

INVISIBLE EDGE AND SOFT TIED IN COMPACTION STRATEGY

Pongstorn Maidee and Somsak Choomchuay
Department of Electronics, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang (KMITL)
Chalongkrung Road, Ladkrabang, Bangkok 10520, Thailand.

Abstract

In conventional one-dimensional compaction, the objects may be tied to reduce the complexity. The compactness of the resulting layout may, however, be reduced when the tied appear in the critical path. In this paper, this problem is solved by the *soft tied* which can be broken to relocate the objects position. The compactness is also limited by the extra edge used to avoid diagonal constraint violation. As well as, the wire ended without any rigid object at its end cause an additional vertex in the constraint graph. The number of this wire gradually increase after jog insertion performed in each compaction step, thus the complexity will be increased. The *invisible edge* is proposed to solve these problem and make any necessary jog can be inserted automatically. The invisible edge and soft tied technique allows the one-dimensional compaction to gain the compactivity close to that obtain by the two-dimensional compaction within reasonable time complexity $O(N^2)$.

1. Introduction

The compaction is necessary along the symbolic layout procedure to correct the design rule and compact the dimension of the layout at the same time. In the other design style, the compaction is useful to reduce the human effort to create the dense layout.

The compaction method can be classified according to computation technique into the virtual grid based and constraint graph based [1]. The first one has been dominated since the constraint graph based compaction yield more compacted layout compared to the virtual grid based. Considering the direction of object movement during compaction, the compaction can be divided into three classes, 1-dimension, $1\frac{1}{2}$ -dimension [2] and 2-dimension. The 2-dimension compaction has been shown to be NP-complete problem and the $1\frac{1}{2}$ -dimension requires rather high computation time while the 1-dimension can be solved in linear time. Even through the 1-dimension compaction may not reach the optimum solution but it can reach the suboptimum solution by alternating the compaction axis within the exceptable time.

The procedure of axis alternating constraint graph based compaction can be depicted as in Fig1. Once the connectivity constraint is generated, the horizontal and vertical compaction will be performed separately. The separation constraint for the compaction axis will be created next, as in section 2, for each compaction step.

Other user-defined constraint can be added at this point [3]. The constraint graph will be solved by the longest path algorithm to find the position of each vertex that confine all constraint. Since some vertex in the graph may have slack, some allowable movement distances, after solving constraint graph, the secondary constraint can be applied to decide which positions of those vertexes should be located, i.e., minimum wire length constraint[5]. From now on, only the horizontal compaction is concern because the vertical compaction takes the same procedure.

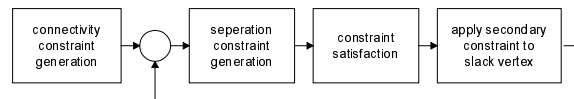


Fig 1 The procedure of axis alternating compaction

Each layout is composed of two object types, rigid object and flexible object, circuit component and wire, respectively. During the constraint graph generation, the diagonal constraints between two rigid objects is also considered to avoid DRC violation when these two object are put closer. This produces an edge which limits the compactness of the resulting layout. These edges are replaced by the invisible edge in section 3, which is invisible to the current constraint graph solving, to push these two objects far away in the next compaction step.

The horizontal wires are treated as flexible, can be shorten and extended. Whereas the vertical wire occupy one vertex in the corresponding constraint graph. In this paper vertical wire may be treated as invisible wire and invisible edge is introduced, as detailed in section 3. The number of vertex can be reduced by this embodiment as well as the complexity of the compaction.

Jog, wire bending, can create more compact layout but it causes wider layout in the other axis and also increases wire length. Then only the jog that makes more compact layout have to be inserted [4,5]. Automatic minimum jog insertion can be done at the same time when invisible edge is introduced with some more consideration after solving constraint graph, as in section 4. This technique make the any wire can be jog to obtain more compact layout even cannot do in the conventional technique.

The objects connected by the same vertical wire is tied to be one object which yield one vertex in the corresponding graph. The size of the graph is reduced and the complexity for graph solving is decrease consequently. Unfortunately, the pitch of the compacted

layout is determined by the longest object in the tied group when the tied occur in the critical path. The soft tied is introduced to solve this problem as in section 3. It allows us to break up a tie and relocate the vertexes to gain more compact layout. The vertex relocation can be done easily because the exact positions of each vertex have been known after solving the constraint graph. The complexity of the proposed compaction strategy is investigated in section 5.

