Incremental network design with shortest paths*

Matthew Baxter Tarek Elgindy Andreas Ernst Thomas Kalinowski
Martin Savelsbergh

Abstract

We introduce a class of incremental network design problems focused on investigating the optimal choice and timing of network expansions. We concentrate on an incremental network design problem with shortest paths. We investigate structural properties of optimal solutions, we show that the simplest variant is NP-hard, we analyze the worst-case performance of natural greedy heuristics, and we derive a 4-approximation algorithm.

1 Introduction

Consider a network optimization problem, e.g., a shortest path problem, a maximum flow problem, or a traveling salesman problem. Next, assume that this optimization problem has to be solved in a number of consecutive time periods and that in each time period the value of an optimal solution is recorded, e.g., the cost of an $s - t$ path, the value of an $s - t$ flow, or the cost of a TSP tour. Let the objective be to minimize or maximize the total cost or value over the planning horizon. At this point, that simply means solving the network optimization problem and multiplying the value of an optimal solution with the number of time periods in the planning horizon. It becomes more interesting when a budget is available in each time period to expand the network, i.e., to build additional links. Expanding the network may improve the cost or value of an optimal solution to the network optimization problem in future time periods and thus may improve the total cost or value over the planning horizon. However, deciding which links to build and the sequence in which to build them is nontrivial. In part, because in some situations the benefits of building a link will only materialize when other links have been built as well, e.g., adding a single link to the network does not lead to a shorter TSP tour, but adding two links to the network does.

We introduce a class of incremental network design problems focused on the optimal choice and timing of network expansions given that these network expansions impact the value a solution to an optimization problem that is solved on the network in each of the periods of the planning horizon. We concentrate on the incremental network design problem with shortest paths. We investigate structural properties of optimal solutions, we show that even the simplest variant is NP-hard, we establish a class of instances that can be solved in

* *European Journal of Operational Research* **238**(3), 675–684, 2014, doi:10.1016/j.ejor.2014.04.018

polynomial time, we analyze the worst-case performance of natural greedy heuristics, and we derive a 4-approximation algorithm.

Even though single-stage or single-period network design problems have been studied extensively, multi-stage or multi-period network design problems, which occur just as often in practice, have received much less attention. We hope that our investigation demonstrates that multi-period network design problems present interesting challenges and can produce intriguing and surprising results.

The remainder of the paper is organized as follows. In Section 2, we introduce the class of incremental network design problems. In Section 3, we present a brief literature review. In Section 4, we introduce the incremental network design problem with shortest paths. In Sections 5 and 6, we analyze the complexity of the incremental network design problem with shortest paths, and we explore the performance of natural greedy heuristics, respectively. In Section 7, we develop a 4-approximation algorithm for the incremental network design problem with shortest paths. Finally, in Section 8, we discuss possible extensions and future research.

2 A Class of Incremental Network Design Problems

Incremental network design problems have two characteristic features: a design feature, since we are deciding which arcs will be part of a network, and a multi-period feature, since the ultimate network design is built over a number of time periods.

The general structure of an incremental network design problem is as follows. We are given a network $D = (N, A)$ with node set N and arc set $A = A_e \cup A_p$, where A_e contains *existing* arcs and A_p contains *potential* arcs. Each arc $a \in A$ has a capacity C_a. Let T be the planning horizon. A budget B^t is available in every time period $t \in \{1, \ldots, T\}$. The budget can be used to build potential arcs $a \in A_p$, which will be available for use in the following period. For each potential arc $a \in A_p$, there is an associated build-cost $c_a \leqslant B$. Let y_a^t be a 0-1 variable indicating whether arc $a \in A_p$ has been built in or before time period t, with all potential arcs initially unbuilt ($y_a^0 = 0$). Thus, $y_a^t - y_a^{t-1} = 1$ indicates that arc a is built in time period t and can be utilized in period $t+1$. In every time period, a network optimization problem P has to be solved over the usable arcs in time period t, i.e., the existing arcs and the potential arcs that have been built before time period t. Let x_a^t represent the flow on arc $a \in A$ in time period $t \in \{1, \ldots, T\}$ in an optimal solution to the network optimization problem. Let $F(P)$ define the "structure" of feasible solutions to the network optimization problem, i.e., the set of constraints imposed on the flow variables (that it has to be an $s - t$ path, $s - t$ flow, a TSP tour, etc.). The value of an optimal solution to the network optimization P in time period t is function of the flows on the arcs in that period and denoted by $c(x^t)$. The objective is to minimize the total cost over the planning period. Thus, the generic formulation of an incremental network design problem

is as follows:

$$\begin{aligned}
\min \sum_{t\in\{1,\ldots,T\}} c(x^t) + \sum_{t\in\{1,\ldots,T\},a\in A_p} c_a(y_a^t - y_a^{t-1}) & \\
\text{s.t.} \quad x^t \in F(P) & \quad \text{for all } t \in \{1,\ldots,T\}, \\
x_a^t \leqslant C_a y_a^{t-1} & \quad \text{for all } a \in A_p, t \in \{1,\ldots,T\}, \\
\sum_{a\in A_p} c_a(y_a^t - y_a^{t-1}) \leqslant B^t & \quad \text{for all } t \in \{1,\ldots,T\}, \\
y_a^t \geqslant y_a^{t-1} & \quad \text{for all } a \in A_p, t \in \{2,\ldots,T\}
\end{aligned}$$

An incremental network design problem has characteristics in common with network design problems and with dynamic facility location problems. A brief review of some relevant literature is given below.

