Drawing Series-Parallel Graphs on Restricted Integer 3D Grids*

Emilio Di Giacomo

Dipartimento di Ingegneria Elettronica e dell'Informazione,
Universitá degli Studi di Perugia, Perugia, Italy
digiacomo@diei.unipg.it

Abstract. A k-track drawing is a crossing-free $3D$ straight-line grid drawing of a graph G on a set of k parallel lines called *tracks*. The minimum value of k for which G admits a k-track drawing is called the *track number* of G. In the existing literature a lower bound of five and an upper bound of fifteen are known for the track number of series-parallel graph. In this paper we reduce this gap for a large subclass of series-parallel graph for which the lower bound remains five but we show an upper bound of eight. We also describe a linear time drawing algorithm that computes a $3D$ straight-line grid drawing of these graphs in volume $4 \times 4 \times 2n$.

1 Introduction

In this paper we study the problem of computing 3D straight-line crossing-free grid drawings of series-parallel graphs with linear volume. Many recent papers devoted to the study of this problem have been presented(see, e.g., [5,6,7,8,9]).

The first subquadratic volume bound for series-parallel graphs was presented in [6] where it is proved that every series-parallel graph admits a drawing in $O(n \log^2 n)$ volume. In [9] Wood gives the first $O(n)$ volume bound for series-parallel graphs. The hidden multiplicative constant in the volume bound provided in [9] is in the order of 10^{16}.

Both papers [6,9] study the problem of computing small-volume 3D drawings by using the following approach, introduced by Felsner Liotta and Wismath [8]. A grid consisting of k parallel lines, called *tracks*, is considered. A drawing of a graph G on such a grid is called a *k-track drawing* of G and the minimum value of k such that G admits a k-track drawing is called the *track number* of G. In [6] it is proved that if a graph has constant track number then it admits a drawing with linear volume. This result suggests that in order to reduce the volume bound one can minimize the track number.

Recent results about track number are due to Dujmović and Wood [7] who proved that k-trees have constant track number, and therefore linear volume bound. As a special case, Dujmović and Wood [7] refine the track number and

* Research supported in part by "Progetto ALINWEB: Algoritmica per Internet e per il Web", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale.

G. Liotta (Ed.): GD 2003, LNCS 2912, pp. 238–246, 2004.

volume bounds of 2-trees (every 2-tree is a series-parallel graph and every series-parallel graph can be augmented to become a 2-tree) and show that the track number of a series-parallel graph is at most 18; as a consequence they show that a series-parallel graph admits a 3D straight-line grid drawings of volume at most $36 \times 37 \times 37\lceil \frac{n}{18} \rceil$.

Very recently, Di Giacomo, Liotta and Meijer [4] improved the results in [7] by presenting new upper bounds on the track number of k-trees; as for series-parallel graph it is proved a track number of at most 15. The corresponding volume described in [4] is $30 \times 31 \times 31\lceil \frac{n}{15} \rceil$. A lower bound of five for the track number of series-parallel graphs is presented in [5].

This paper is motivated by the natural question of reducing the gap between the lower and the upper bound on the track number of series-parallel graphs (i.e. a $5 \div 15$ gap). Our main results can be listed as follows:

- We study a large subclass of series-parallel graphs, called *flat series-parallel graphs*, for which an upper bound of eight on the track number is proved. The lower bound for this class is five.
- We present a linear time drawing algorithm that computes 3D straight-line grid drawings of flat series-parallel graphs in volume $4 \times 4 \times 2n$.

We remark that the class of flat series-parallel graphs have been the subject of investigation in the literature for their interest in book-embedding and VLSI applications (see, e.g., [1]).

