The Vertex-Exchange Graph: A New Concept for Multi-level Crossing Minimisation

Patrick Healy and Ago Kuusik

Department of Computer Science and Information Systems
University of Limerick
Limerick, IRELAND
{Patrick.Healy, Ago.Kuusik}@ul.ie

Abstract. In this paper we consider the problems of testing a multi-level graph for planarity and laying out a multi-level graph. We introduce a new abstraction that we call a vertex-exchange graph. We demonstrate how this concept can be used to solve these problems by providing clear and simple algorithms for testing a multi-level graph for planarity and laying out a multi-level graph when planar. We also show how the concept can be used to solve other problems relating to multi-level graph layout.

1 Introduction

Multi-level crossing minimisation is a well-known problem in graph drawing. Given a layered graph, the multi-level crossing minimisation (MLCM) problem is to reorder vertices on each level so that the number of edge crossings is minimum. The problem is NP-hard, even when there are only two layers and one layer is fixed [4].

Despite the existence of an effective heuristic solution – the Sugiyama algorithm [9] – little is known about the multi-level crossing minimisation problem itself. In particular, questions like what is a graph's crossing number (or bounds on it), what are the subgraphs that can be embedded without any crossings, or how does the ordering of one level influence edge crossings with adjacant levels, do not have answers. We believe that one of the reasons why these questions are still not answered is the lack of a theoretical mechanism giving a global view of the problem. In this paper we propose such a mechanism, the *vertex-exchange graph*. The vertex-exchange graph provides an understanding of how edge crossings relate to each other, it facilitates calculation of a level planar embedding (if it exists) and multi-level crossing minimisation, and can be used to determine successively tighter lower bounds on the crossing number. It may also provide a method for determining a tight upper bound on the number of crossings in the graph.

A multi-level (layer) graph is one where the vertices are placed on discrete levels and edges are allowed only between vertices of adjacent levels. Traditionally, multi-level graph layout algorithms have had three steps [1]. The first step creates a proper levelling of the graph, the second step finds a permutation

J. Kratochvíl (Ed.): GD'99, LNCS 1731, pp. 205–216, 1999.

of vertices on levels minimising the edge crossings, and the final step balances the layout by adjusting each vertex's x-co-ordinate. The algorithm has many refinements and modifications [1,3].

In this paper, we investigate the second step of the multi-level crossing minimisation – the permuting of vertices on levels. The most popular multi-level crossing minimisation algorithm to date is the Sugiyama algorithm [9]. The algorithm is actually a general framework which solves the multi-level crossing minimisation problem as a series of one-level crossing minimisation problems: levels are successively visited (this is called layer-by-layer sweep) and each level is ordered by some one-level crossing minimisation heuristics. Specific implementations of Sugiyama algorithm differ in three aspects: one-level crossing minimisation heuristics, sweeping directions, and stopping criterion.

The choice of the one-level crossing minimisation heuristic has a great influence on the algorithm's speed and accuracy. The barycenter [9] and median [4] heuristics have been preferred the most although there is experimental evidence [7] that the $O(n \log n)$-time *Split* heuristic [2] is superior to the two previous linear-time algorithms. In their experimental study [7] the authors also compared several one-level techniques in the *two-level* crossing minimisation. An interesting outcome was that the barycenter heuristic gave the best results and it outperformed even the authors' own Integer Linear Programming (ILP) technique. This result suggests that the multi-level crossing minimisation is less understood than one-level crossing minimisation.

On the other hand, there may exist good alternatives to the layer-by-layer sweep algorithms. One of the already existing alternatives is an ILP approach by Jünger *et al.* [6]. There exists also an evolutionary algorithmic approach by Utech *et al.*[10] that, naturally, cannot be considered to solve the crossing problem level by level. However it combines a solution to this problem with a solution to the previous step (levelling) so we do not consider it to be a solution to the crossing minimisation problem *per se*.

