A method for designing minimum-cost multisource multisink network layouts

Heijnen, Petra W.; Chappin, Emile J.L.; Herder, Paulien M.

DOI
10.1002/sys.21492

Publication date
2019

Document Version
Final published version

Published in
Systems Engineering

Received: 23 January 2018 | Revised: 3 April 2019 | Accepted: 30 April 2019

DOI: 10.1002/sys.21492

REGULAR PAPER

WILEY

A method for designing minimum-cost multisource multisink network layouts

Petra W. Heijnen | **Emile J.L. Chappin** | **Paulien M. Herder**

Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands

Correspondence
P.W. Heijnen, Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX, Delft, The Netherlands.
Email: p.w.heijnen@tudelft.nl

Abstract

Systems engineers are equipped to design complex networked systems such as infrastructures. A key goal is cost minimization over a vast solution space. However, finding a minimum-cost system while comprehensively satisfying different stakeholders is challenging and lacks proper methodological support. Stakeholders often employ their own expert estimations for lack of suitable decision-support methods. In these settings, systems engineers typically require mid-fidelity, easy-to-use methods. We present a rigorous method that quickly finds minimum-cost solutions for networks with multiple sources and sinks, focusing on pipeline topology, length, and capacity. It can serve as a discussion tool in multiactor design processes, to demarcate the design space, indicate sources of uncertainty, and provoke further analyses, different designs, or contractual negotiations. It is applicable to a wide variety of cases, including many prominent infrastructures needed to mitigate CO_2. We prove that the optimal layout is a minimum-cost Gilbert tree, and develop a heuristic based on the Gilbert-Melzak method. We demonstrate the method's efficacy for a case set regarding solution quality, computational time, and scalability. We also show its efficiency and usefulness for systems engineers in real-world settings. Systems engineers can use the generated cost-optimal system designs to benchmark any design changes in real-world negotiation processes.

KEYWORDS

cost minimization, decision support method, multiactor network design, multisource multisink, network design, system architecting

1 | INTRODUCTION

Networked infrastructures such as roads, telecom, gas, and water pipelines or power grids provide essential utilities and services to society. Common characteristics of such infrastructures include high initial capital costs, generally long lifetimes and as a consequence irreversibility once the construction of such networks has finished.[1] Although Western societies already have mature infrastructure networks, development of new networks and the expansion or adaptation of these networks are very topical and minimization of the associated costs is of high societal relevance, in particular in the energy sector. Pressing, present-day, system-architecting challenges for energy infrastructure networks related to mitigating CO_2 are, for example,

- Biogas, produced on farms in rural areas, can be used to partially replace natural gas. The volume of the produced biogas often surpasses the demand on the farm. Farms could transport the biogas to a network, connecting various farms. Such a network is a determining cost factor. Therefore, farmers, or their intermediates, search for a cost minimum architecture or topology of the biogas network to reduce the cost of biogas distribution. This case is further explored in this paper in Section 4.4.
- A prominent option for meeting Europe's CO_2 targets is to capture CO_2 at various large-capacity point sources such as power plants or steel mills, and transport this CO_2 to subsurface or subsea storage facilities such as depleted gas fields. Realizing carbon capture storage (CCS) is crucial as it is the largest individual measure in terms

of CO_2 reduction until 2030 in the Netherlands (as agreed upon the Dutch government agreement, being 18 Mton out of 56 Mton[2]). However, there is still a huge gap between what is expected and what has been realized.[3] As to date, no CO_2 network exists that connects sources and sinks in a wide geographical spread. Such network investments need to be borne by different actors, and we do not yet know what actors, and such investments have a strong public-private characteristic, given the societal value of CO_2 reduction.[4] It is therefore of crucial importance to find a cost-minimal solution for the design of such network.

- Many large-scale wind farms on the North Sea Germany, the UK, Norway, Denmark, Sweden, and the Netherlands will be developed in the next decades. Research[5,6] has shown that depending on the technologies chosen and the governance framework that is put into place, different topologies of the network will emerge, the theoretical minimum-cost network would give the actors a sense of how far away they are from this minimum.

The systems engineering discipline has traditionally developed methods to minimize the overall cost of complicated engineering systems (eg, airplanes) and also yielded important methods and tools for developing and scanning the trade space, for multiattribute decision-making and for dealing with complexity.[7] In the past decade, the attention has broadened to the design of not only highly complicated systems, but rather to complex sociotechnical systems. The design of large-scale sociotechnical systems, such as the infrastructure systems presented above, is a multiactor process,[8] pulling systems engineering into the social science domain. This means that new disciplines and approaches are needed to explore those problems and solve them; approaches that can deal with the sheer size of the problem in a multiactor context.[9] Infrastructure networks have a large number of degrees of freedom,[10] which makes it hard to intuitively compare the cost effectiveness of alternatives. This paper develops a method that supports a systems engineer by quickly finding least-cost network topologies. The method developed in this paper is particularly useful in a multiactor process, since it enables a quick analysis of low-cost networks, while real-world alternatives are discussed among stakeholders on the negotiation table.

Garber et al[11] recently developed a framework to capture such decision-making by diverse stakeholders. Our infrastructure network design problem is, in their terminology, a cooperation game, especially in those cases where network externalities increase when more sources and sinks are connected. Scholars argue that the decision process needs rigorous models to estimate values for alternative designs, to explore the trade space, find theoretical minimum-cost targets, and find the main values drivers.[12] This requires a deterministic, unbiased, and traceable calculation of those values. Supplying the stakeholders with an agreed upon, efficient (fast) calculation process, will allow them to execute several sensitivity analyses to explore how their stakes change with changes in assumptions or parameters. Such approach might go against some developments in the systems engineering discipline where more and more details from social processes and values are being incorporated into increasingly opaque models. Our proposed approach, using mid to low fidelity, relatively simple models, has shown to contribute to trust in the negotiation process and therefore enhances the possibility of a reasonable outcome of the systems engineering process.[8] Finding adequate models and processes that produce agreed upon values for the huge trade spaces in the sociotechnical systems of systems that we are dealing with is therefore one of the key challenges in systems engineering.

Systems engineering processes are inherently combinatorically complicated, given the many design variables and their combinations into solutions. One important design variable in the design of networked systems is the topology of the network. However, the topology of networked systems, under uncertainty and in the context of many actors is an issue that has received relatively little attention in the systems engineering field. This is, on the one hand, due to the fact that in many design cases, the network topology was more or less fixed—for example, by street patterns to lay out city infrastructure—and, on the other hand, due to the fact that topological design is a mathematically complex problem in itself. For infrastructure developments, like the one we introduced earlier, the location of splitting nodes and junctions is still undecided, and capacities of links are often uncertain, rendering the topology challenge even more daunting.

For our examples, biogas, CO_2, grids at sea, such decision-making is inherently multiactor, and examples of intrinsic uncertainties are: capacity of sources and sinks (how much biogas will be produced, how much CO_2 can the aquifer store, how many wind farms will be built and connected), the cost of right of way (licensing, buying out of property), and willingness to participate by actors (veto rights, political uncertainties, societal acceptance in the CO_2 case). Exploring these uncertainties in a the multiactor design process will help to demarcate the design space, and to pinpoint main areas of uncertainty, and thus provoking further sensitivity analyses, or different designs or move actors into new contractual negotiations. The sensitive and dynamic nature of such design and negotiation processes requires methods that are able to quickly but accurately scan the trade space, so that different solutions can be explored collectively by the stakeholders, for example, by experimenting with parameter values, or with assumptions.

This paper describes such an efficient method to assess the relative cost of different network topologies. Our method is to be used by systems engineers and is especially suitable to explore the trade space in an objective manner in a multiactor setting. The method enables systems engineers to explore the range of most cost-effective physical networks for those cases where topology and capacity are still uncertain and encompass large trade spaces. For the scope of this paper, we disregard control and institutional layers in our algorithm, as well as those laws of nature that may govern the more detailed (multiphase) physical flow within the pipes of those networks, even though they may be mildly dependent on the network topology. We generalize away from those details to develop and test different algorithms, while assuming that the required capacity per producer or consumer can be determined.

It is important to accentuate that we focus on developing an adequate rational and straightforward tool that is to be used in social settings of negotiating actors. Following Refs.13 and 14, we explicitly and

intentionally chose to not include such social factors into our method, as this more than often leads to unrealistic models using simple weight factors for different actors' goals, ultimately leading to opaque model outcomes, and reduced trust in the model's outcomes by actors. By excluding much of those socioeconomic speculations and assumptions from our method, we trust in the shared use of the method in a real-life multiactor setting to account for many of such social values.

Our method determines a network layout that minimizes the initial investment costs that depend on both the length and the capacity of the pipelines, satisfying the demands of the consumers, for multiple sources (suppliers) and multiple sinks (consumers). For ease-of-reading, we will use the term pipelines in this paper to denote various infrastructure connections. We will show in this paper how our algorithm is able to find optimal network topologies, or trees, by explaining how we used the theory and how this leads to feasible and cost-minimum designs in reasonable computing time. The latter performance criterion of our approach is important, given that the algorithm needs to perform in a context where a systems engineer would want to redesign and test solutions as quickly as possible. Also, since we target real-world multiactor settings, table top drawing sessions, or serious-gaming sessions, the computational performance or our algorithm is of crucial importance.

The next section will start with a literature overview of energy network design methods. We will root our approach in operations research, this being an important contributor to many systems engineering optimization problems. After that section, we will formulate the mathematical design problem and discuss our assumptions, the latter being of importance to the users of the approach. We go into detail in explaining the underpinnings of our approach and algorithm. Detailed descriptions can be found in the Appendices and are relevant for users who want to understand the algorithm. Also, it allows systems engineers to use the optimal topologies in discussion with other stakeholders and adjust parameters in the algorithm during these debates. We develop our solution method, and we will show its efficacy and efficiency in Section 4 of this paper. Finally, we test the usability and usefulness of the method on a number of case studies executed by groups of systems engineering students who were challenged with a network design problem under uncertainty in a multiactor setting.

2 | MODELS AND METHODS FOR NETWORK SYSTEM DESIGN

In the previous section, we established the need, from a systems engineering perspective, for an accurate, yet quick optimization method for network topologies for infrastructures. We therefore conducted a thorough review of recent literature from the operations research and systems engineering fields, and this revealed operations research papers on the design of new energy networks, like CO_2 networks[15–21] or heating networks,[22–24] but also new networks to transport hydrogen,[25] animal waste,[26] water,[27–31] or oil.[32,33]

Table A2 shows a short summary of the optimization problems discussed in the papers 15–33 and the methods used to solve these problems. However, none of these papers incorporates the system design variables in the way that is core to our paper: exploring systems options that address the development of a network with uncertain sources and sinks, with an undetermined topology, and executed in a multiactor setting. The papers do contribute various interesting (parts of) answers on how to model particular design variables and constraints that span up our trade space, and how to solve the design problems. We discuss them below.

2.1 | Design variable: system topology

Most authors design the network layout by minimizing investment costs, although some take into account the operational costs on the longer term as well. Only Bietresato et al,[26] Ivić et al,[31] Liu et al,[33] and Steele et al[30] search for networks of minimum length. Naturally, the investment costs depend to a large extent on this length of the pipelines to be built. If connections can only split in a given node (source or sink), then the design of the network is given by the *minimum spanning tree* (MST). This tree can easily be determined by Kruskal's[34] or Prim's[35] algorithm. Our systems engineering problem does allow the introduction of new splitting or connection nodes, thereby vastly increasing the trade space, so these algorithms do not suffice.

Other approaches that look for minimum-length networks do allow extra splitting points to make the network shorter. When these newly introduced nodes can be located on optimal positions, they are called *Steiner points* as they were first defined by Jacob Steiner. The minimum length tree is then called a *Steiner minimal tree* (SMT).[36] For a general number of nodes, the ratio between the length of the SMT and the length of the MST has a lower theoretical bound equal to 0.82.[37] It therefore makes sense to find the Steiner tree when looking for a minimum cost system. Only Bietresato et al[26] and Liu et al[33] search for this SMT to reduce the overall cost of the system.

2.2 | Design variable: link capacity

However, in general, the investment costs do not only depend on the length of the new pipelines, but also increase when larger pipeline capacities are needed. The extra capacity costs can be taken into account by using a cost function that depends both on the length and the capacity. This capacity cost function is often formulated as a concave function $f(q)$ with q the capacity of the pipeline.[15,17,25,36,38–40] Some, however, use only pipelines of specific discrete sizes and adapt their cost function accordingly.[27,28,30,32,41,42] Zhang and Zhu[41] first determine the pipeline capacity on a continuous scale after which they round it to one of the available sizes, in order to allow the use of an Non-Linear-Program (NLP) solver.

In this paper, we use the continuous concave function[40] $f(q) = q^{\beta}$, $0 \leq \beta \leq 1$. If $\beta = 0$, $f(q)$ indicates that the capacity does not influence the investment costs. In that case the problem translates to finding an SMT. If $\beta = 1$, there is no cost reduction (ie, economy of scale) for combined pipelines. The function[17] for costs of pipelines in the CO_2 network corresponds to a β-value of around 0.6. The same holds for the cost function[25] for a hydrogen network. The discrete cost table for the

pipelines in a water network[28] is best fit with β around 0.7. In most practical cases, β will probably be somewhere around 0.6.

If the capacity does not influence the investment cost, then there is no difference in minimum-cost network between a multisource multisink and one-source multisink case. Although demand or supply requirements will also in that case determine the needed capacity of the pipelines, these capacities have no influence on the final investment costs. If capacity does play a role in the investment cost, then this will also influence the optimal network topology. The models in the literature do not endogenously take into account this effect of capacity increases or decreases of links as a result of adding new nodes.