2 Preliminaries

A *series-parallel digraph* (also called as an *SP-digraph* in the following) is a directed planar graph recursively defined as follows [2]: (i) A directed edge $e = (s, t)$ is an SP-digraph; s is the *source pole* and t is the *sink pole* of the SP-digraph. (ii) Let G' and G'' be two SP-digraphs, whose source poles are s' and s'', respectively, and whose sink poles are t' and t'', respectively; the digraph G obtained by identifying t' with s'' is also an SP-graph; s' is the *source pole* of G and t'' is the *sink pole* of G. (iii) Let G' and G'' be two SP-digraphs, whose sources are s' and s'', respectively, and whose sinks are t' and t'', respectively; the digraph G obtained by identifying s' with s'' and t' with t'' is also an SP-digraph; $s' = s''$ is the *source pole* and $t' = t''$ is the *sink pole* of G. The source pole and the sink pole of an SP-digraph G are also called the *poles* of G. The undirected graph underlying an SP-digraph is a *series-parallel graph*, also called SP-graph.

Let G be an SP-digraph. A *split pair* of G is either a separation pair or a pair of adjacent vertices of G. A *split component* of G with respect to a split pair $\{s, t\}$ is either an edge (s, t) or a maximal subgraph C of G such that C is an st-graph [2] and $\{s, t\}$ is not a split pair of C.

We briefly recall the definition of the *SPQ-tree* of G, also called the *decomposition tree* of G (see [3] for a complete definition). A SPQ-tree is an ordered rooted tree T whose nodes are of three types: S,P, or Q. A S-node corresponds

to a series composition of blocks, and it has a child for each block; A P-node corresponds to a parallel composition of split components with respect to a separation pair and it has a child for each split component; A Q-node corresponds to a single edge and it has no child. The *pertinent digraph* of a node μ of T is the subgraph of G whose decomposition tree is the tree rooted at μ. We call *poles* of a node μ of T the poles of the pertinent digraph of μ. Also, to simplify the description of our algorithms we shall distinguish the S-nodes having only Q-nodes as children from the others. Namely, an S-node whose children are all Q-nodes will be called an *S^*-node*.

Two parallel components of an SP-digraph G are said to be *nested* if they share a pole and they are not in series. Let v be a vertex of G, the *depth* of v is the number of nested parallel components having v has a pole. An SP-digraph *has depth k* if each pole has depth at most k. An SP-digraph of depth 1 is said to be *flat*. An SP-graph G is *flat* if there exists an st-orientation [2] of G such that the resulting SP-digraph is flat.

A *track* is a set of 3D grid points on a straight line of infinite length. We always assume that a track is parallel to the x-axis, thus a track is the set of all the grid points having the same y- and z-coordinate. We denote as (x, Y, Z) a track whose points have y-coordinate Y and z-coordinate Z. A *k-track drawing* of a graph G is a 3D straight-line crossing-free grid drawing of G such that each vertex of G is drawn on one of k tracks. The minimum value of k such that G admits a k-track drawing is called the *track number* of G.

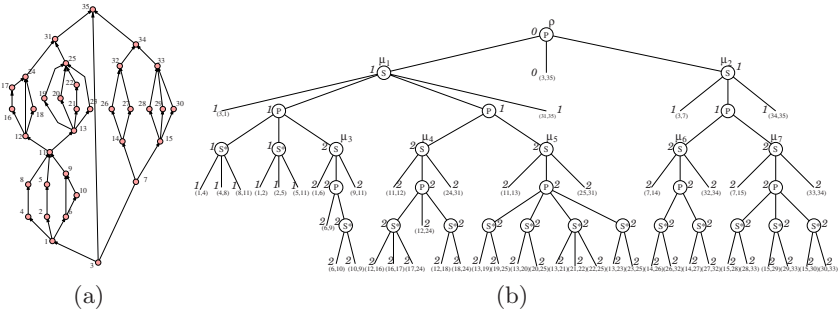
An *8-prism* is the 3D integer grid consisting of the eight tracks $T_0 = (x, 0, 2)$, $T_1 = (x, 0, 1)$, $T_2 = (x, 3, 2)$, $T_3 = (x, 3, 1)$, $T_4 = (x, 1, 0)$, $T_5 = (x, 1, 3)$, $T_6 = (x, 2, 0)$, and $T_7 = (x, 2, 3)$. A *strip* σ_{ij} is the portion of plane delimited by tracks T_i and T_j. The 8 tracks of an 8-prism can be grouped in two subgrids: the first one, called *even ingot*, consists of the four tracks with even numbers; the second one, called *odd ingot*, consists of the four tracks with odd numbers. A track drawing on the eight tracks of an 8-prism will be called *8-prism drawing* in the following. Also, an 8-prism drawing will be specified by assigning to each vertex two numbers: $track(v)$ is an integer that denotes the track to which v is assigned; $x(v)$ is the x-coordinate assigned to v. Finally, $|G|$ will denote the number of vertices of a given digraph G.

