Métodos heurísticos para o problema de programação e roteamento de equipes de trabalho na restauração de redes

Alfredo Moreno
Departamento de Engenharia de Produção, UFSCar - São Carlos
Rod.washington Luis,km 235, São Carlos, 13565-905
alfredmorenoarteaga@gmail.com

Pedro Munari
Departamento de Engenharia de Produção, UFSCar - São Carlos
Rod.washington Luis,km 235, São Carlos, 13565-905
munari@dep.ufscar.br

Douglas Alem
Departamento de Engenharia de Produção, UFSCar - Sorocaba
Rod. João Leme dos Santos, Km 110, Sorocaba, 18052-780
douglas@ufscar.br

RESUMO

Desastres naturais podem danificar estradas ou rodovias, as quais devem ser reparadas o mais rapidamente possível para fazer com que as áreas afetadas fiquem accessíveis e permitir a evacuação das vítimas e/ou a distribuição de produtos emergenciais para centros de auxílios. A reparação de estradas implica em decisões de programação e roteamentos de equipes de trabalho. Dada a urgência com que as operações em situações pós-desastres devem ser realizadas, é necessário obter os planos de programação e roteamento das equipes de trabalho da forma rápida. Assim, o objetivo desse trabalho é propor métodos de solução heurísticos para obter soluções razoáveis em pouco tempo computacional para o problema integrado de programação e roteamento de equipes de trabalho na reparação de estradas. Testes computacionais baseados em instâncias da literatura foram realizados. Os resultados mostram que os algoritmos heurísticos encontram soluções de boa qualidade de forma rápida para as instâncias testadas.

PALAVRAS CHAVE. Problema de reparação de redes, Heurística, Modelagem matemática.

Logística e Transportes, Metaheuristicas, Programação Matemática

ABSTRACT

Natural disasters can damage roads or highways, which must be repaired as soon as possible to make the affected areas accessible and allow the evacuation of victims and/or the distribution of emergency supplies to relief centers. Network repair involves crew scheduling and routing decisions. Given the urgency to perform operations in post-disaster situations, it is necessary to determine crew scheduling and routing quickly. Thus, the purpose of this paper is to propose heuristic solution methods to obtain reasonably good solutions in short computational time for the integrated problem of crew scheduling and routing in the network repair problem. Computational tests based on instances from literature were performed. The results show that the proposed heuristics found good solutions quickly for the tested instances.

KEYWORDS. Network repair problem, Heuristic, Mathematical modeling.

Logistics and Transportation, Metaheuristics, Mathematical Programming

Introdução

A cada ano, centenas de desastres ambientais (terremotos, enchentes, furacões, etc.) ou ocasionados pelo homem (ataques terroristas, vazamentos químicos, etc) ocorrem no mundo, deixando milhares de vítimas fatais e milhões de afetados. Segundo dados do *Emergency Events Database* [EM-DAT, 2016], os últimos 30 anos contabilizaram, aproximadamente, 17.361 desastres ao redor do mundo. Para reduzir o impacto negativo desses eventos, os órgãos responsáveis da gestão de desastres devem responder da melhor maneira possível ante esses acontecimentos através de operações de distribuição de suprimentos e/ou evacuação de vítimas. Porém, uma das principais dificuldades para realizar estas operações é a indisponibilidade da infraestrutura necessária (estradas inacessíveis), pois tais infraestruturas são, na maioria dos casos, seriamente danificadas pelo desastre. Assim, torna-se necessário considerar o planejamento das operações de reparação das infraestruturas atingidas para o efetivo restabelecimento das operações de resposta.

As operações de reparação de estradas envolvem decisões de programação e roteamento de equipes de trabalho, as quais tem sido consideradas de forma integrada na literatura recente. Sahin et al. [2016] definiram um modelo para o problema de restauração de estradas bloqueadas por detritos. O modelo determina a ordem em que os nós de demanda devem ser visitados e o caminho que deve ser utilizado entre dois nós consecutivos de demanda. O objetivo é minimizar o tempo total gasto para chegar até as áreas afetadas. Berktaş et al. [2016] consideraram o mesmo problema usando prioridades para os nós de demanda. Assim, o objetivo é minimizar a soma ponderada dos tempos de visita das áreas afetadas. Maya Duque et al. [2016] trataram de forma integrada as decisões de programação e roteamento de equipes de trabalho visando minimizar o tempo que as áreas afetadas ficam inacessíveis a partir de um depósito central de onde deve ser realizada a distribuição de suprimentos. A integração de todas essas decisões torna os modelos de otimização mais complexos [Maya Duque et al., 2016], fazendo com que a resolução direta desses modelos em *softwares* de otimização de propósito geral seja inviável, mesmo para instâncias de pequeno porte. Portanto, na literatura é comum que o problema seja resolvido utilizando-se técnicas heurísticas e/ou meta-heurísticas. Dentre os métodos de solução mais utilizados encontram-se métodos baseados em meta-heurísticas clássicas como algoritmo genético [Tzeng et al., 2000], GRASP (*Greedy Randomized Adaptive Search Procedure*) [Maya Duque et al., 2016], busca Tabu [Salman e Yucel, 2015] e ACO (*Ant Colony Optimization*) [Xu e Song, 2015]. Alguns autores desenvolveram heurísticas específicas para o problema, como Yan e Shih [2009] que propuseram um algoritmo baseado na decomposição da rede do problema original em várias sub-redes e na priorização de alguns dos pontos danificados para restauração.

