Approximation Algorithms for Finding Best Viewpoints

Michael E. Houle and Richard Webber

Department of Computer Science and Software Engineering
University of Newcastle
Callaghan 2308, Australia
{mike,richard}@cs.newcastle.edu.au

Abstract. We address the problem of finding viewpoints that preserve the relational structure of a three-dimensional graph drawing under orthographic parallel projection. Previously, algorithms for finding the best viewpoints under two natural models of viewpoint "goodness" were proposed. Unfortunately, the inherent combinatorial complexity of the problem makes finding exact solutions is impractical. In this paper, we propose two approximation algorithms for the problem, commenting on their design, and presenting results on their performance.

1 Introduction

Since it was first considered by the graph drawing community [6,10], there has been much research into three-dimensional graph drawing. There is some experimental evidence that three-dimensional graph drawings have advantages over their two-dimensional counterparts. It is claimed [16] that three dimensions allow users to work with larger graphs – the natural three-dimensional actions of rotation and translation allow a user to resolve ambiguities in large drawings while maintaining their overall mental map [12].

Ware et al. [16] conducted a series of experiments on finding paths between vertices in a three-dimensional graph drawing, under a variety of display and navigation combinations. They discovered that giving users control of bidirectional rotation results in lower error rates than continuous unidirectional rotation, but at the cost of increased decision time. In [7], we conjectured that this increase is partly due to the time taken to manually select *good viewpoints*.

Most current systems leave the selection of good viewpoints entirely to the user. We propose that the user should be able to specify those parts of the graph drawing they wish to focus on, then the system should automatically move them to a good viewpoint. For interactive applications, the movement needed to maintain the illusion of three dimensions [3] can be achieved by continuously moving between several viewpoints, or by "rocking" around a single viewpoint.

In graph drawing, Kamada and Kawai [11] model good viewpoints as those that preserve the "shape" of a three-dimensional wire-frame drawing, by excluding viewpoints from which edges appear collinear. They describe the bad

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 210–223, 1998.

viewpoints of their model as a set of great circles on the unit sphere, and present an algorithm to find the best viewpoints of a graph drawing (those viewpoints which are furthest by great-circle distance from the nearest bad viewpoint).

Bose, Gomez, Ramos and Toussaint [2] model good viewpoints for wire-frame drawings as those from which no vertex-vertex or vertex-edge pairs occlude each other, and no three edges appear to cross at the same point. They describe the bad viewpoints of their model as an *arrangement* [8] of curves on the unit sphere.

In computer graphics, *aspect graphs* [15] are used to describe sets of viewpoints from which the two-dimensional images of a three-dimensional polyhedral solid have the same topology. Aspect graphs can be used to find viewpoints from which the maximal number of faces of a polyhedron are visible; in some sense, these viewpoints can be considered "best".

In a previous paper [7], we presented three models of good viewpoints: one that distinguishes between good and bad viewpoints, and two that assign continuous measures of goodness to each viewpoint, leading to the notion of *best viewpoints*. Unfortunately, the complexity of the latter two models is too high for the calculation of theoretically-exact best viewpoints to be practical. In this paper, we address this problem by proposing fast approximation algorithms that yield "reasonably good" viewpoints.

In Sect. 2, we briefly review the models of good viewpoints detailed in [7]. In Sect. 3, we consider several criteria which algorithms for finding best viewpoints should satisfy. In Sects. 4 and 5, we present two classes of approximation algorithms and discuss their relative merits. We conclude in Sect. 6 with a discussion of the experimental performance of these algorithms, and some examples of graph drawings viewed from their best viewpoints.

2 Good Viewpoints

A three-dimensional straight-line drawing $D : V \rightarrow \mathbb{R}^3$ of an *abstract graph* $G = (V, E)$ associates a three-dimensional position $(x_i, y_i, z_i) \in \mathbb{R}^3$ with each vertex $v_i \in V$. Each edge e_{ij} is drawn as a line-segment between its endpoints. We use V_0 to denote the set of isolated vertices; for a *wire-frame* drawing, $V_0 = \emptyset$.

A three-dimensional graph drawing is mapped to a two-dimensional image via a *projection*. In this paper, we consider only *orthographic parallel projections* [9]. An orthographic parallel projection is parameterised by its *viewpoint direction* – a vector from the origin in $\mathbb{R}^3$ to a point p on the unit sphere. A two-dimensional image is formed by translating each point of the three-dimensional drawing, parallel to the vector p, onto a plane (the *projection surface*) that is perpendicular to p. The drawing can be *clipped* by a volume before projection – those portions of the three-dimensional graph drawing outside of this clipping volume do not appear in the resulting two-dimensional image.

If a projection maps two three-dimensional points to the same two-dimensional point, then an *occlusion* occurs. We say the *front* point *occludes* the *rear* point. The concept of occlusion underlies many models of good viewpoints.

2.1 Bad Viewpoint Arrangements

Definition 21 *A good viewpoint is one from which the apparent abstract graph of the two-dimensional image is identical to the abstract graph of the three-dimensional graph drawing.*

For the purposes of this paper we assume that: a two-dimensional image is generated from a three-dimensional graph drawing using an orthographic parallel projection; there is no clipping; and all vertices and edges are mathematical ideals, with zero width for the purpose of calculating occlusions. Under these assumptions, the abstract graphs of a three-dimensional graph drawing and its two-dimensional image appear the same, if and only if there are no occlusions involving elements of the drawing. Viewpoints that result in occlusions are *bad viewpoints*; $\psi(a, b)$ denotes the set of bad viewpoints from which a occludes b.