3 8-Prism Drawings of Flat SP-Digraphs

In this section we present an algorithm that computes an 8-prism drawing of a flat SP-graph. Since a flat st-orientation of a flat SP-graph can be computed in polynomial time we focus on flat SP-digraphs.

Let G be a flat SP-digraph and let T be the SPQ-tree of G. A *level numbering* of T is a numbering of each node μ of T denoted by $level(\mu)$ and computed as follows. The root ρ of T has $level(\rho) = 0$. For each non-root node μ, if μ is an S-node then $level(\mu) = level(parent(\mu)) + 1$, else $level(\mu) = level(parent(\mu))$.

Figure 1 depicts an SP-digraph G and the level numbering of its decomposition tree. Using the level numbering of T we assign a number $level(v)$ to each



(a) (b)

Fig. 1. (a) An SP-digraph G and (b) the level numbering of its decomposition tree.

vertex v of G, called *level of* v. Such number is the minimum among the levels of all the nodes of T having v as a pole. We call *jumping edges* those edges whose endvertices are assigned to different levels. Notice that the level numbering is such that the level number changes in correspondence of the S-nodes.

Lemma 1. *Let G be a flat SP-digraph, let T be the decomposition tree of G, and let μ be an S-node of T such that $level(\mu) = j$ $(j > 0)$. The leftmost child and the rightmost child of μ are both Q-nodes and the edges associated with them are both jumping edges.*

Let μ be either an S-node or the root node of T and let $level(\mu) = j$, we call *j-digraph of μ* the subgraph G^j_μ of G_μ, induced by those vertices having level number j. Notice that a j-digraph may not be connected . The subtree of T_μ consisting of all nodes with level umber equal to j except, for the non-root node, the leftmost child and the rightmost child is called *j-tree rooted at μ* and is denoted as T^j_μ. A *j-component* is the union of all the j-digraph of level j. By Lemma 1, each j-digraph G^j_μ $(j > 0)$ is connected to vertices of level $j - 1$ by two jumping edges, corresponding to the rightmost child and the leftmost child of μ.

We describe now how to compute a drawing of a single j-digraph. Each j-digraph is drawn on one of the two ingots defined by the 8-prism. The four tracks of the ingot are denoted as T_s, T_t, T_0 and T_1 and are arranged as shown in Figure 3. Also, we refer to T_s as the *source track* and to T_t as the *sink track*. Figure 2 shows Algorithm DrawJDigraph() and Figure 3 is an example of a resulting drawing. In the algorithm, μ is an S-node or the root of T and $\nu_0, \ldots, \nu_h$ are the children of μ in T^j_μ. For a child ν_i $(i = 0, \ldots, h)$, the source pole is s_{ν_i} and the sink pole is t_{ν_i}.

Lemma 2. *Let G be a flat SP-digraph, whose decomposition tree is T. Let μ be an S-node of T or the root of T and let $level(\mu) = j$. Let G^j_μ be the j-digraph of μ. Algorithm* DrawJDigraph*() computes an ingot-drawing of G^j_μ in $O(|G^j_\mu|)$ time.*

DrawJDigraph(G^j_μ,x_s)
Input: The j-digraph G^j_μ of an S-node or of the root μ of T and a starting x-coordinate x_s;
Output: A crossing-free straight-line grid drawing of G^j_μ on the ingot defined by T_s, T_t, T_0 and T_1;

