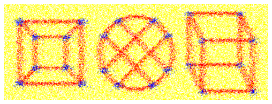
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 21, no. 4, pp. 417–432 (2017)
DOI: 10.7155/jgaa.00423

Incremental Network Design with Minimum Spanning Trees

Konrad Engel[1] *Thomas Kalinowski*[2] *Martin W.P. Savelsbergh*[3]

[1]Institut für Mathematik
Universität Rostock, 18051 Rostock, Germany
[2]School of Mathematical and Physical Sciences
University of Newcastle, Callaghan, NSW 2308, Australia
[3]School of Industrial & Systems Engineering
Georgia Institute of Technology, Atlanta, U.S.A.

Abstract

Given an edge-weighted graph $G = (V, E)$ and a set $E_0 \subset E$, the incremental network design problem with minimum spanning trees asks for a sequence of edges $e'_1, \ldots, e'_T \in E \setminus E_0$ minimizing $\sum_{t=1}^{T} w(X_t)$ where $w(X_t)$ is the weight of a minimum spanning tree X_t for the subgraph $(V, E_0 \cup \{e'_1, \ldots, e'_t\})$ and $T = |E \setminus E_0|$. We prove that this problem can be solved by a greedy algorithm.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
June 2015	November 2016	January 2017	January 2017	January 2017
		Published: February 2017		
	Article type: Regular paper		Communicated by: J. S. B. Mitchell	

The second and the third author were supported by the Australian Research Council and Aurizon under the grant LP140101000.

E-mail addresses: konrad.engel@uni-rostock.de (Konrad Engel) thomas.kalinowski@newcastle.edu.au (Thomas Kalinowski) martin.savelsbergh@isye.gatech.edu (Martin W.P. Savelsbergh)

1 Introduction

Network planning involves two stages. First, the structure of the network needs to be decided. Second, the construction of the network needs to be scheduled. The first stage, the network design stage, has received considerable attention in the operations research literature (see the survey papers [13, 14] and the references therein). The second stage, the network construction stage, has received less attention. However, because the construction of a network often stretches over a long period of time, the sequence in which the network is constructed is important as it defines when specific parts of the network become operational. It may even be beneficial to construct temporary links, i.e., links that are not part of the ultimate network, in order for parts of the network to become operational.

Recently, there has been increased interest in problems that integrate a scheduling component into network design problems, for instance motivated by the restoration of infrastructure networks after disruptions [1, 2, 3, 6, 15, 18]. A classification of such *integrated network design and scheduling* problems, with respect to underlying scheduling environments and network performance measures was presented in [16] along with complexity results and heuristic algorithms.

One of the most basic variants of the framework from [16] was studied under the name *incremental network design problem* in [4], in order to gain insights into the trade-offs between construction cost and operational benefit. More specifically, an incremental network design problem can be associated with any network optimization problem P, e.g., finding a shortest path, finding a maximum flow, etc. In the most basic version, in addition to the network optimization problem P, an instance is given by a network $G = (V, E)$ with vertex set V and edge set E and an existing edge set E_0. The edge set $E \setminus E_0$ is referred to as the potential edge set and its cardinality $T = |E \setminus E_0|$ as the planning horizon. Let $\varphi_P(G)$ denote the value of an optimal solution to network optimization problem P on network G. We are seeking a sequence $E_0 \subset E_1 \subset \cdots \subset E_T = E$ with $|E_i \setminus E_{i-1}| = 1$ giving rise to networks $G_0, G_1, \ldots, G_T = G$, such that $\sum_{t=1}^{T} \varphi_P(G_t)$ is minimum (assuming that P is a minimization problem). That is, in the basic version, a single edge can be built in each period of the planning horizon and we are seeking to minimize the operational costs over the planning horizon.

This setting should be considered as a first purely mathematical step towards real-world applications. In more elaborate versions, a construction cost may be associated with building a potential edge and a budget may be available in each period, and the objective is to minimize the operational costs over the planning horizon subject to the constraint that the construction cost of the set of potential edges built in a period does not exceed the budget in that period.

Two natural heuristics for incremental network design problems, **quickest-improvement** and **quickest-to-ultimate**, are also of interest. Quickest-improvement always seeks to improve the value of the solution to the network optimization as quickly as possible, i.e., by adding as few potential edges to the network as possible. A description of quickest-improvement can be found in Algorithm 1.

Algorithm 1 quickest-improvement

$i \leftarrow 0$; $E' \leftarrow E_0$
while $\varphi_P(G_{E'}) > \varphi_P(G_E)$ **do**
  $k \leftarrow \min\{|E''| \quad : \quad E'' \subseteq E \setminus E',\ \varphi_P(G_{E'}) - \varphi_P(G_{E' \cup E''}) > 0\}$
  $i \leftarrow i + 1$
  $E_i \leftarrow \arg\max\{\varphi_P(G_{E'}) - \varphi_P(G_{E' \cup E''}) \ : \ E'' \subseteq E \setminus E',\ |E''| = k\}$
  $E' \leftarrow E' \cup E_i$
return $(E_1, \ldots, E_i, E \setminus \bigcup_{j=0}^{i} E_j)$

Quickest-to-ultimate first finds an optimal solution to the network optimization on the complete network, referred to as an *ultimate* solution, and then always seeks to improve the value of the solution to the network optimization as quickly as possible, but choosing only potential edges that are part of the ultimate solution. A description of quickest-to-ultimate can be found in Algorithm 2.

Algorithm 2 quickest-to-ultimate

Let $\overline{E} \subseteq E$ be a set of minimum cardinality such that $\varphi_P\left(\overline{E}\right) = \varphi_P(E)$
$i \leftarrow 0$; $E' \leftarrow E_0$
while $\varphi_P(G_{E'}) > \varphi_P(G_E)$ **do**
  $k \leftarrow \min\left\{|E''| \quad : \quad E'' \subseteq \overline{E} \setminus E',\ \varphi_P(G_{E'}) - \varphi_P(G_{E' \cup E''}) > 0\right\}$
  $i \leftarrow i + 1$
  $E_i \leftarrow \arg\max\left\{\varphi_P(G_{E'}) - \varphi_P(G_{E' \cup E''}) \ : \ E'' \subseteq \overline{E} \setminus E',\ |E''| = k\right\}$
  $E' \leftarrow E' \cup E_i$
return $(E_1, \ldots, E_i, E \setminus \bigcup_{j=0}^{i} E_j)$

Incremental network design problems have been studied for the s-t shortest path problem [4] and for the maximum flow problem [12]. In both cases, it was found that even the basic version of the incremental network design problem is NP-complete. For the natural heuristics described above it has been shown that, for the shortest path problem, neither yields a constant factor approximation algorithm, but that for the maximum flow problem with the additional restriction that all arcs have unit capacity, quickest-to-ultimate yields a 2-approximation algorithm and quickest-improvement yields a 3/2-approximation algorithm.