In the following section we introduce the vertex-exchange graph and use it to derive sufficient conditions for level-planarity of a graph. In Sections 3 and 4, respectively, we present algorithms for planarity testing of multi-level graphs and layout of planar multi-level graphs. We then show a further application by using it in an ILP formulation to find the embedding of a graph with the minimum number of crossings. Section 6 concludes the paper.

2 The Vertex-Exchange Graph

As our paper deals with graphs already having a proper levelling, we define a *proper level graph* formally.

Definition 1. *A proper level graph is a graph* $G = (V, E)$*, with vertex set* $V = V_1 \cup V_2 \cup \ldots \cup V_p$*,* $V_i \cap V_j = \emptyset$*,* $i \neq j$*, and edge set* $E = E_1 \cup E_2 \cup \ldots \cup E_{p-1}$*,* $E_i \subseteq V_i \times V_{i+1}$*.*

In what follows, when we refer to a level graph we assume that it is a proper level graph in addition.

We introduce the notation $\langle v, w\rangle$ for an *unordered* pair of same-level vertices v and w: $\langle v, w\rangle \equiv \langle w, v\rangle$. Analogously, for two edges $e, f \in E_r$, $\langle e, f\rangle$ denotes a pair of edges. In contrast, we denote an *ordered* pair of same-level vertices v and w by $[v, w]$, and $[v, w] \not\equiv [w, v]$

2.1 Definition and Properties

We now define the vertex-exchange graph of $G = (V, E)$ as follows.

Definition 2. *The vertex-exchange graph of a level graph $G = (V, E)$ is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_p$, where $\mathcal{V}_r = \{\langle v, w\rangle \mid v, w \in V_r\}$, and the edge set $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_{p-1}$, where $\mathcal{E}_r = \{\langle e, f\rangle \mid e, f \in E_r,\ e = (t, u),\ f = (w, v),\ \langle t, w\rangle \in \mathcal{V}_r,\ \langle u, v\rangle \in \mathcal{V}_{r+1}\}$.*

Informally, the vertices of the vertex-exchange graph are all distinct pairs of same-level vertices of the original level graph. Two vertices of a vertex-exchange graph are connected by an edge whenever the corresponding vertices of the original graph are connected by non-adjacent edges. Examples of vertex-exchange graphs are shown in Figure 1.

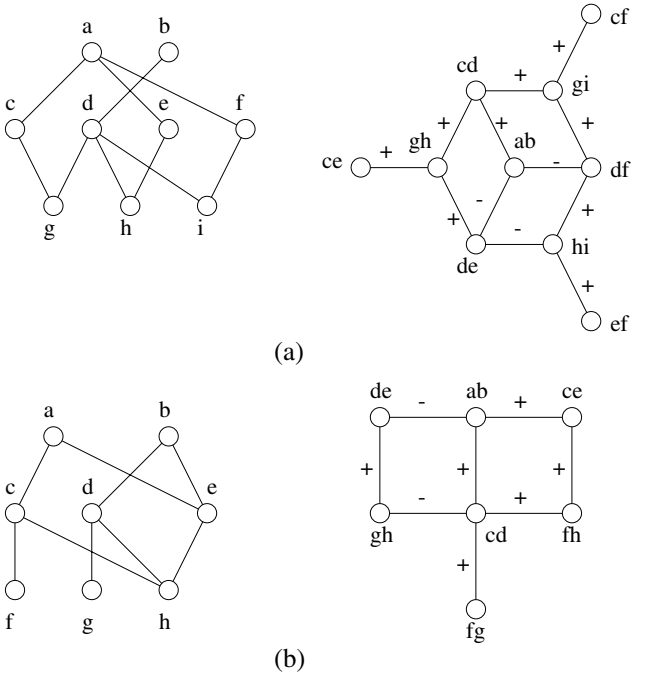

Fig. 1. Examples of vertex-exchange graphs

From the definition of the vertex-exchange graph, we can immediately state some simple properties.

Property 1. If the original graph is a p-level graph then its vertex-exchange graph is also a p-level graph.

From this it follows that the vertex-exchange graph is *bipartite*.

