

Graph Drawing with Algorithm Engineering Methods

Edited by

Camil Demetrescu¹, Michael Kaufmann², Stephen Kobourov³, and
Petra Mutzel⁴

- 1 Sapienza University of Rome, IT, demetres@dis.uniroma1.it
- 2 Universität Tübingen, DE, mk@informatik.uni-tuebingen.de
- 3 University of Arizona – Tucson, US, kobourov@cs.arizona.edu
- 4 TU Dortmund, DE, petra.mutzel@cs.tu-dortmund.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 11191 “Graph Drawing with Algorithm Engineering Methods”. We summarize the talks, open problems, and working group discussions.

Seminar 8.–13. May, 2011 – www.dagstuhl.de/11191

1998 ACM Subject Classification E.1 Data structures, F.2 Analysis of algorithms and problem complexity, G.1.6 Optimization, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Algorithm Engineering, Graph Drawing

Digital Object Identifier 10.4230/DagRep.1.5.47

Edited in cooperation with Carsten Gutwenger and Karsten Klein

1 Executive Summary

Camil Demetrescu
Michael Kaufmann
Stephen Kobourov
Petra Mutzel

License  Creative Commons BY-NC-ND 3.0 Unported license
© Camil Demetrescu, Michael Kaufmann, Stephen Kobourov, and Petra Mutzel

Automated graph drawing deals with the layout of relational data arising from computer science (database design, data mining, software engineering), and other sciences such as bioinformatics and sociology (social networks). The relational data are typically modeled as graphs, which can be visualized through diagrams drawn in the plane. The main objective is to display the data in a meaningful fashion, (i.e., in a way that shows well the underlying structures) and that often depends on the application domain. Although high quality algorithms exist for many optimization problems that arise in graph drawing, they are often complex and difficult to implement, and theoretically efficient algorithms may have unacceptable runtime behavior even for small-to-medium sized real-world instances. Also large graphs like, e.g., molecular interaction networks, may render exact but complex algorithms infeasible and require approximate or heuristic solutions.

Integrating automated graph drawing techniques into real-world software systems poses several algorithm engineering challenges. To achieve effective implementations, algorithms and data structures designed and analyzed on abstract machine models must be carefully tuned for performance on real hardware platforms. This task is becoming increasingly more difficult due to the impressive growth of data to be visualized in modern applications, as



Except where otherwise noted, content of this report is licensed
under a Creative Commons BY-NC-ND 3.0 Unported license

Graph Drawing with Algorithm Engineering Methods, *Dagstuhl Reports*, Vol. 1, Issue 1, pp. 47–60

Editors: Camil Demetrescu, Michael Kaufmann, Stephen Kobourov, and Petra Mutzel



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

well as their highly dynamic and data-intensive nature. Developers can no longer ignore architectural aspects such as the presence of complex memory hierarchies and multiple cores, which are likely to shape the design of novel algorithmic techniques and the way they will be implemented and engineered in the future.

The aim of this seminar was to bring together researchers from the algorithm engineering and graph drawing communities in order to strengthen and foster collaborations in this area and to identify key research directions for the future.

The seminar was attended by 48 participants from both academia and industry. Much was accomplished, fostered by the productive atmosphere of the Dagstuhl Center. Here we describe some of the more important achievements.

The program consisted of a wide variety of presentations, working group sessions and discussion sessions. The presentations included several survey lectures:

- Beppe Liotta provided a survey on graph visualization paradigms, and discussed general design principles for the realization of effective graph visualization systems.
- Emden Gansner suggested rules in order to get efficient and accurate graph drawing algorithms.
- Ulrik Brandes discussed experimental algorithmics and the relationship between graph drawing algorithms and algorithm engineering.
- Rudolf Fleischer's talk about algorithm engineering and his statement (taken from the definition in the German priority program SPP 1307 *Algorithm Engineering*) that the algorithm engineering cycle should be driven by falsifiable hypotheses, started a lively discussion among the participants.
- Rico Jakob provided a talk on engineering architecture aware algorithms and provided some thoughts about hardware sensitive algorithms. He convinced us that the new computer architectures will strongly influence future algorithmic research.
- Kurt Mehlhorn introduced us into the new and exciting area of slime mould that solves shortest path and network design problems. He would be interested in seeing if slime mould could possible solve graph drawing problems.