There are four main types of occlusions: *vertex-vertex occlusions*, *vertex-edge occlusions*, *edge-vertex occlusions*, and *edge-edge occlusions*. If v_i and v_j are both isolated vertices, then $\psi(v_i, v_j)$ is called an *isolated-vertex occlusion*. Edge-edge occlusions only affect the apparent abstract graph if the corresponding two-dimensional edges overlap (intersect at more than one point).

Observation 22 *A good viewpoint is one that does not generate any isolated-vertex, vertex-edge, or edge-vertex occlusions.*

The set of bad viewpoints corresponding to a three-dimensional graph drawing can be represented by a collection Ψ of *occlusion curves* on the unit sphere. Using the techniques described in Bose et al. [2], we construct an arrangement [8] of the elements in Ψ. We refer to this structure as a *bad viewpoint arrangement*.

Lemma 23 [2] *For a given three-dimensional graph drawing, we can build the corresponding bad viewpoint arrangement S in $\mathrm{O}(|\Psi| \log |\Psi| + k)$ time, where $|\Psi| = |V_0|(|V_0| - 1) + 2|V||E|$. The parameter k is the number of intersections between elements of Ψ, which is $\mathrm{O}(|V_0|^2 + |V|^2|E|^2)$ in the worst case. The size of S is $\mathrm{O}(|\Psi| + k)$.*

2.2 Rotational Separation Diagrams

A bad viewpoint arrangement allows us to determine whether a given viewpoint results in an occlusion that affects the apparent abstract graph of a drawing. However, in itself, a bad viewpoint arrangement does not tell us from which points it would be best to view the drawing. To address this issue, we have developed two measures of *goodness* over the set of viewpoints. The first of these is the *rotational separation* measure.

Let $\delta(p, p')$ denote the great-circle distance between two viewpoints p and p'. We define $\mathcal{G}_{\mathrm{rsd}}(p, \psi(a, b))$ to be $\min_{p' \in \psi(a,b)} \delta(p, p')$. The rotational separation $\mathcal{G}_{\mathrm{rsd}}(p, \Psi)$ of p is $\min_{\psi(a,b) \in \Psi} \mathcal{G}_{\mathrm{rsd}}(p, \psi(a, b))$. If this minimum value is achieved for $\psi(a', b')$, then we say that $\psi(a', b')$ *determines* the goodness of p. A *rotational separation diagram* is a variant of a *Voronoi diagram* [8] that uses rotational separation as its distance function.

Lemma 24 *A rotational separation diagram can be constructed in* $O(|S| \log |S|)$ *time, where* S *is the corresponding bad viewpoint arrangement. The resulting diagram can be used to calculate* $\mathcal{G}_{\text{rsd}}(p, \Psi)$ *in logarithmic time.*

A rotational separation diagram can also be used to the find *best viewpoints* of a three-dimensional graph drawing under the rotational separation measure. Clearly, the goodness value $\mathcal{G}_{\text{rsd}}(p, S)$ increases as p moves away from the nearest bad viewpoint(s) in S. This increase is maximised locally, either at a Voronoi vertex of the rotational separation diagram, or in some cases, at a unique internal point of a Voronoi edge.

Theorem 25 *Given a rotational separation diagram, we can find all locally-best viewpoints in* $O(|S|)$ *time, where* S *is the corresponding bad viewpoint arrangement. We can also find a locally-best viewpoint, nearest by great-circle distance to a given viewpoint, in logarithmic time.*

Best viewpoints under the rotational separation measure maximise the amount by which the viewpoint can rotate before an occlusion is generated. This can be beneficial for interactive applications, especially those that use "rocking" to achieve the illusion of three-dimensions [3].

2.3 Observed Separation Diagrams

Our second continuous measure of goodness is *observed separation*. We define $\mathcal{G}_{\text{osd}}(p, \psi(a, b))$ to be the minimum distance between the images of graph elements a and b when viewed from the viewpoint p. The observed separation $\mathcal{G}_{\text{osd}}(p, \Psi)$ of p is $\min_{\psi(a,b)\in\Psi} \mathcal{G}_{\text{osd}}(p, \psi(a, b))$. An *observed separation diagram* is a Voronoi diagram that uses observed separation as its distance function.

A tight bound for the worst case size of an observed separation diagram is currently an open problem. An $O(|S|^2 2^{\alpha(|S|)})$ upper bound is known for the restricted case of three-dimensional point sets ($E = \emptyset$), where α is the inverse of *Ackermann's function* [1]. A bad viewpoint arrangement S exists which yields an $\Omega(|S|^2)$ lower bound; however, it is not yet known whether such an arrangement can be derived from a three-dimensional graph drawing.

Best viewpoints under the observed separation measure maximise the user's ability to resolve elements in the two-dimensional image of a three-dimensional drawing. Example drawings (such as those in Fig. 6) suggest that best viewpoints under observed separation are superior to those under rotational separation for the static display of three-dimensional graph drawings.

