Chapter 10
Macro Cell Placement: Based on a Force Directed Flow

Meththa Samaranayake and Helen Ji

Abstract Macro cells are used more and more in current designs as they provide the benefit of reusability directly resulting in the decrease in design time and cost. However, there lies a gap in the EDA industry for macro-cell placement tools. This chapter looks at the implementation of a force-directed macro-cell placement tool that is developed to target the gap in industry.

Keywords Macro-cell · placement · force directed · EDA · graph drawing

10.1 Introduction

The past few years have seen an exponential rise in the growth rate of the semiconductor industry. The increase in usage and demand of electronic devices among consumers has resulted in the need to provide better and faster design methods. The designers are pushed to their limits in meeting these demands whilst juggling the constraints of power and performance of ever shrinking circuits. To help designers meet their targets, EDA (Electronic Design Automation) tools are used to help fully or partially automate the design processes. One of such important backend processes is the placement component.

The placement problem simply is the problem of finding the ideal locations for each cell in a circuit achieving as many or all of the placement objectives. The two main objectives that every placement tool has to achieve for today's fixed die design are,

- Overlap free layout
- Fit in the given placement area

M. Samaranayake (✉) and H. Ji
Department of Engineering and Technology, Manchester Metropolitan University, Chester Street, Manchester M1 5GD UK
e-mail: meththa.t.samaranayake@stu.mmu.ac.uk; h.ji@mmu.ac.uk

Other objectives may include minimization of wirelength, congestion, area, run time etc. The optimal solution will be one that satisfies all of the given criteria. Achieving such a placement solution is far from possible and even the simplest of cell placement problems are defined to be NP-hard. The consequence of falling short of a good placement could result in an unroutable design or a slower and/or larger chip. This will cost time and money to either correct the placement at a later stage or re-do the design.

In the past, designs mainly carried standard cells but as the complexity and components increased, Macros were utilized to reduce the complexity of circuits. Macros can mainly be seen as black boxes that are designed to carry out specific tasks such as implementation of a logic function (e.g. an IP block). It can also be on-chip memories that are common in SoC (System on Chip) designs. Increased use of Macro cells help designer's reuse of their designs which in turn helps reduce design time and cost.

Previous work [1, 2] has looked at the possibility of using graph-drawing algorithms as the basis of a Macro-cell placement tool. In this work, the authors have extended the capabilities of the algorithms in order to achieve better placements of cells. The two algorithms used in the work are those authored by Kamada and Kawai [3] and by Fruchterman and Reingold [4] (referred to as KK and FR respectively). They were chosen for their ability to handle undirected graphs, their simplicity in implementation, their speed as well as the criteria they follow to produce aesthetically pleasing graphs. In many cases, these criteria are shared by good placements.

10.2 Placement Tools

There are many standard cell placement tools currently available both academically and commercially within the EDA community. Several of them are capable of mixed-mode cell placement i.e. designs that contain both standard cells and modules, but there are only a few placement tools specifically for macro cells. This is because standard cells used to govern most of the circuit designs up till recent times. Recent changes have seen designs to contain Macro-based designs such as memory blocks and IP blocks (Intellectual Property) and furthermore, the hierarchical design methodology intended to tackle design complexity has resulted in macro-dominated designs at the top level. Even though mixed mode placement tools can handle macros, for designs that contain a majority of macros these tools may not place the cells in the best interest of the macros giving the need for a dedicated macro cell placer.

Some leading edge mixed-mode placement tools identified are Capo [5], Dragon [6], FastPlace [7] and APlace [8]. The Capo tool is based on a combination of simulated annealing and recursive bisection techniques and handles macro cells mainly by the help of the floorplanner Parquet [9]. Capo placement tool has a secondary method of placing macro cells where it shreds them to smaller sub-cells. These sub-cells are connected by two pin nets ensuring that they are placed close to one another. The design is then solved as a standard cell placement problem. FastPlace and APlace tools are based on analytical techniques and incorporates macro-cell placement within its normal placement flow. In FastPlace, the macro-cells are given priority during legalization stage where overlaps are resolved between macros before standard cells. Dragon is a hybrid placement tool that combines the use of simulated annealing with min-cut partitioning. To accommodate macro cells, it modifies the min-cut algorithm so that the partitions can be of different sizes. All these placement tools were designed for standard cells and as a result, with the exception of the Capo placer, these tools do not consider factors such as macro pin locations, cell orientation, soft macro cells and fixed cell support etc. Due to the capabilities of the Parquet floorplanner, Capo can handle all factors affecting macro cells aside from pin handling.

