Reconstructing Graphs from Connected Triples

Paul Bastide[1](✉), Linda Cook[2], Jeff Erickson[3], Carla Groenland[4], Marc van Kreveld[4], Isja Mannens[4], and Jordi L. Vermeulen[4]

[1] LaBRI - Bordeaux University, Bordeaux, France
paul.bastide@ens-rennes.fr
[2] Institute for Basic Science, Discrete Math Group, Daejeon, Republic of Korea
linda.cook@ibs.re.kr
[3] University of Illinois, Urbana-Champaign, Champaign, USA
jeffe@illinois.edu
[4] Utrecht University, Utrecht, The Netherlands
{c.e.groenland,m.j.vankreveld,i.m.e.mannens}@uu.nl

Abstract. We introduce a new model of indeterminacy in graphs: instead of specifying all the edges of the graph, the input contains all triples of vertices that form a connected subgraph. In general, different (labelled) graphs may have the same set of connected triples, making unique reconstruction of the original graph from the triples impossible. We identify some families of graphs (including triangle-free graphs) for which all graphs have a different set of connected triples. We also give algorithms that reconstruct a graph from a set of triples, and for testing if this reconstruction is unique. Finally, we study a possible extension of the model in which the subsets of size k that induce a connected graph are given for larger (fixed) values of k.

Keywords: Algorithms · Graph reconstruction · Indeterminacy · Uncertainty · Connected Subgraphs

1 Introduction

Imagine that we get information about a graph, but not its complete structure by a list of edges. Does this information uniquely determine the graph? In this paper we explore the case where the input consists of all triples of vertices whose induced subgraph is connected. In other words, we know for each given triple of vertices that two or three of the possible edges are present, but we do not know which ones. We may be able to deduce the graph fully from all given triples.

LC is supported by the Institute for Basic Science (IBS-R029-C1) and CG by Marie-Skłodowska Curie grant GRAPHCOSY (number 101063180). MvK and JV are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.651.

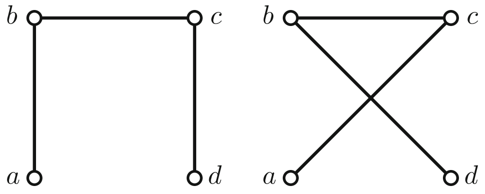

Fig. 1. Two different labelled trees that give the same set of connected triples.

As a simple example, assume we are given the (unordered, labelled) triples abc, bcd, and cde. Then the only (connected) graph that matches this specification by triples is the path a—b—c—d—e. On the other hand, if we are given all possible triples on a set of four vertices a, b, c, d except for abc, then there are several graphs possible. We must have the edges ad, bd, and cd, and zero or one of the edges ab, bc, and ca. See Fig. 1 for another example.

This model of indeterminacy of a graph does not use probability and is perhaps the simplest combinatorial model of partial information. Normally a graph is determined by pairs of vertices which are the edges; now we are given triples of vertices with indeterminacy on the edges between them. As such, we believe this model is interesting to study.

As illustrated in our previous example, there are cases where reconstruction of the graph from the set T of triples is unique and there are cases where multiple (labelled) graphs may have the same set of triples. There are also cases where T is not consistent with any graph, such as $T = \{abc, cde\}$. Can we characterize these cases, and what can we say if we have additional information, for example, when we know that we are reconstructing a tree or a triangle-free graph?

1.1 Our Results

After preliminaries in Sect. 2, we provide two relatively straightforward, general algorithms for reconstruction in Sect. 3. One runs in $O(n^3)$ time when the triples use n vertex labels, and the other runs in $O(n \cdot |T|)$ time when there are $|T|$ triples in the input. These algorithms return a graph that is consistent with the given triples, if one exists, and decide on uniqueness.

Then, we give an $O(|T|)$ time algorithm to reconstruct trees on at least five vertices, provided that the unknown graph is known to be a tree, in Sect. 4. In fact, all triangle-free graphs can be reconstructed, provided we know that the unknown graph is triangle-free. We give an algorithm running in expected $O(|T|)$ time for this in Sect. 5. Moreover, we show that 2-connected outerplanar graphs and triangulated planar graphs can be uniquely reconstructed.

In Sect. 6 we study a natural extension of the model where we are given the connected k-sets of a graph for some fixed $k \geq 4$, rather than the connected triples. We show the largest value of k such that each n-vertex tree is distinguished from other trees by its set of connected k-sets is $\lceil n/2 \rceil$. A similar threshold is shown for the random graph. Finally, we show that graphs with girth strictly larger than k, on at least $2k-1$ vertices, can be uniquely reconstructed, among the class of thus graphs, by their collection of connected k-sets.