These results have raised two questions: (1) Does there exist a network optimization problem for which the incremental design problem is polynomially solvable? (2) Does there exist a network optimization problem for which either quickest-to-ultimate or quickest-improvement solves the incremental design problem optimally?

In this paper, we answer both questions in the affirmative. We show that the

basic version of the incremental network design problem with minimum spanning trees is solved by both quickest-improvement and quickest-to-ultimate. Note that in a slightly more general setting the network design over time for minimum spanning trees is NP-complete. For instance, in the variant considered in [16], the construction of an edge can take multiple time periods and the objective is to minimize a weighted sum of of the weights of minimum spanning trees. This problem is proved to be NP-complete by a reduction from PARTITION.

The *incremental network design problem with minimum spanning trees* (IND-MST) is defined as follows. For a given graph $G = (V, E)$, a weight function $w : E \to \mathbb{R}$, and a set of existing edges E_0, such that the subgraph $G = (V, E_0)$ is connected, find a sequence $X_0, X_1, \ldots, X_T$ of spanning trees which minimizes the sum of the weights $w(X_0) + \cdots + w(X_T)$ subject to the condition that $X_0 \subseteq E_0$ and $|X_i \cap (E \setminus (E_0 \cup X_1 \cup \cdots \cup X_{i-1}))| \leqslant 1$ for $1 \leqslant i \leqslant T$, i.e., at most one edge from $E \setminus E_0$ might be added in each step. This has some similarity with the problem of maintaining a dynamic minimum spanning tree while the network data changes [7, 8, 9, 10, 20]. In contrast to these dynamic minimum spanning tree problems, in our setting the network changes are not given as input, but are part of the decisions to be made. We will show that IND-MST can be solved by a greedy algorithm. This is a consequence of the corresponding result for the incremental matroid design problem with minimum weight matroid bases, which is stated in Section 2 and proved in Section 3.

2 Incremental matroid design

Let $M = (E, \mathcal{I})$ be a matroid of rank r, where E is the ground set, and $\mathcal{I} \subseteq 2^E$ is the collection of independent sets. We follow the notation of Schrijver [17]: the rank of a matroid M is denoted by $\mathrm{rk}(M)$, minimal dependent sets are called *circuits*, for $A \subset E$ and $e \in E$ we write $A + e = A \cup \{e\}$ and $A - e = A \setminus \{e\}$, and we denote the closure of a set $A \subseteq E$ by $\mathrm{span}(A)$:

$$\mathrm{span}(A) = \{e \in E \ : \ \mathrm{rk}(A + e) = \mathrm{rk}(A)\}.$$

For a positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. An important tool in our arguments is the following *strong exchange property* which was first proved by Brualdi [5].

Strong exchange property. If X and Y are bases of a matroid M and $e \in X$, then there exists an element $e' \in Y$ such that $X - e + e'$ and $Y - e' + e$ are bases of M.

As additional input, we are given a weight function $w : E \to \mathbb{R}$ and a subset $E_0 \subset E$ such that E_0 contains a basis of M. The weight function w can be naturally extended to the power set of E by setting $w(X) = \sum_{e \in X} w(e)$ for all $X \subseteq E$. We define a function $f : 2^{E \setminus E_0} \to \mathbb{R}$ by

$$f(A) = \min\{w(X) \ : \ X \subseteq E_0 \cup A \text{ is a basis of } M\} \qquad \text{for } A \subseteq E \setminus E_0.$$

The *incremental matroid design problem with minimum weight bases* (IMD-MWB) problem for the time horizon $T = |E \setminus E_0|$ is the following optimization problem:

$$\min \left\{ \sum_{i=0}^{T} f(A_i) \ : \ A_0 = \emptyset, \ |A_i \setminus A_{i-1}| = 1 \text{ for } i \in [T] \right\}. \tag{1}$$

For a basis X, a pair $(e, e') \in X \times (E \setminus X)$ is called an *exchange pair for* X if $X - e + e'$ is another basis. It is called an *optimal* exchange pair if $w(e) - w(e')$ is maximum. Algorithm 3 is a natural greedy strategy for solving IMD-MWB, where the output defines the sets A_i in (1) via $A_i = \{e'_1, \ldots, e'_i\}$ for $i \leqslant k$ and $A_i = A_{i-1} + e'$ for arbitrary $e' \in E \setminus (E_0 \cup A_{i-1})$ for $k + 1 \leqslant i \leqslant T$. This corresponds to using quickest-improvement.

Algorithm 3 Greedy algorithm for the problem IMD-MWB

$k \leftarrow 0$
$w^* \leftarrow$ minimum weight of a basis of M
$X \leftarrow$ any minimum weight basis of the submatroid induced by E_0
(By assumption, X is also a basis for M)
while $w(X) > w^*$ **do**
  $k \leftarrow k + 1$
  $(e, e') \leftarrow$ an optimal exchange pair for X
  $X \leftarrow X - e + e'$
  $e'_k \leftarrow e'$
return $(e'_1, \ldots, e'_k)$

Our main result is a consequence of the following theorem.

Theorem 1 *Algorithm 3 finds an optimal solution for the problem IMD-MWB.*

If the second component e' of an exchange pair (e, e') for X belongs to $Y \setminus X$, where $Y \subseteq E$, then we call such a pair an *exchange pair for* (X, Y). Before proving Theorem 1 we observe that the search for an optimal exchange pair can be restricted to exchange pairs (e, e') for (X, Y), where Y is a *fixed* minimum weight basis of M. This corresponds to using quickest-to-ultimate and leads to Algorithm 4.