Beyond the survey lectures, highlights of the seminar included the two introductory sessions, the open problem sessions, and the working groups.

In two sessions, we have identified over two dozen open problems, which later crystallized into about a dozen well-defined problems, each of which were of interest to several participants. We had working groups on the following topics: Rotating binary trees, feedback arc set convergence, edge bundling models, co-occurrence in bipartite graphs, RAC drawings, BRAC drawings, minimum branch spanning tree, cluster tree embedding, point set embeddings, parallel graph drawing, and library of graphs. Participants shared ideas and material using the online seminar Wiki.

The dissemination sessions at the end of the workshop showed that many of the working groups have achieved initial results, which may lead to future publications.

Arguably the most-appreciated features of the Seminar were the lively open discussion sessions, which led to several concrete proposals for the future of the field which, as a result of the workshop, are now being actively pursued.

A big step forward has been done concerning an *online library of graphs*. The graph drawing community would like to have such a library, however, there was no consensus about the requirements on such a graph archive. The working group conducted a survey on requirements for a graph archive during the Dagstuhl seminar. Two groups (Dortmund and Tübingen) presented their ideas and prototypes of such an archive. In order to foster

future work and to encourage participation and contributions, it was suggested that the GD proceedings should offer the opportunity to publish papers concerning the library. Moreover, the collection of many benchmark graphs has already begun.

We used the opportunity to bring together experts in algorithm engineering for multi-core algorithms with graph drawing researchers in order to discuss how graph drawing algorithms can be re-engineered to better take advantage of modern computer architecture into account. This working group was inspired by the many different backgrounds of group members. They have discussed how to improve data locality, or exploit multi-core processors, in particular for the widely used Sugiyama drawing method.

Subjectively (from interacting with the attendees) and objectively (from the official feedback data) we believe that the participants enjoyed the great scientific atmosphere offered by Schloß Dagstuhl, and profited from the scientific program and the fruitful discussions. We are grateful for having had the opportunity to organize this seminar. Special thanks are due to Carsten Gutwenger and Karsten Klein for their invaluable assistance in the organization and the running of the seminar.

2 Table of Contents

Executive Summary

Camil Demetrescu, Michael Kaufmann, Stephen Kobourov, and Petra Mutzel . . . 47

Overview of Talks

Experimental Algorithmics
Ulrik Brandes 51

AE meets GD
Rudolf Fleischer 51

Notes on Practical Graph Drawing
Emden R. Gansner 51

Engineering Architecture Aware Algorithms – two case studies and some thoughts
on Algorithms Engineering
Riko Jacob 52

The Anatomy of a graph visualization system
Giuseppe Liotta 52

Working Groups

Graph Archive
*Christian Bachmaier, Franz J. Brandenburg, Philip Effinger, Carsten Gutwenger,
Jyrki Katajainen, Karsten Klein, Miro Spönemann, and Michael Wybrow* 52

Spanning Trees with Few Branches
Markus Chimani, Aparna Das, and Joachim Spoerhase 53

Parallel Graph Drawing
*Deepak Ajwani, Camil Demetrescu, Carsten Gutwenger, Robert Krug, Henning
Meyerhenke, Petra Mutzel, Stefan Näher, Georg Sander, and Matthias Stallmann* . 54

Circular-Arc Drawings with Right-Angle Crossings
*Muhammad Jawaherul Alam, Martin Nöllenburg, Sankar Veeramoni, and Kevin
Verbeek* 55

Edge Bundling
*David Auber, Stefan Diehl, Christian Duncan, Cesim Erten, Rudolf Fleischer,
Emden Gansner, Michael Kaufmann, Lev Nachmanson, and Michael Wybrow* . . . 56

On the Rotation Distance of Rooted Binary Trees
*Ulrik Brandes, Rudolf Fleischer, Seok-Hee Hong, Tamara Mchedlidze, Ignaz Rutter,
and Alexander Wolff* 57





Participants 60

3 Overview of Talks

In this section, we report the abstracts of five survey talks on algorithm engineering aspects arising in graph drawing applications. The presentations provided a starting point for the seminar and covered foundational aspects, allowing participants with different grounds of expertise to share common methodologies and goals.

