Algorithmica (2017) 78:990–1019
DOI 10.1007/s00453-016-0191-2

Connectivity Graphs of Uncertainty Regions

Erin Chambers[1] · Alejandro Erickson[2] · Sándor P. Fekete[3] · Jonathan Lenchner[4] · Jeff Sember[5] · Venkatesh Srinivasan[6] · Ulrike Stege[6] · Svetlana Stolpner[7] · Christophe Weibel[8] · Sue Whitesides[6]

Received: 1 March 2014 / Accepted: 21 July 2016 / Published online: 3 August 2016

Abstract We study connectivity relations among points, where the precise location of each input point lies in a region of uncertainty. We distinguish two fundamental scenarios under which uncertainty arises. In the favorable *Best-Case Uncertainty*, each

A preliminary extended abstract summarizing parts of this paper appears in [5].

✉ Sándor P. Fekete
s.fekete@u-bs.de

Erin Chambers
echambe5@slu.edu

Alejandro Erickson
alejandro.erickson@gmail.com

Jonathan Lenchner
lenchner@us.ibm.com

Jeff Sember
jpsember@cs.ubc.ca

Venkatesh Srinivasan
srinivas@uvic.ca

Ulrike Stege
ustege@uvic.ca

Svetlana Stolpner
svetlana@augsignals.com

Christophe Weibel
christophe.weibel@gmail.com

Sue Whitesides
sue@uvic.ca

1 Department of Computer Science, St. Louis University, St. Louis, MO, USA

2 School of Engineering and Computing Sciences, Durham University, Durham, UK

input point can be chosen from a given set to yield the best possible objective value. In the unfavorable *Worst-Case Uncertainty*, the input set has worst possible objective value among all possible point locations, which are uncertain due, for example, to imprecise data. We consider these notions of uncertainty for the bottleneck spanning tree problem, giving rise to the following *Best-Case Connectivity with Uncertainty* problem: given a family of geometric regions, choose one point per region, such that the longest edge length of an associated geometric spanning tree is minimized. We show that this problem is NP-hard even for very simple scenarios in which the regions are line segments or squares. On the other hand, we give an exact solution for the case in which there are $n + k$ regions, where k of the regions are line segments and n of the regions are fixed points. We then give approximation algorithms for cases where the regions are either all line segments or all unit discs. We also provide approximation methods for the corresponding *Worst-Case Connectivity with Uncertainty* problem: Given a set of uncertainty regions, find the minimal distance r such that for any choice of points, one per region, there is a spanning tree among the points with edge length at most r.

Keywords Uncertainty · Geometric optimization · Connectivity · Worst-case connectivity · Best-case connectivity

1 Introduction

Finding an optimally connected substructure in a network is one of the fundamental combinatorial optimization problems in network design. The standard problem of minimizing the total edge cost in the network amounts to finding a minimum spanning tree, which can be computed by straightforward greedy methods. A closely related problem that has gained in importance in the context of wireless networking is to consider the "bottleneck" problem of minimizing the length of the longest edge. A solution to this problem allows one to choose a point set of lowest power, where equi-power routers are to be placed at nodes, such that a message can be relayed between any two nodes. However, the situation changes when the locations of devices becomes part of the problem: How should each location be chosen from a given neighborhood, such that the solution to the resulting bottleneck connectivity problem is minimized? The neighborhoods can be the result of imprecise input data, or simply arise from a geometric range of possible locations; depending on the scenario, the choice of locations can be optimistic (i.e., best case) or adversarial (i.e., worst case).

[3] Department of Computer Science, Braunschweig University of Technology, Braunschweig, Germany

[4] IBM Research Africa, Nairobi, Kenya

[5] Department of Computer Science, University of British Columbia, Vancouver, BC, Canada

[6] Department of Computer Science, University of Victoria, Victoria, BC, Canada

[7] A.U.G. Signals Ltd., Toronto, Canada

[8] Google, Zurich, Switzerland

Let U, $|U| = n$, denote a family of uncertainty regions, e.g., a family of disks, squares, line segments or pairs of points. For each uncertainty region $u_i \in U$, $1 \leq i \leq n$, one point p_i is to be chosen inside this region u_i. Let P be the set of points chosen. For a value $\alpha \in \mathbb{R}$, we define the *connectivity graph* $G_\alpha = (V, E)$ of P with respect to α as follows: $V = P$ and $E = \{(p_i, p_j) \in P \times P, \|p_i - p_j\|_2 \leq 2\alpha\}$. Thus, the graph connects a pair of points with an edge whenever closed disks of radius α centered at these points intersect. We can now formally define the main problem, *Best-Case Connectivity with Uncertainty* (BCU), that we study in this paper.

The BCU Problem. Given a set $U = \{u_1, \ldots, u_n\}$ of n uncertainty regions, find the minimum value α for which there exists a choice of point set $P = \{p_1, \ldots, p_n\}$, $p_i \in u_i$, such that the connectivity graph G_α of P is connected.
We further study a closely related problem, *Worst-Case Connectivity with Uncertainty* (WCU):

The WCU Problem. Given a set $U = \{u_1, \ldots, u_n\}$ of n uncertainty regions, find the minimum value α, such that for *any* choice of point set $P = \{p_1, \ldots, p_n\}$, $p_i \in u_i$, the connectivity graph G_α of P is connected.

1.1 Related Work

If the n uncertainty regions are points (in other words, there is no uncertainty), then finding the minimum α for which the connectivity graph is connected amounts to finding a minimum Euclidean Bottleneck Spanning Tree (MBST) on the points. Because minimum spanning trees (MSTs) are also MBSTs, a solution can be found in time $O(n \log n)$.

Closely related to our BCU and WCU problems is the well-studied family of range assignment problems. In these problems, the disks centered at the points can be of different radius, and the goal is to minimize the total power consumption under the constraint that the network satisfies certain structural properties like connectivity, strong connectivity, or a particular broadcast property. Most of the work on these problems has considered point sets rather than uncertainty regions (see [2,6,16,19,20]). Thus our work provides an early exploration of connectivity problems, arising in the context of wireless networks, for points lying in nontrivial uncertainty regions.

The minimum spanning tree problem (MST) has been studied in the setting of uncertainty regions. Yang et al. [29] showed that the problem of computing a spanning tree that minimizes the total edge length is NP-hard if the uncertainty regions are non-overlapping unit disks or rectangles. They also give a polynomial-time approximation scheme (PTAS) for the case in which the uncertainty regions are unit disks; this is notably different from our problem, which does not admit a PTAS, unless P = NP. Further approximation results for minimization and maximization versions of the MST with uncertainty were provided by Dorrigiv et al. [11]. Another optimization problem with neighborhoods that have received attention is the Traveling Salesman Problem; e.g. see [4,7,12,14,18,24,25]. The bottleneck version of TSP is known to be NP-hard [17, p. 212]. A 2-approximation has been known since 1984 [27].

Other work on geometric optimization with uncertainty regions has been framed using the notions of imprecise data or neighborhoods. For studies of shortest paths with uncertainty regions see [9,10,26]. For a more general treatment and discussion of problems such as convex hulls, see Löffler and van Kreveld [22], who also considered the size of bounding boxs, diameters and related problems in [23]. A discrete variant in d-dimensional space was considered by Ding and Xu [8], who studied the problem of picking one representative each from a family of finite sets, such that the resulting set has a small enclosing hypersphere. Fiala et al. [15] considered "systems of distant representatives", which amounts to maximizing the minimum distance between selected points. This is related to but different from our work in this paper: we aim for minimizing the maximum length in a spanning tree.

Another angle is to consider topological changes under uncertainty: when does the structure of an optimal solution change when the input data is perturbed? Abellanas et al. [1] studied this structure with respect to the largest perturbation of a set of planar points that keeps the Delaunay triangulation unchanged. Conversely, problems of determining a necessary perturbation in order to achieve a desired change have also been studied; e.g., see Arkin et al. [3] for deciding whether a given set of neighborhoods has a convex stabber, which amounts to deciding whether a given set can be moved into a convex position. For further discussions of related problems, see the excellent exposition by Löffler and van Kreveld [22].

1.2 Our Main Results

After showing that several variants of BCU are NP-hard (some even to approximate), we give exact and approximation algorithms for certain variants. Given the geometric nature of our problems, we use the Euclidean measure of distance. Our main results are as follows:

1. We show that BCU is NP-hard even in the simple cases in which the uncertainty regions are point pairs and vertical line segments, respectively. Our proof technique also works when the regions are all squares. We further show that it is NP-hard to approximate BCU within a factor less than $\sqrt{5}/2$ when the uncertainty regions are pairs of points. See Sect. 2.
2. We present an exact algorithm for BCU when the instance consists of n fixed points and k line segments. The algorithm is polynomial in n for constant k. The output of this algorithm is correct up to precision δ, $\delta > 0$. See Sect. 3.
3. For uncertainty regions that are all unit disks, we give a simple constant additive approximation algorithm for BCU. A slight modification of this algorithm gives a constant multiplicative approximation in case the disks are *non-overlapping*. See Sect. 4.
4. We provide approximation results for the WCU. In particular, we establish methods with additive and multiplicative performance guarantees. See Sect. 5.

2 Hardness Results for BCU

We prove hardness results for three variants of the BCU problem. Our first main result shows NP-hardness when the uncertainty regions are point pairs (Theorem 1). Inter-

estingly, this result also implies a hardness of approximation result for the case of point pairs (Theorem 2), and NP-hardness when the uncertainty regions are line segments (Theorem 3). Our second main result shows NP-hardness when the uncertainty regions are unit squares (Theorem 4). We assume, in all cases, that the uncertainty regions are non-overlapping. All of our reductions are from Planar 3-SAT – in other words 3-SAT with the added condition that the input formula can be represented as a planar graph.

2.1 BCU When Uncertainty Regions Are Point Pairs

We consider the BCU problem for uncertainty regions of vertically aligned pairs of points, unit distance apart with integer coordinates. We study the decision version of the BCU problem for $\alpha = 1$, *i.e.*, we want to decide if $G_\alpha = G_1$ is connected for some choice of points, one for each uncertainty pair.

Theorem 1 *It is NP-hard to find an exact solution to the BCU problem for the case in which the regions of uncertainty are point pairs that have a vertical distance of length one.*

Proof We show this problem is NP-hard, using a reduction from the following formulation of Planar 3-SAT. Let $\Phi = (X, C)$ be an instance of 3-SAT, with variables $X = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$. Each clause consists of exactly three literals, each a variable or its negation. For such an instance, we define a formula graph $H(\Phi)$ as follows: $H(\Phi) = (V, E)$ with vertex set $V = X \cup C$ and edge set $E = E_1 \cup E_2$, such that $E_1 = \{(x_i, x_{i+1}) | i < n\}$, and $E_2 = \{(x_i, c_j) | c_j \text{ contains } x_i \text{ or } \overline{x_i}\}$. A Planar 3-SAT instance is one whose corresponding formula graph $H(\Phi)$ is planar. In the Planar 3-SAT problem, our goal is to determine whether a given Planar 3-SAT instance Φ is satisfiable. This problem is known to be NP-complete [17,21].

Our reduction makes use of the fact that, given a Planar 3-SAT instance Φ with formula graph $H(\Phi)$, this graph has a planar layout on an $O(n + m) \times O(n + m)$ grid [13,28]. Further, in this layout, the vertices (variables and clauses) can be drawn as horizontal line segments and edges as vertical line segments. Henceforth, we equate the formula graph $H(\Phi)$ with the planar layout we have described above.

