Fixed parameter tractability of a biconnected bottleneck Steiner network problem

C. J. Ras*

School of Mathematics and Statistics, University of Melbourne

Abstract

For a given set X of points in the plane, we consider the problem of constructing a 2-vertex-connected network spanning X and at most k additional Steiner points such that the length of the longest edge (the so-called bottleneck) of the network is minimised. When one introduces a constraint on the network specifying that all Steiner points must be of degree 2 the problem remains NP-complete but becomes fixed parameter tractable with respect to k. We prove this by presenting an algorithm which solves the degree-2 bounded problem optimally in a time of $O(n^4k^62^k)$, where $n = |X|$. We also present a simple 3-approximation algorithm for the degree-2 bounded problem and show that the bottleneck length of an optimal solution to the degree-2 bounded problem is at most twice the bottleneck length when degree is not bounded.

Keywords: bottleneck Steiner network; fixed parameter tractability; geometric network; biconnected network; 2-connectivity; approximation algorithms

1 Introduction

Network design problems where the objective is to minimise the longest edge are called *bottleneck* problems or *mini-max* problems. These problems find wide application in facility location; VLSI design; and telecommunications, especially in the optimal deployment and routing of wireless sensor networks (see [14]). Typically, the problem consists of finding a minimum bottleneck network that spans a given set of *terminals* and an additional set of *Steiner points* which model relays or junctions.

*Email: cjras@unimelb.edu.au

The problem has been modelled and studied in two broad domains. In the combinatorial domain one is given a weighted graph G and the objective is to find a subgraph H of G, satisfying some connectivity constraint, such that the length of the longest edge of H (the *bottleneck* of H) is minimised; see for instance [7, 12]. In the geometric domain one is given a set X of points in the plane and an integer $k \geq 0$ and the aim is to construct a minimum bottleneck network spanning X and at most k additional Steiner points in the plane. Here the length of the bottleneck is measured using some geometric norm, for instance the Euclidean norm (see [1, 3]).

In this paper we focus on the problem within the geometric, specifically Euclidean, domain. In addition, we consider the problem where the output network is required to be biconnected (2-vertex-connected). This is an important problem from the standpoint of robust network design. Firstly, the constraint of biconnectivity is commonly employed in practice in order to ensure that there are at least two physically diverse paths of communication between every pair of vertices. In telecommunications networks such as optical fibre networks, this constraint is generally sufficient to ensure that the network remains connected during the critical time-period between vertex failure and subsequent recovery [9].

An exact algorithm for the biconnected bottleneck network problem was developed by Brazil et al. [4, 5]. They show that the complexity of their algorithm is exponential in k but polynomial in $|X|$ for any fixed k, however, the question of whether there exists a *fixed parameter tractable* (FPT) algorithm for this problem (in terms of k) remains open. Recall that an FPT algorithm for a problem with parameter α is an algorithm which exactly solves the problem in time $f(\alpha) \cdot |I|^{O(1)}$, where f is some computable function and $|I|$ is the length of the input instance. It is clear that finding an FPT algorithm for a problem in terms of a parameter α therefore implies that the problem is tractable for small values of α.

In this paper we go part way towards answering this question by developing a fixed parameter tractable algorithm for the problem with an additional constraint that all Steiner points in the output network are of degree 2 (we call such networks *beaded* networks). Beaded networks are interesting in their own right and are motivated by the fact that degree bounds are commonly incorporated into network models, for instance to reduce latency and interference in telecommunications and in the design of computer networks [13]. But, as we will show, beaded networks also serve as 2-factor approximate solutions to the biconnected bottleneck network problem without the degree restriction.

Many bottleneck network design problems are NP-complete when Steiner points are included. The Euclidean and rectilinear bottleneck Steiner tree problems were shown to be NP-complete by Wang and Du [17]. The authors also showed that the problem is not approximable to within factors of $\sqrt{2}$ and 2 in the Euclidean and rectilinear planes respectively. They also present a simple 2-factor approximation for

the problem in the Euclidean plane.

The biconnected bottleneck network design problem in general ℓ_p planes was shown to be NP-hard by Brazil et al. [6]. The authors show that (unless P=NP), an optimal solution cannot be approximated to within a factor of less than $2^{\frac{1}{p}}$ for $p \geq 1$. Their proof also demonstrates that, in contrast to the tree problem studied by Wang and Du [17], the problem remains NP-hard even if one restricts the degree of all Steiner points to 2. This leads to the question of whether there exists a constant factor approximation algorithm for the biconnected problem in the Euclidean plane.

The main result of this paper is the development of a fixed parameter tractable algorithm for the biconnected bottleneck network design problem in the Euclidean plane when Steiner points are restricted to degree 2. We also present an approximation algorithm with performance ratio 3 for this problem. We show that this approximation algorithm is a 6-factor approximation algorithm for the general (non degree-bounded) biconnected bottleneck network design problem in the Euclidean plane. In Section 2 we provide the necessary background and present our approximation algorithm and in Section 3 we present our fixed parameter tractable algorithm.

2 Background

Throughout this paper we use the notation $cost(G)$ to denote the Euclidean length of the longest edge (the bottleneck) in a graph G. We first present some necessary background on the 1-connected (tree) version of the problem.

Definition 2.1 *Let X be a set of points in the plane and let $k \geq 0$ be an integer. A minimum Euclidean bottleneck k-Steiner tree T_{opt} on X is a tree spanning X and a set of k additional Steiner points in the plane such that $cost(T_{\text{opt}})$ is a minimum.*

It was shown in [17] that the above problem is NP-complete. The authors present a simple 2-approximation algorithm for the problem which runs as follows: first a minimum spanning tree (MST) T on X is constructed. Next, a degree-2 Steiner point (referred to as a *bead*) is placed at the midpoint of the longest edge of T. The algorithm then repeats this step until k beads have been added. At each step a longest edge of the current tree is found and a bead is added to this edge. Whenever a bead is added, all beads on the corresponding edge of T are shifted so that they are equally spaced on the edge. The resultant tree is called the *beaded* MST on X and has cost at most twice the cost of T_{opt}. The authors also show that the beaded MST is the optimal tree given the condition that all Steiner points are of degree 2.

Next we provide some background and formalities on biconnected graphs. Let $G = \langle V, E \rangle$ be a graph. We say that G is 2-*connected*, or *biconnected*, if the removal of any

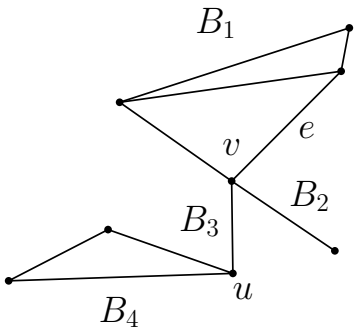


Figure 1: A block-cut tree consisting of four blocks, B_1, B_2, B_3 and B_4, each containing one of the cut-vertices u or v. Note that $B_1 = B(e)$ is a leaf-block consisting of five edges, and includes cut-vertex v. The blocks B_2 (consisting of a single edge) and B_4 (consisting of three edges) are also leaf blocks.

vertex from G does not disconnect G. For every edge e of G, let $B(e)$ be a maximal subset of edges of G containing e such that the subgraph of G induced by $B(e)$ is 2-connected. If no such subgraph containing e exists, then we let $B(e) := \{e\}$; see Figure 1 for an example. The set $\mathcal{B} := \{B(e) : e \in E\}$ is therefore a partition of the edges of G such that every element of $\mathcal{B}$ is either a singleton containing an edge of G or induces an edge-maximal 2-connected subgraph of G. The elements of $\mathcal{B}$ are called *blocks*. Without causing ambiguity, we will treat $B(e)$ interchangeably as either a graph or a set of edges.

Loosely speaking, there is a map from set $\mathcal{B}$ to an acyclic graph $F := F(G)$ where the vertices of F correspond to the members of $\mathcal{B}$ and the cut-vertices of G. The acyclicity follows from the fact that if any two sets $B(e)$ and $B(e')$ share any vertices then they share exactly one vertex, and this vertex will be a cut-vertex of G. In the example of Figure 1 all three blocks B_1, B_2 and B_3 contain the cut-vertex v. The forest F (and also, interchangeably, $\mathcal{B}$) is referred to in the literature as the *block-cut forest* of G, or *block-cut tree* when F is connected. For a more detailed definition and properties of block-cut forests we refer the reader to [4] and [8], and to Tarjan's seminal paper [16] for a fast algorithm to compute them. Here we mention just a few of the properties we need for proving our results.

Any $B(e)$ that contains exactly one cut-vertex of G is called a *leaf-block*; see Figure 1. A block that is a strict subgraph of G and that does not contain any cut-vertices of G is called an *isolated* block. For any block $B(e)$, suppose that V' is the set of cut-vertices of G contained in $B(e)$. We refer to $B(e)\backslash V'$ as the *interior* of $B(e)$, denoted $\text{int}(B(e))$. Note that the interior of an isolated block is the block itself, and the interior of a block may be empty.

We now provide a formal definition of the problem we consider in this paper.