(1) $track(s_{\nu_0}) := T_s$; $x(s_{\nu_0}) := x_s$; $\Delta_x := 1$;
(2) **if** ν_0 is a P-node having S^*-nodes as children
(3) let $\mu_0, \ldots, \mu_\xi$ be the children of ν_0 that are S^*-nodes;
(4) **for** $g = 0$ **to** ξ
(5) let $\{(s_{\nu_0}, v_1), (v_1, v_2), \ldots, (v_m, t_{\nu_0})\}$ be the pertinent digraph of μ_g;
(6) **for** $y = 1$ **to** m
(7) $track(v_y) := T_t$; $x(v_y) := x_s + \Delta_x$; $\Delta_x := \Delta_x + 1$;
(8) $track(t_{\nu_0}) := T_0$; $x(t_{\nu_0}) := x_s + \Delta_x$; $\Delta_x := \Delta_x + 1$;
(9) **for** $i = 1$ **to** $h - 1$
(10) **if** ν_i is a P-node having S^*-nodes as children
(11) let $\mu_0, \ldots, \mu_\xi$ be the children of ν_i that are S^*-nodes;
(12) **for** $g = 0$ **to** ξ
(13) let $\{(s_{\nu_i}, v_1), (v_1, v_2), \ldots, (v_m, t_{\nu_i})\}$ be the pertinent digraph of μ_g;
(14) **for** $y = 1$ **to** m
(15) $track(v_y) := T_t$; $x(v_y) := x_s + \Delta_x$; $\Delta_x := \Delta_x + 1$;
(16) $\alpha := i \mod 2$; $track(t_{\nu_i}) := T_\alpha$; $x(t_{\nu_i}) := x_s + \Delta_x$; $\Delta_x := \Delta_x + 1$;
(17) **if** ν_h is a P-node having S^*-nodes as children
(18) let $\mu_0, \ldots, \mu_\xi$ be the children of ν_h that are S^*-nodes;
(19) **for** $g = 0$ **to** ξ
(20) let $\{(s_{\nu_h}, v_1), (v_1, v_2), \ldots, (v_m, t_{\nu_h})\}$ be the pertinent digraph of μ_g;
(21) **for** $y = 1$ **to** m
(22) $\alpha := h \mod 2$; $track(v_y) := T_\alpha$; $x(v_y) := x_s + \Delta_x$; $\Delta_x := \Delta_x + 1$;
(23) $track(t_{\nu_h}) := T_t$; $x(t_{\nu_h}) := x_s + \Delta_x$;

Fig. 2. Algorithm DrawJDigraph()

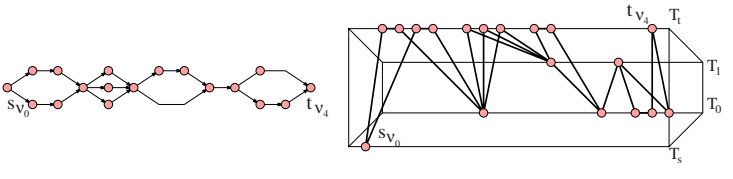


Fig. 3. A j-digraph and its drawing on an ingot.

Proof. We prove that the drawing computed by Algorithm DrawJDigraph() is crossing-free. Suppose that two edges $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ cross each other. Since all the vertices have different x-coordinates, a crossing is possible only if one of the endvertices of e_2 has x-coordinate between $x(u_1)$ and $x(v_1)$ or one of the endvertices of e_1 has x-coordinate between $x(u_2)$ and $x(v_2)$. There are three types of crossings:

1. Crossing on a track (overlap). The only edges having both endvertices on the same track are the edges of the simple paths connecting the poles of a P-child of μ. The endvertices of these edges have consecutive x-coordinates (see rows 4–8, 12–15, 19–22), thus a crossing of this type is not possible.

2. Crossing between edges lying on a strip. Without loss of generality, suppose that e_1 has endvertices with non-consecutive x-coordinates. At least one of the endvertices of e_1 is a pole of a P-child ν_i of μ. We consider only the case in which u_1 coincides with the source pole s_{ν_i} of ν_i, the other cases are analogous. The vertex v_1 can be either the first vertex of any simple path which connects $u_1 = s_{\nu_i}$ to t_{ν_i}, or it may coincide with t_{ν_i}. If $v_1 = t_{\nu_i}$ then the vertices having x-coordinates between $x(u_1)$ and $x(v_1)$ are the vertices of the simple paths connecting $u_1 = s_{\nu_i}$ to $v_1 = t_{\nu_i}$; no one of these vertices is either on the track of u_1 or on the track of v_1, thus edge e_2 cannot have an endvertex with x-coordinate between u_1 and v_1 and a crossing is not possible. If $v_1 \neq t_{\nu_i}$ then the vertices having x-coordinates between $x(u_1)$ and $x(v_1)$ are a subset of the vertices of the simple paths connecting $u_1 = s_{\nu_i}$ to t_{ν_i}; these vertices are on the same track of v_1. The edges having one of these vertices as an endvertex, and lying on the same strip as e_1, have u_1 as the other endvertex. The edge e_2 must be one of these edges. Edges e_1 and e_2 cannot cross because they have a common endvertex.

