Grid Proximity Graphs: LOGs, GIGs and GIRLs

River Allen* Laurie Heyer† Rahnuma Islam Nishat* Sue Whitesides*

Abstract

This paper discusses three types of proximity graphs called LOGs, GIGs and GIRLs, defined on unit grids. We show that it can be decided in linear time whether a LOG graph is a GIG graph. We also show that it is NP-complete to recognize LOGs and GIGs, and explore the relationship between these graph classes and their properties. Enumeration results and open problems are also presented.

1 Introduction

Consider an $m \times n$ unit grid and define on this grid a *limited outdegree grid* directed graph, or LOG graph, as follows: the vertices are the mn vertices of the unit grid, the underlying edges are a subset of the unit grid edges such that each edge has unit length and each vertex has outdegree at most one. In other words, each vertex can point to at most one of its neighbors in the underlying grid. See Figure 1 for an example of a 3×4 LOG graph.

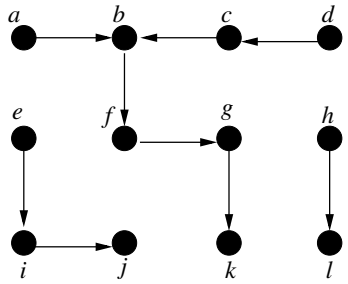

Figure 1: A LOG graph with 12 vertices.

Let $G = (V, E)$ be an $m \times n$ LOG graph with vertex set V and edge set E. For each vertex u in V, let $N(u)$ denote the set of the vertices of G that have unit distance from u in the underlying grid. We call $N(u)$ the *potential neighbors* of u.

One way to obtain a LOG graph is to make a one-to-one assignment of the labels $1, 2, \ldots, mn$ to the mn vertices of the grid, then include a directed edge (u, v) if the label at v is greater than the label of u and the greatest among all the labels of $N(u)$. This construction motivates the following definition, where we denote the label of vertex u by $L(u)$.

Definition 1 *A* Greatest Increase Grid *directed graph or a GIG graph is a LOG graph in which the vertices can be labeled with distinct integers* $1, 2, \ldots, mn$ *such that* $(u, v) \in E$ *if and only if* $v \in N(u)$, $L(v) > L(u)$ *and* $L(v) > L(w)$ *for all* $w \in N(u), w \neq v$.

See Figure 2 for an example construction of a 3×3 GIG graph. In [2], a GIG graph is interpreted as a representation of a discrete 3-dimensional search space in which the vertices of G are the states, and $L(u)$ is the utility of the state. A hill-climbing algorithm with initial state u would succeed in finding the global maximum state if and only if there is a directed path from u to that state.

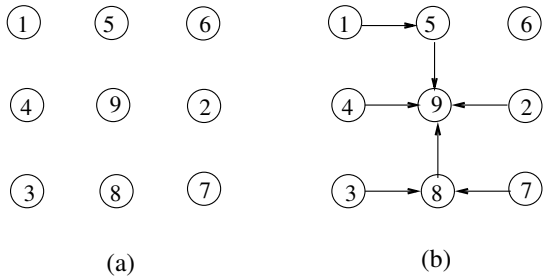

Figure 2: (a) A random labeling of the vertices of a 3×3 grid, and (b) a GIG graph generated from the labeling.

In this paper, we present an alternative interpretation of a GIG as a representation of a folded map. An $m \times n$ *map* is a rectangular piece of paper that is divided into mn unit squares by a $m \times n$ square grid on the paper. The edges of the grid on the paper (not on the boundary of the paper) are called *creases*. A map can be folded only along the creases. Figure 3(a) shows a 2×3 map and it has seven creases, three horizontal and four vertical.

A famous open problem in map folding posed by Jack Edmonds asks whether it can be decided in polynomial time whether a map can be folded into a unit square or not [4]. Suppose that a map can be folded into a unit square. Then in such a folded state, there is a linear ordering of the faces of the map from top to bottom [6] as shown in Figure 3(b).

