

Schematisation of Tree Drawings

Joachim Gudmundsson¹, Marc van Kreveld², and Damian Merrick^{1,3}

¹ National ICT Australia*, Sydney, Australia.

{joachim.gudmundsson,damian.merrick}@nicta.com.au

² Department of Computer Science, Utrecht University, The Netherlands.
marc@cs.uu.nl

³ School of Information Technologies, University of Sydney, Australia.

Abstract. Given a tree T spanning a set of points \mathcal{S} in the plane, we study the problem of drawing T using only line segments aligned with a fixed set of directions \mathcal{C} . The vertices in the drawing must lie within a given distance r of each original point $p \in \mathcal{S}$, and an objective function counting the number of bends must be minimised. We propose five versions of this problem using different objective functions, and algorithms to solve them. This work has potential applications in geographic map schematisation and metro map layout.

1 Introduction

A schematic drawing is one in which line segments conform to a restricted set of orientations, often for the purpose of increasing readability. Schematic drawings are widely used in many application areas, including electronics, software engineering and visualising geographical networks.

There has been extensive research into orthogonal graph drawing, where edges of a graph are drawn as sequences of alternating horizontal and vertical line segments; see Eiglsperger et al. [5] for a survey. Lauther and Stübinger [8], and subsequently Brandes et al. [2], investigated the problem of generating orthogonal schematic plans from sketch drawings. Hong et al. [7], Stott and Rodgers [13] and Nöllenburg and Wolff [12] looked at the layout of metro maps, in which lines generally conform to horizontal, vertical and diagonal (45° and 135°) orientations.

A popular topic in cartography is polygonal line simplification, and this has also been investigated in a setting of restricted orientations [10, 11]. Cabello et al. [3] presented an algorithm to construct various types of schematisations of geographical networks such as roads and railways.

Particularly of interest are certain point placement problems in computational geometry. Cabello and van Kreveld [4] study the problem of

* National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

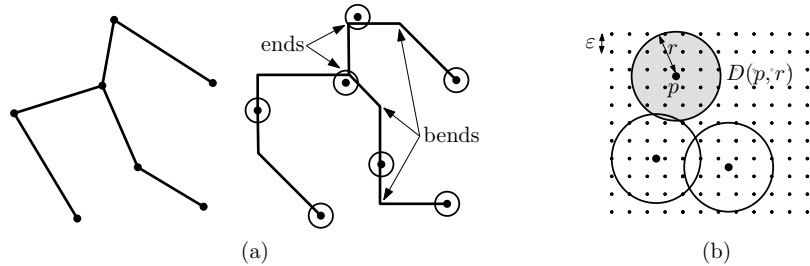


Fig. 1. (a) A tree (left) and a schematisation of the tree (right). Some bends (along edges) and ends (at vertices) are marked. (b) The grid G_ϵ .

aligning as many points as possible in a given set of orientations. If a planar graph is defined on the points, and only points connected in the graph are considered, they prove the problem NP-hard, even when only one orientation is used for alignment. They show that discrete forms of the problem can be solved more efficiently, however. This motivates us to discretise our own problems; we detail this later in this section.

In this paper, we aim to produce schematic drawings of trees from initial sketch, geographical or automatic layouts. We propose five problems. Each takes as input a tree T spanning a point set \mathcal{S} , and the output should be a *schematisation* of the tree's initial layout (See Fig. 1(a)).

To be considered a schematisation, we require that line segments in the drawing align with certain orientations, and that they pass within a certain distance of each original point $p \in \mathcal{S}$. In addition, we want to minimise an objective function $M(p)$ defined for each point $p \in \mathcal{S}$. $M(p)$ measures the complexity of the lines from p to all of p 's children in T .

We formally define the concept of a *tree schematisation* as follows.

Definition 1. *Given a tree T spanning a point set \mathcal{S} , a set of directions \mathcal{C} and a real number $r > 0$, a tree schematisation T' of T is a layout of T such that:*

1. *every line segment of T' is aligned with a direction $c \in \mathcal{C}$,*
2. *T' intersects the disc $D(p, r)$ centred at every $p \in \mathcal{S}$ with radius r , and*
3. *$\sum_{p \in \mathcal{S}} M(p)$ is minimised.*

The complexity of a tree schematisation can be measured in a number of ways. The primary difference between the five problems we propose lies in the objective function $M(p)$. The choice of $M(p)$ defines how to calculate a cost for each point of a given schematisation. Each problem

bases this choice on a particular assumption of how to measure complexity. Some of the problems also take additional input in the form of a set of coloured paths covering T , which in turn affects the objective function.