Corollary 1 *Algorithm 4 finds an optimal solution for the problem IMD-MWB.*

Proof: This follows from the claim that for any basis X of M and any minimum weight basis Y, there is an optimal exchange pair (e, e') for X with $e' \in Y$. Suppose the claim is false and let (e, e') be an optimal exchange pair for X. By the strong exchange property applied to the bases $X' = X - e + e'$ and Y and the element $e' \in X'$, there exists an $e'' \in Y$ such that $X' - e' + e'' = X - e + e''$ and $Y - e'' + e'$ are bases. Our assumption implies $w(e'') > w(e')$, while from the minimality of Y it follows that $w(e'') \leqslant w(e')$. □

Algorithm 4 Simplified greedy algorithm

$X \leftarrow$ any minimum weight basis of the submatroid induced by E_0
$Y \leftarrow$ any minimum weight basis of the matroid M
for $k = 1, \ldots, |Y \setminus X|$ **do**
  $(e, e') \leftarrow$ an optimal exchange pair for (X, Y)
  $X \leftarrow X - e + e'$
  $e'_k \leftarrow e'$
return $(e'_1, \ldots, e'_k)$

3 Proof of Theorem 1

The following lemma is stated in [8] for graphical matroids. The argument works in general, and in order to make our presentation self-contained we include the short proof.

Lemma 1 *Let $M = (E, \mathcal{I})$ be a matroid, $E_0 \subseteq E$, $A \subseteq E \setminus E_0$. In addition, let $M_0 = (E_0, \mathcal{I}_0)$ and $M_A = (E_0 \cup A, \mathcal{I}_A)$ be the matroids induced by E_0 and $E_0 \cup A$, respectively, i.e., $\mathcal{I}_0 = \{X \cap E_0 \ : \ X \in \mathcal{I}\}$ and $\mathcal{I}_A = \{X \cap (E_0 \cup A) \ : \ X \in \mathcal{I}\}$, and let X be a minimum weight basis for the matroid M_0. Then there exists a minimum weight basis Y of M_A such that $Y \subseteq X \cup A$.*

Lemma 1 is proved by iterating the next lemma which states that a single element exchange is sufficient in order to update the minimum weight basis after one potential element is added.

Lemma 2 *Let $A \subseteq E \setminus E_0$, and let X_A be a minimum weight basis for the matroid M_A induced by $E_0 \cup A$. Then, for every $e \in E \setminus (E_0 \cup A)$, the set $X_A + e - e'$ is a minimum weight basis for the matroid M_{A+e} induced by $E_0 \cup A + e$, where e' is an element of maximum weight in the circuit of $X_A + e$.*

Proof: Suppose the statement is false and let Y be a basis of M_{A+e} with $w(Y) < w(X_A) + w(e) - w(e')$. Then $e \in Y$, and by the strong exchange property, there exists $e'' \in X_A$ such that $Y + e'' - e$ and $X_A + e - e''$ are bases. The choice of e' implies $w(e'') \leqslant w(e')$, while minimality of X_A and our assumption on Y imply that

$$\begin{aligned} w(X_A) &\leqslant w(Y - e + e'') = w(Y) - w(e) + w(e'') \\ &< w(X_A) + w(e) - w(e') - w(e) + w(e'') = w(X_A) - w(e') + w(e''), \end{aligned}$$

hence $w(e'') > w(e')$, and this contradiction concludes the proof. □

Proof: [Proof of Lemma 1] Let $A = \{e_1, \ldots, e_k\}$, set $A_0 = \emptyset$ and $A_i = \{e_1, \ldots, e_i\}$ for $i = 1, \ldots, k$, and apply Lemma 2 with $A = A_i$, $e = e_{i+1}$ for $i = 0, \ldots, k-1$. □

For $t = 0, 1, \ldots, T$, let

$$F_t = \min\{f(A) \ : \ A \subseteq E \setminus E_0, \ |A| = t\}.$$

Note that $F_0 > F_1 > \cdots > F_t = F_{t+1} = \cdots = F_T$ for some $t \leqslant \min\{r, T\}$ and Algorithm 3 terminates with $k \geqslant t$. Clearly, $F_0 + F_1 + \cdots + F_T$ is a lower bound for (1). The correctness of Algorithm 3 follows from the fact that it achieves this lower bound, which in turn is a consequence of the following extension property.

Lemma 3 *Let t be the unique index with $F_{t-1} > F_t = F_{t+1}$, let $k < t$, and let $A, B \subseteq E \setminus E_0$ satisfying the following conditions:*

1. *$|A| = k$ and $|B| = k + 1$,*
2. *A and B are optimal solutions for the minimization problems definining F_k and F_{k+1}, respectively, i.e., $f(A) = F_k$ and $f(B) = F_{k+1}$.*

Then there exists $e \in B \setminus A$ such that $f(A + e) = F_{k+1}$.

Proof: Let M_0, M_A and M_B denote the submatroids induced by E_0, $E_0 \cup A$ and $E_0 \cup B$, respectively. We have $\mathrm{rk}(M_0) = \mathrm{rk}(M_A) = \mathrm{rk}(M_B) = r$ because E_0 contains a basis of M. By the optimality of A and B and since $F_{k+1} < F_k$, the sets A and B are contained in every minimum weight basis of M_A and M_B, respectively, and by Lemma 1, there are minimum weight bases X, X_A and X_B for these submatroids with $X_A \setminus X = A$ and $X_B \setminus X = B$. We define a bipartite digraph $(\mathcal{U} \cup \mathcal{V}, \mathcal{A})$ with parts

$$\mathcal{U} = X_A \setminus \mathrm{span}((X_B \cap X) \cup A), \qquad \mathcal{V} = X_B \setminus \mathrm{span}((X_B \cap X) \cup A).$$

