Untangling graphs representing spatial relationships driven by drawing aesthetics

Charis Papadopoulos
Department of Mathematics
University of Ioannina
GR-45110, Ioannina, Greece
charis@cs.uoi.gr

Costas Voglis
Department of Computer Science
University of Ioannina
GR-45110, Ioannina, Greece
voglis@cs.uoi.gr

ABSTRACT

Representing relational data, modeled as a graph, provides visual insight into several application areas. In practice, however, data may contain small but significant errors mainly due to human interaction. Here we address the problem of correcting misplaced edges of a given graph based on straight-line drawings in a plane. In such terms seeking for a solution on graphs that have no repeated pattern nor regular structure seems inapplicable. Therefore we focus on structured graphs representing spatial relationships, that arise in a wide range of applications, and we consider the quality of a drawing as a measure of a graph's correctness. To define an ordering among the modified graphs, we formalize the evaluation of a drawing with respect to certain aesthetic criteria. We give a polynomial-time algorithm that computes a modified graph with a better layout than the original graph when only single-edge replacements are allowed. We study the behavior of the algorithm and illustrate its results on several test sets taken from a sparse matrix collection. In all cases the proposed algorithm manages to identify and correct the misplaced edges within a small number of modifications.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations, Pattern matching*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

graph drawing, data correction, edge modifications, drawing aesthetics, force directed placement

1. INTRODUCTION

PCI 2013 September 19 - 21 2013, Thessaloniki, Greece

http://dx.doi.org/10.1145/2491845.2491853

Automated graph drawing appears in many application areas with an increasing interest on its theoretical and algorithmic aspects [3, 23]. The main goal is to represent relational data, modeled as a graph, in a meaningful way that reveals an underlying structure. Relational data that need this kind of visualization arise from several application areas such as, database design, data mining, structural and software engineering, and social networks (see [3, 23]). One recent exposition is the usage of visualization techniques through concentric drawings that aim to detect user activities that appear at regular time intervals [2]. Another important application is to visualize graphs created from sparse matrices which represent spatial relationships [12]. In this framework graph drawing is successfully applied in order to expose several characteristics of the underlying matrices (e.g., correspondence with a 2D or 3D geometry), since nonzero elements of a sparse matrix correspond to edges of a graph. For graphs coming from the application area mentioned above, there may be misplaced edges due to lacking data or human misinterpretation so that the graph fails to be drawn as expected. In fact high-quality and large scaled data are often not available due to legal, economic, technological or other obstacle [8]. Then one is interested in computing a modified graph with an improved readability. More precisely if a graph is used to model experimental data then edge modifications correspond to correcting errors in the data: adding an edge corrects a false negative error and deleting an edge corrects a false positive error. Therefore in the context of graph drawing, edge modifications may assist to reveal and correct "false" edges.

Graph modification problems related to graph drawing have been studied from the complexity point of view. Close related to graph drawing is the detection of an axial or rotational symmetry. If three graph modification operations are allowed (vertex deletion, edge removal, and edge contraction) then computing the minimum number of modifications that result in a symmetric graph is NP-complete [9]. Based on the latter result, Chuang and Yen [10] considered symmetry as the underlying aesthetic criterion and proposed an algorithm that given the modification operations, a better drawing is output than the conventional spring algorithms. Notice that the three mentioned operations always result in a subgraph of the original graph. Allowing only edge modifications (edge removals and edge additions) has been considered through an integer linear programming technique that minimizes the number of edge modifications [6]. It is important to notice that the running time of the later algorithm is exponential in the size of the input graph.

More generally, if the objective of the graph property for the modification problem is restricted to most of the interesting graph classes (e.g., bipartite, chordal, interval, perfect graphs), then computing the modified graph is NP-hard [16, 24]. Close related to graph drawing is the class of planar graphs that admit layouts with no edge crossings. With respect to edge modifications it is meaningful to minimize the number of edge removals of an arbitrary graph that results in a planar graph (minimum planar deletion); such a computation is again NP-complete [27]. However planar graphs are closed under edge removals. This gives rise to a linear-time algorithm for computing a *minimal* set of edges whose removal results in a planar graph (minimal means that no subset of the removal edge is enough to give another planar graph): if no single edge can be removed then we reach a maximal planar subgraph of the original graph [20]. Another related problem is determining the crossing number of a single-edge modified planar graph [7, 17]. Such graphs are called near-planar graphs and are exactly the graphs containing an edge whose removal zeros the crossing number. In terms of edge modifications we want to add an edge in a planar graph such that the output drawing is the best possible; the latter problem even when adding a vertex can be solved in polynomial time [17]. Apart from the modification towards a specific graph class, the modification task that we consider here seeks for the best layout of a graph.

More precisely, in this work we consider an edge modification scheme based on graph layouts that encode spatial relationships. More precisely we apply at most k edge modifications on a given graph and we seek for a modified graph with the same number of edges that minimizes a specific quantity depending on the layout. In fact we assume that the input graph contains misplaced edges which we attempt to correct guided by its layout. Thus we use the drawing of a graph as a measure of the graph's correctness. Notice, however, that asymmetric graphs (with irregular structures) admit tangled drawings compared to symmetric ones and thus it is unreasonable to modify graphs towards irregular structures. We give a polynomial-time algorithm that computes a modified graph with a better readability than the original graph when only single-edge replacements are allowed. The layout of a graph is achieved through a force-directed algorithm by taking into account only its structural properties (no rendering parameters are considered). Similar to the layout method used in [12], we apply such a drawing technique since it exposes symmetry and uniform vertex distribution [3, 23].