2.3 | Design variable: node function

The demand and supply functions can be formulated in different ways. Thomas and Weng[40] and Trietsch[38] determine in advance for each pair of network nodes the flow between these nodes that the network should be able to transport. They do not explicitly distinguish between the role of the nodes, sources, or sinks. Because of this, the minimum-cost network could be no longer a tree (or forest) but could also contain cycles. These networks are called *Gilbert networks* (GNs)[40] or G-Steiner trees[38] as they were first mentioned by Gilbert[36] and Gilbert and Pollak.[43]

In our paper, we will assume an explicit division between sources and sinks, but we do not require flows between specified pairs of nodes. We only require that the network should be able to satisfy the minimum demand from sinks and/or the maximum supply from sources. This requirement is often used in multisource multisink networks to determine the optimal flow through an existing network[44,45] but as far as we are aware, it has not been used to *design* a minimum-cost network. We assume that this is a realistic requirement for new infrastructures, and will therefore include it into our system problem formulation.

2.4 | Design constraint: operations and regulation

In many infrastructures, the network and its operation are regulated. The supply or demand itself often is not regulated or only partly. For example, in the case of the CO_2 networks, regulation may be applied to the price setting of the sinks and of the network, as these will be monopolies or oligopolies at best. For offshore grids or biogas networks, the networks are regulated for the same reasons: they constitute a monopoly. There might also be requirements on minimum quantities to be delivered when, for example, biogas producers go into a contract with gas distributors and resellers. Large wind farms would typically contract out a certain capacity to the market, and any under- or overestimations of that power (due to wind fluctuations) would be bought or sold at the spot market or other reserve markets. Existing methods in literature have practically disregarded these set of constraints for large networked system designs. In developing our algorithm and model for solving the network topology problem, we have ensured that such constraints can be implemented via the (un)certainty profiles of the sources and sinks.

2.5 | Solving approaches

In general, there are three main approaches to solve networked system design problems. The first one uses heuristics and algorithms from graph theory and geometry[17,25,26,28,39,40,42], like the Gilbert-Melzak method[46–48] to find the optimal topology and the optimal location of Steiner points (added nodes to minimize path length).

The second approach formulates the problem as an Mixed Integer (Non-)Linear Program (MI(N)LP).[15,20–22,30,32,41,45,49] They find (sub)optimal networks using existing solver tools like modeling system for mathematical programming and optimization.[20,30,45] Thapalia et al[45] focus on the allocation of total supply to each demand node without being able to design the network in terms of locating Steiner points in the network topology. Mostly demand nodes are connected directly to one or multiple supply nodes.

A third approach uses agent-based models, more specifically ant colony optimization (ACO) or particle swarm optimization (PSO), to find (sub)optimal network layouts. For example, Maier et al[27] presented an ACO algorithm for the design of water distribution systems and Arnaout[50] further generalizes the ACO algorithm to deal with network infrastructures with an unknown number of elements. Heijnen et al[10] developed an ACO for the one-source multisink problem and Nguyen[51] extended this model to the multisource multisink case. Liu et al[33] use PSO to find an SMT.

In this paper, we will formulate the multisource multisink network design question as

> *How to find a minimum-cost network topology that connects sources and sinks with sufficient network capacity while guaranteeing that all demand of the sinks can be delivered by the supply of the sources?*

Our algorithm will follow the geometric graph theoretical approach, as this is computationally fast, and is comprehensible for actors and systems engineers in infrastructure design settings. After we have explained our approach in more detail in the next chapter, we discuss a few related approaches from the literature in Section 3.4.4.

3 | NETWORK DESIGN ALGORITHMS

For the design of an energy network with uncertain sources and sinks, in line with the literature explored above, we need to develop a cost-efficient network of connecting sources and sinks, and allowing the introduction of new splitting of connecting nodes.

3.1 | General approach

We describe our general approach on the basis of an example for the development of a biogas infrastructure. In many rural areas, farmers operate manure and organic waste digesters to turn the waste into biogas. This biogas is partly reused on the farm, but the overproduction can be fed back into the main gas grid. However, in order to reach this main gas grid, the farmers need to cooperate and build a cost-efficient network that connects their multiple sources (the digesters) to one

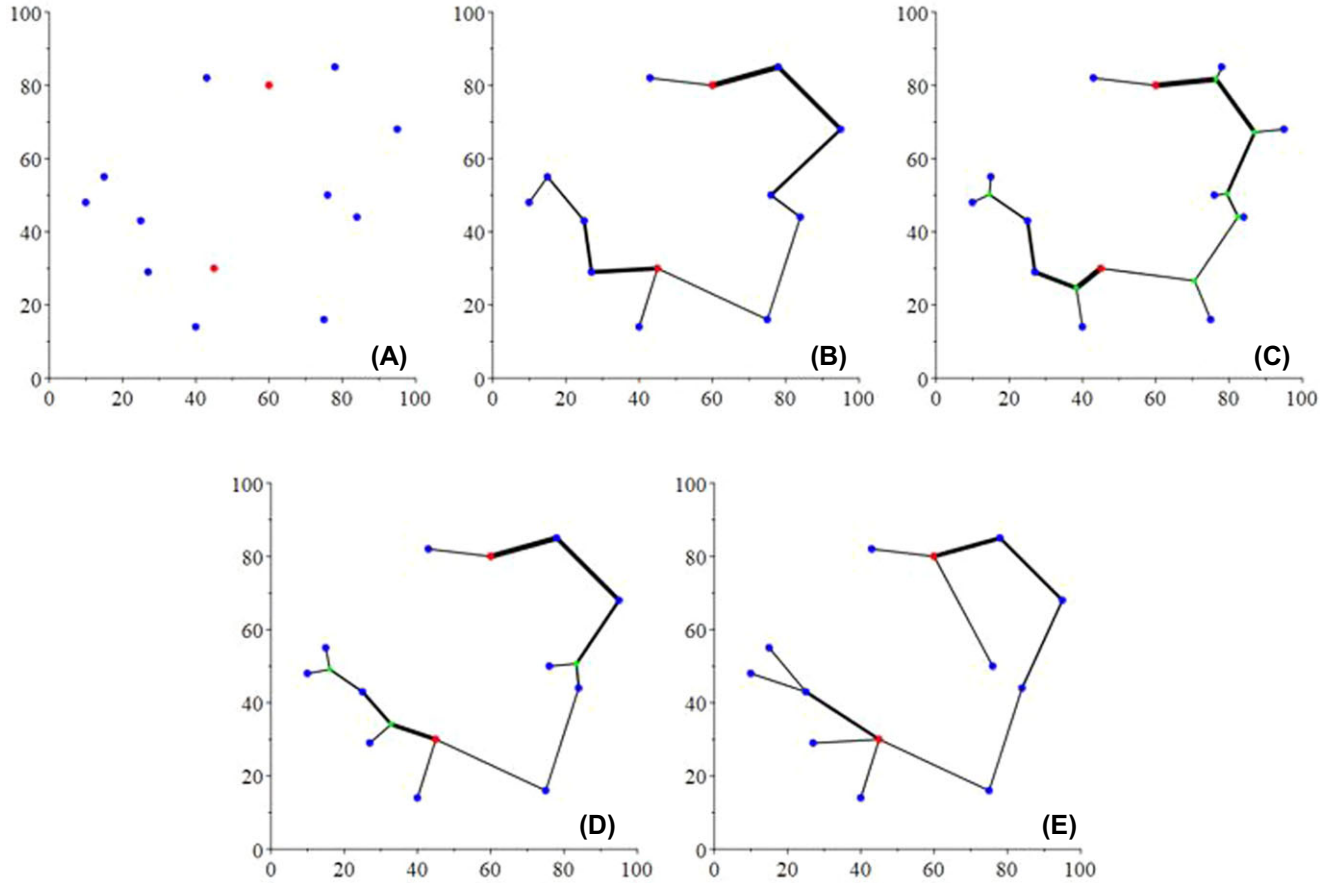

FIGURE 1 Example with two sources (red, supply per source: 11) and 11 sinks (blue, demand per sink: 2), respectively, (A) with no pipelines, (B) minimum spanning tree, (C) $\beta = 0$, (D) $\beta = 0.5$, and (E) $\beta = 0.99$

sink (the main gas network). In recent real-world cases, the designer of such network would make an inventory of the sources and their locations, and would then develop a topology for a shared network or grid, based upon expert estimations of the most efficient layouts. However, adding additional collection points for the biogas, before transporting the gas further downstream to the main gas pipeline is often disregarded. When the topology is constructed by such qualitative approaches and negotiations among farmers, adding such additional collection points would make the design space of the design problem too large to handle manually. However, such additional collection points would ultimately lead to a more cost-effective network as the total length of the pipelines would typically decrease.

In our approach, we use graph theory to construct a *GN*:[40] this will be the shortest length network (pipelines to be laid out by farmers), by allowing the introduction of new nodes (new biogas collection points). The following elements are taken into account in our algorithm to develop minimum-cost networks:

- A GN is a network *G* that connects a given set of terminals, satisfying given flow demands $q_{[i,j]}$ from node *i* to node *j* and has a cost function.
- We include β as a design variable, which represents the capacity cost exponent, a proxy for economies of scale in the investment cost of a pipeline.
- GN or trees can contain Steiner points, these are the "splitting/collection points" that would be added to create a more cost-effective network.
- The size of the flows among the different nodes is included in the design space.

This is a very challenging set of design variables and constraints for minimizing the cost of the network, for which we developed an effective and efficient solving algorithm in the remainder of this section. Figure 1 shows how such networks may look, given this algorithm. Figure 1A shows an example with two sources that can both supply 11 m^3/s and 11 sinks that all need 2 m^3/s. Figure 1B shows the MST to connect both sources with all sinks. The thickness of the network pipelines is relative to the required capacity of the pipelines in order to satisfy all demand, although extra cost for capacity is not taken into account in determining this tree ($\beta = 0$). Figure 1C shows the network when extra points can be included to reduce the total length of the network. Capacity costs are still not incorporated by setting $\beta = 0$. It is clear that the network length and by that the investment costs can significantly be reduced by adding these extra splitting points. Figure 1D shows the result for a medium influence of the capacity on the investment costs ($\beta = 0.5$). The effect of incorporating these costs is that higher capacity, ie, more expensive, pipelines are now shorter in favor of

lower capacity, ie, cheaper, ones. This effect is even more evident when β is increased to 0.99 in Figure 1E. In that case, it is hardly profitable to combine pipelines going to different sinks and most pipes connect a source with a sink in the shortest way. The following sections explain how the algorithm combines our design variables to span up our trade space and how it then finds the cost minimum solution in this space.

3.2 | Objective: finding a minimum-cost Gilbert tree

Crucial for this design problem is that we include the volume of the flows among the different nodes in the design space, and this becomes part of our design variables; as we explained in the previous section, this is crucial, but has not been done yet in the literature. Because we include this design variable into our problem, our problem translates mathematically to finding a so-called *minimum-cost Gilbert tree* (MCGT). Our algorithm intends to find this particular tree. We first explain this challenge.

A *GN*[40] is a network *G*, not necessarily a tree, which connects a given set of terminals, satisfying given flow demands $q_{[i,j]}$ from node *i* to node *j* and with a cost function

$$C(G) = \sum_{e \in E(G)} l_e f(q_e), \tag{1}$$

in which $E(G)$ is the set of all edges in G, l_e is the length of edge e, q_e is the capacity of e, and $f(q)$ is a nonnegative, nondecreasing, triangular function on the capacity q. The network can contain Steiner points. If the network has a tree (or forest) topology, we will call it a *Gilbert tree* (GT). The *minimum-cost GN* (MCGN) is the network among all possible GNs with total minimum costs.

Our problem is a generalization of the MCGN problem described by Thomas and Weng.[40] We use the continuous concave function $f(q) = q^\beta$, $0 \le \beta \le 1$; however, where they set in advance the required flows between each pair of nodes, we only set the demand and supply of the nodes, but leave the decision about which source will deliver which sink as part of the decision problem. If we make this decision upfront, our problem directly transforms into theirs, see Figure 2.

Thomas and Weng[40] make no difference between sources and sinks, but they only require a network with sufficient capacity to supply the flow requirements between each pair of nodes. In that case, for higher capacity cost exponents β (near 1), it might be profitable to build a complete network with a connection between each node pair. Moreover, their MCGN might contain cycles. In our problem, however, capacity is only needed from sources to sinks. Given that, a network that minimizes the investment costs (1) will always have a tree topology, as stated in Theorem 1.

Theorem 1. *For every feasible network topology G satisfying the constraints* (B.2)-(B.4) *from B.1, there exists a GT(or forest) topology T with less than or equal costs of G, where the costs are defined by the cost function* (B.1) *in 0.1.*

A proof of this Theorem can be found in 0.3.

With Theorem 1, our problem translates to finding an MCGT (or forest) given the edge weights $l_e q_e^\beta$. Thomas and Weng[40] indicate that they do not know about any efficient algorithm for solving this problem. They suggest to use the Gilbert-Melzak method[36] for finding a suboptimal solution together with a global optimization technique.

FIGURE 2 Two possible configurations to connect sources [A,B] (with given supply [1,2]) to sinks [C,D] (with given demand [1,2])

3.3 | Improving networks: the adapted Gilbert-Melzak method

Our approach will scan the solution space by using different promising STs as a starting point. We apply our method, the adapted Gilbert-Melzak method, to each of the starting points. Because the adapted Gilbert-Melzak method is a deterministic method, for a single starting tree, we get to the same (suboptimal) MCGT. In this section, we describe this method. Afterward, in Section 3.4, we describe the different starting points we use.

In Heijnen et al,[52] we gave a short explanation of the Gilbert-Melzak method, as we used it for one-source multisink networks. In this paper, we will adapt the method to use it for the multisource multisink case. For the illustration here, we start from an initial noncrossing ST that connects all terminals and had the required capacities as weights on the edges. This initial tree was then used to search for improvements. The same approach is used for multisource multisink networks as long as the initial ST satisfies the constraints (B.2)-(B.4) defined in B.1. Each change in the tree needs to preserve these constraints.

In each step, the current tree (or forest) is locally changed as long as a network with lower total cost is found. The improvements are based on the observation that it is profitable to partly join two adjacent connections if their interior angle is small, ie, smaller than the angle constraint.[40] This angle constraint depends on the value of β. See, for example, the MST in Figure 1B. Almost all angles are small, indicating that when extra costs for capacity are not so high, ie, β is near 0 (Figure 1C), investment costs can be reduced by combining pipelines. Which will not be the case if β is high (Figure 1E).