To reduce from Planar 3-SAT to BCU when the uncertainty regions are pairs of points, we design various gadgets. Specifically, given a layout of a Planar 3-SAT instance using line segments as described above, we replace each horizontal line segment corresponding to a variable by a variable gadget, each horizontal line segment corresponding to a clause by a clause gadget, and each vertical line segment corresponding to an edge in E_1 by a variable-variable connector and each one corresponding to an edge in E_2 by a variable-clause connector. Below we will argue that there exists a choice of point in each of these uncertainty pairs such that the connectivity graph for $\alpha = 1$, G_1, is connected if and only if the corresponding Planar 3-SAT instance is satisfiable. □

2.1.1 Overview of the Gadgets

We present the main ideas behind the clause gadgets, variable gadgets, and connector gadgets mentioned above.

A clause gadget is designed so that it contains three "gates", one for each of the literals in the clause. The gate for each literal is either on the top or the bottom of the clause gadget, depending on whether the literal appears above or below the clause in the planar grid layout of $H(\Phi)$. For the connectivity subgraph corresponding to the clause to be connected to the rest of the graph in G_1, at least one of these three gates must be open. This corresponds to setting the literal to `True` in the clause. This, in turn, ensures that the clause is satisfied.

The role of a variable gadget is to choose and propagate a truth value for the variable to all the clauses containing it in a consistent manner. The variable gadget contains three types of constructs. Type *I* and type *II* constructs help link the variable to all the clauses that contain it and are either above or below it. We have one such type *I*-type *II* pair for every occurrence of the variable in a clause. A construct of type *III* is used to ensure that the subgraph corresponding to the variable gadget can be connected if and only if the truth assignment to the variable in all the copies of type *I*-type *II* pairs are the same. Our construction also prevents subgraphs arising from parts of a variable gadget from being connected through other parts of G_1.

The variable and clause gadgets are linked to each other using two types of connectors. A clause-variable connector replaces an edge of $H(\Phi)$ between a clause vertex and a variable vertex in such a way that points chosen in the corresponding gadgets can be connected through it if and only if the truth value of the variable is consistent with its occurrence (as a literal) in the clause. A variable-variable connector replaces an edge of $H(\Phi)$ between two variable vertices. Points in one variable gadget can be connected to those in another variable gadget via points chosen in the variable-variable connector irrespective of the choice of truth values for each variable gadget.

Having given the overview of the reduction, we now provide a detailed proof by first describing the different gadgets in detail and then arguing the correctness of our reduction.

2.1.2 The Clause Gadget

Figure 1 depicts a schema that describes the functioning of a clause gadget. Each gate in the schema represents the entry of a connection to a literal, with an open gate representing a contribution of `True`. If all gates are closed, then, as suggested by the schema, it is possible that the connectivity graph of the clause gadget is connected, but it is isolated from the rest of the graph. Also, as the schema suggests, if gates to

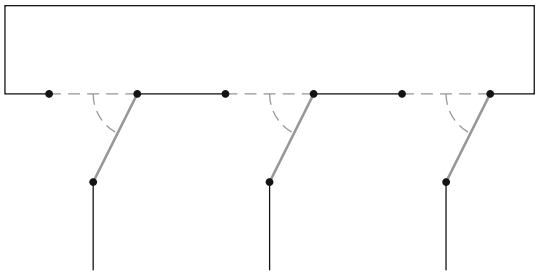

Fig. 1 Schema of the clause gadget. When a *gray* gate is open the entire clause gadget can be connected to the rest of the graph

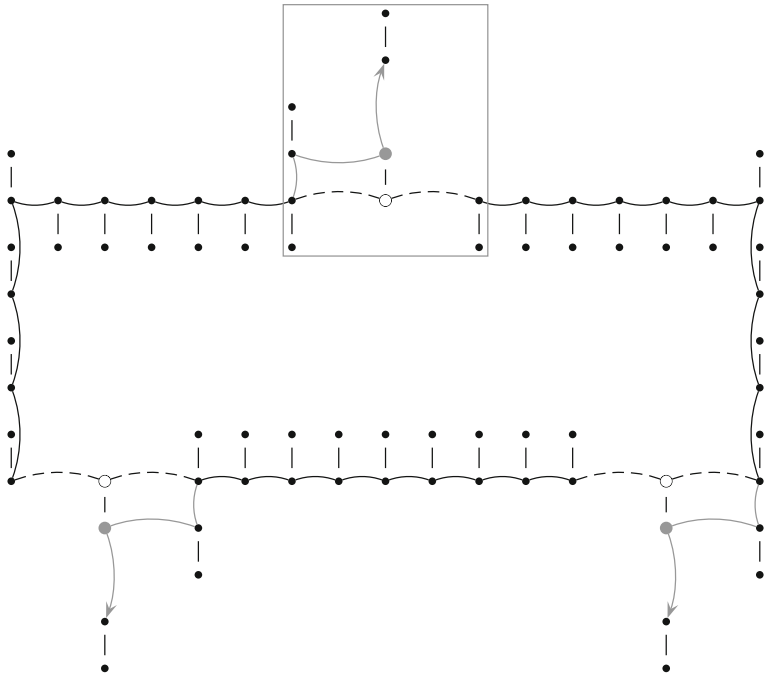

Fig. 2 An example clause gadget. As shown, aside from vertical sequences of uncertainty pairs leading to variable gadgets, there are no other uncertainty pairs in the vicinity of the clause gadget. The graph G_1 can be connected only if at least one of the *gray* points is chosen; that is, the attached literal is set to `True`. The gate inside the *gray box* can be moved up one unit to meet variable-clause connectors at different heights, if necessary

two literals x_i and x_j are both open, then connections are created between the clause gadget and the gadgets for the literals, but no connection via the clause gadget is made between the variable gadgets.

The shape of the clause gadget can be adapted to meet the requirements of the clause vertex it represents in the planar layout of $H(\Phi)$ (e.g., the length of the horizontal line segment representing a particular clause gadget in the planar layout of $H(\Phi)$ by line segments; however many of the horizontal segments representing literals contained in the clause lie below the segment representing the clause and however many above). Figure 2 shows an example of a clause gadget where, in the representation of $H(\Phi)$, the clause was represented by a horizontal segment connected to one horizontal variable segment lying above, and two horizontal variable segments lying below the segment for the clause. The clause gadget is flexible, as its size can be adjusted by adding more uncertainty pairs to the sequence between two connections to the variable gadgets, and to the sequence between connections to variable gadgets and the left and right sides of the gadget. Furthermore, in a straightforward manner we can modify the clause gadget to move the connection to a particular variable gadget vertically by one unit without moving the entire clause gadget; e.g., the uncertainty pairs in the gray box in Fig. 2 can be moved up by one unit.

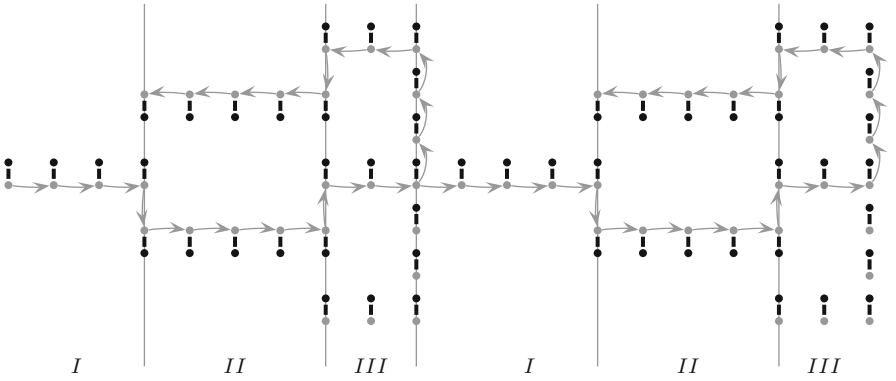

Fig. 3 An example variable gadget

Consider the three uncertainty pairs with white and gray points in Fig. 2. If all three of the white points are chosen, then it is easy to see that, while points can be chosen so that the connectivity subgraph arising from pairs in the clause gadget can be made connected, no such subgraph can be connected to the rest of G_1 for any choice of points in the remaining uncertainty pairs. If a gray point is chosen from a white-gray pair, then the dashed edges shown incident to the pair do not belong to G_1.

Each of the three white-gray uncertainty pairs is connected to a variable gadget by a sequence of vertical uncertainty pairs. The choice of a gray point, shown in the schema as an open gate, is intended to mean that the literal (a variable or its negation) connecting to this open gate contributes a `True` to the clause. Note that the clause gadget never connects two variable gadgets.

Next, we outline how variable gadgets transmit truth values and how consistency of truth assignments is assured.

2.1.3 The Variable Gadget

An example of the variable gadget is shown in Fig. 3. Let the uncertainty pair at the extreme left of the variable gadget be the *reference pair* for this variable. We adopt the interpretation that the choice of black point in the reference pair means a setting of `True` to the variable and the choice of gray point means a setting of `False` to the variable.

In order for the constructs of type *I*, *II*, and *III* to function as described above, in the Overview of the gadgets, we require that the following two properties hold: (1) for all variable gadgets, G_1 is connected only if the subgraph of G_1 restricted to points chosen in a given variable gadget is connected; and, (2) the subgraph of G_1 restricted to points chosen in a variable gadget can be connected if and only if the truth values in type *I* and *II* constructs are consistent with the reference pair (as these are the parts of the gadget that propagate truth values to clause gadgets). Together, these properties ensure that for G_1 to be connected, the points chosen from variable gadgets must be internally connected and, in turn, each variable gadget must propagate consistent truth

values to the clause gadgets that are connected to it via connector gadgets. We prove these two properties now.

Recall that the clause gadget connects to the literals that are satisfied (only) by opening gates, and that no two open gates are connected through the clause gadget. As such, there cannot be a path in G_1 joining points in two (possibly indistinct) variable gadgets through points in a clause gadget. In addition, the edges E_1 join the variable vertices (horizontal lines) in $H(\Phi)$ by a path, not a cycle. Therefore, G_1 can be connected only if the subgraph of G_1 corresponding to points chosen from any variable gadget is connected. This shows the first property.

The choice of any black point in a construct of type *I* or *II* forces the choice of black points in this as well as in all the other constructs of type *I* and *II* inside a variable gadget. The same is true for gray points. In Fig. 3, if the gray point is chosen in the reference pair, gray arrows show the implications that force the choice of gray points in all the type *I* and *II* constructs. The function of the type *III* construct in the variable gadget is to allow this propagation. Thus, the second property holds; the variable gadget can be internally connected if and only if the truth value of the reference pair agrees with the truth value in type *I* and type *II* constructs.

We describe how the type *I* and type *II* constructs are used to connect to clause gadgets above and below the variable gadget. Suppose that the variable associated with this gadget is x. If the literal x appears in a clause gadget embedded above the variable gadget, then the connection from the corresponding clause gadget to this variable gadget (to be described in the next section) is made to the top of a construct of type *I*. In order to connect the variable gadget to the clause gadget, a black point has to be chosen in the reference pair. If the literal $\overline{x}$ appears in a clause above the variable gadget, then the connection is made to the top of a construct of type *II* and a gray point is chosen in a reference pair. Similarly, if the literal x appears in a clause embedded below the variable gadget, the connection from the clause gadget is made to the bottom of a type *II* construct and the black point is chosen in the reference pair. If the literal $\overline{x}$ appears in a clause embedded below the variable gadget, the connection is made to the bottom of a type *I* construct and the gray point is chosen in the reference pair.