Nesse trabalho são propostas heurísticas construtivas e de busca em vizinhança para resolver o problema de programação e roteamento de equipes de trabalho na restauração de redes em situações de desastres. As heurísticas propostas são baseadas na decomposição do problema em sub-problemas mais simples com o objetivo de se obter boas soluções em tempos relativamente curtos. No restante desse trabalho será apresentada a descrição do problema (Seção 2), a descrição do método de solução (Seção 3), os resultados computacionais (Seção 4) e as considerações finais (Seção 5).

Descrição do problema

O problema de programação e roteamento de equipes de trabalho na restauração de redes consiste em, dada uma rede afetada por eventos extremos, definir os pontos danificados que devem ser reparados pela equipe de trabalho disponível para a reparação, determinar a sequência em que a equipe de trabalho deve realizar as operações que lhes foram alocadas (programação) e definir as rotas que a equipe de trabalho deve seguir para chegar até os pontos que precisam ser restaurados (roteamento). Adicionalmente, deve-se determinar o caminho que deve ser utilizado para realizar a distribuição dos suprimentos do depósito até as área afetadas. O objetivo é tornar accessível todos os pontos de demanda no menor tempo possível. Considera-se que um ponto de demanda é accessível quando existe um caminho entre o ponto de demanda e o depósito inicial com um comprimento

menor do que uma distância máxima permitida. Todos os pontos danificados devem ser restaurados, mesmo que nem todos esses pontos sejam necessários para conectar as áreas afetadas com o depósito. A Figura 1 mostra um exemplo ilustrativo de uma rede afetada por desastres na região Serrana do Rio de Janeiro. Os pontos danificados são rodovias que conectam várias das cidades afetadas. Existem pontos que representam a interseção de duas ou mais rodovias, os quais não possuem demanda e não precisam ser reparados. As linhas vermelhas representam o exemplo de uma sequência a ser seguida por uma equipe de trabalho para a reparação das rodovias danificadas. Note que a sequência não indica a rota que deve ser seguida pela equipe, apenas a ordem em que os pontos danificados devem ser visitados. As linhas pretas representam um exemplo de um caminho seguido pela equipe de trabalho para ir de um ponto danificado até outro ponto danificado. Para cada par de pontos danificados consecutivos na sequência de reparação deve ser definido um caminho. As linhas azuis representam um exemplo de caminho entre o depósito e uma área afetada. Se o caminho entre o depósito e uma área afetada utiliza uma rodovia danificada, o tempo em que essa área afetada fica accessível depende do tempo em que a rodovia é reparada. Existe um tempo de reparação para cada ponto danificado e um tempo de transporte necessário para a equipe de trabalho atravessar as rodovias.