Property 2. The vertex-exchange graph does not contain self-loops. Between two vertices $v = \langle v_1, v_2 \rangle$ and $w = \langle w_1, w_2 \rangle$ there can exist at most two edges. If there exists two edges, then $\{v_1, v_2, w_1, w_2\}$ is the vertex set of a $K_{2,2}$ subgraph. The vertex-exchange graph of a $K_{2,2}$-free level graph is simple.

Given an embedding $\pi : V \to \mathbb{Z}$ of the original graph, we represent it as a labelling of the vertex-exchange graph as follows. For each vertex $\langle v, w \rangle$ of the vertex-exchange graph, we assign an ordered pair of vertices $ordering(\langle v, w \rangle) = [v, w]$, if $\pi(v) < \pi(w)$, and $ordering(\langle v, w \rangle) = [w, v]$ otherwise. Also, we label each edge $\langle e, f \rangle$ by $label(\langle e, f \rangle) =$ '-', if e and f cross in embedding π, otherwise by $label(\langle e, f \rangle) =$ '+'.

Suppose we have a vertex-exchange graph, and we wish to determine which labellings are valid. It turns out that not all labellings correspond to feasible embeddings of the original level graph. There are the following constraints:

3-cycle — Vertex orderings must correspond to a feasible linear ordering, e.g., for vertices u, v, and w of the original graph, the configuration of vertex orderings $[u, v]$, $[v, w]$, and $[w, u]$ is invalid, because it signifies a "cyclic" linear ordering $\pi(u) < \pi(v) < \pi(w) < \pi(u)$.

vertex-edge — Edge labels and vertex orderings must be consistent. If there are edges (v_1, w_1) and (v_2, w_2) in the original graph then the vertex-exchange graph's edge $(\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle)$ must have label '+', if the vertex orderings are $[v_1, v_2]$ and $[w_1, w_2]$ or $[v_2, v_1]$ and $[w_2, w_1]$; and by '-' for the 2 other possible vertex orderings.

min-minus — If a level graph is level non-planar then every labelling of its vertex-exchange graph must contain some edges labelled by '-'.

From constraint **min-minus** it follows that if there is a vertex exchange graph with all edges labelled by '+' then the input graph is level planar. We examine now the constraint **min-minus** more closely to determine where unavoidable '-'-edges may occur.

2.2 Odd-Labelled Cycles

First, we describe formally when two vertices of a vertex-exchange graph are *adjacent*. A *path*, a *cycle* and a *connected component* can be defined similarly.

Property 3. Two vertices $v = \langle v_1, v_2 \rangle$ and $w = \langle w_1, w_2 \rangle$ of vertex-exchange graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are adjacent if vertices v_1 and v_2 are on the same level of G and w_1 and w_2 are on the same level of G and the levels are adjacent and there exist either edges (v_1, w_1) and (v_2, w_2) or (v_1, w_2) and (v_2, w_1) in the input graph.

Examples of even- and odd-labelled cycles can be found in Figure 1. The cycle (ab, cd, gi, df) in Figure 1(a) is odd labelled, but the cycle (ab, cd, gh, de) in Figure 1(b) is even-labelled.

To understand the importance of odd-labelled cycles we will describe here briefly how a level planar embedding can be calculated by an algorithm based on a depth-first search of the labelling of $\mathcal{G}$ of an initial embedding, π, of G. Further details of the level planarity testing and level planar embedding algorithm are given in Sections 3 and 4, respectively.

Firstly, we mark all the vertices of $\mathcal{G}$ as "unvisited." We start from some vertex v of $\mathcal{G}$ and mark it "unchanged" and maintain ordering $[w_1, w_2]$. Next, we pick an adjacent vertex w of v; we keep the ordering $ordering(w) = [w_1, w_2]$ and mark w "unchanged" if (v, w) is a '+'-edge. Otherwise we reverse $[w_1, w_2]$ and mark w "reversed." We repeat the procedure for all the "unvisited" adjacent vertices, u, of w, reversing the vertex ordering of u if (w, u) is labelled '-' and maintaining the ordering if the edge is labelled '+'.