1.2 Related Work

The problem of graph reconstruction arises naturally in many cases where some unknown graph is observed indirectly. For instance, we may have some (noisy) measurement of the graph structure, or only have access to an oracle that answers specific types of queries. Much previous research has been done for specific cases, such as reconstructing metric graphs from a density function [10], road networks from a set of trajectories [1], graphs using a shortest path or distance oracle [19], labelled graphs from all r-neighbourhoods [25], or reconstructing phylogenetic trees [7]. A lot of research has been devoted to the *graph reconstruction conjecture* [21,29], which states that it is possible to reconstruct any graph on at least three vertices (up to isomorphism) from the multiset of all (unlabeled) subgraphs obtained through the removal of one vertex. This conjecture is open even for planar graphs and triangle-free graphs, but has been proved for outerplanar graphs [15] and maximal planar graphs [22]. We refer the reader to one of the many surveys (e.g. [5,17,23,28]) for further background. Related to our study of the random graph in Sect. 6 is a result from Cameron and Martins [8] from 1993, which implies that for each graph H, with high probability the random graph $G \sim G(n, \frac{1}{2})$ can be reconstructed from the set of (labelled) subsets that induce a copy of H (up to complementation if H is self-complementary).

Many types of uncertainty in graphs have been studied. Fuzzy graphs [27] are a generalisation of fuzzy sets to relations between elements of such sets. In a fuzzy set, membership of an element is not binary, but a value between zero and one. Fuzzy graphs extend this notion to the edges, which now also have a degree of membership in the set of edges. Uncertain graphs are similar to fuzzy graphs in that each edge has a number between zero and one associated with it, although here this number is a probability of the edge existing. Much work has been done on investigating how the usual graph-theoretic concepts can be generalised or extended to fuzzy and uncertain graphs [20,24].

2 Preliminaries

All graphs in this paper are assumed to be connected, finite, and simple. Let G be an unknown graph with n vertices and let T be the set of all triples of vertices that induce a connected subgraph in G. Since the graph is connected, we can recover the vertex set V of G easily from T. We will use $\overline{T}$ to denote the complement of this set T, i.e. $\overline{T}$ is the set of all triples of vertices for which the induced subgraph is not connected. Note that $\left|T \cup \overline{T}\right| = \binom{n}{3} \in \Theta(n^3)$.

Observe that both the presence and absence of a triple gives important information: in the former case, at most one of the three possible edges is absent, whereas in the latter case, at most one of these edges is present.

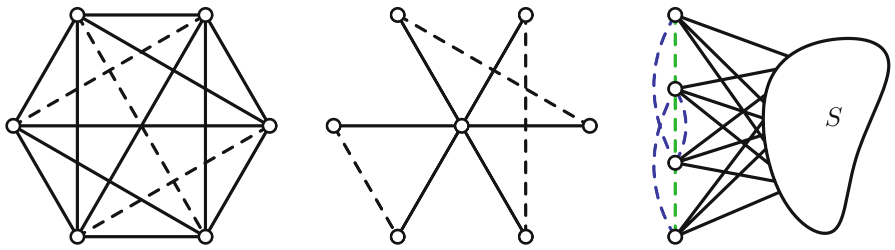

Fig. 2. Three classes of ambiguous triples: a complete graph minus any independent set of edges, a star graph plus any (partial) matching of the leaves, and a path of length four in which all vertices are fully adjacent to some set S. In this last case, we cannot tell the difference between the red and green path. (Color figure online)

It is possible that graphs that are not the same (as labelled graph) or even not isomorphic yield the same set of triples, for example, a path on three vertices and a triangle. We also give examples of larger graphs that cannot be distinguished from their set of connected triples in Fig. 2.

We will make use of (LSD) radix string sorting (as described in e.g. [9]) to sort a collection of t sets of cardinality k in time $\mathcal{O}(tk)$ in several of the algorithms presented in this paper.

3 Algorithm for Finding Consistent Graphs from Triples

Given a set of triples T, we can find a graph G consistent with those triples by solving a 2-SAT formula. The main observation here is that the presence of a triple abc means that at least two of the edges ab, ac and bc must exist, whereas the absence of a triple means at most one of the edges can exist. We can then construct a 2-SAT formula where each variable corresponds to an edge of the graph, and truth represents presence of that edge. For each triple $abc \in T$, we add clauses $(ab \vee ac)$, $(ab \vee bc)$ and $(ac \vee bc)$ to the formula. For each triple $abc \in \overline{T}$, we add clauses $(\neg ab \vee \neg ac)$, $(\neg ab \vee \neg bc)$ and $(\neg ac \vee \neg bc)$. A graph consistent with the set of triples can then be found by solving the resulting 2-SAT formula and taking our set of edges to be the set of true variables in the satisfying assignment. If the formula cannot be satisfied, no graph consistent with T exists.

We can solve the 2-SAT formula in linear time with respect to the length of the formula [2, 11]. We add a constant number of clauses for each element of T and $\overline{T}$, so our formula has length $O(\left|T \cup \overline{T}\right|)$. As $\left|T \cup \overline{T}\right| = \binom{n}{3}$, this gives us an $O(n^3)$ time algorithm to reconstruct a graph with n vertices. However, we prefer an algorithm that depends on the size of T, instead of also on the size of $\overline{T}$. We can eliminate the dependency on the size of $\overline{T}$ by observing that some clauses can be excluded from the formula because the variables cannot be true.