A Java based macro-cell placer [10] based on a force directed placement algorithm has been identified to be different from traditional force directed algorithms. In this work, the cell shapes and sizes have been considered when developing the force equation. A limitation of this tool is that it does not handle placement on a fixed placement area and instead treats the chip as a soft cell with a variable size and aspect ratio. The pads of the chip are also not fixed; therefore, the positions are found with the use of the force directed algorithm at a later stage.

A macro-cell placement method based on net clustering and force directed method is proposed in literature [11]. This approach is unique such that, it treats the nets as the placement components. In the graphs they draw, the nodes represent the nets whilst an edge only exists for the nets that share one or more cells. The forces on the nets determine the initial locations for the cells. Pin locations are determined last, suggesting that this placement tool is mainly focused on soft cell macros. This work reiterates the importance of the pin locations and cell orientation in macro cell placement. Another limitation seen is that the tool only handles connected graphs, again limiting the type of designs that can be processed.

Looking at both macro-cell placers identified above, a common disadvantage recognized is that both tools are not standardized – inputs are not of industry recognized LEF/DEF format [12] but formats limited to the tool. This has limited the tools from reading in standard designs currently available, therefore disabling measuring their quality of placement. The same is true for outputs where they are not given out in any standard format so that the placement can be processed by a routing tool.

10.3 Standard Cells Versus Macro-cells

Macro-cell placement is not as straightforward as standard cell placement. In standard cell placement, the cells are of uniform height and are restricted to rows in which they must sit in. These restrictions allow the placement tools to be more precise in choosing locations for the standard cells and to allocate routing resources. Macro cells on the other hand do not have such restrictions. They can be of any height, width and shape (L, T and U shapes though the most common is rectangular) and are not restricted to a specific location of the placement area. As a result, choosing a good placement for macro-cells can be much harder as the permutations of locations they can be placed-at are unbounded. Similarly, the different shapes of the cells can bring in unwanted limitations on finding placements. This can bring negative results such as more expensive computations and longer runtimes.

It has been found that the size of macro-cells can sometimes be a considerable amount of the total area; sometimes even up to half of the placement area. This can have a significant impact on the finalizing the positions of cells with respect to others. Therefore, it is necessary to give due consideration to the magnitude of cells and the impact they can have on other cells.

As well as cell position, cell size has a significant impact on the position of pins. Unlike in standard cell placement, pin locations can have a significant impact on wirelength, routability and congestion of the chip. To overcome this, the placement tool will need to handle extra features such as cell mirroring and cell rotation to consider the best possible cell orientation in order to minimize the above-mentioned costs and to place the pins in the best locations possible.

Fixed cells are also an important factor that needs to be looked at during cell placement. There are times when one or more components of the design need to be placed in a fixed position within the placement area. For macro-cells, these fixed cells will create a blockage on the area on which cells are to be placed and will need to be given due consideration during the placement process.

It is seen that there are important differences between standard cell designs and macro-cell designs and these differences need to be given appropriate priority during placement. It further reaffirms the need for macro-cell placement tools that are separate from standard cell placement tools. Not doing so will result in poor placements and increased design costs in terms of wirelength, congestion and routing resources etc. that is detrimental for both designers and manufacturers alike.

10.4 Force Directed Graph Drawing Algorithms

Graph drawing algorithms are mainly concerned about nodes lacking any size or shape, whereas for cell placement cell sizes need to be given due consideration. A recent published work introduces methods of successfully modifying graph-drawing algorithms to incorporate dimensions to nodes [13]. This work is mainly aimed towards general graph drawing algorithms and the criteria they use for graph drawing include,

- Vertices are not to overlap
- Edges are not to cross vertices

For this work, the first criterion directly applies, as the objective of the placement tool is to produce a non-overlapping placement. The second criterion also applies as it tends to place directly connected cells together, but it could be too conservative if routing is allowed to be over-the-cell. One of the limitations of this work [13] is that the node orientation is fixed and cannot be mirrored or rotated.