3.1 Experimental Algorithmics

Ulrik Brandes (Universität Konstanz, DE)



License     Creative Commons BY-NC-ND 3.0 Unported license
© Ulrik Brandes

Joint work of Brandes, Ulrik; Karrenbauer, Andreas

The algorithm engineering cycle is said to be driven by falsifiable hypotheses that are validated by experiment. There is little evidence, however, that this is indeed common practice. We sketch the concepts of formal experimentation as established in other disciplines and propose a mapping to experimental algorithmics. After reviewing experimental work in graph drawing, we ask whether more formal experimentation is needed.

3.2 AE meets GD





Rudolf Fleischer (Fudan University – Shanghai, CN)

License     Creative Commons BY-NC-ND 3.0 Unported license
© Rudolf Fleischer

The scientific method postulates a certain framework for doing experiments and how to interpret experimental results theoretically. Experimentors in computer science and graph drawing seem sometimes to be unaware of these principles.

3.3 Notes on Practical Graph Drawing

Emden R. Gansner (AT&T Research – Florham Park, US)

License     Creative Commons BY-NC-ND 3.0 Unported license
© Emden R. Gansner


To be useful, graph drawing algorithms need to be engineered to be efficient and accurate. To be used, it helps if the graph drawing algorithms are implemented and run using some practical rules of thumb. These include:

- Follow general software engineering principles
- Use optimizations where possible; use heuristics when necessary
- Leverage the geometry
- Provide the user with a rich set of drawing features
- Construct simple, flexible, reusable interfaces
- Stress robustness, especially as regards scalability

In this talk, we discuss these rules, providing motivations and examples.

3.4 Engineering Architecture Aware Algorithms – two case studies and some thoughts on Algorithms Engineering


Riko Jacob (TU München, DE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Riko Jacob

I will present preliminary results from two ongoing projects, both regarding memory intensive tasks in (huge) internal memory. One is asking the question if I/O-efficient (sorting based) algorithms can outperform the direct algorithm when permuting several gigabytes of data in internal memory. The other one reports on an efficient implementation for numerical computations in high dimensional settings using so called sparse grids. Again, the improved implementation is inspired by I/O-efficient algorithms. Finally, I will try to show how these two examples fit into a more general theme of using and combining theoretical models and experiments to find the algorithm and implementation that performs best on a given hardware.

3.5 The Anatomy of a graph visualization system

Giuseppe Liotta (University of Perugia, IT)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Giuseppe Liotta

Graph visualization addresses the problem of efficiently conveying the structure of relational information, which is typically modeled by networks. Therefore, graph visualization systems are largely used for information exploration and knowledge discovery, particularly in those applications that need to manage, process and analyze large sets of data. The design of a graph visualization systems typically addresses questions that belong to the intersection of different disciplines, such as graph algorithms, data mining, software engineering, algorithm engineering, and visual analytics. In this talk I will shortly review some common and emerging graph visualization paradigms, discuss general design principles present application examples, and compare different models for the realization of effective graph visualization systems.

4 Working Groups

4.1 Graph Archive

Christian Bachmaier, Franz J. Brandenburg, Philip Effinger, Carsten Gutwenger, Jyrki Katajainen, Karsten Klein, Miro Spönemann, and Michael Wybrow


License  Creative Commons BY-NC-ND 3.0 Unported license
© Christian Bachmaier, Franz J. Brandenburg, Philip Effinger, Carsten Gutwenger, Jyrki Katajainen, Karsten Klein, Miro Spönemann, and Michael Wybrow

In order to evaluate, compare, and tune graph (drawing) algorithms, experiments on well designed benchmark sets have to be performed. This, together with the goal of reproducibility of experimental results, creates a demand for an adequate archive to gather and store graph instances. Such an archive would ideally allow annotation of instances or sets of graphs with

additional information like graph properties and references to the respective experiments and results. We examine the requirements and formulate the next steps needed to produce an easily accessible library of graphs that provides the required services. Through successful community involvement, it is expected that the archive will contain a representative selection of both real-world and generated graph instances, covering significant application areas as well as relevant classes of graphs.