Lemma 1. *We can find a graph G consistent with T in $O(n \cdot |T|)$ time, or output that no consistent graph exists.*

Proof. The basic observation that allows us to exclude certain clauses from the formula is that if there is no connected triple containing two vertices a and b, the variable ab will always be false. Consequently, if we have a triple $abc \in \overline{T}$ for which at most one of the pairs ab, ac and bc appear in some connected triple, we do not need to include its clauses in the formula, as at least two of the variables will be false, making these clauses necessarily satisfied.

We can construct the formula that excludes these unnecessary clauses in $O(n \cdot |T|)$ time as follows. We build a matrix $M(i,j)$, where $i,j \in V(G)$ with each entry containing a list of all vertices with which i and j appear in a connected triple, i.e. $M(i,j) = \{x \mid ijx \in T\}$. This matrix can be constructed in $O(n^2 + |T|)$ time, by first setting every entry to $\emptyset$ (this takes n^2 time) and then running through T, adding every triple abc to the entries $M(a,b)$, $M(b,c)$ and $M(a,c)$. We also sort each list in linear time using e.g. radix sort. As the total length of all lists is $O(|T|)$, this takes $O(n^2 + |T|)$ time in total.

Using this matrix, we can decide which clauses corresponding to triples from $\overline{T}$ to include as follows. For all pairs of vertices (a,b) that appear in some connected triple (i.e. $M(a,b) \neq \emptyset$), we find all x such that $abx \in \overline{T}$. As $M(a,b)$ is sorted, we can find all x in $O(n)$ time by simply recording the missing elements of the list $M(a,b)$. We then check if $M(a,x)$ and $M(b,x)$ are empty. If either one is not, we include the clause associated with $abx \in \overline{T}$ in our formula. Otherwise, we can safely ignore this clause, as it is necessarily satisfied by the variables for ax and bx being false.

Our algorithm takes $O(n)$-time for each non-empty element of $M(i,j)$, of which there are $O(|T|)$, plus $O(n^2)$ time to traverse the matrix. The total time to construct the formula is $O(n^2 + n \cdot |T|)$. As $|T| \in \Omega(n)$ for connected graphs, this simplifies to $O(n \cdot |T|)$ time. The resulting formula also has $O(n \cdot |T|)$ length, and can be solved in time linear in that length. $\square$

Observe that this is only an improvement on the naive $O(n^3)$ approach if $|T| \in o(n^2)$. We also note that we can test the uniqueness of the reconstruction in the same time using Feder's approach for enumerating 2-SAT solutions [12].

4 Unique Reconstruction of Trees

In this section, we prove the following result.

Theorem 1. *Let T be a set of triples, and let it be known that the underlying graph $G = (V,E)$ is a tree. If $n \geq 5$, then G can be uniquely reconstructed in $O(|T|)$ time.*

Let us briefly examine trees with three or four vertices. A tree with three vertices is always a path and it will always have one triple with all three vertices. We do not know which of the three edges is absent. A tree with four vertices is either a path or a star. The path has two triples and the star has three triples. For the star, the centre is the one vertex that appears in all three triples, and hence the reconstruction is unique. For the path, we will know that the graph is a path, but we will not know in what order the middle two vertices appear (see Fig. 1).

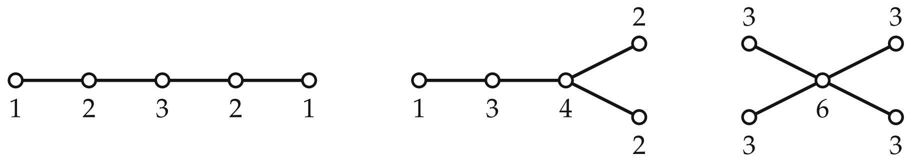

Fig. 3. All trees on five vertices, and the number of triples each vertex occurs in.

Next we consider trees with at least five vertices. We first show that we can recognise all leaves and their neighbours from the triples. In the following, we say that a vertex v *dominates* a vertex u if v appears in all the triples that u appears in. If u is a leaf, then it is dominated by its unique neighbour v. It is possible that u is also dominated by a neighbour of v, but in this case uvw will be the only triple containing u, and v will be dominated by w. Moreover, if $|V| \geq 4$, there exists a triple vwx for some vertex x different from u. This can be used to recognise the leaves as long as $|V| \geq 4$. Moreover, if $|V| \geq 5$, we can also identify the neighbour v of each leaf u as either the unique vertex that dominate u or when u is dominated by two vertices v and w, there exits $x, y \in V$ such that wxy is a triple and w dominates v. We can use this to prove that any tree can be reconstructed from its triples, provided that we know that the result must be a tree and $|V| \geq 5$, since we can iteratively recognise and remove vertices of degree 1, while recording where to 'glue them back at the end' until at most four vertices remain. We can complete the reconstruction via some closer examination of the connected triples in the original tree that contain the remaining vertices.

In order to derive an optimal, $O(|T|)$ time reconstruction algorithm, we will use a further characterisation of vertices of a tree using the triples. The main idea is that we can recognise not only leaves, but also other vertices where we can reduce the tree. If a vertex v has degree 2 in a tree, then there are two nodes w, w' such that every triple with v also contains w or w' (or both). The converse is not true for two reasons: if v is a leaf, it also has the stated property, and if v has degree 3 where at least one neighbour is a leaf, then it has this property as well. This brings us to the following characterisation.

