Crossing Minimization meets Simultaneous Drawing

Markus Chimani*
Technical University of Dortmund, Germany

Michael Jünger†
University of Cologne, Germany

Michael Schulz‡
University of Cologne, Germany

Abstract

We define the concept of crossing numbers for simultaneous graphs by extending the crossing number problem of traditional graphs. We discuss differences to the traditional crossing number problem, and give an NP-completeness proof and lower and upper bounds for the new problem. Furthermore, we show how existing heuristic and exact algorithms for the traditional problem can be adapted to the new task of simultaneous crossing minimization, and report on a brief experimental study of their implementations.

Keywords: Simultaneous Graph Drawing, Crossing Number, NP-completeness, Algorithms, Branch-and-Cut

Index Terms: G.2.2 [Discrete Mathematics]: Graph Theory—Graph algorithms; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—Routing and layout

1 Introduction

Graph structures are an intrinsic part of many real world problems. Using graph drawing to layout the occurring graphs is a good way of visualizing the problem and can help the user to understand complex relationships. Traditional graph drawing deals with a single graph as its input. However, in many graph drawing applications the input does not consist of a single network but a series of graphs. We will give some applications where this situation occurs in a natural way.

Analysis of biological networks. The analysis of metabolic and signal transduction networks is a pivotal element in biochemistry, and a lot of research is conducted on the analysis and identification of central metabolite cycles and the interactions taking place within a cell. The need for good visualizations of such networks is, e.g, described in [1]. The role of simultaneous graph drawing in this field is strengthened by a survey by Suderman and Hallet [31], where they state the unfortunate weak tool support for visualizing similar networks and networks from a time series.

Evolutionary changes and interaction and function prediction. Biochemical experiments show that the same metabolism is usually not identical but very similar, over a set of different organisms, such as human, mouse, yeast and bacteria. Recognizing, understanding exactly how such metabolites differ, and identifying the common and evolutionarily conserved patterns is a crucial step to allow the identification and prediction of biological interactions and functions, see, e.g., [2, 30, 32].

Figure 1 shows a conserved coregulated protein cluster between yeast and fly, as visualized in [32]; the gray dotted lines are necessary to establish the node identification between the two organisms. Figure 2, on the other hand, shows the vitamin B6 metabolism in four different organisms, drawn with a simultaneous layout; the individual graphs are drawn side by side instead of on top of each other. The visualization presents the similarity of the graphs in a clear way as the common subgraphs in the multiple networks are drawn identically.

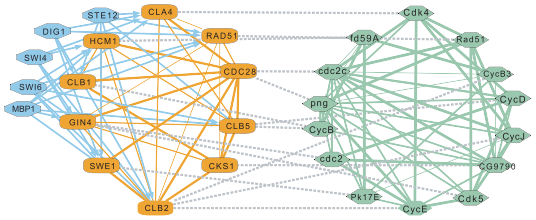


Figure 1: Conserved coregulated protein cluster between yeast and fly, as visualized in [32].

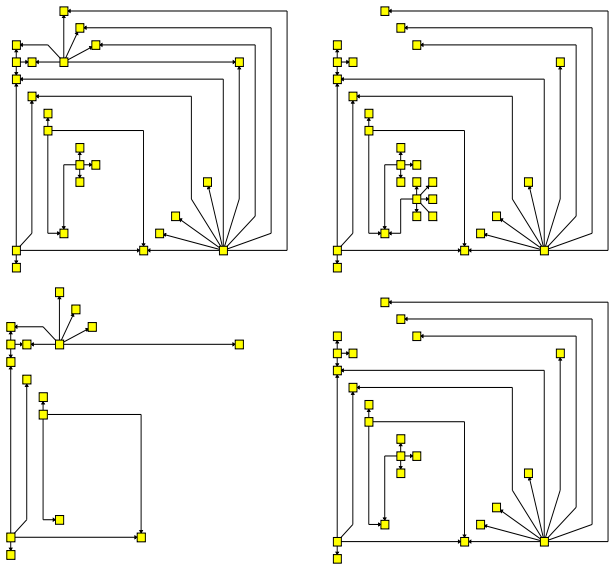


Figure 2: The vitamin B6 metabolism in different species: yeast (Saccharomyces cerevisiae - upper left), silibacter bacterium (upper right), staphylococcus bacterium (lower left) and streptomyces bacterium (lower right). The drawings can be produced using one of the algorithms described in this paper.

Network perturbation and drug design. Understanding the functionality and interrelationship of metabolites within a cell is a crucial task in drug design. A disease modifies metabolic paths. In order to understand a disease it is important to understand the modifications to the affected metabolic networks. New medicaments, on the other hand, are designed to manipulate one metabolic path in a cell, e.g., by removing edges, i.e., preventing biochemical reactions to happen. However, this manipulation can change other pathways as well which can lead to severe side-effects of this drug.

*e-mail: markus.chimani@cs.uni-dortmund.de
†e-mail: mjuenger@informatik.uni-koeln.de
‡e-mail: schulz@informatik.uni-koeln.de
supported by the German Science Foundation (JU 204/11-1)

Biochemists want to display the cellular response arising from perturbing the network [19]. By drawing the networks with and without the disease or drug in a simultaneous way, these changes and interactions are easier to determine.

Temporal changes. While the reactions in a cell take place, the concentration of critical substances within the cell usually changes. Such changes modify the metabolite network by making certain reactions more or less likely to occur. Analyzing the metabolites in these different states can allow new insights into these interaction networks. Santos et al. [28] present a graph-based concept to model such behavior, as well as the network perturbation mentioned above, in which the problem of drawing graphs simultaneously arises.

Social Sciences. Recent studies often use graphs to describe the relationships of groups or individuals. E.g., in a study of population structures within a town [24], the bonds between citizens play a major role. Such connections between people can be created in different ways, such as blood relationship, geographical neighborhood or the membership to an association, giving rise to a set of multiple relationship graphs. It is essential to study the whole collection of connections but also to be able to verify the different types of relationships in single examinations.

Evolution of networks. The layout of UML diagrams is a traditional application field for graph drawing. However, often one wants to visualize modifications to these diagrams, as it can be interesting to see the differences between versions of software. This can happen on a large scale, e.g., comparing different release versions of the Java class collection, or on a small scale for the version-differences arising in concurrent versioning systems like CVS and SVN.

The visualization of the development of the world wide web is just one example for a variety of networks evolving over time. It is a natural task to study its development by forming instances for specific points in time (e.g. an instance for every month) and thus creating a sequence of related data.