4.2 Spanning Trees with Few Branches

Markus Chimani, Aparna Das, and Joachim Spoerhase

License  Creative Commons BY-NC-ND 3.0 Unported license
© Markus Chimani, Aparna Das, and Joachim Spoerhase

Given a graph G finding the spanning tree T of G with the smallest number of *branch nodes* (nodes of degree at least 3) is known as the *minimum branch node spanning tree* problem (MBST). The problem is motivated by optical networks where switches of degree greater than two are expensive as they require sophisticated hardware to split light. Unfortunately [1] shows that the problem is not only NP-hard but also nonapproximable.

Observe that a spanning tree that minimizes the number of branchings also maximizes the number of low-degree nodes (leaves and degree-two nodes). Although this maximization version of MBST leads to the same set optimum solutions it has better approximability properties. In fact any spanning tree is already a 2-approximation since at least half the nodes have degree 1 or 2.

We investigate the following local search algorithm. A *legal k -flip* replaces at most k edges of T with the same number of edges in $E(G) - E(T)$ such that the resulting graph T' is a spanning tree of G with strictly more low-degree nodes than T . (In a more refined algorithm a legal k -flip increases a suitably defined potential function.) Starting with an arbitrary spanning tree we perform legal 2-flips until we obtain a spanning tree (local optimum) for which no legal 2-flips can be performed anymore.


We conjecture that the performance guarantee of this algorithm is strictly better than 2. Our current proof sketch suggests a factor of at most 1.8. Our analysis is based on the observation that the performance ratio of 2 can only be achieved by spanning trees in which (almost) all branch nodes have degree 3. Our approach consists in showing that any node u that has degree 3 in a local optimum either has degree 3 in a global optimum, too, or is “associated” with certain nodes of degree $\neq 3$ in T .

References

- 1 L. Gargano, P. Hell, L. Stacho and U. Vaccaro, Spanning trees with bounded number of branch vertices, Proc. of ICALP 02, LNCS vol. 2380, Springer (2002), pp. 355–365.

4.3 Parallel Graph Drawing

Deepak Ajwani, Camil Demetrescu, Carsten Gutwenger, Robert Krug, Henning Meyerhenke, Petra Mutzel, Stefan Näher, Georg Sander, and Matthias Stallmann

License  Creative Commons BY-NC-ND 3.0 Unported license
 © Deepak Ajwani, Camil Demetrescu, Carsten Gutwenger, Robert Krug, Henning Meyerhenke, Petra Mutzel, Stefan Näher, Georg Sander, and Matthias Stallmann

In recent years hardware architects have hit the power wall. Increasing the processor clock speeds further would yield an overproportional power consumption and heat production. That is why smaller circuit designs are exploited by adding more computing cores instead. Nowadays commodity processors already have four or six cores, a number expected to rise significantly in the next decade. Special processors already support hundreds of hardware threads today. To make efficient use of the available hardware for an algorithmic problem, it is necessary to expose the inherent parallelism in a problem at hand. Recently it became popular to use GPUs to accelerate computations. So it seems natural to parallelize graph drawing algorithms. Few papers consider algorithms based on force-directed methods on GPUs and similar architectures.

One of the most popular algorithms for drawing hierarchical layouts of directed graphs is the Sugiyama algorithm [1]. We are not aware of any parallel Sugiyama-type algorithms. The main phases are ranking, crossing minimization and coordinate assignment. The time-critical task is crossing minimization, whose main procedure sweeps repeatedly up and down the hierarchy. A sweep starts at the second level and computes the barycenter or barycenter values, respectively, for each node on that level. These values are normally based on the position of the neighbors in the previous level. Then the vertices are sorted according to these values and the sweep continues with the next level. The dependence on results from a previous step limits the available parallelism. In our group we came up with new ideas for modifying Sugiyama’s method in order to introduce a higher amount of parallelism.

Partitioning-based Approaches.

First ideas of distributing work to processors were based on partitioning. One option is to cut the layers into horizontal slabs and assign each slab to one processing element. Here the border layers may experience conflicted orderings.

We had several ideas to resolve these conflicts, but it remains to be investigated in experiments which of these techniques are successful. The second possibility is the partitioning into vertical strips. The objective is to partition such that the number of edges running between different vertical slabs is small. Although graph partitioning is an NP-hard problem in general, there are efficient tools that yield good solutions. After that each vertical strip is processed by one processing element in parallel. The hope is that the number of bad orderings close to the strip boundaries is small due to the small number of edges between strips.

Odd-even Layer Processing.