We replace horizontal line segments corresponding to variables in the embedding of the Planar 3-SAT instance by variable gadgets. Note that the width of type *I* and *II* constructs can be adjusted by adding horizontally arranged uncertainty pairs. The number of occurrences of constructs of type *I*, *II* and *III* depends on the number of clauses containing this variable.

2.1.4 Linking the Gadgets

We now explain how to represent the edges of the planar graph $H(\Phi)$, corresponding to an instance Φ of Planar 3-SAT. In the embedding we are considering, edges are represented by vertical line segments. They represent two kinds of connections: (1) between a pair of variables, and (2) between a clause and a variable in that clause. Figure 4 shows vertical constructs of uncertainty pairs that (right) connect pairs of variable gadgets and (left) clause and variable gadgets. We observe the following properties of the two connectors: In a clause-variable connector, the choice of black

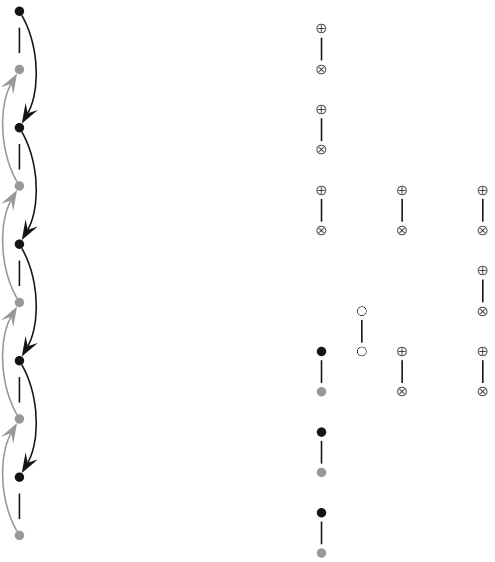

Fig. 4 *Left* A connector between clause and variable gadgets. *Right* A connector between variable gadgets

point in a clause gadget above a variable gadget implies the choice of black point in the variable gadget. The choice of gray point in the clause gadget below a variable gadget forces the choice of gray point in the variable gadget. In the variable-variable connector, for any choice of points in the two vertically extreme uncertainty pairs, there is a path using points in the shown uncertainty pairs that connects the two extreme uncertainty pairs. The white uncertainty pair allows this connection. The n variable gadgets are connected using $n-1$ variable-variable connectors that replace E_1 in $H(\Phi)$.

Note that the parities of the (integer) heights of the tops of type I and type II constructs in the variable gadget differ (see Fig. 3); we must ensure that clause gadgets have the flexibility to accommodate this. Indeed, this is the case; consider again Fig. 2. If necessary, gate-constructs can be shifted vertically by one unit to allow such connections. For example, the gate on the top of the gadget in Fig. 2 can be shifted by moving all the uncertainty pairs in the gray box up by one unit. This change preserves the properties of the clause gadget given earlier.

2.1.5 Correctness of the Reduction

In a line segment embedding of the planar graph $H(\Phi)$ corresponding to a Planar 3-SAT instance Φ, nodes (clauses and variables) are horizontal line segments and edges are vertical line segments. We have presented clause and variable gadgets to replace horizontal line segments and connectors to replace vertical line segments. We argue that the connectivity graph G_1 is connected for some choice of points in these uncertainty pairs if and only if the Planar 3-SAT instance Φ is satisfiable.

If the Planar 3-SAT instance Φ is satisfiable, let us consider the assignment of truth values to the variables of the instance. When a variable is set to `True`, we choose the black point in the reference pair of the corresponding variable gadget. When it is set to

`False`, we choose the gray point. Let P be the set of points chosen and consider the graph G_1 of P. In G_1, points selected in variable gadgets are all internally connected and connected to each other via the variable-variable connectors. As all the clauses are satisfied by the truth assignment, each clause is connected to one or more variable gadgets through edge-disjoint paths. Therefore the graph G_1 is connected.

Suppose that there exists a choice of points in U such that its corresponding connectivity graph G_1 is connected. In G_1, points in each clause gadget are connected to points in one or more variable gadgets through edge-disjoint paths. Therefore, points in two different variable gadgets can never be connected via a path that includes points chosen from a clause gadget. This implies that if G_1 is connected, then all the internally connected variable gadgets are connected to each other via the path in $H(\Phi)$ that is replaced by the $n-1$ variable-variable connectors. A truth value is assigned to each variable solely depending on whether the black or gray point is chosen inside the variable gadget. Since every clause gadget is connected to at least one variable gadget, the truth assignment satisfies every clause. Therefore, this truth assignment satisfies the Planar 3-SAT instance Φ.

This proves Theorem 1.

2.2 Inapproximability for Point Pairs

We observe that for uncertainty regions that are pairs of points, there is no approximation algorithm, polynomial in the size of the input, with an approximation ratio less than $\sqrt{5}/2$, unless $P = NP$. Indeed, we have provided problem instances where the uncertainty regions are vertically aligned pairs of points separated by a distance of one unit such that a Bottleneck Spanning Tree of maximum edge length 2 ($\alpha = 1$) can be found if and only if $P = NP$. But two points on the integer grid, if further apart than distance 2, must be at least distance $\sqrt{5}$ from one another. Hence if we had a polynomial time approximation to the solution with a ratio less than $\sqrt{5}/2$ then we could use the approximation to find a Bottleneck Spanning Tree with maximum edge length not greater than 2, a contradiction (unless $P = NP$).

Theorem 2 *There is no approximation algorithm, polynomial in the size of the input, that solves BCU for point pairs with approximation ratio less than $\sqrt{5}/2$, unless $P = NP$.*

2.3 BCU When the Uncertainty Regions Are Line Segments

We can also prove that the BCU problem is NP-hard for vertical unit segments on an integer grid. To show this, we use the same argument as in the proof of Theorem 1 except that point pairs, unit distance apart, are now replaced by vertical line segments of unit length.

Theorem 3 *It is NP-hard to find an exact solution to the BCU problem for the case in which the regions of uncertainty are vertical unit edges.*

Hardness of approximation, such as for point pairs (Theorem 2), however, does not hold for line segments, because we can use edges of length arbitrarily close to 2.

2.4 BCU When the Uncertainty Regions Are Unit Squares

We prove an NP-hardness result for this problem using a reduction from Planar 3-SAT. Our reduction uses techniques similar to the previous reduction.

Theorem 4 *It is NP-hard to find an exact solution to the BCU problem for the case where the regions of uncertainty are unit squares.*

Proof We use a reduction from the formulation and embedding $H(\Phi)$ of a Planar 3-SAT instance Φ that is given in the proof of Theorem 1.

We need the following terminology: A point p is *l-connected* to a point q if the (Euclidean) distance from p to q does not exceed l. A set S of points is l-connected if the maximum edge length of the minimum spanning tree of S does not exceed l.

We now describe the variable and the clause gadgets as well as the connectors we use to link the variables and clauses. □

2.4.1 The Variable Gadget

For each variable, we create a gadget similar to the one shown in Fig. 5. The variable gadget, as shown in the figure, can be 5-connected in two ways: by choosing the gray points in each square, or by choosing the black points in each square. We call the bold square in the figure the *reference* square. A point in the reference square can only be 5-connected to a point in at most one square among the other squares in the variable gadget. If the black point is chosen to make this connection in the reference square, then we say that the variable associated with this gadget is `True`; if the gray

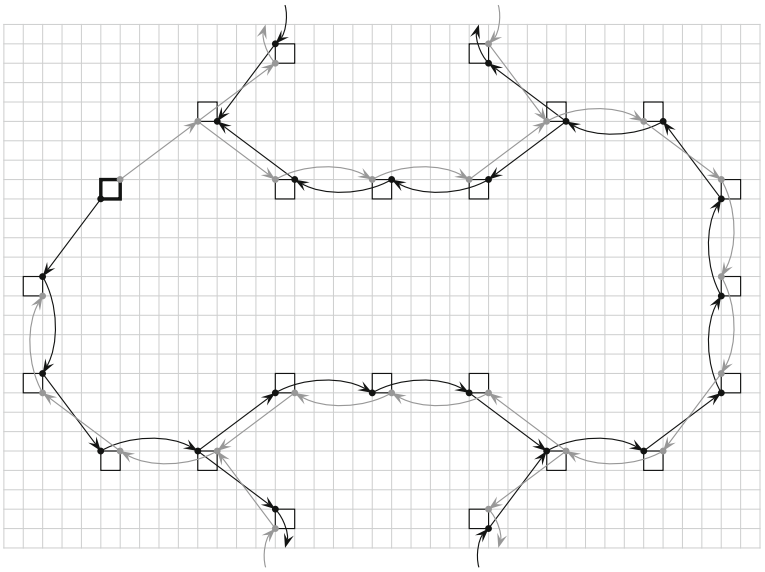

Fig. 5 An example variable gadget

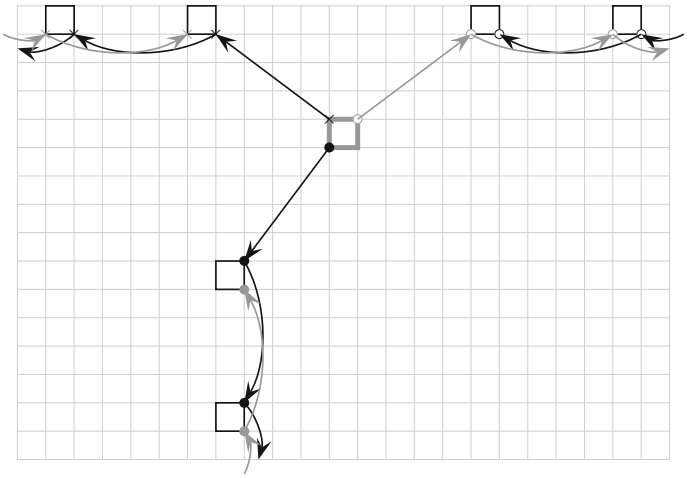

Fig. 6 An example clause gadget. The *gray arrow* connecting the core *square* to the *top right* arm represents a connection to a negated variable

point is chosen, we say that the variable is `False`. Furthermore, the subgraph of G_1 corresponding to points in a variable gadget can be connected if and only if the points (black or gray) chosen for all of its uncertainty regions are consistent.

Connections to clause gadgets (which replace clause vertices embedded as horizontal lines) that contain this variable or its negation are made via constructs similar to the four extreme top and bottom squares shown in the example variable gadget of Fig. 5. Of these, the top left and bottom right connect to clauses containing this variable and the top right and bottom left connect to clauses containing the negation of this variable. The width of the variable gadget can easily be increased and more such constructs can be added to allow additional connections (which replace edges embedded as vertical lines of $H(\Phi)$) to clause gadgets above or below.

2.4.2 The Clause Gadget

For each clause, we create a gadget such as the one shown in Fig. 6. We call the bold square in the clause gadget the *core* square. For each of the 3 literals (a variable or its negation), there is a sequence of squares in the gadget, called an *arm*.