Force directed graph-drawing algorithms generally tend to be analogous to the classic problem of Hookes law for a spring system. Most of the current force directed algorithms follow the footsteps of Eades' spring embedded algorithm [14]. Hooke's law simply stated that the force exerted by an extended spring is proportional to the length of the spring. Eades modeled the graph as a system of rings in place of the nodes and springs for edges. His formula for the forces exerted by the springs differed Hooke's law by the former taking both attraction and repulsion forces in to consideration. The aim of all the force directed algorithms is to find zero-force locations for all nodes – i.e. state of equilibrium for that system.

A comparison [15] of several force-directed algorithms has been carried out where KK and FR algorithms were the two main contenders. It was identified that KK is successful in achieving high computation speed, minimizing the computation time. Even though FR is quick in giving aesthetically pleasing layouts, it is said to suffer from long run times when the number of nodes/edges exceeds 60. One cannot declare a certain algorithm to be the best where each has its pros and cons and how relevant each algorithm is depends on the application [15].

KK Algorithm [3] is concerned about general undirected, connected graphs. It has the ability to handle weighted graphs such that edges with higher weighting are longer than those with a lower weighting. One advantage in this algorithm is that it introduces a "graph theoretic distance" which defines a minimum edge length in order to minimize node overlaps. The main objective of the algorithm is to find a balanced formulation of the spring forces within the system. The graph drawing criteria followed by KK [3] are,

- Reduce number of edge crossings.
- Distribute the vertices and edges uniformly.

Comparing these criteria with those of the macro-cell placement tool, it can be seen that both are related to the 'good placement criteria'. Reducing number of edge crossings results in directly connected cells being placed close to each other. The second criterion allows the nodes to be evenly distributed within the placement area as well as show any symmetry within the layout. This not only is an advantage for graph drawing where the aesthetics are improved, but for cell placement, by illustrating the cell connections in an uncomplicated manner. It is worth pointing out that symmetry is a very important heuristic for placement. While most of placement tools have difficulty in incorporating it into their algorithms, the KK algorithm handles it neatly.

The main objectives of the FR algorithm are to achieve a visually pleasing graph with increased speed and simplicity. Following Eades work, the FR algorithm also makes use of both attraction and repulsion forces, but takes it one-step further by defining that the attraction forces only to be calculated for neighboring nodes whilst repulsion forces are calculated for all nodes within the graph.

Looking at the criteria followed by FR [4] when drawing graphs, it is seen that two main points are considered.

- Vertices connected by an edge should be drawn near each other.
- Vertices should not be drawn too close to each other.

The first criteria does apply for the cell placement tool as the cells connected to one another will need to be close to each other in order to minimize wirelength. This can be further enhanced by edge weights to ensure that cells connected to edges with higher weights are as close as possible. Unfortunately, the current implementation of the FR algorithm does not contain support for edge weights. The second criterion is set quite vaguely and according to literature [4] it depends on the number of nodes and the placement area. Literally, this should mean that the nodes do not overlap each other, which is directly applicable to the objectives of the placement tool.

FR algorithm uses a method similar to simulated annealing to control the 'cooling schedule' of the algorithm, which controls the number of sweeps it goes through in optimizing the layouts. This can be both advantageous and disadvantageous. It is advantageous such that it helps limit the displacement prohibiting the algorithm to be trapped in local minima. It is disadvantageous such that the number of sweeps is kept at a constant so that the algorithm does no check on the quality of placement before ending the sequence.

The main difference between the FR and KK algorithm is that the FR algorithm can handle disconnected graphs. Even though this is not an absolute requirement compared to the objectives of the placement tool, it does give an advantage as to the type of designs the algorithm will be able to handle. Authors of KK [3] points out that even though KK algorithm does not support disconnected graphs, it can be easily extended to do so without a significant delay in time as follows.

> Partition the graph to its connected components giving each component a region of area proportional to its size, with each component laid out independently.

FR algorithm puts this theory into practice in its technique in handling disconnected graphs. Authors of FR names this technique as the *"grid variant option"* where the placement area is divided into a grid and nodes are given locations within the grid. Changes are made to the calculation of the repulsion forces; for each node, the repulsion forces are calculated from the nodes within the current grid as well as those in neighboring grids, unlike the basic algorithm which calculated repulsion forces for all nodes.