If the angle satisfies this constraint, adding a splitting point, ie, Steiner point, *S* will lower the total costs of the network. Using a geometric approach, the optimal location (see Figure 3) of these new Steiner points can exactly be determined (see Heijnen et al[52] for more details). The procedure guarantees that finally, a local minimum is found, since the total investment costs decrease after each network change. The angle with the largest deviation from the constraint value is solved first.

After each local network change, the required capacities of the edges are set again to calculate the overall investment costs for the new network.

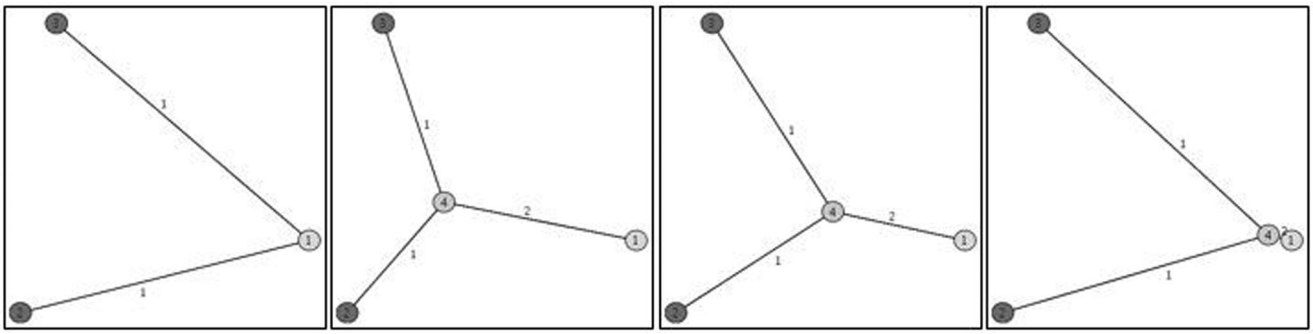

FIGURE 3 Angle between one source (1), two sinks (2,3), and the optimal location of Steiner point (4) when $\beta = 0, 0.5, 0.8$, respectively

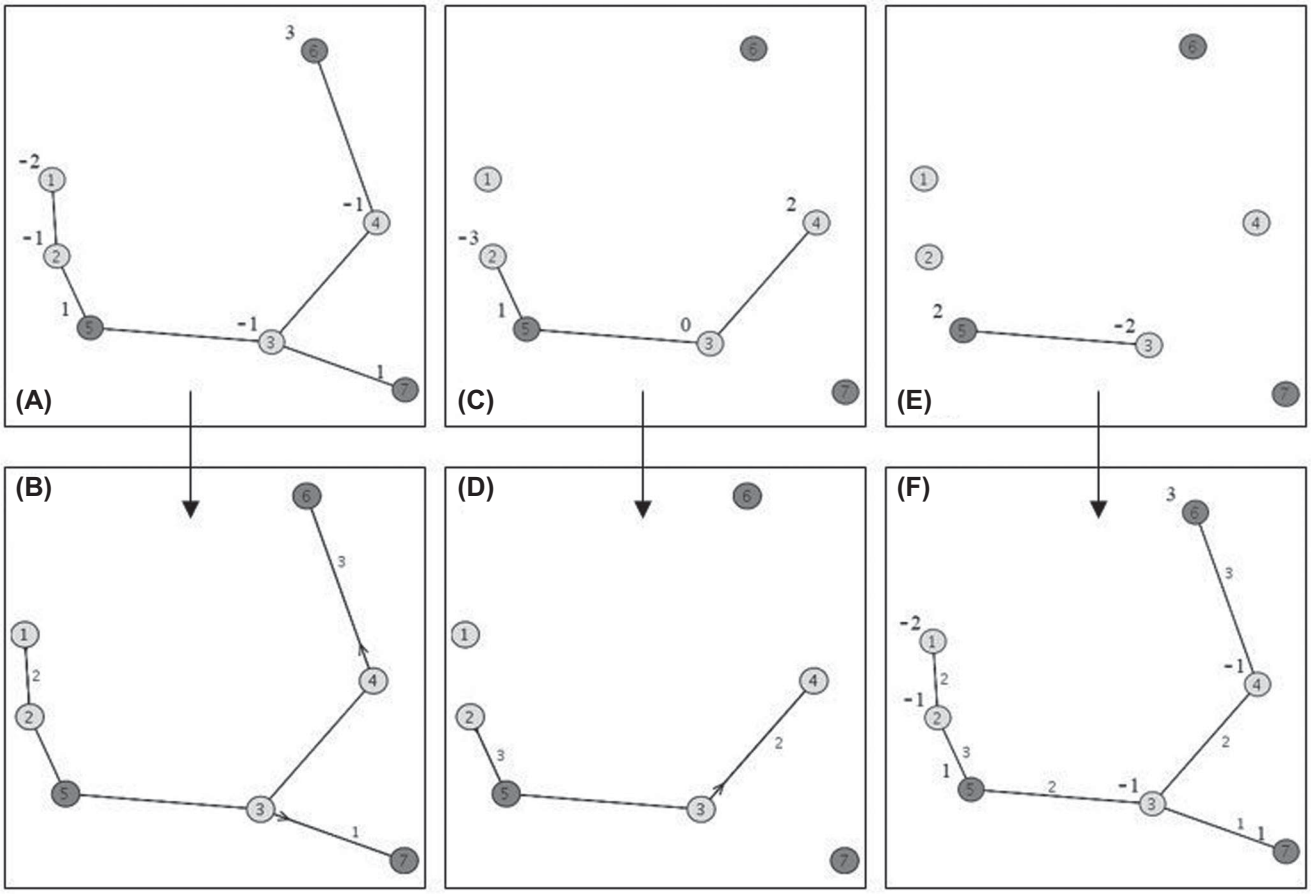

FIGURE 4 Capacity assignment procedure for spanning tree with four sources (nodes: 1-4) and three sinks (nodes: 5-7)

The original procedure of the one-source multisink network needs some adaptations to be applicable on a multisource multisink network. The main one is the fact that the direction of an edge can change if a sink, after a local network change, is supplied by a different source. If capacity needs in the same edge have opposite directions, they are subtracted and only the net capacity will be assigned. The mathematical specification of the adapted procedure to assign capacity to the edges is explained in 0.4. Figure 4 shows the working of this procedure with a small example of four sources and three sinks. The supply of the sources and the demand of the sinks are denoted at the upper left of the nodes. To start off the procedure, the supply is defined as a negative value (Figure 4A). The procedure starts by selecting the leafs of the tree and add the required capacity to the incident leaf edge (Figure 4B). These edges are removed and the supply and demand of the new leaf nodes are updated with the capacity of the removed edges (Figure 4C). The procedure is repeated until all edge capacities are set (Figure 4F).

3.4 | Initial starting points: various spanning trees

The method in §3.3 describes how to improve the network from a particular starting point. To improve the performance of the method as a whole, we scan the solution space by using different promising STs as

starting point. The importance of developing a good starting point is shown first:

Theorem 2. *Every GT topology* S *on* n *terminals can be found by applying the adapted Gilbert-Melzak procedure on a specific ST* T *on these n terminals.*

The proof of Theorem 2 can be found in B.5. The theorem proves that when the algorithm does not find the MCGT, this is not the fault of the adapted Gilbert-Melzak method, but because the algorithm uses an incorrect ST as starting point.

> *The selection of initial spanning trees is therefore key to obtain optimal results in the end.*

The total costs of the final network are determined by the edges in the final GT (or forest), the location of the terminals and (eventually) Steiner points, the capacity of the edges, and the capacity cost exponent. Under the assumption that *lower cost spanning trees will in general lead to lower cost Gilbert trees*, it might be profitable to start from lower cost STs. This assumption is verified in 0.6 by generating all possible STs and their corresponding GTs for small instances of *N*, the total number of terminals.

The results reveal that while the costs of different STs show a high variation, this variation is significantly reduced in the costs of the resulting GTs. For these relatively small-scale examples, we draw the conclusion that many different STs in general result in a much smaller set of GTs after the adapted Gilbert-Melzak procedure is applied and that most STs result in an MCGT (see the peak at 1 on the *x*-axis in Figure B.2B). Moreover, the results show that lower cost STs have a higher probability to end up in a low-cost GT (correlation coefficient: $\rho = 0.652$, $p = 0.000$).

Given the previous observation, our goal is now to find low-cost STs as an input for the adapted Gilbert-Melzak method.

> *Although simple algorithms can find a minimum weight spanning tree if all weights of the possible edges are known, in our case the weights are only known when the full spanning tree is generated since required capacities depend on the structure of the tree.*

There are, however, some special STs that might have low investment costs in special cases.

3.4.1 | Minimum spanning tree

When the capacity cost exponent $\beta = 0$, ie, when only the edge length influences the investment cost, the *MST* is the lowest cost ST. Figure 5A gives an example of this tree.

3.4.2 | Hub network

On the other hand, when the capacity cost exponent $\beta = 1$, ie, parallel edges of capacity 1 are just as expensive as one edge of capacity 2, direct edges between sources and sinks result into the lowest costs.

In a one-source multisink network, this will lead to a star topology with the source functioning as hub. In a multisource multisink network, there are multiple stars when all sources are directly connected to all sinks. To reduce costs, priority is given to the shortest length edges between sources and sinks. If the capacity of a source is not sufficient to supply the nearest sinks, connections to the next nearest sink are also needed. We call this network the *hub network* (HN). Figure 5B gives an example of the HN.

3.4.3 | Minimum-cost spanning tree

If β is low, a minimum-cost ST (MCST) will probably resemble a minimum length ST and if β is high, it will probably have more direct connections between sources and sinks. We use this assumption to propose an intermediate variant, starting from the minimum length ST and using only simple *edge turns* defined as:

1. remove an edge e from the tree;
2. connect the two cuts of the tree by a new edge that has one end point with edge e in common;
3. adapt capacities of the edges where needed (using the capacity procedure in B.4);
4. calculate the total costs of the new tree; and
5. remember the costs and the tree if the costs are lower than before.

Repeat the steps 1- 5 for all edges in the tree and select the best tree so far. The procedure is repeated as long as trees with lower costs can be found. We call this tree the minimum cost spanning tree (MCST) *MCST*. Figure 6 gives an example of this procedure.

Note that even in case $\beta = 0$, the MCST does not have to be equal to the MST, since the MCST can also be a forest instead of a tree, which the MST cannot.

3.4.4 | Benchmarking system designs

From the literature overview in Section 2, we found some papers discussing similar problems to the one in this paper. We shortly discuss these and describe to what extent our approach presented above would or would not be able to use the results of those papers for benchmarking against our system design outcomes.

Stauffer et al[19] use SimCCS to design a CO_2-network for the Ordos Basin in China. The discussed case has 38 CO_2 sources and nine sinks. Detailed information is available of the sinks, but unfortunately not of the sources. They also use predefined potential routing taking into account geographical possibilities and restrictions for the network. The optimization considers emission penalties. Pipelines are only built when they can earn themselves back. The results can therefore not directly be used to evaluate the system design that would result from our method.

Boavida[16] describes a CO_2 network for Spain, Portugal, and Morocco. This network connects 288 sources to 163 sinks. Detailed information is, however, not available. The same holds for the CO_2 network described in Ref. 15.

We found two papers for which the cases might not directly be used with our method, but that describes the proposed algorithms clearly,

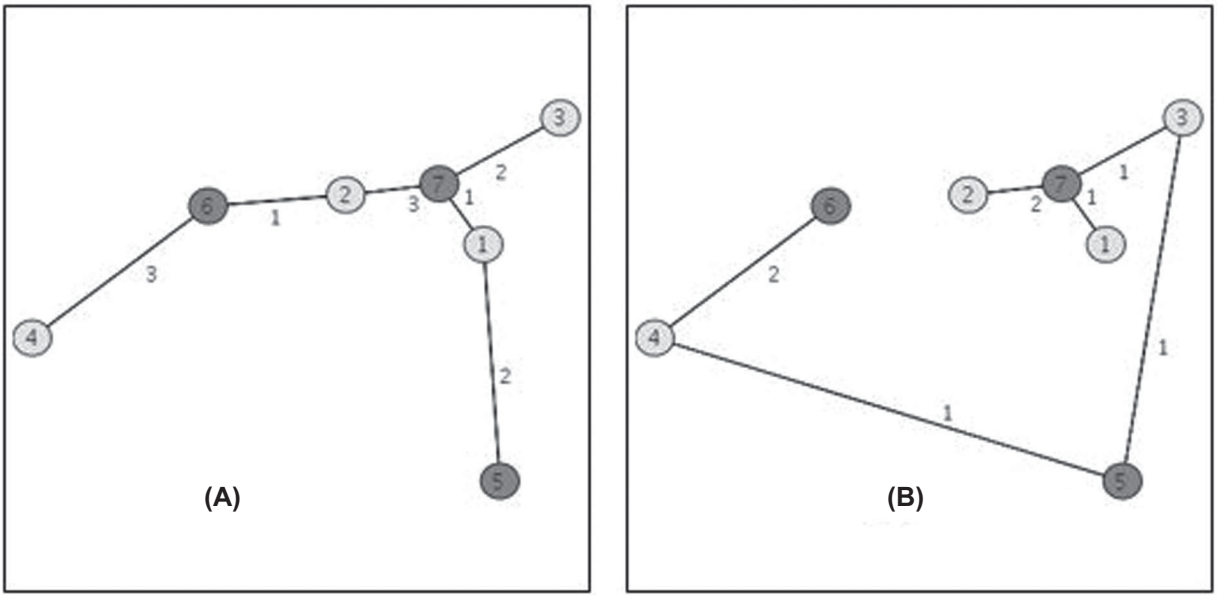

FIGURE 5 Minimum spanning tree (A) and Hub network (B) of a random example with four sources (nodes: 1-4) and three sinks (nodes: 5-7)