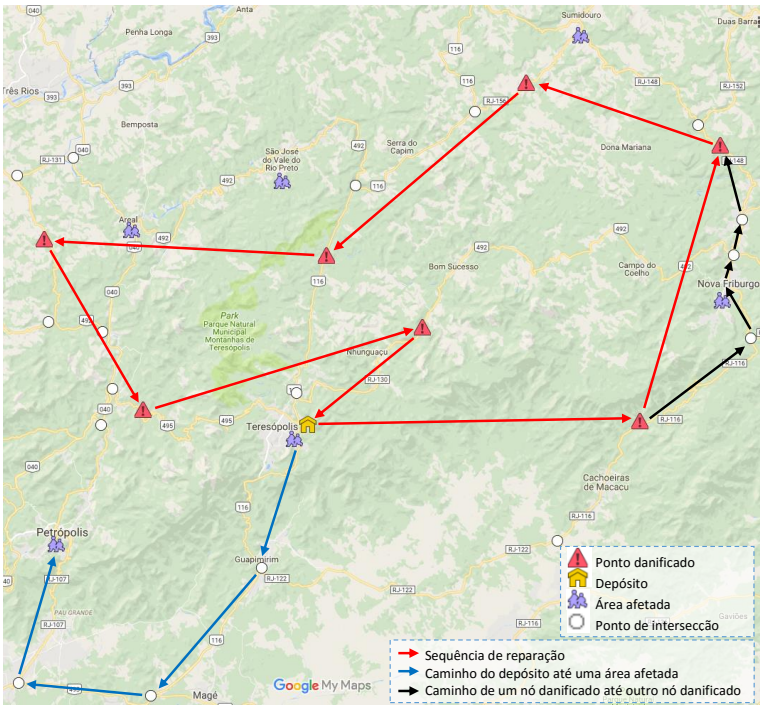

Figura 1: Exemplo de uma rede afetada por desastres.

A rede do problema pode ser representada por um grafo não dirigido e conetado $G = (\mathcal{V}, \mathcal{E})$, em que $\mathcal{V}$ é o conjunto de nós e $\mathcal{E}$ é o conjunto de arestas não dirigidas. Existem nós de demanda ($\mathcal{V}^d \subset \mathcal{V}$) e nós que precisam ser reparados ($\mathcal{V}^r \subset \mathcal{V}$). Sem perda de generalidade, uma aresta que precisa ser reparada é representada como um nó fictício inserido na aresta. Portanto, reparar uma aresta pode ser equivalente a reparar um nó. Os nós que não possuem demanda e não precisam reparação (nós de interseção) são considerados como nós de demanda ($\mathcal{V}^d$) com demanda igual a zero. O grafo que representa a rede da Figura 1 é mostrado na Figura 2.

Modelagem matemática

A partir da representação da rede do problema como um grafo, a formulação matemática proposta por Maya Duque et al. [2016] para o problema utiliza as seguintes definições:

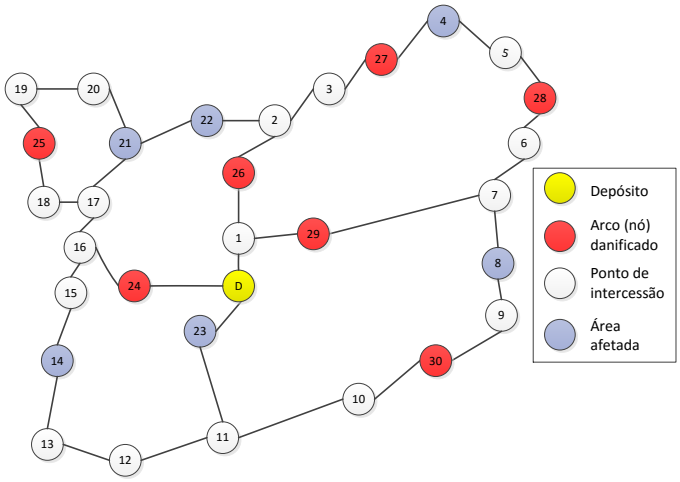

Figura 2: Grafo que representa uma rede afetada por desastres.

Conjuntos

$\mathcal{V}$	Conjunto de nós.
$\mathcal{V}^r \subset \mathcal{V}$	Conjunto de nós danificados.
$\mathcal{V}^d \subset \mathcal{V}$	Conjunto de nós de demanda.
$\mathcal{E}$	Conjunto de arestas.
$\mathcal{E}_i \subseteq \mathcal{E}$	Conjunto de arestas adjacentes ao nó $i \in \mathcal{V}$.

Parâmetros

d_i	Demanda de suprimentos no nó $i \in \mathcal{V}^d$.
δ_i	Tempo requerido para a reparação do nó $i \in \mathcal{V}^r$.
τ_e	Tempo de viagem na aresta e.
ℓ_e	Distância da aresta e.
l_i	Máxima distância aceitável entre o nó de demanda i e o depósito inicial 0.