3 Literature review

Network design is a fundamental optimization problem and has a rich research tradition. The seminal paper by Magnanti and Wong [8] discusses many of its features, applications, models, and algorithms, with an emphasis on network design in transportation planning. Kerivin and Mahjoub [5] survey many of network design problems studied in telecommunications. The paper by Magnanti and Wong [8] mentions "Time Scale" as one of characteristics of a network design problem that can vary in different planning environments, e.g., transportation and water resource design decisions have long-term effects whereas communication system designs frequently are more readily altered. Not withstanding, the paper focuses exclusively on single-period or single-stage network design problems. Recently, the interest in multi-period or multi-stage network design problems in the area of transportation planning has picked up, partly because it better meets practitioners needs, as in many environments network design decisions span planning periods of up to 25 years and the intermediate network configurations are of concern as well as the final network configuration (see for example [6] and [11]). A class of network design problems where construction over time has been studied extensively is dynamic facility location (the recent review of Arabani and Farahani [1] is completely dedicated to dynamic facility location).

Studying approximation algorithms for network design problems has been popular as well, especially in the computer science community. Two prime examples are the papers by Goemans *et al.* [3] and Gupta *et al.* [4]. These approximation algorithms are for single-period network design problems. Multi-period or incremental approximation has been introduced in the context of facility location by Mettu and Plaxton [9]. They consider a situation where a company is building facilities in order to supply its customers, but because of capital considerations, the company will build the facilities over time. The question asked is whether the company can plan its future expansion in such a way that when it has opened the first k facilities, it is close, in value, to that of an optimal solution in which any choice of k facilities is opened. This problem is known as the incremental k-median problem. More formally, in the incremental k-median problem, we are given the input of the k-median

problem without the parameter k and must produce a sequence of the facilities. For each k, consider the ratio of the cost of opening the first k facilities in the ordering to the cost of an optimal k-median solution. The goal of the problem is to find an ordering that minimizes the maximum of this ratio over all values of k. See Lin *et al.* [7] for more on incremental optimization problems.

This brief literature review has covered areas of research that are relevant to the study of the class of incremental network design problems introduced in this paper. The review identified survey papers on single-period network design, recent papers discussing the practical importance of investigating multi-period network design, and fundamental papers on approximation algorithms for network design.

For the remainder of the paper, we focus on one particular incremental network design problem, namely the incremental network design problem with shortest paths (INDP-SP).

4 The Incremental Network Design Problem with Shortest Paths

We are given a network $D = (N, A)$ with node set N ($|N| = n$) and arc set $A = A_e \cup A_p$ ($|A| = m$), where A_e is the set of *existing* arcs and A_p is the set of *potential* arcs, as well as a source $s \in N$ and sink $t \in N$. For each arc $a \in A$, we are given a length $l_a \geqslant 0$. Let $T = |A_p| + 1$ be the planning horizon. In every time period, we have the option to expand the usable network, which initially consists of only the existing arcs, by "building" a single potential arc $a \in A_p$, which will be available for use in the following period. In every period, the cost (or length) of a shortest $s - t$ path is incurred (using only usable arcs, i.e., existing arcs and potential arcs that have been built in previous periods). The objective is to minimize the total cost over the planning horizon. Note that the length of the planning horizon ensures that every potential arc can be built. This also implies that a shortest $s - t$ path in D will always be built. We will refer to such a shortest $s - t$ path as an *ultimate* shortest $s - t$ path to distinguish it from a shortest $s - t$ path in a time period, which depends on the potential arcs that have been built up to and including that period. Note that there may be many ultimate shortest paths in the network, and the sequence of potential arcs built in an optimal solution may not be unique. This problem is related to, but also quite different from, shortest path re-optimization problems as studied by Gallo [2] and extended by Pallottino and Scutellà [10].

An example. Consider the network shown in Figure 1, where solid arcs represent existingarcs and dashed arcs represent potential arcs. A path of length 52 can be built in three periods and the ultimate shortest path of length 4 can be constructed in 4 periods. However, the optimal solution is to first build arcs a, b, and c to give a path of length 54, and then to build arcs d, e, and f to complete the ultimate shortest path for a total cost over the planning horizon of 3174.

We start by observing a useful property of an optimal solution. Let $\bar{T}$ denote the time period in which an ultimate shortest path is completed in an optimal solution.

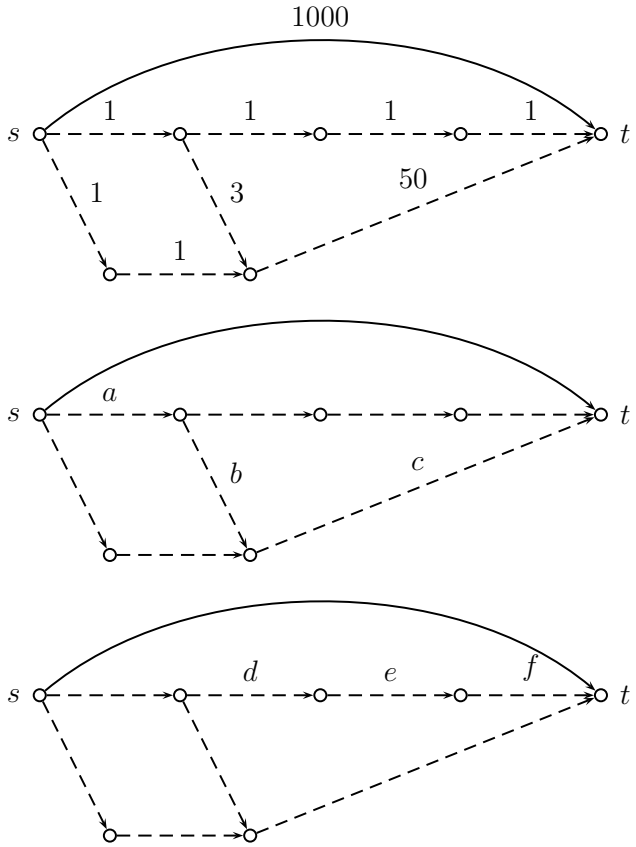

Figure 1: An example.