In order to evaluate the drawing of a graph, several metrics have been proposed [4, 13, 19, 25, 26]. Here we invoke a subset of the known aesthetics and devise a new aesthetic metric that naturally fit in well-structured graphs. Among the modified graphs we keep the same number of edges for two reasons. The evaluation of a layout is established through aesthetic metrics (e.g., crossing number, angular resolution) that are proportional to the number of edges and thus smaller number of edges results in a better aesthetic measure. Further from the practical point of view this is a quite realistic constraint since the number of the edges (and the number of vertices) can easily be detected whenever the original graph has incorrect such number. Following the same lines we restrict to several graph invariants of the original graph (e.g., diameter, minimum and maximum degree) so that the outcome maintains the overall structure of the input. Surprisingly all the above technical details need to be settled for edge modifications within a graph drawing framework, even when applying an algorithm that allows only single-edge replacements. We study the behavior of the proposed algorithm by presenting several drawing examples taken from the sparse matrix collection given in [12]. The experimental results show that the proposed approach with the adopted features achieves the initial graph after a small number of modifications.

The paper is organized as follows. First we give graph-theoretic notations and definitions. Then we define the structural properties of the input graph that are maintained throughout each modified graph. We conclude Section 2 with our proposed metrics needed to evaluate a force-directed drawing and we formally state the considered problem. In Section 3 we give in details our algorithm and present its results on a data set that represents structural, computational fluid dynamics, and thermal problems. We conclude the paper in Section 4 by presenting future directions and potential extensions.

2. PRELIMINARIES

All graphs in this text are undirected and simple. A graph is denoted by $G = (V, E)$ with vertex set V and edge set E. We use the convention that $n = |V|$ and $m = |E|$. For a vertex subset $S \subseteq V$, the *subgraph of G induced by S* is $G[S] = (S, \{\{u, v\} \in E \mid u, v \in S\})$. The *neighborhood* of a vertex x of G is $N_G(x) = \{v \mid \{x, v\} \in E\}$. The *degree* of x is $\deg_G(x) = |N_G(x)|$. The minimum and maximum degree of G are denoted by $\delta(G)$ and $\Delta(G)$, respectively. If $S \subseteq V$, then $N_G(S) = \bigcup_{x \in S} N_G(x) \setminus S$. We will omit the subscripts when there is no misunderstanding. For two sets S_1 and S_2 we write $S_1 \triangle S_2$ to denote their symmetric difference. For two vertices u, v of G we call $\{u, v\}$ a *non-edge* of G if $\{u, v\} \notin E$. We denote by F the set of non-edges of G. For an edge $e \in E$ we write $G-e$ to denote the graph $(V, E \setminus \{e\})$ and for a non-edge $f \in F$ we write $G+f$ to denote the graph $(V, E \cup \{f\})$.

A graph is *connected* if there is a path between any pair of vertices. A *connected component* of G is a maximal connected subgraph of G. The number of connected components of G is denoted by $cc(G)$. For a pair of vertices (u, v), we write $d(u, v)$ to denote the number of vertices in a shortest path between u and v. The *diameter* $d(G)$ of G is defined to be $\max_{u,v \in V} d(u, v)$. The length of the shortest cycle in G is called *girth*, denoted by $g(G)$. If a graph has no cycles then $g(G)$ is undefined.

2.1 Graph invariants of the initial graph

Towards the selection of the modified graph H, one of our main ingredients is that the overall structure of the original graph G should not change in H. This comes from the fact that the original graph is easily noticed if certain structural properties are fulfilled (see also [1]). For instance it does not make sense to modify the number of connected components. Further we use as *boundary vertices* of G the ones with the minimum and maximum degree. Those boundary vertices in the modified graph H should not exceed the corresponding bounds obtained from G so that regular modified graphs are suitable candidates. It is also known that the diameter and the girth of a graph are basic combinatorial characteristics and have tight connections to many other graph properties [5]. Therefore the corresponding values of both graphs G and

H should be the same. Observe that the above properties do not depend on a drawing of G but rely only on the structure imposed by G. In order to capture the structure of G, we summarize in a formal way these properties in the following definition.

Definition 1. Let G and H be two graphs both on n vertices and m edges. We write that $H \in \Pi(G)$ if the following conditions hold:

(i) $cc(H) = cc(G)$, (ii) $\delta(H) \geq \delta(G)$ and $\Delta(H) \leq \Delta(G)$,

(iii) $d(H) = d(G)$, and (iv) $g(H) = g(G)$.

Let us note that given a graph G the most time consuming steps in order to compute all these invariants are the ones involved the diameter and the girth. Both of these variants can be computed in $\mathcal{O}(nm)$ time [11, 22] and hence testing whether a modified graph H satisfy the property $\Pi(G)$ can be done in $\mathcal{O}(nm)$ time. Also note that such a restriction on the class of graphs $\Pi(G)$ improves the computational cost for searching a modified graph H, by reducing the searching area of graphs with similar invariants. Certainly there are several other graph invariants (or even *width parameters*) that reveal structural properties of a graph [14], but here we focus on invariants that allow efficient computation.