Variáveis de decisão

$X_{ij}\begin{cases} 1, \text{ se o nó } j \in \mathcal{V}^r \text{ é reparado após o nó } i \in \mathcal{V}^r \cup \{0\}. \\ 0, \text{ caso contrário.} \end{cases}$

$P_{eij}\begin{cases} 1, \text{ se a aresta } e \in \mathcal{E} \text{ é usada no caminho do nó } i \in \mathcal{V}^r \cup \{0\} \text{ para o nó } j \in \mathcal{V}^r. \\ 0, \text{ caso contrário.} \end{cases}$

$N_{kij}\begin{cases} 1, \text{ se o nó } k \in \mathcal{V} \text{ é usado no caminho do nó } i \in \mathcal{V}^r \cup \{0\} \text{ para o nó } j \in \mathcal{V}^r. \\ 0, \text{ caso contrário.} \end{cases}$

$Y_{ej}\begin{cases} 1, \text{ se a aresta } e \in \mathcal{E} \text{ é usado no caminho do depósito inicial } 0 \text{ para o nó } j \in \mathcal{V}^d. \\ 0, \text{ caso contrário.} \end{cases}$

$V_{kj}\begin{cases} 1, \text{ se o nó } k \in \mathcal{V} \text{ é usado no caminho do depósito inicial } 0 \text{ para o nó } j \in \mathcal{V}^d. \\ 0, \text{ caso contrário.} \end{cases}$

Z_i^r	Tempo em que o nó $i \in \mathcal{V}^r$ fica completamente reparado.
Z_i^d	Tempo em que o nó $i \in \mathcal{V}^d$ fica acessível.
R_i	Variável auxiliar para eliminação de sub-tours na rota entre os nós $i \in \mathcal{V}^r \cup \{0\}$.

A partir dessas definições, obtém-se o seguinte modelo de programação inteira mista:

$$\min \quad \sum_{i \in \mathcal{V}^d} d_i \cdot Z_i^d \tag{1}$$

$$\text{s.a.} \sum_{j \in \mathcal{V}^r} X_{ij} = 1, \forall\, i \in \mathcal{V}^r \cup \{0\}. \tag{2}$$

$$\sum_{i \in \mathcal{V}^r \cup \{0\}} X_{ij} = 1, \forall\, j \in \mathcal{V}^r. \tag{3}$$

$$\sum_{e \in \mathcal{E}_i} P_{eij} = X_{ij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r. \tag{4}$$

$$\sum_{e \in \mathcal{E}_j} P_{eij} = X_{ij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r. \tag{5}$$

$$\sum_{e \in \mathcal{E}_k} P_{eij} = 2 \cdot N_{kij}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r,\ k \in \mathcal{V} \setminus \{i,\ j\}. \tag{6}$$

$$\sum_{e \in \mathcal{E}_0} Y_{e0j} = 1, \forall\, j \in \mathcal{V}^d. \tag{7}$$

$$\sum_{e \in \mathcal{E}_j} Y_{e0j} = 1, \forall\, j \in \mathcal{V}^d. \tag{8}$$

$$\sum_{e \in \mathcal{E}_k} Y_{e0j} = 2 \cdot V_{k0j}, \forall\, j \in \mathcal{V}^d,\ k \in \mathcal{V} \setminus \{0,\ j\}. \tag{9}$$

$$\sum_{e \in \mathcal{E}} Y_{e0j} \cdot \ell_e \leq l_j, \forall\, j \in \mathcal{V}^d. \tag{10}$$

$$Z_j^r \geq Z_i^r + \sum_{e \in \mathcal{E}} P_{eij} \cdot \tau_e + (X_{ij} - 1) \cdot M + \delta_j, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r. \tag{11}$$

$$Z_j^r \geq Z_k^r + (N_{kij} - 1) \cdot M, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r,\ k \in \mathcal{V}^r. \tag{12}$$

$$Z_i^d \geq Z_k^r + (V_{k0i} - 1) \cdot M, \forall\, i \in \mathcal{V}^d,\ k \in \mathcal{V}^r. \tag{13}$$

$$R_j \geq R_i + 1 - |\mathcal{V}^r \cup \{0\}| \cdot (1 - X_{ij}), \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r. \tag{14}$$

$$X_{ij}, P_{eij}, N_{kij} \in \{0,\ 1\}, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^r \cup \{0\},\ k \in \mathcal{V},\ e \in \mathcal{E}. \tag{15}$$

$$Y_{e0j}, V_{k0j} \in \{0,\ 1\}, \forall\, j \in \mathcal{V}^d,\ k \in \mathcal{V},\ e \in \mathcal{E}. \tag{16}$$

$$Z_i^r, R_i, Z_j^d \geq 0, \forall\, i \in \mathcal{V}^r \cup \{0\},\ j \in \mathcal{V}^d. \tag{17}$$