Observe that the core square can be 5-connected to only a single arm, and once this arm is selected, the choice of points in its squares is *fixed* in order for the squares to be 5-connected. The choice of points within the other arms' squares is *free* since they do not need to connect to the core point. These squares can be made connected to each other.

2.4.3 Linking the Gadgets

Here we explain how variable gadgets can be linked to clause gadgets and how variable gadgets can be linked to each other.

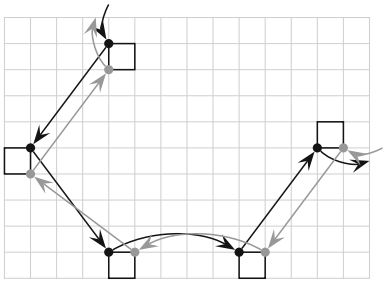

Fig. 7 An example corner gadget

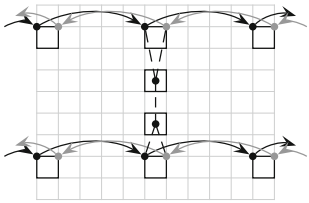

Fig. 8 An example of a loose connection

Clause-variable connectors represent edges of $H(\Phi)$ and, as such, they propagate truth values vertically. In order to attach them to the horizontal arms of the clause gadget, shown in Fig. 6, we use a corner gadget, shown in Fig. 7, to change the alignment of the gray and black points in the square regions from horizontal to vertical. The corner and clause gadgets may be reflected as necessary in order to attach to three clause-variable connectors above and/or below the clause gadget.

To complete the connection between vertex and clause gadgets, it remains to explain how to connect pairs of square uncertainty regions, one in the variable gadget and the other in an extension of one of the arms of the clause gadgets. We would like to propagate truth assignments (choice of gray or black point in the variable gadget) consistently along such connectors. When the distance between the points to be joined in the two squares is a multiple of 5 and these points are vertically aligned, the connection can be made in the way analogous to that shown in Fig. 4(left). Otherwise the construction of such connectors can be easily accomplished but is quite tedious to describe. We leave this construction to the reader.

To connect variable gadgets to each other (in place of the edge subset E_1 of the Planar 3-SAT embedding), we add $n-1$ *loose* connections between the n variable gadgets by using a sequence of squares that allow the variable gadgets to be 5-connected to each other, regardless of the choice of point in the squares of the gadget. Figure 8 shows such a loose connection between two horizontal sequences of squares. The variable gadgets may need to be expanded by adding horizontal sequences of squares in order to allow these connections.

2.4.4 Correctness of the Reduction

We now argue that there exists a choice of point in each of the square uncertainty regions such that the connectivity graph for $\alpha = 5/2$ is connected if and only if the corresponding Planar 3-SAT instance is satisfiable.

Suppose that there is a satisfying assignment to the Planar 3-SAT instance. Then, in each variable gadget we choose the black corner of the reference square when that variable is `True` and the gray corner when that variable is `False` in the satisfying assignment. Our construction shares the property with the one in Sect. 2.1, for point-pairs, that points chosen in a variable gadget must be internally connected for G_1 to be connected. As such, we must choose points in the other squares of the variable gadgets to be the same colour as that of the reference square. The choice of gray or black point is then propagated to every clause gadget satisfied by this assignment via the connectors. Note that the truth value is correctly inverted by the variable gadget when connecting a negated variable to a clause that includes this literal. Since all the clauses are satisfied, the core square in each clause gadget is connected to a variable that satisfies the clause. The arms that do not connect to the core square are connected to their respective variable gadgets. At this point we have created n trees, one for each variable. Points chosen in $n-1$ connectors between variable gadgets make $G_{5/2}$ connected.

Let P be any choice of points in the square regions for which the corresponding graph $G_{5/2}$ is connected. Since $G_{5/2}$ is connected, points in each clause gadget are connected to points in exactly one variable gadget. Because variable gadgets can never be connected to each other via a clause gadget, each variable gadget must be internally connected. The choice of gray or black point in the reference square of each variable gadget assigns the satisfying truth assignment to the variable associated with that variable gadget in the Planar 3-SAT instance.

By reduction from Planar 3-SAT, BCU for non-overlapping unit square uncertainty regions whose corners can be given integer coordinates is NP-hard.

Once again, hardness of approximation does not hold because we can use edges of length arbitrarily close to 5.

3 An Exact Algorithm for Solving BCU for n Fixed Points and k Segments

We present an exact algorithm that solves BCU—for a given precision δ—when the input consist of n fixed points, and the uncertainty regions are k line segments, possibly of varying length, in general position. That is, no two of the line segments are parallel. Our algorithm determines, in a time that is polynomial in n for any fixed k and constant precision δ, a set of point positions on the line segments that permits a spanning tree whose longest edge is of minimum length amongst all spanning trees that connect exactly one point from each segment as well as all fixed points.

Rather than give a practical algorithm, the aim of this section is to show that BCU is indeed computable and provide an upper bound on the running time. Nevertheless, we introduce two tools below, minimum solution trees and critical paths, in order to discretize and prune the search space (over all possible labelled spanning trees on $n+k$ vertices) considerably.

We highlight in the Key ideas how the general position assumption simplifies the proof of correctness of the algorithm, but the extension to inputs that include parallel

line segments is not difficult to conceive. We omit the details of such an extension for the sake of clarity.

3.1 Key Ideas

In our search for an optimum solution we focus on determining optimum solutions that satisfy slightly stronger additional properties: we seek a selection of point locations on the line segments which supports a *minimum solution tree*. A minimum solution tree is a spanning tree for segment locations and fixed points that does not just have a shortest longest edge, but also has a shortest second longest edge amongst all such solutions, and so forth. Looking for minimum solution trees, instead of optimum solutions, can decrease the search space considerably by virtue of the fact that for certain problem instances an infinite number of optimum solutions exist due to freedom of point selection on the line segments.

Critical paths give support when determining minimum solution trees. These are paths in spanning trees where all edges are of equal length and all inner path points are located on line segments. Further, moving the location of any of the points on segments will lengthen at least one of the edges while shortening another.

Our algorithm determines point locations on all segments that support a minimum solution tree with longest tree edge of length 2α as follows. We enumerate candidates for critical paths, from longest to shortest in terms of path length. Each candidate path is tested to see whether or not it supports a critical path. If successful, and if the edge length of this critical path is no longer than the longest edge of the solution tree for the best point set found so far, the line segments of that critical path are replaced by their corresponding point locations. The general position assumption provides that these point locations are unique, and simplifies our method of computing them. We then recurse on the updated input. Once all points on segments are determined, a greedy algorithm to determine the corresponding minimum solution tree can be applied.

3.2 Minimum Solution Trees

To describe minimum solution trees formally, we begin by defining a way that allows us to compare different spanning trees that correspond to optimum solutions. We partition into equivalence classes the set of all spanning trees taken over all fixed points and all point choices on the k segments, and we define a linear ordering on the equivalence classes such that a minimum solution tree is a smallest spanning tree w.r.t. the linear ordering.

For any two selections of points on the k segments, and for any two of their corresponding spanning trees, let $\mathcal{L}$ and $\mathcal{L}'$ be ordered lists of lengths of all edges in the two trees, sorted from longest to shortest. That is, $\mathcal{L} = (l_1, l_2, \ldots, l_{n+k-1})$ and $\mathcal{L}' = (l'_1, l'_2, \ldots, l'_{n+k-1})$, with $l_i \geq l_{i+1}$ and $l'_i \geq l'_{i+1}$ for all I. We say that $\mathcal{L}$ *is preferred over* $\mathcal{L}'$ if for a certain I, $l_i < l'_i$, and $l_j = l'_j$ for all $j < i$. If $\mathcal{T}$ and $\mathcal{T}'$ are spanning trees with edge lists $\mathcal{L}$ and $\mathcal{L}'$, respectively, we also say $\mathcal{T}$ is preferred over

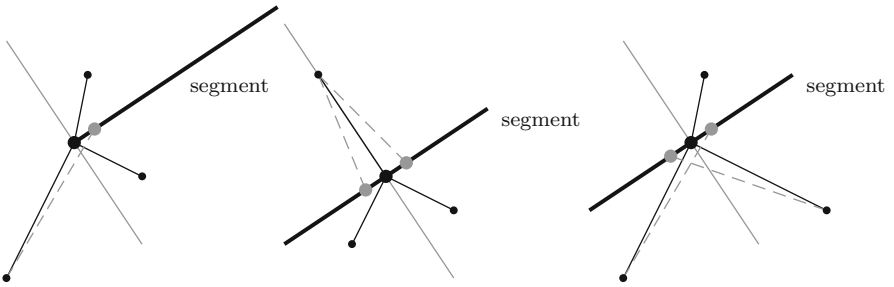

Fig. 9 Points (in *black*) of Type 1, 2, and 3 respectively, with the segment shown in *bold*, incident edges in *black*, and perpendicular (to the segment) shown in *gray*. In each case, moving the point along the segment results in a longer locally longest incident edge

$\mathcal{T}'$ if $\mathcal{L}$ is preferred over $\mathcal{L}'$. Note that this defines a linear ordering on lists in general, and not only those derived from spanning tree edge-lengths.

Our algorithm seeks to choose points on segments that result in a spanning tree such that no other spanning tree is preferred over it. We call such a tree a *minimum solution tree* $\mathcal{T}$.[1] We call a choice of points on segments that results in a minimum solution tree $\mathcal{T}$ a *best point set* for $\mathcal{T}$.

In a minimum solution tree $\mathcal{T}$, not only are longest edges as short as possible, but also the number of longest edges is minimum. In other words, a tree with a smallest number of shortest longest edges is preferred over the ones with more edges of the same length. Further, for all I the i^{th} longest edge is as small as possible, and the number of edges of that length is minimum.

The above conditions imply convenient properties on the best point set w.r.t. a minimum solution tree. Note that, for any point p on a segment in a best point set, it is impossible to *improve* the solution by slightly moving p on its segment; in fact, any perturbation of a point must lengthen at least one of the edges that is longest among all edges incident to p. We now list the possibilities for a point p on a segment in a best point set (see Fig. 9) in distinguishing three different types. Given a point p on a segment, we call an edge e that belongs to a minimum solution tree $\mathcal{T}$ incident to p *locally longest* if no other tree edge of $\mathcal{T}$ incident to p is longer than e. Then, in a minimum solution tree the possibilities for a point located on a segment w.r.t. its locally longest edges are as follows.

Type 1 Point p lies at an extremity of the segment. Then, one of the locally longest edge, e, incident to p lies on the half plane that is delimited by a line perpendicular to the segment and does not contain the segment. We observe that moving p would lengthen e.

Type 2 Point p is on the relative interior of the segment and e, one of the locally longest edges incident to p, is perpendicular to the segment. We observe that moving p in any direction would lengthen e.

[1] We remark that for lists $\mathcal{L}$ and $\mathcal{L}'$ for two different spanning trees with two different sets of points on the segments, it is possible that $\mathcal{L} = \mathcal{L}'$, so that optimum solutions that permit minimum solutions trees are in general not unique.

Type 3 Point p is on the relative interior of the segment but not of Type 2. Then there are two locally longest edges incident to p laying in different half-planes delimited by a line perpendicular to the segment passing through p. We observe that moving p in any direction would increase the length of one of these two edges.