For instance, applying this algorithm for vertex-exchange graph in Figure 1(b), starting from vertex fg, would result in de and gh being marked "reversed." This corresponds exactly to a level planar embedding.

It is not difficult to see that if a vertex-exchange graph contains an odd-labelled cycle then conflicting markings of some vertex of the odd-labelled cycle occur because the two paths along the cycle impose different markings. For example, if we start from vertex ab in Figure 1(a), then gi would get different marking by paths (ab, cd, gi) and (ab, df, gi). The next theorem shows that the odd-labelled cycles in the vertex-exchange graph are the *only* causes of level non-planarity.

Theorem 1. *A level graph G is level planar if and only if the vertex-exchange graph $\mathcal{G}$ of G does not contain any odd-labelled cycles.*

Proof. Level planar $\Rightarrow$ no odd labelled cycles. Every level planar graph has a level planar embedding. The labelling of the vertex-exchange graph $\mathcal{G}$ corresponding to a level planar embedding has all edge labels '+', which means all cycles of $\mathcal{G}$ are trivially even-labelled.

No odd-labelled cycles $\Rightarrow$ level planar. Clearly, if $\mathcal{G}$ does not contain any odd-labelled cycles, it is possible to reorder vertex pairs without any conflicts. Also, due to the way the algorithm works, it is guaranteed that edge and vertex labels match. What needs to be shown, is that the calculated vertex labels do not cause any three-cycle conflicts.

Let us assume that we have a sequence of vertices a, b, and c on some level of the input graph G. To have a three-cycle violation, there must be an even-labelled path $(\langle a, b\rangle, \ldots, \langle b, c\rangle)$ and an odd-labelled path (paths) $(\langle a, b\rangle, \ldots, \langle a, c\rangle)$ and (or) $(\langle b, c\rangle, \ldots, \langle a, c\rangle)$ in the vertex-exchange graph. Next, let x be the vertex of the even-labelled path corresponding to the highest or lowest vertex pair $\langle x_1, x_2\rangle$ of the input graph. Let y and $\langle y_1, y_2\rangle$ be those of an odd-labelled path. Let $(\langle b, c\rangle, \ldots, y)$ be an odd-labelled path and $(y, \ldots, \langle a, c\rangle)$ be an even-labelled path. From the vertex-exchange graph we construct the input graph which will be level non-planar.

The situation where $(\langle b, c\rangle, \ldots, y)$ is the even-labelled path and $(y, \ldots, \langle a, c\rangle)$ is the odd-labelled path can be proved in a similar fashion and likewise for the cases where $x \equiv y$ or the pairs share only one vertex such as $x_1 \equiv y_2$.

It is worthwhile noting that an alternative proof method of Theorem 1 is to show that the vertex-exchange graphs of each type of minimal level non-planar subgraphs [5] possess at least one odd-labelled cycle. (A minimal level non-planar graph $G = (V, E)$ is a non-planar graph such that $G' = (V, E \setminus \{e\}), \forall e \in E$ is planar.)

In fact, minimal level planar subgraphs are directly related to odd-labelled cycles, as shown by the following theorem.

Theorem 2. *Let $\mathcal{C}$ be an odd-labelled cycle in a vertex-exchange graph and C be the subset of edges of the input graph which are mapped to the edges of $\mathcal{C}$. Then there exists a subset of edges $F \subseteq C$ which is the edge set of a minimal level non-planar subgraph.*

Proof. Consider a graph G which includes exactly one level minimal non-planar subgraph, H. Then, from Theorem 1, the vertex-exchange graph $\mathcal{G}$ of G has at least one odd-labelled cycle. By the removal of any edge from H, the graph G becomes level planar and, therefore, the odd labelled cycle disappears from the vertex-exchange graph $\mathcal{G}$. Hence, every edge of H maps to some edge of the odd-labelled cycle $\mathcal{C}$ of $\mathcal{G}$.