A função objetivo (1) minimiza a soma ponderada do tempo que as áreas afetadas ficam inacessíveis para a distribuição de suprimentos. A prioridade é dada pela quantidade de demanda em cada área afetada. As restrições (2) e (3) garantem que cada nó danificado seja reparado apenas uma vez. As restrições (4), (5) e (6) garantem a conservação do fluxo no percurso da equipe de trabalho entre os nós i e j, i.e., se existe um caminho entre os nós i e j que precisam ser reparados ($X_{ij} = 1$), então existe um arco saindo de i (restrição (4)) e um arco entrando em j (restrição (5)) nesse caminho. Além disso, devem existir dois arcos (um arco entrando e um arco saindo) em cada nó k (restrição (6)) dentro do caminho de i para j ($N_{kij} = 1$). De forma similar, as restrições (7), (8) e (9) garantem a conservação do fluxo no caminho de distribuição de produtos entre o depósito ($i = 0$) e o nó de demanda j. A restrição (10) evita que o caminho para a distribuição de produtos até a área afetada $i \in \mathcal{V}^d$ seja maior do que a distância máxima permitida l_i. A restrição (11) define o tempo no qual um nó $j \in \mathcal{V}^r$ fica completamente reparado. A restrição (12) evita que um nó k que ainda não foi reparado seja considerado no caminho da equipe de trabalho entre dois nós i e j. A restrição (13) determina o tempo em que os pontos de demanda $i \in \mathcal{V}^d$ ficam acessíveis a partir do depósito. A restrição (14) evita a formação de sub-tours no roteamento da equipe de trabalho, i.e., para a equipe de trabalho existe uma única rota que deve estar conectada ao depósitos inicial. As restrições (15)-(17) definem o domínio das variáveis.

Método de solução

Nessa seção são apresentados os métodos de solução propostos para resolver o problema. Foi proposta uma heurística construtiva e duas heurísticas de busca local para melhorar o resultado da

heurística construtiva. Os métodos se baseiam na decomposição do problema em três subproblemas.

O primeiro deles, chamado de SP1, é semelhante ao Problema do Caixeiro Viajante (PCV) e define a sequência de programação das equipes de trabalho. O subproblema SP2 consiste em determinar o caminho mínimo entre nós consecutivos na sequência de reparação das equipes de trabalho que define a rota que deve seguir a equipe de trabalho dada uma programação do subproblema SP1. O subproblema SP3 também busca determinar um caminho mínimo, porém entre o depósito e as áreas afetadas.

Heurística construtiva

O subproblema SP1 é considerado como um PCV em que as cidades a serem visitadas representam os nós danificados. Uma heurística de construção simples para esse problema é uma heurística gulosa que a cada iteração faz uma escolha localmente ótima com o objetivo de possivelmente encontrar soluções globalmente ótimas ou perto do ótimo global. O algoritmo começa no depósito, seleciona o nó factível j não visitado com o menor tempo t_{ij} como o seguinte nó a ser considerado na programação. Um nó $j \in \mathcal{V}^r$ é considerado factível se existe um caminho entre o nó atual i e o nó j que não usa arcos danificados não reparados. O subproblema de caminho mínimo (SP2) entre o nó danificado i e o nó danificado j é resolvido utilizando o algoritmo de *Dijkstra* [Dijkstra, 1959]. Apenas nós factíveis podem ser selecionados na programação e, portanto, são sempre geradas soluções factíveis para o problema original. Uma vez que é obtida a sequência das equipes de trabalhos e os caminhos mínimos entre cada par de nós danificados consecutivos na sequência, devem-se resolver subproblemas de caminho mínimo (SP3) entre o depósito e cada uma das áreas afetadas utilizando o algoritmo de *Dijkstra*.

A heurística construtiva é apresentada no Algoritmo 1. Como saída de algoritmo é definida a programação que deve seguir a equipe de trabalho $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$ e os caminhos mínimos entre cada par de nós danificados consecutivos na sequência K e entre o depósito e as áreas afetadas. O termo v_i indica o nó danificado na posição i na programação de reparação e $t_{v_i j}$ indica o tempo do caminho mínimo para ir do nó na posição i até o nó j.

Algoritmo 1 Heurística construtiva.

`Input`:
Grafo $G = (\mathcal{V}, \mathcal{E})$;
`Output`:
Programação $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$ da equipe de trabalho;
Caminhos mínimos entre cada par de nós danificados $v_{(i-1)}$ e v_i na sequência K.
Caminhos mínimos entre o depósito e as áreas afetadas $i \in \mathcal{V}^d$.