Notably, if we know for any point p on a segment that it is of Type 1 or 2 and what its locally longest incident edge e is, then we can deduce p's position on the segment without any knowledge of other incident edges of p, just by minimizing the length of e. Similarly, if we know that for any point p on a segment that it is of Type 3 and what its pair of locally longest incident edges e, f is, then we can deduce p's position on the segment without any knowledge of other incident edges of p, just by minimizing the length of e and f.

3.3 Critical Paths

Let $(E_1, \ldots, E_m)$ denote a sequence of fixed points and segments, where E_1 and E_m are fixed points or segments, and $E_2, \ldots, E_{m-1}$ are segments. A *critical path supported by* $(E_1, \ldots, E_m)$ consists of points $p_1, \ldots, p_m$, where each p_i is located at a selected position on segment E_i. The p_is are connected by edges e_i such that (1) the edges are all of identical length, and (2) no different selection of point locations on these segments results in a sequence where no edge is longer but some edge is strictly shorter. We may specify that the edges are of length λ by writing λ-*critical path*.

Critical paths are useful in constructing minimum solution trees. Our algorithm makes use of the fact that it is possible to reduce the construction of a minimum solution tree to computing a set of critical paths. We will show below (1) that a sequence $(E_1, \ldots, E_m)$ supports at most one critical path (under the assumption of general position) and (2) how to compute a critical path supported by $(E_1, \ldots, E_m)$—in case of existence—for a given precision δ.

We introduce terminology that will aid us for both purposes. Given a sequence $(E_1, \ldots, E_m)$ and a positive number λ, let $U_i(\lambda)$ be the area around E_i that can be reached from E_1 via $E_2, \ldots, E_{i-1}$ by edges of length at most λ. More exactly, let

- $U_1(\lambda)$ be the set of points in the plane reachable from E_1 by an edge of length at most λ, and
- $U_i(\lambda)$, $i > 1$, be the set of points on the plane reachable from $U_{i-1}(\lambda) \cap E_i$ by an edge of length at most λ.

Let $S_i(\lambda)$ be the set of points on the plane reachable from E_i by an edge of length exactly λ, such that $p \in S_i(\lambda)$ implies that there is a λ-critical path $p_1, p_2, \ldots, p_i, p$ supported by $(E_1, E_2, \ldots, E_i, \{p\})$, with $p_i \in E_i$ (note that p need not be in one of the uncertainty regions). In particular, $S_i(\lambda)$ is contained, for $i > 1$, in the set of points on the plane reachable from $S_{i-1}(\lambda) \cap E_i$ by an edge of length exactly λ. We characterise the set $S_i(\lambda)$ more directly in Lemma 3.

We study the properties of $U_i(\lambda)$ and $S_i(\lambda)$. By definition, E_1 consists of either a single point or a segment (Fig. 10). Further, $U_1(\lambda)$ consists either of a circle of radius λ (Fig. 10) or the Minkowski sum of a circle of radius λ and a segment, and therefore also $U_1(\lambda) \cap E_2$—if not empty—consists of either a point or a subsegment of E_2.

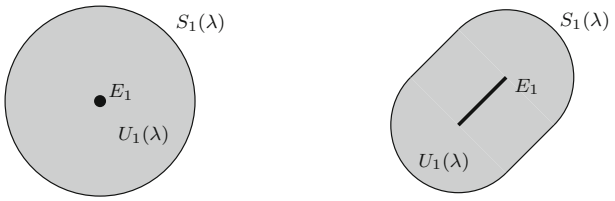

Fig. 10 Examples of $U_1(\lambda)$ and $S_1(\lambda)$ for the cases that E_1 is a fixed point (*left*) and a segment (*right*)

In general we can deduce inductively the following lemma.

Lemma 1 *For $1 < i \leq m$, if $U_{i-1}(\lambda) \cap E_i \neq \varnothing$ then $U_{i-1}(\lambda) \cap E_i$ consists of either a single point of E_i or a subsegment of E_i.*

Using the definitions of $U_i(\lambda)$ and $S_i(\lambda)$, we have the following observation.

Lemma 2 *For all $i \geq 1$, $S_i(\lambda) \subseteq$ boundary of $U_i(\lambda)$.*

Proof For $i = 1$, it follows easily from the definition that $S_1(\lambda)$ is contained in the boundary of $U_1(\lambda)$. Let B be an open ball contained in $U_i(\lambda)$, with $p \in B$; we will show that p cannot be in $S_i(\lambda)$, and hence $S_i(\lambda)$ can only contain boundary points. For any $p' \in B$, since $B \subset U_i(\lambda)$, there is a path on points $(p_1, p_2, \ldots, p_i, p')$, with $p_j \in E_j$, for $1 \leq j \leq i$, with no edge longer than λ. On the other hand, since B is an open set, there exists a $p' \in B$ such that $\|p_i p\| < \|p_i p'\|$, for all choices of p_i in E_i, and hence there exists a path on points $(p_1, p_2, \ldots, p_i, p')$ where $\|p_i p\| < \|p_i p'\| \leq \lambda$. Thus we have points $(p_1, p_2, \ldots, p_i, p)$ with $\|p_j p_{j+1}\| \leq \lambda$, for $1 \leq j < i$, and $\|p_i p\| < \lambda$ implying that $(E_1, E_2, \ldots, E_i, \{p\})$ does not support a λ-critical path, and therefore $p \notin S_i(\lambda)$. □

We can conclude from the above lemma that, if $U_{i-1}(\lambda) \cap E_i = \varnothing$ then $S_{i-1}(\lambda) \cap E_i = \varnothing$. Further, if $U_{i-1}(\lambda) \cap E_i$ consists of a single point p, then either $S_{i-1}(\lambda) \cap E_i = \varnothing$ or $S_{i-1}(\lambda) \cap E_i = \{p\}$. If $U_{i-1}(\lambda) \cap E_i$ is a subsegment of E_i, then $S_{i-1}(\lambda) \cap E_i$ can be empty, consist of one or both extremities of the subsegment. In fact, we can prove the following lemma that describes $S_i(\lambda)$ more directly.

Lemma 3 1. *The set $S_1(\lambda)$ is the boundary of $U_1(\lambda)$.*
2. *For all $i > 1$, the set $S_i(\lambda)$ is the intersection of the boundary of $U_i(\lambda)$ with the Minkowski sum of a circle of radius λ and $S_{i-1}(\lambda) \cap E_i$.*

Proof 1. This fact follows from the definition of $S_1(\lambda)$ and $U_1(\lambda)$.

2. $\subseteq$: We make two observations. (1) By definition, $S_i(\lambda) \subseteq$ {points at distance exactly λ from $S_{i-1}(\lambda) \cap E_i$}. This set, in turn, is contained in the Minkowski sum of a circle of radius λ and $S_{i-1}(\lambda) \cap E_i$. (2) From Lemma 2, we know that $S_i(\lambda) \subseteq$ boundary of $U_i(\lambda)$. Combining (1) and (2) gives us the result.

   $\supseteq$: Proof by Induction. Let p be any point in the intersection of the boundary of $U_i(\lambda)$ with the Minkowski sum of a circle of radius λ and $S_{i-1}(\lambda) \cap E_i$. Then

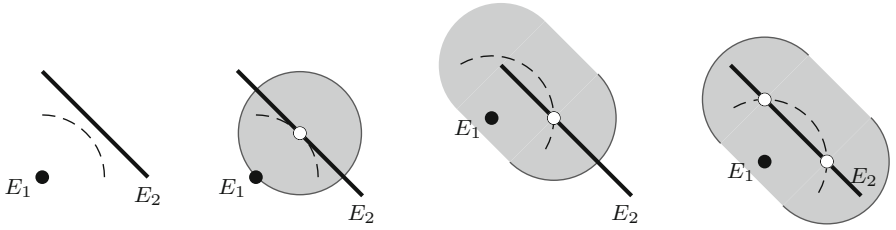

Fig. 11 Shapes of $S_2(\lambda)$ of Type a, b, c, and d, respectively. $S_1(\lambda)$ is indicated with a *dashed line*, $S_1(\lambda) \cap E_2$ with *open dots*, and $S_2(\lambda)$ with *dark gray curves*

by definition there exists $q \in S_{i-1}(\lambda) \cap E_i$ and points $q_1, q_2, \ldots, q_{i-1}$ such that $\|qp\| = \lambda$ and the path $Q = (q_1, q_2, \ldots, q_{i-1}, q)$ is a λ-critical path supported by $(E_1, E_2, \ldots, E_{i-1}, \{q\})$.

Suppose that $p \notin S_i(\lambda)$. This implies that $(E_1, E_2, \ldots, E_{i-1}, E_i, \{p\})$ does not support a λ-critical path. In particular no edge of $(p_1, p_2, \ldots, p_{i-1}, p_i, p)$ is longer than λ, but some edge is strictly shorter, for some choice of points $p_j \in E_j$, for $1 \leq j \leq i$; in particular, we have $\|p_i p\| \leq \lambda$. On the other hand, we also have $\|p_i p\| \geq \lambda$, since p is in the boundary of $U_i(\lambda)$, so by assumption, $\|p_i p\| = \lambda$. If one of the edges in $(p_1, p_2, \ldots, p_{i-1}, p_i)$ is shorter than λ, then $p_i \in E_i(\lambda) \backslash S_{i-1}(\lambda)$, by definition of $S_{i-1}(\lambda)$. On the other hand, there is exactly one point in E_i that is distance λ from p, and thus the existence of p and $\|p_i p\| = \lambda$ imply that $p_i = q$, and therefore $p_i \in S_{i-1} \cap E_i$. This is a contradiction, and therefore $p \in S_i(\lambda)$, as required. □

We deduce that the following cases for $S_i(\lambda)$ are possible (see Fig. 11, depicting possibilities for $S_2(\lambda)$ for the case that E_1 is a fixed point).

Type a. $S_{i-1}(\lambda) \cap E_i = \varnothing$ and therefore $S_i(\lambda) = \varnothing$.

Type b. $U_{i-1}(\lambda) \cap E_i$ consists of a single point p and $S_{i-1}(\lambda) \cap E_i = \{p\}$. Then $S_i(\lambda)$ is a circle of radius λ centered around p.

Type c. $U_{i-1}(\lambda) \cap E_i$ is a subsegment of E_i and $S_{i-1}(\lambda) \cap E_i$ is a single extremity of the subsegment. In this case, $U_i(\lambda)$ is the Minkowski sum of the subsegment and a ball of radius λ, and $S_i(\lambda)$ is the half circle of radius λ centered on $S_{i-1}(\lambda) \cap E_i$ on the boundary of $U_i(\lambda)$.

Type d. $U_{i-1}(\lambda) \cap E_i$ is a subsegment of E_i and $S_{i-1}(\lambda) \cap E_i$ consists of both extremities of the subsegment. In this case, $U_i(\lambda)$ is the Minkowski sum of the subsegment and a ball of radius λ, and $S_i(\lambda)$ consists of both half circles of radius λ each centered on a point of $S_{i-1}(\lambda) \cap E_i$ on the boundary of $U_i(\lambda)$.

For a given $(E_1, \ldots, E_m)$ and length λ it is therefore possible to compute successively the $U_i(\lambda)$'s and $S_i(\lambda)$'s. With this knowledge in hand, we are now ready to describe the computation of critical paths. The following lemma is crucial towards this goal.