Lemma 2. *A vertex v of a tree G of at least five vertices with triple set T is:*

(i) *a leaf if and only if v is dominated by some vertex w and does not dominate any vertex itself;*

(ii) *if v is not a leaf, then v is (a) a node of degree 2, or (b) a node of degree 3 with at least one leaf neighbour, if and only if there are two nodes w_1, w_2 such that all triples with v also contain w_1 or w_2.*

Moreover, both characterisations can be checked in time $O(|T_v|)$, when the set of triples T_v that include v is given for all $v \in V(G)$.

Proof. A leaf v can necessarily only appear in triples with its adjacent vertex w, as it is not adjacent to any other vertices by definition. A leaf is therefore always dominated by its neighbour w. Since $|V| \geq 5$, v does not dominate any vertex. Conversely, suppose that v is dominated by some vertex w and does not dominate any other vertex. It is straightforward to check that v cannot be dominated if it has three neighbours or if it has two non-leaf neighbours. Since v does not dominate any vertices, it does not have a leaf neighbour. So v must be a leaf itself. This proves (i).

The second characterisation can be seen as follows. If v has degree at least 4, then no w_1, w_2 as in (ii) exist, which is easily verified by looking at the triples with v and its neighbours only. Furthermore, if v has degree 3 and none of its neighbours are leaves, then again there are no such w_1, w_2. On the other hand, in case (a) the two neighbours can be taken as w_1, w_2 and in case (b) w_1, w_2 can be chosen to be two neighbours of v so that the unique neighbour of v that is not in $\{w_1, w_2\}$ is a leaf.

For testing (i), take any triple $vab \in T_v$, and test both a and b separately if they are the sought w. For testing (ii), take any triple $vab \in T_v$. If characterisation (ii) holds, then w_1 must be a or b. We try both as follows: For $w \in \{a, b\}$ we remove all triples with w from T_v. Then $w = w_1$ if and only if all of the remaining triples of T_v all contain some $w_2 \neq v$. We test this by looking at some remaining triple in $vcd \in T_v$ and testing whether either c or d is contained in every other remaining triple. In total, we get four options to test for w_1 and w_2; each option is easily checked in $O(|T_v|)$ time. □

The vertices of V partition into V', V'', and V''', where V' contains the leaves, V'' contains the vertices that are not leaves but satisfy the second condition of the lemma, and $V''' = V \setminus (V' \cup V'')$. Note that more than half of the vertices of G are in $V' \cup V''$. We next turn to the proof of Theorem 1. We will assume that there is a total order on the vertex set of G and that the connected triples uvw are stored in an ordered tuple with $u < v < w$. For each triple uvw in T, we generate vwu and wuv as well. We collect the triples with the same first vertex to generate T_v for all $v \in V$. We will begin by showing how we recognize whether each vertex is in V', V'' or V'''.

Then, for all $v \in V$, we use T_v to test if v is dominated by some vertex u. We can find the vertices that dominate v, in time $O(|T_v|)$, by examining the first triple vwx and noting that only w and x can dominate v. We then check every other triple in T_v for the presence of w and x. By labeling dominated/dominating vertices as we go we can, in time $O(|T|)$, find all leaves as vertices that are dominated by some vertex and don't dominate any vertex themselves, i.e. all vertices that satisfy condition (i) of Lemma 2. We then check T_v for the remaining vertices, to see if condition (ii) is satisfied and find the vertices w_1 and w_2 in a similar fashion. We again start with the first triple vwx and check if there is some triple vyz that does not contain w or x. If not, then $w1 = w$ and $w_2 = x$. Otherwise we have four candidates w, x, y, z for w_1 and w_2 and we check, for every pair, whether there is a triple that contains neither of them. This can be done in $O(|T_v|)$ time.

After this, we remove all triples containing a leaf from T. Let G' be the graph obtained by removing all leaves and incident edges. Then the new triple set is the set of connected triples for this graph G', and we can recover G from G'.

For all vertices in V'' note that they can no longer be vertices of degree 3 in G', but they may have become leaves. We test this and consider the subset $W \subseteq V''$ of vertices that have not become leaves.

The subgraph of G' induced on W consists of a disjoint union of paths. Let $v \in W$. Then v has exactly two neighbours in $V(G')$ and they are the two vertices w_1, w_2 satisfying the second condition of the Lemma 2. As mentioned above, we can find these two vertices w_1, w_2 for each $v \in W$ in time $O(|T_v|)$. In particular, we know all path components of $G'[W]$, as well as the unique vertices in $V(G') \setminus W$ that the endpoints of any such path are adjacent to. Suppose that v_1—v_2—...—v_ℓ. is one of the path components of $G[W]$. Let $x_1, x_2 \in V(G') \setminus W'$ such that x_1 is the other neighbour of v_1, and x_2 is the other neighbour of v_k (in G'). We record the edges x_1v_1 and x_2v_k, as well as the edges and vertices in the path $v_1, \ldots, v_\ell$. Then we replace each triple ux_1v_1 by ux_1x_2 and each triple v_kx_2u by x_1x_2u. Afterwards, we discard all triples that contain any of $v_1, \ldots, v_\ell$. Let G'' be the graph obtained by deleting $v_1, \ldots, v_\ell$ and adding the edge $\{x_1, x_2\}$. The resulting triple set is the triple set for G'', and we can recover G from G''. We repeat this for all path components. Note that G'' is a tree, if and only if G' is a tree and thus we maintain throughout that the stored triple set corresponds to a tree, and that we can reconstruct the original tree G from knowing this tree and the additional information that we record.