The next idea was to reduce the sequential dependencies to exploit more parallelism. If the tasks to be performed have no sequential dependencies, any mapping to processing elements would allow for concurrent and conflict-free computations. In the traditional approach layers must be treated one after each other. Therefore, we tried to decouple the dependency of

adjacent layers. For this purpose we have developed the *odd-even barycentric algorithm* (OEBA).

Instead of up- or downward sweeps, the algorithm proceeds in passes. Similar to odd-even sort, in each pass one processes either the odd or the even layers. All odd layers are mutually independent, likewise all even layers. Hence, in each pass the currently visited layers can be processed independently in parallel.

Within each layer L_i , for each node $v \in L_i$ the new position is computed as the both-sided barycenter, i.e. taking predecessors and successors into account at the same time. Two alternatives have been considered, either taking the average over all neighbors or by taking the average of the barycenter of the neighbors in the layer above and of the barycenter of the neighbors in the below.

Next, the new positions of the nodes within the layer need to be determined. This is either done in the classical way by sorting the nodes according to their barycentric values directly after the barycentric computations of layer L_i have been finished. Alternatively, one computes the barycentric values of all layers first and then sorts all layers at once. These options allow for different types of parallelism and also alter the control flow and thus the output of the algorithm. Experiments will have to show which method yields the best performance.

The experiments we have conducted so far are not entirely conclusive yet, but certainly promising. For dense random graphs OEBA seems to yield crossing number results similar to the traditional barycentric method. However, for certain sparse graphs (Rome graphs) OEBA does reduce the crossings significantly, but not quite as much as the sequential approach.

Parallel Implementation Techniques.

We aim at shared-memory parallelism for multicore processors. Therefore, we anticipate two main parallel implementation techniques. One is OpenMP, a user-friendly standardized runtime system offered by most current compilers. The second one is a task-parallel job queue runtime system on top of the operating system's native threads. While the latter requires the implementor to break down the work into small independent pieces, it is expected to give the programmer more opportunities for low-level optimizations.

References

- 1 K. Sugiyama and S. Tagawa and M. Toda, Methods for Visual Understanding of Hierarchical System Structures, *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

4.4 Circular-Arc Drawings with Right-Angle Crossings

Muhammad Jawaherul Alam, Martin Nöllenburg, Sankar Veeramoni, and Kevin Verbeek

License  Creative Commons BY-NC-ND 3.0 Unported license
© Muhammad Jawaherul Alam, Martin Nöllenburg, Sankar Veeramoni, and Kevin Verbeek

We studied a problem arising from the combination of two recent topics in graph drawing: circular-arc or Lombardi drawings [3] and right-angle crossing (RAC) drawings [2]. More precisely, we looked for drawings, where each edge is drawn as a single circular arc and where the arc tangents at each crossing of two edges form right angles. Such a drawing is called a *circular-arc RAC* drawing or *CRAC* drawing.

During the seminar we obtained some initial results for characterizing graphs that have CRAC drawings. The drawing cannot have four mutually crossing arcs such that all crossings form right angles, i.e., the crossing graph of the drawing cannot contain K_4 as a subgraph. Furthermore it seems that the upper bound of Arikushi et al. [1] that a graph admitting a 1-bend polyline RAC drawing has at most $6.5n - 13$ edges still holds for CRAC drawings.

We also looked at some specific examples of graphs and constructed CRAC drawings for $K_{3,5}$, K_6 , and $K_{4,4}$; however we did not yet succeed in drawing K_7 , which does have a 1-bend RAC drawing. We suspect that neither CRAC nor 1-bend RAC drawings are proper subclasses of each other.

We can use the same construction showing that every graph has a 3-bend RAC drawing [2] to show that every graph has a CRAC drawing where each edge is a differentiable poly-arc consisting of five circular arcs. What happens, if we reduce the complexity of the poly-arcs to three or four arcs? What is the relationship between the classes of k -bend RAC drawings and k -bend smooth CRAC drawings?