Lemma 4 *A critical path exists for a sequence $(E_1, \ldots, E_m)$ if and only if there exists a λ^* such that $U_{m-1}(\lambda^*) \cap E_m = S_{m-1}(\lambda^*) \cap E_m \neq \varnothing$. Furthermore, such a λ^* is the smallest λ such that $U_{m-1}(\lambda) \cap E_m \neq \varnothing$.*

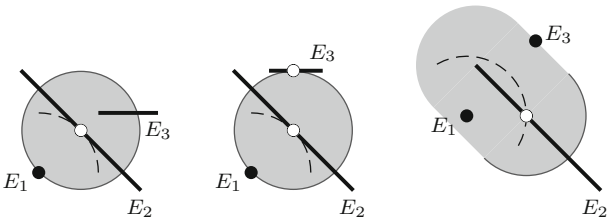

Fig. 12 Sequence (E_1, E_2, E_3) with outcomes of Type α, β, and γ respectively for the case that $U_2(\lambda) \cap E_3 \neq \varnothing$. $S_1(\lambda)$ is depicted by *dashed lines*, $S_1(\lambda) \cap E_2$ and $S_2(\lambda) \cap E_3$ by *open dots*, and $S_2(\lambda)$ in *dark gray*

Proof Figure 12 illustrates the proof idea by describing types of possible outcomes for any λ for the case of a sequence (E_1, E_2, E_3) with E_1 a fixed point, for which $U_2(\lambda) \cap E_3$ is not empty. In general for any sequence $(E_1, \ldots, E_m)$ the following shapes of $S_m(\lambda)$ are possible. Recall our general position assumption that no two lines are parallel. Using Lemma 1, if $U_{m-1}(\lambda) \cap E_m \neq \varnothing$, either $U_{m-1}(\lambda) \cap E_m$ is a subsegment of E_m (Type α below) or it is a single point of E_m (Type β and γ below).

Type α. If E_m intersects the interior of $U_{m-1}(\lambda)$ then there is no λ-critical path. A path with edges of length exactly λ will not be critical as it can be shortened by moving the point location on E_m.

Type β. If $U_{m-1}(\lambda) \cap E_m = S_{m-1}(\lambda) \cap E_m = \{p\}$ then there is a λ-critical path as moving the point location on E_m only increases the edge length.

Type γ. If $U_{m-1}(\lambda) \cap E_m$ is a single point and $S_{m-1}(\lambda) \cap E_m = \varnothing$, there is no λ-critical path. In this case, moving the point location on E_{m-1} gives a path with shorter edge length.

To summarize, we observe that a critical path only exists for outcomes of Type β, and that this is the only outcome for which the condition given in the lemma is satisfied. Moreover, the λ^* in outcomes of Type β is the smallest value of λ for which $U_{m-1}(\lambda) \cap E_m \neq \varnothing$ since $U_{m-1}(\lambda) \cap E_m$ is a single point and decreasing the value of λ any further will make it empty. □

Given $(E_1, \ldots, E_m)$, the following algorithm determines whether $(E_1, \ldots, E_m)$ supports a critical path or not. If it does, the algorithm outputs the length of the edges in the critical path with a precision δ'. Note that the need for a precision parameter arises only due to the algebraic nature of the problem since our computation involves binary search over real values. δ' is chosen depending on the precision bound, δ, specified by the user and the input instance so that the loss in precision over all the steps of the algorithm is below the bound δ. More precisely, we will choose $\delta' = \delta/k$. We describe the reason for this choice after describing our algorithm.

1. Initialize $\lambda_{\min}$ and $\lambda_{\max}$ to be the minimum and the maximum distance between any adjacent pair (E_i, E_{i+1}) in the input sequence. Initialize λ to be $(\lambda_{\min} + \lambda_{\max})/2$.
2. While $(\lambda_{\max} - \lambda_{\min} > \delta')$ do
   - Compute $U_{m-1}(\lambda)$.

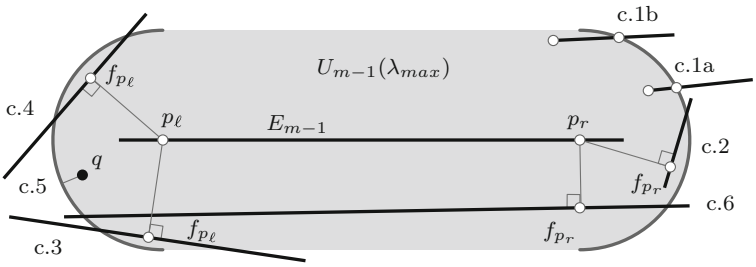

Fig. 13 Cases 1–6 where we test for for λ^* of Type-β (up to perturbation δ'). *Each line* segment labelled $c.i$, for $1 \leq i \leq 6$, represents a possible instance of E_m

- If $U_{m-1}(\lambda) \cap E_m = \varnothing$, then set $\lambda_{\min}$ to λ, set λ to $(\lambda_{\min} + \lambda_{\max})/2$ and go to (2).
- If $U_{m-1}(\lambda) \cap E_m \neq \varnothing$, then set $\lambda_{\max}$ to λ, set λ to $(\lambda_{\min} + \lambda_{\max})/2$ and go to (2).

3. Check if $U_{m-1}(\lambda_{\max}) \cap E_m = S_{m-1}(\lambda_{\max}) \cap E_m \neq \varnothing$ up to perturbation δ' as explained below. If so, output $\lambda_{\max}$ as the edge-length of the critical path. If not, output that no critical path exists.

We now give more details about the last step of our algorithm when the shape of $S_{m-1}(\lambda_{\max})$ is of Type d and $m \geq 3$; that is, where $S_{m-2}(\lambda_{\max}) \cap E_{m-1}$, for $m \geq 3$, comprises exactly two distinct points, denoted p_ℓ and p_r. This procedure is similar to what is needed to handle other non-trivial shapes of $S_{m-1}(\lambda_{\max})$, i.e., Types b and c as well as $S_1(\lambda_{\max})$, when $m = 2$. We will denote $U_{m-1}(\lambda_{\max}) \cap E_m$ by L if it is a line segment and by q if it is a point. For a point $p \in \{p_\ell, p_r\}$ and a line segment L, let f_p denote the point on L closest to point p. The algorithm will detect if one of the six (mutually exclusive) cases described in Fig. 13 occurs, where we assume that in all cases the segment $p_\ell p_r$ does not intersect E_m. Procedures for this detection can be easily derived from high level descriptions[2] of the cases. In each case the algorithm performs a test, as follows:

Case 1a and 1b: Test if the length of line segment L is at most δ'.
Cases 2–4: Test if $\lambda_{\max} - d(p, f_p)$ is at most δ'. Here, p will be either p_ℓ or p_r depending on which semicircle is intersected by L.
Case 5: Test if the minimum distance from q to either semicircle is at most δ'.
Case 6: Test if $\lambda_{\max} - d(p, f_p)$ is at most δ'. Here, p will be either p_ℓ or p_r depending on which is closer to L (Note that by the general position assumption, one of them will be closer).

If one of the above cases occurs and its test returns true, there is a $\lambda^* \in [\lambda_{\min}, \lambda_{\max}]$ of Type β, as illustrated in Fig. 12, and hence a critical path exists. For the two other possible scenarios (Type α and Type γ) either none of cases 1-6 occur or the

[2] The descriptions are evident from Fig. 13, except perhaps the difference between c.1b and c.2. These are distinguished by the fact that f_{p_r} lies outside the semi-disc for c.1b, and inside the semi-disc for c.2, where p is p_ℓ or p_r, as appropriate.

corresponding test fails, and the algorithm concludes that the critical path does not exist. These cases are also illustrated in Fig. 12.

As a consequence of Lemma 4, we have the following corollary. It will help us convert a set of line segments, for which we have found a critical path, into a set of fixed points, resulting in an input with fewer line segments.

Corollary 1 *If* $(E_1, \ldots, E_m)$ *supports a critical path consisting of edges that are locally longest in a minimum solution tree, then there is a unique choice of point locations that defines the critical path. Furthermore, this choice of point locations is a part of the optimal solution.*

3.4 Description of the Algorithm

We now show how to compute a best point set by examining all possible critical paths.

Note that if we had an oracle giving us a sequence $(E_1, E_2, \ldots, E_m)$ that supports a critical path consisting of edges that are locally longest in the best point set, then we could determine the choice of points on $(E_1, E_2, \ldots, E_m)$ these elements in the best point set. We could then replace the segments in the sequence by fixed points and solve the rest of the problem separately. This eventually would allow us to replace all segments by fixed points, and then to solve the problem by finding an associated MBST. Lacking an oracle we determine these sequences *by complete enumeration* of *all possible sequences* $(E_1, E_2, \ldots, E_m)$ that contain at least one segment, where E_1 and E_m are fixed points or segments, and $E_2, E_3, \ldots, E_{m-1}$ are segments. There are $O(n^2 \cdot k! \cdot k)$ such sequences.[3] This enumeration accounts for most of the complexity of our recursive algorithm, in which we initialize b to ∞ at the top level call, and proceed as follows:

1. If the instance does not contain any segments, we compute an MST with a greedy algorithm, in polynomial time, and report the length b' of the bottleneck edge.
2. Else, for each sequence in the enumeration, if it supports an ℓ-critical path, for some ℓ, do the following:
   (a) If $\ell > b$, do nothing; that is, we prune these sequences in the enumeration as we find them.
   (b) Else, if $\ell \leq b$, we replace the segments of the sequence with the unique fixed points defined by the ℓ-critical path and recurse on the updated set of sequences and points, with b (in the recursive call only) initialized to ℓ. Let b' be the output of the recursive call.
   (c) If $b' < b$, set b to b'.
3. If no MST is found, report ∞, else report b.

We remark that it is crucial to compute critical paths with progressively decreasing edge lengths since positions of points on segments are determined by locally longest edges incident to them; this is enforced in recursive calls with the parameter b. Furthermore, updating b in step 2(c) allows us to prune the search in step 2(a) by discarding critical paths whose longest edge is too long. Finally, the algorithm terminates because

[3] Count the number of ordered I-subsets of k segments by $\sum_{i=1}^{k} k!/(k-i)!$, and multiply these by the $(n^2 + n + 1)$ to count ways that E_1 and E_m might be points.

we only enumerate sequences with at least one segment. We show by induction on k that the algorithm correctly outputs the length of the bottleneck edge of a minimum solution tree. The algorithm is correct when there are 0 segments. Suppose it is correct for up to $k-1$ segments. If we call the algorithm with k segments, for $k > 0$, then at least one of the sequences $(E_1, E_2, \ldots, E_m)$ supports an ℓ-critical path whose edges are not only locally longest in a minimum solution tree, but also globally longest among sequences of critical paths supported by at least one segment. Thus the sequence satisfies the antecedents of Corollary 1, and furthermore, all other critical paths in the minimum solution tree will be found in the recursive calls. By the inductive hypothesis, the recursive call returns the length of the bottleneck edge b' of a minimum solution tree on the reduced input, and by Corollary 1 this is also the length of a bottleneck edge on the unreduced input (we may have $b' > \ell$, however). We have shown that for k segments the output is at most the length of a bottleneck edge of a minimum solution tree on the input points and segments; on the other hand, the output value cannot be smaller than this, since the value is, in any case, derived from the bottleneck edge of an MST on a set of points chosen from the $n+k$ input regions.