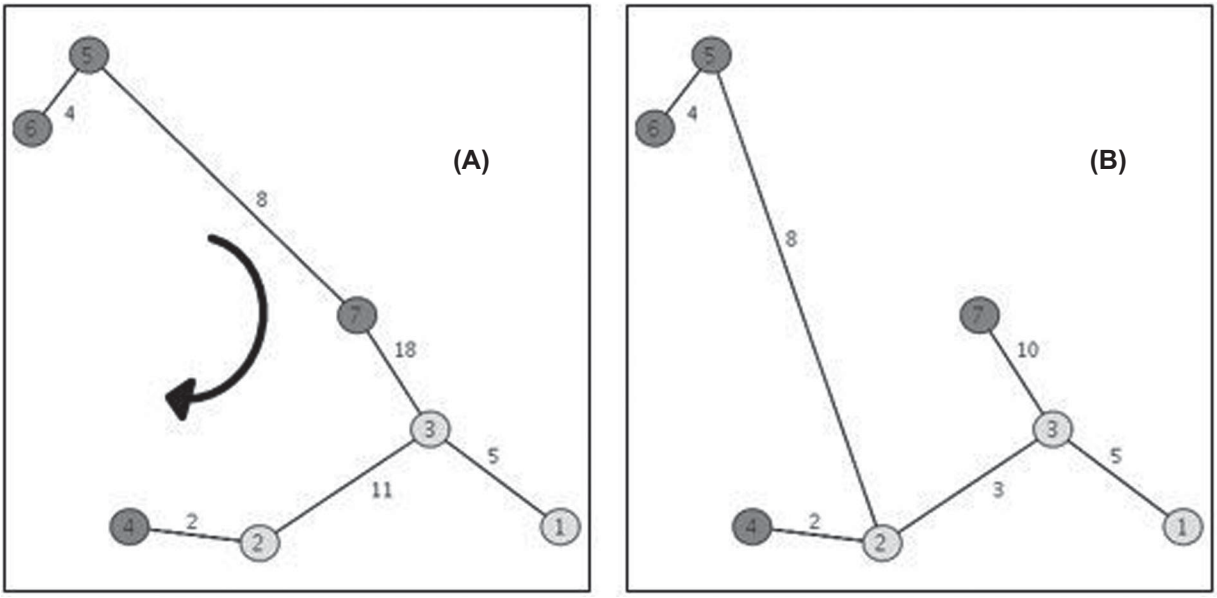

FIGURE 6 Example of a profitable edge turn in MST (A) to find MCST (B) with three sources (nodes: 1-3) and four sinks (nodes: 4-7)

so we can easily implement and use them on our own case studies (see the next section).

- Kazmierczak et al[17] describe a heuristic to determine the optimal layout of a CO_2 network. They consider all possible pipelines and add them one-by-one to the network on the basis of minimal additional costs. They also add splitting nodes to make use of existing paths. This method can be used to benchmark our results.
- André et al[25] develop a Delta change algorithm that, starting from an MST, changes edges searching for a cheaper network. This procedure does not make use of extra splitting nodes, like Steiner points. This method might lead to a good initial ST for our approach.

The comparable methods from the literature discuss CO_2 networks. However, the methods are not restricted to these types of networks. We did not find useful equivalent methods applied on other type of networks. None of the available literature allowed us to benchmark the outcomes of our algorithm against theoretical or other cases. We therefore set out to test the performance (efficiency, efficacy, and usability) of our algorithm on our hypothetical and real-world cases. The results are reported in the next section.

4 | METHOD PERFORMANCE RESULTS

This section shows the performance results from our method. First, we will test the efficiency of our method in Sections 4.1-4.3. Since the MCGT-Problem is NP-hard, no optimal solutions can be determined within polynomial time. To analyze our method, we will compare our results with computable optimal layouts for very small problems. We will also apply our method on different initial STs to compare the results. Moreover, we will use the two alternative optimization techniques from literature.[17,25] In Section 4.4, we will test the efficacy of our method for an empirical systems engineering problem: the development of the biogas network in a real-world setting. Finally, in Section 4.5, we illustrate the usability of our method by describing the outcomes of a number of systems engineering student groups that used the method in various systems design problems. These students were relatively unfamiliar with the underlying math but were reasonably experienced in working with different systems engineering models and approaches in multiactor decision-making processes.

4.1 | Method performance for small problems

For small problems, the optimal solution can be found by generating all possible Steiner tree topologies, adding the minimum required capacities to the edges, locating the Steiner points to their optimal location using the adapted Gilbert-Melzak method and calculating the total costs.

We use again Prüfer sequences (see B.6) to efficiently generate all possible Steiner tree topologies using the following characteristics:

- All Steiner points have a degree equal to 3.
- A Steiner tree with N terminals can have at most $N-2$ Steiner points.

To obtain all possible Steiner tree topologies, we generate all Prüfer sequences L with the following characteristics:

- $L = [L_1, L_2, \dots, L_{N-2}]$ with $L_i \in \{1, 2, \dots, N\}$ for all topologies without extra Steiner points.
- $L = [L_1, L_2, \dots, L_{N+S-2}]$ with $L_i \in \{1, 2, \dots, N, N+1, \dots, N+S\}$ and $|\{i|L_i = N+j, L_i \in L\}| = 2$, $j, 1 \le j \le S$ for all topologies with S, $1 \le S \le N-2$, Steiner points.

The last characteristic guarantees that no more than $N-2$ extra points are added to the STs and that these extra points have a degree equal to 3.

We randomly generated 25 examples with four nodes of which two are sources and 25 examples with five nodes of which two or three are sources.

Conclusion: For all these examples, our algorithm starting from the different initial STs found the optimal solution in less than 1 s, which is, on average, 1% of the time needed to check all possible Steiner topologies.

TABLE 1 Characteristics of the 100 random examples

Characteristic	Notation	Range
Total number of nodes	N	[7,..,15]
Total number of sources	S	[2,3,4]
Capacity cost exponent	β	[0,..,0.9]
Total demand	D	[8,..,138]

4.2 | Method performance for many larger examples

To obtain statistically significant differences (if any), we generated 100 examples randomly with the characteristics as described in Table 1 and applied both methods from literature[17,25] and our method on these examples. For a fair comparison, we used our cost function, as defined in (B.1) in B.1, in all three methods.

Table 2 shows the results on both the average relative costs and the average computational time. To give an idea about the spread, also the standard deviations are given. To make the results of the different examples comparable, we use the minimum-cost tree of each example as a reference tree with costs $C_{0,j}$ for case j, $1 \le j \le 100$. The relative difference $\delta_{i,j}$ of the costs $C_{i,j}$ of one of the other trees is then calculated by

$$\delta_{i,j} = \frac{C_{i,j} - C_{0,j}}{C_{0,j}}. \tag{2}$$

We generated the lowest cost GT on the three initial STs: MST, HN (HN), and MCST. Intermediate and final results are given in the first three rows. The overall results of our method are given in row 4.

The delta change algorithm[25] found in 8 of the 100 cases a network layout with lower investment costs than the minimum costs of the three GTs found by our algorithm. The trees resulting from the algorithm in Ref. 17 found seven times the lowest cost tree. From our three initial STs, the MCST led 57 times to the best tree. The MST and the HN gave, on average, comparable results.

If it comes to the relative cost deviations, the network layout from the delta change algorithm was found to be on average 4% more expensive than the best tree found in our method. The network layout from Kzamierczak's algorithm was on average 5.9% more expensive than the best tree from our method.

For the three initial STs in our method, the minimum-cost ST is on average only 4.2% more expensive than the best tree found. The GT resulting from this initial tree has an average cost deviation to the best tree of 1.5%.

The computational time results show clearly that Kzamierczak's algorithm[17] is the most time-consuming one on average. Second worst is the procedure to determine the MCST. Although its cost results are very good, we pay a price for these results in computational time.

Given the fact that the results of the delta change algorithm[25] are comparable to the MCST and much faster, we will also use this tree (denoted by Δ-tree) as initial ST for our GT method to see if that leads to further improvements. To do so, the results on the 100 examples are recalculated and shown in Table 3.

TABLE 2 Results of 100 random examples. Relative costs compared to minimum-cost found in example

Trees	Best tree (out of 100)	Average relative costs		St.dev. relative costs		Average time (s)		St.dev. time (s)	
		ST	GT	ST	GT	ST	GT	ST	GT
Gilbert tree from MST	34	0.120	0.055	0.132	0.089	0.015	0.359	0.006	0.317
Gilbert tree from HN	33	0.427	0.056	0.363	0.078	0.005	0.553	0.007	0.356
Gilbert tree from MCST	57	0.042	0.015	0.038	0.026	1.269	0.279	0.954	0.249
Gilbert tree (Total)	**87**		**0.003**		**0.010**		**2.479**		**1.437**
Δ-tree[25]	8		0.040		0.058		0.564		0.346
Kzamierczak's method[17]	7		0.059		0.047		7.198		6.024

TABLE 3 Recalculated results with Δ-tree[25] as extra initial spanning tree

Trees	Best tree found (out of 100)	Average relative costs		St.dev. relative costs		Average time (s)		St.dev. time (s)	
		ST	GT	ST	GT	ST	GT	ST	GT
Gilbert from MST	33	0.123	0.058	0.141	0.095	0.015	0.354	0.006	0.316
Gilbert from HN	27	0.429	0.058	0.362	0.078	0.005	0.547	0.007	0.354
Gilbert from MCST	50	0.044	0.017	0.041	0.030	1.270	0.280	0.954	0.248
Gilbert from Δ-tree	60	0.042	0.061	0.057	0.049	0.564	0.311	0.346	0.260

TABLE 4 Parameter values for three different sets of 40 examples

	SET 1	SET 2	SET 3
# Examples	40	40	40
Nodes	5-44	5-44	5-44
β	0.4	0.6	0.8
Total demand	300	300	300
Sources	3	3	3

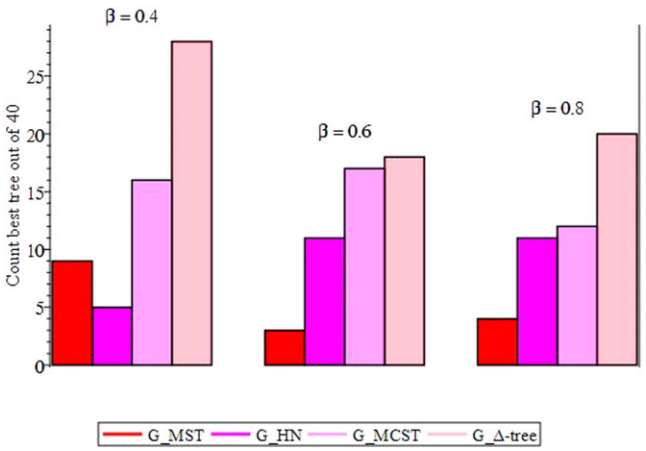

FIGURE 7 Bar chart of the number of times (out of 40) the best tree was found in sets 1-3, respectively

The results reveal that the use of the Δ-tree as an additional initial ST gives very good results, both in finding the best minimum-cost tree as in computational time.

In case of time restrictions, one might use only a selection of the initial trees to obtain final results. We analyzed the cost and time results for all different combinations of initial STs. The results (in C) show that a good selection would be to combine the HN with the Δ-tree.

4.3 | Method performance for large networks

Realistic networks might contain a multiple of the number of nodes that we used in the 100 random examples of §4.2. To investigate the scalability of the proposed method, we generated 40 random examples with an increasing number of nodes ranging from 4 to 44 for three different settings of the parameter β (see Table 4).

In the following analyses, we left out Kzamierczak's method[17] because of the lagging performance in the smaller examples and we applied our method on the four initial STs (MST, HN, MCST, Δ-tree) as defined before.

Figure 7 shows the number of times out of the 40 examples per set that the lowest cost GT was found from a specific initial ST. Clearly, also, for the larger networks, the Δ-tree as initial tree gives the best results. As could be expected, for lower values of β, the MST gives better results than for higher values of β, which is the opposite for the HN.

To show a possible relation between the results and the network size, Figure 8 shows a bar chart of the average relative costs compared to the best GT found where the results of the networks with 4-14 nodes, 15-24 nodes, 25-34 nodes, and 35-44 nodes are taken together. In this bar chart, lower bars reveal better results. Since the randomly generated examples might show a large variation based on the spread of the demand over the sinks and the location of the nodes, the results do not give an unambiguous interpretation on all aspects. However, we might draw the following conclusions with some caution:

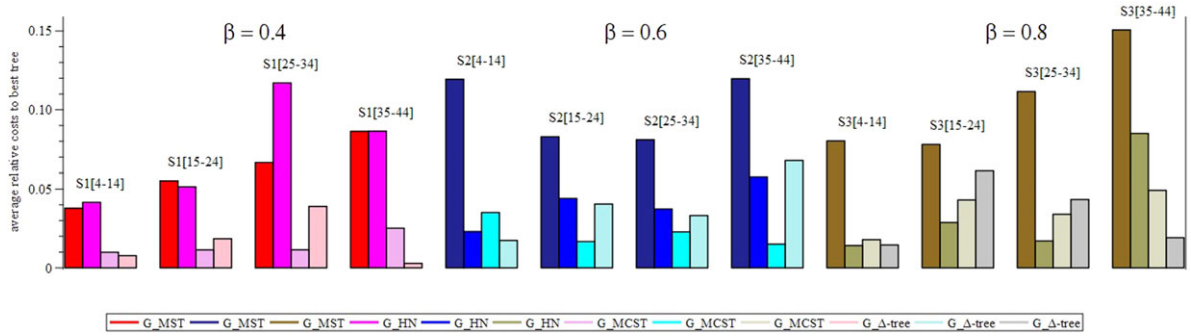

FIGURE 8 Bar chart of the average relative costs divided over different network sizes for sets 1-3, respectively

- For high values of β, the GT from the HN gives good results, specifically for smaller networks.
- The GT from the Δ-tree or from the MCST gives in general good results. No clear relation is found with the value of β or the network size.
- The GT from the MST gives in general not so good results compared to the others, although possibly a bit better for lower values of β.

Figure 9 shows the computational time for the different examples in the three sets. The time to generate the MCST grows exponentially with the size of the network and becomes worse for higher values of β. The computation time for the method applied on the other three initial STs does also exponentially grow with the size of the network but with much lower growth constants.

4.4 | Method efficacy for finding a biogas network

We will also apply all methods on the following real-world case (adapted from Heijnen et al[52]).