Another difference between the two algorithms is that KK does not specify a clear placement area for the graph whereas FR implements support for a customizable placement area. Whilst for graph drawing this may not be very important, it does carry greater significance in cell placement where the cells are expected to be placed within the given placement area in order for the placement to be legal. It is believed that limitation on placing components within the placement area can be imposed upon in later stages when being used in the placement tool. Table 10.1 summarizes the basic capabilities of the two algorithms.

10.5 Implementation Details

Initial work carried out [1] has proved that both KK and FR are good candidates as a basis for a macro-cell placement tool. The basic algorithms of both KK and FR were while sufficient as graph drawing algorithms, lacked the necessary functionalities to

Table 10.1 Comparison of Kamada-Kawai and Fruchterman-Reingold algorithms

KK algorithm	FR algorithm
Undirected graphs	Undirected graphs
Cannot handle disconnected graphs	Can handle disconnected graphs
Objective is to evenly distribute nodes and edges	Objective is to place nodes close to each other
Does not respect boundary conditions	Nodes are placed within the given boundary
Supports the use of edge weights	Current implementation cannot handle edge weights

be used as a module placement algorithm. Further modifications were introduced to the algorithms in-order to function better. The basic implementations of the two algorithms were taken from the peer reviewed Boost [16] library.

10.5.1 Non-zero Size Vertices Implementation

Traditional force directed algorithms tend to treat the cells as points that do not posses any size or shape. The edges do not connect to any pins but to the nodes that represent the cells. This method may be acceptable for standard cell design [10] but in Macro cell placement it can cause inaccuracies of positions, wirelength, area, congestion etc. due to the cell dimensions. Recent literature [13, 17, 18] has been found to carry out work regarding the implementation of different size nodes for graph drawing.

The simplest method of representing a cell is to consider the node to be circular [2]. However, previous work showed that for macro-cell placement, circular nodes could introduce inaccuracies of the actual dimensions and shapes of the cells. Cells with high aspect ratios can overlap one another requiring further work towards legalizing the placement. To tackle this, the elliptic spring method [13] was applied to FR algorithm. The attraction and repulsion forces for the cells are calculated such that assuming the nodes are elliptical in shape. The values of the forces are selected based on the condition if the source and target nodes are overlapping or not. All the modules are assumed rectangular shaped and the width and height of each cell is used to calculate the radii of the elliptical vertex that will represent the module.

For KK, this method could not be applied. During the flow of the algorithm, KK constantly calculates the distance between the boundaries of the connecting nodes. Whilst considering the nodes as circular, this was easily achievable. However, if the nodes were to be considered elliptical, this would cause much complexity in the algorithm increasing the runtime significantly; therefore this was not applied to KK.

10.5.2 Fixed Node Support

Another feature that was lacking within the graph drawing algorithms was support to handle fixed nodes. This is especially useful when designers may specify locations for some of the cells to be fixed or for the placement of the IO (Input Output) pads, which communicate with the external world. In force directed algorithms, since there are attraction and repulsion forces that affects all cells, it was needed to ensure that the forces emitted by the fixed cells were still being taken into account whilst the forces felt upon the fixed cells do not cause the fixed cell to displace as is illustrated in Fig. 10.3.

The algorithm of FR was altered so that the fixed nodes are treated equally as movable cells during force calculation. During displacement calculations, fixed nodes are ignored and for added measure ignored during positional updating of the cells as well. This has shown to be a more accurate method of force calculation for the algorithm when containing fixed cells.

The KK algorithm was modified to filter out the fixed nodes during the energy minimization calculations. This was accomplished in a manner that, whilst minimizing the energy function for the movable cells, the affect made from fixed nodes are still felt. Again, this has been proven successful in implementation.

10.5.3 Input/Output Format

Not all placement tools follow a single format for input and output. This hinders benchmarking and comparison of placement tools. It is with this in mind that it was thought best to use the industry standard formats; the Cadence LEF/DEF file format. The LEF/DEF format is written in ASCII format and can be easily understood. The DEF file contains all the information relevant to the design, constraints, layout and netlist information whilst the LEF file contains the library information of all the cells and modules within the design as well as information regarding layers, vias and core area information. In order to read in the necessary information for the placement tool, a parser was developed. The parser reads data in from the two files, extracts the necessary data and saves it into a text file, which can then be read in by the placement algorithms. It is hoped that in the future, this will be integrated within the placement algorithm itself so that the data input will be a one-step process.