In all of the described applications the visualization of the occurring relationships between differing data helps the user to understand the structure he is facing. The information can naturally be modelled as a series of graphs with a common set of nodes (proteins, inhabitants, etc.) while the edges differ between the individual graphs, modelling the nodes' relationship based on certain criteria (illness, time, type of relationship, etc.).

The layout of several graphs has become an important task within graph drawing. During the 2006 Graph Drawing Symposium, a contest [14] was held with the goal to best visualize a set of graphs extracted from the statistics of all the soccer world championships. Each graph presents the games of an individual championship, whereby the nodes represent the participating countries.

The task to present such series of graphs in form of graph drawings includes the creation of nice layouts for each individual graph but also the support of understanding the relationship between these graphs. Erten et al. [9] fix the two important criteria for simultaneous graph drawing: the readability of the individual layouts and the mental map preservation in the series of drawings.

1.1 Overview on this paper

Formally, a drawing of a graph G on the plane is an injection of its vertices to points in the plane and a mapping of the edges to simple continuous curves between the images of their endpoints, without containing the image of any other vertex. Any point other than the images of the vertices may only be contained in at most two curves. Such a point which is contained in exactly two curves is called a *crossing*. A drawing without crossings is called *plane*; we can obtain a *planarization* of G by substituting each crossing by a dummy vertex. The crossing minimization problem is to find a drawing of G with the smallest number of crossings. The problem is known to be NP-complete [12] and has been studied extensively in literature, see, e.g., [33] for an extensive bibliography.

In this paper, we deal with the problem of *crossing minimization for simultaneous graphs*, which, to our knowledge, has not been studied before: given a series of graphs, we consider layouts for each graph such that all vertices and edges belonging to more than one graph are drawn identically in all drawings in which they appear. The objective is to minimize the total number of edge crossings over all these drawings. We also consider the *weighted* problem, where a small number of crossings may be more important for certain graphs of the series than for others.

We will first briefly summarize the state-of-the-art regarding simultaneous drawings in Section 1.2. We will then formally define the simultaneous crossing number in Section 2, and discuss some of its main differences to the traditional crossing number in Section 3. Therein we show upper and lower bounds for the simultaneous crossing number, and prove its NP-completeness.

In Section 4 we will show how to extend both heuristic and exact crossing minimization algorithms. We use these algorithms for a brief experimental study in Section 5 and offer some insights on the quantitative difference between the two types of crossing numbers.

1.2 Preliminaries

Erten et al. [9] created layout algorithms and visualization schemes for simultaneous drawings, utilizing a modified force-directed method. Kobourov and Pitta [20] presented an interactive multi-user system for drawing graphs simultaneously. Their system uses the help of the human viewer and a rudimentary crossing minimization heuristic to minimize the number of crossings in a straight-line drawing of two graphs simultaneously.

Research on simultaneous embeddings of graphs started in 2003 by Brass et al. [4], and resulted in a number of publications [4, 8, 11, 13, 15, 16] that deal with different kinds of simultaneous embeddings, e.g., simultaneous embeddings without restrictions on the edge drawings, simultaneous embeddings with fixed edges, simultaneous geometric embeddings. The main interest was the examination whether given pairs of graphs allow a simultaneous embedding or not. Recently, also related complexity questions have been studied [10, 13].

2 Defining Simultaneous Crossing Minimization

Let $\{G_i = (V_i, E_i) \mid i = 1, \ldots, k\}$ be a set of graphs called *basic graphs*. We define their *simultaneous graph* $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ by $\mathscr{V} := \bigcup_{i=1,\ldots,k} V_i$ and $\mathscr{E} := \bigcup_{i=1,\ldots,k} E_i$. Instead of creating layouts for each graph G_i we draw $\mathscr{G}$ in the plane. We obtain a drawing for any G_i by deleting all images of the vertices and edges not belonging to G_i. Thus, a drawing Γ of $\mathscr{G}$ leads to a drawing Γ_i for each G_i such that all vertices and edges belonging to more than one graph are drawn equally in all corresponding drawings.

Consider a drawing Γ of $\mathscr{G}$ with crossings. We can distinguish between two different kinds of crossings: Assume there is a crossing between the edges e and f. We say it is a *phantom crossing*, if there is no basic graph G_i which contains both e and f. It is a *proper crossing* otherwise. Hence, phantom crossings do not correspond to a crossing in a drawing of any basic graph, as none of these graphs contains both edges. Thus, a drawing of a simultaneous graph $\mathscr{G}$ yields a set of planar drawings for each basic graph, if it contains only phantom crossings, if any at all.

Definition 1 (Simultaneous Planarity)**.** *A set of graphs G_i, $i = 1, \ldots, k$, as well as their simultaneous graph $\mathscr{G}$, are called* simultaneously planar *if $\mathscr{G}$ can be drawn with only phantom crossings, if any at all.*

Obviously, a planar simultaneous graph is always simultaneously planar. This definition of simultaneous planarity is equivalent to the definition of simultaneous embedding with fixed edges [8, 11, 13, 16]. Thus, we can reformulate a result of Gassner et al. as follows:

Theorem 1 (Gassner et al. [13])**.** *It is NP-complete to decide whether three or more basic graphs are simultaneously planar.*

Theorem 1 shows that testing simultaneous planarity is NP-complete for three or more graphs. The corresponding problem for two graphs is open, without strong conjections for either a polynomial time algorithm or for NP-completeness. The problem to decide whether two (or more) graphs have a simultaneously planar straight-line drawing is known to be NP-hard [10].

We use the new planarity definition instead of the usual embedding definition to emphasize our ambition to minimize crossings and produce simultaneous drawings.

Given a simultaneous graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, we define *crossing costs* $\chi(e,f) := |\{i \in \{1,\dots,k\} \mid e,f \in E_i\}|$ for each edge pair $(e,f) \in \binom{\mathscr{E}}{2}$, i.e., $\chi(e,f)$ is the number of basic graphs which contain both e and f. Hence, $\chi(e,f)$ reflects the number of basic graphs whose induced drawings will have a crossing if e and f cross in a drawing of the simultaneous graph $\mathscr{G}$. Given a drawing Γ of $\mathscr{G}$, we define $\mathscr{R}_\Gamma$ as a multi-set of edge pairs which lists all crossings in Γ. Unlike for the traditional crossing minimization, multiple crossings may be induced by the same pair of edges (e,f) (see Section 3); $\mathscr{R}_\Gamma$ contains exactly as many instances of (e,f) as there are crossings induced by this pair.

We define the simultaneous crossing number of a simultaneous graph $\mathscr{G}$ with respect to a drawing Γ of $\mathscr{G}$ as $\operatorname{scr}(\mathscr{G},\Gamma) := \sum_{(e,f)\in\mathscr{R}_\Gamma} \chi(e,f)$. Thus, $\operatorname{scr}(\mathscr{G},\Gamma)$ is the total number of (proper) crossings in the drawings of all basic graphs which are induced by Γ.