As an illustration of this theorem, consider cycle (cd, gi, df, hi, de, gh) in Figure 1(a). It corresponds to the subgraph with vertex set $\{c, d, e, f, g, h, i\}$ of the original graph. This subgraph is a 2-level minimal non-planar subgraph called a double-claw.

3 Level Planarity Testing by the Vertex-Exchange Graph

The basic idea of the level planarity testing algorithm was given in the previous section. The algorithm consists of calling the depth first search routine (Algorithm 1) for each connected component of a vertex-exchange graph. If each component search returns 'true' then the whole vertex-exchange graph has no odd-labelled cycles and the input graph is level planar.

We have used the following encoding for vertex labels. Each vertex v of the vertex-exchange graph has an attribute $value(v)$ which signifies the ordering of the corresponding vertices of the input graph. Three values are possible for $value(v)$: *unknown* for not visited, *true* for the original ordering, and $false$ for reverse ordering. Note that we do not actually change any *edge* labels, because they will finally be all '+' for a level planar graph anyway.

Using the depth first search routine above we can achieve a bound of $O(|V|^2)$ on the running time. Naively, the running time of the algorithm for an arbitrary input graph, is $O(|\mathcal{E}|) = O(|E|^2)$ – since the depth-first search visits each edge exactly once. However, it would be foolish to apply the algorithm to arbitrary

Algorithm 1 *LevelPlanarityDFS*$(\mathcal{G}, v, b)$

```
 1: if value(v) = unknown then
 2:   value(v) = b
 3:   for all vertices w adjacent to v in G do
 4:     if label((v, w)) = '+' then
 5:       result = LevelPlanarityDFS(G, w, b)
 6:     else
 7:       result = LevelPlanarityDFS(G, w, ¬b)
 8:     end if
 9:     if result = false then
10:       return false
11:     end if
12:   end for
13: else if value(v) ≠ b then
14:   return false
15: end if
16: return true
```

graphs because graphs with $|E| > 2|V| - 4$ cannot be planar by a corollary of Euler's formula for bipartite graphs. A vertex-exchange graph is bipartite and planarity is a necessary condition for level planarity.

Consequently, when applying this algorithm, it makes sense to perform the simple check that $|E| \leq 2|V| - 4$ first.

Asymptotically, this algorithm does not compete with an efficient PQ-tree-based linear-time algorithm by Leipert [8]. However, our algorithm has a good trade-off between efficiency and conceptual simplicity. It can be understood and implemented with reasonably small effort yet it can test level planarity of a 200-vertex graph in no more than 2 seconds[1]. So, we consider our algorithm preferable in situations where the graphs are not very big and a PQ-tree library is not available.

4 Layout Calculation of a Level Planar Graph

In addition to level planarity testing, Algorithm 1 can also be used to calculate the layout of a level planar graph. After the completion of *LevelPlanarityDFS*, all the vertex *value*s are either 'true' or 'false'. The embedding can then be found from these values by using a sorting algorithm.

However, the requirement that the 3-cycle constraints be satisfied is a complication. As we have shown, 3-cycle constraints are automatically satisfied in a connected component of a vertex-exchange graph. The 3-cycle constraints formed by the vertices belonging to different connected components need extra care. Figure 2 provides an example graph where the 3-cycles cause a problem.

[1] All experimental work was carried out on a 300MHz DEC AlphaStation.

For instance, if we assign vertex labels as follows:

$$value(\langle a, b\rangle) = true \Rightarrow value(\langle d, f\rangle) = false$$
$$value(\langle d, e\rangle) = true$$
$$value(\langle e, f\rangle) = true$$

then we get a 3-cycle violation since the latter two assignments imply $value(\langle d, f\rangle) = true$.