Once the algorithms have generated a placement, it will output the summary in text format and plot the placement in order to inspect the results achieved. In future, it is hoped to output the data into a DEF file such that the final placements then can be routed. Routing congestion is another important quality measurement of the placement.

10.6 Experimentation and Results

With the implementation of different features to the algorithms, they were simulated under different conditions to identify their strengths and weaknesses. To start, the two algorithms were subjected to a selection of graphs, some with known golden topologies. Cells of different dimensions were used to observe the impact the changes described in this chapter. The simulations were run on an Intel Pentium IV PC running at 3.2 GHz and with 2 GB of RAM.

Table 10.2 compares the results obtained through this exercise. The runtime and HPWL (Half-perimeter wirelength) are the cost factors looked at during the experimentation to evaluate the performance of the algorithms. The first two columns show previous results of KK and FR whilst using circular nodes. The results shown for the FR are those obtained for the grid option, which allows the use of disconnected graphs and with elliptical nodes. As it was not possible to use elliptical nodes for KK, the placement of KK was taken as an initial placement and further enhanced by FR signified as the KK_FR flow. Figure 10.1 gives the placement of FR with circular nodes, FR with elliptical nodes and that of KK_FR for graph5 and graph6.

It is noted that the runtime of FR algorithm has increased due to the increased complexity of the algorithm by the changeover from circular nodes to elliptical nodes. Wirelength has seen in increase as well, however, this can be said to be due to the reduction in overlap where the elliptical nodes has allowed the cells to be better placed. This is further apparent in Fig. 10.1a where we see in the previous

Table 10.2 Comparison of runtime (ms) and wirelength (μm) results from previous work (marked as old) and from current work

	KK (old)		FR (old)		FR		KK_FR	
	Time	WL	Time	WL	Time	WL	Time	WL
G1	0	360	15	369	15	416	0	415
G2	15	390	15	364	15	412	0	366
G3	15	509	16	522	15	648	15	588
G4	46	1,281	15	1,163	2,187	1,489	1,296	1,461
G5	78	1,072	31	1,011	3,812	1,060	2,750	1,051
graph1	0	429	0	446	1,312	576	484	631
graph2	15	231	0	235	1,125	318	343	257
graph3	15	619	15	624	93	843	15	820
graph4	15	23	15	32	15	27	0	45
graph5	15	321	0	366	1,390	371	15	361
graph6	16	887	15	819	2,171	919	1,281	982
graph7	343	975	140	751	125	877	156	723
graph8	46	763	46	786	2,625	1,042	31	1,080
graph9	296	980	78	999	93	1,276	171	1,082
graph10	31	157	15	147	15	127	578	131
graph11	219	480	93	459	109	461	124	461

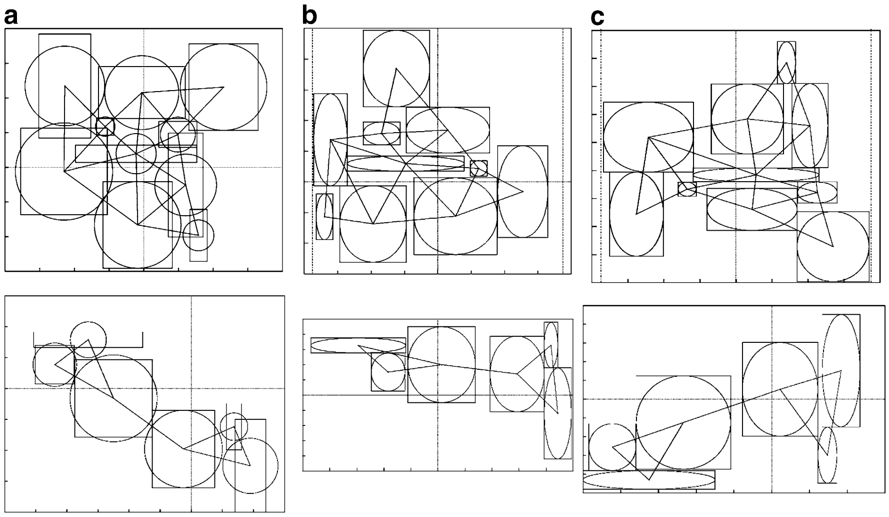

Fig. 10.1 Placement results of graph5 (*above*) and graph6 (*below*) as achieved by (**a**) FR_old, (**b**) FR with elliptical nodes, and (**c**) KK_FR placement flow

work FR has overlaps between the actual cells. However in Fig. 10.1b the reduction in overlaps is visible due to the fact that long rectangles are better represented by the elliptical nodes.