3 Approximate Solutions

We propose that, for a viewpoint-finding algorithm to be useful, it should satisfy these criteria:

Quality – The viewpoint found should be "equivalent" to a theoretically-exact best viewpoint. Exactly what it means for two viewpoints to be "equivalent" varies among applications and users.If our output device has finite resolution,

then equivalence can be quantified according to the threshold angle below which a given three-dimensional point is expected to map to the same pixel in both of the resulting two-dimensional images. This threshold angle is equivalent to half the minimum angle of rotation θ that would result in a vertex v_i shifting from one edge of a pixel in the two-dimensional projection to the opposite edge of the same pixel. Hence, on an $n \times n$ pixel display, two viewpoints can be considered "equivalent" when the angle between them is less than $\frac{\theta}{2} = \arcsin \frac{2}{n}$.

Locality – The viewpoint p' found should be "close" to the user's specified viewpoint p. This criterion is important to help preserve the user's mental map of the three-dimensional graph drawing. One may set a tolerance radius ϵ for the change in viewpoint $\delta(p, p')$. By restricting p' to satisfy $\delta(p, p') \leq \epsilon$, we ensure that the user's mental map is preserved. Alternatively, if $\delta(p, p') > \epsilon$, then the movement can be animated such that the change of viewpoint between two frames is always within the radius ϵ. This introduces the notion of two tolerance radii: ϵ_1, the maximum change allowed over the entire animation; and ϵ_2, the maximum change allowed in any step of the animation.

The application of a viewpoint-finding algorithm occurs in two phases. In the first phase, a three-dimensional graph drawing is preprocessed to build a data structure which identifies its best viewpoints. In the second phase, this data structure is repeatedly queried, to find viewpoints that satisfy the above criteria. These two phases suggest two additional criteria.

Preprocessing Speed – Preprocessing should be fast enough to allow access to the query algorithm within a "reasonable time". Ideally, the algorithm should be available as soon as the drawing is loaded into the application. To achieve this, preprocessing can be carried out when the drawing is generated, and the resulting data structures loaded into the application along with the drawing. However, applications that generate graph drawings interactively must still perform preprocessing "on the fly".

Query Speed – The query algorithm should return a viewpoint "quickly". Ideally, a viewpoint should be found in no more time than it takes to render the drawing. Rendering a graph as a simple line-drawing usually requires time linear in the number of graph elements $|G| = |V| + |E|$.

The algorithms for finding best viewpoints given in Sects. 2.2 and 2.3 return best viewpoints within the precision used to build the corresponding diagrams. This easily satisfies the quality criterion for any reasonable precision. These algorithms also satisfy the locality criterion, in the sense that they return the locally-best viewpoint nearest by great-circle distance to a given viewpoint. The query speed criterion is also satisfied – indeed over-satisfied – as these algorithms return a viewpoint in logarithmic time. However, these algorithms require extensive preprocessing before their query algorithms can be used. For a given graph G, preprocessing can require $\Omega(|G|^4 \log |G|)$ time under the rotational separation measure, and even more under the observed separation measure. These time requirements are clearly too large for these algorithms to be useful in practice.

In order to reduce the computational cost associated with finding a locally-best viewpoint, we present algorithms that relax the quality criterion to find *"reasonably good" viewpoints*, while satisfying the remaining criteria.

4 Iterative Improvement Algorithms

Iterative improvement [14] is a simple search technique used to find points in a given space that optimise a given function. An iterative improvement algorithm works by repeatedly selecting a trial point and evaluating the optimisation function, retaining the point which yields the best value. The choice of the optimisation function, the way in which a trial point is chosen, and the way in which the decision is made to terminate, all influence the running time of the algorithm, and the quality of the solution produced. For the problem at hand, the optimisation function is either of the goodness measures, denoted $\mathcal{G}(p, \Psi)$.

Various methods can be used to choose a trial viewpoint. Choosing a viewpoint p' at random from the set of all viewpoints (*blind random search*) does not satisfy the locality criterion, as the initial viewpoint has no effect on the final viewpoint. A better method is to choose a trial viewpoint p' from within the intersection of two limiting circles: one centred on the initial viewpoint, with radius ϵ_1; and one centred on the current viewpoint, with radius ϵ_2 (recall that ϵ_1 and ϵ_2 are the tolerance radii that preserve the user's mental map).

Calculating $\mathcal{G}(p, \Psi)$ requires $O(|G|^2)$ time. It follows that t should be in $O(1/|G|^2)$ to satisfy the query speed criterion. However, for large graph drawings, t may become too small to find a "reasonably good" viewpoint within this time. This problem can be (partially) solved by animating the algorithm.

Algorithm 1 Animated Iterative Approximation

Inputs*: initial viewpoint p; goodness function $\mathcal{G}$; occlusion curves Ψ; limiting radii ϵ_1, ϵ_2; number of iterations t.*

Outputs*: final viewpoint.*

1. *Limiting centre $c_1 \leftarrow p$.*
2. **While** *p is* **not** *"reasonably good":*
   (a) *Limiting centre $c_2 \leftarrow p$.*
   (b) **For** *t iterations:*
      i. *Choose $p' \neq p$, such that $\delta(c_1, p') \leq \epsilon_1$,* **and** *$\delta(c_2, p') \leq \epsilon_2$.*
      ii. **If** *$\mathcal{G}(p', \Psi) > \mathcal{G}(p, \Psi)$,* **then** *$p \leftarrow p'$.*
   (c) *Display the graph drawing from the viewpoint p.*
3. **Return** *p.*

On each step of the animation, t trial viewpoints are assessed, requiring $O(t|G|^2)$ time; then the graph is displayed. Between steps, the centre of the inner limiting circle moves to the current viewpoint p. Alternatively, the inner limiting circle can be moved each time a better viewpoint is found.