References

- 1 K. Arikushi, R. Fulek, B. Keszegh, F. Moric, and C. D. Tóth. Graphs that admit right angle crossing drawings. In D. M. Thilikos, editor, *Proc. 36th International Workshop on Graph Theoretic Concepts in Computer Science*, volume 6410, pages 135–146, 2010. http://dx.doi.org/10.1007/978-3-642-16926-7_14.
- 2 W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. In F. Dehne, M. Gavrilova, J.-R. Sack, and C. D. Tóth, editors, *Proc. 11th International Symposium on Algorithms and Data Structures (WADS'09)*, volume 5664, pages 206–217, 2009. http://dx.doi.org/10.1007/978-3-642-03367-4_19.
- 3 C. A. Duncan, D. Eppstein, M. T. Goodrich, S. G. Kobourov, and M. Nöllenburg. Lombardi drawings of graphs. In U. Brandes and S. Cornelsen, editors, *Proc. 18th International Symposium on Graph Drawing (GD'10)*, volume 6502, pages 195–207, 2011. http://dx.doi.org/10.1007/978-3-642-18469-7_18.

4.5 Edge Bundling

David Auber, Stefan Diehl, Christian Duncan, Cesim Erten, Rudolf Fleischer, Emden Gansner, Michael Kaufmann, Lev Nachmanson, and Michael Wybrow

License  Creative Commons BY-NC-ND 3.0 Unported license

© David Auber, Stefan Diehl, Christian Duncan, Cesim Erten, Rudolf Fleischer, Emden Gansner, Michael Kaufmann, Lev Nachmanson, and Michael Wybrow

The background of the participants covered a wide range of fields information visualization, algorithm engineering, algorithms, graph theory. After having considered the existing practical approaches (e.g., [6, 4, 3, 1, 7, 5, 2]), the group tried to formulate a possibly unified mathematical model for the edge bundling problem. In several iterations, this model has been designed, checked on validity and reformulated. After that, the group including several subgroups considered the reformulation of the most important approaches within the proposed model. This led to interesting and stimulating insights about the differences of those approaches, and how to measure the differences.

A possible survey paper on the topic of this working group has been prospected and sketched.

References

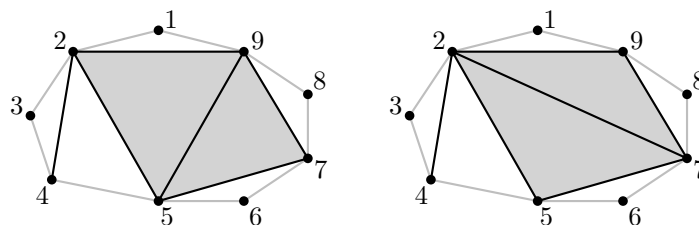
- 1 Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1277–1284, 2008.
- 2 E.R. Gansner, Yifan Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 187–194, march 2011.
- 3 Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006.
- 4 Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990, 2009.
- 5 Antoine Lambert, Romain Bourqui, and David Auber. Winding roads: Routing edges into bundles. *Comput. Graph. Forum*, 29(3):853–862, 2010.
- 6 Doantam Phan, Ling Xiao, Ron B. Yeh, Pat Hanrahan, and Terry Winograd. Flow map layout. In *INFOVIS*, page 29. IEEE Computer Society, 2005.
- 7 Sergey Pupyrev, Lev Nachmanson, and Michael Kaufmann. Improving layered graph layouts with edge bundling. In *Proceedings of the 18th international conference on Graph drawing, GD’10*, pages 329–340, Berlin, Heidelberg, 2011. Springer-Verlag.

4.6 On the Rotation Distance of Rooted Binary Trees

Ulrik Brandes, Rudolf Fleischer, Seok-Hee Hong, Tamara Mchedlidze, Ignaz Rutter, and Alexander Wolff

License © © ⊖ Creative Commons BY-NC-ND 3.0 Unported license
 © Ulrik Brandes, Rudolf Fleischer, Seok-Hee Hong, Tamara Mchedlidze, Ignaz Rutter, and Alexander Wolff

Given a triangulation of a regular n -gon with $n \geq 4$, a new triangulation can be obtained by *flipping* any internal edge. The triangulation resulting from flipping an edge e is obtained by first removing the edge e , and then inserting the diagonal of the resulting quadrilateral that is different from e ; see Fig. 1.



■ **Figure 1** An example of a flip in a triangulation.