Finally, we also remove all leaves in $V'' \setminus W$ by discarding more triples, similar to the first leaf removal. This process takes time linear in $|T|$, and reduces the number of vertices occurring in T to half or less. We recurse the process on the remaining tree until it has size five, at which point we can uniquely identify the structure of the tree by simply looking at the number of triples each vertex occurs in (see Fig. 3). We may not remove all vertices of V' or V'' if the remaining tree would be smaller than five vertices; in that case, we can simply leave some leaf or not contract the paths in W completely. A standard recurrence shows that the total time used is $O(|T|)$. This finishes the proof of Theorem 1.

We note that if the tree contains no leaves that are siblings, then we do not need to know that the graph is a tree for unique reconstruction.

5 Further Reconstructible Graph Classes

In this section, we give larger classes of graphs for which the graphs that are determined by their set of connected triples.

Theorem 2. *There is an algorithm that reconstructs a graph G on $n \geq 5$ vertices that is known to be triangle-free from its set T of triples in deterministic $O(|T| \log(|T|))$ time or randomized $O(|T|)$ expected time.*

Proof. Let T be the given list of connected triples. For every triple abc we create three ordered copies abc, acb, bca. We then sort the list in lexicographical order in $O(|T|)$ time using radix sort. For every potential edge ab that appears as the first two vertices of some triple we test whether it is an edge as follows.

If we find two triples abc and abd for some $c, d \in V(G)$, we search for the triples acd and bcd. If $ab \notin E(G)$, then we must have that $bc, ac, bd, ad \in E(G)$, and thus both acd and bcd are connected. Therefore, if either triple is not in the list, we know that ab is an edge. Otherwise, we find that $\{a, b, c, d\}$ induces a C_4. We then search for another vertex e that appears in a triple with any of a, b, c, d and reconstruct the labeling of the C_4 as follows. Suppose $G[\{a, b, c, d\}]$ induces a C_4 and we try to retrieve the exact order of the vertices in the cycle. Assume w.l.o.g. that a has the remaining vertex e as a neighbour, then since G is triangle-free, e is not adjacent to b and d. This means that bde is known to be disconnected, whereas abe, ade are connected. If e is not a neighbour of c, then ace is not connected. Hence, if any of a, b, c, d has a private neighbour (one not adjacent to other vertices in the cycle), then we get the labeling of our C_4 (and find that e is a private neighbour of a). If e is adjacent to c besides a, then we know the following two vertex sets also induce C_4's: $abce, acde$. We know e is adjacent to two out of $\{a, b, c\}, \{a, d, c\}$ but not to b and d. So we find e is adjacent to a and c and also have found our labeling.

Suppose abc is the unique triple we found in our list that begins with ab. We check for triples starting in ac or bc. Since one of the three vertices involved must be adjacent to some other vertex d, one of these two potential edges must appear in at least two triples. We can then use the previous methods to reconstruct some subgraph containing a, b and c. The result will tell us whether ab is an edge or not.

Note that we can search our list for a specific triple or a triple starting with a specific pair of vertices, in time $O(\log(|T|))$ using binary search. This means that the above checks can be done in $O(\log(|T|))$ time. By handling potential edges in the order in which they appear in the list, we only need to run through the list once, and thus we obtain a runtime of $O(|T| \log(|T|))$.

Using a data structure for "perfect-hashing" like the one described by Fredman, Komlós and Szemerédi [13], we can query the required triples in $O(1)$ time and thus reconstruct the graph in $O(|T|)$ time. Given a list S of n distinct items from a set of m items, [13] describes an algorithm to create a data-structure that can store n items, and allow for membership in S to be queried in constant time for all m and n. The construction of the data-structure is a randomized process that takes $O(n)$ time in expectation. In our case S is the list of triples and our universe is $V(G) \times V(G) \times V(G)$ so the process takes time $O(T)$. □

We also prove the following two results in the full version [3].

Theorem 3. *We can reconstruct any graph on $n \geq 6$ vertices that is known to be 2-connected and outerplanar from its list of connected triples.*

Our approach is similar to the one for trees: we show that we can identify a vertex of degree two, and remove it from the graph by 'merging' it with one of its neighbours.

A *triangulated planar graph*, also called a *maximal planar graph*, is a planar graph where every face (including the outer face) is a triangle.