*Department of Computer Science, University of Victoria, BC, Canada, riverallen@gmail.com,rnishat@uvic.ca, sue@uvic.ca

†Mathematics Department, Davidson College, NC, USA, laheyer@davidson.edu

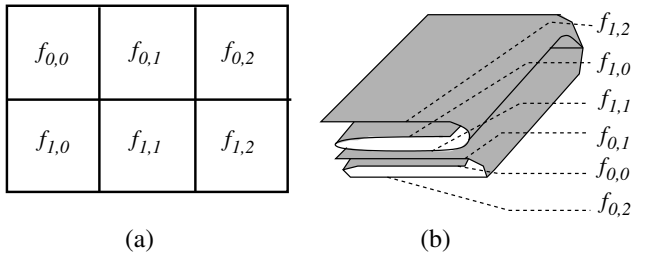

Figure 3: (a) A 2×3 map with the faces labeled, (b) the map folded on a unit square.

We represent each face of the map as a fixed vertex of a GIG G depending on the row and column of the face as shown in Figure 4(a). We then label the vertices of the GIG G as shown in Figure 4(b), where each vertex of G gets a label depending on its height from the plane on which the faces are stacked, and add the directed edges according to the definition of a GIG graph. A directed edge (u, v) in this graph G denotes that u is below v in the stack of faces and v is the topmost among all the neighbors of u.

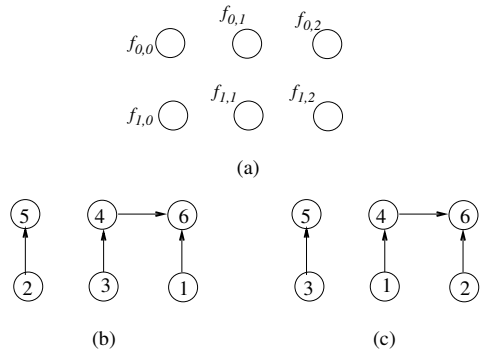

Figure 4: (a) The faces of the map in Figure 3(a) shown as vertices of G, (b) labeling of the vertices G according to the ordering in Figure 3(b), (c) another labeling of the same GIG G which causes the paper to self-intersect.

Now, suppose a GIG G is given that represents a possible linear ordering of the faces of a map. Depending on the labeling of the vertices of G, we might or might not get a linear ordering of the faces that is valid. Figures 4(b) and (c) show two different labelings of the same GIG associated with a 2×3 map. Although the labeling of Figure 4(b) gives the valid linear ordering for the folded state in Figure 3(b), the ordering from Figure 4(c) cannot be obtained.

Figure 5(a) and (b) show a 2×4 map and a GIG representing a possible linear ordering of its faces. In this GIG, the faces $f_{0,2}$ and $f_{1,3}$ must receive the labels 7 and 8, respectively. For this reason, any labeling of this GIG gives an ordering of the faces of the map such that the paper would have to self-intersect.

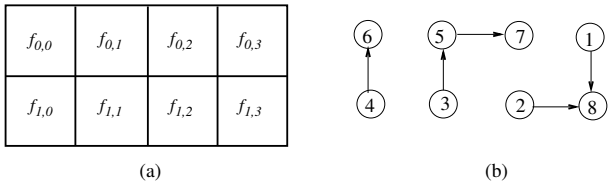

Figure 5: (a) A 2×4 map with the faces labeled, (b) a GIG that gives no linear ordering.

The rest of the paper is organized as follows. Section 2 gives an algorithm to decide whether a given LOG graph is a GIG graph. Section 3 presents an algorithm to generate all possible labelings of the vertices of a GIG graph. In Section 4, we show that it is NP-complete to decide whether a graph is a LOG or a GIG. Section 5 and 6 give generalizations and variations. Section 7 concludes the paper.

2 Recognizing GIG graphs

Since the set of GIG graphs on an $m \times n$ grid is a proper subset of the set of LOG graphs on the same size grid (e.g., edge-free LOG graphs are not GIG graphs), we are interested in deciding whether a given LOG graph is a GIG graph. Here, we give a polynomial time algorithm to solve this decision problem. The algorithm rests on the construction of a new graph that represents a set of inequalities implied by the edges in G. We define this new graph as follows:

Definition 2 *The* augmented graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *of a LOG graph* $G = (V, E)$ *is a directed graph that satisfies the following conditions.*

(a) $\mathcal{V} = V$ *and* $E \subset \mathcal{E}$.