2.2 Evaluating a graph layout

Graph layout algorithms usually conform explicitly or implicitly to one or more aesthetic criteria (i.e., number of edges crossings, symmetries, drawing area, etc.). In this work we focus on classical force-directed algorithms that attempt to minimize a certain energy function [3]. Although an important feature of force-directed algorithms is that they produce drawings with a high degree of symmetry [15], their energy function is not directly mapped to specific aesthetic measure. For this reason we apply formal metrics for evaluating the aesthetic presence of a straight-line drawing. Here we define measures that are continuous and scaled in range $[0, 1]$, where 0 indicates the best score and 1 is the worse. Following this assumption, the lowest the score of a measure the highest the aesthetic quality. Scaling these quantities ensures that their values are independent of the nature (size, structure) of the underlying graph. They are also intended to be applicable in the analysis of drawings of any graph of any structure or size, and allow quantitative comparisons between drawings of different graphs.

Several metrics have been proposed in order to evaluate a drawing of a given graph [4, 13, 19, 25, 26]. Such metrics have been successfully applied in order to estimate different layouts of the same graph. Here we focus on evaluating different drawings of edge modified graphs. Thus we invoke a subset of the aesthetics that naturally fits in well-structured graphs coming from specific applications. Our goal is to use independent and competitive aesthetic metrics that will contribute equally in the overall evaluation. Furthermore, scaling of the proposed metrics is performed with a different strategy than the ordinary methods. We consider the input graph to be the worse in terms of aesthetic measures and we scale all metrics of a modified graph against it. Thus our proposed metrics will be presented in terms of the original drawing of the input graph.

Let $G_0 = (V_0, E_0)$ be the input graph and let $G = (V, E)$ be a modified graph such that $V_0 = V$ with $n = |V|$ and $|E_0| = |E| = m$. For the two graphs G_0 and G we invoke an algorithm that computes straight-line drawings in the plane denoted by $\Gamma(G_0)$ and $\Gamma(G)$, respectively[1]. Let $q(G)$ be a countable quantity (or a metric) on a drawing $\Gamma(G)$ of graph G. Then the *scaled quantity* $\tilde{q}$ *of* $\Gamma(G)$ *with respect to* $\Gamma(G_0)$ is defined as

$$\tilde{q}(G, G_0) = \begin{cases} \dfrac{q(G)}{q(G_0)}, & \text{if } q(G) < q(G_0); \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

Thus all scaled metrics are real-valued and constrained to $[0, 1]$. We use four different metrics to quantify the quality of $\Gamma(G)$ with respect to the drawing of an initial graph $\Gamma(G_0)$. The first three metrics have already been proposed, whereas the fourth metric to the best of our knowledge has not been used.

edge crossings: $q_{cross}(G)$ is the number of edge crossings in $\Gamma(G)$ and it is fairly common that such a value must be small in an easy-to-read drawing. It is fairly common that an easy-to-read drawing of G must have small number of edge crossings. Observe that this metric takes integer values and $q_{cross}(G) \geq 0$.

edge lengths: $q_{edge}(G) = q_{ext} + q_{std}$ where q_{ext} is the difference between the maximum and minimum edge lengths, and q_{std} is the standard deviation of the edge lengths. In the ideal case the edge lengths are equal to mean edge length and that corresponds to the case where all edges have the same length. The rationale of this measure is to penalize drawings with long and short edges (extreme bounds) with respect to the preferred edge length that is represented by the mean value. This is a real valued metric and $q_{edge}(G) \geq 0$.

angular resolution: $q_{angle}(G) = \min\{|\theta_i - b_i|/\theta_i\}$ where θ_i is the ideal angle and b_i the minimum angle between the $\deg(v_i)$ edges incident to v_i. In nice drawings every vertex must have its minimum angle close to its ideal one [25].More precisely the ideal angle is defined as $\theta_i = \frac{2\pi}{deg(v_i)}$ for vertices of $\deg(v_i) \geq 2$. It it known that in a nice drawing every vertex must have its minimum angle close to its ideal one, meaning that $q_{angle}(G) \approx 0$. It is clear that $q_{angle}(G)$ is already constrained in $0 \leq q_{angle} \leq 1$. In Figure 1 we illustrate the minimum angle metric for vertex u_1.

non-adjacent vertex proximity: $q_{prox}(G) = \sum_{i=1}^{n} \kappa_i$, where κ_i is the number of non-adjacent vertices that are drawn inside the sphere centered at v_i with radius equal to the longest edge incident to v_i. It is not difficult to see that non-adjacent vertices must not be too close in an easy-to-read drawing, since for otherwise adjacencies between vertices are confused. Therefore for every vertex, κ_i should be as small as possible and $q_{prox}(G)$ is an integer valued metric where $q_{prox}(G) \geq 0$. In Figure 2 we provide an example of this metric.

overall evaluation: Scaling on the above metrics is achieved by Eq 1. Thus the overall aesthetic metric of a drawing of a graph $\Gamma(G)$ with respect to $\Gamma(G_0)$ is defined as

$$\begin{aligned} W(G, G_0) &= \tilde{q}_{cross}(G, G_0) + \tilde{q}_{edge}(G, G_0) \\ &+ \tilde{q}_{angle}(G, G_0) + \tilde{q}_{prox}(G, G_0). \end{aligned} \quad (2)$$

[1]Although it is a bit ambiguous to use the same drawing Γ on both graphs G_0 and G, hereafter we write $\Gamma(G_0)$ and $\Gamma(G)$ to denote two different drawings of different graphs.

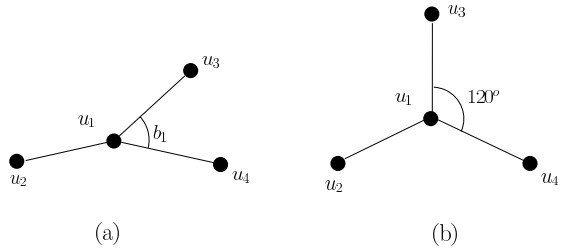


Figure 1: An example of minimum angle. In the left figure we denote by $b_1 = 30^o$ the minimum angle whereas in the right figure the ideal angle of 120^o is shown for a vertex of degree 3. For this case $q_{angle}(G) = 0.75$.