Henceforth, we call the five problems $\pi_1 - \pi_5$. The objective functions associated with each of these are denoted $M_1(p) - M_5(p)$. We refer to the value of these functions for a schematisation as the M_1 cost, M_2 cost, etc.

Each objective function measures the number of *bends* and *ends* in a given schematisation, as illustrated in Fig. 1(a). We use the term *bend* to mean a change in direction of a line between two vertices in the tree T . We use *end* to mean a bend at a vertex; that is, a line that does not continue through a vertex in a single direction. What counts as an end depends on the particular problem; we detail this in the following sections.

In this paper, we assume that the set of directions \mathcal{C} is constructed by dividing the angle 2π by a given even integer m of desired directions, i.e. the i th direction in \mathcal{C} is defined by an angle of $\frac{2\pi i}{m}$. m must be even to ensure that for every direction $c \in \mathcal{C}$, the opposite direction \bar{c} is also in \mathcal{C} . We look at a discretised version of the general problem of constructing tree schematisations. We place a finite set of points inside each disc $D(p, r)$, and the constructed schematisation must pass through one of these points for each point $p \in T$. To obtain the set of points inside each disc, we use a global grid G_ε with ε horizontal and vertical spacing between grid points. The set of points for a disc $D(p, r)$, denoted $G_\varepsilon(p)$, is the set of grid points in G_ε that lie inside $D(p, r)$. There are $\mathcal{O}(\frac{r}{\varepsilon^2})$ such points (See Fig. 1(b)).

Sections 2 and 3 define $\pi_1 - \pi_5$ in detail. Section 4.1 presents a general method for computing tree schematisations on the grid G_ε . This method runs in time exponential in the degree of the tree. In many applications, the degree may be considered constant, and this may be acceptable. However, $\pi_1 - \pi_5$ can be solved in time polynomial in the degree of the tree; Sections 4.2 - 4.5 give algorithms for this, based on the general method.

2 Coloured Tree Problems

Problems π_1 and π_2 involve coloured trees; they take as input a set \mathcal{P} of elementary paths covering the tree T , with each path identified by a distinct colour (See Fig. 2).

Problem π_1 : Single edges only. In π_1 , we assume that the paths in the set \mathcal{P} are edge-disjoint (as in Fig. 2(a)), and we want to minimise the number of bends and ends along each of the coloured paths independently.

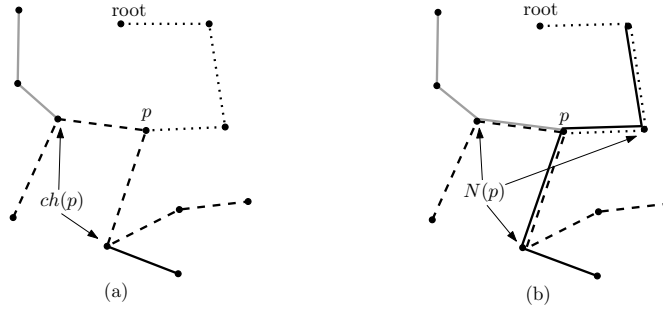


Fig. 2. An example of a “coloured tree”, with solid, dashed, dotted and light grey lines representing four different coloured paths. The left example shows an edge-disjoint path cover (π_1), and the right shows a path cover with multiple edges allowed (π_2). The children $ch(p)$ and neighbourhood $N(p)$ of the point p are illustrated.

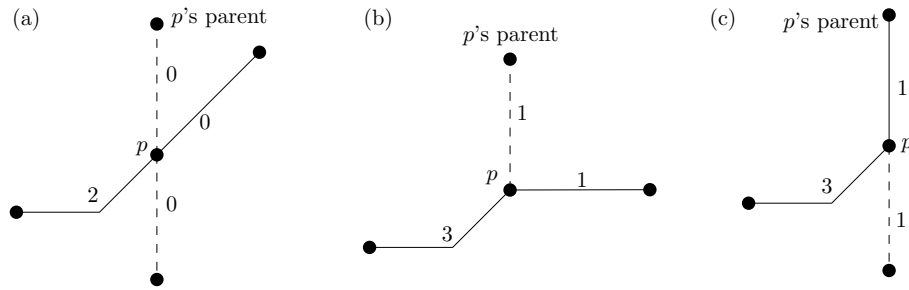


Fig. 3. Counting ends and bends in π_1 . A bend costs the same as two ends. The M_1 costs are (a) (1 bend) = 2, (b) (1 bend + 3 ends) = 5, and (c) (1 bend + 3 ends) = 5.