For any $n \geq 4$, there is a one-to-one correspondence between $(n - 2)$ -node ordered binary trees and triangulations of the regular (or any other convex) n -gon (with a designated “root” edge). The correspondence is such that rotations between pairs of trees are translated into flips between triangulations. This was observed by Sleator et al. [4]. Since flips are simpler to visualize and understand, we will use the flip-and-triangulation language in the following.

Let \mathcal{P}_n be the regular n -gon, with vertices labeled $1, 2, \dots, n$ in counterclockwise order. The *flip graph* \mathcal{F}_n is the undirected graph whose vertices correspond to the triangulations

of \mathcal{P}_n ; the graph has an edge between two triangulations if they can be obtained from each other by flipping a single edge.

The *flip distance* of two triangulations \mathcal{T}_1 and \mathcal{T}_2 of \mathcal{P}_n is the minimum number of flips needed to transform \mathcal{T}_1 into \mathcal{T}_2 . This is the length of a shortest path from \mathcal{T}_1 to \mathcal{T}_2 in \mathcal{F}_n . Sleator et al. [4] showed that the diameter of \mathcal{F}_n is bounded by $2n - 6$. Using involved arguments from hyperbolic geometry, they proved that this bound is tight for large values of n .

The complexity of the problem of computing the flip distance between two given triangulations of the same convex polygon (with a designated root edge) is unknown; there is neither an efficient algorithm nor an NP-hardness proof. Cleary and St. John [1] showed, however, that the problem is fixed-parameter tractable (FPT) with respect to the parameter flip distance. Their FPT algorithm runs in $O(n + 4^{10k})$ time, where k is the flip distance between the given pair of triangulations of the regular n -gon. There are also three approximation algorithms. The first algorithm, by Li and Zhang [3], depends on the maximum number of diagonals, Δ , incident to a vertex in either of the given triangulations. Their algorithm has an approximation factor of $(2 - 2/(4(\Delta - 1)(\Delta + 6) + 1))$ and runs in cubic time. Note that this approximation factor is always less than 2 and tends to 2 when Δ grows. The second, by the same authors [3], yields a 1.97-approximation if none of the triangulations contains an inner triangle. The third, by Cleary and St. John [2], yields a 2-approximation in linear time.

We developed a simpler and much faster FPT algorithm than that of Cleary and St. John. Our algorithm (see below) runs in $O(n + 4^k/\sqrt{k})$ time.

We want to solve the following problem, which we call k -FLIPDISTANCE. Given a pair $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ of triangulations of \mathcal{P}_n and a positive integer k , determine whether the flip distance of \mathcal{T}_1 and \mathcal{T}_2 is at most k and, if so, compute a shortest \mathcal{T}_1 - \mathcal{T}_2 path in \mathcal{F}_n .

Sleator et al. [4] observed that, on a shortest path connecting \mathcal{T}_1 and \mathcal{T}_2 in \mathcal{F}_n , diagonals common to \mathcal{T}_1 and \mathcal{T}_2 are never flipped. Moreover, if \mathcal{T}_1 admits a flip that increases the number of diagonals common to \mathcal{T}_1 and \mathcal{T}_2 , then there is a shortest \mathcal{T}_1 - \mathcal{T}_2 path in \mathcal{F}_n starting with that flip.

Let c be the number of diagonals common to \mathcal{T}_1 and \mathcal{T}_2 , and let d be the number of diagonals that are in \mathcal{T}_1 but not in \mathcal{T}_2 . Clearly, $c + d = n - 3$. Since the problem k -FLIPDISTANCE allows us to flip at most k diagonals, we have $d \leq k$ for all YES-instances.

Our very simple algorithm is as follows. If $d > k$, we return “no”. Otherwise, we split the problem into $c' \leq c + 1$ independent subproblems of sizes $d_1, \dots, d_{c'}$ with $\sum_{i=1}^{c'} d_i = d$. For subproblem $i = 1, \dots, c'$, we do a breadth-first search in \mathcal{F}_{d_i} starting with the appropriate part of \mathcal{T}_1 . We stop as soon as we have reached the corresponding part of \mathcal{T}_2 . Let ℓ_i be the length of the path between the two parts in \mathcal{F}_{d_i} . We return “no” if $\sum_{i=1}^{c'} \ell_i > k$, “yes” otherwise.