Definition 2.2 *Let X be a set of points in the plane and let $k \geq 0$ be an integer. A minimum Euclidean 2-connected bottleneck k-Steiner network N_{opt} on X is a 2-connected network spanning X and a set of k additional Steiner points in the plane such that $cost(N_{\text{opt}})$ is a minimum.*

It was shown in [6] this this problem is NP-complete and cannot be approximated in polynomial time (unless P=NP) to within a factor less than $\sqrt{2}$. In analogy to the 1-connected version described in Definition 2.1, a natural question is whether some beaded 2-connected network can serve as an approximation for the general 2-connected problem of Definition 2.2. We therefore define the following.

Definition 2.3 *Let X be a set of points in the plane and let $k \geq 0$ be an integer. A minimum beaded Euclidean 2-connected bottleneck k-Steiner network N_{bead} on X is a 2-connected network spanning X and a set of k additional Steiner points in the plane such that every Steiner point in N_{bead} is of degree 2 and $cost(N_{\text{bead}})$ is minimised.*

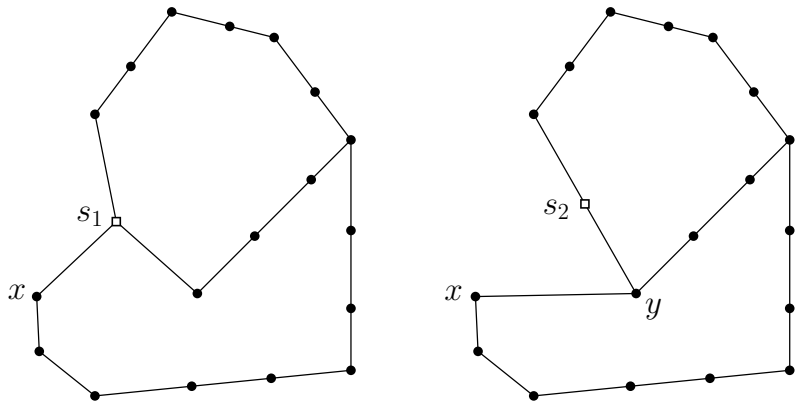


Figure 2: Optimal bottleneck 2-connected networks on the same set of 18 terminals (black filled circles). Here $k = 1$, so both networks contain a single Steiner point (empty squares). The network on the left satisfies Definition 2.2, i.e., there is no restriction on the degree of the Steiner point, s_1. The bottleneck of this network is edge xs_1. The network on the right satisfies Definition 2.3 and therefore the Steiner point s_2 is a bead (i.e., is of degree 2). The bottleneck of the network on the right is xy, which is clearly longer than xs_1.

See Figure 2 for an example of two networks satisfying Definitions 2.2 and 2.3 respectively.

Next we will show that $cost(N_{\text{bead}})$ is at most twice $cost(N_{\text{opt}})$, in analogy to what was shown for the 1-connected version of the problem in [17]. Unfortunately, even

constructing N_{bead} is an NP-complete problem (see [6]). However, due to this approximation ratio the construction of N_{bead} is an important problem. The main result of this paper is that there exists a fixed parameter tractable algorithm (with respect to k) to construct N_{bead}.

To prove the approximation ratio we first need two lemmas that follow directly from results proved by Luebke and Provan in [11] and Hvam et al. in [10] respectively. We assume that N_{opt} and N_{bead} are *Steiner edge-critical* – meaning that the removal of any Steiner edge or any path containing only beads in its interior reduces connectivity. Such an assumption is reasonable, as non-critical paths and edges can simply be removed from these graphs.

A *Steiner network* on X is any network spanning X that potentially also contains Steiner points. Let N' be any 2-connected Steiner network on X.

Definition 2.4 *Let e be an edge of N' and let Q be the union of all maximal length paths in N' that contain e but do not contain internal vertices that are terminals. Then Q is called a full Steiner component of N'.*

Lemma 2.5 ([11]) *The full Steiner components of both N_{opt} and N_{bead} are trees, and the leaves of any such tree are terminals.*

We refer to the full Steiner components of N' that are trees as *full Steiner trees*. Clearly, in both N_{opt} and N_{bead}, the set of all full Steiner trees partitions the edge-set of the network. The next lemma follows directly from results in [10].

Lemma 2.6 ([10]) *Let Q be a full Steiner tree of N' (if one exists), let X_Q be the set of leaves of Q and let S_Q be the set of Steiner points of Q. Let T_Q be any tree spanning X_Q such that none of the internal vertices of T_Q are in $N' \backslash S_Q$. Then $(N' \backslash S_Q) \cup T_Q$ is 2-connected.*

Proposition 2.7 $cost(N_{\text{bead}}) \leq 2\, cost(N_{\text{opt}})$

Proof. For each full Steiner tree Q of N_{opt} we perform the following procedure. Let X_Q be the terminals (leaves) of Q. We replace Q in N_{opt} by the beaded MST on X_Q with the same number of beads as there are Steiner points in Q. By the previous lemma, this replacement maintains 2-connectivity. Also, as proved in [17], the beaded MST has at most twice the cost of the optimal Steiner tree connecting X_Q. Finally, observe that once all full Steiner trees have been replaced, all Steiner points in the resulting network are of degree 2. The result follows. ■

Finally, in Algorithm 1 we present a simple algorithm (based on Sekanina's algorithm; see [2, 15]) which is a 3-approximation for the minimum beaded 2-connected bottleneck k-Steiner network problem, and hence a 6-approximation algorithm for the general minimum 2-connected bottleneck k-Steiner network problem.

Algorithm 1 (Approximation algorithm)

Input: A set X of n points in the plane and an integer $k \geq 0$

Output: A 2-connected network N_{approx} on $X \cup S$ where S is a set of k Steiner points.

1: Construct a minimum spanning tree T on X
2: Construct the beaded MST T_{bead} by introducing a set S of k beads to T
3: Construct the cube T^3_{bead} of T_{bead} and find a Hamiltonian cycle H in T^3_{bead}
4: Set $N_{\text{approx}} := H$

Note that all steps of Algorithm 1 can be performed in polynomial time. In particular, Step 1 can be performed in time $O(n \log n)$ by employing the Delaunay triangulation of X. Step 2 can be performed in time $O(k \log n)$ (see [17]) and Step 3 can be performed in time $O(n)$ (see [2]). This gives an asymptotic running time of $O((n+k)\log n)$ for Algorithm 1.

Theorem 2.8 *$cost(N_{\text{approx}}) \leq 3\, cost(N_{\text{bead}})$ and $cost(N_{\text{approx}}) \leq 6\, cost(N_{\text{opt}})$*

Proof. As proved in [17], the tree T_{bead} is an optimal solution to the minimum beaded bottleneck k-Steiner tree problem on X, and is a 2-factor approximation for the general bottleneck k-Steiner tree problem. Therefore $cost(N_{\text{bead}}) \geq cost(T_{\text{bead}})$ and $cost(T_{\text{bead}}) \leq 2\, cost(T_{\text{opt}})$. Clearly also $cost(N_{\text{opt}}) \geq cost(T_{\text{opt}})$.

Now, $cost(N_{\text{approx}}) = cost(H) \leq cost(T^3_{\text{bead}})$. Also, by the triangle inequality, $cost(T^3_{\text{bead}}) \leq 3\, cost(T_{\text{bead}}) \leq 3\, cost(N_{\text{bead}})$. Therefore the first inequality follows.

Next, note that $cost(N_{\text{approx}}) \leq 3\, cost(T_{\text{bead}}) \leq 6\, cost(T_{\text{opt}}) \leq 6\, cost(N_{\text{opt}})$, and the second inequality follows. ■

The next example shows that the worst case approximation ratio for Algorithm 1 is at least 2.

Example

Let $x_1, ..., x_n$ be a set of equally spaced terminals on the boundary of a circle. We normalise so that that the distance between consecutive terminals is 2 units. Let $k = n$. A minimum beaded 2-connected bottleneck k-Steiner network on $X' := \{x_i : 1 \leq i \leq n\}$ has cost 1, since the optimal network consists of all edges $x_i x_{i+1}$, $i < n$ and the edge $x_1 x_n$ (i.e, a cycle through X'), where the edges of length 2 are beaded by a single Steiner point (at the midpoint of the edge).

Now, let T be a minimum spanning tree on X'. Without loss of generality, T consists of exactly the edges $x_i x_{i+1}$, $i < n$. Then $n-2$ edges of T receive exactly one bead in

the tree T_{bead} on X', and exactly one edge of T has two equally spaced beads in T_{bead}, dividing this edge of T into three edges of length 2/3. Without loss of generality we assume that x_1x_2 is the edge of T which acquires two beads. We denote the Steiner points of T_{bead} by $S' = \{s_i\}$, so that the edges of T_{bead} consist of x_1s_1, s_1s_2, s_2x_2 and all edges x_is_{i+1} and $s_{i+1}x_{i+1}$ for $1 < i < n$. It is straightforward to show that the following Hamiltonian cycle in T^3_{bead} has minimum cost amongst all Hamiltonian cycles in T^3_{bead}: let H be the cycle $x_1, s_1, x_2, x_3, x_4, ..., x_n, s_n, s_{n-1}, s_{n-2}, ...s_3, s_2, x_1$. The cost of H is 2, which is twice the cost of the optimal solution on X'.

3 A fixed parameter tractable algorithm

In this section we present a fixed parameter tractable (FPT) algorithm of complexity $O(n^4k^62^k)$ which, for any set X of n points, finds a Euclidean minimum beaded 2-connected bottleneck k-Steiner network on X.