(b) If there is an edge from u *to* v *in* G*, then* $\mathcal{G}$ *also has edges from all other potential neighbors of* u *to* v*. In other words,* $(w, v) \in \mathcal{E}$ *for each* $w \in N(u), w \neq v$.

(c) If the outdegree of u *is 0 in* G*, then for every* $w \in N(u)$*, there must be an edge* (w, u) *in* $\mathcal{E}$.

Figure 6 shows an example of the augmented graph of GIG graph on a 3×3 grid.

If a $\mathcal{G}$ is an augmented graph of a LOG graph that is a GIG graph, then $\mathcal{G}$ can be reconstructed from the labeled grid vertices of the GIG. Thus the labeled vertices provide a geometric and compressed representation of $\mathcal{G}$.

Theorem 3 *Let* G *be a LOG graph with* mn *vertices and let* $\mathcal{G}$ *be the augmented graph of* G*. Then* G *is a GIG graph if and only if* $\mathcal{G}$ *is acyclic.*

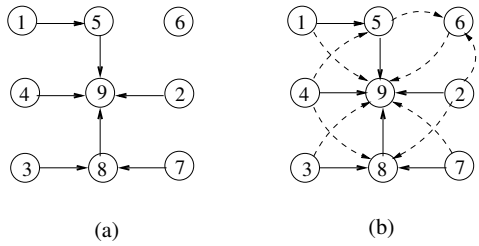

Figure 6: (a) A GIG graph on a 3×3 grid, and (b) its augmented graph.

Proof. If G is a GIG graph then there exists a labeling of the vertices of G such that every vertex points to its biggest potential neighbor that has a bigger label than the vertex itself in the grid embedding of G. Then by the definition of augmented graphs, each directed edge (u, v) in $\mathcal{G}$ denotes that $v > u$. Suppose there is a directed cycle $v_1, v_2, \ldots, v_k, v_1$ in $\mathcal{G}$. Then we get $v_1 < v_2 < \ldots < v_k < v_1$, which is a contradiction. Therefore, $\mathcal{G}$ is acyclic.

We now assume that the augmented graph $\mathcal{G}$ is acyclic. Then we can get a topological sort [3] of the vertices of $\mathcal{G}$ and we assign the resulting labels $1, 2, \ldots, mn$ to the vertices of $\mathcal{G}$. We now show that any of these labelings satisfies the definition of GIG graphs. Let u be a vertex in G. We have to consider two cases:

(a) G contains an outgoing edge (u, v) in G. Then in the augmented graph $\mathcal{G}$, we have an edge (w, v) for each $w \in N(u)$, where $w \neq v$. Thus, all the potential neighbors of u receive smaller index than v.

(b) There is no outgoing edge from u in G. Then all its potential neighbors points to it in $\mathcal{G}$ and therefore they all receive smaller index than v.

Therefore, the labeling of G obtained above satisfies the definition of GIGs and hence G is a GIG graph. □

3 Generating all the Labelings

In this section, we give an algorithm to generate all possible labelings of the vertices of a GIG graph since a GIG graph can have multiple labelings (see Figure 7).

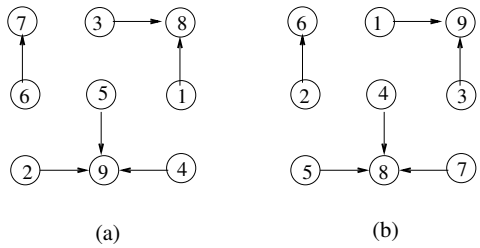

Figure 7: Two labelings of the same GIG graph.

Pruesse and Ruskey [7] gave an output sensitive algorithm to generate all possible linear extensions of a given poset. Let $\mathcal{P}$ be a poset and let $\mathcal{E}(\mathcal{P})$ be the set of all linear extensions of $\mathcal{P}$. Then their algorithm generates all linear extensions in time $O(|\mathcal{E}(\mathcal{P})|)$, which results in constant amortized time. We can use this algorithm to generate all possible labelings of a given GIG graph in constant amortized time. Here we give a sketch of a simpler recursive algorithm which suffices for our purpose.