Choose an arbitrary leaf of T to be the root vertex. For any vertex p in T , let $N(p)$ be the set of vertices adjacent to p and let $ch(p) \subseteq N(p)$ be the set of p 's children. We define π_1 's objective function $M_1(p)$ as:

$$M_1(p) = \sum_{q \in N(p)} \text{ends}_1(q, p) + \sum_{q \in ch(p)} 2 \times \text{bends}(q, p)$$

where $\text{ends}_1(q, p) = 0$ if there is an edge of the same colour as (q, p) entering p from the opposite side, or 1 otherwise, and $\text{bends}(q, p)$ is the number of bends along the edge between q and p . Fig. 3 shows examples.

Problem π_2 : Multiple independent edges. Problem π_2 removes the requirement for the coloured paths to be edge-disjoint; more than one path may contain an edge between the same two vertices (See Fig. 2(b)). Given multiple edges in the input, it holds that $M_2(p) = M_1(p)$. Every



Fig. 4. Three lines entering a vertex in the same direction, with (a) no lines, (b) one line and (c) three lines in the opposite direction.

edge between the same pair of vertices counts towards the cost, where in π_1 there could only be one edge between each adjacent pair of vertices.

π_2 requires that the endpoints of multiple edges between two nodes must coincide in the schematisation, but not the number of bends in the edge nor the orientations of the line segments.

3 Uncoloured Tree Problems

The three remaining problems, $\pi_3 - \pi_5$, take uncoloured trees; they assume that no covering set of paths is given. This leads to ambiguity in how to count ends, particularly when many lines in the schematisation enter a vertex in the same direction. $\pi_3 - \pi_5$ model three different possibilities.

Problem π_3 : Parallel lines remain separate. π_3 assumes that multiple edges, or any lines entering a vertex in a single direction, are drawn separately. For every line entering a vertex in a certain direction, there must be another line entering the vertex in the opposite direction, otherwise the line is counted as an end. The examples shown in Fig. 4(a), (b) and (c) give 3, 2 and 0 ends, respectively. The objective function is:

$$M_3(p) = \sum_{c \in \mathcal{C}} \frac{1}{2} \times |\text{edges}(p, c) - \text{edges}(p, \bar{c})| + \sum_{q \in \text{ch}(p)} 2 \times \text{bends}(q, p)$$

where $\text{edges}(p, c)$ is the number of c -directed lines entering p , and $\text{edges}(p, \bar{c})$ is the number of lines entering p in the opposite direction to c . The factor of $\frac{1}{2}$ ensures counting ends in each pair of opposing directions only once.

Problem π_4 : Parallel lines merge to line on opposite side. Problem π_4 assumes that lines entering a vertex from the same direction are allowed to merge without cost inside the vertex, given the presence of at least one line in the opposing direction. If there is no line in a given direction, an end is counted for every line in the opposing direction. If one or more lines are placed in a pair of opposing directions, there are no ends.

$$M_4(p) = \sum_{q \in \mathcal{S}, (q, p) \in T} \text{ends}_4(q, p) + 2 \times \text{bends}(q, p)$$

where $\text{ends}_4(q, p) = 0$ if there is any edge entering p from the opposite direction to the edge (q, p) , or 1 otherwise. In this problem, the examples in Fig. 4(a), (b) and (c) produce 3, 0 and 0 ends respectively.

Problem π_5 : Parallel lines always merge. The final problem, π_5 , assumes that lines entering a vertex from the same direction are drawn as a single line; that is, parallel lines merge to a single line *before* a vertex. As soon as at least one line is present in the opposite direction, no ends are counted. $M_5(p)$ is equal to:

$$\frac{1}{2} \sum_{c \in \mathcal{C}} |\text{sign}(\text{edges}(p, c)) - \text{sign}(\text{edges}(p, \bar{c}))| + 2 \sum_{q \in \text{ch}(p)} \text{bends}(q, p)$$

where $\text{sign}(x) = 1$ if $x > 0$, or 0 otherwise. For the examples in Fig. 4, the number of ends are 1, 0 and 0 for (a), (b) and (c) respectively.