First we state some definitions and a proposition. Let N be a Euclidean minimum beaded 2-connected bottleneck k-Steiner network on X. As before, we assume that N is Steiner-edge critical. In addition, throughout we assume that N is *bottleneck complete*. This means that N contains all edges between vertices of X that have length at most the bottleneck of N. This is a reasonable assumption, since all such edges can be added to N without increasing the bottleneck length.

The *underlying graph* of N is the graph $U(N)$ we obtain by replacing each path between terminals whose internal vertices are all Steiner points (beads) by an edge; see Figure 3 for an example. A *Steiner edge* of the underlying graph is an edge that replaced a path in N containing at least one Steiner point. Note that since N is Steiner-edge critical, the graph $U(N)$ is also Steiner-edge critical.

Let E be the set of all Steiner edges of $U(N)$ and let N_0 be the subgraph of $U(N)$ obtained by deleting all Steiner edges (see Figure 4). For some ordering $e_1, ..., e_q$ of the edges in E, let $N_i = N_0 \cup \{e_1, ..., e_i\}$. Therefore $U(N) = N_q$.

Proposition 3.1 *There exists an ordering $e_1, ..., e_q$ of the edges in E such that e_i is incident to either a leaf-block interior or an isolated block of N_{i-1} for all $i \in \{1, ..., q\}$.*

Proof. Consider N_0. Every leaf-block interior and isolated block of N_0 must contain at least one endpoint of an edge from E, for otherwise N would not be 2-connected. Add such a set of edges from E to N_0 to get a new graph $N(1)$ and let $e_1, ..., e_{t_1}$ be an arbitrary ordering of these edges. Note that if $t_1 < q$ then $N(1)$ is not 2-connected, since N is optimal. We now repeat this process, noting that since $N(1)$ is not 2-connected it must contain leaf-block interiors or isolated blocks, and each such set of vertices must contain an endpoint of a Steiner edge in $E - \{e_1, ..., e_{t_1}\}$. We order this

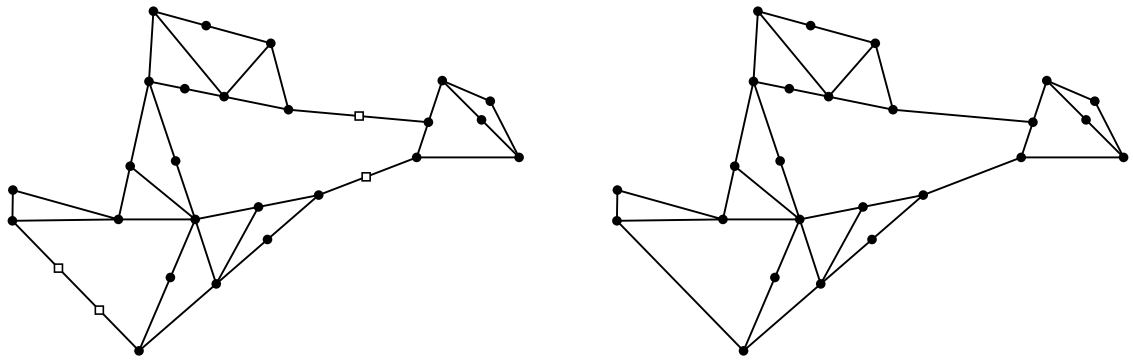


Figure 3: An example of a beaded Steiner network N (left) and its underlying graph $U(N)$ (right).

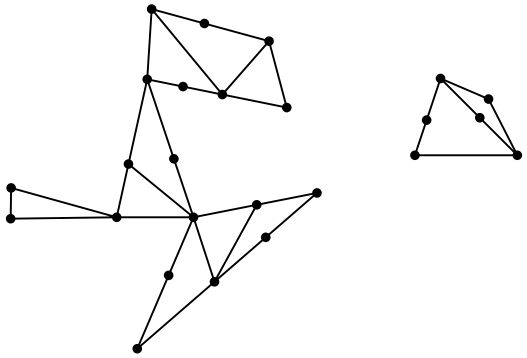


Figure 4: The graph N_0 for the example network N from Figure 3. Note that N_0 here consists of one isolated block and a component containing three leaf-blocks.

set of Steiner edges as $e_{t_1+1}, ..., e_{t_2}$ and let $N(2) = N(1) \cup \{e_{t_1+1}, ..., e_{t_2}\}$. Once again, if $t_2 < q$ then $N(2)$ is not 2-connected and we repeat. We terminate the process when we have added all Steiner edges and have an ordering $e_1, ..., e_q$. ■

The next corollary will be used when proving the complexity of our FPT algorithm.

Corollary 3.2 *For all* $\in \{0, ..., q\}$*, the graph* N_i *contains at most* $2k$ *leaf-blocks or isolated blocks.*

Proof. This follows immediately for N_0 from the previous lemma, given that $q \leq k$. Note also that the number of leaf-blocks or isolated blocks in N_i is at most the number in N_{i-1}, for all i; in fact, this quantity is monotone with respect to subgraphs. ■

Our FPT algorithm will use the fact of Proposition 3.1 to consecutively add Steiner edges to a graph until 2-connectivity is obtained, where at each step a leaf-block

interior or an isolated block is selected in order to contain an endpoint of the current Steiner edge, e_i. There are $O(k)$ choices for this endpoint, as proved in the previous corollary. In general, the other endpoint of e_i does not necessarily lie in a leaf-block interior (or isolated block) of the graph obtained up to that point. Therefore a choice will be made at each step to decide what subset of the underlying graph the other endpoint is contained in. As we will show, to obtain FPT complexity there must be $O(f(k))$ such choices, where $f(k)$ is a function only of k. The next subsection leads up to a method for choosing such subsets of the underlying graph at each step.

3.1 Selecting endpoints for e_i

We need the following result by Dirac [8]. Recall the notation $F(G)$ for the block-cut forest of a graph G.

Lemma 3.3 *Let G be a 2-connected graph and let e be a critical edge of G (i.e., removing e reduces the connectivity of G). Then $F(G\backslash\{e\})$ is a path and the endpoints of e lie in respective leaf-block interiors of $G\backslash\{e\}$. In fact, adding an edge e' to $G\backslash\{e\}$ results in a 2-connected graph if and only if the endpoints of e' are any two vertices in distinct leaf-blocks interiors of $G\backslash\{e\}$.*

Now consider the graph $N' = U(N)\backslash\{e_i\}$ and note that, since N is Steiner-edge critical, $F(N')$ is a non-trivial path. Denote the leaf-blocks of N' by W and W'; see Figure 5. Denote by $c(W)$ the cut-vertex of N' contained in W and define $c(W')$ similarly. Note that $c(W)$ and $c(W')$ coincide if and only if N' has exactly two blocks. Let x be the endpoint of e_i in $\mathrm{int}(W)$ and let x' be the endpoint in $\mathrm{int}(W')$. Let C' be a maximal component of N_0 containing x'. Note therefore that C' does not contain any Steiner edges.

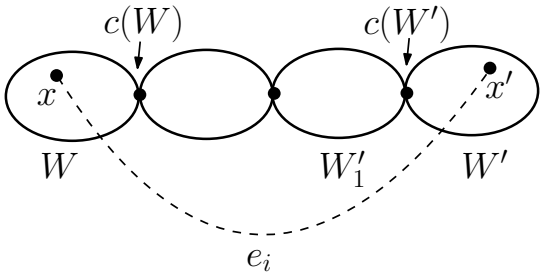


Figure 5: The graph N' depicted in block-cut form. The blocks of N' are depicted as ellipses and cut-vertices are depicted by black-filled circles at the intersection points of ellipses.

Now let B^{i-1} be the leaf-block or isolated block of N_{i-1} such that e_i has an endpoint in the interior of B^{i-1}. We use the notation $V(G)$ to denote the vertex-set of a graph G.

Lemma 3.4 *$V(B^{i-1}) \subseteq V(W)$.*

Proof. Suppose some vertex $z \neq x$ of B^{i-1} is not contained in W. Since B^{i-1} is 2-connected there exist a pair of disjoint paths in B^{i-1} connecting z and x. These paths also exist in N'. But then $c(W)$ cannot be a cut-vertex of N'. Contradiction. ■

The previous result follows from the following more general result, which is proved similarly.

Lemma 3.5 *Let D be a 2-connected subgraph of N' and suppose some edge of D lies in a block Y of N'. Then D is contained in Y.*

As a result of Lemma 3.3 and Lemma 3.4, if we fix the end-vertex of e_i in $\text{int}(W')$, then we can move the end-vertex of e_i in W to any point in the interior of B^{i-1} without destroying 2-connectivity. This is true unless $c(W)$ is contained in the interior of B^{i-1}, which is one of the cases that will be considered separately by our algorithm.

We now consider the endpoint of e_i in W'. The question we ask is: if the endpoint of e_i in W is fixed in $\text{int}(W)$, where can the other endpoint of e_i be located without destroying 2-connectivity. Of course, as long as this endpoint is contained in $\text{int}(W')$ then 2-connectivity will be maintained, but since our FPT algorithm will construct an optimal network by consecutively adding Steiner edges, block W' may not exist at the ith step of the algorithm (when e_i is added). We require a domain for this endpoint which is independent of the Steiner edges that will be added after e_i, and therefore we will choose a domain in N_0.