Note that $\mathrm{rk}((X_B \cap X) \cup A) \leqslant |(X_B \cap X) \cup A| = r - 1$, hence $\mathcal{U}, \mathcal{V} \neq \emptyset$. Also, $\mathcal{U} \subseteq X \subseteq E_0$ and $\mathcal{V} \subseteq B \subseteq E \setminus E_0$, hence $\mathcal{U} \cap \mathcal{V} = \emptyset$. The arc set is defined by

$$\begin{aligned}\mathcal{A} = \{(e, e') \in \mathcal{U} \times \mathcal{V} \ : \ & (e, e') \text{ is an exchange pair for } X_A\} \\ & \cup \{(e, e') \in \mathcal{V} \times \mathcal{U} \ : \ (e, e') \text{ is an exchange pair for } X_B\}.\end{aligned}$$

Let $e' \in \mathcal{V}$. Then $e' \in E \setminus (E_0 \cup A)$, hence $e' \notin X_A$ and $X_A + e'$ contains a circuit C. We claim that $C - e' \not\subseteq \mathrm{span}((X_B \cap X) \cup A)$, and this implies that there exists $e \in C \cap \mathcal{U}$, and consequently $(e, e') \in \mathcal{A}$. For the sake of contradiction, suppose the claim is false and $C - e' \subseteq \mathrm{span}((X_B \cap X) \cup A)$. From the fact that C is a circuit, it follows that

$$e' \in \mathrm{span}(C - e') \subseteq \mathrm{span}((X_B \cap X) \cup A)$$

which is a contradiction to $e' \in \mathcal{V}$. Similarly, if $e' \in \mathcal{U}$, then $e' \in X_A \setminus A \subseteq E_0$ and $e' \notin X_B \cap X$, which implies $e' \notin X_B$, hence there exists a circuit C in $X_B + e'$. As before, the assumption that $C - e'$ is contained in $\mathrm{span}((X_B \cap X) \cup A)$ leads to the contradiction $e' \in \mathrm{span}((X_B \cap X) \cup A)$. By this argument, for every $e' \in \mathcal{U}$ there exists an $e \in \mathcal{V}$ with $(e, e') \in \mathcal{A}$. We conclude that every node in the digraph $(\mathcal{U} \cup \mathcal{V}, \mathcal{A})$ has positive indegree, thus the digraph contains a

directed cycle, and this implies that there are $e', e'' \in \mathcal{U}$ and $e \in \mathcal{V}$ such that $(e', e) \in \mathcal{A}$, $(e, e'') \in \mathcal{A}$, and $w(e') \geqslant w(e'')$. From this we derive

$$f(A+e) \leqslant w(X_A + e - e') = f(A) + w(e) - w(e') \leqslant F_k + w(e) - w(e'')$$
$$\leqslant f(B-e)+w(e)-w(e'') \leqslant w(X_B - e + e'') + w(e) - w(e'') = w(X_B) = F_{k+1}.$$

The converse inequality $f(A+e) \geqslant F_{k+1}$ is obvious and this concludes the proof. □

4 Run-time analysis

A rough upper bound for the run-time of a naive implementation of Algorithm 4 for the IND-MST problem on a graph with n vertices can be obtained as follows: The optimal exchange pair in each step of the for-loop can be found in time $O(n^2)$ by running through $O(n)$ candidates for e' and then by determining the best partner e for e' in linear time. Since $Y \setminus X$ has size $O(n)$ this gives in total a run-time of $O(n^3)$ which dominates the time needed to find the minimum spanning trees X and Y. In the following, we provide a more thorough estimate for the run-time.

In order to bound the time complexities of the problems IMD-MWB and IND-MST, we argue that the initial basis X, the ultimate basis Y and the list of exchange pairs $\mathcal{E}$ can be determined simultaneously. The idea is to consider the elements of E in order of nondecreasing weights and to construct and maintain three independent sets X, Y and Z using the following update rules:

1. An element $e \in E$ is added to X if and only if $e \in E_0$ and the addition of e does not create a circuit in X. Hence X is an initial minimum weight basis when the algorithm terminates.

2. An element $e \in E$ is added to Y if and only if the addition of e does not create a circuit in Y. Hence Y is an ultimate minimum weight basis when the algorithm terminates.

3. An element $e \in E$ is added to Z if and only if it is added to X or Y. We will prove the following fact: If an edge added to Y does not create a circuit in Y then it does not create a circuit in Z. By this fact, an edge e added to Z can create a circuit C in Z only if it has been added to X. This implies $e \in E_0$ and C must contain an element of $E \setminus E_0$ (otherwise e would have created a circuit in X). In this situation, a maximum weight element e' of $C \setminus E_0$ is removed from Z to preserve the independence of Z. The pair (e, e') is added to the set $\mathcal{E}$ of exchange pairs.

To finish up, the set $\mathcal{E}$ of exchange pairs (e, e') is ordered such that $w(e) - w(e')$ is nonincreasing, and ties are broken in favor of the pair that was added to $\mathcal{E}$ last. More precisely, we index $\mathcal{E} = \{(e_1, e'_1), \ldots, (e_k, e'_k)\}$, such that for every $i \in \{1, \ldots, k-1\}$, we have either $w(e_i) - w(e'_i) > w(e_{i+1}) - w(e'_{i+1})$, or $w(e_{i+1}) - w(e'_{i+1}) = w(e_i) - w(e'_i)$ and the pair (e_i, e'_i) is added to $\mathcal{E}$ after (e_{i+1}, e'_{i+1}).

A formal description can be found in Algorithm 5. For the remainder of the section, let k be the size of the set $\mathcal{E}$ returned by the algorithm.

Algorithm 5 Efficient solution of the problem IMD-MWB

$X \leftarrow \emptyset$; $Y \leftarrow \emptyset$; $Z \leftarrow \emptyset$ // initialize three independent sets
$\mathcal{E} \leftarrow \emptyset$ // initialize set of exchange pairs
for $e \in E$ **do** // in nondecreasing order of weight
  if $e \in E_0$ **then**
    if $X + e \in \mathcal{I}$ **then**
      $X \leftarrow X + e$; $Z \leftarrow Z + e$
      if Z contains a circuit C **then**
        $e' = \arg\max\{w(e'') \ : \ e'' \in C \setminus E_0\}$
        $Z \leftarrow Z - e'$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{(e, e')\}$
  if $Y + e \in \mathcal{I}$ **then**
      $Y \leftarrow Y + e$; $Z \leftarrow Z + e$
sort $\mathcal{E}$
return X and $\mathcal{E} = \{(e_1, e'_1), \ldots, (e_k, e'_k)\}$

In order to show the correctness of Algorithm 5, we introduce some additional notation. Let $X_0 = X$ and $X_i = X_{i-1} - e_i + e'_i$ for $(e_i, e'_i) \in \mathcal{E}$, $i = 1, 2, \ldots, k$. Let $m = |E|$. We say that e has *position* p, denoted by $\mathrm{pos}(e) = p$, if e is the element that is handled in the p-th iteration of the for-loop, $1 \leqslant p \leqslant m$. Note that $\mathrm{pos}(e) < \mathrm{pos}(e')$ implies $w(e) \leqslant w(e')$.