Explicitly calculating whether the current viewpoint p is "reasonably good" is difficult, as there is no efficient way of comparing p against the theoretically-exact best viewpoints of the graph drawing. Instead, we use heuristics to decide

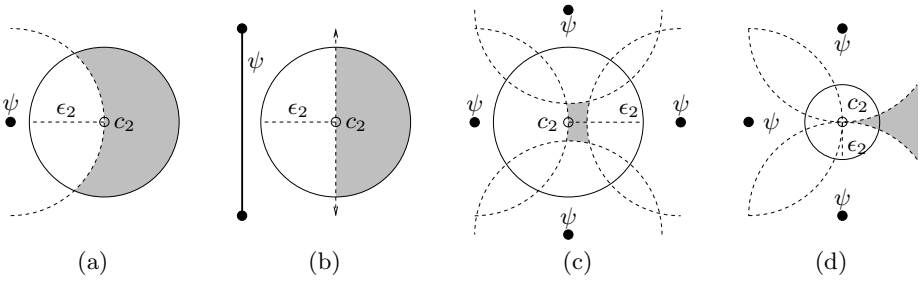


Fig. 1. Regions of better viewpoints within a limiting circle.

whether p is "reasonably good". Highly interactive applications can continue searching for a better viewpoint until the user interrupts the process, placing the burden of determining what is "reasonably good" on the user. Other applications can terminate the while-loop when the current viewpoint p does not change for a set number of trials. We use (c, ϵ) to denote a lesser circle, centred on c, with radius ϵ. Let B be the region of viewpoints p' within (c_2, ϵ_2) such that $\mathcal{G}(p', \Psi) > \mathcal{G}(p, \Psi)$, let the variable $R = \frac{|B|}{|(c_2,\epsilon_2)|}$ be the proportion of the area of (c_2, ϵ_2) taken up by B, and let T be the random variable of the number of trials in which p has not changed. It can be shown that

$$\boldsymbol{E}[T] = \frac{1}{R} - 1 \quad \text{and} \quad \boldsymbol{Pr}[T \geq t \mid R \geq k] \leq (1-k)^t \ .$$

If p is within a small distance of an occlusion curve, then $R \geq 0.5$ (as illustrated in Figs. 1a and 1b). It follows that the iterative approximation algorithm moves the current viewpoint quickly away from bad viewpoints. However, as p approaches a locally-best viewpoint, R approaches 0, and the expected number of trials needed to improve on the current viewpoint increases dramatically.

To avoid the situations in which R approaches 0, when the viewpoint p has not changed for a set number of trials, the inner limiting radius ϵ_2 is changed in an attempt to increase the value of R. There are two cases to consider:

1. If B is relatively small and is wholly contained within (c_2, ϵ_2), then decreasing ϵ_2 makes (c_2, ϵ_2) a tighter approximation to B (see Fig. 1c).
2. If the current viewpoint is on a "spike" of B, and B is a small part of a larger region of better viewpoints, then increasing ϵ_2 can increase the proportion of this region in (c_2, ϵ_2) (see Fig. 1d).

Obviously, these two cases conflict, and it cannot be determined with certainty which case applies for any given viewpoint. Our approach is to use a dynamic inner limiting radius ϵ_2', initially set to ϵ_2. At each step, if p does not change, then ϵ_2' is decreased. Otherwise, ϵ_2' is increased, but is never set higher than ϵ_2. If ϵ_2' reaches a lower limit ϵ_θ (determined by the "equivalence" angle of the quality criterion), then the algorithm decides that it has achieved a "reasonably good" viewpoint and terminates.

4.1 Pruning

The iterative improvement algorithm described so far is a practical method for approximating the best viewpoints of small three-dimensional graph drawings. However, for larger graph drawings, the $O(|\Psi|)$ time taken to compute the goodness of each trial viewpoint becomes too great. One approach to this problem is to *prune* those occlusions curves which cannot possibly determine the goodness of a trial viewpoint. Let $\mathcal{G}_{\min}(\psi(a,b),(c,\epsilon)) = \min_{p\in(c,\epsilon)} \mathcal{G}(p,\psi(a,b))$ and $\mathcal{G}_{\max}(\psi(a,b),(c,\epsilon)) = \max_{p\in(c,\epsilon)} \mathcal{G}(p,\psi(a,b))$. Initially, we prune Ψ to yield

$$\Psi_1 = \left\{ \psi(a,b) \;\middle|\; \mathcal{G}_{\min}(\psi(a,b),(c_1,\epsilon_1)) \leq \min_{\psi(a',b')\in\Psi} \mathcal{G}_{\max}(\psi(a',b'),(c_1,\epsilon_1)) \right\} .$$

The time taken to generate Ψ_1 is $O(|\Psi|)$. This cost is paid at the beginning of each query, after which each viewpoint can be evaluated in $O(\Psi_1)$ time. If ϵ_2' is significantly smaller than ϵ_1, then a second stage of pruning can further improve the running time of our algorithm. At the start of each animation step, we can prune Ψ_1 by the inner limiting circle (c_2,ϵ_2') to yield Ψ_2. This second pruning is advantageous when $\frac{|\Psi_2|}{|\Psi_1|} < 1 - \Theta(\frac{1}{t})$, where t is the number of trials per step.