Definition 2 (Simultaneous Crossing Number)**.** *We define the* simultaneous crossing number $\operatorname{scr}(\mathscr{G})$ *of a simultaneous graph $\mathscr{G}$ as the minimum number* $\operatorname{scr}(\mathscr{G},\Gamma)$ *over all drawings Γ of $\mathscr{G}$.*

Note that a phantom crossing between two edges e and f has no effect on the simultaneous crossing number, as $\chi(e,f) = 0$. Furthermore, we easily see the relationship to the previously defined simultaneous planarity.

Proposition 1. *A simultaneous graph is simultaneously planar if and only if its simultaneous crossing number is zero.*

Formally, we can state the problem:

Definition 3 (Simultaneous Crossing Minimization (SCM))**.** *Given a simultaneous graph $\mathscr{G}$, find* $\operatorname{scr}(\mathscr{G})$.

The crossing minimization for the simultaneous crossing number does not include the minimization of phantom crossings as they have cost zero. However, we still think that these crossings should be avoided if possible. Hence we also consider

Definition 4 (Phantom Crossing Aware Simultaneous Crossing Minimization (SCM^+))**.** *Given a simultaneous graph $\mathscr{G}$, find* $\operatorname{scr}(\mathscr{G})$ *and the smallest number of phantom crossings possible among all drawings realizing* $\operatorname{scr}(\mathscr{G})$.

If not mentioned otherwise, we refer to a drawing realizing SCM^+ as an *optimal drawing*. Clearly, a solution to this problem also solves SCM. Considering SCM^+ instead of SCM becomes crucial for the implementation and runtime analysis of our algorithms.

Sometimes a small number of crossings is more important for certain basic graphs than for others, e.g., we might want to have few crossings in the first and the last graph of the series. This can be achieved by considering a function $w : \{1,\dots,k\} \to \mathbb{R}^+ \setminus \{0\}$ that specifies a positive weight for each graph G_i. We then define *weighted crossing costs* $\chi_w(e,f) := \sum_{i\in M_{e,f}} w(i)$ where $M_{e,f} = \{i \in \{1,\dots,k\} \mid e,f \in E_i\}$ is the index set of the basic graphs that contain e and f. This leads to the *Weighted Simultaneous Crossing Number* $\operatorname{wscr}(\mathscr{G},w)$, and therefore to the *Weighted Simultaneous Crossing Minimization* (wSCM) and the *Phantom Crossing Aware Weighted Simultaneous Crossing Minimization* (wSCM^+) problems.

While this paper will discuss SCM and SCM^+ only, all algorithms can be used for wSCM and wSCM^+ as well.

3 Bounds and Complexity

Gassner et al. [13] also discuss the relationship between simultaneous planarity and the weak realizability problem. Furthermore, the problem is closely related to the abstract topological graph problem introduced by Kratochvíl [21]. Both the abstract topological graph problem and the simultaneous crossing minimization problem are quite different from traditional crossing minimization as we demonstrate in Figure 4 with a slight modification of an example by Kratochvíl.

It is well-known that every graph admits a drawing with a minimum number of crossings, in which adjacent edges do not cross and non-adjacent edges cross each other at most once. However, these facts do not hold for simultaneous crossing minimization:

Proposition 2. *There are simultaneous graphs $\mathscr{G}$ with $k \geq 2$ whose optimal drawings require that pairs of edges cross multiple times.*

Proof. An example of a simultaneous graph with only two basic graphs where a pair of edges crosses at least twice in an optimal SCM^+ drawing can be extracted from [10], cf. Figure 3: define the simultaneous graph with its two basic graphs as described in the construction to the proof to Theorem 1 in [10] for the 3SAT-instance $(x \vee x \vee x) \wedge (\bar{x} \vee \bar{x} \vee \bar{x})$. As this formula is not satisfiable, there is no simultaneous geometric embedding of $\mathscr{G}$ and in particular, there is no simultaneously planar drawing of the instance that involves single crossings on each pair of edges. However, it was shown that there exists a simultaneously planar drawing if an arbitrary number of phantom crossings is allowed: One clause is satisfiable while the other is not, depending on the value of x. The satisfiable one is no problem as it can be drawn straight-line; the other one, however, always involves an edge pair that cross twice in any simultaneously planar drawing. □

Proposition 3. *There are simultaneous graphs $\mathscr{G}$ with $k \geq 3$ whose optimal drawings require that pairs of adjacent edges cross multiple times.*

Proof. In Figure 4, we present a simultaneous graph $\mathscr{G}$ consisting of three basic graphs G_i, $i = 1,2,3$, that has simultaneous crossing number zero but all possible drawings involve three phantom crossings on a pair of adjacent edges. By enlarging the example at the zigzag line, we obtain even linearly many phantom crossings between these two edges. □

For a simultaneous graph $\mathscr{G}$ neither the traditional crossing number is bounded by the simultaneous crossing number nor vice-versa. In fact, Figure 4 shows an example of a simultaneously planar (no proper crossings) but not planar (traditional crossing number greater than zero) graph. On the other hand, assume a non-planar graph G and construct a simultaneous graph $\mathscr{G}$ by defining basic graphs $G_1 = G_2 = G$. Then the simultaneous crossing number is twice the traditional crossing number of $\mathscr{G}$. Using this idea, we develop an upper bound of the simultaneous crossing number:

Lemma 1. *Let $\mathscr{G}$ be a simultaneous graph with k basic graphs. Then* $\operatorname{scr}(\mathscr{G}) \leq k \cdot \operatorname{cr}(\mathscr{G})$.