Observation 1. Let $(a_1, a_2, \ldots, a_{\bar{T}})$ be an optimal sequence of potential arcs to build $(a_i \in A_p)$. Let $(P_1, P_2, \ldots, P_{\bar{T}}, P_{\bar{T}+1})$ be a sequence of associated shortest paths, where P_i is the path used while a_i is being constructed. Let c_i be the cost of path P_i. By grouping paths with identical costs, we obtain subsequences $S_1, S_2, \ldots, S_K$. Let P_{S_i} denote a path associated with subsequence S_i. Let A_{S_i} denote the potential arcs built during subsequence S_i. Then we have

$$A_{S_i} = P_{S_{i+1}} \cap A_p \setminus (P_{S_1} \cup \cdots \cup P_{S_i}) \text{ for } i = 1, \ldots, K-1.$$

Proof. Each arc a_j for $j = 1, \ldots, \bar{T}$ must contribute to the construction of some path P_{S_i} with $i \in \{1, \ldots, K\}$, otherwise an improved solution is obtained by simply removing a_j from $(a_1, a_2, \ldots, a_{\bar{T}})$. Furthermore, suppose there exist j and j' with $j < j'$ and i and i' with $i < i'$ such that $a_{j'}$ contributes to the construction of path P_{S_i} and a_j contributes to the construction of path $P_{S_{i'}}$, but not to the construction of paths P_{S_k} with $k \in \{i, i+1, \ldots, i'-1\}$. By interchanging a_j and $a_{j'}$, the paths P_{S_k} with $k \in \{i, i+1, \ldots, i'-1\}$ are completed earlier, while all other paths are completed at the same time, which reduces the

total cost. □

Thus, the problem can be viewed as seeking a sequence of potential arcs to build, but also as seeking a sequence of paths to be constructed, and the last property establishes that in an optimal solution any potential arc that is built before an ultimate shortest path is completed will be part of the next shortest path.

5 Complexity

Since the shortest path problem is polynomially solvable, the complexity of the incremental network design problem with shortest paths is not obvious. The following theorem shows that even the simplest variant of the incremental network design problem with shortest paths as defined above is NP-hard.

Theorem 1. *The incremental incremental network design problem with shortest paths is NP-hard.*

Proof. Reduction from 3-SAT. Consider an instance of 3-SAT with m clauses, i.e., $c_1 \wedge c_2 \wedge \cdots \wedge c_m$, each of which contains three literals, i.e., $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$, where each literal is a Boolean variable x_j or its negation. Let there be n Boolean variables. We construct the instance of the incremental network design problem with shortest paths given in Figure 2. As before, existing arcs are shown as solid arcs and potential arcs are shown as dashed arcs. Let $M_m = 1$, $M_{m-1} = 2$, …, $M_1 = m$, $M_0 = m + 1$ and $I = \lceil \frac{(m+1)(n+1)}{2} \rceil + 1$, and all other arcs have a cost of 0. Observe that initially there is only a single path from s to t with cost I. Next observe that a shorter path of length M_0 can be constructed in n periods by building one of the two potential arcs associated with a variable x_i (i.e., x_i or $\neg x_i$) for each of the n variables. Next observe that a path from the source to the sink associated with clause c_i requires that at least one of the potential arcs associated with the literals in c_i has to be build and that the cost of such a path is M_i. It is now easy to see that if the instance of SAT-3 can be satisfied, a solution to the incremental shortest path problem with value $nI + (n+1)M_0 + M_1 + M_2 + \cdots + M_m$ exists (all but n of the potential arcs, all associated with the Boolean variables, will be build and the ultimate shortest path, with length 0, will be used in the last in periods). Furthermore, if the instance of 3-SAT cannot be satisfied, a solution with strictly greater value will be obtained. Note that sequence of potential arcs through nodes $s_1, s_2, \ldots, s_n$ is introduced to ensure that it is always advantages to build n potential arcs associated with the Boolean variables before building a path associated with a clause. □

Even though the incremental network design problem with shortest paths is NP-hard, we next show that there exist special cases which are polynomially solvable. Consider the special case of *disjoint paths* graphs, in which the arcs $a \in A$ form node-disjoint paths from s to t. More specifically, consider a disjoint paths graph with $r + 1$ paths $P_0, P_1, \ldots, P_r$. Because the paths are disjoint, the order in which the potential arcs on a path are build is immaterial. Therefore, for $i \in \{0, 1, \ldots, r\}$, let q_i denote the number of potential arcs on P_i, and, as before, let c_i denote the cost of path P_i. It is easy to see that if $c_i > c_j$ and