3. Crossing between edges lying on two strips. For such a crossing, edges e_1 and e_2 must lie on two different strips which cross each other along a straight line which is not one of the four tracks T_s, T_t, T_0, and T_1. The only two strips which cross in such a way are the strip $\sigma_{s,1}$, and the strip $\sigma_{t,0}$. On the strip $\sigma_{s,1}$ we have no edges. Namely, the only vertex on track T_s is the source pole s_{ν_0} of ν_0 and it can be connected either to vertices on track T_t or to a vertex on track T_0. It follows that a crossing is not possible.

Since the algorithm visit a vertex at a time and executes a constant number of operations for each vertex, its time complexity is $O(|G^j_\mu|)$. □

Algorithm DrawJDigraph() draws a single j-digraph on an ingot. The drawing of a j-component $G(j)$ can be obtained by two different algorithms called GrainedDrawing() and CounterGrainedDrawing(), respectively. Both algorithms have two main steps: in the first step a drawing is computed by Algorithm DrawJDigraph() for each j-digraph $G^j_{\mu_i} \in G(j)$, and all these drawings are consecutively arranged on the four tracks, according to the left to right order of the nodes μ_i $(i = 1, \ldots, n_j)$ in T. In the second step vertices on each track are shifted. More precisely, Algorithm GrainedDrawing() shifts vertices so that: (i) all vertices on track T_0 precede all vertices on track T_1, which precede all vertices on track T_s, which precede all vertices on track T_t; (ii) the vertices on track T_0 and T_1 have odd coordinates, while the vertices on T_s and T_t have even coordinates. Algorithm CounterGrainedDrawing() shifts vertices so that: (i) all vertices on track T_1 precede all vertices on track T_0, which precede all vertices on track T_s, which precede all vertices on track T_t; (ii) the vertices on track T_0 and T_1 have odd coordinates, while the vertices on T_s and T_t have even coordinates. We call a drawing produced by Algorithm GrainedDrawing()

grained and a drawing produced by Algorithm CounterGrainedDrawing() *counter-grained*.

Lemma 3. *Let G be a flat SP-digraph, with decomposition tree T, and let $G(j) = \{G^j_{\mu_1}, \ldots, G^j_{\mu_{n_j}}\}$ be the j-component of level j. Then the Algorithm* GrainedDrawing() *and the Algorithm* CounterGrainedDrawing() *compute an ingot-drawing of $G(j)$ in $O(|G(j)|)$ time.*

To conclude the description of our technique we present now an algorithm, called 8PrismDrawing(), that draws all the j-components $G(j)$ ($j = 0, \ldots, n_{max}$) of an SP-digraph G on an 8-prism. The different j-components of G are considered consecutively from $G(n_{max})$ to $G(0)$. If j is odd $G(j)$ is drawn on the odd ingot and its drawing is a grained drawing. If j is even $G(j)$ is drawn on the even ingot and its drawing is a counter-grained drawing. Moreover the source track of $G(j)$ coincides with track T_a, where $a = j \mod 4$; as a consequence the sink track of the drawing coincides with track T_b, where $b = (j + 2) \mod 4$. Figure 4 depicts the arrangement of the j-components on the 8-prism.