The next lemma will provide a basis for choosing such a domain. Let W'_1 be the neighbouring block of W' in N'.

Lemma 3.6 *Suppose that some edge of C' is contained in W'_1. Then there exists a pair of leaf-blocks of C', say R_0^{i-1} and R_1^{i-1}, such that x' is contained in a block in the path connecting R_0^{i-1} and R_1^{i-1} in $F(C')$. In addition, block R_0^{i-1} is contained in W', and block R_1^{i-1} is contained in $N' \backslash \text{int}(W')$.*

Proof. Note that x' is in $\text{int}(W')$ therefore some edge of C' is in W'. Therefore, some block of C' lies in W'. Also, since some edge of C' is in W'_1 it follows that some block

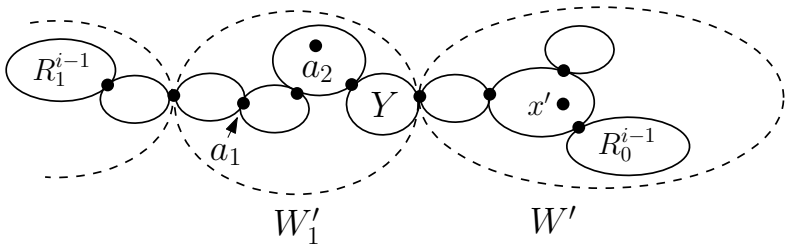


Figure 6: The block-cut tree of C' in Lemma 3.6 and blocks of N' (depicted as dashed ellipses). Observe that x' is contained in a block of C' within W'.

of C', say Y, is contained in W_1'; see Figure 6. Suppose we root $F(C')$ at Y. Then either Y is a leaf-block of C' or some leaf-block descendant of Y does not contain edges in W' (since $c(W')$ is a cut-vertex of N'). In the former case let $R_1^{i-1} = Y$, else let R_1^{i-1} be this leaf-block descendant. Now consider a maximal length path in $F(C')$ that starts at Y and passes through a block containing x'. This path must terminate at a leaf-block contained in W'. Let R_0^{i-1} be this leaf-block. ■

In the case where the assumption of the previous lemma does not hold, i.e., when C' is contained in W', we let R_0^{i-1} and R_1^{i-1} be any pair of leaf-blocks of C' such that x' is contained in a block on the path connecting R_0^{i-1} and R_1^{i-1} in $F(C')$ (note that if C' is an isolated block then simply $R_0^{i-1} = R_1^{i-1} = C'$).

Now let $P_{i-1} = B_0, B_1, ..., B_p$ be the path in $F(C')$ such that $B_0 = R_0^{i-1}$ and $B_p = R_1^{i-1}$. We create a distance function d as follows: for any vertex $a \in V(P_{i-1})$ we let $d(a) = 2j$ if a is in the interior of B_j; and if a is the cut-vertex shared by B_j and B_{j+1} then $d(a) = 2j + 1$. We say that d is the *block distance function* from R_0^{i-1} to R_1^{i-1}. As an example, refer to Figure 6. In this example the path P_{i-1} consists of 9 blocks (i.e., $p = 8$) and $d(x') = 2(1) = 2$ since x' is interior to B_1. Also, vertex a_1 is the cut-vertex shared by B_5 and B_6, so that $d(a_1) = 11$. Finally, vertex a_2 is interior to block B_4. Therefore $d(a_2) = 8$.

Function $d(a)$ essentially tells us how close a is to being in the interior of R_0^{i-1} along path P_{i-1}. As we will show below, 2-connectivity is maintained if the end-vertex of e_i in W' is located so that its distance from R_0^{i-1} (as specified by the function d) is at most the distance of x' from R_0^{i-1}, as long as this end-vertex does not coincide with $c(W')$.

3.2 Canonical optimal networks on X

We now show that there exists a certain canonical beaded 2-connected Steiner network derived from N, with optimal bottleneck length, such that the end-vertices of

all Steiner edges in this graph are contained in the sets specified in the previous subsection. Another key canonical property of this network is that each Steiner edge minimises the block distance function d of one of its end-vertices (with respect to some pair of leaf-blocks or an isolated block of N_0). The canonical network is exactly the output of our FPT algorithm.

Let ℓ^* be the length of the bottleneck of N and let $b(e_i)$ be the number of beads on e_i. Note that the length of e_i is at most $\ell^*(b(e_i)+1)$. Let $A_0^i(N)$ be the graph that results by replacing e_i by an edge e_i^* with the following property: e_i^* connects a vertex y in $\text{int}(B^{i-1})$ to a vertex y' in P_{i-1} such that $d(y')$ is minimised and the length of e_i^* is at most $\ell^*(b(e_i)+1)$.

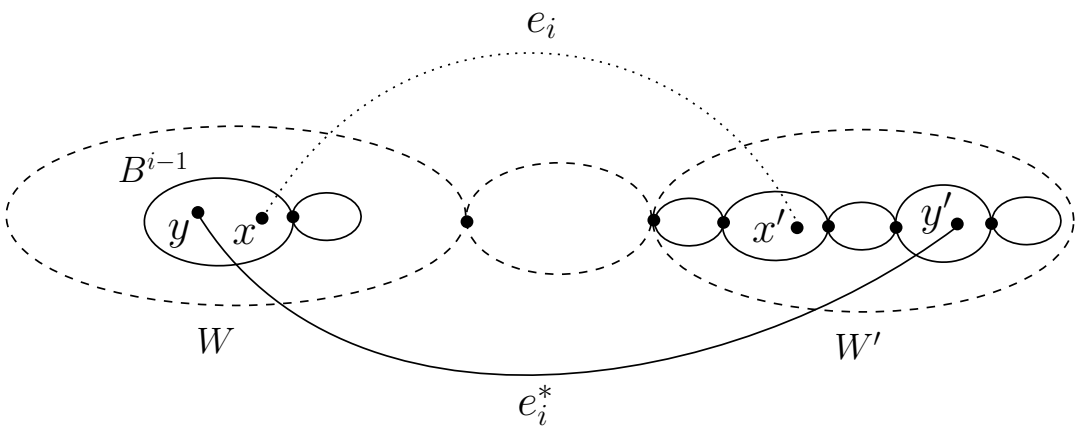


Figure 7: Construction of the graph $A_0^i(N)$.

Lemma 3.7 *If $A_0^i(N)$ is not 2-connected then either $y = c(W)$ or $y' = c(W')$, or both.*

Proof. Note first that $y \in V(W)$ because $V(B^{i-1}) \subseteq V(W)$.

Next we consider y'. Note first that $d(x') \geq d(y')$ since x' has the following properties: it is the endpoint of an edge (namely e_i) with one endpoint in $\text{int}(B^{i-1})$ and the other in P_{i-1}, and the length of e_i is at most $l^*(b(e_i)+1)$. Furthermore y' is the endpoint of such an edge that minimises d.

Let $B(x')$ be a block of C' containing x'. Therefore $B(x')$ is contained in W'. Let P be the path of blocks connecting $B(x')$ and R_0^{i-1} in C'. Then clearly P is contained in W'. Since $d(x') \geq d(y')$ it follows that y' is contained in P and therefore $y' \in V(W')$.

Now, since $y \in V(W)$ and $y' \in V(W')$, and by Lemma 3.3, $A_0^i(N)$ is 2-connected if and only if $y \in \text{int}(V(W))$ and $y' \in \text{int}(V(W'))$, the result follows. ■

We now specify four networks based on e_i. Each network is derived from N by replacing e_i by an edge $e_i(\alpha)$, where $\alpha \in \{0, ..., 3\}$. Note that Case 2a and 2b may both hold simultaneously.

Case 1 ($\alpha = 0$): Suppose first that $y \neq c(W)$ and $y' \neq c(W')$. Then $A_0^i(N)$ is 2-connected. Furthermore, $cost(A_0^i(N)) = \ell^*$. In this case we let $\alpha := 0$ and $e_i(0) := e_i^*$. See Figure 7 for an schematic example of this case.

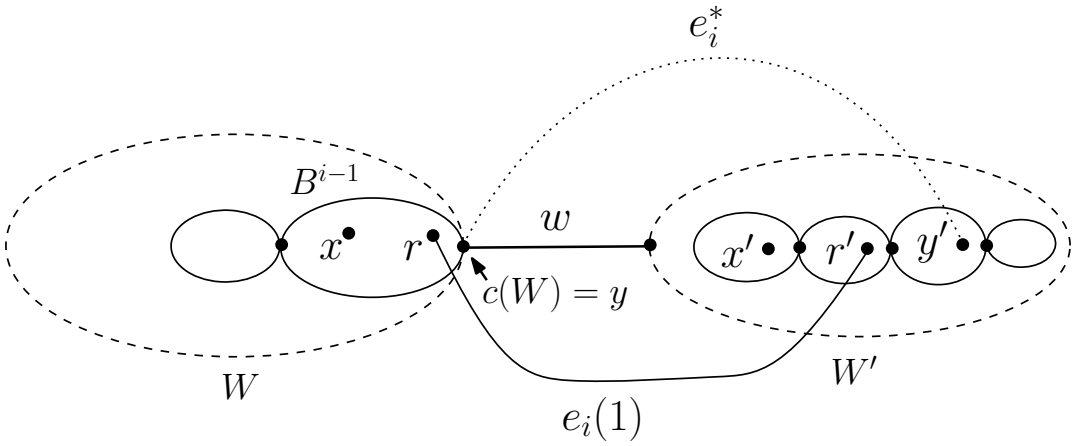


Figure 8: An example of $A_1^i(N)$ in Case 2a, where $A_1^i(N)$ is 2-connected. Note that in this case $A_0^i(N)$ is not 2-connected, since end-vertex y of e_i^* coincides with $c(W)$. Since y is in the interior of B^{i-1} this implies that the depicted edge w in this example is a Steiner edge.