2. Conventional constraint graph generation

Constraint graph represents the relative of the position of the objects in compact direction. There are many kind of the constraint graph generation [1,2,6,7]. The shadowing algorithm is outlined here because it is easy to see the constraint graph generation mechanism, even though it is not efficient. In Fig2a, there are 5 rigid object. The artificial light projected behind the object A impact object C, D and E, then the edge between vertex of A to corresponding vertex of these object are create in the constraint graph, as in Fig2b. The light have to extend beyond the edge of the object A to avoid diagonal constraint when D is compacted close to A. The light projection is circulated around each object in the layout to create the complete constraint graph. To show the number of wire object, the typical layout in [9] have 9 wires object among 24 object in the horizontal constraint graph.

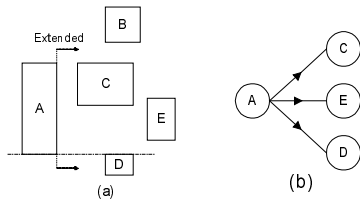


Fig 2 a) Shadowing constraint graph generation
b) The corresponding constraint graph of a).

3. The Proposed consideration during Constraint graph generation

This section is divided into three subsections corresponding to each problem explained in Section 1.

3.1. Invisible wire and invisible edge

Wire usually is the arbitrary object. Vertical wire is treated as the other object in conventional constraint graph generation. The other technique is done by discarding every wire while maintains its routability during compaction has been proposed in [8]. Wire can change its shape to make a compact layout by joggling [1,4,5,9]. Let us consider the wire pass through two objects in Fig4a. The jog can be introduced as the result in Fig4c. It is obvious that the jog cannot be done if the vertical separation between A and B is too small which mean the wire obstructs the movement of the object B.

If the wire is made to be invisible as in Fig5a, the constraint graph can be constructed as in Fig5b. The invisible edge is put between object A and B for carrying

the separation constraint to the next consecutive compaction to create the enough separation for the wire to be joggled. This constraint does not cause more complexity to the current compaction because it is invisible during the current longest path solving but may do in the next compaction step. The resulted layout after solving is shown in Fig5c.

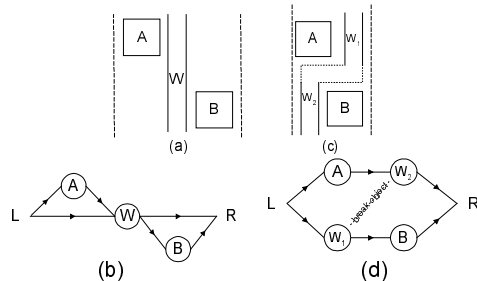


Fig 4 a) a typical situation layout b) the horizontal constraint graph of 4a. c) Layout after apply a jog to 4a d) the horizontal constraint graph of 4c

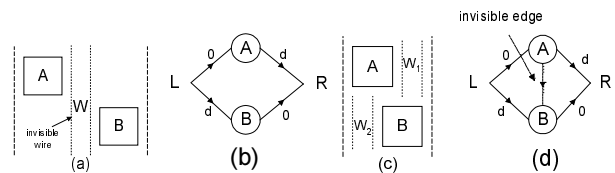


Fig 5 a) treating wire in 4a to be invisible b) the horizontal constraint graph of 5a c) layout in 5a after compaction (jog automatically introduce) d) corresponding horizontal constraint graph of 5c

3.2. Unextended shadowing and invisible edge

Consider two rigid objects lie above and under the same line as in Fig6a. The edge between A and B is obtained by the original shadowing algorithm because of the extended light for preventing the diagonal constraint violation. If this happen in the critical path the pitch of the resulting layout is the same as when A and B is on the same level. By discarding the extend light, The object A and B now are in different path which yield shorter longest path as in Fig6c. The invisible edge have to be introduced between A and B to push these two object out in the next compaction step for preventing diagonal constraint violation. One may think this invisible edge as geometric reorganization, [4], in the next compaction step. The weight of the invisible edge introduced is the minimum distance to separate this two object away to confine the design rule.