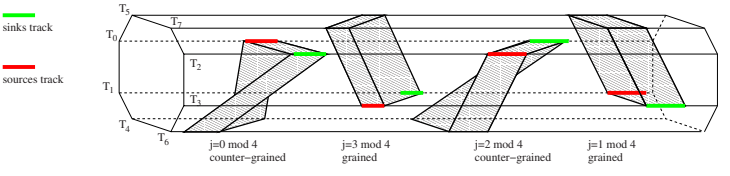


Fig. 4. The arrangement of the drawings of the j-components on the 8-prism.

In order to prove that the jumping edges connecting the j-digraph of different levels can be added to the drawing computed by Algorithm 8PrismDrawing() without creating crossings, we recall some properties of jumping edges. Each j-digraph G^j_μ ($j > 0$) is connected to vertices of level $j - 1$ by two jumping edges, corresponding to the rightmost child and leftmost child of μ. We call *core poles* the poles of a j-digraph and *simple poles* the poles that are not core poles. Thus, given a jumping edge, the endvertex with the larger level number is a core pole of a j-digraph G^j_μ. The other endvertex can be either a core pole or a simple pole of a $(j-1)$-digraph. If a jumping edge connect two core poles, then these two core poles are of the same type, i.e. they are either both source core pole or both sink core poles. The following lemma proves that the drawing obtained by Algorithm 8PrismDrawing() is crossing-free.

Lemma 4. *Let G be a flat SP-digraph. Algorithm* 8PrismDrawing() *computes an 8-prism drawing of G.*

Proof. The drawing of a single j-component $G(j)$ ($0 \leq j \leq n_{max}$) computed either by Algorithm GrainedDrawing() or by Algorithm CounterGrained-Drawing() is crossing free by Lemma 3.

We show that the jumping edges do not cross each other and do not cross any other edge. As pointed out the jumping edges connect either source (sink) core pole to source (sink) core pole, or simple poles to core poles. Thus, the jumping edges are drawn on strips $\sigma_{1,2}$, $\sigma_{0,3}$, $\sigma_{0,1}$, $\sigma_{2,3}$, $\sigma_{0,5}$, $\sigma_{0,7}$, $\sigma_{2,5}$, $\sigma_{2,7}$, $\sigma_{1,4}$, $\sigma_{1,6}$, $\sigma_{3,4}$, and $\sigma_{3,6}$. First, we prove that jumping edges lying on the same strip do not cross each other. Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two jumping edges such that $track(u_1) = track(u_2)$ and $track(v_1) = track(v_2)$. It is either $level(u_1) = level(v_1)+1$ and $level(u_2) = level(v_2)+1$ or $level(v_1) = level(u_1)+1$ and $level(v_2) = level(u_2)+1$. Assume without loss of generality that $level(v_1) = level(u_1) + 1$ and $level(v_2) = level(u_2) + 1$. It follows that v_1 is the core pole of a j-digraph $G^j_{\mu_1}$ and that v_2 is the core pole of a h-digraph $G^h_{\mu_2}$. Consider the case when $j \neq h$, and assume $j < h$. Since v_1 and v_2 are on the same track then h is at least $j + 2$ and therefore $level(u_1) < level(v_1) < level(u_2) < level(v_2)$. It follows that $x(u_1) < x(v_1) < x(u_2) < x(v_2)$ and a crossing is not possible. If $j = h$ then $G^j_{\mu_1}$ and $G^h_{\mu_2}$ are in the same j-component $G(j)$. Since $x(v_1) < x(v_2)$ then μ_1 is to the left of μ_2 in the SPQ-tree. If μ_1 and μ_2 have the same parent ν then $u_1 = u_2$ and the two edges can not cross since they have a common vertex. If μ_1 and μ_2 have different parents ν_1 and ν_2, then since μ_1 is to the left of μ_2, it follows that ν_1 is to the left of ν_2, and therefore $x(u_1) < x(u_2)$. A crossing is not possible either in this case.

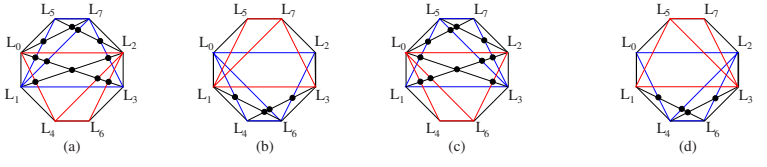


Fig. 5. The possible crossings of the jumping edges.