Case 2a ($\alpha = 1$): Next suppose that $y = c(W)$. Let $A_1^i(N)$ be the graph that results by replacing e_i by an edge $e_i(1)$ with the following property: $e_i(1)$ connects a vertex r in $\text{int}(B^{i-1}) - \{y\}$ to a vertex r' in P_{i-1} such that $d(r')$ is minimised and the length of $e_i(1)$ is at most $l^*(b(e_i) + 1)$. Note that such an edge must exist, since $x \in \text{int}(B^{i-1}) - \{y\}$. If $r' \neq c(W')$ then $A_1^i(N)$ is 2-connected and $cost(A_1^i(N) = \ell^*$. An illustrative example of $A_1^i(N)$ is given in Figure 8.

Case 2b ($\alpha = 2$): Suppose that $y' = c(W')$. This case is similar to Case 2a. As before we create a graph $A_2^i(N)$ where $e_i(2)$ has replaced e_i, and the end-vertices of $e_i(2)$ are $s \in \text{int}(B^{i-1})$ and $s' \in C' - \{y'\}$.

Case 3 ($\alpha = 3$): Suppose that either Case 2a holds but $A_1^i(N)$ is not 2-connected, or Case 2b holds but $A_2^i(N)$ is not 2-connected. We assume that Case 2a holds; the other case is similar and gives the same output. Since $A_1^i(N)$ is not 2-connected we must have $r' = c(W')$. Let $A_3^i(N)$ be the graph that results by replacing e_i by an edge $e_i(3)$ with the following property: $e_i(3)$ connects a vertex u in $\text{int}(B^{i-1}) - \{y\}$ to a vertex u' in $C' - \{r'\}$ such that $d(u')$ is minimised and the length of $e_i(3)$ is at most $l^*(b(e_i) + 1)$. Note that such an edge must exist, since $x \in \text{int}(B^{i-1}) - \{y\}$

and $x' \in C' - \{r'\}$. Finally, note that by Lemma 3.3, $A_3^i(N)$ is 2-connected and, by definition, $cost(A_3^i(N)) = \ell^*$.

Starting from N, we now run Algorithm 2 to convert N into a canonical beaded 2-connected Steiner network whilst maintaining the same minimum cost.

Algorithm 2 (Converting N into a canonical network)

Input: N
Output: N
1: **for** $i := 1 : q$ **do**
2: Find $\alpha \in \{0, ..., 3\}$ such that $A_\alpha^i(N)$ is defined and is 2-connected
3: $N := A_\alpha^i(N)$

3.3 The FPT algorithm

Algorithm 3 below is a formal description of our FPT algorithm. The algorithm takes as input a set X of points in the plane and an integer k representing the number of Steiner points. The output is N_{bead}, which is a Euclidean minimum beaded 2-connected bottleneck k-Steiner network on X.

Intuitively, our algorithm first constructs a complete graph M on X and then selects a value ℓ as the potentially optimal bottleneck length. Next it constructs an underlying graph M_0 consisting of all edges of M of length at most ℓ (i.e., when ℓ is known then the non-Steiner edges of N_{bead} are also known). Next the algorithm selects the number of Steiner points k_i on each edge Steiner edge e_i that will be added to M_0. The Steiner edges are then added one at a time, where at each step a leaf-block interior or isolated block of the current graph is selected to contain one end-vertex of the Steiner edge, and a path of blocks of M_0 is selected to contain the other. All four cases $\alpha \in \{0, ..., 3\}$ are then generated (as described above) as potentially optimal solutions. Once all Steiner edges have been added and all combinations considered, the algorithm outputs a cheapest 2-connected network found. The individual lines of Algorithm 3 are discussed in detail in the proof of Theorem 3.9.

We now prove the correctness and time complexity of Algorithm 3.

Lemma 3.8 *(see [16]) The block-cut forest of a graph with n vertices can be constructed in time $O(n^2)$.*

Theorem 3.9 *Algorithm 3 outputs a Euclidean minimum beaded 2-connected bottleneck k-Steiner network on X in time $O(n^4k^6 2^k)$.*

Algorithm 3 (The FPT algorithm)

Input: A set X of n points in the plane and an integer $k \geq 0$

Output: A network N_{bead} which is a Euclidean minimum beaded 2-connected bottleneck k-Steiner network on X.

Construct M, the complete graph on X
Calculate L, the set of edge lengths of edges in M
$costopt := \infty$
for every $\ell' \in L$ and every $k' \in \{1, ..., k+1\}$ **do**
  $\ell := \ell'/k'$
  Construct M_0, the subgraph of M containing all edges of length at most ℓ.
  if M_0 has more than $2k$ leaf-blocks or isolated blocks **then**
    Exit
  else if $k = 0$ **and** $cost(M_0) < costopt$ **and** M_0 is 2-connected **then**
    $costopt := cost(M_0)$
    $N_{\text{bead}} := M_0$
    Exit
  end if
  Assign m' to be the number of edges in $M \backslash M_0$
  Construct $\mathcal{P}$, the set of all ordered partitions of all $k'' \leq k$ containing at most m' elements
  for each $\{k_1, ..., k_m\} \in \mathcal{P}$ **do**
    $counter(0) := 0$
    $M_0(counter(0)) := M_0$
    for $i := 1 : m$ **do**
      $counter(i) = 0$
      for $t := 0 : counter(i-1)$ **do**
        for every leaf-block or isolated block B_{i-1} of $M_{i-1}(t)$ and every directed path P_{i-1} in $F(M_0)$ **do**
          Calculate R_0^{i-1} and R_1^{i-1}, the initial and final endpoints of P_{i-1} respectively
          Construct d, the block distance function from R_0^{i-1} to R_1^{i-1}
          for $\alpha \in \{0, ..., 3\}$ **do**
            $M_i(counter(i)) := M_{i-1}(t) \cup \{e_i(\alpha)\}$
            Place k_i equally spaced beads on $e_i(\alpha)$
            if $i = m$ **and** $M_i(counter(i))$ is 2-connected **and** $cost(M_i(counter(i))) < costopt$ **then**
              $costopt := cost(M_i(counter(i))$
              $N_{\text{bead}} := M_i(counter(i))$
            end if
            $counter(i) := counter(i) + 1$

Proof. We first show correctness of the algorithm. Let N be the optimal canonical solution for X and let $e_1, ..., e_q$ be the Steiner edges of N, ordered according to Proposition 3.1. Now, Line 4 of Algorithm 3 iterates through every possible bottleneck length and therefore must select the optimal bottleneck length $\ell = \ell^*$ at some iteration. Therefore, since N is bottleneck complete, we may assume that $M_0 = N_0$. Since, by Corollary 3.2, N_0 has at most $2k$ leaf-blocks or isolated blocks, in Line 7 the current loop iteration is exited if this condition does not also hold for M_0. Clearly if $k = 0$ then M_0 for this iteration must be optimal (this case is handled in Line 9).

In Line 16 the algorithm considers every possible ordered partition of k containing $m \leq m'$ elements, where m' is the number of edges in $M \backslash M_0$. Therefore some iteration must find the partition $\{k_1, ..., k_q\}$, where k_i is the number of beads on e_i in N.

When Algorithm 3 prepares to add the first Steiner edge ($i = 1$ in Line 19) there are a number of choices depending on the leaf-block interior or isolated block which is to contain one end-point of the Steiner edge, and the path of blocks (with both orderings of the path used to define d) containing the other end-point. A choice of α is also made in Line 25 for selecting $e_1(\alpha)$. The algorithm keeps track of all the networks resulting from these choices by means of the variable $counter(1)$. In general, the variable $counter(i)$ initialised in Line 17 and Line 20 keeps track of the number of networks constructed by Line 26 during the ith iteration of the loop in Line 19. Specifically, for each t in Line 21 the graph $M_{i-1}(t)$ is the tth graph produced in the $(i-1)$th iteration of the loop in Line 19.

Therefore, since all choices are considered, in the ith iteration of the loop in Line 19 some choice will construct the graph $A^i_\alpha(N)$ produced in the ith iteration of Algorithm 1. Therefore N will be constructed in the qth iteration of Algorithm 3. This implies that Algorithm 3 is correct and produces a Euclidean minimum beaded 2-connected bottleneck k-Steiner network as output.

Next we analyse the time complexity. The loop in Line 4 iterates $O(n^2k)$ times. In Line 7 the algorithm needs to construct the block-cut forest $F(M_0)$. This requires $O(n^2)$ time and can be reused in Line 9 to check if M_0 is 2-connected.