Let X^p, Y^p, Z^p denote the sets X, Y and Z after the p-th iteration of the for-loop has been completed, $0 \leqslant p \leqslant m$. It is obvious, that if $p \leqslant l$, then

$$X^p \subseteq X^l \qquad \text{and} \qquad Y^p \subseteq Y^l.$$

For $i = 1, \ldots, k$, we denote the circuit that leads to the deletion of e'_i from Z by C_i. In other words, if $\mathrm{pos}(e_i) = p$ then C_i is the unique circuit in $Z^{p-1} + e_i$ and we have $Z^p = Z^{p-1} + e_i - e'_i$. If e is any element of C_i, then $\mathrm{pos}(e) \leqslant \mathrm{pos}(e_i)$ and hence $w(e) \leqslant w(e_i)$. By the choice of e'_i we have

$$w(e) \leqslant w(e'_i) \text{ for all } e \in C_i \setminus E_0. \tag{2}$$

Let

$$H^p = \{e_i : i \in [k] \text{ and } \mathrm{pos}(e_i) \leqslant p\}, \text{ and}$$
$$H'^p = \{e'_i : i \in [k] \text{ and } \mathrm{pos}(e_i) \leqslant p\}.$$

That is, the sets H^p and H'^p contain the edges involved in exchanges occurring either before or when the edge in position p is examined. More precisely, we have

$$(X^p \times Y^p) \cap \mathcal{E} = \{(e_i, e'_i) : e_i \in H^p \text{ and } e'_i \in H'^p\}.$$

Lemma 4 *We have for* $0 \leqslant p \leqslant m$

1. $X^p \subseteq Z^p \subseteq X^p \cup Y^p$,
2. $\mathrm{span}(Y^p) = \mathrm{span}(Z^p)$,
3. $Y^p \setminus X^p \subseteq E \setminus E_0$,
4. $X^p, Y^p, Z^p \in \mathcal{I}$,
5. $X^p \setminus Y^p = H^p$,
6. $Y^p \setminus Z^p = H'^p$.

Proof: We proceed by induction on p. The case $p = 0$ is trivial. Now consider the step $p - 1 \to p$. From $X^{p-1} \subseteq Z^{p-1} \subseteq X^{p-1} \cup Y^{p-1}$ we obtain directly $X^p \subseteq Z^p \subseteq X^p \cup Y^p$ since an element is added to Z iff it is added to X or to Y and an element is deleted from Z only if this element does not belong to E_0, i.e., not to X. Now we prove the other assertions. Let e be the element with $\mathrm{pos}(e) = p$.

If $X^p = X^{p-1}$ and $Y^p = Y^{p-1}$ then also $Z^p = Z^{p-1}$, $H^p = H^{p-1}$ and $H'^p = H'^{p-1}$ and the assertion follows from the induction hypothesis. So there are three main cases:

Case 1. $X^p = X^{p-1} + e$ and $Y^p = Y^{p-1}$. Then $e \in \mathrm{span}(Y^{p-1})$. By the induction hypothesis, $e \in \mathrm{span}(Z^{p-1})$ and consequently $Z^p = (Z^{p-1} + e) - e'$ for some e' in the unique circuit of $Z^{p-1} + e$ where $e' \subseteq E \setminus E_0$ and $e' \neq e$. Obviously, $Z^p \in \mathcal{I}$ and $\mathrm{span}(Z^p) = \mathrm{span}(Z^{p-1} + e) = \mathrm{span}(Z^{p-1}) = \mathrm{span}(Y^{p-1}) = \mathrm{span}(Y^p)$. Obviously, X^p, $Y^p \in \mathcal{I}$ and $Y^p \setminus X^p \subseteq Y^{p-1} \setminus X^{p-1} \subseteq E \setminus E_0$. Finally, $X^p \setminus Y^p = (X^{p-1} + e) \setminus Y^{p-1} = (X^{p-1} \setminus Y^{p-1}) + e = H^{p-1} + e = H^p$ and $Y^p \setminus Z^p = Y^{p-1} \setminus ((Z^{p-1} + e) - e') = (Y^{p-1} \setminus Z^{p-1}) + e' = H'^{p-1} + e' = H'^p$.

Case 2. $X^p = X^{p-1}$ and $Y^p = Y^{p-1} + e$. Then $Z^p = Z^{p-1} + e$ and hence $\mathrm{span}(Y^p) = \mathrm{span}(Z^p)$. Obviously, X^p, $Y^p \in \mathcal{I}$ and $e \notin \mathrm{span}(Y^{p-1})$, i.e., $e \notin \mathrm{span}(Z^{p-1})$. Thus $Z^p \in \mathcal{I}$.

If $e \in E_0$ then $e \in \mathrm{span}(X^{p-1}) \subseteq \mathrm{span}(Z^{p-1}) = \mathrm{span}(Y^{p-1})$, which contradicts $Y^{p-1} + e \in \mathcal{I}$. Hence $e \notin E_0$. Then $Y^p \setminus X^p = (Y^{p-1} + e) \setminus X^{p-1} \subseteq E \setminus E_0$. Finally, $X^p \setminus Y^p = X^{p-1} \setminus (Y^{p-1} + e) = X^{p-1} \setminus Y^{p-1} = H^{p-1} = H^p$ and $Y^p \setminus Z^p = Y^{p-1} \setminus Z^{p-1} = H'^{p-1} = H'^p$.