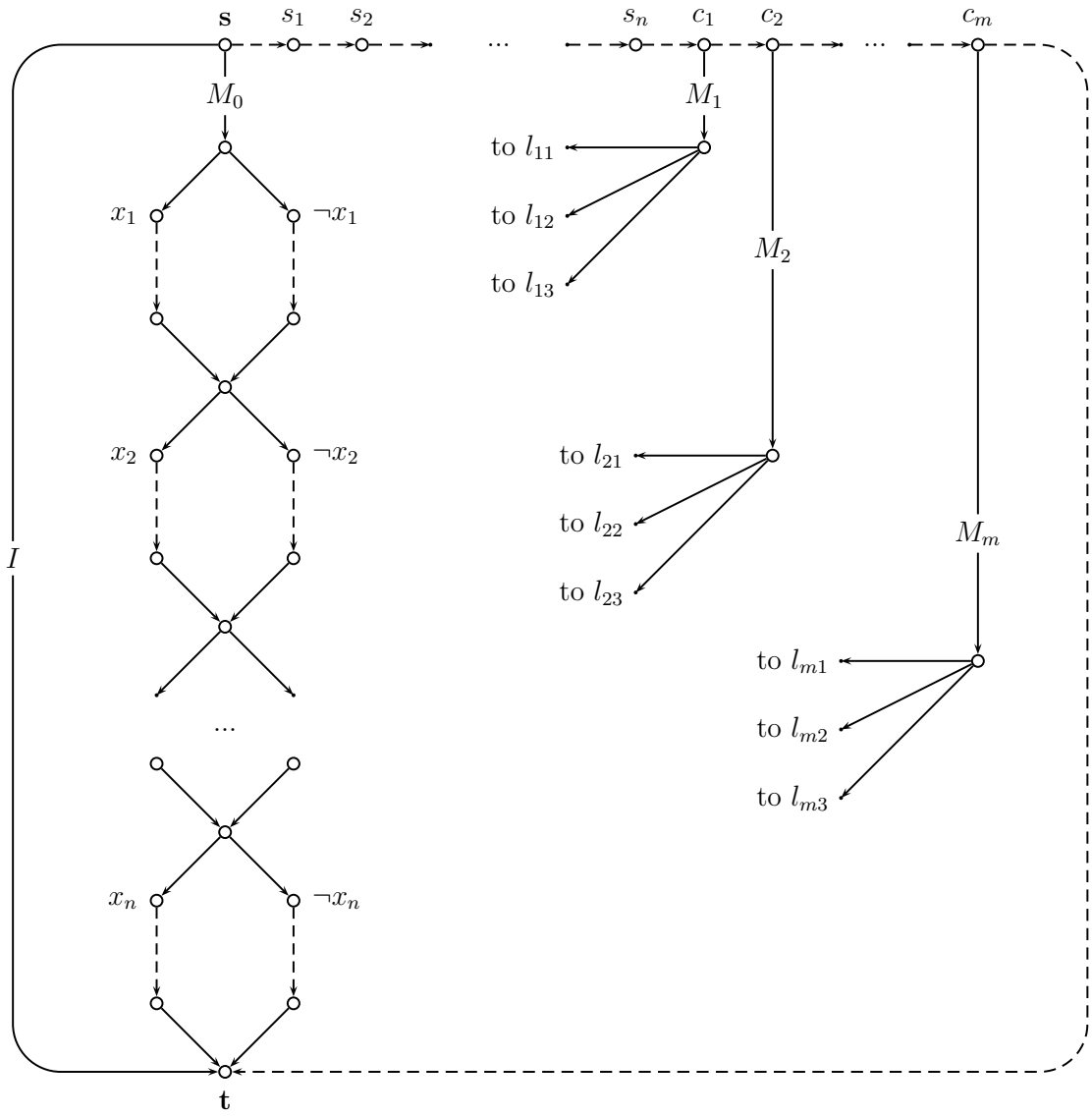

Figure 2: Instance of INDP-SP used in the reduction from 3-SAT.

$q_i \geqslant q_j$, then the path P_i will not be constructed in an optimal solution. Therefore, we may assume that $0 = q_0 < q_1 < \cdots < q_r$ and $c_0 > c_1 > \cdots > c_r$. We construct an auxiliary digraph $\bar{D}$ with node set $\{0, 1, \ldots, r+1\}$. An arc set of $(r, r+1)$ with weight c_r, and arcs (i, j) for $0 \leqslant i < j \leqslant r$ with weights

$$w(i,j) = q_j c_i + (\sum_{k=i+1}^{j-1} q_k) c_r.$$

The paths from 0 to r in $\bar{D}$ are in one-to-one correspondence to solutions of the given disjoint paths instance: a path $(0 = i_0, i_1, \ldots, i_s = r)$ corresponds to building the paths $P_{i_1}, P_{i_2}, \ldots, P_{i_s}$ in this order, and the weight of the path in $\bar{D}$ equals the objective value for the disjoint paths instance. Note that $q_{i_j} c_{i_{j-1}}$ for $j = 1, \ldots, s$ captures the cost of building path P_{i_j} and $(\sum_{i \in \{1,\ldots,r\} \setminus \{i_1, i_2, \ldots, i_s\}} q_i) c_r$ captures the cost incurred after the ultimate shortest path has been built. Thus, we have proved the following result.

Proposition 1. *An instance associated with a disjoint paths graph with $r+1$ paths and characterized by vectors $(q_1, \ldots, q_r)$ and $(c_0, \ldots, c_r)$ can be solved in time $O(r^2)$.*

In the following proposition, we completely characterize the instances associated with disjoint paths graphs where all potential arcs need to be build.

Proposition 2. *For an instance associated with a disjoint paths graph with $r+1$ paths and characterized by vectors $(q_1, \ldots, q_r)$ and $(c_0, \ldots, c_r)$ all potential arcs have to be built in any optimal solution if and only if*

$$(q_{i+1} - q_i) c_{i-1} + q_i c_r > q_{i+1} c_i \tag{1}$$

for all $i \in \{1, 2, \ldots, r-1\}$.