4 Algorithms for Tree Schematisation

In this section, we present an algorithm to produce a schematisation of a given tree, without considering the specific objective function being used. The algorithm is then modified for each of the problems $\pi_1 - \pi_5$ to provide a more efficient solution in each case. In all cases, we use the grid discretisation introduced in Section 1. First, we define an important concept used throughout this section.

Definition 2. *Given a set of directions \mathcal{C} , a \mathcal{C} -directed path between two points p and q is a polygonal chain from p to q in which every line segment is parallel to some direction in \mathcal{C} .*

4.1 A General Solution

Start by choosing an arbitrary leaf vertex to be the root vertex p_{root} of the tree T . Perform a post-order traversal on T , visiting each point $p \in \mathcal{S}$ once from the leaves to the root. The algorithm finds optimal solutions for each subtree T_p of T rooted at p , and during its traversal of T propagates these solutions upward, until a complete solution has been generated at the root vertex. We use the term M cost to denote the value of the objective function $M(p)$. The M cost of a subtree of T is the summed M cost of all points in the schematisation of the subtree.

For each vertex p , given a grid point $p' \in G_\varepsilon(p)$ and an direction $c \in \mathcal{C}$, let $M_{\min}(p, p', c)$ be the minimum M cost of the subtree of T rooted at p , assuming that the schematisation passes through p' , and that the path from p to p 's parent will leave p' in direction c . We will refer to the direction c as the *parent direction* from p . We assume in this description that there is only one edge from p to p 's parent, and hence only one \mathcal{C} -directed path can be constructed in the schematisation. In the problem π_2 , p may

have multiple edges to its parent, and the schematisation may use different directions for each. In this case, we perform the same steps several times, once for each edge. This process is further detailed in Section 4.2.

When the algorithm visits the vertex p , it computes and stores for every point $p' \in G_\varepsilon(p)$, and for every parent direction $c \in \mathcal{C}$ the value of $M_{min}(p, p', c)$, along with the corresponding solution. If p has k children, then k paths must be stored for each solution. Thus $\mathcal{O}(\frac{|\mathcal{C}|kr}{\varepsilon^2})$ values are stored at each vertex p .

As introduced in Section 1, the objective function being minimised by the algorithm may be divided into a count of *bends* and *ends*. We can count the number of bends by considering each child $q \in \text{ch}(p)$ in turn.

Given two grid points $p' \in G_\varepsilon(p)$, $q' \in G_\varepsilon(q)$, let $\delta(q', p', c_1, c_2)$ be the minimum number of bends needed by a \mathcal{C} -directed path from q' to p' that starts in direction $c_1 \in \mathcal{C}$ and ends in direction $c_2 \in \mathcal{C}$. This value can be computed in $O(1)$ time. Compute $\delta(q', p', c_1, c_2)$ for every pair of grid points $p' \in G_\varepsilon(p)$, $q' \in G_\varepsilon(q)$ and for each pair of directions $c_1, c_2 \in \mathcal{C}$. These values are the minimum number of bends required for any possible path from q to p . Now add $M_{min}(q, q', c_1)$ to these and store the minimum solution for every child q , grid point p' , final path direction c_2 , and parent direction c . To calculate $M_{min}(p, p', c)$, it remains only to count the number of ends.

Unlike bends, ends cannot generally be counted independently for each of p 's children. Let k be the number of children of p . If $k \leq 1$, i.e. p is a leaf vertex or a vertex with only one child, then only the edge between p and p 's parent, or its interaction with the single child edge, needs to be considered in calculating the number of ends.

If $k > 1$ then the solution will, in general, be far more complicated, since the end cost of the \mathcal{C} -directed path from each child will depend on the paths from the other children. It can be shown that it is NP-hard to compute the solution with minimum end cost, even in a somewhat restricted case (See Section 4.5).

A brute force approach can be used to explore all solutions over the entire set of p 's children. There are $\binom{|\mathcal{C}|k}{k}$ such solutions, which is a number exponential in k . Sections 4.2 - 4.5 give methods to compute minimum end cost solutions for $\pi_1 - \pi_5$ that need only time polynomial in k .