The region of Salland, the Netherlands, has the ambition to become fully CO_2-neutral by the year 2030. To achieve this goal, a biogas network has been proposed as one of the spearhead projects. The first question in this explorative phase of the project is the financial feasibility. Thirty-six farmers with a large manure supply might be willing to connect to the network if the proposal is profitable. The construction of the network encompasses roughly 50% of the cost component for this project. To progress toward the next phase, the project group requires an indication of the path of the pipelines and the associated costs. To apply our method, we assume that there will be two biogas plants part of the new network.

Figure 10A-C shows the best result found by our method and by the algorithms from Refs. 25 and 17, respectively. The red and blue nodes give the location of the biogas plants and the farmers, respectively. The green nodes show the location of the added Steiner points. The thickness of the pipelines is relative to their capacity.

From Table 5, the advantages of our method are clear. The best network is found by applying our method on the Δ-tree (Figure 10F). André et al[25] (Figure 10A) find a comparable network layout in short time, but since the method only makes use of STs and does not add profitable splitting points, the final costs can be expected to be higher.

TABLE 5 Different methods applied on a biogas network with 36 farmers and two biogas plants. Relative costs compared to lowest cost found

Method	Relative costs	Computational time (s)
Δ-tree[25]	6.1%	19
Kzamierczak's method[17]	4.9%	4821
Gilbert from MST	16.4%	3
Gilbert from HN	6.9%	3
Gilbert from MCST	0.7%	233
Gilbert from Δ-tree	0%	21

Kazmierczak et al[17] (Figure 10B) build the network step-by-step and allows the addition of splitting points. However, because they fix the pipelines from earlier steps, these splitting points will almost never be on an optimal position. Moreover, shorter, and by that, cheaper pipelines will be added first, forcing the network into a certain layout that cannot be changed afterward. Besides, the computational time for this method is relatively high, since in each step, it compares all possible new connections of our method, starting from three different starting trees (Figure 10C-E), which are locally optimized by adding profitable Steiner points, gives a high probability to find the low-cost network in a reasonable time. The differences in computation time are mainly caused by the time needed to generate the starting trees. The MST and the HN can be found in a very short time since the potential connections need to be sorted on their length only once. The MCST needs several iterative steps to reduce total costs and by that is more time-consuming. The best results are found when our GT method is combined with the method from André et al[25] (Figure 10F).

4.5 | Method usability

During 2017-2018, eight systems engineering student groups (of four students each) have used our method to explore and design an energy network (new or revamp), and emulate the multistakeholder setting of such systems engineering problem. This was done in the context of the design course in the master program of Complex Systems Engineering and Management of Delft University of Technology (course code: SEN1531). In this course, students were assigned with a systems engineering task in a complex multiactor environment, and they need to solve this problem by using multiple methods from the systems

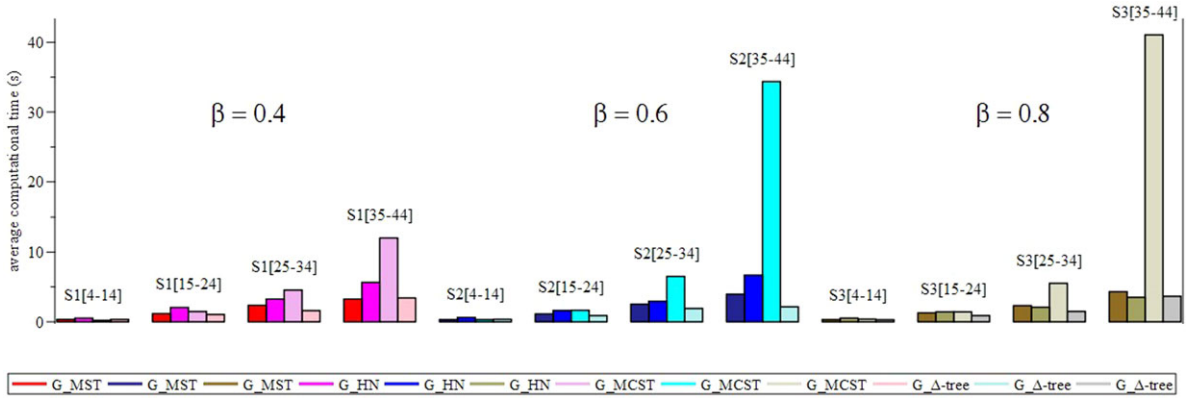

FIGURE 9 Bar chart of the average computational time divided over different network sizes for sets 1-3, respectively

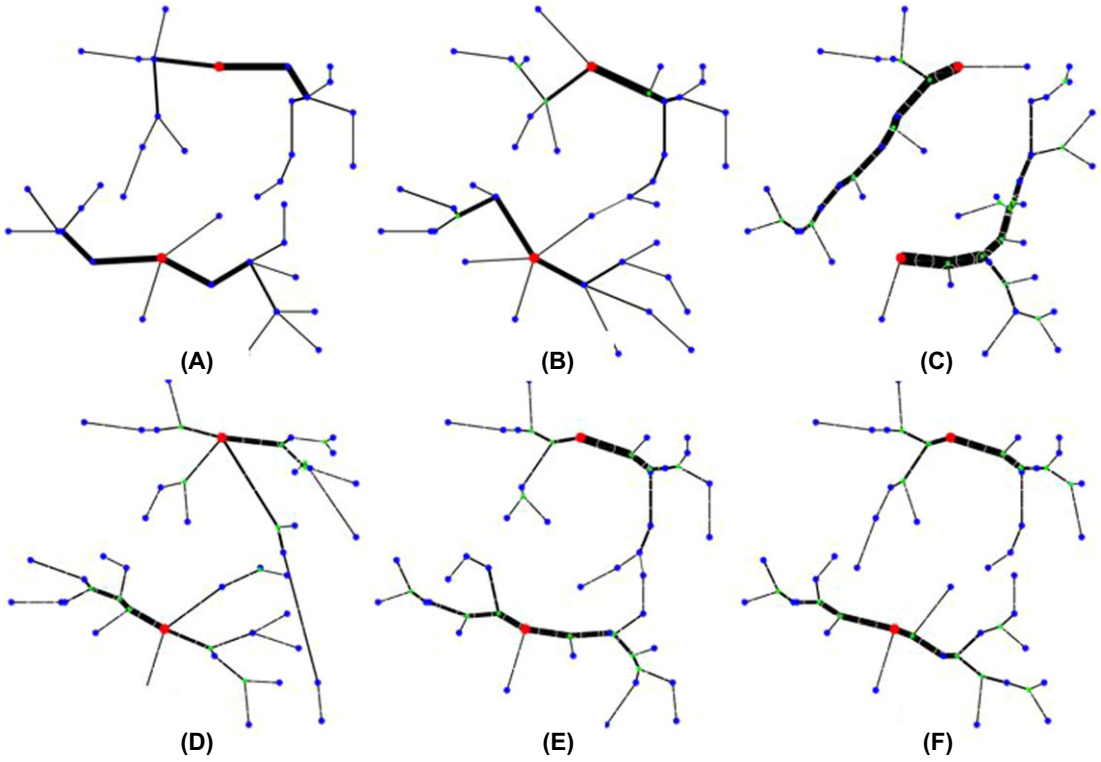

FIGURE 10 Best network layout from the methods: the delta change algorithm[25] (A), the heuristic algorithm[17] (B), our Gilbert tree method on minimum spanning tree (C), hub network (D), minimum-cost spanning tree (E), and Δ-tree (F)

engineering domain. The student groups who selected our method explored the following system design problems, showcasing that our method is widely usable for various network design problems:

- expansion of gas networks to accommodate shale gas exploration in the Netherlands,
- creation of power grids in rural India,
- hydrogen refueling infrastructure,
- district heating network,
- international electricity grid interconnections, and
- alternative urban heating systems (no more natural gas).

The specific network outcomes of these projects are not relevant for discussion in this paper, but the general observations from the application of the method by students confirm our claim that the method is easy to use and produces relevant results. In many cases, students realized more cost-effective (cheaper) networks than the ones they would develop by manually exploring the (vast) design space. Especially in "green-field" cases, students were unable to introduce new nodes for reducing system cost, without the support of our method. We also observed that students used the method as a transparent approach to find a theoretical minimum-cost target, before delving into other operational, institutional, or other stakeholder-related issues with colleagues in their group. Issues that would surface

in those social processes, and to which the network topology would need to be adapted in an iterative fashion, included, for example,

- operational constraints, eg, downtime issues due to (lack of) maintenance;
- power play of actors in the network, eg, blocking optimal location of splitting nodes;
- need for coordination and public acceptance, eg, the acceptance of shale gas networks;
- speed of infrastructure rollout in relation to required related developments, eg, hydrogen vehicle adoption;
- speed of innovation of competing technology, eg, heat pump technology; and
- market design, eg, transmission fees.

A very interesting finding was that students would not revert to trying to adapt or refine the algorithm in order to include these issues in solving the design problem, but rather would use the optimal networks as a starting point for the ensuing discussions. This intuitive approach kept the model and the solutions transparent, and helped to identify clearly the difference between rational cost-optimal solutions and modified solutions that would actually work and be accepted in practice. These observations strengthen our assumption, supported by complex decision-making literature, that many stakeholder and social issues should not be modeled endogenously into a model, but should rather be kept out in the open discussions with the stakeholders. Although our observations concerning our method were done for students in a learning situation, there is no reason to assume that this would not translate to actual real-world situations.

5 | CONCLUSIONS

The purpose of this paper is to develop a method to support infrastructure system design processes in multiactor contexts. In these multiactor settings with diverging stakes and interests, actors need to develop large complex sociotechnical systems of systems, scanning vast trade spaces under a plethora, or diverging assumptions. The systems engineering toolbox and theories point to the need to estimate the value of different design options, while problematizing such value determination in a multiactor complex setting. We showed that quite some developments point into the direction of more complicated models, trying to collapse social and engineering design variables and processes into one big model or framework. This paper begs to differ.

Our paper has shown that complex systems engineering processes do not necessarily require more complicated tools or methods to include all stakeholder values in one overarching model from the outset. On the contrary, we posit that when a fast and reliable straightforward method is used, this creates so much time and room for actors to explore the system on their own terms that the discussions will focus on the assumptions and parameters instead of the contested outcomes. Such discussions have shown to increase trust in the design process and may therefore yield (more) acceptable outcomes within acceptable times. This effect is confirmed in the complex decision-making literature.

From several contemporary examples in the infrastructure design field, CO_2, biogas, and offshore grids, we derived the need for quick, yet accurate, algorithms to find low-cost or minimum-cost network topologies. We also concluded that such algorithm should empower the different actors to bring in their assumptions and uncertainties independently as to explore the individual stakes and sensitivities in a shared network development project.

We presented an effective and robust method to find a minimum-cost layout for new infrastructures of multiple sources and multiple sinks following a geometric graph-theoretical approach. The location of these sources and sinks and their demand or supply are known. The approach determines the topology of the network, and the length and capacity of the pipelines to be constructed. This paper proves that a minimum-cost multisource multisink network will never contain a cycle and is therefore a MCGT (or forest), not a network. This MCGT can best be found by applying the adapted Gilbert-Melzak method starting from a well-chosen initial ST.

A core finding of this paper is that finding a very good candidate for the initial ST is crucial because the process is deterministic and having the best starting topology matters for the final results. We found several good candidates for this purpose. The MCST, found by making profitable edge turns improving the MST, gives good results but becomes computationally challenging for very large networks. The Δ-tree[25] also gives very good results as initial tree and can, in general, be found in less time. It is crucial, however, to use a set of starting trees to increase the probability to find the lowest cost network layout. A combination of the defined HN and the Δ-tree has shown to be Pareto-optimal with respect to minimal costs and minimum computation time.

Our approach outperforms existing methods on average with 4% to 6% of the costs of the final network. Even though this seems insignificant, the actual real-world money value can be significant since infrastructure development involves large budgets. More importantly, however, is that our approach is in general fast and finds a minimum-cost network at lower cost than conventional methods. This makes our method substantially more suitable to support a multiactor systems engineering process than any conventional method, as it can scan vast trade spaces in a fraction of the time.

Our proposed method is fast and gives good results, and in our future research endeavors, we will apply the method in real-world multistakeholder systems engineering setting. Our MCGT would be the target topology for a set of stakeholders to work with. Stakeholders could use this MCGT as an outlook for the minimum-cost network would give them an indication on what land additional hubs or splitting points would preferably be built and where most of the cost would go.

Using such relatively straightforward mathematical optimal design in a highly stakeholder-driven systems engineering process will lead to rationalization of the process, and could move negotiators and systems engineers away from "estimates" of experts who typically do not include additional hubs (Steiner points) in their informal designs. Our method therefore illustrates that multiactor complex design

challenges do not, as a matter of course, require complex, all-encompassing methods into which all values of different actors are collapsed. Instead, we have shown and underpinned that simple, straightforward methods may even be more useful in some of such cases. Our findings thus call for a serious discussion in the systems engineering discipline regarding the extent to which we should mathematically try to include all social phenomena into our models. Instead of the observed trend in the systems engineering community toward including and endogenously modeling all of these social science aspects and turning them, erroneously, into one-dimensional performance criteria, we should find a proper balance between mathematical modeling and actual stakeholder interaction.