4.2 Clipping

While calculating $\mathcal{G}(p,\Psi)$, the bad viewpoint p' that minimises $\mathcal{G}(p,\Psi)$ is implicitly identified. It is natural to expect that, if a better viewpoint than p exists, then it should lie in the direction from p opposite to that of p'. This intuition is supported by the observations in Fig. 1. A worthwhile heuristic then, is to restrict the choice of trial viewpoints to those viewpoints in that half-disc of (c_2,ϵ_2') that lies furthest from p'. This *single-clipping* heuristic potentially doubles the probability of generating a better viewpoint on any given trial.

The probability of generating a better viewpoint may be further increased by also considering the bad viewpoint p'' that results in the second (possibly equally) minimal value of $\mathcal{G}(p,\Psi)$. If the goodness values implied by p' and p'' are approximately equal, then the choice of trial viewpoint is restricted to those viewpoints in the intersection of the two half-discs of (c_2,ϵ_2') that lie furthest from p' and p'', respectively. This *double-clipping* heuristic is particularly effective at increasing the probability R when the current viewpoint is in a "spike" (Fig. 1d).

Unfortunately, in some situations, these clipping heuristics can result in all viewpoints better than p being excluded from the set of trial viewpoints. To allow for this possibility, if no improvement in the current viewpoint occurs within a set number of trials, then the clipping heuristics are disabled, until such time as a better viewpoint is found.

5 Force-Directed Algorithms

Force-direction is a well-established paradigm in the automatic layout of graph drawings [4,10]. Force-directed algorithms model graph drawings as physical systems – they calculate the forces applied to a vertex of the system by all other

vertices and edges, then moving the vertex in the direction resulting from the combination of these forces. Using force-directed methods to find best viewpoints is somewhat simpler, as this only requires movement of the current viewpoint.

The force calculations used to find best viewpoints are based on the double-clipping heuristic described earlier. The force applied by a bad viewpoint on the current viewpoint p is directed along the great-circle arc from the bad viewpoint through p. Let p' and p'' be two bad viewpoints from distinct occlusion curves that determine the two smallest value of $\mathcal{G}(p, \Psi)$, such that $\mathcal{G}(p, p') \leq \mathcal{G}(p, p'')$. If $\mathcal{G}(p, p') \approx \mathcal{G}(p, p'')$, then the movement of p is directed by the average of the forces applied by both p' and p''; otherwise, only p' is used. In the unlikely event that the forces applied by p' and p'' exactly cancel each other out, the unwanted stability is avoided by means of a small random shift in viewpoint.

Once a direction has been decided, we must determine the distance $\epsilon_3 \leq \epsilon_2$ by which to move the current viewpoint. We base our approach on that of Bruß and Frick [4]. Initially, we use a small distance ϵ_3. On all subsequent moves, we adjust the value of ϵ_3: increasing it if the current move is in the same direction as the previous move, and decreasing it if the current move is in the opposite direction. The small initial value of ϵ_3 is chosen to prevent the viewpoint moving out of its cell in the bad viewpoint arrangement. This helps to satisfy the locality criterion, independent of the choice of an outer limiting radius ϵ_1. If the viewpoint jumps outside its cell, then the algorithm can stabilise at a locally-best viewpoint far from the initial viewpoint (but still within ϵ_1).

Algorithm 2 Force-Directed Approximation

Inputs*: initial viewpoint p; goodness function $\mathcal{G}$; occlusion curves Ψ; limiting radii $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_\theta$; number of moves t.*

Outputs*: final viewpoint.*

1. *Limiting centre $c_1 \leftarrow p$; direction $d' \leftarrow 0$; limiting radius $\epsilon'_3 \leftarrow \epsilon_3$.*
2. **While** $\epsilon'_3 > \epsilon_\theta$*:*
   (a) *Limiting centre $c_2 \leftarrow p$.*
   (b) **For** t *moves:*
      i. *Find a bad viewpoint $p' \in \Psi$, that minimises $\mathcal{G}(p, p')$.*
      ii. *Find a bad viewpoint $p'' \neq p' \in \Psi$, that minimises $\mathcal{G}(p, p'')$.*
      iii. **If** $\mathcal{G}(p, p') \approx \mathcal{G}(p, p'')$, **then** $d \leftarrow direct(p, p', p'')$, **else** $d \leftarrow direct(p, p')$.
      iv. $p \leftarrow p + \epsilon'_3 d$.
      v. *Clip p by (c_1, ϵ_1) and (c_2, ϵ_2).*
      vi. **If** $d \sim d'$, **then** $\epsilon'_3 \leftarrow \min(increase(\epsilon'_3), \epsilon_2)$.
      vii. **If** $d \sim -d'$, **then** $\epsilon'_3 \leftarrow \min(decrease(\epsilon'_3), \epsilon_3)$.
      viii. $d' \leftarrow d$.
   (c) *Display the graph drawing from the viewpoint p.*
3. **Return** p.

5.1 Randomised Force-Direction

Like the iterative improvement algorithm, the force-directed algorithm requires $O(\Psi)$ time to calculate each move. A first stage of pruning can be used to decrease

the number of occlusion curves to be considered in each move; however, a second stage of pruning is seldom effective, as there are often only a few moves before the viewpoint reaches the edge of the inner limiting circle.

Another approach, which proves to be highly effective in reducing the time taken to calculate each move, is *randomisation* [13]. When processing the initial viewpoint, rather than using all the occlusion curves in Ψ, we use a random sample Ψ_S of occlusion curves chosen uniformly from Ψ. Of these, the occlusion curves that imply the worst $w \geq 2$ goodness values are retained in a list Ψ_W. After each move, a new random sample is taken from Ψ and used to update Ψ_W.

