperatura mínima utilizado foi de 0.001 (t_{min} = 0.001) e o número de iterações feitas com a mesma temperatura foi de número de vértice elevado ao quadrado (num_it = n^2). Tais valores foram obtidos empiricamente após testes preliminares de calibração.

4. Experimentos computacionais

Essa seção descreve a configuração do computador e do ambiente que foi usado na experimentação, as características das instâncias e, por fim, os resultados obtidos.

4.1. Ambiente de desenvolvimento

O algoritmo proposto foi codificado na linguagem C/C++, compilado com GNU g++ versão 4.8.4 e executado, de modo sequencial, em um computador Intel Core i5-3570 CPU @ 3.40GHz x 4, com 16GB de memória RAM e sistema operacional Ubuntu 14.04 LTS, 64 bits.

4.2. Instâncias

Foram utilizados dois conjuntos de instâncias para se fazer a experimentação do algoritmo proposto. Os conjuntos de instâncias foram desenvolvidos e disponibilizados pelos autores Averbakh e Pereira [2015]. O primeiro conjunto é composto por 640 instâncias geradas aleatoriamente e o segundo conjunto de instâncias trata-se de instâncias cujos dados foram obtidos de um desastre real, o terremoto que ocorreu em fevereiro de 2010 no Chile.

As instâncias do conjunto gerado aleatoriamente diferem entre si através de características definidas por 3 parâmetros: número de vértices (n); *range of due dates* (RDD); *tardiness factor* (TF). Os parâmetros RDD e TF foram utilizados para gerar os valores de *deadline* dos vértices, sendo que para cada vértice o valor de *deadline* é um número inteiro gerado através da distribuição uniforme descrita pela fórmula (14), onde P é o comprimento da árvore geradora mínima da rede.

$$\left[\mathrm{P}\left(1-\mathrm{TF}-\frac{\mathrm{RDD}}{2}\right), \mathrm{P}\left(1-\mathrm{TF}+\frac{\mathrm{RDD}}{2}\right)\right] \tag{14}$$

O comprimento P da árvore geradora mínima e os parâmetros RDD e TF definem os limites inferior e superior ($\mathrm{P}\,(1-\mathrm{TF}-\mathrm{RDD}/2)$ e $\mathrm{P}\,(1-\mathrm{TF}+\mathrm{RDD}/2)$) dos valores atribuídos aos *deadlines* dos vértices da rede.

Averbakh e Pereira [2015] geraram instâncias de 25, 30, 35, 40, 45, 50, 55 e 60 vértices e definiram quatro valores distintos $\{0.2, 0.4, 0.5, 0.8\}$ para os parâmetros RDD e TF. A combinação dos valores definidos para os três parâmetros resultam em 128 tipos de instâncias. Para cada um desses 128 tipos diferentes foram geradas 5 instâncias, totalizando 640.

As instâncias relacionadas ao terremoto que ocorreu em fevereiro de 2010 no Chile foram geradas através de informações disponibilizadas pelo governo chileno. Foram geradas 80 instâncias, todas de 53 vértices, sendo que a rede de conexão é a mesma para todas elas. As 80 instâncias diferem umas das outras apenas pelos valores de *deadline*, que também foram gerados através da distruibuição uniforme descrita pela fórmula (14) utilizando os mesmos valores dos parâmetros RDD e TF usados nas instâncias aleatórias.

Mais detalhes sobre a geração de instâncias e obtenção de informações podem ser obtidos no trabalho de Averbakh e Pereira [2015].

4.3. Resultados

O algoritmo B&B, proposto por Averbakh e Pereira [2015], mostrou-se eficiente para a maioria das instâncias testadas. Os autores limitaram o tempo de execução de seu algoritmo a 600 segundos, ou seja, caso o B&B não encontrasse a solução ótima com menos de 600 segundos, a melhor solução conhecida era retornada. Do conjunto de 640 instâncias aleatórias, o B&B encontrou a solução ótima para 506 delas. No conjunto das 80 instâncias relativas aos dados do terremoto ocorrido no Chile, o B&B encontrou a solução ótima para todas elas.

A tabela 1 mostra um resumo dos resultados encontrados pela heurística proposta e pelo algoritmo B&B descrito por Averbakh e Pereira [2015], para as instâncias geradas aleatoriamente. As colunas n e RDD definem as características das instâncias (os diferentes valores de TF foram agrupados), sendo n o número de vértices e RDD o *range of due dates*. Para cada valor de n e RDD existem 20 instâncias diferentes. As informações do algoritmo B&B foram extraídas de Averbakh e Pereira [2015] e são descritas nas colunas #, t(s) e Nodes, respectivamente o número de instâncias que o algoritmo encontrou a solução ótima, o tempo médio gasto para executar as 20 instâncias e a quantidade média (em milhões) de nós expandidos pelo B&B para as 20 instâncias. Os resultados da heurística proposta ocupam quatro colunas da tabela 1, sendo que as colunas P, I e M indicam o número de instâncias que tal heurística encontrou resultados piores, iguais e melhores, respectivamente, que o B&B. Já a coluna t(s) indica o tempo médio gasto pela heurística para executar as 20 instâncias.