Proof. Suppose the inequality (1) holds for all $i \in \{1, \ldots, r\}$ and there is an optimal solution with $I \subseteq \{1, \ldots, r\}$ being the set of indices i such that P_i is built in this solution. Writing $I = \{i_1 < i_2 < \cdots < i_s = r\}$ and putting $i_0 = 0$, the optimal objective value is

$$f(I) = \sum_{k=1}^{s} q_{i_k} c_{i_{k-1}} + c_r \left(\sum_{i \in \{1,\ldots,r\} \setminus I} q_i + 1 \right)$$

Suppose there is some $i \in \{1, \ldots, r-1\}$ with $i \notin I$. Then we may assume that $i = i_l - 1 > i_{l-1}$ for some $l \in \{1, \ldots, s\}$ and the objective value for the index set $I' = I \cup \{i\}$ is

$$f(I') = f(I) - (q_{i+1} - q_i) c_{i_{l-1}} + q_{i+1} c_i - c_r q_i < f(I)$$

and this contradicts the optimality of I.

Conversely, suppose $I = \{1, 2, \ldots, r\}$ is the index set of the paths build in the unique optimal solution with objective value $f(I)$ and that there is an $i \in \{1, \ldots, r-1\}$ violating (1). The objective value for building the paths with with indices in $I' = I \setminus \{i\}$ is

$$f(I') = f(I) + (q_{i+1} - q_i) c_{i-1} - q_{i+1} c_i + q_i c_r \leqslant f(I)$$

contradicting the assumption that all paths have to be build in any optimal solution. □

One possible way to satisfy the condition in Proposition 2 is to put $q_i = i$ for $i = 0, 1, 2, \ldots, r$, and to define c_r recursively by $c_r = 0$ and $c_{i-1} = (i+1)c_i + 1$ for $i = r-1, r-2, \ldots, 0$. This is illustrated in Figure 3 for $r = 5$.

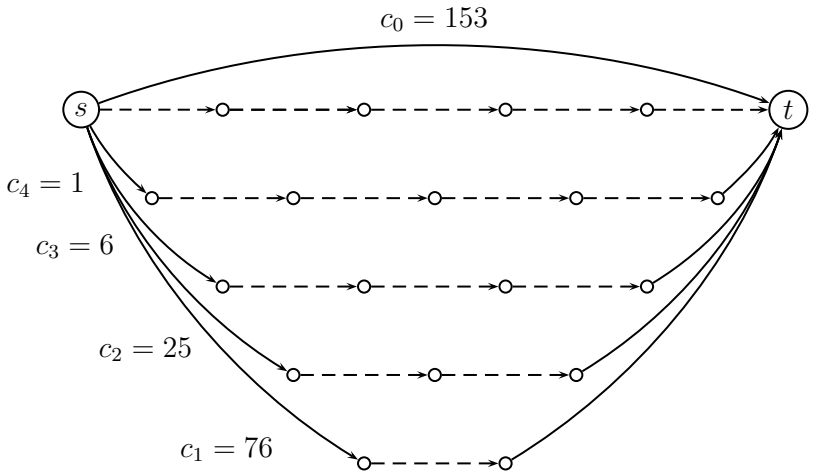

Figure 3: An instance associated with a disjoint paths graph in which all potential arcs need to be build in an optimal solution.

6 Greedy Heuristics

Two simple natural greedy heuristics are to

1. Always build arcs that lead to a shorter $s - t$ path as quickly as possible (H_1); and

2. Build an ultimate shortest path as quickly as possible (H_2).

Next, we show that the performance of both these heuristics, as well as of the heuristic where we take the best of the two solutions (H_3), can be bad. Let the sequence (c_k) be given by values $c_0 = 1$, $c_1 = 1$, $c_2 = 4$, and the recursion $c_k = 5c_{k-1} - 3c_{k-2}$ for $k > 2$.

Lemma 1. *We have* $c_k = \sum_{i=1}^{k-1} (i+2)c_{k-i} + c_0$ *for all* k.

Proof. Induction on k. For $k \leqslant 2$ it's easy to check, and for $k > 2$ we get

$$\begin{aligned} c_k = 5c_{k-1} - 3c_{k-2} &= 3c_{k-1} + 2\left(\sum_{i=1}^{k-2}(i+2)c_{k-1-i} + c_0\right) - 2c_{k-2} - \left(\sum_{i=1}^{k-3}(i+2)c_{k-2-i} + c_0\right) \\ &= 3c_{k-1} + 4c_{k-2} + \left(\sum_{i=2}^{k-2}2(i+2)c_{k-1-i} - \sum_{i=1}^{k-3}(i+2)c_{k-2-i}\right) + c_0 \\ &= 3c_{k-1} + 4c_{k-2} + \left(\sum_{i=3}^{k-1}2(i+1)c_{k-i} - \sum_{i=3}^{k-1}ic_{k-i}\right) + c_0 \\ &= \sum_{i=1}^{k-1}(i+2)c_{k-i} + c_0. \end{aligned}$$

□

Now consider the instance given in Figure 4. We have $T = |A_p| + 1 = 2 + 3 + \cdots + (k+1) + T/2 + 1$, hence $T = (k+1)(k+2)$. We compare the following three strategies