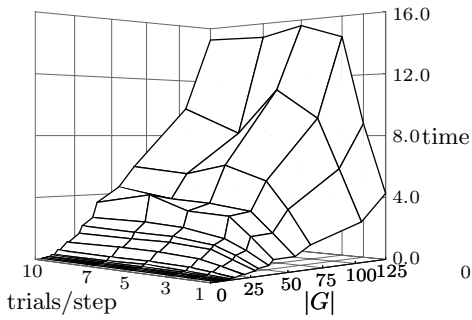
The randomised force-directed algorithm behaves identically to the deterministic version as long as the two bad viewpoints most restrictive to the current viewpoint are on occlusion curves contained in Ψ_W. If we assume that the current viewpoint is static, then the expected number of moves until both of these occlusion curves are discovered is at most $\frac{2|\Psi|}{|\Psi_S|}$. If the current viewpoint moves, then the two most restrictive occlusion curves can change. In practice, even if w is chosen to be a constant such as 10, the two most restrictive curves are almost always found in Ψ_W, due to the restriction imposed on each move by ϵ_2.

The time required to calculate each move is $O(|\Psi_S| + w \log w)$. If w is chosen to be a constant, then the expected work done to find the two most restrictive occlusion curves is $\frac{2|\Psi|}{|\Psi_S|}O(|\Psi_S|) = O(|\Psi|)$., after which the algorithm is expected to converge to a locally-best viewpoint. Ideally, to satisfy the query speed criterion, we would like $|\Psi_S|$ in $O(|G|)$. However, the expected number of moves before the algorithm finds the two most restrictive bad viewpoints could then be as large as $O(|G|)$. Rather, we choose $|\Psi_S|$ in $O(|\Psi|^{\frac{3}{4}})$; the expected number of moves is then no larger than $O(\sqrt{|G|})$. The time taken by each move is $O(|G|^{\frac{3}{2}})$ for sparse graphs ($|E|$ in $O(|V|)$), and $O(|G|^{\frac{9}{8}})$ for dense graphs ($|E|$ in $\Theta(|V|^2)$).

6 Experimental Results

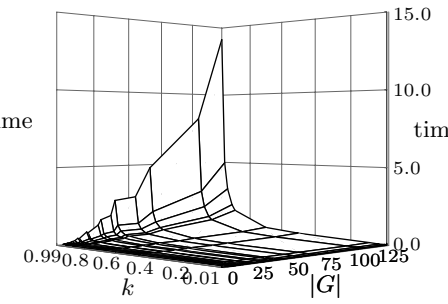
In this section, we present some experimental results on the performance of our approximation algorithms. All results were obtained using a set of randomly generated three-dimensional graph drawings. Vertices were chosen uniformly at random within the unit sphere, and edges were then chosen uniformly at random from the set of all possible edges on these vertices. The algorithms were run several hundred times on each drawing, using randomly generated initial viewpoints, and the averages taken over these runs. In this paper, we limit ourselves to the rotational separation measure; initial results for the observed separation measure are similar.

Figure 2 shows the time taken for the iterative approximation algorithm to terminate, plotted against $|G|$, and the number of trials in each animation step. These values were generated using $\epsilon_1 = \frac{\pi}{6}$, $\epsilon_2 = \frac{\pi}{30}$, and terminating when the viewpoint remained static for 54 trials; this number was chosen as it is the number of trials needed to reduce $\frac{\pi}{6}$ to $\arcsin(\frac{2}{1024})$ when ϵ_2' is reduced by 10% on each unsuccessful trial. The time taken for the algorithm to terminate increases rapidly as the number of trials per step increases from 1. This increase peaks at 5,



time v. trials per step v. $|G|$

Fig. 2. Varying the trials per step.



time v. k v. $|G|$

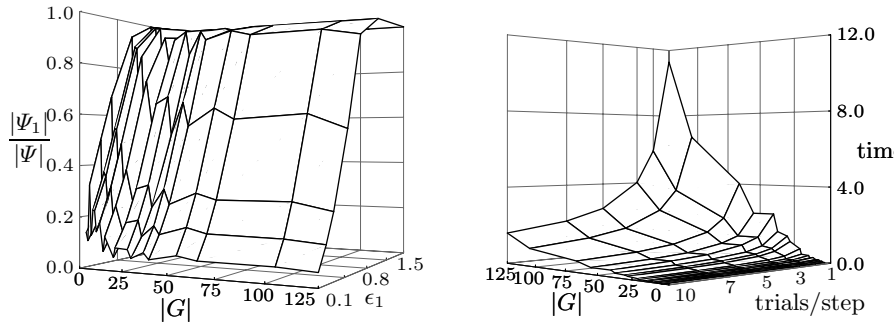
Fig. 3. Varying convergence rate.

beyond which the time taken appears to be relatively independent of the number of trials per step. It follows that (if pruning is not used) the inner limiting circle should be updated as soon as a better viewpoint is found.