Case 3. $X^p = X^{p-1} + e$ and $Y^p = Y^{p-1} + e$. Then X^p, $Y^p \in \mathcal{I}$ and $e \notin \mathrm{span}(Y^{p-1})$, i.e., $e \notin \mathrm{span}(Z^{p-1})$. Thus $Z^{p-1} + e \in \mathcal{I}$. This implies $Z^p = Z^{p-1} + e$ and consequently $\mathrm{span}(Y^p) = \mathrm{span}(Z^p)$ as well as $Y^p \setminus X^p = Y^{p-1} \setminus X^{p-1} \subseteq E \setminus E_0$. Finally, $X^p \setminus Y^p = X^{p-1} \setminus Y^{p-1} = H^{p-1} = H^p$ and $Y^p \setminus Z^p = Y^{p-1} \setminus Z^{p-1} = H'^{p-1} = H'^p$.

□

In the following, we denote the three independent sets that the algorithm terminates with by X, Y, and Z, i.e., $X = X^m$, $Y = Y^m$ and $Z = Z^m$.

Corollary 2 *We have*

1. $X = Z$,
2. $X \setminus Y = \{e_1, \ldots, e_k\}$,
3. $Y \setminus X = \{e'_1, \ldots, e'_k\}$.

Proof: By our general supposition that E_0 contains a basis, the set X is a basis for M. Since, by Lemma 4, Z is independent and $X \subseteq Z$, we have $X = Z$. Again, by Lemma 4, $X \setminus Y = X^m \setminus Y^m = H^m = \{e_1, \ldots, e_k\}$ and $Y \setminus X = Y \setminus Z = Y^m \setminus Z^m = H'^m = \{e'_1, \ldots, e'_k\}$. □

Corollary 3 *If* $\mathrm{pos}(e_i) = p$, $i \in [k]$, *then* $C_i \cap H'^p = \{e'_i\}$.

Proof: We have $C_i \subseteq Z^{p-1} + e_i = Z^p + e'_i$ and thus $C_i \cap H'^p = C_i \cap (Y^p \setminus Z^p) \subseteq (Z^p + e'_i) \cap (Y^p \setminus Z^p) = \{e'_i\}$. Clearly, $e'_i \in C_i \cap H'^p$. □

Lemma 5 *Let* C *be a circuit in* $X \cup Y$. *Let*

$$e^* = \arg\max\{\mathrm{pos}(e) : e \in C\}.$$

Then $e^* = e_l$ *for some* $l \in [k]$.

Proof: Let $\mathrm{pos}(e^*) = q$. Then $C - e^* \subseteq X^{q-1} \cup Y^{q-1}$. This implies $e^* \in \mathrm{span}(X^{q-1} \cup Y^{q-1})$ and from Lemma 4 we obtain $e^* \in \mathrm{span}(X^{q-1} \cup Z^{q-1}) = \mathrm{span}(Z^{q-1})$. Since $e^* \in X^q \cup Y^q$, but $e^* \notin X^{q-1} \cup Y^{q-1}$ we must have $X^q = X^{q-1} + e^*$ and $Z^q = Z^{q-1} + e^* - e'$ for some $e' \in E \setminus E_0$. Hence $e^* = e_l$ for some $l \in [k]$. □

Lemma 6 *We have* $C_i \subseteq X \cup \{e'_1, \ldots, e'_i\}$ *for every* $i \in [k]$.

Proof: Using Corollary 2, we obtain $C_i \subseteq X \cup Y = X \cup (Y \setminus X) = X \cup \{e'_1, \ldots, e'_k\}$. Assume that there is some $j > i$ such that $e'_j \in C_i$. Then $\mathrm{pos}(e_j) > \mathrm{pos}(e_i)$ and thus $w(e_j) \geqslant w(e_i)$. From (2) and e'_j in $C_i \setminus E_0$, we obtain $w(e'_i) \geqslant w(e'_j)$. Consequently, $w(e_j) - w(e'_j) \geqslant w(e_i) - w(e'_i)$, a contradiction to the ordering of $\mathcal{E}$. □

Lemma 7 *For* $i \in [k]$, *the pair* (e_i, e'_i) *is an exchange pair for* (X_{i-1}, Y).

Proof: Clearly, $e'_i \in Y \setminus (X \cup \{e'_1, \ldots, e'_{i-1}\}) \subseteq Y \setminus X_{i-1}$. We have to show that e_i lies in the unique circuit of $X_{i-1} + e'_i = (X \setminus \{e_1, \ldots, e_{i-1}\}) \cup \{e'_1, \ldots, e'_i\}$. An equivalent statement is that there is a circuit in $(X \setminus \{e_1, \ldots, e_{i-1}\}) \cup \{e'_1, \ldots, e'_i\}$ containing e_i. From Lemma 6, we know that there is at least a circuit in $X \cup \{e'_1, \ldots, e'_i\}$ containing e_i, namely C_i. For a circuit C, let

$$\mu^i_C = \max\{\mathrm{pos}(e_j) : j \in [i-1] \text{ and } e_j \in C\},$$

where the maximum extended over an empty set is defined to be $-\infty$.

Assume that there is no circuit in $(X\setminus\{e_1,\ldots,e_{i-1}\})\cup\{e'_1,\ldots,e'_i\}$ containing e_i. Then we choose a circuit C in $X\cup\{e'_1,\ldots,e'_i\}$ containing e_i such that μ^i_C is minimal. By our assumption, μ^i_C is finite and there exists an integer $l\in[i-1]$ such that $e_l\in C$ and $\mathrm{pos}(e_l)=\mu^i_C$. The circuit C_l also contains e_l and is contained in $X\cup\{e'_1,\ldots,e'_i\}$ in view of Lemma 6 and $l<i$. Note that $\mathrm{pos}(e)<\mathrm{pos}(e_l)$ for all $e\in C_l-e_l$. Now C and C_l are two distinct circuits with $e_l\in C\cap C_l$, and therefore there is also a circuit $\widetilde{C}\subseteq C\cup C_l-e_l$. Obviously, $\widetilde{C}\subseteq X\cup\{e'_1,\ldots,e'_i\}$ and $\mu^i_{\widetilde{C}}<\mu^i_C$, a contradiction to the choice of C. □