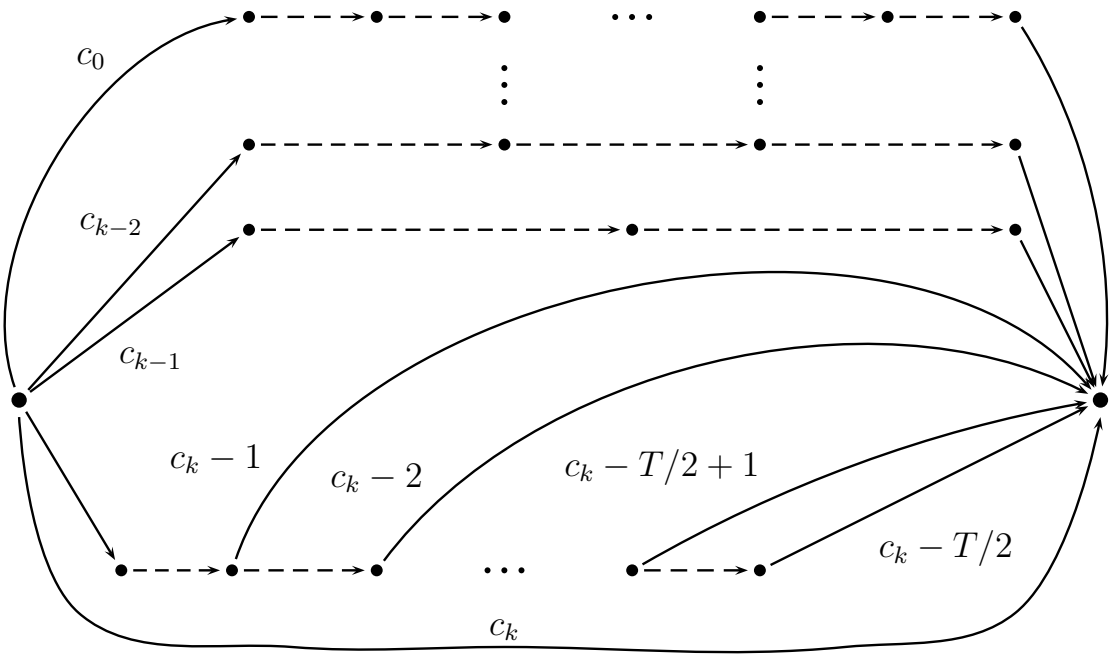

Figure 4: Instance on which natural greedy heuristics perform bad.

1. H_1, i.e. build the bottom path first, and then the remaining paths from bottom to top;
2. H_2, i.e., build only the top path; and
3. Build all paths except the bottom one.

For large enough k, we can bound the corresponding objective function values as follows.

1. For H_1 we get

$$\sum_{i=0}^{T/2}(c_k - i) + (c_k - T/2) + \sum_{i=1}^{k-1}(i+2)c_{k-i} + c_0 = \left(\frac{T}{2} + 3\right)c_k - \frac{T(T+2)}{8} - \frac{T}{2} \geqslant kc_k.$$

2. For H_2 we get

$$(k+1)c_k + (T-k-1)c_0 \geqslant kc_k.$$

3. For the third strategy we get

$$2c_k + \sum_{i=1}^{k-1}(i+2)c_{k-i} + c_0 + \frac{T}{2}c_0 = 3c_k + \frac{T}{2}c_0 \leqslant 4c_k.$$

This implies that the two simple natural greedy heuristics produce solutions with values that are at least $\frac{k}{4}$ times the optimal value for this instance. The above shows that the two simple natural greedy heuristics do not provide constant-factor approximation algorithms.

7 Approximation Algorithm

For $k = 0, 1, \ldots, T$, let d_k deonte the cost of the shortest path from source to sink using at most k arcs from A_p. The value of d_k can be computed using the modified Bellman-Ford algorithm shown in Algorithm 1, where $d_k(u, v)$ is the cost of a shortest $u - v$ path using at most k arcs from A_p. Set $\Delta' = d_T$ to be the cost of an ultimate shortest path, and $\Delta = d_0 - d_T$ be the cost of the initial shortest path, less the cost of an ultimate shortest path.

Algorithm 1 k-costs from the source.

for $k = 0, \ldots, T$ **do**
  if $k = 0$ **then**
    $d_k(s, s) \leftarrow 0$
    $d_k(s, v) \leftarrow \infty$ for all $v \neq s$
  else
    $d_k(s, v) \leftarrow d_{k-1}(s, v)$ for all $v \in N$
    for $a = (v, w) \in A_p$ **do**
      $d_k(s, w) \leftarrow \min\{d_k(s, w),\ d_k(s, v) + c_a\}$
  for $i = 1, \ldots, n$ **do**
    for $a = (v, w) \in A_e$
      $d_k(s, w) \leftarrow \min\{d_k(s, w),\ d_k(s, v) + c_a\}$
return $d_k(s, t)$ as d_k for all k

Denote the value of κ_i as the minimum number of arcs that need to be constructed to obtain an $s-t$ path with cost less than $\Delta' + \frac{\Delta}{2^i}$. For convenience, we also introduce parameters $k_0 = \kappa_0$ and $k_i = \kappa_i - \kappa_{i-1}$ for $i \geqslant 1$, which gives $\kappa_i = k_0 + k_1 + \cdots + k_i$. Let K be the minimal index k with $d_k = d_T$, and r be the smallest integer with $d_{K-1} \geqslant \Delta' + \frac{\Delta}{2^r}$. This gives $\kappa_r = k_0 + k_1 + \cdots + k_r = K$.

Algorithm 2 Approximately solving the incremental network design problem with shortest paths.

Compute the values d_k using Algorithm 1
$K \leftarrow \min\{k \ : \ d_k = d_T\}$
$I = \{i \text{ such that } k_i > 0\}$
for $i = 0, \ldots, r$ **do**
    $\kappa_i \leftarrow \min\{k \ : \ d_k < d_T + (d_0 - d_T)/2^i\}$
$j \leftarrow 0$ {number of added arcs}
$\overline{A} \leftarrow \varnothing$ {set of added arcs}
for $i \in I$ **do**
    Determine an s-t path P of cost d_{κ_i} using at most κ_i arcs from A_p
    for $a \in (P \cap A_p) \setminus \overline{A}$ **do**
        $j \leftarrow j + 1$
        $a_j \leftarrow a$
        $\overline{A} \leftarrow \overline{A} \cup \{a\}$
Build arcs $a_1, a_2, \ldots, a_j$ in that order.