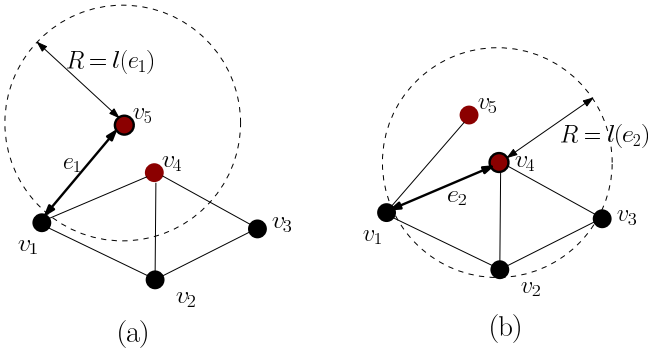
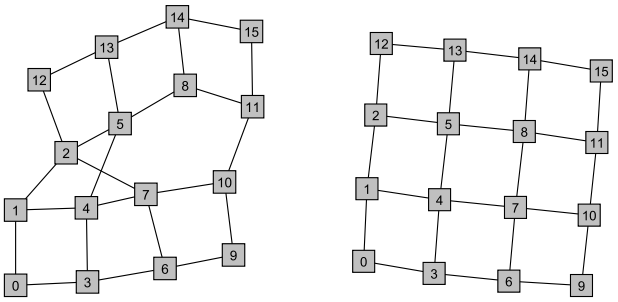


Figure 2: Example of non-adjacent vertex proximity violations. Vertex v_4 is inside the sphere centered at v_5 with radius $l(e_1)$. Analogously v_5 violates the proximity constraint regarding v_4.



(a) Initial graph G_0 (b) Modified graph G

Metrics	$\Gamma(G_0)$	$\Gamma(G)$	Scaled metrics	$\Gamma(G)$
q_{cross}	1	0	$\tilde{q}_{cross}$	0.00
$q_{edge}\begin{cases} q_{ext} \\ q_{std} \end{cases}$	10.02 2.37	3.98 0.78	$\tilde{q}_{edge}\begin{cases} \tilde{q}_{ext} \\ \tilde{q}_{std} \end{cases}$	0.39 0.33
q_{angle}	0.77	0.5	$\tilde{q}_{angle}$	0.64
q_{prox}	4	0	$\tilde{q}_{prox}$	0.00
$w(\cdot)$	18.16	5.26	$W(G, G_0)$	**1.36**

Figure 3: An example of an edge modification between two different graphs and a table with the corresponding metrics. Note that $W(G_0, G_0) = 5$ which serves also as an upper bound for $W(G, G_0)$.

Whenever we write $w(G)$ we refer to the corresponding quantities $q(G)$ which are not scaled with respect to a drawing of a different graph. Observe that in our implementation the metrics are weighted equally and this reflects equivalence among the corresponding aesthetic qualities. We are not concerned with the distribution of the metric values since we scale with respect to the initial graph (relative scaling) without using absolute upper bounds. Furthermore it is not difficult to see that computing $W(G, G_0)$ requires $\mathcal{O}(nm)$ time according to the corresponding definitions. In Figure 3 we illustrate an example of the proposed scaled metric.

Notice that $W(G, G_0)$ in Equation 2 contains four metrics that can be balanced accordingly by simple adding weights that are associated with each metric. Then the overall metric can be used to achieve other known modification tasks. For example altering the edges of a non-planar graph in order to minimize the edge crossings (i.e., planar editing) can be accomplished by setting the weight for the edge crossings $q_{cross}(G)$ to one and excluding the other three metrics by setting the corresponding weights to zero. Thus $W(G, G_0)$ can be seen as a more general evaluation of the drawing that captures (equally weighted) important aesthetic metrics.

2.3 Edge modification problem driven by drawing aesthetics

In general an *edge modification problem* with respect to a given property $\mathcal{P}$ can be stated as follows. It takes as input a graph $G = (V, E)$ and a non-negative integer k, and the question is whether there exists a graph H such that $H \in \mathcal{P}$ and H is obtained from G by altogether at most k edge additions and edge deletions. More formally the task is to find a subset $S \subseteq V \times V$ with $|S| \leq k$ such that the graph $H = (V, E \triangle S)$ satisfies $\mathcal{P}$. If we restrict ourselves so that $|E \triangle S| = |E|$ then we are only allowed to *move* the original edges of G. By *moving* (also known as *rewiring*) an edge e we mean a modification to the endpoints of e. In such a case both graphs G and H have the same number of vertices and edges. This is exactly the problem we are dealing with, related to *nice* straight-line drawings of graphs that improve readability. Hereafter nice drawings refer to layouts with small measure evaluated by the proposed aesthetic metrics given in $W(H, G)$. Notice that a movement of an edge corresponds to two edge modifications, one for the addition of a non-edge and one for the deletion of an edge. In the following when we refer to an edge modification we imply a movement of an edge that typically results from two edge modifications. We formalize the edge modification problem in our settings:

Edge Drawing Modifications

Input: An undirected graph $G = (V, E)$ and a non-negative integer k.

Question: Can we apply to G at most k edge replacements in order to attain a graph H that achieves the minimum $W(H, G)$?