Let G be the given GIG graph with n vertices and let $\mathcal{G}$ be the augmented graph of G. Then $\mathcal{G}$ must be a directed acyclic graph. Since there are no cycles in G, there must be at least one vertex in G that has no incoming edges. We denote by *sources* the vertices with no incoming edges in G. We now label any of the sources with the least available index from $1, \ldots, n$. Let the source be v and label of the source be $l(v) = i$. We then remove v from G and recurse the procedure for $G \setminus v$, where the least available index is $i + 1$. A pseudocode of our algorithm **LabelGIG** is given below. The initial call is **LabelGIG**$(G, 1)$.

Algorithm 1: LabelGIG(G, i)

```
1 S is the set of all the sources in G
2 if G = ∅ then
3 |   print the labeling.
4 |   return.
5 for each v ∈ S do
6 |   l(v) = i
7 |   LabelGIG(G \ v, i + 1)
```

4 Complexity of Embedding a GIG on a Grid

In this section, we show that it is NP-complete to determine whether a given abstract graph is a LOG graph. We also show that the recognition problem remains NP-complete for GIGs.

A formal definition of the LOG recognition problem is given below.

Problem : LOG-Recog

Instance : Two integers $m, n > 0$ and a planar directed graph G with mn vertices such that the maximum degree of G is less than or equal to four and each vertex has outdegree less than or equal to one.

Question : Does G have a plane rectilinear embedding on an $m \times n$ integer grid?

We reduce the 3-Partition problem to LOG-Recog to prove the NP-hardness. The 3-Partition problem is described as follows.

Problem : 3-Partition

Instance : A set of integers $S = \{x_1, x_2, \ldots, x_{3p}\}$, where $p > 0$, and an integer $B > 0$ such that $\sum_{i=1}^{3p} x_i = pB$ and $B/2 > x_i > B/4$, $1 \leq i \leq 3p$.

Question : Can S be partitioned into p disjoint sets such that each set contains exactly three integers that sum up to B?

We use a very similar construction as Dolev *et al.* [5]. We create a *directed frame tree* as shown in Figure 8(b) from the frame tree of Dolev *et al.* shown in Figure 8(a). We choose the degree four vertex r as the root and direct each edge from the child node to the parent node.

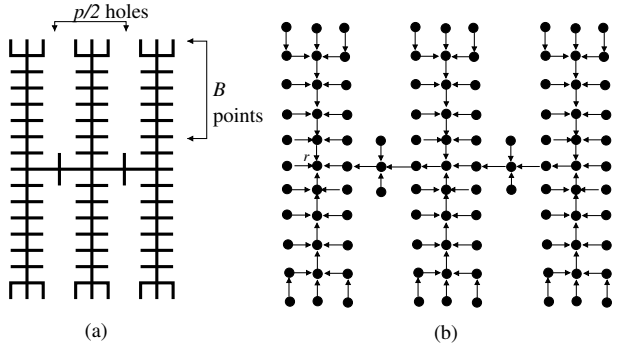

Figure 8: (a) The frame tree used by Dolev *et al.* [5], and (b) the corresponding directed frame tree.

We now prove the NP-completeness of LOG-Recog.

Theorem 4 LOG-Recog *is NP-complete.*

Proof. If a rectilinear embedding of G is given on an $m \times n$ grid, it can be checked in polynomial time whether each edge has unit length. Thus the problem is in NP. To prove that it is NP-hard, take an instance $S = \{x_1, x_2, \ldots, x_{3p}\}$ of 3-Partition, where $\sum_{i=1}^{3p} x_i = pB$ and $B/2 > x_i > B/4$, $1 \leq i \leq 3p$, and construct an instance of LOG-Recog from the instance of 3-Partition as follows.

First assume that p is even. Create a directed frame tree T with $(2p+3)(2B+3) - pB$ vertices as shown in Figure 8(b). Then create a *directed path* of x_i vertices for each of the integers $x_i \in S$, $1 \leq i \leq 3p$. The graph G containing T and all the $3p$ directed paths has $(2p+3)(2B+3)$ vertices. Finally, choose $m = 2p+3$ and $n = 2B+3$. We now show that the given instance of the 3-Partition problem has a solution if and only if G has a rectilinear embedding on an $m \times n$ integer grid.

