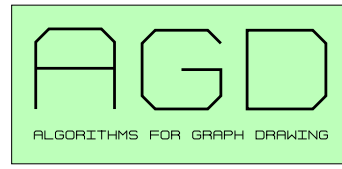
A Library of Algorithms for Graph Drawing*

Petra Mutzel[1], Carsten Gutwenger[1], Ralf Brockenauer[1], Sergej Fialko[1], Gunnar Klau[1], Michael Krüger[1], Thomas Ziegler[1], Stefan Näher[2], David Alberts[2], Dirk Ambras[2], Gunter Koch[2], Michael Jünger[3], Christoph Buchheim[3], and Sebastian Leipert[3]

[1] Max-Planck-Institut für Informatik Saarbrücken
[2] Universität Halle
[3] Universität zu Köln



This poster presents AGD, a library of **A**lgorithms for **G**raph **D**rawing. The library offers a broad range of existing algorithms for two-dimensional graph drawing and tools for implementing new algorithms. The algorithms include planar graph drawing methods such as straight-line, polyline, orthogonal, visibility, and tree drawing methods. In order to make these algorithms useful for general graphs, we provide various planarization methods ranging from heuristic to optimal algorithms. Data structures, like, e.g., PQ-trees, have been especially tailored for applications in graph drawing. Users can engineer their own hybrid methods by combining the provided tools like planarization, 2-layer crossing minimization, and various shelling orders (see Figure 1).

Most graph drawing algorithms place a set of restrictions on the input graph like planarity or biconnectivity. We provide a mechanism for declaring such a precondition for a particular algorithm and checking it for potential input graphs. A drawing model can be characterized by a set of properties of the drawing. We call these properties the postcondition of the algorithm. There is support for maintaining and retrieving the postcondition of an algorithm.

AGD is written in the programming language `C++` and uses the LEDA platform for combinatorial and geometric computing. The design of the library is based on the object oriented features of `C++`. Graph drawing algorithms as well as combinatorial algorithms are modeled as classes. The implementations of exact optimization algorithms for NP-hard problems use the branch-and-cut system ABACUS. The algorithms are implemented independently of a visualization or graphics system by using a generic layout interface. Layout interfaces are currently available for the LEDA data type GraphWin and for the graph editor Graphlet. The open design makes AGD very easy to use and to extend. AGD is publically available via `http://www.mpi-sb.mpg.de/AGD/`.

* This project was partially supported by DFG–grants Ju204/7–3, Mu1129/3–1, Na 303/1–3, Forschungsschwerpunkt "Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen"

S.H. Whitesides (Ed.): GD'98, LNCS 1547, pp. 456–457, 1998.

A paper on the design of AGD has appeared in: G.F. Italiano, S. Orlando (eds.), *Proc. of the Workshop on Algorithm Engineering (WAE '97)*, Venice, (`http://www.dsi.unive.it/~wae97/proceedings/`).

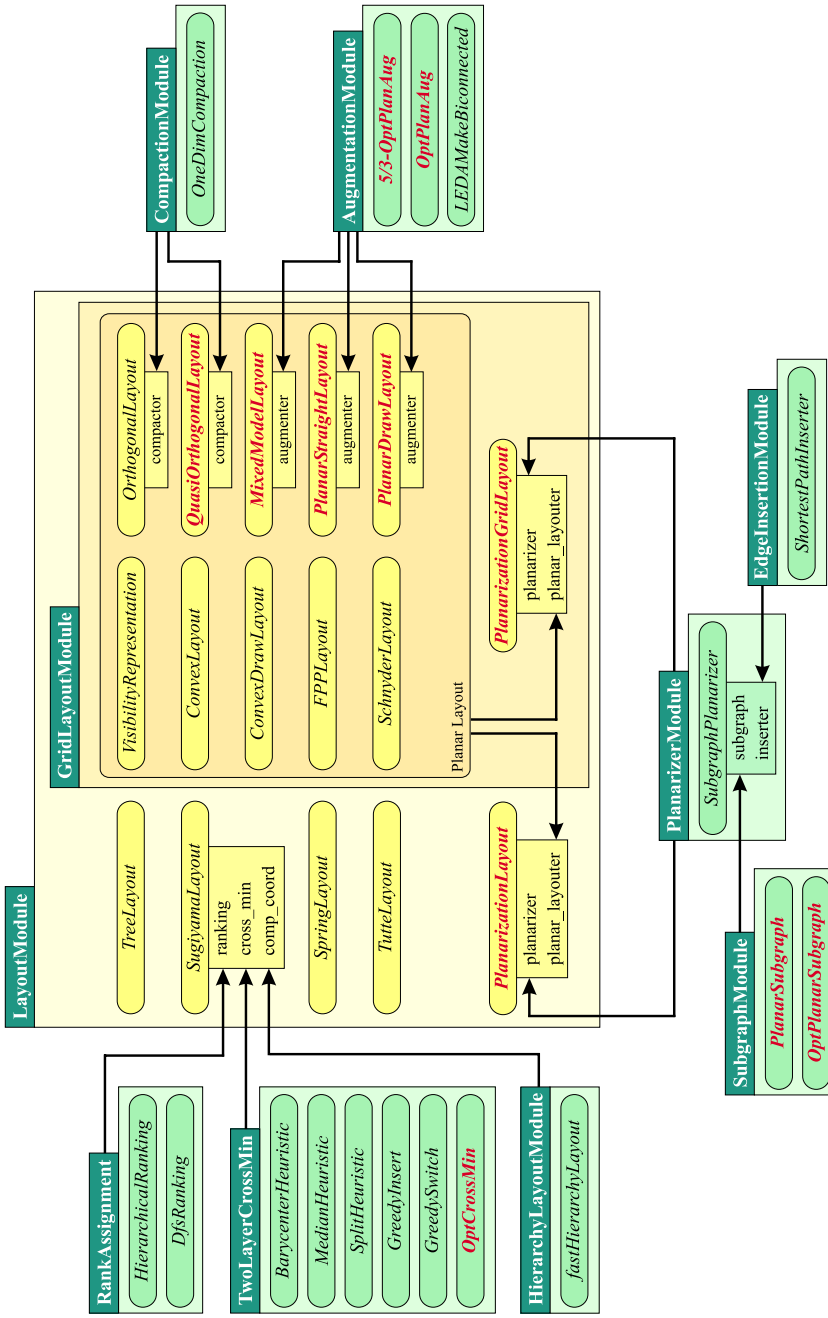


Fig. 1. Overview of the AGD library; grey labels indicate new or improved algorithms