With regards to precision, when we replace the line segments with fixed points, our choice is correct within δ' of the true value. Since our algorithm is recursive, this choice of points will in turn influence the choice of points in the next round of recursion. Since our algorithm is computing distances between points, the edge lengths of the critical path computed in level I of the recursion will be correct up to a precision of $i\delta$ (using the triangle inequality). Since the last level of recursion, level k, requires a precision of δ, we will fix $k\delta' = \delta$ in our analysis.

The enumeration in our algorithm described above is superexponential in the number k of segments, which is not surprising since we have shown the problem with no fixed points is NP-hard. For constant k the problem is, however, polynomial in the number n of points, as our running time analysis will show.

The (multiple recursive) enumeration results in a search tree of size $O((n^2 \cdot k! \cdot k)^k)$ with an $O(k)$ running time for each node in the search tree. Thus the total time complexity is $O((n^2 \cdot k! \cdot k)^k \cdot k)$.

Theorem 5 *The BCU problem for a set of n fixed points and k line segments can be solved in time $O((n^2 \cdot k! \cdot k)^k \cdot k)$, for any fixed precision δ.*

4 Constant-Factor and Additive Approximations

We begin by considering the Best-Case Connectivity with Uncertainty problem (BCU).

Lemma 5 *Given a set of uncertainty regions that are unit disks $D_1, \ldots, D_n$ with centers $p_1, \ldots, p_n$, let L be the largest edge of a minimum bottleneck spanning tree on $\{p_i : 1 \le i \le n\}$. Then choosing locations $\ell_i = p_i$ and $\alpha = L/2$ is at worst an OPT+1 approximation to the BCU Problem. In other words, if OPT denotes the smallest radius α for any choice of $\ell_i \in D_i$, then $L/2 \le OPT + 1$. This approximation can be computed in polynomial time.*

Proof Consider the best choice of the $\{\ell_i \in D_i : 1 \le i \le n\}$ and an associated MBST on these $\{\ell_i : 1 \le i \le n\}$. The edges of this MBST are each at most 2 shorter than

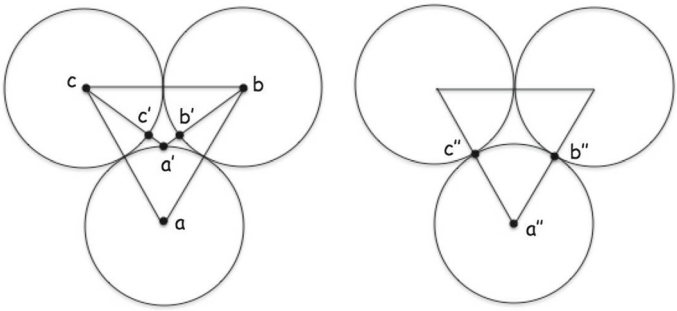

Fig. 14 The BCU problem for three (almost) tangent unit disks. OPT, as shown on the *left* is given by the choice of locations (a', b', c'), not quite on an equilateral triangle, with MBST cost $(\sqrt{1+(\sqrt{3}-1)^2}-1)/2 \approx .12$, while the "cinch-up" heuristic, on the *right*, chooses locations (a'', b'', c'') with cost 0.5

the corresponding edges of a spanning tree, S, on the corresponding $\{p_i : 1 \leq i \leq n\}$. Thus the maximum length of any edge in S is at most 2 greater than the maximum length edge in the MBST on $\{\ell_i : 1 \leq i \leq n\}$, and, similarly, the maximum length, L, of any edge of an MBST on the $\{p_i : 1 \leq i \leq n\}$ must be at most 2 greater than the maximum length edge in the MBST on $\{\ell_i : 1 \leq i \leq n\}$. The result follows. □

Our approximation for the BCU Problem, which we dub the "broadcast-from-center" hueristic, is not necessarily a constant-factor approximation, because if one takes n unit disks with non-empty intersection, then the ℓ_i can all be taken to equal one of the intersection points so that OPT$= 0$ while $L/2$ can be non-zero (and as big as 1). However, we can modify our heuristic to obtain a constant-factor approximation for *non-overlapping* unit disks, for a result analogous to that obtained by Yang et al. [29] for the case of MST with neighborhoods.

A problem for our heuristic, as it stands, in the case of non-overlapping disks, occurs if we have just two disks and these two disks are within ϵ of being tangent to one another. As $\epsilon \to 0$ one can choose broadcast locations ℓ_i increasingly close together so broadcast-from-center becomes arbitrarily bad. However, we can either deal with two disks as a special case, or take the following more principled approach: begin as in broadcast-from-center by picking the centers of all uncertainty disks, and then find an MBST on these centers, but at the end, "cinch-up" any leaf nodes by bringing the broadcast locations for these disks as close as possible to their parent nodes. In the case that the MBST is actually a simple path, cinch-up twice, first at one end, then at the other. This process ensures that we always obtain OPT for two disks. For three disks, we are not guaranteed to have OPT, but because points on three unit disks cannot come arbitrarily close to one another, the modified broadcast-from-center heuristic is a constant-factor approximation. See Fig. 14.

5 The WCU Problem

We next consider the Worst-Case Connectivity with Uncertainty (WCU) problem: Find the minimum value α such that for *any* choice of points P, the connectivity

graph G_α of P is connected. In what follows we assume that the number, n, of points and associated uncertainty regions is at least 2, since otherwise the problem is trivial.

We show a simple approximation algorithm for WCU that is within an additive factor of 1 and a multiplicative factor of 2 when the uncertainty regions are unit disks. Let $D(p; \lambda)$ denote the closed disk of radius λ about the point p.

Theorem 6 *Given a set of uncertainty regions that are unit disks $D_1, \ldots, D_n$ with centers $p_1, \ldots, p_n$, let L be the largest edge of an MBST on $\{p_i : 1 \leq i \leq n\}$. Then choosing $\alpha = L/2 + 1$ always results in the connectivity graph being connected and is at worst an OPT+1 approximation to the WCU Problem.*

Proof First note that the connectivity graph given by any selection of $\ell_i \in D(p_i; 1)$ and $\{D(\ell_i; L/2 + 1) : 1 \leq i \leq n\}$ is connected, because if (p_i, p_j) is an edge of an MBST on $\{p_k : 1 \leq k \leq n\}$ then $D(\ell_i; L/2 + 1) \cap D(\ell_j; L/2 + 1) \neq \varnothing$. We are thus left to show that choosing $\alpha = L/2 + 1$ is at worst an OPT+1 approximation. But clearly we can choose $\ell_i = p_i$ for all I and so the minimum α is $L/2$. Hence OPT $\geq L/2$ and the theorem is established. □

Theorem 7 *Given a set of uncertainty regions that are unit disks $D_1, \ldots, D_n$ with centers $p_1, \ldots, p_n$, let L be the largest edge of an MBST on $\{p_i : 1 \leq i \leq n\}$. Then choosing $\alpha = L/2 + 1$ is at worst a factor 2 approximation to OPT for the WCU problem.*

Proof Note that OPT+1 $\leq$ 2OPT as long as OPT ≥ 1 so, by Theorem 6, it suffices to show that OPT ≥ 1. As noted in the first paragraph of this section we are assuming that $n > 1$. Let p_j be a leftmost point amongst the $\{p_i : 1 \leq i \leq n\}$ and choose ℓ_j to be the leftmost point in $D(p_j; 1)$ and for all other $D(p_k; 1)_{k \neq j}$ choose ℓ_k to be the rightmost point in $D(p_k; 1)$. Then ℓ_j is at least distance 2 from each of the other ℓ_k and so we must choose $\alpha \geq 1$ to keep the connectivity graph connected. It follows that OPT ≥ 1 and the theorem is established. □

Theorem 7 shows that the simple broadcast-from-center heuristic can be no worse than a 2-approximation to the solution of the WCU Problem. However, we have thus far only found examples showing that the approximation can be (asymptotically) as bad as a $\sqrt{2}$-approximation, as the next theorem asserts.

Theorem 8 *Given any $L > 2$, there is an instance of the WCU Problem with uncertainty regions that are unit disks $D_1, \ldots, D_n$ with centers $p_1, \ldots, p_n$, such that the longest edge of an MBST on $\{p_i\}_{i=1}^n$ is L and OPT is as small as $\frac{\sqrt{L^2+4}}{2}$, and therefore the algorithm of Theorem 6 is at best a factor $\sqrt{2} - \epsilon$ approximation for arbitrarily small ϵ.*

Proof We distribute an even number of unit disks with centers equally spaced along a very large (relative to the unit disks) circle C. Let us call the distance between consecutive centers of disks centered along the large circle L. We will add more disks, but L will remain the longest edge of a spanning tree of the disk centers. Additionally, let us pick $\epsilon \ll L - 2$.

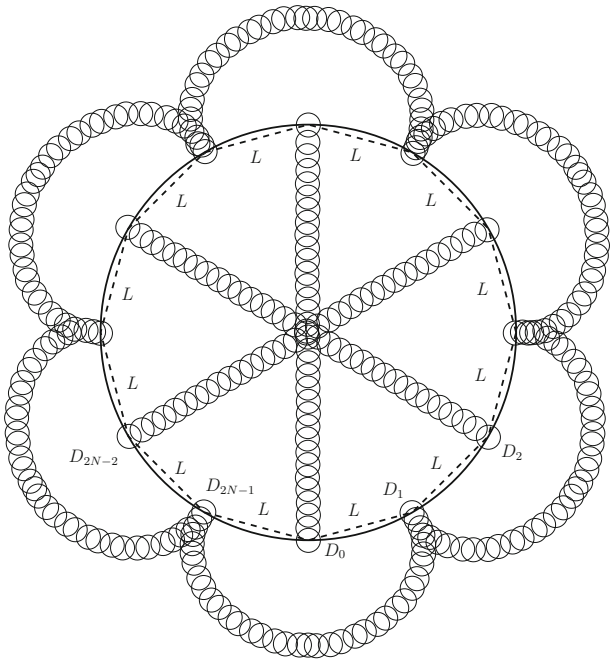

Fig. 15 The construction begins with an even number of equally spaced unit disks with centers along a *very large circle C*

The construction contains a large number of highly overlapping disks in addition to the disks whose centers lie along C. See Fig. 15 for a sketch. The drawing is approximate in several respects. First of all, C is much, much larger than drawn, so that if the bottom of C is, say, tangent to the x-axis, and the center of the bottom unit disk, D_0, has y-coordinate equal to 0, then the center-points of the first unit disks to the left and right of D_0 along C, each have y-coordinate less than $\epsilon/3$. In addition to the disks along C, there is a sequence of disks going from D_0 to its diametrically opposite unit disk whose centers lie along the connecting diameter. The centers of these disks are all distance ϵ, one from the next, along the diameter. If we number the C-centered disks in counter-clockwise order, D_0, ..., D_{2N-1}, then we have a similar set of disks extending from each of the disks D_2, D_4, ..., D_{2N-2}. The key observation is that we can add such diametrically centered disks in such a way that the center of disks extending from D_j to the center of C are each more than distance L from any other D_k for $k \neq j$ – thus the choice of D_1 and D_{2N-1} with y-coordinate less than $\epsilon/3$. On the other hand, the odd numbered unit disks D_1, D_3, ..., D_{2N-1}, with representative element that we shall call D_i, each have a set of unit disks running from D_i to D_{i+2} with centers each ϵ from the next, but with the disks running in almost circular patterns on the outside of C. An important point in this case, is that the disks start out emanating from D_i along a diametric line, and then bend around so that their centers are never within L of D_{i+1}.