First assume that the instance of 3-Partition has a solution, so S can be partitioned into p disjoint sets $S_1, S_2, \ldots, S_p$ such that each of the sets has exactly three integers that sum to B. From Lemma 6 of [5], T has only two possible embeddings on an $m \times n$ grid, and in each of the cases there are p holes of B grid points each. Therefore, for each S_j, $1 \leq j \leq p$, lay the paths corresponding to the integers in S_j in one hole and get a rectilinear embedding of G.

Now assume that G has an embedding on the integer grid. Since any embedding of T leaves p holes of B grid points and the number of grid points is equal to the number of vertices in G, each of the holes must contain three paths that have B vertices in total. Take the integers corresponding to the paths in a hole to form a subset. In this way we get a partition of S into p disjoint subsets as required.

Now assume that p is odd. Create a directed frame tree T with $(2(p+1)+3)(2B+3)-(p+1)B$ vertices and the $3p$ directed paths representing $3p$ integers as before. We also create a directed path of B vertices as shown in Figure 9. In this case G contains $(2p+5)(2B+3)$ vertices and hence, we take $m = 2p+5$ and $n = 2B+3$. As in the previous case, it can be proved that the given instance of 3-Partition has a solution if and only if G has a rectilinear embedding on an $m \times n$ integer grid. □

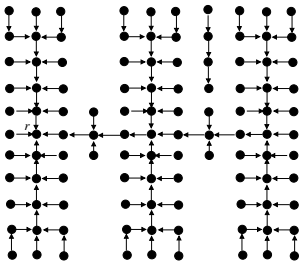

Figure 9: The directed frame tree and the directed path of B vertices when p is odd.

We now define the GIG recognition problem.

Problem : GIG-Recog

Instance : Two integers $m, n > 0$ and a planar directed graph G with mn vertices such that the maximum degree of G is less than or equal to four and each vertex has outdegree less than or equal to one.

Question : Does G have a plane rectilinear embedding on an $m \times n$ integer grid such that the augmented graph is a directed acyclic graph?

Now we prove that GIG-Recog is NP-complete.

Theorem 5 GIG-Recog *is NP-complete.*

Proof. If a rectilinear embedding of G is given on an $m \times n$ grid, it can be checked in polynomial time whether each edge has unit length and whether the augmented graph is a directed acyclic graph . Thus the problem is in NP. To prove that it is NP-hard, reduce the 3-Partition problem to GIG-Recog as in the proof of Theorem 4.

Create a directed frame tree T with $3p(B+3)+4(p-1)+12\times 2 = 3pB+13p+20$ vertices as shown in Figure 10. Choose $m=4$ and $n = p(B+3)+p-1+3\times 2 = pB+4p+5$. On an $m\times n$ grid, T has the unique embedding shown in Figure 10, which creates p holes with $B+3$ grid points in each. Then create a *directed path* of x_i+1 vertices for each of the integers $x_i \in S$, $1 \le i \le 3p$. The graph G containing T and all the $3p$ directed paths has $3pB+13p+20+pB+3p = 4(pB+4p+5)$ vertices in total. As in the proof of Theorem 4, it can be shown that the given instance of 3-PARTITION has a solution if and only if G has a rectilinear embedding on the $m\times n$ integer grid, where each edge on the directed paths points right to left. We now show that the augmented graph obtained from such an embedding of G is acyclic.

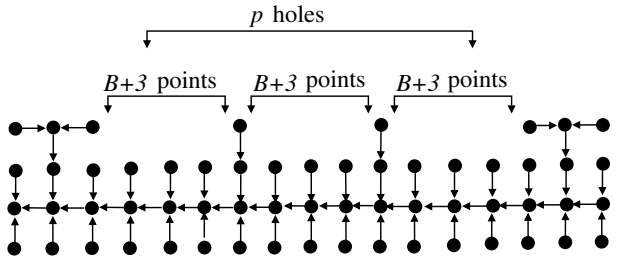

Figure 10: The frame tree for the proof of Theorem 5.