Theorem 4. *Let T be a set of triples, and let it be known that the underlying graph $G = (V, E)$ is planar and triangulated. Then G can be uniquely reconstructed from T if $n \geq 7$.*

To show this result, we first show that unique reconstruction of such graphs is possible if they do not contain any separating triangles: A *separating triangle* is a triangle in the graph whose removal would result in the graph being disconnected. We then show we can reduce the problem of reconstructing a triangulated planar graph that contains a separating triangle to reconstructing triangulated planar graphs that do *not* contain a separating triangle.

6 Reconstruction from Connected k-Sets

For $k \geq 2$ and a graph $G = (V, E)$, we define the *connected k-sets* of G as the set $\{X \subseteq V \mid |X| = k \text{ and } G[X] \text{ is connected}\}$. We will denote the set of neighbours of a vertex v by $N(v)$.

Observation 1. *For $k' \geq k \geq 2$, the connected k'-sets of a graph are determined by the connected k-sets.*

Indeed, a $(k+1)$-set $X = \{x_1, \ldots, x_{k+1}\} \subseteq V$ induces a connected subgraph of G if and only if for some $y, z \in X$, both $G[X \setminus \{y\}]$ and $G[X \setminus \{z\}]$ are connected.

Given a class $\mathcal{C}$ of graphs, we can consider the function $k(n)$, where for any integer $n \geq 1$, we define $k(n)$ to be the largest integer $k \geq 2$ such that all (labelled) n-vertex graphs in $\mathcal{C}$ have a different collection of connected k-sets. By Observation 1, asking for the largest such k is a sensible question: reconstruction becomes more difficult as k increases. We will always assume that we only have to differentiate the graph from other graphs in the graph class, and remark that often the *recognition problem* (is $G \in \mathcal{C}$?) cannot be solved even from the connected triples.

First, we give an analogue of Theorem 1. The proof is given in the full version [3].

Theorem 5. *If it is known that the input graph is a tree, then the threshold for reconstructing trees is at $\lceil n/2 \rceil$: we can reconstruct an n-vertex tree from the connected k-sets if $k \leq \lceil n/2 \rceil$ and we cannot reconstruct the order of the vertices in an n-vertex path if $k \geq \lceil n/2 \rceil + 1$.*

In the full version [3], using the theorem above, we give examples showing that for every $k \geq 2$, there are infinitely many graphs that are determined by their connected k-sets but not by their connected $(k+1)$-sets.

We next show that a threshold near $n/2$ that we saw above for trees, holds for almost every n-vertex graph. The Erdős-Renyi random graph $G \sim G(n, \frac{1}{2})$ has n vertices and each edge is present with probability $\frac{1}{2}$, independently of the other edges. This yields the uniform distribution over the collection of (labelled) graphs on n vertices. If something holds for the random graph with high probability (that is, with a probability that tends to 1 as $n \to \infty$), then we say that it holds for *almost every graph*. The random graph is also interesting since it is often use to deduce the existence of extremal graphs for many problems. More information can be found in e.g. [4,14,18].

We say an n-vertex graph $G = (V, E)$ is *random-like* if the following three properties hold (with log of base 2).

1. For every vertex $v \in V$,
$$n/2 - 3\sqrt{n \log n} \leq |N(v)| \leq n/2 + 3\sqrt{n \log n}.$$
2. For every pair of distinct vertices $v, w \in V$,
$$n/4 - 3\sqrt{n \log n} \leq |N(v) \cap (V \setminus N(w))| \leq n/4 + 3\sqrt{n \log n}.$$
3. There are no disjoint subsets $A, B \subseteq V$ with $|A|, |B| \geq 2 \log n$ such that there are no edges between a vertex in A and a vertex in B.

Lemma 3. *For $G \sim G(n, \frac{1}{2})$, with high probability G is random-like.*

The claimed properties of the random graph are well-known, nonetheless we added a proof in full version for the convenience of the reader [3].

Theorem 6. *For all sufficiently large n, any n-vertex graph G that is random-like can be reconstructed from the set of connected k-sets for $2 \leq k \leq \frac{1}{2}n - 4\sqrt{n \log n}$ in time $O(n^{k+1})$. On the other hand, $G[S]$ is connected for all subsets S of size at least $\frac{1}{2}n + 4\sqrt{n \log n}$.*

In particular, for almost every graph (combining Lemma 3 and Theorem 6), the connectivity of k-tuples for $k \geq \frac{1}{2}n + 4\sqrt{n \log n}$ gives no information whatsoever, whereas for $k \leq \frac{1}{2}n - 4\sqrt{n \log n}$ it completely determines the graph.

Proof (of Theorem 6). Let K be the set of connected k-sets.

We first prove the second part of the statement. Let $k \geq \frac{1}{2}n + 4\sqrt{n \log n}$ be an integer and let S be a subset of V of size at least k. Consider two vertices $u, v \in S$. We will prove that u and v are in the same connected component of $G[S]$. By the first random-like property, there are at most $\frac{1}{2}n + 3\sqrt{n \log n}$ vertices non-adjacent to u, which implies that u has at least $\sqrt{n \log n} - 1 \geq 2 \log n$ neighbours in S. Note that, for the same reason, this is also true for v. Therefore, we can apply the third random-like property on $A = N(u) \cap S$ and $B = N(v) \cap S$ to ensure that there is an edge between the two sets. We conclude that there must exist a path from u to v.