3. AN ALGORITHM FOR SINGLE-EDGE MODIFICATIONS

Here we consider the modification problem under the constraint that only single-edge replacement is allowed at each modification step. In this situation we will not achieve the minimum number of edge modifications but we will study the outcome of an algorithm that considers at each modification step the graph having the minimum aesthetic evaluation. Clearly there can be fewer number of single-edge

modifications that lead to the final graph.

We assume that a set $S \subseteq V$ is given together with the initial graph $G = (V, E)$ so that the allowed edge modifications take place only in $G[S]$. The set S reflects an estimation of the modified area taken from an initial layout of G and note that in the worst (unbiased) case $S = V$. Let $G = (V, E)$ be a graph, let F be the set of its non-edges and let $S \subseteq V$. We denote by $\mathcal{H}(G)$ the class of graphs that are obtained from G by removing an edge and adding a non-edge both incident to vertices of S. That is, for every $H \in \mathcal{H}(G)$, there are $e \in E$ and $f \in F$ with their endpoints in S for which $H = G - e + f$.

Let us now describe the corresponding problem in terms of single-edge modifications. We are given a graph $G = (V, E)$, an integer $k > 0$, and a set $S \subseteq V$. The task is to compute a graph $G' = (V, E')$ such that there is a sequence of graphs $G = G_0, \ldots, G_{k'} = G'$ with $k' \leq k$ and for every $1 \leq i \leq k'$, $G_i \in \mathcal{H}(G_{i-1})$, $G_i \in \Pi(G_{i-1})$, $w(G_i) < w(G_{i-1})$ and G_i admits the minimum $W(G_i, G_{i-1})$. The graph G' is called *single-edge modified graph within distance k from G*. At each edge modification step we seek for a graph G_i that uniformly reduces the objective metrics.

Our algorithm tries all possible single replacements in $G[S]$ and chooses the one having the minimum objective metric. If at each step there are no more allowed edge modifications or there is no graph with smaller overall evaluation with respect to the previous graph, then we output the graph computed at that step and the algorithm stops. In order to compare the overall evaluations of two consecutive graphs we require their difference to be greater than a small constant. As explained previously before computing and evaluating a modified drawing, the modified graph must satisfy the graph properties of the original graph. The formal description of the proposed algorithm, called Min_Search(G, k), is given below.

Algorithm Min_Search(G, k)
Input: a graph $G = (V, E)$, an integer $k > 0$, and a set $S \subseteq V$
Output: a single-edge modified graph $G' = (V, E')$ within distance k from G

Let $\min_G = W(G, G)$
for each edge $e \in E$ and non-edge $f \in F$ with their endpoints in S **do**
  Let $G' = G - e + f$
  if $G' \in \Pi(G)$ **then**
    Compute a drawing $\Gamma(G')$ and evaluate $w(G')$, $W(G', G)$
    if $\min_G > W(G', G)$ **then**
      $\min_G = W(G', G)$ and $\hat{G} = G'$
end-for
if $W(G, G) - \min_G < \epsilon$ (no minimum found) **or** $k = 1$ **then**
  output G and **halt**
call Min_Search$(\hat{G}, k - 1)$

In our implementation the computed drawing $\Gamma(G)$ is achieved by a force directed drawing algorithm that requires $\mathcal{O}(n \log n + m)$ time [18]. As already mentioned, testing whether $G' \in \Pi(G)$ and computing the evaluation $W(G', G)$ can be done in $\mathcal{O}(nm)$ time. There are $\mathcal{O}(|S|^2)$ pairs of vertices in $G[S]$ that implies a total of $\mathcal{O}(|S|^4)$ iterations of the main for-loop. Therefore the total running time of the algorithm at each call is polynomial in the size of G, i.e., the algorithm runs in $\mathcal{O}(|S|^4 nm)$ time for a single-edge modification.

It is not difficult to see that the algorithm terminates in a finite number of iterations (recursive calls) and avoids reprocessing already processed graphs, since it gradually moves towards the minimum objective metric. During the execution of the algorithm there are at most k modified graphs that the algorithm promotes corresponding to its k recursive calls. At each call there are at most $|S|^4$ produced graphs for consideration. In fact, as illustrated in the next section, only few of these candidate graphs are promoted for computing and evaluating their drawing. Among the produced graphs that satisfy the graph properties $\Pi(G)$, the algorithm outputs the graph G' with the minimum $w(G')$. However it is not guaranteed that the resulting graph G' requires exactly k edge modifications since there might be fewer edge modifications resulting to G'. We illustrate that the algorithm Min_Search(G, k) performs well in practice in order to untangle the final drawing with small number of modifications.

3.1 Implementation and results

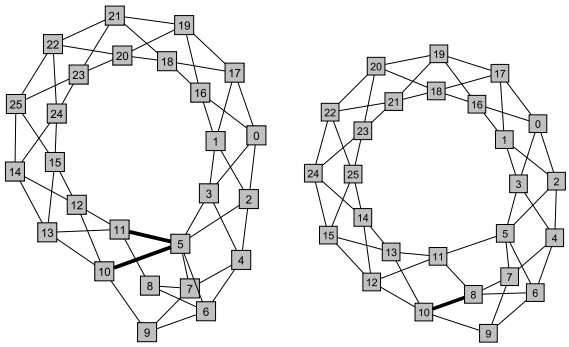
We have implemented algorithm Min_Search(G, k) using the Open Graph Drawing Framework (OGDF) C++ library which provides sophisticated algorithms and data structures [21]. For drawing every intermediate graph we apply the fast multipole multilevel method (FMMM) introduced in [18]. Notice that any force-directed drawing algorithm can be adopted for producing the intermediate layouts; also in [12] a similar drawing algorithm is applied for the same group of tests.