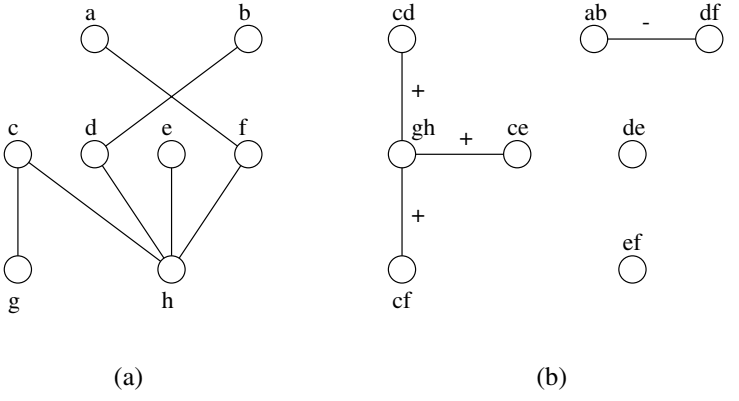

Fig. 2. A graph whose layout calculation has a 3-cycle problem (a); and its vertex-exchange graph (b).

This complication is solved as follows. The vertex-exchange graph is divided into connected components as in the level planarity testing case. Then, a table of all those 3-cycles which possess vertices of different components is constructed. There are 6 fields in the table: identifiers and values of all 3-cycle vertices. For each table record, a mapping from each of the three participating vertices is made. This permits location of an entry of the table of 3-cycles quickly (in almost constant or logarithmic time depending on the implementation of the map).

Another auxiliary data structure is the queue of component assignments. The queue items have fields for vertex identifier, vertex value, and the component the vertex belongs to. The queue is used for identifying the next component which needs processing by *LevelPlanarityDFS*. Initially, the queue is empty.

Now whenever a vertex is assigned a value (line 2 of Algorithm 1), the table is updated accordingly. If the update results in the assignment of a single remaining vertex value constrained by the other two vertex values in a 3-cycle then the remaining vertex identifier with its value and component identifier is inserted into the queue. If a component has been processed by *LevelPlanarityDFS* and the next component has to be selected and there is an item in the queue then the

component corresponding to the queue item is taken as the next one. The first vertex to be assigned and its value is then taken from the queue item instead of taking an arbitrary vertex and value. A minor technical detail is that for each component we have a boolean value indicating whether it is processed or not in order to avoid processing the same component twice.

The enhancements for keeping track of three-cycles unfortunately make the level planar layout algorithm slower than the level planarity testing algorithm. Since the 3-cycle table has $O(|V|^3)$ items then its construction – and, therefore, the whole algorithm – has $O(|V|^3)$ running time.

5 Crossing Minimisation

Crossing minimisation of general level graphs is far more interesting than layout calculation of level planar graphs from a practical perspective. As we observed before, there are three different types of constraints which need to be satisfied when minimising crossings using the vertex-exchange graph. Thus it is natural to suggest Integer Linear Programming (ILP) as a convenient method to satisfy a big variety of constraints. We will see below that properties of the vertex-exchange graph play a role here also.

Jünger *et al.* [6] have proposed an ILP formulation of the crossing minimisation problem. They encode the problem using binary variables for linear ordering. For notational convenience we will refer to a vertex v with $\pi(v) = i$ as vertex i and, similarly, an edge (v, w) with $\pi(v) = i$ and $\pi(w) = j$ as edge (i, j).

$$x_{ij}^r = \begin{cases} 1 & \text{if vertex } i \text{ is placed before vertex } j \text{ on level } r, \\ 0 & \text{if vertex } i \text{ is placed after vertex } j \text{ on level } r. \end{cases}$$

and for the occurrence of a crossing:

$$c_{ijkl}^r = \begin{cases} 1 & \text{if edges } (i, j) \text{ and } (k, l) \text{ cross}, \\ 0 & \text{otherwise}, \end{cases}$$

The inequalities represent 3-cycle constraints and the relation between linear orderings of vertices and crossings (constraints **3-cycle** and **vertex-edge**). The whole ILP formulation is expressed as follows.