Let us now prove the first part of the statement. Let $2 \leq k \leq \lfloor \frac{1}{2}n - 4\sqrt{n \log n} \rfloor$. Let $u \in V(G)$. We claim that the set of vertices $V \setminus N[v]$ that are not adjacent or equal to u, is the largest set S such that $G[S]$ is connected and $G[S \cup \{u\}]$ is not. Note that the two conditions directly imply that $S \subseteq V \setminus N[u]$. To prove equality, it is sufficient to prove that $G[V \setminus N[u]]$ is connected. Consider two vertices $v, w \in G[V \setminus N[u]]$. By the second random-like property, the sets $A = N(v) \cap (V \setminus N[u])$ and $B = N(w) \cap (V \setminus N[u])$ have size at least $n/4 - 3\sqrt{n \log n}$. By the third property of random-like, there is therefore an edge between A and B.

This proves that there is a path between v and w using vertices in $V \setminus N[u]$, and so $G[V \setminus N[v]]$ is connected.

For each vertex u, the set of vertices it is not adjacent to can now be found by finding the largest set S such that $G[S]$ is connected and $G[S \cup \{u\}]$ is not. Since any such S is a subset of $V \setminus N[u]$, there is a unique maximal (and unique maximum) such S. We now give the $O(n^{k+1})$ time algorithm for this.

We begin by constructing a data structure like the deterministic version described by Fredman et al. [13], which allows us to query the required k-sets in $O(1)$. This takes deterministic time of $O(|K| \log |K|) = O(n^k k \log n) = O(n^{k+1})$. For a vertex v, we give an algorithm to reconstruct the neighbourhood of v in time $O(n^{k+1})$.

1. We first run over the subsets S of size k until we find one for which $G[S]$ is connected but $G[S \cup \{v\}]$ is not. This can be done in time $O(kn^k)$: if $G[S]$ is connected, then $G[S \cup \{v\}]$ is disconnected if and only if $G[S \setminus \{s\} \cup \{v\}]$ is disconnected for all $s \in S$.
2. For each vertex $w \in V \setminus (S \cup \{v\})$, we check whether there is a subset $U \subseteq S$ of size $k-1$ for which $U \cup \{w\}$ is connected, and whether for each subset $U' \subseteq S$ of size $k-2$, $U' \cup \{w, v\}$ is not connected. If both are true for the vertex w, then $G[S \cup \{w\}]$ is connected and $G[S \cup \{w, v\}]$ is not connected, so we add w to S and repeat this step.
3. If no vertex can be added anymore, we stop and output $V \setminus (S \cup \{v\})$ as the set of neighbours of v.

We repeat step 2 at most n times, and each time we try at most n vertices as potential w and run over subsets of size at most $k-1$. Hence, this part runs in time $O(n^{k+1})$. We repeat the algorithm above n times (once per vertex) in order to reconstruct all edges. □

We prove the following analogue to Theorem 2 in the full version [3].

Theorem 7. *Let $k \geq 4$ be an integer. Every graph on at least $2k-1$ vertices that is known to have no cycles of length at most k is determined by its connected k-sets.*

7 Conclusion

We have presented a new model of uncertainty in graphs, in which we only receive all triples of vertices that form a connected induced subgraph. In a way, this is the simplest model of combinatorial indeterminacy in graphs. We have studied some basic properties of this model, and provided an algorithm for finding a graph consistent with the given indeterminacies. We also proved that trees, triangle-free graphs and various other families of graphs are determined by the connected triples, although we need to know the family the sought graph belongs to. In order to obtain a full characterisation, it is natural to put conditions on the way a triangle may connect to the rest of the graph, for instance, it is not too difficult to recognise that a, b, c induces a triangle if at least two of a, b, c have private neighbour, whereas it is impossible to distinguish whether a, b, c induce a triangle or a path if all three vertices have the same neighbours outside of the triangle. We leave this open for future work.

Similar to what has been done for graph reconstruction (see e.g. [23]), another natural direction is to loosen the objective of reconstruction, and to see if rather than determining the (labelled) graph, we can recover some graph property such as the number of edges or the diameter. A natural question is also how many connected triples are required (when given a 'subcollection', similar to [6,16,26], or when we may perform adaptive queries, as in [19]).

We gave various results in an extension of our model to larger k-sets, including trees and random graphs. There are several other logical extensions to the concept of reconstructing a graph from connected triples. We could define a (k, ℓ)-representation T to contain all k-sets that are connected and contain at least ℓ edges. The definition of connected triples would then be a $(3, 2)$-representation. Note that in this case some vertices may not appear in T, or T might even be empty altogether (e.g. for trees when $\ell \geq k$). Another natural extension would be to specify the edge count for each k-set, but this gives too much information even when $k = n-2$: the existence of any edge $\{u, v\}$ can be determined from the number of edges among vertices in the four sets $V, V \setminus \{u\}, V \setminus \{v\}$ and $V \setminus \{u, v\}$.