The approach that is taken in Algorithm 2, is to firstly determine the smallest number of arcs that can be constructed such that cost of a shortest $s-t$ path is less than $\frac{\Delta}{2^i} + \Delta'$ for each $i = 0, 1, \ldots, r$. This defines the values of κ_i and k_i. If $k_i > 0$, then $\frac{\Delta}{2^i} + \Delta'$ provides a lower bound on the cost incurred while constructing a path that has k_i more potential arcs than the previously used shortest $s-t$ path. Furthermore, $\frac{\Delta}{2^{i-1}} + \Delta'$ provides an upper bound on the cost incurred while constructing a path that has k_i more potential arcs then the previously used shortest $s-t$ path. The number of arcs that must be constructed to complete this path depends on the number of arcs previously constructed on the $s-t$ path that is utilized. The lower bound on the number of arcs that need to be constructed is k_i, and the upper bound is given by κ_i. The upper bound represents the case where the path is disjoint, and requires all potential arcs in the path to be constructed. In Figure 5, these two cases refer to the top branch and the bottom $r+1$ branches respectively. We can combine the bounds on both the number of arcs constructed and their cost to obtain upper and lower bounds on the solution returned by Algorithm 2.

Lower bound A cost of at least Δ' is incurred every time period so the baseline cost is $T\Delta'$. As a lower bound, $\frac{\Delta}{2^i}$ is incurred for k_i periods, while constructing the path that has k_i more potential arcs than the previously used shortest $s-t$ path. The total cost of construction is bounded below by $L = T\Delta' + \sum_{i \in I} k_i \frac{\Delta}{2^i}$.

Upper bound Again, the baseline cost is $T\Delta'$. As an upper bound, $\frac{\Delta}{2^{i-1}}$ is incurred for κ_i periods, while constructing the path that has k_i more potential arcs than the previously used shortest $s-t$ path. The total cost of construction is bounded above by $U = T\Delta' + \sum_{i \in I} \kappa_i \frac{\Delta}{2^{i-1}}$.

Combining these two bounds, we can bound the approximation ratio of Algorithm 2.

Proposition 3. *Algorithm 2 is a 4-approximation for the incremental network design problem with shortest paths.*

Proof. We can now demonstrate that Algorithm 2 is a 4-approximation, by showing that $U \leqslant 4L$:

$$U = T\Delta' + \sum_{i \in I} \kappa_i \frac{\Delta}{2^{i-1}} = T\Delta' + \sum_{i \in I} \left[\left(\sum_{j=0}^{i} k_j \right) \frac{\Delta}{2^{i-1}} \right]$$

$$= T\Delta' + \sum_{j=0}^{r} \left[\left(\sum_{i \in I,\, i \geq j} \frac{\Delta}{2^{i-1}} \right) k_j \right] \leqslant T\Delta' + \sum_{j=0}^{r} \frac{\Delta}{2^{j-2}} k_j \leqslant 4T\Delta' + 4 \sum_{j=0}^{r} \frac{\Delta}{2^j} k_j = 4L. \quad \square$$

The worst case behaviour for this algorithm can be observed in Figure 5. The optimal solution is to build the upper path which gives an objective value of

$$2 \cdot 2^r + k \left(2^{r-1} + 2^{r-2} + \cdots + 2^1 + 2^0 \right) = (k+2)2^r - k.$$

But Algorithm 2 builds the other paths from top to bottom and this yields

$$2^r + \sum_{i=1}^{r+1} (ik+1)(2^{r-i+1} - 1) = (4k+3)2^r - k\frac{(r+2)(r+3)}{2} - k - r - 2.$$

Now the claim follows since for sufficiently large $k = r$,

$$(4k+3)2^r - k\frac{(r+2)(r+3)}{2} - k - r - 2 > (4-\varepsilon)\left[(k+2)2^r - k\right].$$

8 Conclusions and Extensions

In this paper, we have introduced a class of incremental network design problems and studied one member of the class, the incremental network design problem with shortest paths, in more detail. We have established that the problem is NP-hard, but have also identified a polynomially solvable special case. Natural greedy heuristics have been analyzed and a 4-approximation algorithm has been developed.

A natural extension to the variant of the incremental network design problem with shortest paths considered in this paper is obtained by introducing arc construction costs and construction budgets. In that case, we are also given for each arc $a \in A_p$ a construction cost $\bar{c}_a$ and for each period t a budget B_t and the option to expand the network in each