Once p_{root} has been processed, the entire tree has been traversed. Now find the minimum value of $M_{min}(p_{root}, p', c)$ over all grid points in $G_\varepsilon(p_{root})$. A schematisation can be traced back through the tree, from root to leaves, following stored solutions of minimum cost at each vertex.

Since we start by finding a locally optimal solution at each leaf vertex, and then augment these solutions to find an optimal solution for the subtree rooted at every parent vertex, it is clear by induction that the solution obtained at p_{root} is optimal for the entire tree.

Time complexity. The time complexity of the algorithm depends on the specific problem being solved. However, a minimum time complexity of $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3k)$ for processing each vertex p can be noted, as in all five cases $\delta(q', p', c_1, c_2)$ values must be computed for the k children of p . This equates to $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3n)$ over an entire tree of n vertices.

4.2 Problems π_1 and π_2

The objective function $M_1(p)$ counts bends and ends only along each individual coloured path. Hence, the solution for one path has no effect on the solution for another, and they may be treated independently. $M_2(p)$ is the same, but multiple edges must be taken into account.

The algorithm. For any coloured path containing the vertex p , one can consider all solutions in polynomial time, since there are at most two edges of that colour incident to p . Find the best solution for every colour, and sum the M_1 costs to get $M_{min}(p, p', c)$ over all of p 's children.

In π_2 , there may be multiple coloured edges from p to p 's parent. In this case, consider a different parent direction for each colour in turn, and repeat the above process. As the colours are independent, choose the best solution for each and sum them to get the total M_2 cost.

Time complexity. We compute the $M_{min}(p, p', c)$ values for each $p \in \mathcal{S}$ in time $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3k)$. We visit each of the n points in the tree only a constant number of times, so a total of $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3n)$ time is needed to solve π_1 . The time complexity for π_2 increases in the worst case by a factor of $|\mathcal{P}|$, giving $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3|\mathcal{P}|n)$ overall.

4.3 Problem π_3

In order to minimise the cost function $M_3(p)$, we want to match each individual \mathcal{C} -directed path entering p with a different path coming in to p in the opposite direction. We only want to do this if the number of bends on the edges does not overcome the reduced end cost, however.

We propose a transformation from this problem to a *minimum cost maximum cardinality matching* problem. A matching on a graph is a set of

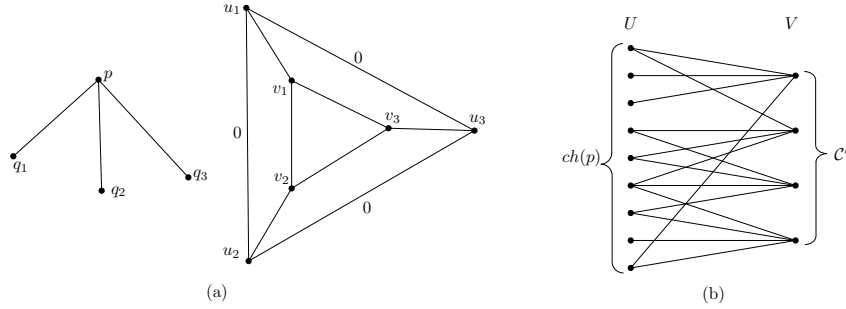


Fig. 5. (a) A vertex p with three neighbours q_1, q_2, q_3 (left) and the corresponding matching graph $G = (V, E)$ (right). (b) The bipartite matching graph $G = (U, V, E)$ constructed for π_4 . Every node in V is connected by $|V| = |C'|$ edges to nodes in U , representing the $|C'|$ lowest cost paths entering p in the corresponding direction.

vertex-disjoint edges. A maximum cardinality matching is such a set that is as large as possible. A minimum cost maximum cardinality matching is a maximum cardinality matching in which the sum of edge weights is minimal. A detailed introduction to graph matching can be found in the literature [9]. Gabow [6] details an algorithm to solve weighted matching problems for a graph $G = (V, E)$ in time $\mathcal{O}(|V|(|E| + |V| \log |V|))$.