Lemma 8 *For $i\in[k]$, the pair (e_i,e'_i) is an optimal exchange pair for (X_{i-1},Y).*

Proof: Assume that (e_i,e'_i) is not optimal. Then there is a better exchange pair $(\hat{e},\hat{e}')$ for (X_{i-1},Y). We have $\hat{e}'\in Y\setminus X_{i-1}=(Y\setminus(X\cup\{e'_1,\ldots,e'_{i-1}\}))\cup(Y\cap\{e_1,\ldots,e_{i-1}\})$. From Corollary 2 it follows that $\hat{e}'=e'_j$ for some $j\in[k]$. Since $e'_1,\ldots,e'_{i-1}\notin Y\setminus X_{i-1}$,

$$j\geqslant i. \tag{3}$$

Let $\widehat{C}$ be the unique circuit in $X_{i-1}+e'_j$ (containing $\hat{e}$) and let $e^*=\arg\max\{\mathrm{pos}(e):e\in\widehat{C}\}$. Then $w(e^*)\geqslant w(\hat{e})$. By Lemma 5, $e^*=e_l$ for some $l\in[k]$. In particular, $e^*\neq e'_j$ and thus (e^*,e'_j) is also an exchange pair for (X_{i-1},Y). Since $e_1,\ldots,e_{i-1}\notin X_{i-1}$,

$$l\geqslant i. \tag{4}$$

We have (with $e^*=e_l$) $w(e_l)-w(e'_j)\geqslant w(\hat{e})-w(e'_j)$ and hence

$$w(e_l)-w(e'_j)>w(e_i)-w(e'_i). \tag{5}$$

By the ordering of $\mathcal{E}$ and by (4),

$$w(e_i)-w(e'_i)\geqslant w(e_l)-w(e'_l). \tag{6}$$

The inequalities (5) and (6) imply

$$w(e'_j)<w(e'_l). \tag{7}$$

Let $p=\mathrm{pos}(e_l)$. Then $\mathrm{pos}(e_j)<p$, because otherwise $w(e_j)\geqslant w(e_l)$ and hence, in view of (3) and the ordering of $\mathcal{E}$,

$$w(e_l)-w(e'_j)\leqslant w(e_j)-w(e'_j)\leqslant w(e_i)-w(e'_i),$$

a contradiction to (5). Moreover, $\widehat{C}\subseteq X^p\cup Y^p$ since e_l has maximal position in $\widehat{C}$. For a circuit C let

$$\alpha^p_C=\max\{w(e'_h):h\in[k],e'_h\in C\text{ and }\mathrm{pos}(e_h)\leqslant p\},$$
$$\nu_C=\min\{\mathrm{pos}(e_h):h\in[k]\text{ and }e'_h\in C\},$$

where the maximum (resp. minimum) extended over an empty set is defined to be $-\infty$ (resp. ∞). For $\widehat{C}$ these values are finite since $e'_j\in\widehat{C}$ and $\mathrm{pos}(e_j)<p$.

Hence there is an integer $g \in [k]$ such that $e'_g \in \widehat{C}$, $\mathrm{pos}(e_g) \leqslant p$ and $w(e'_g) = \alpha^p_{\widehat{C}}$. Note that

$$g \leqslant i-1 \text{ or } g = j \tag{8}$$

since $\widehat{C} \subseteq X \cup \{e'_1, \ldots, e'_{i-1}, e'_j\}$. We say that a circuit C is *p-majorized* by a number α if

$$w(e'_h) \leqslant \alpha \quad \text{for all } h \in [k] \text{ with } e'_h \in C \text{ and } \mathrm{pos}(e_h) \leqslant p.$$

Note that $\widehat{C}$ is p-majorized by $\alpha^p_{\widehat{C}}$. Now choose a circuit C^* in $X^p \cup Y^p$ that contains e_l, is p-majorized by $\alpha^p_{\widehat{C}}$ and has maximal ν-value.

Note that, in view of Lemma 4 (parts 1 and 6), $C^* \subseteq X^p \cup Y^p \subseteq Z^p \cup Y^p = Z^p \cup (Y^p \setminus Z^p) = Z^p \cup H'^p$. We have $\nu_{C^*} \leqslant p$ because otherwise $C^* \cap H'^p = \emptyset$ and C^* would be a circuit in Z^p, contradicting Lemma 4 (part 4).

Assume that $\nu_{C^*} < p$. Then choose $q \in [k]$ such that $e'_q \in C^*$ and $\mathrm{pos}(e_q) = \nu_{C^*}$ and consider the circuit C_q. We have $C_q \subseteq X^p \cup Y^p$ because of $\mathrm{pos}(e_q) < p$. Moreover, $w(e'_h) \leqslant w(e'_q)$ and $\mathrm{pos}(e_h) \geqslant \mathrm{pos}(e_q)$ for all $h \in [k]$ with $e'_h \in C_q$ by the choice of e'_q in C_q and Corollary 3. Since C^* is p-majorized by $\alpha^p_{\widehat{C}}$, in particular $w(e'_q) \leqslant \alpha^p_{\widehat{C}}$ and thus also C_q is p-majorized by $\alpha^p_{\widehat{C}}$. By definition, we have $C_q \subseteq Z^{p'-1} + e_q$ for $p' = \mathrm{pos}(e_q) < p = \mathrm{pos}(e_l)$, and therefore $e_l \notin C_q$. From $e'_q \in C^* \cap C_q$ and $e_l \in C^* \setminus C_q$ it follows that there is a circuit $\widetilde{C} \subseteq C^* \cup C_q - e'_q$ containing e_l. Clearly, $\widetilde{C} \subseteq X^p \cup Y^p$ and $\widetilde{C}$ is p-majorized by $\alpha^p_{\widehat{C}}$. Obviously, $\min\{\mathrm{pos}(e_h) : h \in [k] \text{ and } e'_h \in C_q \cup C^*\} = \mathrm{pos}(e_q)$. Thus $\nu_{\widetilde{C}} > \nu_{C^*}$, a contradiction to the choice of C^*.