Tabela 1: Resultado da comparação entre os métodos utilizando as instâncias aleatórias

n	RDD	B&B [1]			SA			
		#	t(s)	Nodes	P	I	M	t(s)
25	0.2	20	0.0	0.00	0	20	0	6.0
	0.4	20	0.0	0.01	0	20	0	5.9
	0.6	20	0.0	0.01	1	19	0	5.9
	0.8	20	0.0	0.02	0	20	0	5.9
30	0.2	20	0.1	0.01	2	18	0	9.5
	0.4	20	0.0	0.05	1	19	0	9.5
	0.6	20	0.3	0.11	0	20	0	9.5
	0.8	20	0.2	0.10	0	20	0	9.7
35	0.2	20	0.8	0.12	0	20	0	15.6
	0.4	20	2.6	0.41	0	20	0	15.6
	0.6	20	3.1	0.51	1	19	0	15.6
	0.8	20	2.0	0.34	0	20	0	15.6
40	0.2	20	1.6	0.32	0	20	0	23.4
	0.4	20	18.1	2.03	4	16	0	23.4
	0.6	20	47.4	5.54	2	18	0	23.4
	0.8	20	45.0	5.57	1	19	0	23.4
45	0.2	20	10.9	1.17	2	18	0	36.4
	0.4	18	152.9	16.23	1	18	1	36.0
	0.6	15	245.1	26.07	3	16	1	37.2
	0.8	20	170.8	18.41	4	16	0	37.6
50	0.2	20	7.8	0.81	4	16	0	51.8
	0.4	18	113.8	10.50	4	16	0	51.6
	0.6	6	483.2	47.05	0	15	5	50.7
	0.8	8	389.8	38.46	0	14	6	50.7
55	0.2	20	48.7	4.68	4	16	0	70.9
	0.4	9	382.3	34.27	9	8	3	73.7
	0.6	3	533.0	46.09	5	8	7	73.9
	0.8	4	515.3	45.28	5	5	10	73.8
60	0.2	19	110.7	9.38	5	15	0	98.9
	0.4	5	467.4	38.64	1	16	3	98.4
	0.6	1	577.3	46.74	1	3	16	100.1
	0.8	0	600.0	50.64	0	8	12	100.6
Soma	-	506	4930.2	-	60	516	64	1260.0

[1] O algoritmo B&B foi executado em um computador Inter Core 2 Duo 2.33 gigahertz, com 2 gigabytes de RAM e sistema operacional Mac OS X 10.4.11

Os resultados da tabela 1 mostram que o algoritmo B&B encontrou a solução ótima para todas as 340 instâncias de $n \leq 40$, enquanto o SA encontrou a solução ótima para 308 instâncias. Também é possível constatar que o B&B se comporta melhor para os menores valores de RDD,

pois quanto menor o RDD mais homogêneos são os valores associados aos *deadlines* dos vértices, facilitando as podas do algoritmo B&B, o que pode ser comprovado pela coluna Nodes da tabela 1.

O SA proposto começou a ser competitivo com o algoritmo B&B a partir das instâncias de $n \geq 45$, pois ele encontrou poucas soluções piores que o B&B com tempo de execução menor para quase todos os grupos de instâncias. Para as instâncias com $n \geq 55$, ele encontra várias soluções mehores, principalmente para valores mais altos de RDD.

A tabela 2 mostra as mesmas informações mostradas pela tabela 1 (menos a coluna Nodes que informa o número médio de nós expandidos), porém considerando as instâncias relativas ao terremoto chileno de 2010.

Tabela 2: Resultado da comparação entre os métodos utilizando as instâncias relativas ao terremoto chileno de 2010

n	RDD	B&B [2]		SA			
		#	t(s)	P	I	M	t(s)
53	0.2	20	0.0	0	20	0	21.3
	0.4	20	1.8	0	20	0	21.2
	0.6	20	4.5	0	20	0	21.2
	0.8	20	2.9	0	20	0	21.2
Soma	-	80	9.2	0	80	0	84.9

[2] O algoritmo B&B foi executado em um computador Inter Core 2 Duo 2.33 gigahertz, com 2 gigabytes de RAM e sistema operacional Mac OS X 10.4.11

Como pode ver visto, o SA encontrou soluções ótimas para todas as 80 instâncias testadas, porém com tempo de execução superior aos do algoritmo B&B.