1: `Inicialização`:
2: Selecionar o nó 0 como o primeiro nó da programação K ($v_0 = 0$);
3: **para** $i = 1$ **to** $|\mathcal{V}^r|$ **faça**
4: Encontrar o caminho mínimo (subproblema SP2) entre o nó $v_{(i-1)}$ e os nós $j \in \mathcal{V}^r$ ainda não considerados na programação;
5: Selecionar o nó factível não visitado $j \in \mathcal{V}^r$ com o menor tempo $t_{v_{(i-1)}j}$;
6: $v_i = j$;
7: **fim para**
8: Encontrar o caminho mínimo (subproblema SP3) entre o depósito e cada um dos nós $i \in \mathcal{V}^d$;

Heurísticas de busca local

A partir da solução da heurística construtiva, dois operadores de busca local são implementados para melhorar a solução inicial K. O primeiro dos operadores é uma troca de posições (*swap*) entre dois nós danificados na sequência K. O segundo operador é uma troca de pares de arcos (*2opt*) que consiste em remover dois arcos e substituir os mesmos com dois arcos diferentes que reconectem os fragmentos criados. Seja W_K^n o conjunto de todos as soluções na vizinhança de K obtidas a partir de aplicar o operador n na solução (programação) K, em que $n \in \{swap, 2opt\}$.

Seja Θ^K o custo da programação K. Seja $\widehat{K}_i$ o i-ésimo elemento do conjunto W_K^n. O algoritmo de busca local baseado nos dois operadores (*swap* e *2-opt*) é mostrado no Algoritmo 2. Como entrada do algoritmo é utilizada a solução da heurística construtiva. Como saída do algoritmo é obtida uma solução localmente ótima. O subproblema SP2 de caminho mínimo entre cada par de nós consecutivos na programação de reparação K é resolvido para verificar se a programação gerada a partir da aplicação do operador n é factível. Uma solução é factível se para todo par $v_{(i-1)}$-v_i na programação K existem caminhos mínimos sem a necessidade de usar nós danificados que ainda não foram reparados. Se a programação K é factível, as decisões de caminho mínimo obtidas a partir do problema SP2 são fixadas e o subproblema SP3 de caminho mínimo entre o depósito e as áreas afetadas é resolvido para calcular o custo da programação K. Quando é encontrada uma solução melhor do que a solução atual, o processo de busca local começa novamente a partir da solução encontrada (*first improvement strategy*)[Hansen e Mladenović, 2006].

Algoritmo 2 Algoritmo de busca local aplicando o operador $n \in \{swap, 2opt\}$.

`Input:`
Programação inicial $K = (v_0, v_1, ..., v_j, ..., v_{|\mathcal{V}^r|})$;
Custo Θ^K da programação K;
`Output:`
Programação $K^* = (v_0^*, v_1^*, ..., v_j^*, ..., v_{|\mathcal{V}^r|}^*)$;

`Inicialização`:
improvement = 1;
$K^* = K$;
enquanto improvement $= 1$ **faça**
    i = 1;
    improvement = 0;
    Determinar conjunto W_K^n;
    enquanto $i \leq |W_K^n|$ **faça**
        Obter $\widehat{K}_i$;
        Resolver subproblema SP1 (caminhos mínimos) para a programação $\widehat{K}_i$;
        se Programação $\widehat{K}_i$ é factível **então**
            Calcular custo $\Theta^{\widehat{K}_i}$ da programação $\widehat{K}_i$ usando subproblema SP2;
            se $\Theta^{\widehat{K}_i} < \Theta^K$ **então**
                $\Theta^K = \Theta^{\widehat{K}_i}$;
                $i = |W_K^n| + 1$;
                $K = \widehat{K}_i$;
                $K^* = K$;
                Determinar novo conjunto W_K^n;
                improvement = 1;
            senão
                $i = i + 1$;
            fim se
        senão
            $i = i + 1$;
        fim se
    fim enquanto
fim enquanto

Resultados computacionais

O modelo apresentado na Seção 2 e os métodos heurísticos propostos na Seção 3 foram programados em linguagem C++. O modelo foi resolvido com o *solver* CPLEX 12.6 em um computador com 16 GB de memória RAM e processador Intel Core i7. Foi estabelecido como critério de parada um limite de tempo de 3600 segundos. Os experimentos computacionais foram realizados utilizando um grupo de instâncias da literatura [Maya Duque et al., 2016]. A Tabela 1

mostra o conjunto de instâncias consideradas. O parâmetro α define a porcentagem de arcos da rede que estão danificados. O parâmetro β define a porcentagem pela qual um caminho conetando o depósito com as áreas afetadas pode ser maior do que o caminho mínimo em situação pré-desastre. Combinando os diferentes valores de α e β são geradas várias instâncias a partir da mesma rede, cujo total é apresentado na última coluna da tabela.

Tabela 1: Conjunto de instâncias consideradas.