Obviously, the overall running time of our algorithm is bounded by the size of \mathcal{F}_d . It is well-known that the number of vertices of \mathcal{F}_d equals the $(d - 2)$ -th Catalan number C_{d-2} . The number of edges of \mathcal{F}_d is $(d - 3)C_{d-2}/2$ since \mathcal{P}_d has $d - 3$ diagonals. Recall that $C_d = \binom{2d}{d}/(d + 1) \approx 4^d/(d^{3/2}\sqrt{\pi})$. Hence, our algorithm runs in $O(n + 4^d/\sqrt{d})$ time. We summarize our result with the following theorem.

► **Theorem 1.** *The problem k -FLIPDISTANCE can be decided in time $O(n + 4^k/\sqrt{k})$, where n is the size of the polygon. For YES-instances, a corresponding flip sequence can be computed within the same time bound.*

References

- 1 Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109:918–922, 2009.

- 2 Sean Cleary and Katherine St. John. A linear-time approximation for rotation distance. *J. Graph Algorithms Appl.*, 14(2):385–390, 2010.
- 3 Ming Li and Louxin Zhang. Better approximation of diagonal-flip transformation and rotation transformation. In Wen-Lian Hsu and Ming-Yang Kao, editors, *Proc. 4th Annu. Int. Conf. Comput. Combin. (COCOON'98)*, volume 1449, pages 85–94, 1998.
- 4 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations and hyperbolic geometry. *J. Amer. Math. Soc.*, 1(3):647–681, 1988.

Participants

- Deepak Ajwani
University College Cork, IE
- Muhammad Jawaherul Alam
Univ. of Arizona – Tucson, US
- David Auber
Université Bordeaux, FR
- Christian Bachmaier
Universität Passau, DE
- Melanie Badent
Universität Konstanz, DE
- Carla Binucci
University of Perugia, IT
- Franz J. Brandenburg
Universität Passau, DE
- Ulrik Brandes
Universität Konstanz, DE
- Markus Chimani
Universität Jena, DE
- Aparna Das
Univ. of Arizona – Tucson, US
- Camil Demetrescu
University of Rome "La Sapienza", IT
- Walter Didimo
University of Perugia, IT
- Stephan Diehl
Universität Trier, DE
- Christian Duncan
Louisiana Tech University, US
- Philip Effinger
Universität Tübingen, DE
- Cesim Erten
Kadir Has Univ. Istanbul, TR
- Rudolf Fleischer
Fudan University – Shanghai, CN
- Emden R. Gansner
AT&T Res. – Florham Park, US
- Carsten Gutwenger
TU Dortmund, DE
- Seok-Hee Hong
The University of Sydney, AU
- Riko Jacob
TU München, DE
- Jyrki Katajainen
University of Copenhagen, DK
- Michael Kaufmann
Universität Tübingen, DE
- Karsten Klein
TU Dortmund, DE
- Stephen Kobourov
Univ. of Arizona – Tucson, US
- Marcus Krug
KIT – Karlsruhe Institute of Technology, DE
- Robert Krug
Universität Tübingen, DE
- Giuseppe Liotta
University of Perugia, IT
- Tamara Mchedlidze
National TU – Athens, GR
- Kurt Mehlhorn
MPI für Informatik – Saarbrücken, DE
- Henning Meyerhenke
Georgia Inst. of Technology, US
- Petra Mutzel
TU Dortmund, DE
- Lev Nachmanson
Microsoft Corp. – Redmond, US
- Stefan Näher
Universität Trier, DE
- Martin Nöllenburg
KIT – Karlsruhe Institute of Technology, DE
- Maurizio Patrignani
Roma TRE University, IT
- Md. Saidur Rahman
Bangladesh University of Eng.& Technology, BD
- Ignaz Rutter
KIT – Karlsruhe Institute of Technology, DE
- Georg Sander
IBM – Frankfurt, DE
- Miro Spönemann
Universität Kiel, DE
- Joachim Spoerhase
Universität Würzburg, DE
- Matt Stallmann
North Carolina State Univ., US
- Sankar Veeramoni
Univ. of Arizona – Tucson, US
- Kevin Verbeek
TU Eindhoven, NL
- Stephen Wismath
University of Lethbridge, CA
- Alexander Wolff
Universität Würzburg, DE
- Michael Wybrow
Monash Univ. – Clayton, AU
- Katharina A. Zweig
Universität Heidelberg, DE