Minimise

$$\sum_{r=1}^{p-1} \sum_{(i,j)(k,l) \in E_r} c_{ijkl}^r \tag{1}$$

subject to

$$-c_{ijkl}^r \le x_{jl}^{r+1} - x_{ik}^r \le c_{ijkl}^r \qquad (i,j),\ (k,l) \in E_r,\ j < l \tag{2}$$

$$1 - c_{ijkl}^r \le x_{jl}^{r+1} + x_{ik}^r \le 1 + c_{ijkl}^r \qquad (i,j),\ (k,l) \in E_r,\ j > l \tag{3}$$

$$0 \le x_{ij}^r + x_{jk}^r - x_{ik}^r \le 1 \qquad 1 \le i < j < k \le |V_r| \tag{4}$$

$$x_{ij}^r, y_{ij}^r, c_{ijkl}^r \in \{0, 1\}\ . \tag{5}$$

This ILP formulation can find the layout with minimum number of crossings of medium-sized graphs (about 30-40 vertices and edges) in a few seconds. However, the authors suggest the need for additional inequalities in order to develop a practically useful algorithm.

We propose to employ the odd-labelled cycles for additional inequalities. The basic idea for additional constraints is that we switch the labelling of the edges of a vertex-exchange graph deliberately so that all the cycles become even-labelled. For the input graph, this means that we deliberately create some crossings. As the goal is to have the minimum number of crossings in the final layout, we are interested in as few edge-label switchings as possible. A switching of an edge label means that we set a crossing variable $c_{ijkl} = 1$ for the edges (i, j) and (k, l) of the input graph that the edge of the vertex-exchange graph is mapped from.

Fundamental cycles are of use in finding new equalities. A fundamental cycle is defined as follows.

Definition 3. *Given a simple path $\mathcal{P} = (u, \ldots, w)$ in a spanning tree T of $G = (V, E)$, then if there is a chord $(u, w) \in E$, then $\mathcal{C} = \mathcal{P} \cup (u, w)$ is a fundamental cycle of G.*

We now make use of the following simple fact regarding fundamental cycles.

Lemma 1. *If all the fundamental cycles of a vertex-exchange graph $\mathcal{G}$ are even-labelled then all the cycles of $\mathcal{G}$ are even-labelled.*

Proof. Omitted.

For convenience we will change our notation slightly. Let e be an edge belonging to a fundamental cycle $\mathcal{C}$. In terms of the original graph, let $e = (\langle i, k\rangle, \langle j, l\rangle)$ for some i, j, k, and l. We will use c_e to denote c_{ijkl}.

Now the constraints are expressed as follows. For each odd-labelled fundamental cycle $\mathcal{C}$:

$$\sum_{e \in \mathcal{C}} c_e = 2k_{\mathcal{C}} + 1, \qquad 0 \leq k_{\mathcal{C}} \leq |\mathcal{C}|/2 - 1 \tag{6}$$

and for each even-labelled fundamental cycle $\mathcal{C}$:

$$\sum_{e \in \mathcal{C}} c_e = 2k_{\mathcal{C}}, \qquad 0 \leq k_{\mathcal{C}} \leq |\mathcal{C}|/2 \tag{7}$$

where $c_e \in \{0, 1\}$ and $k_{\mathcal{C}} \in \mathbb{Z}$.

Equation (6) expresses the constraint that an odd-labelled cycle can have only an odd number of switched labellings; equation (7) constrains to be even the number of switched labellings that an even-labelled cycle can have.

Let $\mathcal{C}$ be an odd labelled cycle. Then it is possible to express constraint (6) without the extra integer variables $k_{\mathcal{C}}$:

$$-|\mathcal{D}| + 1 \leq -\sum_{e \in \mathcal{D}} c_e + \sum_{e \in \mathcal{C} \setminus \mathcal{D}} c_e \leq |\mathcal{C}| - |\mathcal{D}| - 1$$
$$\forall \mathcal{D} \subset \mathcal{C},\ |\mathcal{D}| = 0, 2, \ldots, 2\lfloor |\mathcal{C}|/4 \rfloor \tag{8}$$

In the same way, (7) can be expanded.

$$-|\mathcal{D}|+1 \leq -\sum_{e\in\mathcal{D}} c_e + \sum_{e\in\mathcal{C}\setminus\mathcal{D}} c_e \leq |\mathcal{C}|-|\mathcal{D}|-1$$
$$\forall \mathcal{D}\subset\mathcal{C},\ |\mathcal{D}| = 1,3,\ldots,2\lfloor(|\mathcal{C}|-2)/4\rfloor+1 \quad (9)$$