ORCID

Petra W. Heijnen https://orcid.org/0000-0002-0028-6745

REFERENCES

1. Herder PM, de Joode J, Ligtvoet A, Schenk S, Taneja P. Buying real options – valuing uncertainty in infrastructure planning. *Futures*. 2011;43(9):961–969. https://doi.org/10.1016/j.futures.2011.06.005
2. VVD, CDA, D66, ChristenUnie. *Vertrouwen in de Toekomst, Regeerakkoord 2017–2021*. 2017. https://www.kabinetsformatie2017.nl/documenten/publicaties/2017/10/10/regeerakkoord-vertrouwen-in-de-toekomst.
3. Viebahn P, Chappin EJL. Scrutinising the gap between the expected and actual deployment of carbon capture and storage - a bibliometric analysis. *Energies*. 2018;11:1–45. https://doi.org/10.3390/en11092319
4. Melese Y, Lumbreras S, Ramos A, Stikkelman R, Herder P. Cooperation under uncertainty: Assessing the value of risk sharing and determining the optimal risk-sharing rule for agents with pre-existing business and diverging risk attitudes. *Int J Project Manage*. 2017;35(3):530–540. https://doi.org/10.1016/j.ijproman.2016.11.007
5. Dedecca JG, Lumbreras S, Ramos A, Hakvoort RA, Herder PM. Expansion planning of the North Sea offshore grid: Simulation of integrated governance constraints. *Energy Econ*. 2018;72:376–392. https://doi.org/10.1016/j.eneco.2018.04.037
6. Caunhye AM, Cardin M-A. Towards more resilient integrated power grid capacity expansion: A robust optimization approach with operational flexibility. *Energy Econ*. 2018;72:20–34. https://doi.org/10.1016/J.ENECO.2018.03.014
7. Sinha K, De Weck OL. A network-based structural complexity metric for engineered complex systems. In: *SysCon 2013 - 7th Annual IEEE International Systems Conference, Proceedings*; 2013:426–430. https://doi.org/10.1109/SysCon.2013.6549917
8. de Bruijn H, Herder PM. System and actor perspectives on sociotechnical systems. *IEEE Trans Syst Man Cybernet A: Syst Hum*. 2009;39(5):981–992. https://doi.org/10.1109/TSMCA.2009.2025452
9. Szajnfarber Z, Herder P. Highlights from CESUN 2016: Contemporary issues in methodological rigor for systems research. *Syst Eng*. 2017;20(6):481–482. https://doi.org/10.1002/sys.21416
10. Heijnen PW, Chappin EJL, Nikolic I. Infrastructure network design with a multi-model approach — comparing geometric graph theory with an agent-based implementation of an ant colony optimization. *J Artif Soc Social Simul*. 2014;17(4):1–23. http://jasss.soc.surrey.ac.uk/17/4/1.html
11. Garber M, Sarkani S, Mazzuchi T. A framework for multiobjective decision management with diverse stakeholders. *Syst Eng*. 2017;20(4):335–356. https://doi.org/10.1002/sys.21398
12. Simpson TW, Miller S, Tibor EB, et al. Adding value to trade space exploration when designing complex engineered systems. *Syst Eng*. 2017;20(2):131–146.
13. Frey DD, Herder PM, Wijnia Y, Subrahmanian E, Katsikopoulos K, Clausing DP. The Pugh controlled convergence method: Model-based evaluation and implications for design theory. *Res Eng Des*. 2009;20(1):41–58. https://doi.org/10.1007/s00163-008-0056-z
14. Frey DD, Herder PM, Wijnia Y, et al. Research in engineering design: The role of mathematical theory and empirical evidence. *Res Eng Des*. 2010;21(3):145–151. https://doi.org/10.1007/s00163-010-0085-2
15. van den Broek MA. *Modelling Approaches to Assess and Design the Deployment of CO2 Capture, Transport, and Storage*. Utrecht: BOXPress Oisterwijk; 2010.
16. Boavida D, Carneiro J, Martinez R, van den Broek MA. Planning CCS development in the west Mediterranean. *Energy Procedia*. 2013;37:3212–3220. https://doi.org/10.1016/j.egypro.2013.06.208
17. Kazmierczak T, Brandsma R, Neele F, Hendriks C. Algorithm to create a CCS low-cost pipeline network. *Energy Procedia*. 2009;1(1):1617–1623. https://doi.org/10.1016/j.egypro.2009.01.212
18. Weihs GAF, Wiley DE, Ho M. Steady-state optimisation of CCS pipeline networks for cases with multiple emission sources and injection sites: South-East Queensland case study. *Energy Procedia*. 2011;4:2748–2755. https://doi.org/10.1016/j.egypro.2011.02.177
19. Stauffer P, Middleton RS, Bing B, Ellett K, Rupp J, Xiaochun L. System integration linking CO_2 sources, sinks, and infrastructure for the Ordos Basin, China. *Energy Procedia*. 2014;63(63):2702–2709. https://doi.org/10.1016/j.egypro.2014.11.292
20. Sun L, Chen W. Development and application of a multi-stage CCUS source – sink matching model. *Appl Energy*. 2016;185:1–9. https://doi.org/10.1016/j.apenergy.2016.01.009
21. Middleton RS, Bielicki JM. A scalable infrastructure model for carbon capture and storage: SimCCS. *Energy Policy*. 2009;37:1052–1060. https://doi.org/10.1016/j.enpol.2008.09.049
22. Mertz T, Serra S, Henon A, Reneaume JM. A MINLP optimization of the configuration and the design of a district heating network: Academic study cases. *Energy*. 2016;117:450–464. https://doi.org/10.1016/j.energy.2016.07.106
23. Karschin I, Geldermann J. Efficient cogeneration and district heating systems in bioenergy villages: An optimization approach. *J Clean Prod*. 2015;104:305–314. https://doi.org/10.1016/j.jclepro.2015.03.086
24. Delmastro C, Mutani G, Schranz L. The evaluation of buildings energy consumption and the optimization of district heating networks: A GIS-based model. *Int J Energy Environ Eng*. 2016;7:343–351. https://doi.org/10.1007/s40095-015-0161-5
25. Andre J, Auray S, Brac J, et al. Design and dimensioning of hydrogen transmission pipeline networks. *Eur J Oper Res*. 2013;229(1):239–251. https://doi.org/10.1016/j.ejor.2013.02.036
26. Bietresato M, Friso D, Sartori L. An operative approach for designing and optimising a pipeline network for slurry collection from dairy farms across a wide geographical area. *Biosyst Eng*. 2013;115(3):354–368. https://doi.org/10.1016/j.biosystemseng.2013.01.008
27. Maier HR, Simpson AR, Zecchin AC, et al. Ant colony optimization for design of water distribution systems. *J Water Resour Plan Manage*. 2003;129(3):200–209. https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(200)
28. Gonçalves GM, Gouveia L, Pato MV, Marques Goncalves G, Luís G, Vaz Pato M. An improved decomposition-based heuristic to design a water distribution network for an irrigation system. *Annu Oper Res*. 2014;219(1):141–167. https://doi.org/10.1007/s10479-011-1036-7
29. Cozzolino L, Cimorelli L, Covelli C, Mucherino C, Pianese D. An innovative approach for drainage network sizing. *Water*. 2015;7:546–567. https://doi.org/10.3390/w7020546
30. Steele JC, Mahoney K, Karovic O, Mays LW. Heuristic optimization model for the optimal layout and pipe design of sewer systems.

Water Resour Manage. 2016;30:1605–1620. https://doi.org/10.1007/s11269-015-1191-8
31. Ivić S, Grbcic L, Druzeta S. Cooperative random walk for pipe network layout optimization. *Int J Appl Eng Res*. 2016;11(4):2839–2847.
32. Brimberg J, Hansen P, Lin K, Mladenovi N, Breton M, Brimberg J. An oil pipeline design problem. *Oper Res*. 2003;51(2):228–239. http://doi.org/10.1287/opre.51.2.228.12786
33. Liu Q, Mao L, Li F. An intelligent optimization method for oil-gas gathering and transportation pipeline network layout. In: *28th Chinese Control and Decision Conference (CCDC)*; 2016:4621–4626.
34. Kruskal JB. On the shortest spanning subtree of a graph and the traveling salesman problem. *Am Math Soc*. 1956;7:48–50.
35. Prim RC. Shortest connection networks and some generalizations. *Bell Syst Tech J*. 1957;36(6):1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x
36. Gilbert EN. Minimum cost communication networks. *AT&T Tech J*. 1967;46(9):2209–2227. https://doi.org/10.1002/j.1538-7305.1967.tb04250.x
37. Chung FRK, Graham RL. A new bound for Euclidean Steiner minimal trees. *Ann NY Acad Sci*. 1985;440(1):328–346. http://onlinelibrary.wiley.com/doi/10.1111/j.1749-6632.1985.tb14564.x/abstract.
38. Trietsch D. *Minimal Euclidean Networks with Flow Dependent Costs*. Discussion Paper, Northwestern University, 1985. http://ac.aua.am/trietsch/web/EuclideanNetworksMonograph.pdf.
39. Xue G, Lillys TP, Dougherty DE. Computing the minimum cost pipe network interconnecting one sink and many sources. *SIAM J Optim*. 1999;10(1):22–42. https://doi.org/10.1137/S1052623496313684
40. Thomas DA, Weng JF. Minimum cost flow-dependent communication networks. *Networks*. 2006;48(1):39–46. https://doi.org/10.1002/net.20118
41. Zhang J, Zhu D. A bilevel programming method for pipe network optimization. *SIAM J Optim*. 1996;6(3):838–857.
42. Awerbuch B, Azar Y. Buy-at-bulk network design. In: *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*; 1997:542–547.
43. Gilbert EN, Pollak HO. Steiner minimal trees. *SIAM J Appl Math*. 1968;16(1):1–29.
44. Nussbaum Y. *Multiple-Source Multiple-Sink Maximum Flow in Planar Graphs*. 2010. Computing Research Repository, 1008.5332, http://arxiv.org/abs/1008.5332
45. Thapalia, BK, Crainic, TG, Kaut, M, Wallace, SW. Single-commodity stochastic network design with multiple sources and sinks. *Information Systems and Operational Research*. 2011;49(3):193–211. ISSN 0315-5986.
46. Melzak ZA. On the problem of Steiner. *Can Math Soc*. 1961;4(2):143–148. https://doi.org/10.4153/CMB-1961-016-2
47. Trietsch D, Weng JF. Pseudo-Gilbert – Steiner trees. *Networks*. 1999;33(3):175–178.
48. Weng JF, Smith JM. Steiner minimal trees with one polygonal obstacle. *Algorithmica*. 2001;29:638–648. https://doi.org/10.1007/s00453-001-0002-1
49. Gavish B. Topological design of centralized computer networks—formulations and algorithms. *Networks*. 1982;12(4):355–377. https://doi.org/10.1002/net.3230120402
50. Arnaout J-P. Ant colony optimization algorithm for the Euclidean location-allocation problem with unknown number of facilities. *J Intell Manuf*. 2011;24(1):45–54. https://doi.org/10.1007/s10845-011-0536-2
51. Viet NQ. *Non-Deterministic Networked Infrastructure Design of Multiple Sources and Multiple Sinks*. Master's Thesis; 2015. uuid:617d26baf229-4dfb-b46f-36c1d3defbde.
52. Heijnen PW, Ligtvoet A, Stikkelman RM, Herder PM. Maximising the worth of nascent networks. *Netw Spatial Econ*. 2013;14(1):27–46. https://doi.org/10.1007/s11067-013-9199-1
53. Wu BY, Chao KM. Counting spanning trees. In: Bang YW, Kun-Mao C, eds. *Spanning Trees and Optimization Problems*. Boca Raton, FL: Chapman & Hall/CRC; 2004. https://doi.org/10.1016/0016-0032(89)90011-2

AUTHOR BIOGRAPHIES

PETRA W. HEIJNEN studied Applied Mathematics at Delft University of Technology. From 1994 to 1998, she did a four-year PhD research in coding theory at the Faculty of Applied Mathematics at the Eindhoven University of Technology. In 1998, she worked as a research member at the National Institute Statistics Netherlands. Since 1999, she is working at the Faculty of Technology, Policy and Management at Delft University of Technology, first as a project member of the E.E.T.-project Batch Processes: Cleaner and more efficient. Since 2002, she works at the faculty as an assistant professor in the Energy & Industry Group. She teaches courses in network analysis, operations research, and statistics. Her research field focusses on graph theory for the design and analysis of energy and industrial infrastructures, optimization in energy and industrial infrastructures, and applied statistics.

EMILE J.L. CHAPPIN is an associate professor at TU Delft, Energy and Industry Section of the Faculty of Technology, Policy and Management and a senior research fellow at the Wuppertal Institute for Climate, Energy and Environment. He graduated as a complex systems engineer and obtained his PhD "Simulating Energy Transitions" from TU Delft. He specializes in agent-based modeling, with a focus on sustainability, carbon and renewables policies, energy markets, and adaptation to climate change. Emile focuses on the perspective of complex sociotechnical systems and uses models and serious games to enable the support of policy interventions.

PAULIEN M. HERDER is a full professor in Engineering Systems Design in Energy. She has an MSc degree in Chemical Engineering and a PhD degree in the Engineering Sciences. She studies the design and management of energy systems and she is, more specifically, interested in energy systems integration and the transition to a sustainable energy provision. In this research, engineering systems sciences as well as so*cial and behavioral sciences play an important role.* Her research interests concern the design and the design process of large-scale networked systems, with a focus on systems engineering, engineering design under uncertainty, design processes, complex systems, energy systems, and energy systems integration.

How to cite this article: Heijnen PW, Chappin EJL, Herder PM. A method for designing minimum-cost multisource multisink network layouts. *Systems Engineering*. 2019;1–22. https://doi.org/10.1002/sys.21492

APPENDIX A: LITERATURE OVERVIEW

Appendix A shows a short summary in Table A2 of the optimization problems discussed in the literature as described in Section 2 and the methods used to solve these problems. Table A1 gives the abbreviations used in Table A2.

TABLE A1 Abbreviations used in Table A2

Topic	Abbreviation	Explanation
System	HD	Hydrogen
	CO	CO_2
	HT	Heating
	AW	Animal Waste
	WT	Water
	OI	Oil
Nodes	1N	1 source, multiple sinks
	N1	multiple sources, 1 sink
	NN	multiple sources, multiple sinks
Objective	IC	Investment Costs
	OC	Operational Costs
	LN	Length
Constraints	SC	Spatial Constraints, like crossings or corridors
	PC	Physical Constraints, eg, on pressure or velocity
	PR	Potential Routing restricting the layout of the pipelines
Decision	LY	Layout of the pipelines
	SZ	Size, ie, diameter or capacity of the pipelines
	PP	Physical Properties, like elevation or slope
Extra nodes	SN	Steiner nodes
	TN	Other transmission nodes
Topology	TR	Tree
	NW	Network
Method	GA	Genetic algorithm
	SA	Simulated annealing
	ACO	Ant colony optimization
	PSO	Particle swarm optimization
	GS	General solver, like GAMS
	DS	Dedicated software, like SimCCS, MARKAL
	HA	Heuristic algorithm

APPENDIX B: MATHEMATICAL BACKGROUND

This appendix gives some mathematical background for the discussion in Section 3.