Some interesting algorithmic questions remain open as well. In particular, we presented an efficient algorithm to specify whether a collection of connected k-sets, for $k = 3$ uniquely determines a graph, but do not know how to solve this efficiently for larger values of k. Is the following decision problem solvable in polynomial time: given a graph G and an integer k, is G determined by its collection of k-tuples? We note that when k equals 4, membership in coNP is clear (just give another graph with the same connected 4-sets) whereas even NP-membership is unclear.

Finally, a natural question is whether the running time of $O(n \cdot |T|)$ for finding a consistent graph with a set of connected triples (Lemma 1) can be improved to $O(|T|)$ (or expected time $O(|T|)$).

References

1. Ahmed, M., Wenk, C.: Constructing street networks from GPS trajectories. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 60–71. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33090-2_7
2. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Inf. Process. Lett. **8**(3), 121–123 (1979)
3. Bastide, P., et al.: Reconstructing graphs from connected triples (2023)
4. Bollobás, B.: Random graphs. In: Modern Graph Theory. GTM, vol. 184, pp. 215–252. Springer, New York (1998). https://doi.org/10.1007/978-1-4612-0619-4_7
5. Bondy, J.A., Hemminger, R.L.: Graph reconstruction - a survey. J. Graph Theory **1**(3), 227–268 (1977)
6. Bowler, A., Brown, P., Fenner, T.: Families of pairs of graphs with a large number of common cards. J. Graph Theory **63**(2), 146–163 (2010)

7. Brandes, U., Cornelsen, S.: Phylogenetic graph models beyond trees. Discrete Appl. Math. **157**(10), 2361–2369 (2009)
8. Cameron, P.J., Martins, C.: A theorem on reconstruction of random graphs. Comb. Probab. Comput. **2**(1), 1–9 (1993)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, pp. 197–200. MIT press (2022)
10. Dey, T.K., Wang, J., Wang, Y.: Graph reconstruction by discrete Morse theory. In: Proceedings of the 34th International Symposium on Computational Geometry. Leibniz International Proceedings in Informatics (LIPIcs), vol. 99, pp. 31:1–31:15 (2018)
11. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**(4), 691–703 (1976)
12. Feder, T.: Network flow and 2-satisfiability. Algorithmica **11**(3), 291–319 (1994)
13. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a sparse table with 0(1) worst case access time. J. ACM **31**(3), 538–544 (1984)
14. Frieze, A., Karonski, M.: Introduction to Random Graphs. Cambridge University Press, New York (2015)
15. Giles, W.B.: The reconstruction of outerplanar graphs. J. Comb. Theory Ser. B **16**(3), 215–226 (1974)
16. Groenland, C., Guggiari, H., Scott, A.: Size reconstructibility of graphs. J. Graph Theory **96**(2), 326–337 (2021)
17. Harary, F.: A survey of the reconstruction conjecture. In: Bari, R.A., Harary, F. (eds.) Graphs and Combinatorics. LNCS, vol. 406, pp. 18–28. Springer, Berlin (1974). https://doi.org/10.1007/BFb0066431
18. Janson, S., Rucinski, A., Luczak, T.: Random Graphs. John Wiley & Sons, Hoboken (2011)
19. Kannan, S., Mathieu, C., Zhou, H.: Graph reconstruction and verification. ACM Trans. Algorithms **14**(4), 1–30 (2018)
20. Kassiano, V., Gounaris, A., Papadopoulos, A.N., Tsichlas, K.: Mining uncertain graphs: an overview. In: Sellis, T., Oikonomou, K. (eds.) ALGOCLOUD 2016. LNCS, vol. 10230, pp. 87–116. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57045-7_6
21. Kelly, P.J.: On Isometric Transformations. Ph.D. thesis, University of Wisconsin (1942)
22. Lauri, J.: The reconstruction of maximal planar graphs. J. Combi. Theory Ser. B **30**(2), 196–214 (1981)
23. Lauri, J., Scapellato, R.: Topics in Graph Automorphisms and Reconstruction. Cambridge University Press, Cambridge (2016)
24. Mordeson, J.N., Peng, C.S.: Operations on fuzzy graphs. Inf. Sci. **79**(3), 159–170 (1994)
25. Mossel, E., Ross, N.: Shotgun assembly of labeled graphs. IEEE Trans. Netw. Sci. Eng. **6**(2), 145–157 (2017)
26. Myrvold, W.: The degree sequence is reconstructible from $n-1$ cards. Discrete Math. **102**(2), 187–196 (1992)
27. Rosenfeld, A.: Fuzzy graphs. In: Zadeh, L.A., Fu, K.S., Tanaka, K., Shimura, M. (eds.) Fuzzy Sets and their Applications to Cognitive and Decision Processes, pp. 77–95. Elsevier (1975)
28. Tutte, W.: All the king's horses. A guide to reconstruction. Graph Theory Relat. Top., 15–33 (1979)
29. Ulam, S.M.: A Collection of Mathematical Problems, Interscience Tracts in Pure and Applied Mathematics, vol. 8. Interscience Publishers (1960)