Looking at the results achieved by KK_FR, it is seen that for the same designs, runtime is less than when FR is applied on its own. It is believed that the initial placement given by KK has helped FR to reduce the complexity of the placement problem and therefore achieves better placements in a lower amount of time.

10.7 Future Work and Conclusion

In conclusion it can be said that the use of elliptical nodes has helped the FR graph drawing algorithm find more accurate placements than when using circular nodes. Even though it was not possible to apply elliptical nodes to KK algorithm, initial work carried out here has suggested that the algorithm will be successful in generating initial layouts which can then be improved by a secondary algorithm. Future work will focus on optimizing those added features and in developing the idea of having multiple algorithms within the placement flow to reduce the runtime and produce higher quality placements. In addition, further work will include establishing techniques to use pin locations to optimize wirelength by means of rotating and/or mirroring of cells. The experiments carried out so far have given positive results in achieving good layouts even with the presence of cell dimensions.

References

1. Samaranayake, M., Ji, H., Ainscough, J.: A force directed macro cell placement tool. The 2009 International Conference of Electrical and Electronics Engineering, London, 1–3 July 2009
2. Samaranayake, M., Ji, H., Ainscough, J.: Force directed graph drawing algorithms for Macro cell placement. World Congress on Engineering, London, 2–4 July 2008
3. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inform. Process. Lett. **31**:15 (1989)
4. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw.-Pract. Exp. **21**:1129–1164 (November 1991)
5. Adya, S., Chaturvedi, S., Roy, J., Papa, D.A., Markov, I.L.: Unification of partitioning, placement and floorplanning. International Conference of Computer Aided Design, pp. 550–557, November 2004
6. Cong, J., Kong, T., Shinnerl, J.R., Xie, M., Yuan, X.: Large-scale circuit placement: gap and promise. IEEE/ACM International Conference on Computer-Aided Design, pp. 883–890, 2003
7. Viswanathan, N., Pan, M., Chu, C.: FastPlace 3.0: a fast multilevel quadratic placement algorithm with placement congestion control. Asia and South Pacific Design Automation Conference, pp. 135–140, 23–26, January 2007
8. Kahng, A.B., Reda, S., Wang, Q.: APlace: A general analytic placement framework. International Symposium of Physical Design, pp. 233–235, San Francisco, CA, April 2005
9. Adya, S., Markov, I.L.: Fixed-outline floorplanning: enabling hierarchical design. IEEE Trans.VLSI Syst. **11**:1120–1135 (December 2003)
10. Mo, F., Tabbara, A., Brayton, R.K.: A force-directed macro-cell placer. International Conference on Computer-Aided Design, pp. 177–180, San Jose, CA, November 2000
11. Alupoaei, S., Katkoori, S.: Net-based force-directed macro cell placement for wirelength optimization. IEEE Trans. VLSI Syst. **10**:824–835 (December 2002)
12. Sanrarini, M.: Open source website offers LEF/DEF formats. EE Times (2000)
13. Harel, D., Koren, Y.: Drawing graphs with non-uniform vertices. Proceedings of Working Conference on Advanced Visual Interfaces, pp. 157–166 (2002)
14. Eades, P.: A heuristic for graph drawing. Congressus Numerantium, pp. 149–160 (1984)
15. Brandenburg, F.J., Himsholt, M., Rohrer, C.: An experimental comparison of force-directed and randomized graph drawing algorithms. Symposium on Graph Drawing, pp. 76–87, 20–22 September 1995
16. Boost. http://www.boost.org/. Accessed September 2007
17. Wang, X., Miyamoto, I.: Generating customized layouts. In: Brandenburg, F.J. (ed.) Graph Drawing, vol. 1027, pp. 504–515. Springer, Berlin (1996)
18. Gansner, E., North, S.: Improved force-directed layouts. In: Whitesides, S.H. (ed.) Graph Drawing, vol. 1547, pp. 364–373. Springer, Berlin (1998)