B.1 Mathematical formulation

Mathematically, we translate the problem as follows, comparable to the formulation used by Xue et al.[39]

Let $A = [a_1, a_2, .., a_n]$ be a list of sources on known locations in 2D Euclidean plane and with given supply $[s_1, s_2, .., s_n]$. Let $B = [b_1, b_2, .., b_m]$ be a list of sinks on known locations in 2D Euclidean plane and with given demand $[d_1, d_2, .., d_m]$. In this paper, we assume that each node is or a sink or a source. In case the node is both sink and source, we only use the net demand or supply to determine the network capacity.

Let G be a network connecting the sources with the sinks. We want to find the network G that minimizes the total investment costs, given by

$$C(G) = \sum_{\forall e \in E(G)} l_e q_e^{\beta}, \tag{B.1}$$

in which $E(G)$ is the set of all pipelines, called *edges*, in G, l_e is the length of an edge e, q_e is the capacity of an edge e, and $\beta, 0 \le \beta \le 1$, is the capacity cost exponent. The network should satisfy the following constraints.

In the final supply network G, each edge will have a certain direction, denoted by $e = [i, j]$, ie, the flow will go from node i to node j.

Let $V(G)$ be the set of all nodes in G, then for all nodes, $v \in V(G) \backslash (A \cup B)$, which are the *Steiner nodes*,

$$\sum_{\forall [v,j] \in E(G)} q_{[v,j]} = \sum_{\forall [i,v] \in E(G)} q_{[i,v]}. \tag{B.2}$$

This constraint guarantees that the Steiner nodes are only transmission nodes and do not consume or supply.

The sinks and sources (together called the *terminals*) can be both transmission nodes and supply or demand nodes, respectively, which lead to the following constraints for all source nodes v ∈A

$$\sum_{\forall [v,j] \in E(G)} q_{[v,j]} - \sum_{\forall [j,v] \in E(G)} q_{[j,v]} \le s_v. \tag{B.3}$$

Moreover, for all sink nodes v ∈B

$$\sum_{\forall [j,v] \in E(G)} q_{[j,v]} - \sum_{\forall [v,j] \in E(G)} q_{[v,j]} = d_v. \tag{B.4}$$

Without loss of generality, we assume that the total supply of the sources covers at least the total demand of the sinks.

The formulated problem is NP-hard.[39]

B.2 Formal optimization

The optimization problem formulated above can be rewritten in a formal optimization NLP form.

Given the following input:

- $\beta, 0 \le \beta \le 1$ the capacity cost exponent,
- N the total number of nodes to be connected,

TABLE A2 Overview of recent literature on the design of network layouts

Reference	System	Nodes	Objective	Constraints	Decision	Extra nodes	Topology	Method
25	HD	1N	IC	PC	LY, SZ		TR	HA
26	AW	N1	LN		LY	SN	TR	HA
16	CO	NN	IC	SC	LY, SZ		TR	DS
32	OI	N1	IC	SC, PR	LY, SZ		TR	HA
15	CO	NN	IC	SC	LY, SZ	ON	TR	DS
29	WT	N1	IC	PC, PR	LY, SZ, PP		TR	GA
24	HT	1N	IC	SC, PC	LY, SZ	SN	TR	GS
18	CO	NN	IC, OC	PC	LY, SZ		TR	GA, HA
31	WT	1N	LN	PC, PR	LY, SZ	ON	TR	ACO
23	HT	1N	IC, OC	PC, PR	LY		TR	DS
17	CO	NN	IC, OC		LY, SZ	ON	TR	HA
33	OI	N1	LN		LY	SN	TR	PSO
27	WT	1N	IC	PC	LY, SZ		NW	ACO
28	WT	1N	IC, OC	PC, PR	LY, SZ, PP	ON	TR	HA
22	HT	1N	IC, OC	SC, PC	LY, SZ		TR	GS
21	CO	NN	IC, OC	SC, PR	LY, SZ	ON	TR	DS
19	CO	NN	IC	SC, PR	LY, SZ		TR	DS
30	WT	N1	LN	PC, PR	LY, SZ, PP		TR	GS, SA
20	CO	NN	IC, OC	PC, PR	LY		TR	GS

- n the total number of sources,
- s_i, $1 \le i \le n$ the supply of sources,
- d_i, $1 \le i \le N - n$ the demand of sinks,
- $[p_{1i}, p_{2i}]$, $1 \le i \le N$ the coordinates of all nodes in 2D Euclidean plane,
- $Q_v = \begin{cases} -s_i & 1 \le i \le n \\ d_{i-n} & n+1 \le i \le N \\ 0 & N+1 \le i \le 2(N-1) \end{cases}$ the capacity demand of nodes.

Find the decision variables:

- $q_{[i,j]}$, $1 \le i, j \le 2N-2$ the capacity of an edge from node i to node j,
- $[p_{1i}, p_{2i}]$, $N+1 \le i \le 2N-2$ the coordinates of possible Steiner nodes.

For which

$$\sum_{i=1}^{2N-2} \sum_{j=1}^{2N-2} F(i,j) * q_{[i,j]}^{\beta} \tag{B.5}$$

is minimized with

$$F(i,j) = \sqrt{(p_{1i} - p_{1j})^2 + (p_{2i} - p_{2j})^2}. \tag{B.6}$$

Under the constraints:

$$\sum_{j=1}^{2N-2} q_{[j,v]} = Q_v + \sum_{j=1}^{2N-2} q_{[v,j]}, 1 \le v \le 2N-2, \tag{B.7}$$

$$q_{[i,j]} \ge 0, 1 \le i, j \le 2N-2. \tag{B.8}$$

This optimization problem has many local minima. Each local minimum represents a graph connecting the terminals and with the minimum needed capacity assigned to the edges. From such a local minimum, it is hard to find a better solution since decreasing an edge capacity with a small amount makes the solution infeasible. Only by deleting one edge $[i_1, j_1]$, ie, capacity $q_{[i_1,j_1]}$ becomes 0, and by adding a new edge $[i_2, j_2]$, ie, capacity $q_{[i_2,j_2]}$ becomes larger than 0, can lead to a new feasible solution. This whimsical nature of the solution space makes it very hard to find the global optimum with a common solver.

B.3 Proof of Theorem 1

Theorem 1. *For every feasible network topology G satisfying the constraints* (B.2)-(B.4)*, there exists a Gilbert tree (or forest) topology T with costs less than or equal to the costs of G, where the costs are defined by* (B.1).

Proof. Theorem 3.1[39] proves this theorem for one-source multisink networks. We extend their proof for multisource multisink networks.

A non-tree multisource multisink network contains at least one cycle. Like all edges in the network, also the cycle edges will have a unique direction. Let $Cyc = [e_1, e_2, e_3, \dots, e_k]$ be a cycle in the network G. Let $CW = \{e | e \in Cyc,\ e \text{ in clockwise direction}\}$ be the subset of all clockwise edges in the cycle and let $CCW = \{e | e \in Cyc,\ e \text{ in counterclockwise direction}\}$ be the subset of all counterclockwise edges in the cycle. So, each edge is either in CW or in CCW.

Just as before, let q_e be the capacity of all edges e with $q_e \ge 0$, $\forall e \in Cyc$.

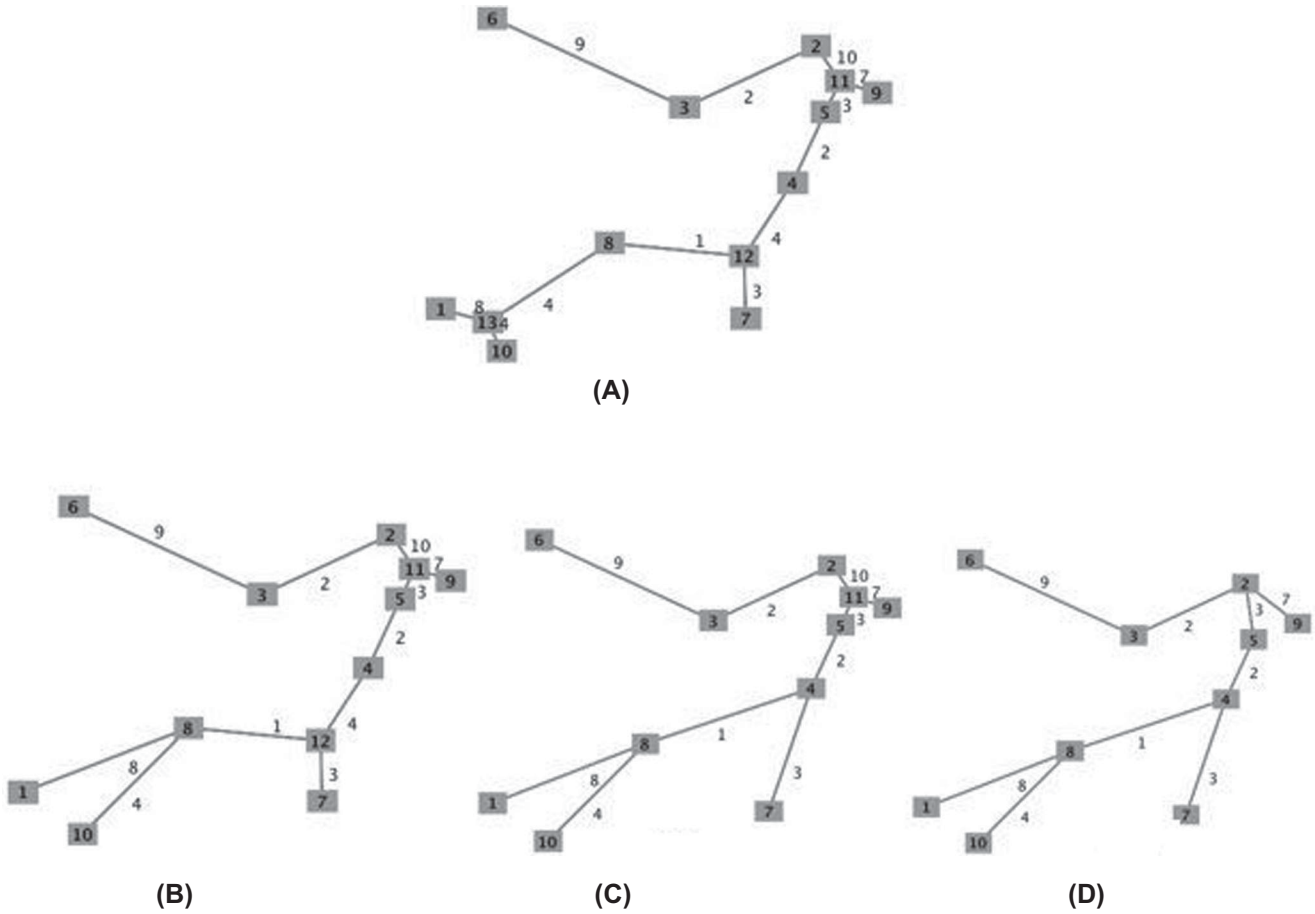

FIGURE B.1 From Gilbert tree topology (A) to spanning tree (D)

We define $r = \min(q_e | e \in Cyc)$, as the minimum capacity of all edges in the cycle and for each edge, we define $a_e = q_e - r,\ \forall e \in CW$ and $b_e = q_e - r,\ \forall e \in CCW$, so all a_e and b_e are nonnegative.

Let $f(q)$ be a concave function. Comparable to Xue et al,[39] we define a new function F by

$$F(z) = \sum_{\forall e \in CW} l_e f\left(a_e + r - z\right) + \sum_{\forall e \in CCW} l_e f\left(b_e + r + z\right) \tag{B.10}$$

for all $z \in [-r, r]$.

Since F is a linear combination of concave functions with coefficients $l_e > 0$, F itself will also be concave. Then,

$$F(0) = F\left(\frac{1}{2}(-r) + \frac{1}{2}r\right) \geq \frac{1}{2}\left(F(-r) + F(r)\right) \geq \min\left\{F(-r), F(r)\right\}. \tag{B.11}$$

For all $z \in [-r, r]$, then $F(z) - F(0)$ is the increase in network cost caused by adding capacity z to the capacity of the counterclockwise cycle edges and subtracting capacity z from the clockwise cycle edges.

Now, if $F(r) = \min(F(-r), F(r))$, then $F(r) - F(0) \leq 0$. Otherwise, if $F(-r) = \min(F(-r), F(r))$, then $F(-r) - F(0) \leq 0$.

Therefore, *either* adding capacity r to $e \in CW$ and subtract capacity r to $e \in CCW$ will reduce the network costs, *or* adding capacity r to $e \in CCW$ and subtract capacity r to $e \in CW$ will reduce the network costs.

However, if we add capacity r (or –r) to the cycle edges, the network costs will be reduced, but the cycle will not per se be broken. When we repeat the procedure, each cycle in the network will finally terminate and the network will have a tree or forest topology with lower costs than the original network. ■

B.4 Capacity assignment procedure

Before an initial ST can be used to find local changes, sufficient, but not too much, capacity needs to be assigned to the edges in such a way that the constraints (B.2)-(B.4) defined in B.1 are satisfied and the total costs are minimal. This can only be done in one unique way, given by the following general procedure.

Given a tree T connecting all sources with sinks by a set of edges $E(T)$ and some possible Steiner nodes (only when T is not an ST).

Let $V(T)$ be the set of all nodes in T. Let the demand $Q = [-s_1, -s_2, .., -s_n, d_1, d_2, \ldots, d_m, 0, 0, .., 0]$, ie, the demand is the negative supply of the sources, the positive demand of the sinks, and 0 for Steiner points.

