A Library of Algorithms for Graph Drawing*

Carsten Gutwenger[1], Michael Jünger[2], Gunnar W. Klau[3], Sebastian Leipert[1], Petra Mutzel[3], and René Weiskircher[3]

[1] Stiftung casesar, Bonn, Germany
[2] Universität zu Köln, Germany
[3] Technische Universität Wien, Austria

1 Short Description

The AGD library provides algorithms, data structures, and tools to create geometric representations of graphs and aims at bridging the gap between theory and practice in the area of graph drawing. It consists of C++ classes and is built on top of the library of efficient data types and algorithms LEDA; an optional add-on to AGD requires ABACUS, a framework for the implementation of branch-and-cut algorithms, and contains implementations of exact algorithms for many *NP*-hard optimization problems in algorithmic graph drawing.

The fully documented library is freely available for non-commercial use at `http://www.ads.tuwien.ac.at/AGD`. The site also contains an online manual, links to AGD related papers, and contact information.

2 Layout Algorithms and Layout Features

The library contains a large number of state-of-the-art drawing algorithms for many of which implementations can only be found in AGD. Figure 1 shows UML-diagrams of three major components of the library: (a) planar drawing algorithms, (b) hierarchical drawing algorithms, and (c) the planarization method. Among the highlights in the latest version are implementations of the Kandinsky- and the Giotto-algorithm, new heuristics and exact algorithms for two-dimensional compaction, and new strategies for crossing minimization based on a linear-time implementation of SPQR-trees.

* Partially supported by DFG-grants Ju204/7-3, Mu1129/3-1, and Na 303/1-3. In addition to the authors, the following persons have contributed to AGD: D. Alberts, D. Ambras, R. Brockenauer, C. Buchheim, M. Elf, S. Fialko, K. Klein, G. Koch, T. Lange, D. Lüttke-Hüttmann, S. Näher, T. Ziegler

P. Mutzel, M. Jünger, and S. Leipert (Eds.): GD 2001, LNCS 2265, pp. 473–474, 2002.

3 Architecture

Most successful approaches for drawing graphs consist of several phases. The open and modular design of the library, realized by a strict compliance to the object-oriented programming paradigm, facilitates the experimentation with different approaches to the various subtasks. It is easy to replace or extend existing modules by plugging in an alternative method or providing a derived solution.

3.1 Programming Language and Operating Systems

The library is written in C++ and can be downloaded for Linux, Solaris, and Windows platforms and different compilers.

4 Interfaces

AGD implements a generic layout interface that is independent of a fixed drawing component. This partition makes it easy to integrate the library's functionality within application programs. However, several interface implementations are already included in the library; most notably, two demo programs that use the graph editor in LEDA provide instant access to the functionality of the base part and the optional add-on of AGD. These interfaces can be easily extended and adapted. Furthermore, due to the file- and socket-based AGD server interface, the usage of AGD is not restricted to C++ applications.

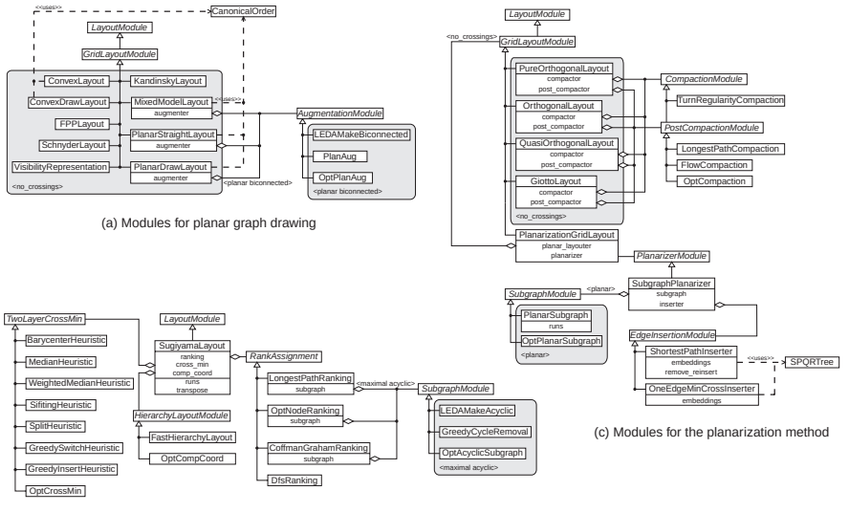

Fig. 1. UML-diagrams of selected AGD modules