Rede	Nós de demanda	Valor de α (%)	Valor de β (%)	Instâncias Totais
A	25	5, 10, 25, 30, 50	5, 10, 25, 50	20
B	25	5, 10, 25, 30, 50	5, 10, 25, 50	20
C	25	5, 10, 25, 30, 50	5, 10, 25, 50	20
D	30	5, 10, 25, 30, 50	5, 10, 25, 50	20
E	30	5, 10, 25, 30, 50	5, 10, 25, 50	20
F	30	5, 10, 25, 30, 50	5, 10, 25, 50	20
G	35	5, 10, 25, 30, 50	5, 10, 25, 50	20
H	35	5, 10, 25, 30, 50	5, 10, 25, 50	20
I	35	5, 10, 25, 30, 50	5, 10, 25, 50	20
1	20	5, 10, 25, 30, 50	5, 10, 25, 50	20
2	20	5, 10, 25, 30, 50	5, 10, 25, 50	20
3	20	5, 10, 25, 30, 50	5, 10, 25, 50	20
4	40	5, 10, 25, 30, 50	5, 10, 25, 50	20
5	40	5, 10, 25, 30, 50	5, 10, 25, 50	20
6	40	5, 10, 25, 30, 50	5, 10, 25, 50	20
Total				300

Foram testadas quatro heurísticas a partir dos operadores de busca local definidos na Seção 3.2. As heurísticas H1 e H2 aplicam respectivamente os operadores *2-opt* e *swap* a partir da solução inicial da heurística construtiva. A heurística H3 aplica o operador *2-opt* a partir da heurística construtiva e então o operador *swap* a partir da solução obtida com o operador *2-opt*. Por outro lado, a heurística H4 aplica o operador *swap* a partir da heurística construtiva e, em seguida, o operador *2-opt* a partir da solução obtida com o operador *swap*. A heurística H0 é a heurística construtiva apresentada na Seção 3.1. A Tabela 2 apresenta a média dos resultados das heurísticas para as instâncias consideradas. O carácter "–" indica que o modelo não encontrou solução factível para pelo menos uma das instâncias da rede. Evidentemente, com as heurísticas propostas foram obtidas soluções factíveis para todas as instâncias testadas.

Como esperado, a heurística construtiva foi a mais rápida das heurísticas propostas, porém, foi a que apresentou as piores soluções. As heurísticas H1 e H2, que usam apenas um operador de busca local, apresentaram piores resultados do que aquelas heurísticas que usuram os dois operadores, H3 e H4. Quando comparados os resultados das heurísticas H1 e H2 é possível ver que o operador de troca de posição de dois nós danificados (*swap*) apresentou um melhor resultado do que o operador de troca de remoção de arcos (*2-opt*). No caso da heurísticas usando os dois operadores, H3 e H4, a melhor foi aquela que explora inicialmente a vizinhança das soluções baseada numa busca *2-opt* e depois, a partir da solução obtida, explora a nova vizinhança através do operador *swap*. Em média, a melhor estratégia foi H3, com uma solução média 196% menor do que a heurística construtiva.

Com o modelo matemático foram obtidas soluções factíveis para todas as instâncias da rede B, C, 1, 2 e 3. Quando comparadas as soluções do modelo para essas instâncias com as soluções da heurística H3, é possível observar que as soluções do modelo são muito próximas das soluções da heurística. De fato, a heurística obteve soluções em média 0,34% maiores do que as soluções do modelo para as instâncias da rede B e soluções 4,66%, 0,21%, 1,4% e 4,81% menores do que as soluções do modelo para as instâncias das redes C, 1, 2 e 3, respectivamente.

Tabela 2: Resultados médios obtidos com os métodos de solução propostos.