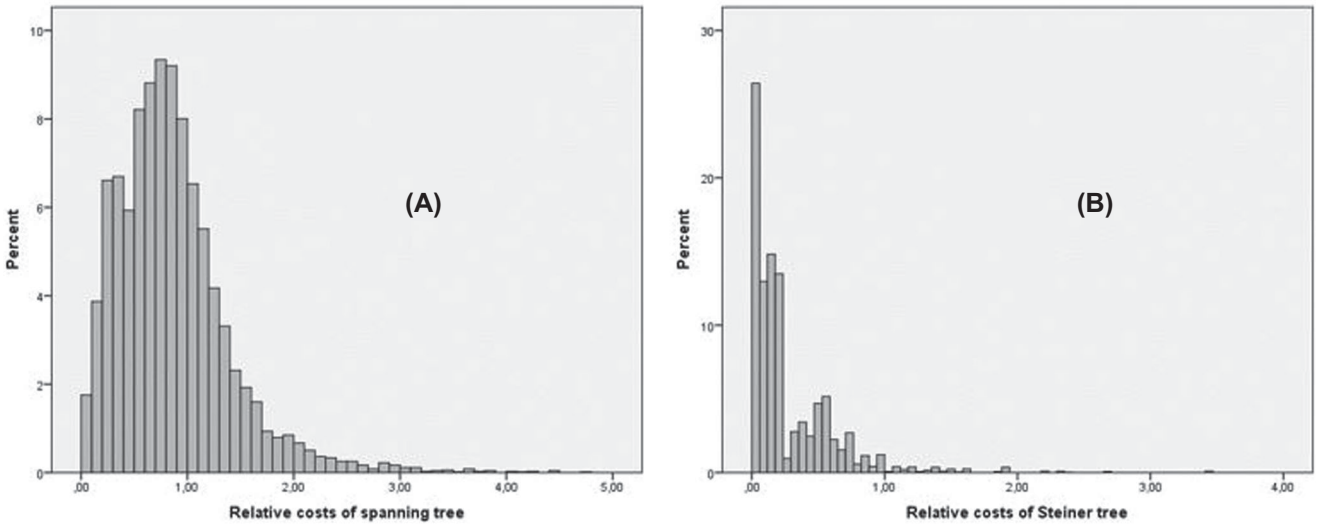

FIGURE B.2 Relative costs of spanning trees (A) and resulting Steiner trees (B)

Assigning capacity procedure
(Comparable to the procedure in Ref. 39 for the one-source multisink case)

1. Let $T1$ be a copy of the tree T with only bidirectional edges.
2. Set all capacities $q_{i,j} = q_{j,i} = 0$ for all edges $\{i, j\}$ in $E(T)$.
3. Find set L of all leafs of tree $T1$ (A leaf is a node of degree 1).
4. For all i L, find the unique edge $\{i, j\} \in E(T1)$ Set $q_{i,j} \rightarrow -Q_i$ if $Q_i < 0$ Set $q_{i,j} \rightarrow Q_i$ if $Q_i > 0$ Update $Q_j \rightarrow Q_j + Q_i$ Delete edge $\{i, j\}$ from $T1$.
5. Repeat from step 3 until all capacities are set.

B.5 Importance of initial spanning trees

The Gilbert-Melzak method is a deterministic method. Starting from one specific initial ST, it will always result in the same GT.

Theorem 2. *Every GT topology* S *on* n *terminals can be found by applying the adapted Gilbert-Melzak procedure on a specific ST* T *on these* n *terminals.*

Proof. The adapted Gilbert-Melzak procedure used here to find an MCGT topology from a specific initial ST improves the tree locally in an iterative process starting by the worst angle in the current tree and continuing as long as improvements on the total costs can be made.

In return, starting by a GT, one can degenerate a Steiner point, by contracting one of its incident edges and adapting the capacity of the remaining edges accordingly. Just like Steiner Trees, each GT topology consists of several full Steiner subtrees. A *full Steiner tree* is a tree with k terminals and exactly $k-2$ Steiner points. These subtrees are connected in a terminal. Degenerating a Steiner point in one of the full Steiner subtrees has no influence on the capacity of the edges in the other subtrees.

By degenerating the Steiner points in the opposite order of which they would have been added guarantees that the GT topology will be found back when applying the adapted Gilbert-Melzak method on the resulting ST, see Figure B.1. ■

For 15 randomly generated experiments with total number of nodes $N = 4$, 5, or 6 and total number of sources is 1, 2, or 3, we generate all spanning trees using Prüfer sequences[53] and assign the required capacities to the edges using the procedure from B.4. We applied the adapted Gilbert-Melzak method on all STs to determine a (suboptimal) MCGT. To study the relation between the costs of the original ST and the resulting Gilbert tree over different examples, the relative costs $CR_{k,i}$ (example k, run i) are considered, defined by

$$CR_{k,i} = \frac{C_{k,i} - C_{min,k}}{C_{min,k}}, \tag{B.12}$$

with $C_{k,i}$ the costs of the ST (respectively, resulting GT) of run i in example k, and $C_{min,k}$ are the minimum costs over all runs in example k of all STs and Gilbert trees together. The costs of a tree are defined as in Equation (1) and are only used to compare the results, not to give a realistic measure for the network investment costs.

The histograms in Figure B.1 reveal that while the costs of different STs show a high variation, this variation is significantly reduced in the costs of the resulting GTs. For these relatively small-scale examples, we draw the conclusion that many different STs in general result in a much smaller set of Gilbert trees after the adapted Gilbert-Melzak procedure is applied and that most STs result in an MCGT (see the peak at 1 on the x-axis in Figure B.1B). Moreover, the results show that lower cost STs have a higher probability to end up in a low-cost GT (correlation coefficient: $\rho = 0.652$, $p = 0.000$).

B.6 Spanning trees from Prüfer sequences

Prüfer sequences of length $N - 2$ have a one-to-one relation with all STs on N nodes.[53] Using Prüfer sequences, all possible STs can easily

be generated. There are $N^{(N-2)}$ different STs on N nodes, both crossing and noncrossing. We will check them all, although crossing trees will always be more costly than noncrossing trees.[47]

For 15 randomly generated experiments with total number of nodes $N = 4$, 5, or 6 and total number of sources is 1, 2, or 3, we generate all STs and assign the required capacities to the edges using the procedure from §B.4. We applied the adapted Gilbert-Melzak method on all STs to determine a (suboptimal) MCGT. To study the relation between the costs of the original ST and the resulting GT over different examples, the relative costs $CR_{k,i}$ (example k, run i) are considered, defined by

$$CR_{k,i} = \frac{C_{k,i} - C_{min,k}}{C_{min,k}}, \tag{B.13}$$

with $C_{k,i}$ the costs of the ST (respectively, resulting GT) of run i in example k, and $C_{min,k}$ are the minimum costs over all runs in example k of all STs and GTs together.

APPENDIX C: RESULTS OF SELECTION OF SPANNING TREES

Table C.1 shows the results of the GT method when only a selection of the four initial STs (MST = minimum spanning tree, HN = hub network, MCST = minimum-cost spanning tree, Δ-tree = Resulting tree[25]) was used. The "Best tree" column tells how many times the lowest cost tree was found by the selection of the initial STs out of the 100 examples. The "Average relative costs if not" column shows the average cost deviation from the best tree when the initial ST selection did not find the best tree. It is an indication of how bad the results are when the best tree is not found. Remember that the best tree is not always the global minimum-cost tree. The "Average time (s)" column gives the average time in seconds that was needed to find the end results for this combination of initial STs.

Figure C.1 shows a scatter plot of the results from Table C.1. To obtain low relative costs and low computation time, the selections 1, 4, 9, and 15 are Pareto-optimal, as highlighted in Table C.1.

Figure C.2 shows for the 1000 random examples the box plot of the relative costs compared to the best tree. The box plots show that

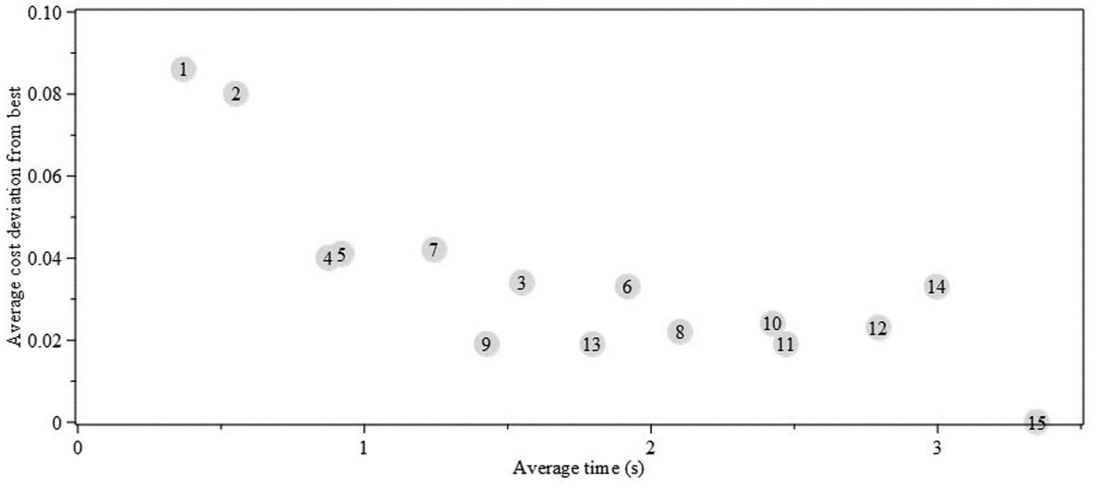

FIGURE C.1 Scatter plot of relative costs and time for different spanning tree selections

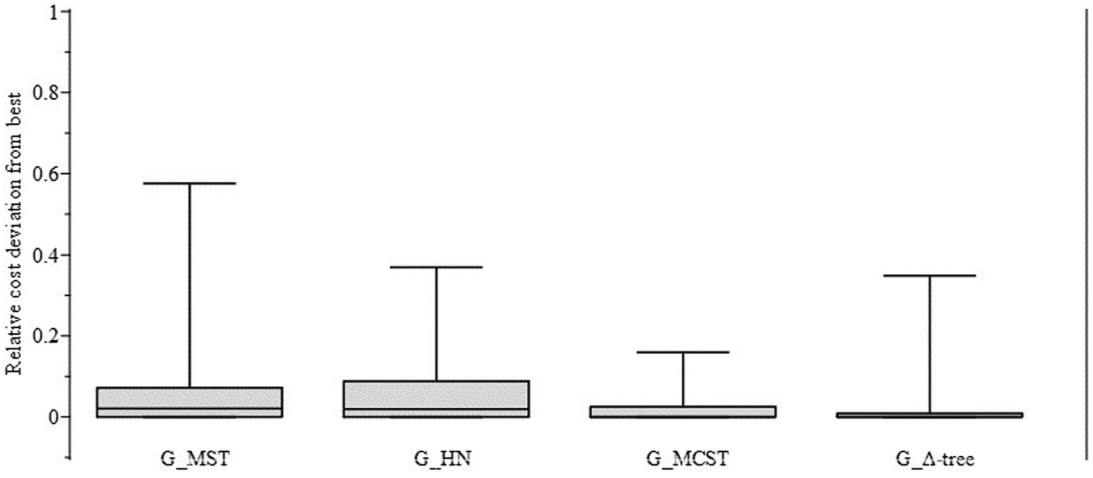

FIGURE C.2 Box plot of the relative costs compared to best tree for Gilbert tree from four different spanning trees

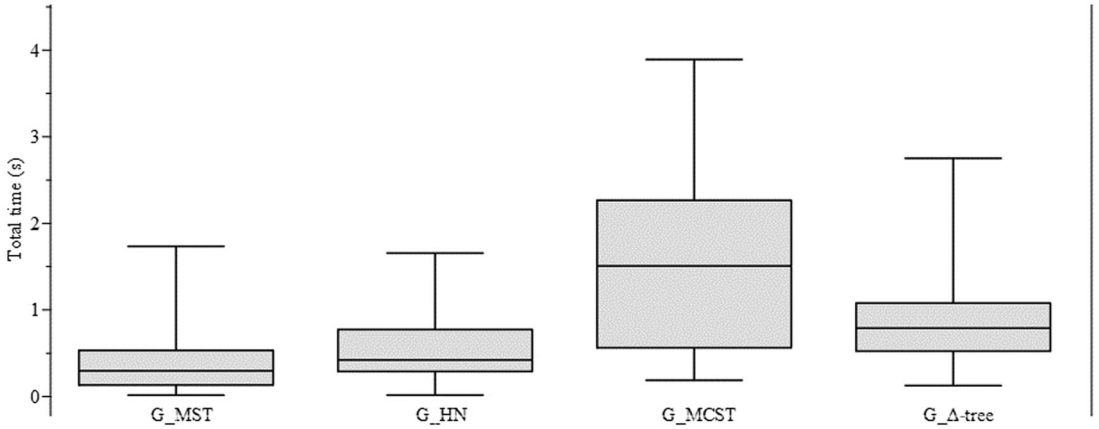

FIGURE C.3 Computational time needed to find Gilbert tree from four different spanning trees, including the time for generating the spanning tree

TABLE C.1 Results of the 100 random examples for selections of initial spanning trees

	Selection spanning trees	Best tree	Average relative costs if not	Average time (s)
1	MST	33	0.086	0.369
2	HN	27	0.080	0.552
3	MCST	50	0.034	1.550
4	Δ-tree	60	0.040	0.875
5	MST, HN	56	0.041	0.921
6	MST, MCST	55	0.033	1.919
7	MST, Δ-tree	65	0.042	1.244
8	HN, MCST	69	0.022	2.102
9	HN, Δ-tree	79	0.019	1.427
10	MCST, Δ-tree	81	0.024	2.425
11	MST, HN, MCST	74	0.019	2.471
12	MST, MCST, Δ-tree	83	0.023	2.794
13	MST, HN, Δ-tree	84	0.019	1.796
14	HN, MCST, Δ-tree	98	0.033	2.997
15	MST, HN, MCST, Δ-tree	100	-	3.346

the end results of the GTs resulting from the MCST have the lowest spread. However, the interquartile range of the GTs resulting from the Δ-tree is smaller, but the Δ-tree leads in some cases to more extreme results.

Figure C.3 shows box plots for the computational time to find the initial ST and the GT from this ST for the 100 random examples. It is clear that the generating the MCST and the resulting GT needs most time in general.