The test graphs presented here are taken from the sparse matrix collection given in [12] representing structural, computational fluid dynamics, and thermal problems. In Table 1 we present a short description and the corresponding application area for each case.

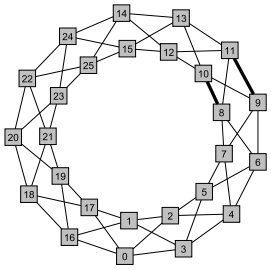
In order to illustrate the edge modifications steps of the algorithm we give an example shown in Figure 4 that is not contained in the sparse matrix collection [12]. The particular figure contains the overall computed sequence consisting of three graphs. For every other example we present two graphs, the initial graph G_0 with the selection of the vertices contained in S and the modified graph G_k produced by the algorithm with input k. We use dark grey color for the vertices of $S \subset V$ except for the cases presented in Figures 4 and 5 in which we set $S = V$. In each example taken from the sparse matrix collection [12], the initial graph is artificially obtained by rewiring a small number of edges.

The rewiring is performed by a uniform random selection of four endpoints that correspond to a removal and insertion of an edge. During an edge modification we require the resulting graph to preserve the initial graph invariants. A small number of random modifications were tested for each case and all of them resulted in the corresponding initial graph. Furthermore the selection for S performed on G_0 is quite realistic; i.e., S corresponds to a non-symmetric and irregular area that is easily detected in the layout.

Regarding the running time the algorithm is heavily depended by the size of the input graph and the size of selected area as already described. However in practice we achieve acceptable running times since we allow only a subset of the produced graphs to be drawn and evaluated through the graph invariant criterion. We have not observed any divergence from the expected theoretical computational cost and in each example the measured running time matches the corresponding bound. In Table 2 we report the percentage of the promoted graphs that match the properties of the input graph. Notice that the girth of the input graph



(a) Initial graph G_0 (b) Graph G_1

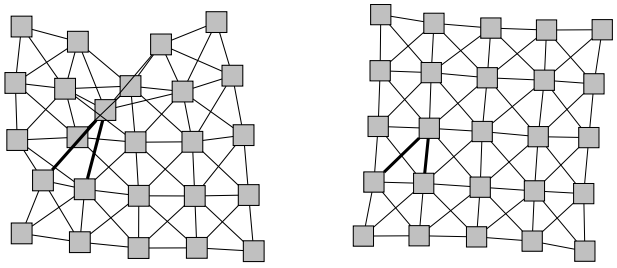


(c) Modified graph G_2

Figure 4: An example of 2 edge modifications where $S = V$ for any $k \geq 2$. Observe that if we replace the edge $\{10, 5\}$ by $\{10, 8\}$ in G_0 then we decrease the metric $q_{cross}(G_0)$ and the value that contributes to the minimum angle $q_{angle}(G_0)$ (by increasing the degree of vertex 8). In a similar way the algorithm Min_Search(G, k) replaces the edge $\{11, 5\}$ of G_1 by $\{11, 9\}$ ending up with the graph G_2.

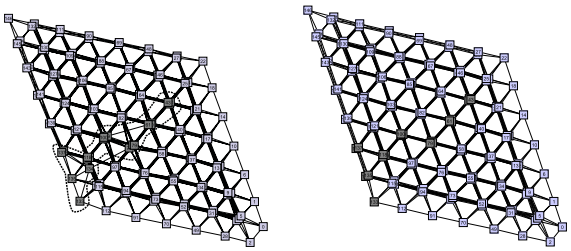
plays a crucial role for the number of promoted graphs as depicted in the examples `rdb200`, `grid1_dual`, and `pde900` since only a few of the tested graphs have shortest cycle of length four. On the other hand, the minimum and maximum degree maintain a small amount of considered graphs as shown in the examples `lund_a` and `lshp_265`. In general the combination of all proposed invariants ensures that more than half of the produced graphs are discarded, leading to a significant speed-up of the whole process.

There are cases for which the algorithm reveals a desired graph for any $k \geq c$ where c is a specific constant, meaning that the algorithm halts after *exactly* c edge modifications. Cases for which c equals the minimum number of misplaced edges are given in Figures 6, 7, 9, 10, and 12. On the other hand, examples of the algorithm performing more modifications than expected are depicted in Figures 5 and 11. However there are also examples that the algorithm reaches a desired layout, though it continues to produce further modified graphs. This comes from the fact that the desired graph might not have the smaller evaluation among its single-edge modified graphs. Thus in such cases it is expected that the algorithm does not always produce the desired graph within the computed sequence of graphs. We write that the algorithm halts for $k = c$ if for $k > c$ the algorithm performs strictly more than c steps. An example for $k = c$ is presented in Figure 8. In Table 2 we summarize our results on the test graphs.



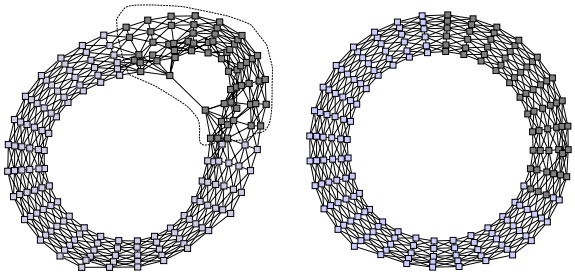
(a) Initial graph G_0 (b) Modified graph G_6

Figure 5: (`lap_25`) An example of 6 edge modifications where $S = V$ for any $k \geq 6$. Notice that G_6 can be obtained from G_0 by modifying only 2 edges.