Uma análise da qualidade das soluções obtidas pela metaheurística proposta e pelo algoritmo B&B é mostrado na tabela 3. Para tal análise foi utilizada a métrica RPD (*Relative Percentage Deviation*) que é uma métrica usada para avaliar resultados de algoritmos heurísticos. A fórmula (15) mostra como a métrica é calculada.

$$\text{RPD}_{método} = \left| \frac{f_{melhor} - f_{método}}{f_{melhor}} \right| \times 100\% \tag{15}$$

onde f_{melhor} é o valor da melhor solução encontrado entre algoritmos comparados e $f_{método}$ é o valor da solução de um dado método. Quanto menor o RPD melhor, pois mais próximo a solução comparada está da melhor solução conhecida.

A tabela 3 apresenta as médias dos RPD's para as instâncias aleatórias, sendo possível notar que o SA proposto se comporta melhor que o algoritmo B&B para as maiores instâncias que apresentam RDD igual 0.6 e 0.8.

A figura 4 mostra o gráfico dos valores de RPD da tabela 3, no qual é possível ver mais claramente o comportamento positivo do SA proposto à medida que as instâncias aumentam de tamanho e RDD.

Tabela 3: Médias dos RPD's para as instâncias aleatórias

n	RDD	$\text{RPD}_{B\&B}$	RPD_{SA}	n	RDD	$\text{RPD}_{B\&B}$	RPD_{SA}
25	0.2	0.00%	0.00%	45	0.2	0.00%	0.03%
	0.4	0.00%	0.00%		0.4	0.06%	0.24%
	0.6	0.00%	0.01%		0.6	0.03%	0.07%
	0.8	0.00%	0.00%		0.8	0.00%	5.98%
30	0.2	0.00%	0.78%	50	0.2	0.00%	0.43%
	0.4	0.00%	0.24%		0.4	0.00%	1.34%
	0.6	0.00%	0.00%		0.6	4.40%	0.00%
	0.8	0.00%	0.00%		0.8	4.00%	0.00%
35	0.2	0.00%	0.00%	55	0.2	0.00%	0.73%
	0.4	0.00%	0.00%		0.4	0.27%	5.86%
	0.6	0.00%	0.00%		0.6	0.63%	0.45%
	0.8	0.00%	0.00%		0.8	7.13%	0.47%
40	0.2	0.00%	0.00%	60	0.2	0.00%	0.37%
	0.4	0.00%	3.53%		0.4	0.34%	1.45%
	0.6	0.00%	0.46%		0.6	162.04%	0.04%
	0.8	0.00%	0.37%		0.8	12.27%	0.00%
Soma	-	0.00%	5.39%	Soma	-	191.67%	17.68%