	Modelo MIP		H0		H1		H2		H3		H4	
Instâncias	Solução	Tempo	Solução	Tempo	Solução	Tempo	Solução	Tempo	Solução	Tempo	Solução	Tempo
A	–	738.7	25,233	0.001	9,775	0.06	9,745	0.09	9,745	0.07	9,745	0.10
B	34,565	1,115.9	61,324	0.001	35,977	0.04	34,685	0.04	34,685	0.05	34,685	0.04
C	52,988	1,204.6	86,897	0.001	51,660	0.09	49,864	0.06	50,630	0.11	49,864	0.06
D	–	1,571.6	73,992	0.005	18,133	1.68	18,410	3.04	18,133	1.60	18,179	2.43
E	–	2,161.1	67,097	0.006	24,781	3.61	23,147	29.00	23,133	25.35	23,144	29.78
F	–	1,980.6	77,103	0.005	22,226	1.02	21,726	2.35	21,388	1.43	21,726	2.53
G	–	2,162.1	158,990	0.015	36,532	5.55	36,532	11.29	36,532	6.86	36,532	12.30
H	–	2,166.4	210,934	0.013	30,852	7.23	26,073	8.89	26,049	8.15	26,073	11.05
I	–	2,173.4	135,703	0.013	35,443	12.04	33,953	15.39	33,953	11.86	33,953	15.49
1	48,945	1,175.0	62,628	0.001	49,380	0.01	48,849	0.04	48,843	0.03	48,849	0.04
2	39,176	1,465.7	53,647	0.000	38,716	0.08	39,095	0.08	38,634	0.10	39,095	0.09
3	29,400	577.3	94,089	0.000	30,939	0.04	28,607	0.03	28,050	0.05	28,607	0.04
4	–	2,466.2	75,474	0.031	23,759	23.75	23,528	30.87	23,706	28.09	23,528	34.90
5	–	2,573.3	179,029	0.028	81,374	56.30	82,629	168.42	81,114	60.35	81,075	172.59
6	–	2,724.7	225,997	0.023	60,356	111.37	60,454	134.62	60,354	125.97	60,268	156.60
Média	–	1,750.4	105,876	0.010	36,660	14.86	35,820	26.95	35,663	18.00	35,688	29.20

Em relação ao tempo computacional, com a heurística H4 foram obtidas em média soluções 97 vezes mais rápido do que com o modelo MIP. O gap médio do modelo foi 11,4%, 13,2%, 12,2%, 15,9% e 6,1% para as instâncias da rede B, C, 1, 2 e 3, respectivamente.

Considerações finais

Nesse artigo, foram propostos algoritmos heurísticos para resolver o problema de programação e roteamento de equipes de trabalho na restauração de redes afetadas por eventos extremos. Os algoritmos propostos foram baseados na decomposição do problema em subproblemas mais simples e na aplicação de operadores clássicos de busca local. Adicionalmente, O algoritmo *Dijkstra* foi utilizado para auxiliar a resolução de subproblemas de caminho mínimo. Os resultados mostram que os algoritmos heurísticos encontram soluções factíveis para todas as instâncias, as quais possuem qualidade melhor do que as soluções obtidas pela formulação matemática.

Os algoritmos heurísticos propostos encontram boas soluções de forma rápida, que podem ser utilizadas como soluções iniciais para outros métodos de solução mais sofisticados. Assim, uma proposta de trabalho futuro seria o desenvolvimento de meta-heurísticas, que são capazes de coordenar a aplicação sucessiva de buscas locais de modo a fugir de ótimos locais e, assim, obter soluções de melhor qualidade. Além disso, a decomposição do problema em sub-problemas mais simples pode ser utilizada também para o desenvolvimento de outro tipo de métodos, como decomposição de *Benders* ou *Dantzig-Wolfe*.

Agradecimentos

O primeiro autor agradece à FAPESP (processo 2016/15966-6) pelo apoio financeiro.

Referências

Berktaş, N., Kara, B. Y., e Karasan, O. E. (2016). Solution methodologies for debris removal in disaster response. *EURO Journal on Computational Optimization*, p. 1–43. ISSN 2192-4406.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

EM-DAT (2016). The international disaster database. URL `http://www.emdat.be/advanced_search/index.html`. Acessado em 01/12/2016.

Hansen, P. e Mladenović, N. (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802 – 817. ISSN 0166-218X.

Maya Duque, P. A., Dolinskaya, I. S., e Sörensenrensen, K. (2016). Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, 248(1):272 – 285. ISSN 0377-2217.

Sahin, H., Kara, B. Y., e Karasan, O. E. (2016). Debris removal during disaster response: A case for Turkey. *Socio-Economic Planning Sciences*, 53:49–59. ISSN 00380121.

Salman, F. S. e Yucel, E. (2015). Emergency facility location under random network damage: Insights from the Istanbul case. *Computers and Operations Research*, 62:266–281. ISSN 03050548.

Tzeng, G.-H., Chen, Y.-W., e Lin, C.-Y. (2000). Fuzzy multi-objective reconstruction plan for post-earthquake road-network by genetic algorithm. In *Research and Practice in Multiple Criteria Decision Making*, volume 487, p. 510–528. Springer Berlin Heidelberg.

Xu, B. e Song, Y. (2015). An Ant Colony-based Heuristic Algorithm for Joint Scheduling of Post-earthquake Road Repair and Relief Distribution. *Telecommunication Computing Electronics and Control*, 13(2):632. ISSN 2302-9293.

Yan, S. e Shih, Y.-L. (2009). Optimal scheduling of emergency roadway repair and subsequent relief distribution. *Computers & Operations Research*, 36(6):2049–2065. ISSN 03050548.