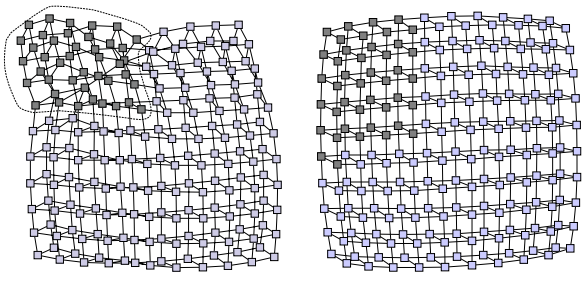
(a) Selecting S on G_0 (b) Modified graph G_3

Figure 6: (`lund_a`) An example of 3 edge modifications for any $k \geq 3$.



(a) Selecting S on G_0 (b) Modified graph G_2

Figure 7: (`can_187`) An example of 2 edge modifications for any $k \geq 2$.

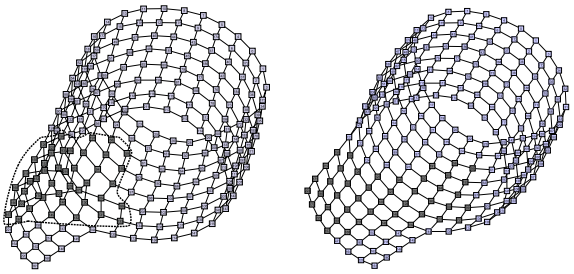


(a) Selecting S on G_0 (b) Modified graph G_1

Figure 8: (`rdb200`) An example of a single edge modification for $k = 1$.

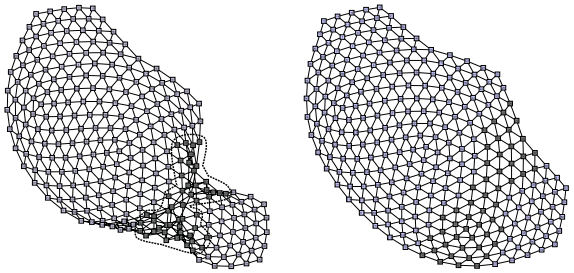
4. CONCLUDING REMARKS

We have presented a polynomial-time algorithm for single-



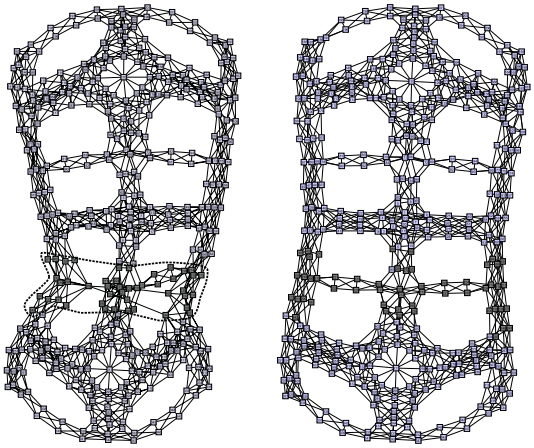
(a) Selecting S on G_0 (b) Modified graph G_2

Figure 9: (`grid1_dual`) An example of 2 edge modifications for any $k \geq 2$.



(a) Selecting S on G_0 (b) Modified graph G_3

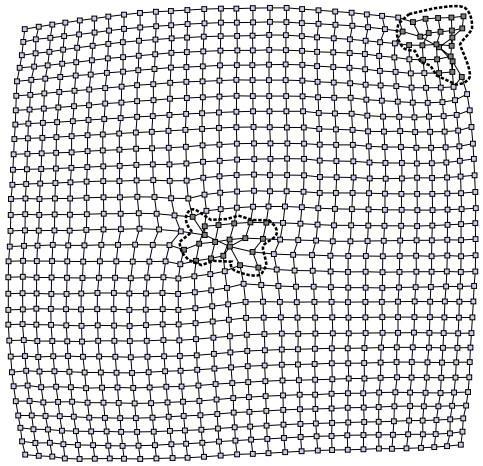
Figure 10: (`lshp_265`) An example of 3 edge modifications for any $k \geq 3$.



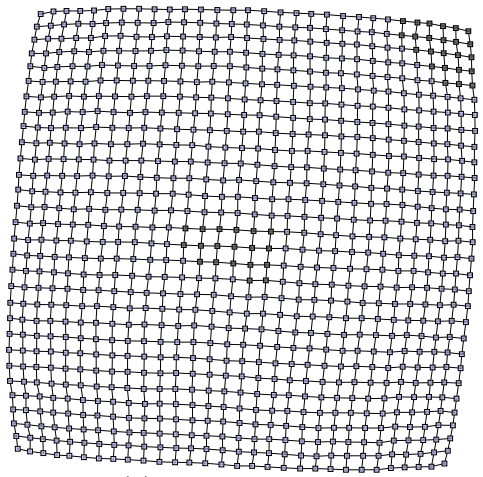
(a) Selecting S on G_0 (b) Modified graph G_5

Figure 11: (`dwt_419`) An example of 5 edge modifications for any $k \geq 5$. Notice that G_5 can be obtained from G_0 by modifying only 2 edges.

edge modifications based on the layouts of graphs that encode spatial relationships. We have considered the drawing of a graph as a measure of the graph's correctness and proposed a composite metric based on aesthetic criteria. Further investigation would include extension to other aesthetic metrics (such as the drawing area or orthogonality [25]). As shown in the presented figures it is natural to seek for an automatic calculation of the precomputed set S which reflects an estimation of the area to be modified. Some preliminary results show that the proposed metrics can be used as



(a) Selecting S on G_0



(b) Modified graph G_2

Figure 12: (`pde900`) An example of 2 edge modifications for any $k \geq 2$.