Consequently, $\nu_{C^*} = p$. Since e_l is the (unique) element of position p, necessarily $e'_l \in C^*$. Because C^* is p-majorized by $\alpha^p_{\widehat{C}}$, in particular $w(e'_l) \leqslant \alpha^p_{\widehat{C}} = w(e'_g)$. From (7) we obtain $g \neq j$, hence by (8) and (4),

$$g \leqslant i-1 < l. \tag{9}$$

On the other hand, the relation $\mathrm{pos}(e_g) \leqslant \mathrm{pos}(e_l)$ implies $w(e_g) \leqslant w(e_l)$. Consequently,

$$w(e_l) - w(e'_l) \geqslant w(e_g) - w(e'_g),$$

and then the ordering of $\mathcal{E}$ implies $l \leqslant g$, which contradicts (9). Thus our first assumption was false, and the proof is complete. □

For the runtime analysis, we assume that Algorithm 5 gets as input the elements of E in nondecreasing order of weights, and that we have an algorithm which, for a given set $X \subseteq E$, decides if $X \in \mathcal{I}$. Let A denote an upper bound for the runtime of this independence test. In each iteration of the for-loop there are three independence tests, and at most one search for the element $e' \in Z$ to be removed. Since the final X has r elements, this latter step is necessary at most r times and can be done in time $O(A|Z|) = O(Ar)$, where $r = \mathrm{rk}(M)$ is the rank of the matroid: just call the independence test for each of the sets $Z - e'$ in decreasing order of $w(e')$ until you find an independent set. So the for-loop

terminates in time $O(A(|E| + r^2))$. The final sorting of $\mathcal{E}$ takes time $O(r \log r)$. To summarize, we have proved the following bound for the time complexity of the problem IMD-MWB.

Theorem 2 *For a matroid $M = (E, \mathcal{I})$ of rank r, where the elements of E are given in nondecreasing order of weights, the problem IMD-MWB can be solved in time $O(A(|E| + r^2))$, where A is a runtime bound for an algorithm that decides the independence of a set $X \subseteq E$.*

Finally, we consider the special case where M is a graphical matroid to solve our original problem IND-MST.

Theorem 3 *The problem IND-MST for a graph with n vertices and m edges can be solved in time*

$$O(m + n \log n).$$

Proof: Using Fibonacci heaps [11], minimum spanning trees X and Y for the graphs $G = (V, E)$ and $G_0 = (V, E_0)$, respectively, can be constructed in time $O(m + n \log n)$. We can then run Algorithm 5 on the graph $G = (V, X \cup Y)$ which has only $O(n)$ edges. Using dynamic trees [19] to represent Z, each of the exchange pairs (e, e') can be found in time $O(\log n)$: if $e = \{u, v\}$ is the edge that creates a circuit in Z then e' is an edge of maximum weight on the path between u and v in Z. □

Acknowledgements

We thank the anonymous referees for valuable comments that helped improve and clarify the presentation of the results.

References

[1] I. Averbakh. Emergency path restoration problems. *Discrete Optimization*, 9(1):58–64, 2012. doi:10.1016/j.disopt.2012.01.001.

[2] I. Averbakh and J. Pereira. The flowtime network construction problem. *IIE Transactions*, 44(8):681–694, 2012. doi:10.1080/0740817X.2011.636792.

[3] I. Averbakh and J. Pereira. Network construction problems with due dates. *European Journal of Operational Research*, 244:715–729, 2015. doi:10.1016/j.ejor.2015.02.014.

[4] M. Baxter, T. Elgindy, A. T. Ernst, T. Kalinowski, and M. W. Savelsbergh. Incremental network design with shortest paths. *European Journal of Operational Research*, 238(3):675–684, 2014. doi:10.1016/j.ejor.2014.04.018.

[5] R. A. Brualdi. Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society*, 1(2):161–167, 1969. doi:10.1017/S000497270004140X.

[6] B. Cavdaroglu, E. Hammel, J. Mitchell, T. Sharkey, and W. Wallace. Integrating restoration and scheduling decisions for disrupted interdependent infrastructure systems. *Annals of Operations Research*, 203(1):279–294, 2013. doi:10.1080/0740817X.2011.636792.

[7] F. Chin and D. Houck. Algorithms for updating minimal spanning trees. *Journal of Computer and System Sciences*, 16(3):333–344, 1978. doi:10.1016/0022-0000(78)90022-3.

[8] D. Eppstein. Offline algorithms for dynamic minimum spanning tree problems. *Journal of Algorithms*, 17(2):237–250, 1994. doi:10.1006/jagm.1994.1033.

[9] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44(5):669–696, 1997. doi:10.1145/265910.265914.

[10] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM Journal on Computing*, 14(4):781–798, 1985. doi:10.1137/0214055.

[11] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. URL: http://dx.doi.org/10.1145/28869.28874, doi:10.1145/28869.28874.

[12] T. Kalinowski, D. Matsypura, and M. W. Savelsbergh. Incremental network design with maximum flows. *European Journal of Operational Research*, 242(1):51–62, 2015. doi:10.1016/j.ejor.2014.10.003.

[13] H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005. doi:10.1002/net.20072.

[14] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984. doi:10.1287/trsc.18.1.1.

[15] S. G. Nurre, B. Cavdaroglu, J. E. Mitchell, T. C. Sharkey, and W. A. Wallace. Restoring infrastructure systems: An integrated network design and scheduling (INDS) problem. *European Journal of Operational Research*, 223(3):794–806, 2012. doi:10.1016/j.ejor.2012.07.010.

[16] S. G. Nurre and T. C. Sharkey. Integrated network design and scheduling problems with parallel identical machines: Complexity results and dispatching rules. *Networks*, 63(4):306–326, 2014. doi:10.1002/net.21547.

[17] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[18] T. C. Sharkey, B. Cavdaroglu, H. Nguyen, J. Holman, J. E. Mitchell, and W. A. Wallace. Interdependent network restoration: On the value of information-sharing. *European Journal of Operational Research*, 244(1):309–321, 2015. doi:10.1016/j.ejor.2014.12.051.

[19] D. D. Sleator and E. R. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, Jun 1983. doi:10.1016/0022-0000(83)90006-5.

[20] P. M. Spira and A. Pan. On finding and updating spanning trees and shortest paths. *SIAM Journal on Computing*, 4(3):375–380, 1975. doi:10.1137/0204032.