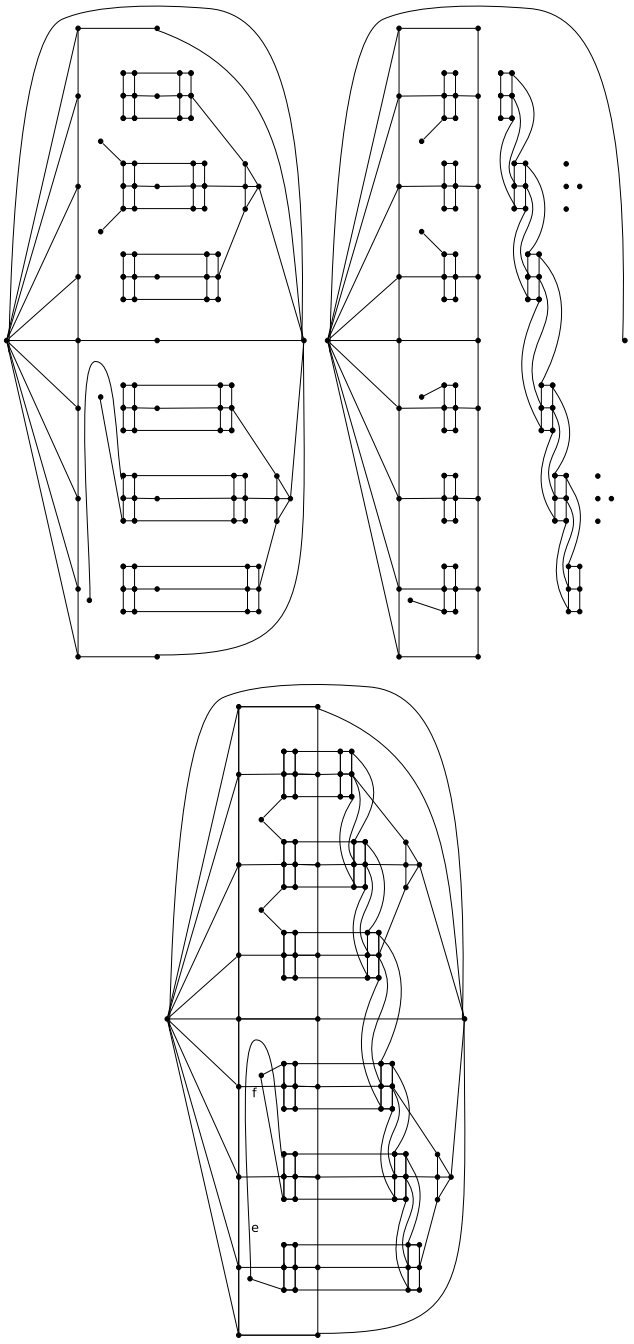


Figure 3: Two simultaneously planar basic graphs (top) where one pair of edges crosses at least twice in an optimal drawing of their simultaneous graph (bottom). In this drawing the edges e and f cross twice.

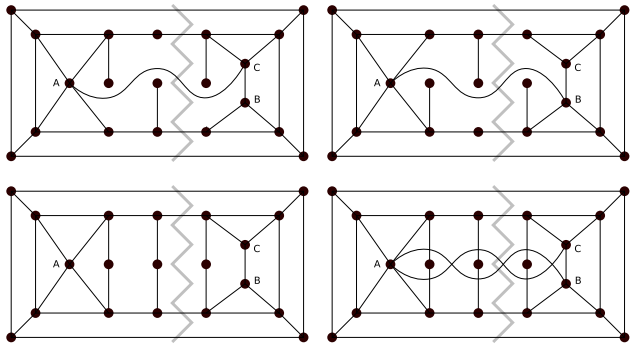


Figure 4: A simultaneously planar simultaneous graph (shown in the bottom right) with three basic graphs. In the drawings that realize SCM$^+$, the adjacent edges (A,B) and (A,C) have multiple crossings.

Proof. Starting with a simultaneous graph $\mathscr{G}$ with k basic graphs, we define a new simultaneous graph $\mathscr{H}$ by a set of k basic graphs $H_1 := \ldots H_k := \mathscr{G}$. By construction we have $\mathrm{scr}(\mathscr{H}) = k \cdot \mathrm{cr}(\mathscr{G})$, where $\mathrm{cr}(\mathscr{G})$ is the ordinary crossing number of $\mathscr{G}$. On the other hand $\mathrm{scr}(\mathscr{G}) \leq \mathrm{scr}(\mathscr{H})$ as every crossing pair in $\mathscr{G}$ costs at most as much in $\mathscr{G}$ as in $\mathscr{H}$. □

Corollary 1. *The simultaneous crossing number* $\mathrm{scr}(\mathscr{G})$*—and therefore the number of proper crossings in any optimal drawing of* $\mathscr{G}$*—is polynomial in the number of vertices and basic graphs for any simultaneous graph* $\mathscr{G}$*.*

A big difference to the traditional problem is the lack of upper bounds for the *cumulative crossing count* $\mathrm{ccc}(\mathscr{G})$, i.e., the sum of phantom and proper crossings, in the solution of an SCM$^+$ instance. As we have seen in Figure 4, two edges can cross more than once. Therefore, we cannot bound the number of crossings per edge by $|\mathscr{E}| - 1$ in an optimal drawing as in traditional crossing minimization. Furthermore we have:

Proposition 4. *There are simultaneous graphs* $\mathscr{G}$ *whose optimal drawings require that an edge is involved in an exponential number of crossings.*

Proof. Figure 5 shows a simultaneous graph with this property in any optimal drawing. The graph is adapted from Kratochvíl and Matoušek [22] where it was used in the context of string graphs. □

It is well-known that the traditional crossing number problem is NP-complete [12]. Since the simultaneous crossing number for a single basic graph (i.e., $k = 1$ and $\mathscr{G} = G_1$) is equal to the ordinary crossing number, we have NP-hardness for SCM. But the fact that an exponential number of phantom crossings may be necessary for any drawing realizing $\mathrm{scr}(\mathscr{G})$ raises the question of NP-membership of SCM. We can prove this membership by showing a relation to the NP-complete Weak Realizability problem [29].

Definition 5 (Weak Realizability)**.** *Given a graph* $G = (V, E)$ *and a set of edge pairs* $R \subseteq \binom{E}{2}$*, does there exist a drawing of* G *where all crossing pairs lie in* R*?*

Lemma 2. *SCM reduces NP-many-one to Weak Realizability.*

Proof. Given a simultaneous graph $\mathscr{G}$ and a positive integral number h, we can test $\mathrm{scr}(\mathscr{G}) \leq h$ in the following way. We guess $\ell \leq h$ pairs of edges with non-zero crossing cost and whose crossing costs sum up to at most h. Our guessing includes the order in which each edge is crossed. It is allowed to guess the same edge pairs multiple times. Each edge e involved in these crossing pairs is split into a path by introducing a new dummy vertex for each guessed crossing. The new dummy vertices have degree four as they are simultaneously used