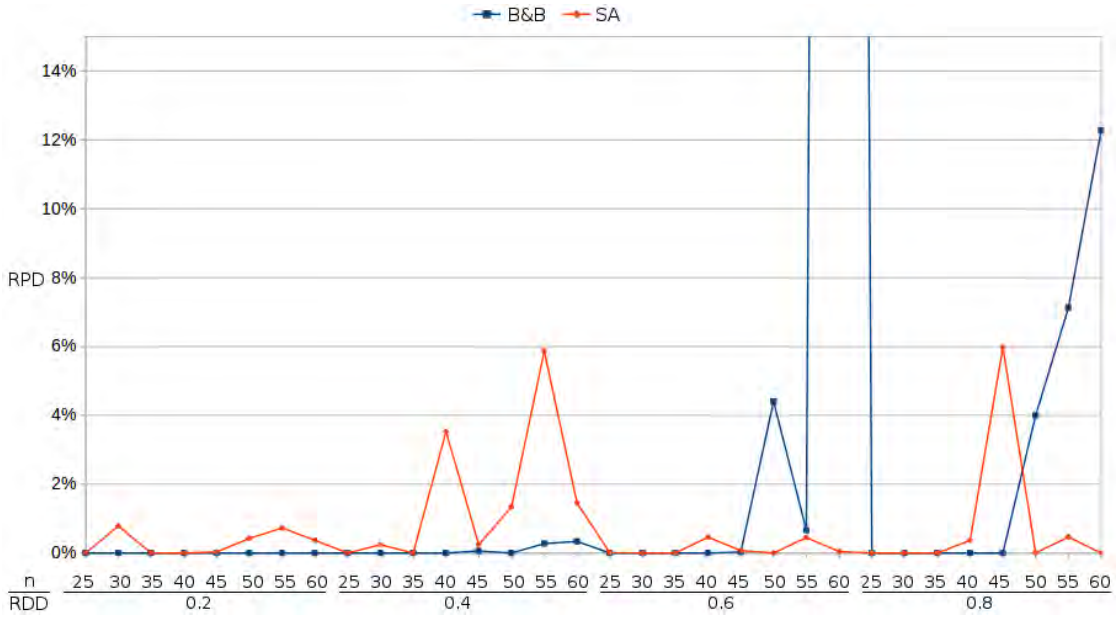


Figura 4: Médias dos RPD's para as instâncias aleatórias - Gráfico da tabela 3

5. Conclusões e trabalhos futuros

Neste trabalho foi proposta uma abordagem heurística baseada na metaheurística Simulated Annealing para o Problema de Reconstrução de Redes de Transporte com Prazos de Recuperação. A abordagem proposta foi testada usando as instâncias definidas por Averbakh e Pereira [2015] com o objetivo de encontrar melhores soluções que as já encontradas na literatura.

Através dos resultados obtidos foi constatado uma superioridade significativa da abordagem proposta (SA) em comparação com o B&B da literatura para as maiores instâncias testadas. Considerando as instâncias aleatórias de até 40 vértices, o B&B se mostrou superior ao SA, encontrando a solução ótima para todas as instâncias, enquanto o SA encontrou a solução ótima para 90.6% delas. Para as instâncias aleatórias com mais de 40 vértices, o SA encontrou soluções piores que o B&B para apenas 15% das instâncias, em 65% das instâncias foi encontrado a mesma solução e para 20% das instâncias o SA superou o B&B. Para as instâncias relativas ao terremoto Chileno

de 2010, o B&B e a abordagem proposta encontraram a solução ótima para todas as 80 instâncias.

Analisando a qualidade das soluções para todas as instâncias testadas foi constatado que a abordagem proposta foi mais eficiente que o B&B, pois a soma dos valores de RPD de cada grupo de instâncias foi menor que a soma desses valores para o B&B.

Por fim, com relação ao tempo de execução, o B&B se mostrou ser mais eficiente que a abordagem proposta para as instâncias de até 40 vértices e para as instâncias de 45, 50 e 55 vértices com RDD igual a 0.2. Analisando de forma geral, o SA foi mais eficiente que o B&B, uma vez que, a soma dos tempos médios de cada grupo de instâncias gasto pelo B&B foi maior que o tempo gasto pela abordagem proposta.

Como trabalho futuro, pretende-se implementar o algoritmo B&B proposto na literatura, fixando seu lower-bound inicial com o valor retornado pela aborgadem aqui proposta. O objetivo de tal trabalho será buscar melhores soluções para instâncias maiores ou até provar a otimalidade para algumas instâncias.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por estimular o desenvolvimento deste trabalho através da bolsa de estudos e também à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) pelo apoio financeiro para a participação no evento.

Referências

Abadie, A. e Gardeazabal, J. (2008). Terrorism and the world economy. *European Economic Review*, 52(1):1–27.

Averbakh, I. (2012). Emergency path restoration problems. *Discrete Optimization*, 9(1):58–64.

Averbakh, I. e Pereira, J. (2012). The flowtime network construction problem. *IIE Transactions*, 44 (8):681–694.

Averbakh, I. e Pereira, J. (2015). Network construction problems with due dates. *European Journal of Operational Research*, 244(3):715–729.

Berktaş, N., Kara, B. Y., e Karaşan, O. E. (2016). Solution methodologies for debris removal in disaster response. *EURO Journal on Computational Optimization*, p. 1–43.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51.

Feng, C.-M. e Wang, T.-C. (2003). Highway emergency rehabilitation scheduling in post-earthquake 72 hours. *Journal of the 5th Eastern Asia Society for Transportation Studies*, 5: 3276–3285.

Guha-Sapir, D., Hoyois, P., e Below, R. (2015). Annual disaster statistical review 2014: The numbers and trends. Technical report, Institute of Health and Society (IRSS) - Université catholique de Louvain – Brussels, Belgium.

Hu, Z.-H. e Sheu, J.-B. (2013). Post-disaster debris reverse logistics management under psychological cost minimization. *Transportation Research Part B: Methodological*, 55:118–141.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986.

PAHO (2000). *Natural disasters: Protecting the public's health*. Number 575. Pan American Health Organization.

Tavares, J. (2004). The open society assesses its enemies: shocks, disasters and terrorist attacks. *Journal of monetary economics*, 51(5):1039–1070.