In Line 16, the number of ordered partitions of all $k'' \leq k$ is at most $O(k2^k)$. Note also that $m \leq k$, so that the loop in Line 19 runs at most k times. Next we bound $counter(i)$, since this bounds the number of all iterations of all loops from Line 21 onwards. The number of leaf-blocks or isolated blocks of $M_{i-1}(t)$ is at most $2k$, and therefore the number of paths of blocks of M_0 is at most $O(k^2)$. Therefore the loop in Line 22 iterates at most $O(k^3)$ times. Therefore $counter(i)$ is at most $O(k^3)$.

Note that when calculating $M_i(counter(i))$ in Line 26 the algorithm updates the block-cut forest (in time at most $O(n^2)$) so that this information is available in constant time when selecting blocks in Line 22 and Line 23. Finally, note that constructing $e_i(\alpha)$ requires at most $O(n^2)$ time, since we need to choose an end-vertex for $e_i(\alpha)$

in B_{i-1} and an end-vertex in P_{i-1}, both of which have cardinality at most $O(n)$.

Total time complexity is therefore $O(n^2k \cdot (n^2 + k2^k \cdot k \cdot k^3 \cdot (n^2 + n^2)) = O(n^4k^6 2^k)$.

■

Corollary 3.10 *Algorithm 3 is a fixed parameter tractable algorithm with respect to k.*

4 Conclusion

This paper has presented the first fixed parameter tractable algorithm for the beaded 2-connected bottleneck Steiner network problem. By employing block-cut tree structures we showed that the problem can be exactly solved in a time of $O(n^4k^6 2^k)$, where n is the number of terminals and k is the number of Steiner points. This implies that the combinatorial explosion in this problem is due to the number of Steiner points, and consequently the beaded 2-connected bottleneck Steiner network problem is tractable for small k. An important open problem in this field, which is also a future research direction for the author, is the development of a FPT algorithm for the case when the degrees of Steiner points are not restricted.

References

[1] A.K. Abu-Affash, P. Carmi, M.J. Katz and M. Segal, *The Euclidean bottleneck Steiner path problem and other applications of (α, β)-pair decomposition*, Discrete and Computational Geometry **51** (2014), 1–23.

[2] T. Andreae and H-J Bandelt, *Performance guarantees for approximation algorithms depending on parametrized triangle inequalities*, SIAM Journal on Discrete Mathematics **8** (1995), 1–16.

[3] S.W Bae, S. Choi, C. Lee and S. Tanigawa, *Exact algorithms for the bottleneck Steiner tree problem*, Proceedings of 20th International Symposium on Algorithms and Computation, LNCS, vol. 5878, 2009, pp. 24–33.

[4] M. Brazil, C.J. Ras and D.A. Thomas, *The bottleneck 2-connected k-Steiner network problem for $k \leq 2$*, Discrete Applied Mathematics **160** (2012), 1028–1038.

[5] M. Brazil, C.J. Ras and D.A. Thomas, *An exact algorithm for the bottleneck 2-connected k-Steiner network problem in L_p planes*, Discrete Applied Mathematics **201** (2016), 47–69.

[6] M. Brazil, C.J. Ras, D.A. Thomas and G. Xu, *The* 2*-connected bottleneck Steiner network problem is NP-hard in any* L_p *plane*, arXiv:1907.03474.

[7] M.S. Chang, C.Y. Tang and R.C.T. Lee, *Solving the Euclidean bottleneck biconnected edge subgraph problem by* 2*-relative neighborhood graphs*, Discrete Applied Mathematics **39** (1992), 1–12.

[8] G.A. Dirac, *Minimally* 2*-connected graphs*, J. Reine Angew. Math. **228** (1967), 204–216.

[9] M. Grotschel, C. Monma and M. Stoer, *Design of survivable networks*, Handbooks in Operations Research and Management Science **7** (1995), 617–672.

[10] K. Hvam, L. Reinhardt, P. Winter and M. Zachariasen, *Bounding component sizes of two-connected Steiner networks*, Information Processing Letters **104** (2007), 159–163.

[11] E.L. Luebke and J.S. Provan, *On the structure and complexity of the* 2*-connected Steiner network problem in the plane*, Operations Research Letters **26** (2000), 111–116.

[12] G.S. Manku, *A linear time algorithm for the Bottleneck Biconnected Spanning Subgraph problem*, Information Processing Letters **59** (1996), 1–7.

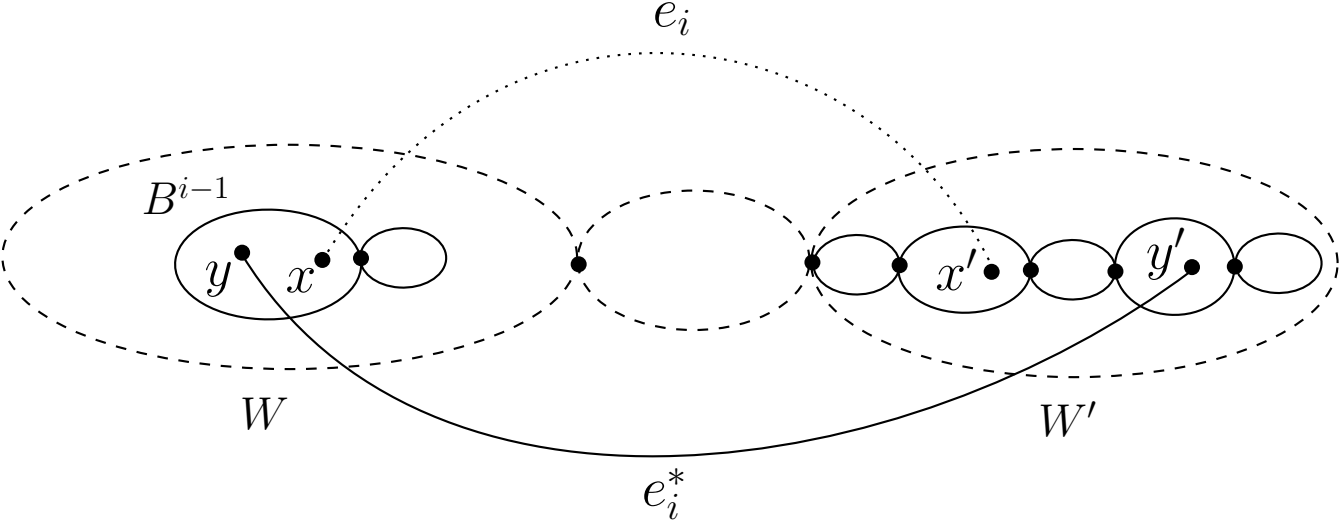
[13] C. Ribeiro and M. Souza, *Variable neighborhood search for the degree-constrained minimum spanning tree problem*, Discrete Applied Mathematics **118** (2002), 43–54.

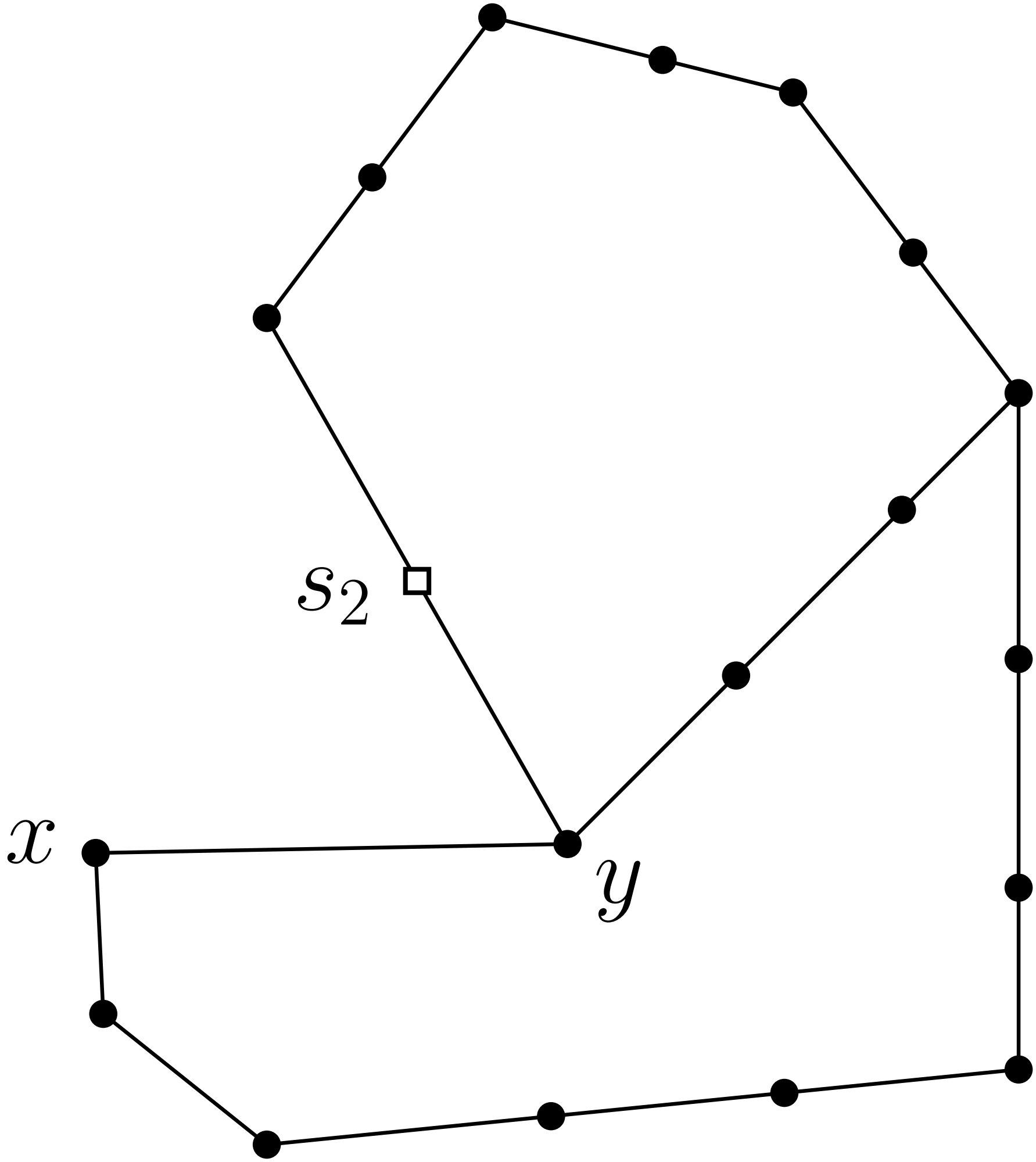
[14] M. Sarrafzadeh and C.K. Wong, Bottleneck *Steiner trees in the plane*, IEEE Trans. Comput **41** (1992), 370–374.