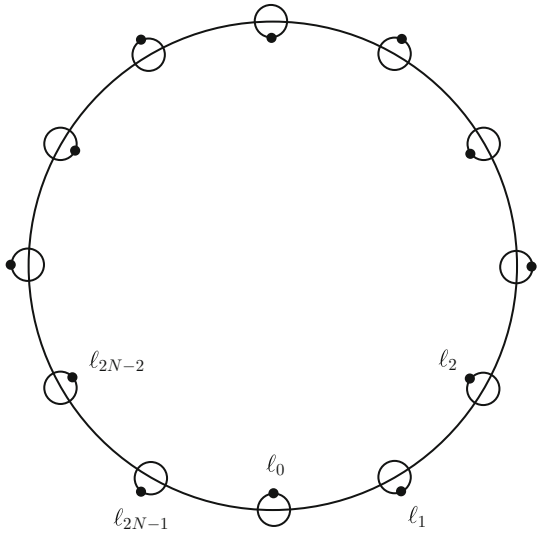

Fig. 16 The distance between ℓ_i and ℓ_{i+1} (in the cyclical ordering) is just slightly less than $\sqrt{L^2+4}$ since the distance between successive disk centers (which we suppose to be x-axis aligned) is L, and the distance between the *top* and *bottom* of the successive disks, in the y-direction, is approximately 2

We claim that for such an arrangement of unit disks, the maximum distance between locations $\ell_r \in D_r$ in a spanning tree can be as small as (and in fact slightly smaller than) $\sqrt{L^2+4}$, where the set $\{D_r\}$ consists not just of the disks D_i with centers along C, but all the other unit disks depicted in Fig. 15 as well. If D_i, D_{i+2} are two consecutive disks in the cyclical ordering of C-centered disks with I odd, let $\{D_{i_k}\}$ denote the set of disks running from D_i to D_{i+2} outside of C. Further, if D_j, D_{j+N} are diametrically opposite C-centered disks with j even, let $\{D_{j_k}\}$ denote the set of disks running diametrically between D_j and D_{j+N}. To verify our claim about $\{\ell_i\}$ with maximum bottleneck spanning tree edge length slightly less than $\sqrt{L^2+4}$, pick $\ell_i \in D_i$ for even I to be the point in D_i closest to the center of C and $\ell_i \in D_i$ for odd I to be the point in D_i furthest from the center of C. See Fig. 16. Regardless of the choice of the $\ell_{i_j} \in D_{i_j}$ it is clear that $\bigcup\{D(\ell_i;\alpha)\} \cup \bigcup\{D(\ell_{i_j};\alpha)\}$ is connected if 2α is the distance between consecutive locations ℓ_i, ℓ_{i+1} (in the cyclical ordering), and that this distance is, as claimed, just slightly less than $\sqrt{L^2+4}$. Let us designate this distinguished choice of the $\ell_i \in D_i$ by ℓ_i^*, and the associated α by α^*.

For these $\{D_i\}$ and $\{D_{i_j}\}$, if there were any choice of $\{\ell_i\}, \{\ell_{i_j}\}$ making α any larger, then we would have to pick one of the ℓ_i to the left or right of the diametric line through the center of C and D_i. It is easy to check that the result of such a choice is that there would be some cyclically ordered pair ℓ_j, ℓ_{j+1} whose distance $d(\ell_j, \ell_{j+1}) < d(\ell_j^*, \ell_{j+1}^*) = \alpha^*$. But then $D(\ell_j;\alpha^*) \cup D(\ell_{j+1};\alpha^*)$ connects ℓ_j, ℓ_{j+1} and $\bigcup\{D(\ell_{2k};\alpha^*)\} \cup \bigcup\{D(\ell_{(2k)_j};\alpha^*)\}$ connects the even-indexed ℓ_{2k} and any associated choices for $\ell_{(2k)_j}$, while $\bigcup\{D(\ell_{2k+1};\alpha^*)\} \cup \bigcup\{D(\ell_{(2k+1)_j};\alpha^*)\}$ connects the odd-indexed ℓ_{2k+1} and any associated choices for $\ell_{(2k+1)_j}$. It follows that $\alpha \le \alpha^*$, contrary to assumption, and so the fact that OPT can be as small as $\frac{\sqrt{L^2+4}}{2}$ is established.

The algorithm of Theorem 6 picked $\alpha = \frac{L}{2} + 1$, so picking L sufficiently close to 2 yields $\frac{\frac{L}{2}+1}{\frac{\sqrt{L^2+4}}{2}} = \frac{L+2}{\sqrt{L^2+4}}$ sufficiently close to $\sqrt{2}$, completing the proof. □

6 Conclusions

A number of open problems remain. It would be interesting to show NP-hardness results for the BCU problem for other uncertainty regions, such as disks. It is also possible that techniques from convex optimization could be used to design approximation algorithms for BCU for, say, line segments or squares. We conjecture that BCU for the case of line segments is W[1]-hard and hence our exact algorithm is unlikely to be improved upon significantly.

Although we have been able to obtain several NP-hardness results for BCU, we do not have any complexity lower bounds for WCU which, a priori, seems harder. It is an interesting open question to improve our approximation algorithms for both these problems.

In conclusion, our work on connectivity problems for uncertainty regions motivated by wireless network scenarios suggests that this area provides a rich collection of problems for further investigation.

Acknowledgements We are grateful for two Bellairs workshops supporting this research: the 8th and 9th McGill-INRIA-UVictoria Workshop on Computational Geometry in 2009 and 2010. We also thank the anonymous reviewers for many helpful and constructive comments that greatly helped to improve the overall presentation. We also acknowledge financial support by a number of different agencies, as follows. Erin Chambers was supported by NSF Grants CCF 1054779 and IIS 1319573. Alejandro Erickson was supported by the EPSRC, Grant No. EP/K015680/1. Ulrike Stege was supported by an NSERC Discovery Grant. Svetlana Stolpner was supported by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT). Venkatesh Srinivasan was supported by an NSERC Discovery Grant. Sue Whitesides was supported by an NSERC Discovery Grant.

References

1. Abellanas, M., Hurtado, F., Ramos, P.: Structural tolerance and Delaunay triangulation. Inf. Process. Lett. **71**, 221–227 (1999)
2. Alt, H., Arkin, E., Brönnimann, H., Erickson, J., Fekete, S., Knauer, C., Lenchner, J., Mitchell, J., Whittlesey, K.: Minimum-cost coverage of point sets by disks. In: Proceedings of the 22nd ACM Symposium on Computational Geometry (SoCG), pp. 449–458 (2006)
3. Arkin, E.M., Dieckmann, C., Knauer, C., Mitchell, J.S.B., Polishchuk, V., Schlipf, L., Yang, S.: Convex transversals. In: Proceedings of the 12th International Symposium on Algorithms and Data Structures (WADS), pp. 49–60 (2011)
4. Arkin, E.M., Hassin, R.: Approximation algorithms for the geometric covering salesman problem. Discrete Appl. Math. **55**(3), 197–218 (1994)
5. Chambers, E.W., Erickson, A., Fekete, S.P., Lenchner, J., Sember, J., Srinivasan, V., Stege, U., Stolpner, S., Weibel, C., Whitesides, S.: Connectivity graphs of uncertainty regions. In: 21st International Symposium on Algorithms and Computation (ISAAC), number 5307 in Springer LNCS, pp. 434–445 (2010)
6. Clementi, A.E.F., Penna, P., Silvestri, R.: On the power assignment problem in radio networks. Technical Report TR00-054, Electronic Colloquium on Computational Complexity (2000)
7. de Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulos, C., Overmars, M.H., van der Stappen, A.F.: TSP with neighborhoods of varying size. J. Algorithms **57**(1), 22–36 (2005)

8. Ding, H., Xu, J.: Solving the chromatic cone clustering problem via minimum spanning sphere. In: 38th International Colloquium on Automata, Languages and Programming (ICALP), volume 6755 of Springer LNCS, pp. 773–784 (2011)
9. Disser, Y., Mihalák, M., Montanari, S.: Max shortest path for imprecise points. In: 31st European Workshop on Computational Geometry, pp. 184–187 (2015)
10. Disser, Y., Mihalák, M., Montanari, S., Widmayer, P.: Rectilinear shortest path and rectilinear minimum spanning tree with neighborhoods. In: Proceedings of the 3rd International Symposium on Combinatorial Optimization (ISCO), pp. 208–220 (2014)
11. Dorrigiv, R., Fraser, R., He, M., Kamali, S., Kawamura, A., López-Ortiz, A., Seco, D.: On minimum- and maximum-weight minimum spanning trees with neighborhoods. In: Proceedings of the 10th Workshop on Approximation and Online Algorithms (WAOA), pp. 93–106 (2012)
12. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 473–482 (2003)
13. Duchet, P., Hamidoune, Y.O., Vergnas, M.L., Meyniel, H.: Representing a planar graph by vertical lines joining different levels. Discrete Math. **46**(3), 319–321 (1983)
14. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. In: Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 38–46 (2001)
15. Fiala, J., Kratochvíl, J., Proskurowski, A.: Systems of distant representatives. Discrete Appl. Math. **145**(2), 306–316 (2005)
16. Fuchs, B.: On the hardness of range assignment problems. In: Proceedings of the 6th Italian Conference on Algorithms and Complecity (CIAC), pp. 127–138 (2006)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
18. Gudmundsson, J., Levcopoulos, C.: A fast approximation algorithm for TSP with neighborhoods. Nord. J. Comput. **6**(4), 469–488 (1999)
19. Lev-Tov, N., Peleg, D.: Exact algorithms and approximation schemes for base station placement problems. In: Proceedings of the 8th Scandinavian Workshop Algorithm Theory (SWAT), pp. 90–99 (2002)
20. Lev-Tov, N., Peleg, D.: Polynomial time approximation schemes for base station coverage with minimum total radii. Comput. Netw. **47**(4), 489–501 (2005)
21. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982)
22. Löffler, M., van Kreveld, M.: Largest and smallest convex hulls for imprecise points. Algorithmica **56**, 235–269 (2010)
23. Löffler, M., van Kreveld, M.: Largest bounding box, smallest diameter, and related problems on imprecise points. Comput. Geom. Theory Appl. **43**, 419–433 (2010)
24. Mata, C., Mitchell, J.: Approximation algorithms for geometric tour and network design problems. In: Proceedings of the 11th ACM Symposium on Computational Geometry (SoCG), pp. 360–369 (1995)
25. Mitchell, J.S.B.: A PTAS for TSP with neighborhoods among fat regions in the plane. In: Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 11–18 (2007)
26. Pan, X., Li, F., Klette, R.: Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons. In: Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG), pp. 175–178 (2010)
27. Parker, G., Rardin, R.L.: Guaranteed performance heuristics for the bottleneck traveling salesman problem. Oper. Res. Lett. **2**(6), 269–272 (1984)
28. Rosenstiehl, P., Tarjan, R.E.: Rectilinear planar layouts and bipolar orientations of planar graphs. Discrete Comput. Geom. **1**, 343–353 (1986)
29. Yang, Y., Lin, M., Xu, J., Xie, Y.: Minimum spanning tree with neighborhoods. In: Proceedings of the 3rd Conference on Algorithmic Aspects on Information and Management (AAIM), pp. 306–316. Springer, Berlin, Heidelberg (2007)