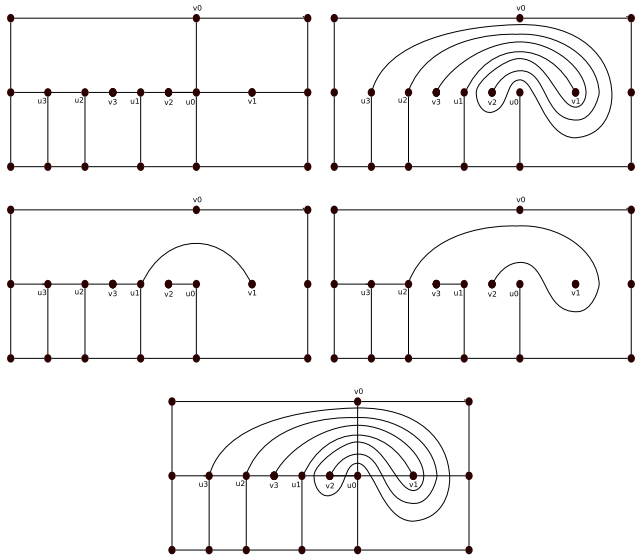


Figure 5: A simultaneously planar simultaneous graph (shown in the lowest picture) with $n+1$ basic graphs. Edge (u_0, v_0) is involved in $2^n - 1$ crossings. The pictures show the case $n = 3$.

in the paths for both edges involved in the corresponding crossing. The guessed crossing order is maintained by the construction of the paths. This transformation can be realized in time polynomially in $|\mathscr{E}|$ and h.

We further define the set of allowed crossing pairs by the set of edge pairs that create a phantom crossing. Path edges inherit their original edge's crossing costs. Edges of one constructed path are, by construction, not allowed to cross each other.

Notice that our guessing requires only a polynomial number of crossings as h and ℓ, the number of proper crossings, are polynomial (cf. Corollary 1). When we correctly guess the crossings (including the order for each edge) that lead to a solution of the original SCM problem, the constructed Weak Realizability problem solves the original SCM problem as it will find the corresponding phantom crossings. □

Since we have NP-hardness for SCM, we can conclude:

Theorem 2. *SCM is NP-complete.*

Corollary 2. *SCM$^+$ is NP-hard.*

This relationship between SCM and Weak Realizability allows us to use a similar result for Weak Realizability (see [27, Theorem 2]) to show that there are at most exponentially many crossings per edge in an optimal drawing. Even more, we have an exponential bound on the number of all crossings of the drawing.

Theorem 3. *The cumulative crossing count* $\mathrm{ccc}(\mathscr{G})$ *in an optimal solution to SCM$^+$ for a simultaneous graph $\mathscr{G}$ is at most exponential. There are at most $(4m)^{12m+24}$ phantom crossings where m is the number of edges in $\mathscr{G}$ while the number of proper crossings is polynomial.*

A main difference to the traditional problem lies in the fact that the actual number of crossings is not proportional to the simultaneous crossing number as a higher number of phantom crossings but lower number of proper crossings is preferred to a higher number of proper crossings (independent of the total number of crossings). Thus, the overall number of crossings in a drawing of $\mathscr{G}$ that realizes SCM or SCM$^+$ cannot be bounded by the crossing number, the simultaneous crossing number, or by the number of crossings in any other drawing of $\mathscr{G}$.

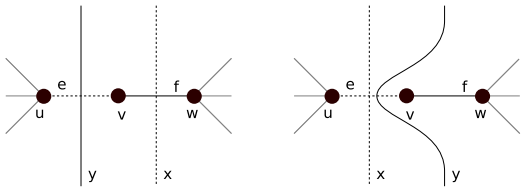


Figure 6: In general, a vertex with degree two cannot be shrinked by the traditional reduction techniques, cf. text. Solid edges are in G_1, dashed edges in G_2. All edges except for e and f are also in G_3.

4 Algorithms

All algorithms, including the reduction techniques, are implemented in the OGDF (Open Graph Drawing Framework) [26], an open-source C++ library for graph drawing algorithms.

4.1 Reduction techniques

In traditional crossing minimization applications the input graphs can usually be reduced in size by graph transformations and transformed back after the crossing minimization. These transformations are done in such a way that the crossing number of the original graph and of the transformed graph are equal. For example, Chimani and Gutwenger [6] developed a non-planar core reduction that yields a significant speed-up in practical crossing minimization computations. However, these graph reduction techniques cannot be used for simultaneous crossing minimization in general. As we shall see, even trivial reductions are not always possible. We describe two reduction techniques for the SCM and SCM$^+$ problems:

Biconnected Components. Trivially, the simultaneous crossing number of a graph is the sum of the simultaneous crossing numbers of the connected components, when computed separately. Furthermore, as for the traditional crossing number problem, it is also valid to solve the problem for each biconnected component separately: the obtained drawings of the blocks can be glued together at their common vertices, without introducing additional crossings. This implies that edges with a degree one vertex can be removed from the problem instance recursively, as they will never cause a crossing in a drawing that realizes SCM$^+$.

Chain Reduction. A *2-chain* is a pair of edges $e = (u,v)$ and $f = (v,w)$, in which the common vertex v has degree two. For the traditional crossing number problem, we could merge them into a single edge $g = (u,w)$. In general, this would be invalid for simultaneous graphs, cf. Figure 6: Assume that $e \in E_1 \setminus E_2$ and $f \in E_2 \setminus E_1$. Let there be two edges $x \in E_1 \cap E_3 \setminus E_2$ and $y \in E_2 \cap E_3 \setminus E_1$. Depending on the relative order in which they cross the 2-chain, they may induce either two phantom crossings, or one phantom and one proper crossing. Replacing e and f by some edge $g = (u,w)$ can never reflect both situations.

Nevertheless, we can define a valid chain reduction based on a subset relation. Let $\mathscr{B}(e)$ and $\mathscr{B}(f)$ be the sets of basic graphs that contain e and f, respectively. If $\mathscr{B}(e) \subseteq \mathscr{B}(f)$, we can replace the edges by $g = (u,w)$ with $\mathscr{B}(g) := \mathscr{B}(e)$. Clearly, this reduction can be performed recursively on the newly generated edge, in order to reduce even longer chains.

4.2 Heuristics

Since crossing minimization is an NP-complete problem, a number of heuristic approaches have been developed over the years. The probably most prominent method is the planarization approach [3]: in a first step a small number of edges is removed such that the remaining graph is planar, and in a second step the deleted edges are reinserted with a minimum number of crossings. Both steps are NP-complete.

Planar Subgraph. If $\mathscr{G}$ is planar in the traditional sense, it is also simultaneously planar. Hence we can use any known algorithm to solve the maximum or maximal planar subgraph problem, as it is done for the traditional planarization approach. The resulting graph may not be maximally simultaneously planar, in the sense that we could insert additional edges without introducing proper crossings. But it is maximally simultaneously planar in the sense that it does not contain any phantom crossing, and the insertion of any edge would lead to at least one phantom or proper crossing.