The algorithm. Let $q_1, q_2, \dots, q_{|N(p)|}$ be the neighbouring vertices of p in T . Consider in turn each $q_i, 1 \leq i \leq |N(p)|$. For every direction $c_2 \in \mathcal{C}$, compute the minimum M_3 cost assuming that the minimum-bend \mathcal{C} -directed path from q_i to p enters p in direction c_2 and that there is no matching path in the direction opposite to c_2 . Store the minimum of these values, and the associated direction. Consider now every pair of p 's neighbours $q_i, q_j, 1 \leq i \leq |N(p)|, 1 \leq j \leq |N(p)|$, and similarly compute and store the minimum M_3 cost solution for every $c_2 \in \mathcal{C}$ assuming that the \mathcal{C} -directed path from q_i enters p in direction c_2 and the path from q_j enters in the direction opposite to c_2 . Ties for the minimum cost may be broken arbitrarily. If either q_i or q_j is p 's parent, consider only the parent direction c for that path from that vertex.

Construct a graph $G = (V, E)$ as follows (See Fig. 5(a)). Add a node v_i to V for every q_i . Add an edge to E between every pair of nodes v_i, v_j , with a weight of the minimum M_3 cost if the \mathcal{C} -directed paths from q_i and q_j enter p from opposite directions. This allows each path to be matched in G , with the same cost as if they entered p in opposite directions.

Next add a second node u_i to V for every q_i . Add an edge to E between every pair of nodes u_i, v_i with a weight of the minimum M_3 cost if the path from q_i to p is unmatched by a path entering p in the opposite direction. Finally, add a zero weight edge between every node pair u_i, u_j .

Compute a minimum cost maximum cardinality matching η on G . Transform back to the original problem by considering each pair of matched nodes in η . If both nodes correspond to the same child q_i , then the direction that gives the minimal M_3 cost independent of other children is taken for q_i 's path to p . Otherwise, the nodes correspond to two different children in the original problem, and their paths are made to enter p in the pair of opposing directions that gave minimal M_3 costs.

Time complexity. The graph G takes $\mathcal{O}(|\mathcal{C}|k^2)$ time to generate, and contains $\mathcal{O}(k)$ nodes and $\mathcal{O}(k^2)$ edges. Computing the matching therefore takes $\mathcal{O}(k^3)$ time. Hence, the total time complexity is $\mathcal{O}(\frac{r^2}{\varepsilon^4}|\mathcal{C}|^3n + \frac{r}{\varepsilon^2} \sum_{p \in \mathcal{S}} (|\mathcal{C}|d_T(p)^3 + |\mathcal{C}|^2d_T(p)^2))$, where $d_T(p)$ is the degree of p in T .

4.4 Problem π_4

In contrast to π_3 , in π_4 we need only one path entering p in a given direction to match many in the opposite direction.

Throughout this section, we say that a direction c is *used* by a schematisation if there is a \mathcal{C} -directed path in the schematisation from at least one neighbour of p that enters p in direction c . The direction c is *not used* if there is no such path.

Our approach again uses a transformation to a graph matching problem, this time matching edges to each of a subset $\mathcal{C}' \subseteq \mathcal{C}$ of directions. The matching problem assumes that an optimal solution exists that uses every direction in the given subset \mathcal{C}' , and does not use any other directions. To find a globally optimal solution, the algorithm processes all $2^{|\mathcal{C}'|}$ subsets of \mathcal{C} . We expect $|\mathcal{C}'|$ to be a small constant in most practical applications.

The algorithm. Process every subset $\mathcal{C}' \subseteq \mathcal{C}$ as follows. Let q_1, q_2, \dots, q_k be the children of p in T . For each $q_i, 1 \leq i \leq k$ and every direction $c \in \mathcal{C}'$, calculate the M_4 cost if the \mathcal{C} -directed path from q_i enters p in direction c , with no path entering p in the opposite direction. For now, store the solution that gives the lowest M_4 cost, breaking ties arbitrarily.

Once all children have been processed, an initial schematisation is constructed from the set of stored solutions. If this schematisation uses every direction $c \in \mathcal{C}'$, then we are done; there is no possibility of reducing the end count, as an end will only be counted if there is an unused direction.

If one or more directions are unused, we must satisfy our assumption of an optimal solution using exactly the set of directions \mathcal{C}' , by choosing some of the paths to enter p in the unused directions.