In general, this transformation does not seem to be very efficient because the number of inequalities for one cycle becomes $\sum_{i=1}^{|\mathcal{C}|/4}\binom{|\mathcal{C}|}{2i} = O(2^{|\mathcal{C}|})$. Nonetheless, practical results show that for each fundamental odd-labelled cycle $\mathcal{C}$ the single inequality $\sum_{e\in\mathcal{C}} c_e \geq 1$ alone decreases the number of branch-and-bound nodes visited by the ILP solver.

5.1 Crossing Number Bounds

We have implemented the new ILP formulation, and we have observed that the size of the branch-and-bound tree with new equalities and inequalities is approximately half of that of the original formulation. One of the reasons why the new constraints have such influence can be explained by the improved *lower bound* calculation. It is easy to see that the LP-relaxation of the original ILP (1)–(5) results always with objective value 0, all $c_{ijkl} = 0$, and all $x_{ij} = 0.5$. The new cycle constraints do not allow this, since they force some crossing variables to take non-zero values.

Thus, repeated addition of constraints derived from the vertex-exchange graph provide a sequence of improved lower bounds for the crossing number.

We have also observed in our work that the number of crossings in any embedding of G is always less than or equal to the number of odd-labelled fundamental cycles. Thus we make the following conjecture.

Conjecture 1. The crossing number of a graph, G, is bounded from above by the number of odd-labelled fundamental cycles of its vertex-exchange graph $\mathcal{G}$.

6 Conclusions

In this paper we have proposed a new technique for analysing proper level graphs. The technique – which we call the vertex-exchange graph – admits in an obvious way a method for testing the planarity of (proper) level graphs. Although not asymptotically optimal its conceptual straightforwardness and ease of coding may make the algorithm a viable alternative to known optimal algorithms. Building on this algorithm we have shown how the layout of a planar level graph can be determined. To the best of our knowledge this is the first algorithm that does not lay out such a graph in a level-by-level fashion.

We presented a further application of this idea by using it in an ILP formulation of crossing minimisation. This yielded an improving sequence of lower bounds on the crossing number. Although we do not show it here, under certain conditions these constraints can be shown to be facet defining for the polytope.

Though we have shown several uses of the vertex-exchange graph, we believe others applications exist. For instance, it would be interesting to investigate how the vertex-exchange graph can be employed by heuristic crossing minimisation algorithms. Also, our conjecture that the number of odd-labelled fundamental cycles is an upper bound on the crossing number of the levelled graph would be an interesting and important result.

References

1. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing, Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
2. P. Eades and D. Kelly. Heuristics for drawing 2-layered networks. *Ars Combinatoria*, 21-A:89–98, 1986.
3. P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4):424–437, 1990.
4. P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994.
5. P. Healy and A. Kuusik. Characterisation of level non-planar graphs by minimal patterns. Technical Report UL-CSIS-98-4, University of Limerick, 1998.
6. M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to multi-layer crossing minimization problem. In *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 1997*, volume 1353 of *Lecture Notes in Computer Science*, pages 13–24. Springer-Verlag, 1997.
7. M. Jünger and P. Mutzel. Exact and heuristic algorithms for 2-layer crossing minimization. In *Graph Drawing, Symposium on Graph Drawing, GD '95. Passau, Germany, September 20–22, 1995*, volume 1027 of *Lecture Notes in Computer Science*, pages 337–348. Springer-Verlag, 1995.
8. S. Leipert. *Level planarity testing and embedding in linear time*. PhD thesis, Institut für Informatik, Universität zu Köln, 1998.
9. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.
10. J. Utech, J. Branke, H. Schmeck, and P. Eades. An evolutionary algorithm for drawing directed graphs. In *Proceedings of the 1998 International Conference on Imaging Science, Systems, and Technology (CISST'98)*, pages 154 – 160, 1998.