Edge Insertion. The insertion of all removed edges into the planar subgraph with the minimum number of crossings is still NP-hard [25]. Hence the traditional approach is to insert the edges one by one. All occurring crossings are replaced by dummy vertices such that the graph remains planar after each reinsertion step. For the reinsertion of an edge e into a planar graph G, we can fix a combinatorial embedding of G and insert e into this embedding along a shortest-path in the dual graph of G. This can result in an unnecessary high number of crossings if the chosen embedding is disadvantageous. A stronger algorithm [18] inserts e optimally over all possible combinatorial embeddings in linear time.

We can adapt both algorithms for usage in simultaneous crossing minimization by modifying the crossing cost calculations. Given an edge e and a planar simultaneous graph $\mathscr{G}$, we want to add e to $\mathscr{G}$ such that all introduced crossings lie on e and the simultaneous crossing number of the resulting graph is minimized. In traditional crossing minimization, the crossing cost for each edge f already in $\mathscr{G}$ is independent of e; in the unweighted case it is fixed to 1. However, in simultaneous crossing minimization the cost for crossing f depends on e: the cost is given by $\chi(e,f)$, i.e., the number of basic graphs which contain both e and f. As mentioned above, we also want to minimize the number of phantom crossings and set the cost for edges f that do not have a common basic graph with e to some small positive number ε. Hence – for each reinsertion step and independently on whether we solve the insertion problem for a fixed embedding or over all embeddings – the crossing cost for each edge f in $\mathscr{G}$ must be calculated anew to reflect the current cost depending on the inserted edge e. This is the only change required in both algorithms.

Clearly, the quality of the solution depends on the order in which the edges are inserted. As discussed in [17], there are several strong post-processing strategies, based on removing and reinserting edges after the first heuristic solution. All of them can also be used for the simultaneous crossing minimization.

4.3 Exact Crossing Minimization

Recently, Buchheim et al. [5] presented an integer linear programming (ILP) formulation of the crossing minimization problem and demonstrated how to solve it to optimality with a Branch-and-Cut approach. This approach was made more practical for medium sized graphs by the introduction of a combinatorial column generation scheme by Chimani et al. [7]. In this section, we show how to extend this approach to solve the SCM$^+$ problem to optimality.

Like the heuristic algorithms above, we must change the crossing cost per edge pair. But this seemingly straight-forward modification is not sufficient.

The basic ILP. We will only briefly sketch the ILP approach mentioned above; for details see [5]. The conceptual idea is to have a variable $\bar{x}_{e,f}$ for each pair of edges which is 1 if the two edges cross, and 0 otherwise. This variable set would not be sufficient to test the validity of a solution if an edge e is crossed by two or more other edges, because the order of these crossings is unknown. Therefore each edge of the given graph is replaced by a path of α edges, called *segments*, where α is the maximum number of crossings per edge and is bounded both by the number of other edges and by any heuristic solution for the crossing number problem. Hence, we use variables $x_{e,f}$ for each pair of segments in this expanded graph. A column generation scheme, described below, can be used to partially avoid the huge blow-up by this transformation.

A graph is planar if and only if it does not contain any *Kuratowski subdivision*, i.e., a subdivision of a K_5 or a $K_{3,3}$ [23]. The Kuratowski-constraints used in [5] force at least one crossing on each such substructure. We do not need to modify these constraints nor their separation for usage in SCM$^+$.

Variables and Expansion of Edges. For the traditional crossing number, we are able to bound α, the number of crossings per edge, by the number of graph edges and by any heuristic solution. But as discussed in Section 3, there are no bounds of practical interest in case of the SCM$^+$ problem. We define $\mathrm{ccc}(\mathscr{G},\Gamma)$ as the sum of phantom and proper crossing in the drawing Γ of $\mathscr{G}$. We cannot even use $\mathrm{ccc}(\mathscr{G},\Gamma)$, for some heuristically computed Γ, as a valid upper bound for the number of crossings per edge: the optimal drawing Γ^* might have fewer proper crossings than Γ, but require many more phantom crossings, possibly resulting in $\mathrm{ccc}(\mathscr{G}) = \mathrm{ccc}(\mathscr{G},\Gamma^*) > \mathrm{ccc}(\mathscr{G},\Gamma)$.

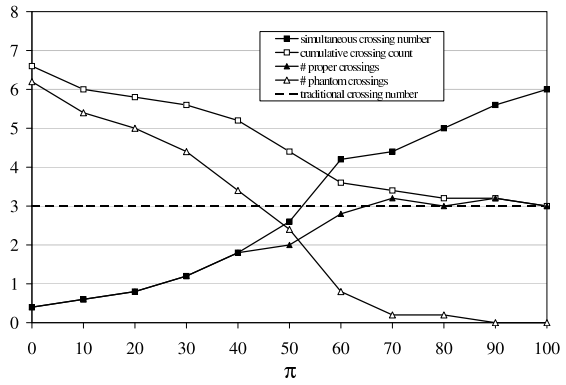
Hence we consider a maximum expansion with exponentially many segments per edge. This drawback, although terrifying on first sight, turns out to be of little relevance in practice, since the column generation scheme described below does not generate this enormous amount of additional variables in any of our tests. Hence it becomes clear that the use of the column generation scheme is not only an enhancement to improve the runtime, but the column generation scheme becomes a compulsory element of the algorithm.

Also, the original approach allows to leave out the variables $x_{e,f}$ for adjacent edges e and f. Additionally, it uses constraints that allow at most one crossing per pair of original edges. Both these possibilities are no longer valid.

Crossing Costs and Column Generation Scheme. The main idea for computing a solution to the SCM$^+$ problem with our ILP is to use the crossing cost $\chi(e,f)$ as the coefficient of the variable $x_{e,f}$ in the objective function, for each pair of edges e,f. Since phantom crossings correspond to $\chi(e,f) = 0$, the ILP would not solve SCM$^+$ correctly. Hence we set the crossing costs for phantom crossing to some small enough value $\hat{\varepsilon} > 0$. This approach comes with certain difficulties regarding the column generation scheme: [7] compared two different column generation schemes, a general approach based on reduced costs, and a second approach based on combinatorial arguments. It became clear that the latter is far more efficient and should therefore be used.