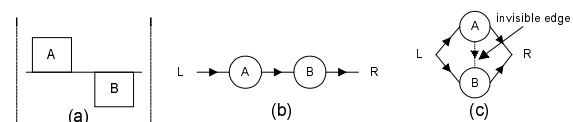


Fig 6 a) two object locate on and under the same horizontal line b) the constraint graph of 6a c) constraint graph when invisible edge has been introduced

3.3. Soft tied

The object connected by vertical wire are traditionally looked at as one vertex in the corresponding constraint graph, it is called *hard tied* from now on. The hard tied will limit the compactness when it appear on the critical path as in Fig 7b. The simple way to avoid this problem is to break this tie up which cause complexity penalty since more vertexes are obtained. The soft tied is introduced to solve this problem. Soft tied group the object as hard tied do but it can be broken up to create more compact layout as show in Fig 5c.

To establish the soft tied, the separation distance is determined from the object in the same level and the relative position between this object to the position of the group are kept during constraint graph generation. In example, the separation distance between B and E is determined from D and the relative position of D to C is kept.

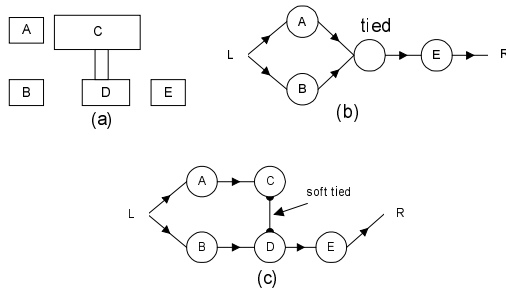


Fig 7 a) a typical layout with one tied b) corresponding constraint graph of 7a c) constraint graph after introducing and breaking the soft tied

4. Solving the constraint graph with minimum number of jog inserted and reasonable area

The constraint graph obtained in section 3 can be solved by the traditional longest path algorithm, since invisible edge is invisible during the current constraint graph solving. Once the results are obtained, the position of each vertex is known. The position of the vertex in the critical path cannot move aside except the vertexes outside this path. The movement of the vertex is called slack which can be distinguished to be local slack and global slack [5]. All vertexes are always put to the topmost (leftmost) for vertical (horizontal) compaction after longest path solving. The secondary constraint, wire length minimization, can be applied here in conventional method to the slack vertexes to choose which position should be for each slack vertex.

In practice, the direction and weight of each invisible edge corresponds to the current compaction direction. Among the set of invisible edges, the topology can be established. The invisible edge will be considered in topological order. For each invisible edge, the object at the end of edge have to be pull out to confine the weight of the edge. The invisible edge that can be confined will be delete out. The remaining invisible edge will be posed

to the next compaction step. In case of the invisible edge corresponds to invisible wire, the unnecessary jog is obviously eliminated if the corresponding edge is confined. If there are some slack left to move, the traditional secondary constraint can be applied further.

If there are some soft tied objects in the critical path, the tied is broken up as shown in Fig7c. The new position along the critical path from the tied object can be reassigned by the relative position of the object in the tied which appear in the critical path, such as object D in Fig7c.

5. Complexity computation

The fastest constraint graph algorithm generation have a time complexity of $O(N)$ where N is the total number of object in the layout [6,7]. The conventional algorithm can create invisible edge for avoiding diagonal constraint with a little modification. To create invisible edge for invisible wire, the comparison between the object and others object in the preceding scanline is needed. It takes $(N-1)$ comparison in worst case for each object, thus the complexity needed for constraint graph generation is $O(N^2)$.

The longest path algorithm takes $O(N+E)$ time complexity, where N and E is the number of vertexes and edge in the graph, respectively. Because E must not exceed $N(N-1)/2$, then the complexity is $O(N^2)$ in the worst case but $O(N)$ in practical.

The *invisible edge consideration after solving constraint graph to relocate the object and perform jog insertion needs the time complexity $O(E)$ in the worst case*. The $O(N)$ time complexity for relocating others vertexes positions for each tied broken is needed in worst case. While the maximum number of tied is $N/2$. Then the soft tied relocation takes $O(N^2)$ in worse case. Due to there are only a few soft tied appear on the critical path and the number of vertex relocation for each breaking is depend on the position of the tied occurred. The average time complexity is far less than $O(N^2)$ which is the worst case.