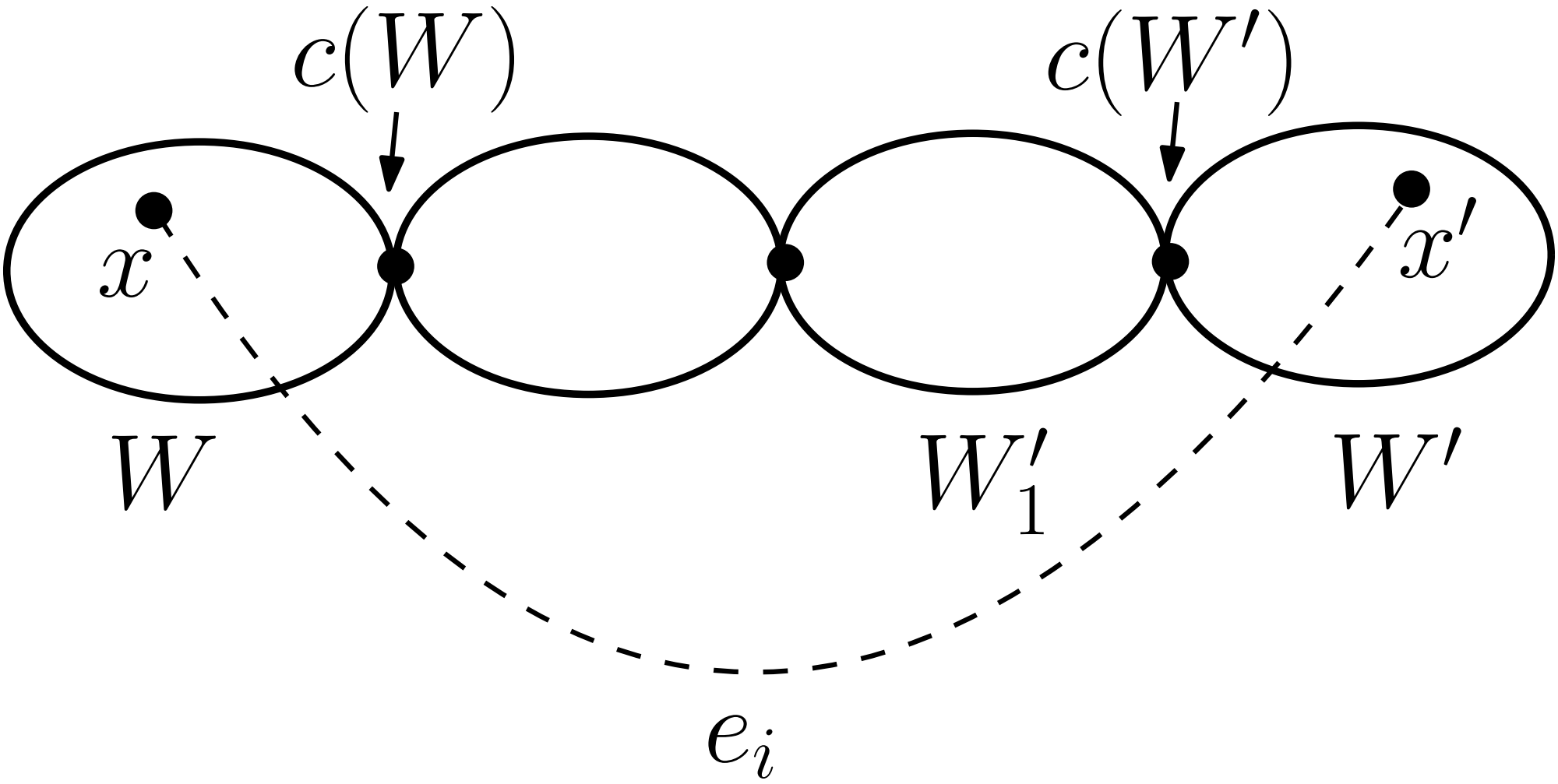
[15] M. Sekanina, *On an ordering of the set of vertices of a connected graph*, Publications of the Faculty of Science, University of Brno, **412** (1960), 137–142.

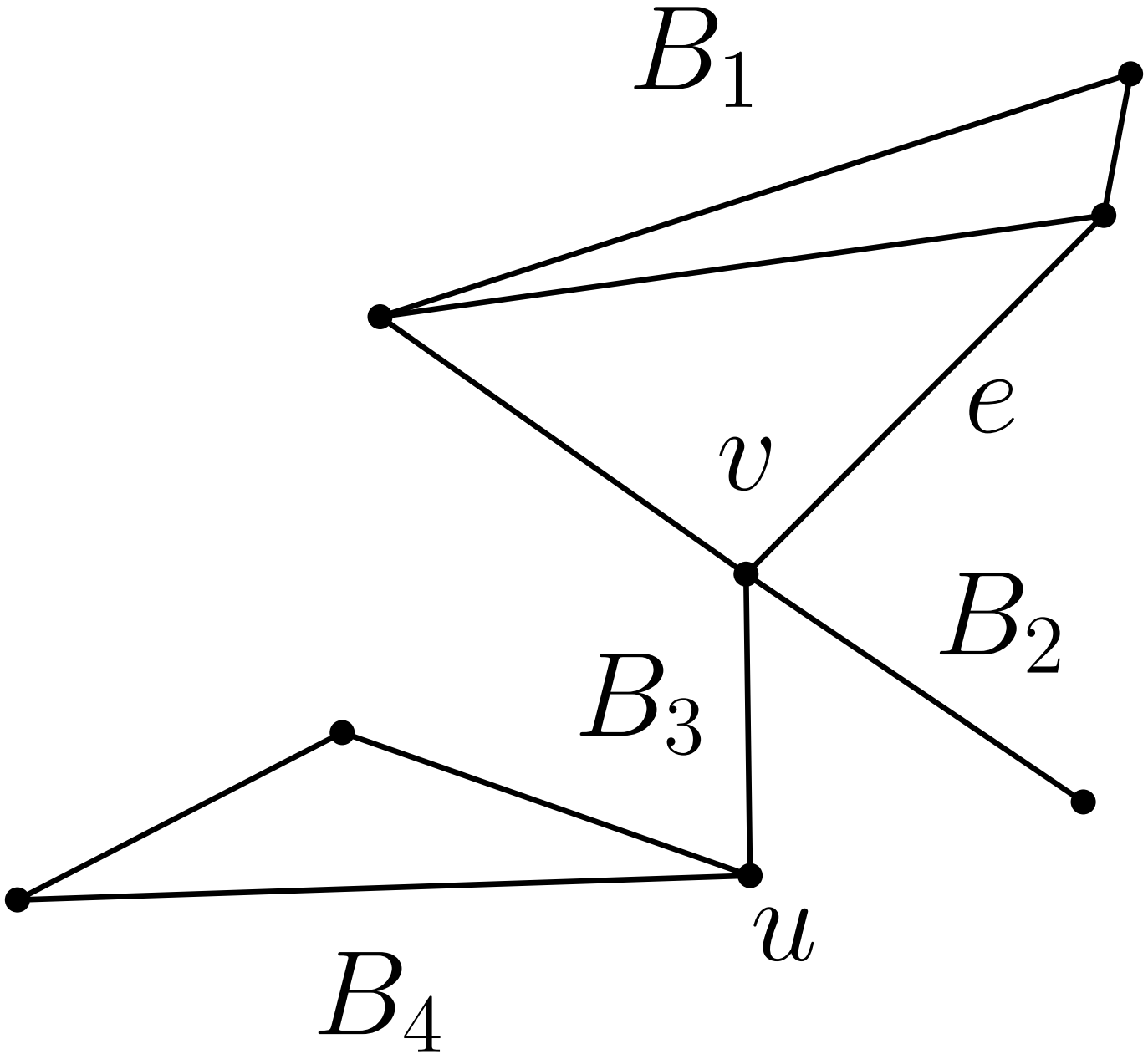
[16] R. Tarjan, *Depth first search and linear graph algorithms*, SIAM Journal on Computing **1** (1972), 146–160.