Compute for every direction $c \in \mathcal{C}'$ the set of $|\mathcal{C}'|$ children whose \mathcal{C} -directed paths to p give the lowest additional M_4 cost if required to enter p in direction c . Call this set $\text{ch}_c(p)$. The additional M_4 cost for each child $q \in \text{ch}_c(p)$ will be the cost difference between the initial schematisation and the one obtained if the path from q enters p in direction c .

Now construct a bipartite graph $G = (U, V, E)$ for matching (See Fig. 5(b)). For each child q_i that was added to $\text{ch}_c(p)$ for any $c \in \mathcal{C}'$, add a node u_i to U . For every direction $c_j \in \mathcal{C}'$, add a node v_j to V . Construct an edge between every pair of nodes $u_i \in U, v_j \in V$ for which it holds that $u_i \in \text{ch}_{c_j}(p)$. Set the weight of the edge u_i, v_j equal to the additional M_4 cost for the child q_i 's \mathcal{C} -directed path to enter p in direction c_j .

G now contains $\mathcal{O}(|\mathcal{C}'|^2)$ nodes and $\mathcal{O}(|\mathcal{C}'|^2)$ edges. Compute a matching from U to V as in Section 4.3. Once a matching is computed, modify the initial schematisation as follows. For each matched pair of nodes $u_i \in U, v_j \in V$, replace the initial path from q_i to p with a minimum-bend \mathcal{C} -directed path that enters p in direction v_j . All other paths remain as in the initial schematisation. We now have a schematisation that uses every direction in the set \mathcal{C}' , and is of minimal cost for \mathcal{C}' .

After computing such a schematisation for all subsets $\mathcal{C}' \subseteq \mathcal{C}$, take the schematisation that produces the minimum M_4 cost, breaking ties arbitrarily; this is an optimal solution. Note that if $k < |\mathcal{C}|$, the algorithm needs only to consider those subsets $\mathcal{C}' \subseteq \mathcal{C}$ where $|\mathcal{C}'| \leq k$.

Time complexity. The algorithm explores $\mathcal{O}(2^{|\mathcal{C}|})$ subsets of directions. For each of these subsets, we can then populate the set $\text{ch}_c(p)$ for every direction $c \in \mathcal{C}'$ in $\mathcal{O}(|\mathcal{C}|^2 k)$ time, since choosing the i th largest of n numbers can be done in $\mathcal{O}(n)$ time [1]. The graph G is constructed in $\mathcal{O}(|\mathcal{C}|^2)$ time, and $\mathcal{O}(|\mathcal{C}|^4 \log |\mathcal{C}|)$ time is needed to compute a matching from G . We therefore need $\mathcal{O}(\frac{r^2}{\varepsilon^4} |\mathcal{C}|^3 k + \frac{r}{\varepsilon^2} 2^{|\mathcal{C}|} (|\mathcal{C}|^6 \log |\mathcal{C}| + |\mathcal{C}|^4 k))$ time to process the vertex p , and $\mathcal{O}(\frac{r^2}{\varepsilon^4} |\mathcal{C}|^3 n + \frac{r}{\varepsilon^2} 2^{|\mathcal{C}|} |\mathcal{C}|^6 \log |\mathcal{C}| n)$ time for T .

4.5 Problem π_5

π_5 counts 1 end for each direction used if the opposing direction is not used, or 0 otherwise. Let $\mu(q, c_j) = M_5(q, q', c_1) + \delta(q', p', c_1, c_j)$ be the M_5 cost calculated by the algorithm of Section 4.1, before counting ends.

In order to count ends, we need only consider those directions $c_j \in \mathcal{C}$ that correspond to a cost of at most 1 more than the minimum $\mu(q, c_j)$

value. By using two opposite directions for the \mathcal{C} -directed paths from two children $q_1, q_2 \in \text{ch}(p)$, we can save at most 2 from the M_5 cost. Hence, if either path entering p in those directions had a $\mu(q, c_j)$ value of two or more greater than the optimal, there must be a pair of unopposed directions for the two paths with equal or lower M_5 cost.

Given this fact, π_5 appears somewhat simpler than π_4 . However, it can be shown to be NP-hard. Let CHILDPATHPLACEMENT be the problem of choosing a direction $c_j \in \mathcal{C}$ for the path from each $q_i \in \text{ch}(p)$ such that the corresponding schematisation gives a total M_5 cost of m .