Name	Description	Application
`lap_25`	Laplacian on a 5 by 5 grid	finite element structural problem
`lund_a`	Lund eigenvalue problem	generalized eigenvalue problem
`can_187`	symmetric pattern	finite element structures in aircraft design problem
`rdb200`	reaction-diffusion model	fluid dynamics problem
`grid1_dual`	2D/3D dual grid	finite element problem
`lshp_265`	symmetric matrix	L-shape thermal problem
`dwt_419`	symmetric matrix	structural ship design problem
`pde900`	2D/3D grid	partial differential equation problem

Table 1: Description and application areas of the test graphs taken from [12].

a vertex scoring mechanism. More specifically vertices can be ranked according to their contribution on the constituent

Name	n	m	$\lvert S\rvert$	$\% \in \Pi(G)$	$k_{\min}$	output
lap_25	25	72	25	39.8%	2	$\forall k \geq 6$
lund_a	147	1151	20	7.8%	3	$\forall k \geq 3$
can_187	187	651	63	37.2%	3	$\forall k \geq 2$
rdb200	200	460	42	8.05%	1	for $k = 1$
grid1_dual	224	420	45	0.7%	2	$\forall k \geq 2$
lshp_265	265	744	51	1.5%	3	$\forall k \geq 3$
dwt_419	419	1572	53	44.1%	2	$\forall k \geq 5$
pde900	900	1740	44	1.02%	2	$\forall k \geq 2$

Table 2: A summary of the experimental results on the test graphs. The fifth column ($\% \in \Pi(G)$) corresponds to the percentage of promoted graphs that satisfy the invariant properties. The last two columns correspond to the minimum number of edge modifications and the number of edge modifications performed by the algorithm Min_Search(G, k), respectively.

metrics. Initial experimentation reveal that such a ranking scheme is quite efficient in locating edges that are candidate for removal, though potential non-edges that are candidate for addition need a more careful analysis. Finally the overall computational time could be reduced by introducing a force-directed drawing algorithm that uses the proposed metrics as edge-weights; the weights operate relaxing factors in order to unfold "ruined" areas.

5. REFERENCES

[1] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Trans. Vis. Comput. Graph.*, 17(4):539–552, April 2011.

[2] E. N. Argyriou and A. Symvonis. Detecting periodicity in serial data through visualization. In *Advances in Visual Computing - 8th International Symposium, ISVC*, pages 295–304, 2012.

[3] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Algorithms for the Visualization of Graphs.* Prentice-Hall, New Jersey, 1999.

[4] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch. The aesthetics of graph visualizations. In *Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 57–64, 2007.

[5] D. Bokal, G. Fijavz, and B. Mohar. The minor crossing number. *SIAM J. Discrete Math.*, 20(2):344–356, April 2006.

[6] C. Buchheim and M. Jünger. An integer programming approach to fuzzy symmetry detection. In *Graph Drawing*, pages 166–177, 2003.

[7] S. Cabello and B. Mohar. Crossing number and weighted crossing number of near-planar graphs. *Algorithmica*, 60(3):484–504, July 2011.

[8] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), March 2006.

[9] H.-L. Chen, H.-I. Lu, and H.-C. Yen. On maximum symmetric subgraphs. In *Graph Drawing*, pages 373–383, 2000.

[10] H.-L. Chen, H.-I. Lu, and H.-C. Yen. On nearly symmetric drawings of graphs. In *Information Visualisation*, pages 489–494, 2002.

[11] F. R. K. Chung. Diameters of graphs: old problems and new results. *Congressus Numerantium*, 60:295–317, 1987.

[12] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. 38(1):1–25, November 2011.

[13] H. R. Dehkordi, Q. H. Nguyen, P. Eades, and S.-H. Hong. Circular graph drawings with large crossing angles. In *WALCOM: Algorithms and Computation, 7th International Workshop*, pages 298–309, 2013.

[14] R. Diestel. *Graph Theory.* Springer Verlag, Heidelberg, 2005.

[15] P. Eades and X. Lin. Spring algorithms and symmetry. *Theor. Comput. Sci.*, 240(2):379–405, June 2000.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman and Co., New York, 1978.

[17] C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, April 2005.

[18] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing*, pages 285–295, 2004.

[19] D. Harel. Mathematics of program construction. In *Graph Drawing*, pages 1–5, 1998.

[20] W.-L. Hsu. A linear time algorithm for finding a maximal planar subgraph based on pc-trees. In *COCOON: Computing and Combinatorics, 11th Annual International Conference*, pages 787–797, 2005.

[21] `http://www.ogdf.net`. *The Open Graph Drawing Framework.*

[22] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, September 1978.

[23] M. Kaufmann and D. Wagner. Drawing graphs, methods and models. In *Lecture Notes in Comput. Sci.*, volume 2025. Springer, 2001.

[24] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, September 2001.

[25] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages and Computing*, 13(5):501–516, October 2002.

[26] M. Taylor and P. Rodgers. Applying graphical design techniques to graph visualisation. In *Information Visualisation*, pages 651–656, 2005.

[27] M. Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, April 1981.