The overall time complexity of the proposed strategy is $O(N)$ and $O(N^2)$ for average and worst case, respectively.

VI. Conclusion

The object connected with the vertical (horizontal) wire during the horizontal (vertical) compaction is tied to be one vertex in the corresponding constraint graph by traditional compaction. Although the problem size is shrunk but the resulting layout may enlarge as seen in subsection 3.3. This problem is overcome by introducing soft tied which can be broken to relocate the object if that tied appear in the critical path which need a bit more space complexity.

During each compaction step in one-dimensional compaction, the edge for preventing diagonal constraint violation is obtained by the conventional method, which

limit the minimum width of the layout. These edge are replaced to make more compact layout by the invisible edge which put some constraint into the next compaction step without introducing significant complexity.

More compact layout is feasible by jog insertion but it may extend the width of the perpendicular direction. Jog reorganization is introduced with the cost of some more time complexity [4]. Whereas scanning to create the potential jogs and determine which potential jog should be translated to be a jog need the worst case time complexity of $O(N^2)$ [5]. Unfortunately, the number of wire without any object at its ends will increase when jogs have been inserted for each compaction. Thus with the conventional method the size of the will grow up.

Introducing invisible edge also let the size of problem not grow with the number of compaction step by treat the wire invisible. The time complexity of $O(N^2)$ in worst case is need to solve all the problem above when introducing invisible edge and soft tied.

In contrast to the traditional one-dimensional compaction which performs the compaction step by step separately, thus some information between the consecutive step are loosed. Invisible edge is introduced to one-dimensional compaction to communicate between the adjacent compaction step, then it is obvious to produce the compacted layout more closer to those of two-dimensional compaction with reasonable run time.

Reference

- [1] N. Sherwani, "Algorithms for VLSI physical design automation," Kluwer Academic Publishers, Massachusetts 1995
- [2] H. Shin, A.L. Sangiovanni-vincentelli and C.H. Sequin, "Zone-refining techniques for IC layout compaction," *IEEE Transactions on Computer-Aided Design*, pp.167-179, vol.9, no.2, February 1990
- [3] Ajay D. Kamdar, "Method and apparatus for constraining the compaction of components of a circuit layout," U.S patent no. 5636132, June 1997.
- [4] W.H. Wolf, R.G. Mathews, J.A. Newkirk and R.W. Dutton, "Algorithms for optimizing, Two-Dimensional symbolic layout compaction," *IEEE Transactions on Computer-Aided Design*, pp.451-466, vol.7, no.4, April 1988
- [5] J.F. Lee and C.K. Wong, "A performance-Aimed Cell Compactor with Automatic Jogs", *IEEE Transactions on Computer-Aided Design*, pp.1495-1507, vol.11, no.12, December 1992
- [6] J. Fang, J.S.L. Wong, K. Zhang and P. Tang, "A new fast constrain graph generation algorithm for VLSI layout compaction," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp2858-2861, 1991
- [7] J.F Lee, "A new framework of design rules for compaction of VLSI layouts," *IEEE Transactions on*

Computer-Aided Design, pp.1195-1204, vol.7, no.11, November 1988

- [8] K. Mehlhorn and S. Naher, "A faster compaction algorithm with automatic jog insertion," *IEEE Transactions on Computer-Aided Design*, pp.158-166, vol.9, no.2, February 1990
- [9] M. Sarrafzadeh and C.K. Wong, "An introduction to VLSI physical design," McGraw Hill, international edition, Singapore 1996

In detail, there are two vertical wire types, the wire with and without rigid object at its end. The number of the latter wire is unfortunately increase for each compaction step due to jog insertion.

The invisible edge let us discard some vertex, in subsection A, and some edge, in subsection B, in the constraint graph which corresponding to treat some wire are invisible and avoid diagonal constraint, respectively. The soft tied allow us to break the tied in the case that it can make more compact layout.

Two strategies is introduced, invisible edge and soft tied, to avoid three problems.