Figure 11(a) shows the augmented graph of the leftmost portion of the directed frame tree T with just one hole, and Figure 11(b) shows the augmented graph when the directed paths representing the integers in a subset of S are laid out such that each edge on the directed paths points right to left. Because of the symmetric structure of the augmented graph, it is easy to see that the augmented graph in Figure 11(b) is acyclic and hence, the augmented graph of such a rectilinear embedding of G on the grid is an acyclic directed graph. □

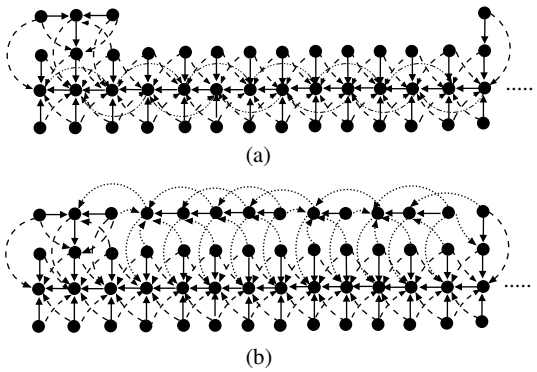

Figure 11: Illustration for the proof of Theorem 5. The dot dashed lines in (a) are not shown in (b).

5 Generalizations of LOGs and GIGs

In this section, we show that the concepts of LOG graphs and GIG graphs extend to $\mathbb{R}^d$, where $d \ge 2$, and also to other kinds of grids than rectangular grids.

Figure 12(a) and (b) show a LOG and a GIG in $\mathbb{R}^3$. Each vertex here has at most 6 potential neighbors and

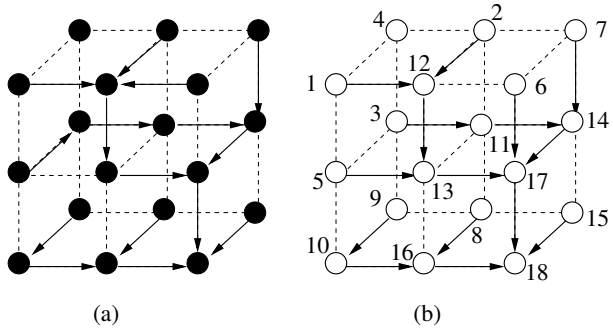

Figure 12: (a) A $3\times 3\times 2$ LOG, (b) a $3\times 3\times 2$ GIG.

outdegree at most one. Therefore, in $\mathbb{R}^d$, each vertex has at most $2d$ potential neighbors. In other words, if we have a directed graph with outdegree at most one and maximum degree Δ, a necessary condition for it to be represented as a LOG in $\mathbb{R}^d$ is that $d \ge \Delta/2$. Figure 13(a) and (b) show examples of LOG graphs on a triangular grid and a hexagonal grid in $\mathbb{R}^2$, respectively. Each vertex of a LOG has at most 6 and 3 potential

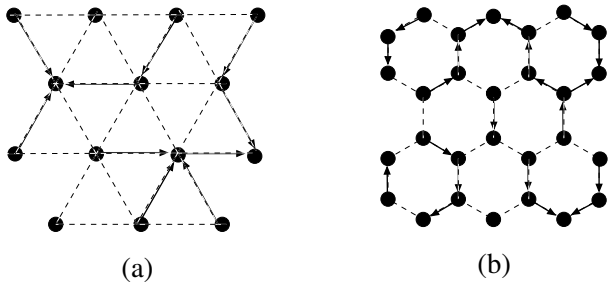

Figure 13: LOGs on (a) a triangular grid, and (b) a hexagonal grid.

neighbors on a triangular and on a hexagonal grid, respectively. This observation raises the following open problem.

Open Problem: Let G be a directed graph with outdegree at most one, maximum degree 6 and lmn vertices. What is the complexity of determining whether it is a LOG graph on a rectangular $(l\times m\times n)$ unit grid in $\mathbb{R}^3$ or a LOG graph on a triangular grid in $\mathbb{R}^2$?

6 Labeling LOGs with Repetition

In this section, we introduce another subclass of LOG graphs: the *Greatest Increase with Repeated Labels Al-*

lowed Grid directed graphs, or GIRL graphs for short.

GIRL graphs are a modification of GIG graphs where we allow repeated use of labels, but at each vertex the labels of its neighbors must be distinct; the label of a vertex may appear among the labels of its neighbors.