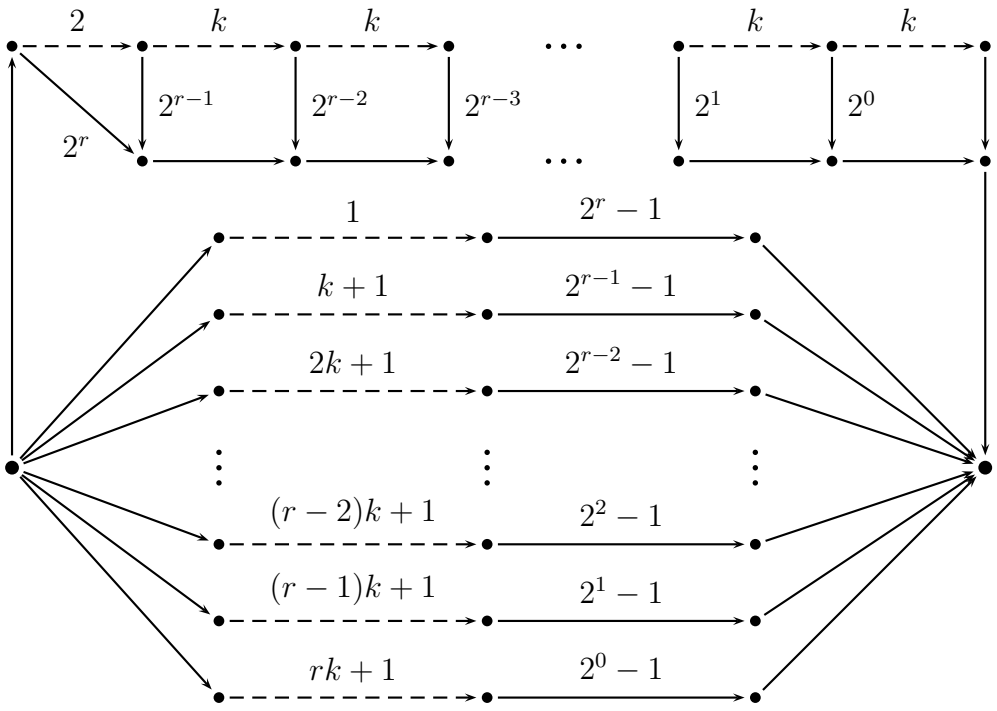

Figure 5: Instance on which Algorithm 2 performs badly. A dashed arc with label i indicates a path of i potential arcs, and a solid arc with label i indicates an existing arc with cost i. All potential arcs and the unlabeled solid arcs have zero cost.

period by building potential arcs $a \in A_p$ subject to the constraint that the construction costs do not exceed the budget. As before, the objective is to minimize the total cost over the planning period, which now includes construction costs and shortest $s - t$ path costs.

Another natural extension is to consider multiple commodities, where each commodity i requires one unit of flow to be send from source s_i to sink t_i. This incremental network design problem with multi-commodity flows can be uncapacitated, i.e., there is no limit on the flow on an arc, or capacitated, where the flow on arc has to be less than or equal to a given capacity. When the capacity is equal to one for all arcs, the subproblem in each period reduces to finding arc-disjoint paths.

Rather than considering network expansion, we can consider network maintenance. A class of multi-period network maintenance problems can be defined in a similar way to the class of incremental network design problems. In this setting, arcs are required to undergo maintenance, either once or with a certain frequency, and when an arc undergoes maintenance it is unavailable and cannot be used in the network optimization problem that has to be solved each period. This class of multi-period network maintenance problems also offers a rich environment for further research.

Finally, it is not difficult to formulate incremental network design problems and multi-period network maintenance problems as integer programs. We are currently exploring whether integer programming formulations of the incremental network design problem with shortest paths and the incremental network design problem with multi-commodity flows are computationally tractable and whether valid or facet inducing inequalities can be derived from the multi-period structure.

References

[1] A.B. Arabani and R.Z. Farahani. "Facility location dynamics: An overview of classifications and applications". In: *Computers & Industrial Engineering* (2011), pp. 408–420. DOI: 10.1016/j.cie.2011.09.018.

[2] G. Gallo. "Reoptimization procedures in shortest path problem". In: *Decisions in Economics and Finance* 3.1 (1980), pp. 3–13.

[3] M.X. Goemans, A.V. Goldberg, S. Plotkin, D.B. Shmoys, E. Tardos and D.P. Williamson. "Improved approximation algorithms for network design problems". In: *Proc. 5th ACM-SIAM symposium on Discrete algorithms SODA 1994*. Society for Industrial and Applied Mathematics. 1994, pp. 223–232.

[4] A. Gupta, A. Kumar and T. Roughgarden. "Simpler and better approximation algorithms for network design". In: *Proc. 35th ACM symposium on Theory of computing STOC 2003*. ACM. 2003, pp. 365–372. DOI: 10.1145/780542.780597.

[5] H. Kerivin and A.R. Mahjoub. "Design of survivable networks: A survey". In: *Networks* 46.1 (2005), pp. 1–21. DOI: 10.1002/net.20072.

[6] B.J. Kim, W. Kim and B.H. Song. "Sequencing and scheduling highway network expansion using a discrete network design model". In: *The Annals of Regional Science* 42.3 (2008), pp. 621–642. DOI: 10.1007/s00168-007-0170-2.

[7] G. Lin, C. Nagarajan, R. Rajaraman and D.P. Williamson. "A general approach for incremental approximation and hierarchical clustering". In: *SIAM Journal on Computing* 39.8 (2010), pp. 3633–3669. DOI: 10.1137/070698257.

[8] T.L. Magnanti and R.T. Wong. "Network design and transportation planning: Models and algorithms". In: *Transportation Science* 18.1 (1984), pp. 1–55. DOI: 10.1287/trsc.18.1.1.

[9] R.R. Mettu and C.G. Plaxton. "The online median problem". In: *SIAM Journal on Computing* 32.3 (2003), pp. 816–832. DOI: 10.1137/S0097539701383443.

[10] S. Pallottino and M.G. Scutellà. "A new algorithm for reoptimizing shortest paths when the arc costs change". In: *Operations Research Letters* 31.2 (2003), pp. 149–160. DOI: 10.1016/S0167-6377(02)00192-X.

[11] S.V. Ukkusuri and G. Patil. "Multi-period transportation network design under demand uncertainty". In: *Transportation Research Part B: Methodological* 43.6 (2009), pp. 625–642. DOI: 10.1016/j.trb.2009.01.004.