Theorem 1. CHILDPATHPLACEMENT is NP-complete.

The proof of Theorem 1 is by a straightforward reduction to SET-COVER; we omit it here due to space restrictions (See Appendix A-1).

It is clear that CHILDPATHPLACEMENT can be solved if the π_5 problem can be solved, which leads to our final result for this section.

Corollary 1. π_5 is NP-hard.

π_5 can be solved using the same algorithm as π_4 .

References

1. M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest and R. E. Tarjan. Time bounds for selection. *Journal of Comp. and Sys. Sciences*, vol. 7, no. 4, p. 448–461, 1973.
2. U. Brandes, M. Eiglsperger, M. Kaufmann and D. Wagner. Sketch-driven orthogonal graph drawing. In *Proc. Graph Drawing 2002*, p. 1–11, 2002.
3. S. Cabello, M. de Berg and M. van Kreveld. Schematization of networks. *Computational Geometry and Applications*, vol. 30, p. 223–238, 2005.
4. S. Cabello and M. van Kreveld. Approximation algorithms for aligning points. *Algorithmica*, vol. 37, p. 211–232, 2003.
5. M. Eiglsperger, S. P. Fekete and G. W. Klau. Orthogonal graph drawing. *Drawing Graphs*, p. 121–171, Springer-Verlag Berlin Heidelberg, 2001.
6. H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. ACM-SIAM Symp. Discrete Alg.*, p. 434–443, 1990.
7. S.-H. Hong, D. Merrick and H. A. D. do Nascimento. The metro map layout problem. In *Proc. Graph Drawing 2004*, p. 482–491, 2004.
8. U. Lauther and A. Stübinger. Generating schematic cable plans using springembedder methods. In *Proc. Graph Drawing 2001*, p. 465–466, 2002.
9. L. Lovász and M. D. Plummer. *Matching Theory*. Elsevier, Amsterdam, 1986.
10. D. Merrick and J. Gudmundsson. Path simplification for metro map layout. Submitted to *Graph Drawing*, June 2006.
11. G. Neyer. Line simplification in restricted orientations. In *Proc. of the 6th International workshop on Algorithms and Data Structures*, p. 13–24, 1999.
12. M. Nöllenburg and A. Wolff. In *Proc. Graph Drawing 2005*, p. 321–333, 2006.
13. J. M. Stott and P. Rodgers. Metro map layout using multicriteria optimization. In *Proc. Information Visualisation*, p. 355–362, 2004.

A-1 Appendix: Hardness of CHILDPATHPLACEMENT

Theorem 1. CHILDPATHPLACEMENT is NP-complete.

Proof. We will refer to a solution of the CHILDPATHPLACEMENT problem as a *child path placement* of p . The reduction is from SETCOVER. Let χ be a collection of subsets of a finite set σ , and suppose we wish to determine whether a set cover of size m exists.

We construct an instance of the CHILDPATHPLACEMENT problem as follows. Consider the following tree T with root q and with $ch(q) = \{q_1, q_2, \dots, q_{|\sigma|}\}$, where each child q_i corresponds to the element $s_i \in \sigma$. Let \mathcal{C} be the set of directions such that $|\mathcal{C}| = 2|\chi|$, uniformly distributed through an angle of 2π radians. (1) Let each set $x_j \in \chi$ correspond to a single direction $c_j \in \mathcal{C}$ such that no two of these directions oppose each other. (2) For every $s_i \in \sigma$ and for every direction $c_j \in \mathcal{C}$ set: $\mu(q_i, c_j) = 0$ if $s_i \in x_j$ and $x_j \in \chi$ and $\mu(q_i, c_j) = 2$ if $s_i \in x_j$ and $x_j \notin \chi$.

The CHILDPATHPLACEMENT problem will return a direction for each child $q_i \in ch(q)$. Consider a solution for CHILDPATHPLACEMENT and let \mathcal{C}' be the set of directions used in the solution. From the above reduction it follows that \mathcal{C}' corresponds to a collection of sets that covers χ , i.e., a set cover of size $|\mathcal{C}'|$.

Hence, the question whether a set cover of size m exists of χ can be answered by answering the question whether a child path placement exists of size m . This proves NP-completeness of the decision version of the CHILDPATHPLACEMENT problem. \square