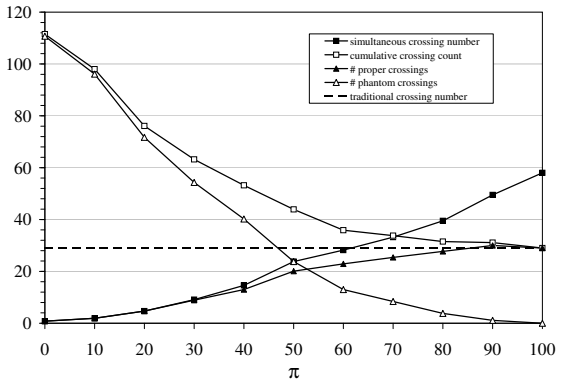
The central idea of the combinatorial column generation scheme is to start with one segment per edge, and adding additional segments when necessary. It is crucial for the validity of the algorithm that these additional segments are a bit cheaper to cross than the first segment of an edge, say by some small enough $\varepsilon^* > 0$. Hence we have to be careful about mixing these two different epsilons. Note that also crossings with cost $\hat{\varepsilon}$ have to be reduced by ε^* for later (i.e., not the first) segments. Hence we require $\varepsilon^* < \hat{\varepsilon}$ to prohibit negative costs. But we cannot choose $\varepsilon^* \ll \hat{\varepsilon}$ due to the risk of numerical instabilities.

A nice property of the original column generation approach, using a suitable ε^*, is that the objective value of the ILP can always be rounded up to obtain the unskewed real integer objective value, i.e., any solution of the ILP is of the form $\left\lceil \mathrm{cr}(G) - \ell \cdot \varepsilon^* - \ell' \cdot \varepsilon^{*2} \right\rceil = \mathrm{cr}(G)$. Thereby, ℓ and ℓ' reflect the number of crossings on later segments. We loose this property by the introduction of $\hat{\varepsilon}$, since crossings with such cost have to be rounded down to obtain the graph theoretic crossing cost 0. Hence the ILP solution is of the form $\mathrm{scr}(\mathscr{G}) - \ell \cdot \varepsilon^* - \ell' \cdot \varepsilon^{*2} + \hat{\ell} \cdot \hat{\varepsilon}$ which can be greater than $\mathrm{scr}(\mathscr{G})$. Overall, we can in fact use both ε^* and $\hat{\varepsilon}$ simultaneously, choosing, e.g., $2\varepsilon^* = \hat{\varepsilon}$, but we must adapt all bounding schemes accordingly. By choosing the epsilons carefully, we can change the $\lceil . \rceil$-function



π	scr	*prop*	*phan*	ccc
0	0.4	0.4	6.2	6.6
10	0.6	0.6	5.4	6.0
20	0.8	0.8	5.0	5.8
30	1.2	1.2	4.4	5.6
40	1.8	1.8	3.4	5.2
50	2.6	2.0	2.4	4.4
60	4.2	2.8	0.8	3.6
70	4.4	3.2	0.2	3.4
80	5.0	3.0	0.2	3.2
90	5.6	3.2	0.0	3.2
100	6.0	3.0	0.0	3.0

Figure 7: (Exact) The figure shows the computed simultaneous crossing number (scr), the number of proper (*prop*) and phantom (*phan*) crossings and the cumulative crossing count (ccc) for different values of π. The underlying graph has 10 nodes and 23 edges. The results were computed using the exact Branch-and-Cut algorithm. For each value π, the values were averaged over 5 simultaneous graphs generated from G.



π	scr	*prop*	*phan*	ccc
0	0.8	0.8	110.7	111.5
10	1.9	1.9	96.1	98.0
20	4.7	4.7	71.7	76.4
30	9.1	8.9	54.3	63.2
40	14.6	13.0	40.2	53.2
50	23.8	20.1	23.8	43.9
60	28.2	22.9	13.0	35.9
70	33.2	25.4	8.4	33.8
80	39.5	27.7	3.8	31.5
90	49.5	30.0	1.1	31.1
100	58.0	29.0	0.0	29.0

Figure 8: (Heuristic) The figure shows the computed simultaneous crossing number (scr), the number of proper (*prop*) and phantom (*phan*) crossings and the cumulative crossing count (ccc) for different values of π. The underlying graph G has 33 nodes and 70 edges. The results were computed heuristically. For each value π, the values were averaged over 10 simultaneous graphs generated from G.

into a rounding scheme which transforms any value within the interval $[a-0.5, a+0.5)$ into the integral value a, and still use the combinatorial column generation scheme.

5 Experiments

We implemented both heuristic approaches, as well as the exact algorithm, as part of the open-source C++-library *Open Graph Drawing Framework* [26]. We ran experiments to test the behavior of the presented crossing minimization concept.

First, we created a set of 10 random graphs with varying size. We then interpreted each graph G of this set as a simultaneous graph of two basic graphs by randomly choosing the basic graphs for each edge of G: we set the probability for every edge to belong to both basic graphs to some percentage π. All edges which do not belong to both basic graphs had an equal chance to belong to either basic graph. We used different values for π: 0, 10, 20, …, 90, and created a set of 10 simultaneous graphs for each original graph and each value of π. This resulted in 1000 simultaneous graphs.

We applied the heuristic algorithms to this test set and computed the simultaneous crossing number, together with the number of phantom and proper crossings. Assume a simultaneous graph $\mathscr{G}$ generated from G with $\pi = 100$: all edges belong to both basic graphs and thus this instance reflects the traditional crossing minimization problem: there are no phantom crossings, we have $\text{scr}(\mathscr{G}) = 2 \cdot \text{ccc}(\mathscr{G})$, and for optimal solutions we have $\text{ccc}(\mathscr{G}) = \text{cr}(G)$. We performed similar experiments with the exact algorithm. These were run on smaller and less dense graphs, as the algorithm's running time is highly dependent on the crossings number (cf. [7]), in our case on the cumulative crossing count $\text{ccc}(\mathscr{G})$.

Generally, we encountered the same effects throughout all instances and algorithms. Figure 7 and Figure 8 show two representative examples: the former one was computed via the exact algorithm, the second one via the heuristic which optimizes the edge insertions over all planar embeddings. The heuristic-specific parameters are the ones proposed in [17].

The simultaneous crossing number and the number of proper crossings increase as π gets larger. On the other hand, the number of phantom crossings strongly decreases at the same time, and thus the total number of crossings slightly decreases. Interestingly, while the simultaneous crossing number of course monotonically increases, this does not hold in general for the number of proper crossings due to their differing crossing costs.

Over all values for π, the runtime of the heuristic algorithms was in general roughly four times larger than the original heuristics, most runtime loss was due to the more complex crossing cost calculation for each pair of edges. For the tested ILP instances, the heuristic was always able to find the optimal solution. Hence the exact Branch-and-Cut algorithm had only to proof this optimality: the running time was thereby highly dependent on the bounding efficiency, ranging from 10 to 1000 seconds for graphs with a crossing count of up to 7.

6 Open Problems

We showed SCM to be NP-complete (Theorem 2), while SCM^+ is NP-hard (Corollary 2). The NP-membership of SCM^+ remains as an open problem.