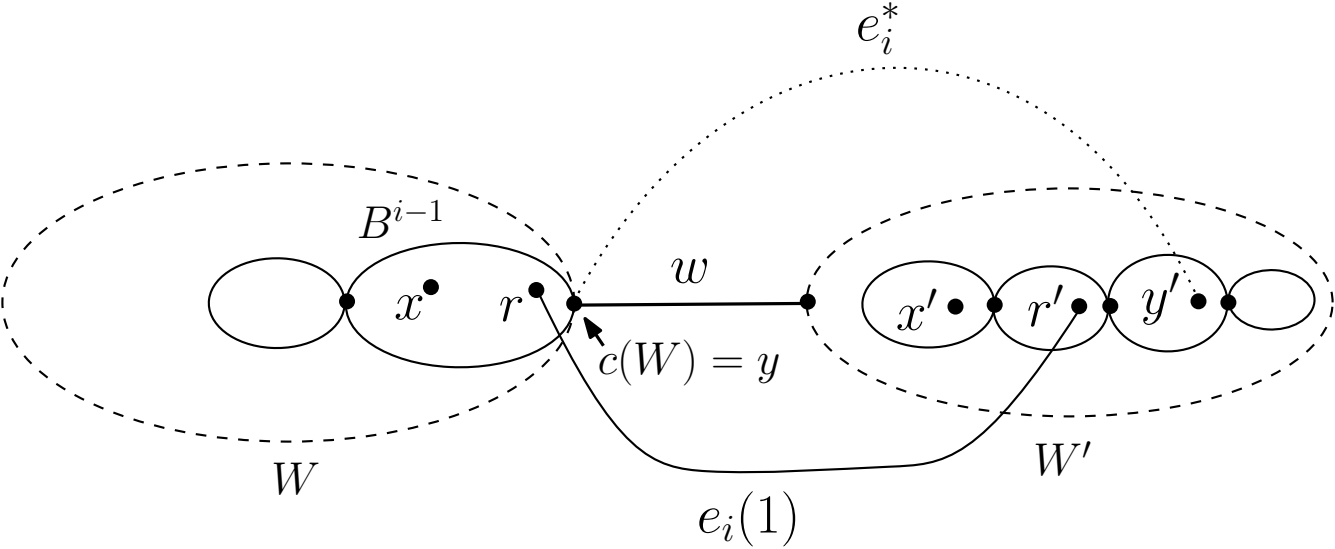
[17] L. Wang and D.Z. Du, *Approximations for a bottleneck Steiner tree problem*, Algorithmica **32** (2002), 554–561.

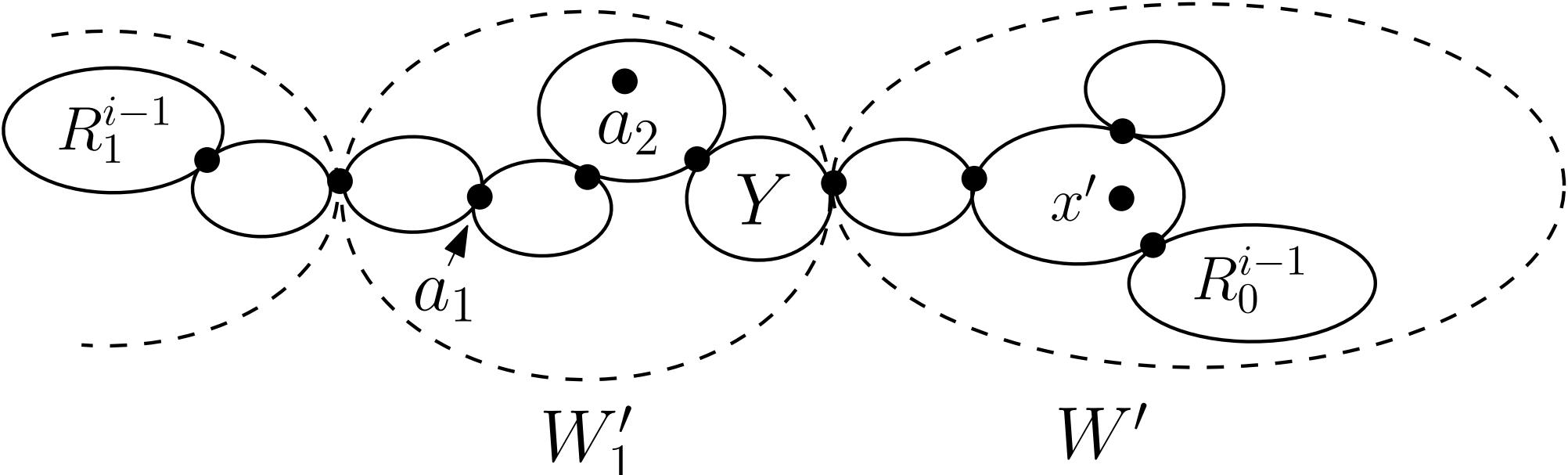

e_i
B^{i-1}
y
x
x'
y'
W
W'
e_i^*

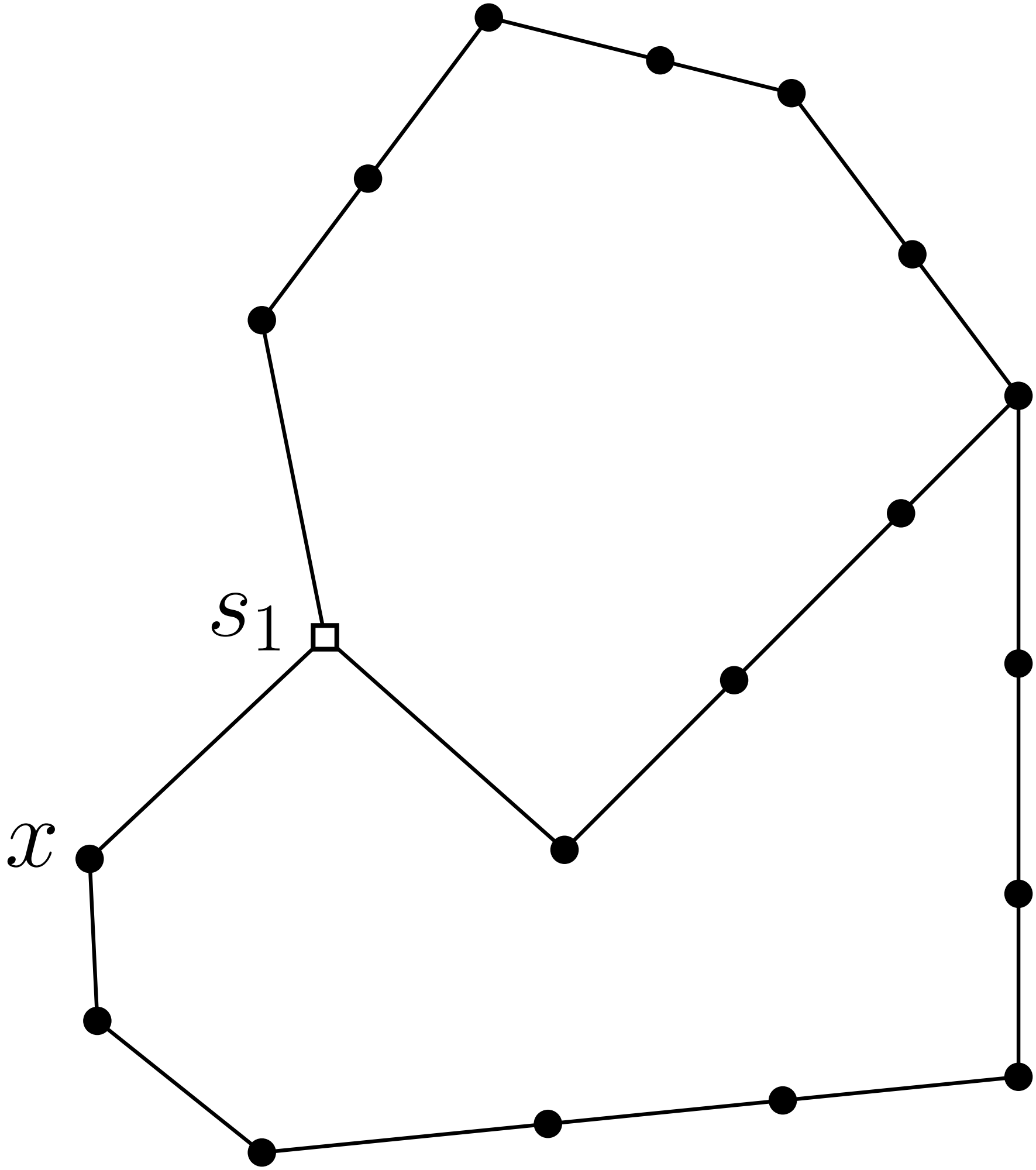

s_2
x
y


$c(W)$
$c(W')$
x
x'
W
W_1'
W'
e_i


B_1
e
v
B_2
B_3
u
B_4


e_i^*
B^{i-1}
w
x
r
$c(W) = y$
x'
r'
y'
W
W'
$e_i(1)$


R_1^{i-1}
a_2
Y
x'
a_1
R_0^{i-1}
W_1'
W'


s_1
x