Definition 6 *A* Greatest Increase with Repeated Labels Allowed Grid *directed graph is a LOG graph in which the vertices can be labeled with integers* $1, 2, \ldots, mn$ *such that the directed edge* $(u, v) \in E$ *if and only if* $v \in N(u)$, $L(v) > L(u)$ *and* $L(v) = max(\{L(w)|w \in N(u)\})$*; furthermore* $\forall v \in V, \|N(v)\| = \|\{L(w)|w \in N(v)\}\|$, *i.e. for each vertex the labels of its neighbors must be distinct.*

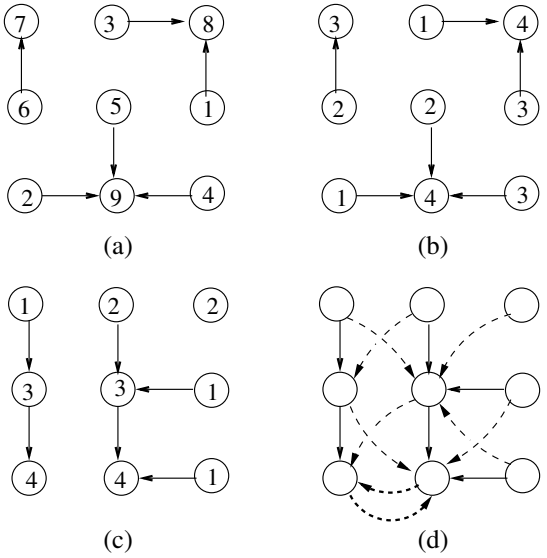

Figure 14: (a) A GIG graph, where each vertex has a unique label, (b) an equivalent GIRL representation, (c) A GIRL graph that is not a GIG graph and (d) the augmented graph of the underlying LOG graph for (c) which has a directed cycle shown in bold dashed lines.

Note that, since we can determine if a LOG graph is a GIG graph, we can determine if a GIRL graph is a GIG graph. For example, the GIRL graph in Figure 14(c) cannot be a GIG graph because its augmented graph contains a directed cycle as shown in Figure 14(d). Also note that there are LOG graphs that are not GIRL graphs: for example, a 3×3 LOG graph without any edges is not a GIRL graph. Thus the inclusions in Figure 15 are strict.

The possiblity of repeated labels leads to several questions:

- Given a LOG graph, is it a GIRL graph?

It is easy to check that Figure 14(b) uses a minimum label set.

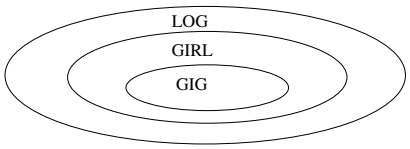

Figure 15: The inclusions are strict.

- Given a GIG graph, what is the minimum set of labels needed to represent it as a GIRL graph?

7 Conclusion

We have studied the LOG graphs and GIG graphs, introduced the augmented graphs of a GIG graph and characterized GIG graphs in terms of their augmented graphs. We show that LOG graph and GIG graph recognition is NP-complete. We have also introduced generalizations of LOG graphs and a significant superclass of GIG graphs called the GIRL graphs. We close with the following question, similar in spirit to graph decomposition problem such as linear arboricity [1].

Open problem: Given any directed acyclic graph G, can we decompose G into (a minimum number of) augmented graphs of GIG or GIRL graphs?

References

[1] Noga Alon. The linear arboricity of graphs. *Israel Journal of Mathematics*, 62(3):311–325, 1988.

[2] Joshua Chester, Linnea Edlin, Jonah Galeota-Sprung, Bradley Isom, Andrew Lantz, Alexander Moore, Virginia Perkins, E. Tucker Whitesides, A. Malcolm Campbell, Todd T. Eckdahl, Laurie J. Heyer, and Jeffrey L. Poet. On counting limited outdegree grid digraphs and greatest increase grid digraphs. Unpublished manuscript, 2012.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[4] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, NY, USA, 2007.

[5] Danny Dolev, Tom Leighton, and Howard Trickey. Planar embedding of planar graphs. In *Advances in Computing Research*, pages 147–161, 1984.

[6] Rahnuma Islam Nishat. Map folding. Master's thesis, University of Victoria, BC, Canada, April 2013.

[7] Gara Pruesse and Frank Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386, 1994.