In Proposition 3 we required three basic graphs to force adjacent edges to cross. It is yet open whether such crossings can be forced with only two basic graphs.

In Proposition 4 we required an unbounded number of basic graphs to force an exponential number of crossings on a single edge.

Is it possible to find an example with a fixed number of basic graphs?

7 Conclusion

In this paper we examined crossing minimization in the context of simultaneous graph drawing. We developed a natural way to extend the concept of crossing minimization for simultaneous graphs and discussed differences to the ordinary crossing number. We showed how traditional crossing minimization methods and algorithms can be adapted to applications where more than one graph is given at a time. We modified existing heuristic crossing minimization algorithms and used an approach to the exact crossing minimization problem to construct the first exact crossing minimization method for simultaneous drawing.

Furthermore, the latter algorithm gives us the first simultaneous planarity testing algorithm, since it recognizes simultaneously planar graphs $\mathscr{G}$ by computing $\text{scr}(\mathscr{G}) = 0$. It is not surprising that the runtime of this algorithm is not practical for large graphs as the corresponding combinatorial problem is NP-complete (cf. Theorem 1) for three or more basic graphs.

Acknowledgements

We would like to thank Marcus Schaefer for pointing out the relationship between SCM and Weak Realizability and the resulting upper bound on the number of crossings as described in Theorem 3, and Karsten Klein for sharing his biochemical knowledge.

References

[1] P. Aloy and R. B. Russell. Structure-based systems biology: a zoom lens for the cell. *FEBS Lett.*, 579(8):1854–1858, 2005.

[2] S. Bandyopadhyay, R. Sharan, and T. Ideker. Systematic identification of functional orthologs based on protein network comparison. *Genome Res.*, 16(3):428–435, 2006.

[3] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *Journal of Systems and Software*, 4:163–173, 1984.

[4] P. Braß, E. Čenek, C. A. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous planar graph embeddings. *Comput. Geom.*, 36(2):117–130, 2007.

[5] C. Buchheim, M. Chimani, D. Ebner, C. Gutwenger, M. Jünger, G. W. Klau, P. Mutzel, and R. Weiskircher. A branch-and-cut approach to the crossing number problem. *Discrete Optimization*, 2007. to appear. A preliminary version appeared in *Proc. GD '05*, LNCS 3843, pp. 37–48.

[6] M. Chimani and C. Gutwenger. Non-planar core reduction of graphs. *Discrete Mathematics*, 2007. to appear. A preliminary version appeared in *Proc. GD '05*, LNCS 3843, pp. 223–234.

[7] M. Chimani, C. Gutwenger, and P. Mutzel. Experiments on exact crossing minimization using column generation. In M. Serna, editor, *Proc. WEA '06*, volume 4007 of *LNCS*, pages 303–315, 2006.

[8] C. Erten and S. G. Kobourov. Simultaneous embedding of planar graphs with few bends. In J. Pach, editor, *Proc. GD '04*, volume 3383 of *LNCS*, pages 195–205, 2005.

[9] C. Erten, S. G. Kobourov, V. Le, and A. Navabi. Simultaneous graph drawing: Layout algorithms and visualization schemes. In G. Liotta, editor, *Proc. GD '03*, volume 2912 of *LNCS*, pages 437–449, 2004.

[10] A. Estrella-Balderrama, E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous geometric graph embeddings. In *Proc. GD '07*, volume 4875 of *LNCS*, 2007. to appear.

[11] F. Frati. Embedding graphs simultaneously with fixed edges. In M. Kaufmann and D. Wagner, editors, *Proc. GD '06*, volume 4372 of *LNCS*, pages 108–113, 2007.

[12] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[13] E. Gassner, M. Jünger, M. Percan, M. Schaefer, and M. Schulz. Simultaneous graph embeddings with fixed edges. In F. V. Fomin, editor, *Proc. WG '06*, volume 4271 of *LNCS*, pages 325–335, 2006.

[14] GD'06 competition. http://gd2006.org/contest/, 2006.

[15] M. Geyer, M. Kaufmann, and I. Vrt'o. Two trees which are self-intersecting when drawn simultaneously. In P. Healy and N. S. Nikolov, editors, *Proc. GD '05*, volume 3843 of *LNCS*, pages 201–210, 2006.

[16] E. D. Giacomo and G. Liotta. A note on simultaneous embedding of planar graphs. In *Proc. EWCG '05*, pages 207–210, 2005.

[17] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In G. Liotta, editor, *Proc. GD '03*, volume 2912 of *LNCS*, pages 13–24, 2004.

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

[19] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood. Integrated genomic and proteomic analyses of a systematically pertubed metabolic network. *Science*, 1(292):929–934, 2001.

[20] S. G. Kobourov and C. Pitta. An interactive multi-user system for simultaneous graph drawing. In J. Pach, editor, *Proc. GD '04*, volume 3383 of *LNCS*, pages 492–501, 2005.

[21] J. Kratochvíl. Crossing number of abstract topological graphs. In S. H. Whitesides, editor, *Proc. GD '98*, volume 1547 of *LNCS*, pages 238–245, 1998.

[22] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory*, B 53:1–4, 1991.

[23] C. Kuratowski. Sur le problème des courbes gauches en topologie. *Fund. Math.*, 15:271–283, 1930.

[24] C. Lipp. Corporate birthmarks of civil society: Kinship and kinship networks in the early 19th century. *Journal of Family History*, 30(4):347–365, 2005.

[25] P. Mutzel and T. Ziegler. The constrained crossing minimization problem. In J. Kratochvil, editor, *Proc. GD '99*, volume 1731 of *LNCS*, pages 175–185, 1999.

[26] *OGDF – Open Graph Drawing Framework*. http://www.ogdf.net, 2007.

[27] J. Pach and G. Tóth. Recognizing string graphs is decidable. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. GD '01*, volume 2265 of *LNCS*, pages 247–260, 2002.

[28] S. D. M. Santos, P. Verveer, and P. I. H. Bastiaens. Growth factor-induced MAPK network topology shapes erk response determining pc-12 cell fate. *Nature Cell Biology*, 9:324–330, 2007.

[29] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in *NP*. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[30] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proc Natl Acad Sci USA*, 102(6):1974–1979, 2005.

[31] M. Suderman and M. Hallett. Tools for visually exploring biological networks. *Bioinformatics*, 2007.

[32] K. Tan, T. Shlomi, H. Feizi, T. Ideker, and R. Sharan. Transcriptional regulation of protein complexes within and across species. *PNAS*, 104(4):1283–1288, 2007.

[33] I. Vrt'o. Crossing numbers of graphs: A bibliography. ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf, 2007.