Now we prove that the jumping edges do not cross any other edge on different strips. Depending on the value of j the jumping edges between levels j and $j+1$ are on different strips and therefore the possible crossings are different. Figure 5 shows all the possible cases depending on the value of j. In Figure 5(a) the drawing on the even ingot has level $j \mod 4 = 0$ and the drawing on the odd ingot has level $(j-1) \mod 4 = 3$. In Figure 5(b) the drawing on the odd ingot has level $j \mod 4 = 3$ and the drawing on the odd ingot has level $(j-1) \mod 4 = 2$. In Figure 5(c) the drawing on the even ingot has level $j \mod 4 = 2$ and the drawing on the odd ingot has level $(j-1) \mod 4 = 1$. In Figure 5(d) the drawing on the odd ingot has level $j \mod 4 = 1$ and the drawing on the even ingot has level $j-1 \mod 4 = 0$. The black dots represent the 30 possible crossings.

Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be two edges; if they cross each other, the four endvertices are co-planar. The co-planarity condition of the four vertices

can be expressed by means of an equation on the x-, y- and z-coordinates of the vertices and since the y- and z- coordinates are determined by the choice of the tracks, the equation reduces to a co-planarity condition on the x-coordinates. For each of the 30 cases the x-coordinates assigned to vertices are such that the equation above is never satisfied. This is a consequence of the fact that some vertices have even x-coordinates and some vertices have odd x-coordinates, and of the relative position of the vertices induced by the grained and counter-grained drawings. The analysis of the 30 cases is straightforward but tediousand is omitted here for reasons of brevity. □

Theorem 1. *Let G be a flat series-parallel digraph with n vertices. Then G has track number eight and there exists an algorithm that computes an 8-track-drawing of G in $O(n)$ time and with at most $4 \times 4 \times 2n$ volume.*

Proof. The drawing computed by Algorithm 8PrismEmbedder() is crossing free by Lemma 4. For what concern the time complexity of the algorithm: the SPQ-tree T and the level numbering of G can be computed in $O(n)$ time. The drawing of each j-component $G(j)$ can be computed in $O(|G(j)|)$ time by Lemma 3. It follows that Algorithm 8PrismEmbedder() has time complexity $O(n)$. For what concerns the volume of the drawing, we have that: (i) each vertex has a different x-coordinate; (ii) the grained or counter-grained drawing of each j-component $G(j)$ requires a portion of the 8-prism of length $2|G(j)|-1$, in order to guarantee that some vertices have even x-coordinates and some other have odd x-coordinates. The length of the drawing is then $\sum_{j=0}^{n_{max}}(2|G(j)|-1) = 2n-n_{max}$, and the overall volume is at most $4 \times 4 \times 2n$. □

References

1. F. R. K. Chung, F. T. Leighton, and A. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM J on Alg. and Disc. Methods*, 8:33–58, 1987.
2. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.
3. G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996.
4. E. Di Giacomo, G. Liotta, and H. Meijer. 3D straight-line drawings of k-trees. Submitted for publication.
5. E. Di Giacomo, G. Liotta, and S. K. Wismath. Drawing series-parallel graphs on a box. In *Proc. CCCG 2002*, 2002.
6. V. Dujmović, P.Morin, and D. Wood. Pathwidth and three-dimensional straight line grid drawings of graphs. In *Proc. GD 2002*, volume 2528 of *LNCS*, pages 42–53. Springer-Verlag, 2002.
7. V. Dujmović and D. R. Wood. Tree-partitions of k-trees with application in graph layout. In *Proc. WG 2003*, LNCS. Springer-Verlag, to appear.
8. S. Felsner, G.Liotta, and S. K. Wismath. Straight line drawings on restricted integer grids in two and three dimensions. In *Proc. GD 2001)*, volume 2265 of *LNCS*, pages 328–342. Springer-Verlag, 2001.
9. D. R. Wood. Queue layouts, tree-width, and three-dimensional graph drawing. In *Proc. FSTTCS '02*, volume 2556 of *LNCS*, pages 348–359. Springer-Verlag, 2002.