Figure 3 shows the effects of altering the inner limiting radius on the time taken for the iterative approximation algorithm to terminate. These results were generated using $\epsilon_1 = \epsilon_2 = \frac{\pi}{6}$, and terminating when $\epsilon_2' < \arcsin\left(\frac{2}{1024}\right)$. The inner limiting radius ϵ_2' was multiplied by the *convergence parameter* k on each unsuccessful trial, and multiplied by $\frac{2\epsilon_2'}{k}$ on each successful trial. The parameter k was varied between 0.01 and 0.99. The time taken to terminate increases slightly super-linearly in terms of $|G|$, but exponentially in terms of k. Based on this figure, one might be tempted to choose a very low value for k. Unfortunately, as k decreases, the chance of the algorithm terminating before a "reasonably good" viewpoint is reached increases.

Figure 4a shows the effectiveness of pruning, measured by the ratio $\frac{|\Psi_1|}{|\Psi|}$, plotted against ϵ_1 and $|G|$. These values were generated by pruning with a limiting circles of radius ϵ_1, centred on viewpoints chosen uniformly at random from the unit sphere. The value of ϵ_1 was varied between 0.1 and 1.5. As expected, the effectiveness of pruning increases (the ratio $\frac{|\Psi_1|}{|\Psi|}$ decreases) as ϵ_1 decreases. For small graph drawings, the effectiveness of pruning varies significantly, reflecting its dependence on the exact configuration of the occlusion curves. For larger drawings, the effectiveness of pruning appears to be independent of $|G|$.

Figure 4b shows the effectiveness of pruning, measured by the time taken for the iterative approximation algorithm to terminate when two stages of pruning are used. These results were generated using $\epsilon_1 = \frac{\pi}{6}$, $\epsilon_2 = \frac{\pi}{30}$, and $k = 0.9$. The number of trials per step was varied between 1 and 10. As the number of trials per step increases, there is a corresponding decrease in the time taken for the algorithm to terminate, relative to the size of the graph. To understand this



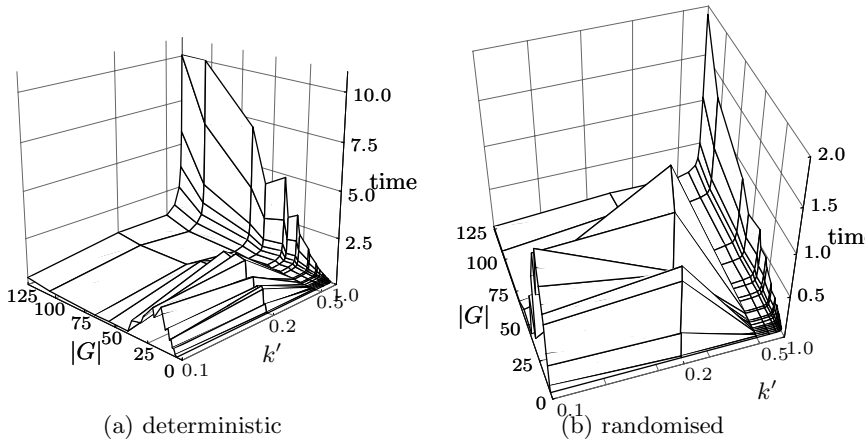
(a) $\frac{|\Psi_1|}{|\Psi|}$ v. ϵ_1 v. $|G|$ (b) time v. trials per step v. $|G|$

Fig. 4. The effectiveness of pruning.

behavior, recall that the work done in each animation step is dominated by the second stage of pruning, which is amortised over the number of trials. It follows that the effective cost of pruning can be reduced, by increasing the number of trials per step. However, for small drawings, the number of trials per step needed to make pruning worthwhile can result in many of these trials being wasted.

Finally, Figs. 5a and 5b show the time taken for the force-directed algorithm to terminate, using the deterministic and randomised approaches, respectively. These results were generated using $\epsilon_1 = \frac{\pi}{6}$, $\epsilon_2 = \frac{\pi}{30}$, $\epsilon_3 = \frac{\pi}{300}$, and terminating when $\epsilon_3' < \arcsin\left(\frac{2}{1024}\right)$. The distance functions were set such that $\text{decrease}(\epsilon_3') = k'\epsilon_3'$, and $\text{increase}(\epsilon_3') = \frac{\epsilon_3'}{k'}$. The convergence parameter k' was varied between 0.1 and 0.99. The sample sizes were set to $|\Psi_S| = |\Psi|^{\frac{3}{4}}$, and $w = 10$. For large values of k', the time taken for these algorithms to terminate exhibits a slightly super-linear dependence on $|G|$. When $|G|$ is fixed, as k' decreases from 1, the time taken decreases exponentially. This behaviour is analogous to that exhibited by iterative approximation in Fig. 3. However, for small graph drawings with $k' < 0.5$, the performance of these algorithms degrades and becomes erratic – this is especially true of the randomised algorithm. This occurs because the low value of k' causes the viewpoint to jump from cell to cell of the bad viewpoint arrangement. For small drawings, this behaviour can persist for many moves. For larger drawings, the cells of the bad viewpoint arrangement are usually too small for this behaviour to impact on the average performance. The results indicate that, as long as $\frac{1}{k'}$ is set to some small proportion of $|G|$, the force-directed algorithms provide a fast and reliable means of finding best viewpoints.

To summarise, both the iterative improvement and force-directed approaches result in useful algorithms for finding "reasonably good" viewpoints for three-dimensional graph drawings. Both approaches benefit from the use of a dynamic inner limiting radius, as long as the convergence parameters are set sufficiently close to 1 to ensure a "reasonably good" viewpoint is reached. When applied to large graph drawings, the iterative improvement approach can benefit from the



(a) deterministic (b) randomised

Fig. 5. The performance of force-directed approximation.

use of pruning. Similarly, the force-directed algorithms can benefit from the use of randomisation. Indeed, the randomised force-directed algorithm out-performs the other algorithms presented in this paper by a significant margin.

