The Open Graph Drawing Framework

Markus Chimani[1], Carsten Gutwenger[1], Michael Jünger[2], Karsten Klein[1],
Petra Mutzel[1], and Michael Schulz[2]

[1] University of Dortmund, Germany
{markus.chimani,carsten.gutwenger,karsten.klein,petra.mutzel}@cs.uni-dortmund.de
[2] University of Cologne, Germany
{mjuenger,schulz}@informatik.uni-koeln.de

1 Introduction

Many aspects of graph drawing research are motivated from practice, and practical evaluation of graph drawing algorithms is essential. However, graph drawing has now grown for several decades and a huge amount of algorithms for various drawing styles and applications has been proposed. Many sophisticated algorithms build upon complex data structures and other algorithms, thus making new implementations from scratch cumbersome and time-consuming. Obviously, graph drawing libraries can ease the implementation of new algorithms a lot. The LEDA-based C++-library AGD was very popular in the past, since it covers a wide range of graph drawing algorithms. However, the lack of publicly available source-code restricted the portability and extendability, not to mention the understanding of the particular implementations. Other currently available graph drawing libraries suffer from the same problems, or are even only commercially available or focus only on special graph layout methods.

Our goals for the *Open Graph Drawing Framework* (*OGDF*) were to transfer essential design concepts of AGD and to overcome its main deficiencies for use in academic research:

- A wide range of graph drawing algorithms that allow to reuse and replace particular algorithm phases by using a dedicated module mechanism.
- Sophisticated data structures that are commonly used in graph drawing, equipped with rich public interfaces.
- Self-contained code that does not require any additional libraries (except for some optional branch-and-cut algorithms).
- Portable C++-code that supports the most important compilers for Linux, MacOS, and Windows operating systems.
- Open source code available under the terms of the GPL.

2 Algorithms and Data Structures

A key advantage of OGDF is that it provides a wide range of adaptable layout algorithms like its customizable implementation of the planarization method. The crossing minimization module can also be used independently of the layout algorithm and provides unique features like optimal edge insertion with variable embedding, non-planar core graph reduction, and exact crossing minimization based on branch-and-cut.

We extended these well known and practically successful methods to simultaneous drawing and hypergraphs. Recently, we studied the relationship of the minor crossing number and the hypergraph crossing number (in the edge-standard) by considering a generalization of the minor crossing number. For the former problem, only some theoretical results had been published so far, whereas the latter problem has important applications in practice (e.g., layout of electrical wiring schemes). The key idea for generalizing the planarization approach to hypergraphs was the generalization of the edge insertion algorithms. We could show that this is possible with the same time complexity as for regular graphs, even in the variable embedding case. OGDF contains implementations of these yet unpublished algorithms.

Most data structures and algorithms used in the implementation of the planarization approach are available as building blocks for developing new algorithms. These include planarity testing and embedding algorithms, algorithms for computing planar subgraphs, and algorithms for optimizing planar embeddings (e.g., minimum depth, maximum external face). Moreover, a recently proposed method for the extraction of a large number of Kuratowski subdivisions (which is used by the optimal crossing minimization algorithm) is provided.

The main components for optimizing over the set of all planar embeddings are the data structures BC- and SPQR-trees. OGDF contains not only efficient implementations of the static variants (such as the static SPQR-trees in AGD), but also efficient dynamic implementations. To our knowledge, this is the only implementation of dynamic SPQR-trees.

The implementation of the Sugiyama approach is also highly customizable and implements various algorithms (including optimal node ranking and coordinate assignment). Further algorithms deal with clustered graphs, planar drawing, planar augmentation, and upward planarity, as well as force-directed layout for large graphs with the fast multipole multilevel method (FM^3).

At the moment, a new graph editor based on OGDF is in preparation, which does not only provide easy access to OGDF's graph layout algorithms, but also introduces powerful features like graph perspectives and a plugin mechanism that allows users to easily extend the editor with new functionality. More information about the OGDF project can be found at:

`http://ls11-www.cs.uni-dortmund.de/ogdf`

Acknowledgements. Many further people contributed to the OGDF. We wish to thank Sebastian Leipert, Christoph Buchheim, Stefan Hachul, Andrea Wagner, Joachim Kupke, Matthias Elf, Dino Ahr, Hoi-Ming Wong, and our students Martin Gronemann, Bernd Zey, Mathias Jansen, Thorsten Kerkhof, Jan Papenfuß, and Jens Schmidt.