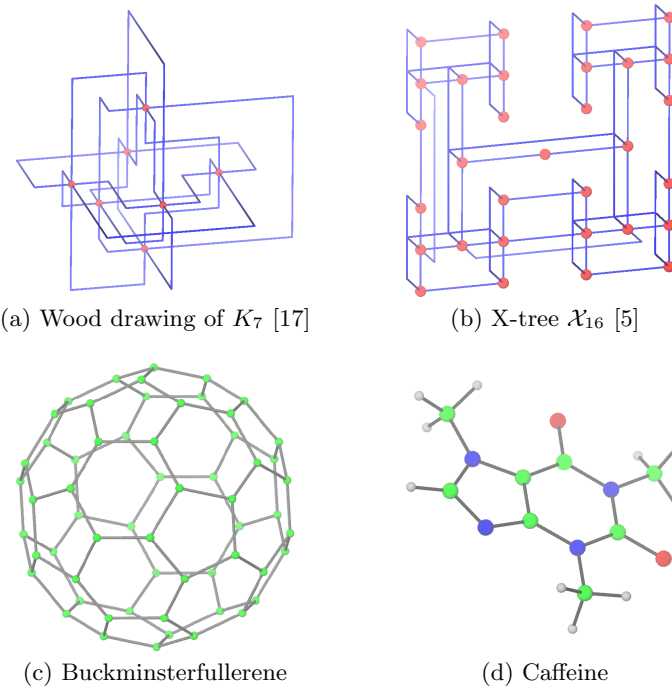
We conclude with several examples of three-dimensional graph drawings, viewed from their best viewpoints.

Acknowledgements

Thanks to Peter Eades for many helpful suggestions. Also thanks to Titto Patrignani, David Wood, Tiziana Calamoneri, Annalisa Massini and Mark Najork, for contributing three-dimensional graph drawings.

References

1. P. K. **Agarwal** (1991): *Intersection and Decomposition Algorithms for Planar Arrangements*; Cambridge University Press
2. P. **Bose**, F. **Gomez**, P. **Ramos**, G. **Toussaint** (1995): "Drawing Nice Projections of Objects in Space" in *Proc. 3rd Int. Symp. Graph Drawing* (Passau, Germany); Springer-Verlag, *LNCS*; 1027:52–63
3. M. L. **Braunstein** (1976): *Depth Perception through Motion*; Academic Press
4. I. **Bruß**, A. K. **Frick** (1995): "Fast Interactive 3-D Graph Visualization" in *Proc. 3rd Int. Symp. Graph Drawing* (Passau, Germany); Springer-Verlag, *LNCS*; 1027:99–110
5. T. **Calamoneri**, A. **Massini** (1997): "On Three-Dimensional Layout of Interconnection Networks" in *Proc. 5th Int. Symp. Graph Drawing* (Rome); Springer-Verlag, *LNCS*; 1353:64–75
6. R. F. **Cohen**, P. D. **Eades**, T. **Lin**, F. **Ruskey** (1997): "Three-Dimensional Graph Drawing" in *Algorithmica*; 17(2):199–208



(a) Wood drawing of K_7 [17] (b) X-tree $\mathcal{X}_{16}$ [5]

(c) Buckminsterfullerene (d) Caffeine

Fig. 6. Some example graph drawings viewed from their best viewpoints.

7. P. D. **Eades**, M. E. **Houle**, R. **Webber** (1997): "Finding the Best Viewpoints for Three-Dimensional Graph Drawings" in *Proc. 5th Int. Symp. Graph Drawing* (Rome); Springer-Verlag, *LNCS*; 1353:87–98
8. H. **Edelsbrunner** (1987): *Algorithms in Combinatorial Geometry*; Springer-Verlag
9. J. D. **Foley**, A. van **Dam**, S. **Feiner**, J. **Hughes** (1990): *Computer Graphics: Principles and Practice*, 2nd ed.; Addison-Wesley
10. T. M. J. **Fruchterman**, E. M. **Reingold** (1991): "Graph Drawing by Force-Directed Placement" in *Software – Practice and Experience*; 21(11):1129–1164
11. T. **Kamada**, S. **Kawai** (1988): "A Simple Method for Computing General Position in Displaying Three-Dimensional Objects" in *Computer Vision, Graphics and Image Processing*; 41(1):43–56
12. K. **Misue**, P. D. **Eades**, W. **Lai**, K. **Sugiyama** (1995): "Layout Adjustment and the Mental Map" in *J. Visual Languages and Computing*; 6:183–210
13. R. **Motwani**, P. **Raghavan** (1995): *Randomized Algorithms*; Cambridge University Press
14. P. H. J. M. **Otten**, L. P. P. P. van **Ginneken** (1989): *The Annealing Algorithm*; Kluwer Academic
15. H. **Plantinga**, C. R. **Dyer** (1990): "Visibility, Occlusion, and the Aspect Graph" in *Int. J. Computer Vision*; 5(2):137–160
16. C. **Ware**, G. **Franck** (1996): "Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions" in *ACM Trans. Graphics*; 15(2):121–140
17. D. R. **Wood** (1998): "Two-Bend Three-Dimensional Orthogonal Grid Drawing of Maximum Degree Five Graphs"; in this proceedings