State of the Art

June 2, 2016

Contents

1 Big words, big definitions

2 Questions

2.1 Why do we even bother with hypergraphs?

All hypergraphs can be represented as regular graphs, and lots of tools already exists to handle graphs. When we want to express the wire length between two nodes, although it can easily be done using graph's edge weights, it is not expressable as a hypergraph's hyperedge weight, the hyperedge linking several nodes all connected with different wire lengths.

However, when tackling the partitioning problem, working with hypergraphs gives the vertices belonging to the same hyperedge a coherence. In the context of 3D ICs, we work with buses connecting blocks of gates, and we want them to be handled as such.

Moreover, Ihler et al. [1993] demonstrated that the mincut partition obtained for a circuit is not as accurate as would be a hypergraph's. More precisely, they define a *cut-model* in definitions 1 and 2, and show that there is no such thing in general.

Definition 1: Cut-model
An edge-weighted graph (V, E) is a cut-model for an edge-weighted hypergraph (V, H) if the weight of the edges cut by any bipartition of V in the graph is

the same as the weight of the hyperedges cut by the same bipartition in the hypergraph.

A more formal way to define the principle is as follows :

Definition 2: Cut-model and mincut-model
A graph (V, E) on k vertices is a cut-model (for a unit weight hyperedge on k vertices) if the weight of any cut induced by a non-empty proper subset W of V is equal to one.

A graph $(V \cup D, E)$ on $k+d$ vertices is called a min-cut-model (for a unit weight hyperedge on k vertices) if for every non-empty subset W of V we have that the weight of any cut with minimum weight (mincut) under those separating W from $V \setminus W$ is equal to one. It must be zero fo $W = \emptyset$.

3 Recurring notions

3.1 Complexity

3.1.1 Non-deterministic Turing Machine

In this kind of Turing machines, the transition *rule* becomes a transition *function*. It means that at a given state, with a given symbol, it has many possible execution instead of just the one with a deterministic Turing machine. We can see that as the machine always taking the best possible guess, leading to a finishing state in the end, or the machine is branching into all the possible states from the point of decision.

3.1.2 NP

Acronym for “non-deterministic polynomial-time". A “yes"-instance can be found in polynomial-time by a non-deterministic Turing machine, and this solution can be verified in polynomial-time by a deterministic Turing machine.

3.1.3 NP Hard

Acronym for “non-deterministic polynomial-time hard".
Note: it has never been proven that there is no polynomial-time algorithm to solve those problems. Informally, it means that those problems are at least as hard as the hardest problems in NP.

3.1.4 NP Complete

A problem is NP complete when it is both NP and NP hard at the same time. Although the solution can be verified in polynomial time, there is no known algorithm to find this solution quickly. The problem to know if such algorithm exist is the "P = NP" problem.

An NP-complete problem differs from a regular NP problem in the sense that a problem is NP-complete if every other problem in NP can be reduced to it.ea

3.1.5 P=NP

Problem asking if there exist polynomial time algorithm solving NP-complete problems (an by corollary all NP problems).

3.2 Optimization

3.2.1 Linear Programming

Optimization of a problem for which the requirements are represented using linear relationships.

3.2.2 Semidefinite Programming

Extends the linear programming problems, but using a positive semidefinite matrix as unknown value, instead of a vector of real integers.

4 Graph Partitioning

For a graph $G = (V, E)$, with V the set of n vertices and E the set of edges.

4.1 Balanced partition problem

In a (k, v) balanced partition problem, we want to partition G into k partitions of at most size $v \cdot \frac{n}{k}$, minimizing the capacity of the edges spanning multiple partitions Andreev and Räcke [2004]. This could allow to perform unbalanced paritioning based on the factor v; if $v = 1$, all partitions will have the same size (considering $v \in \mathbb{N}_0$). However, it is still an upper bound, not an objective. All the partitions will have the same freedom in size, this does not enforce an asymmetry in the partitions size. This problem is also know as bicriteria approximation or ressource augmentation. While we are talking

about vocabulary, when $k = 2$, we have a *bipartition problem*, and if $k > 2$, we are proceeding to a *k-way partitioning*.

4.2 Algorithms

Lim, Lim [2008]

4.2.1 Kernighan-Lin Algorithm [Kernighan and Lin, 1970]

4.2.2 Fiduccia-Mattheyses Algorithm [Fiduccia and Mattheyses, 1982]

4.2.3 EIG Algorithm [Hagen and Kahng, 1992]

4.2.4 FBB Algorithm [Yang and Wong, 1996]

4.3 Metaheuristics

Local Search Strategy A local search will look in the neighbourhood of the current solution for a better one. If there is nothing better in the direct vicinity of the position, just stop. As Pirlot [1996] says, this trategy is also known as “descent" strategy and is basically what was done in `phoney` when we tried to disbalance the energy repartition in the partitions, while keeping the area balanced.

The problem of this strategy is that it will get stuck in local minima. Escaping requires a deterioration of the solution, which is done by Simulated Annealing (SA) and Tabu Search (TS).

4.3.1 Simulated Annealing

Pirlot [1996] presents this heuristic kind of like the evolution of a local search. Instead of looking in the neighbourhood of the current solution, we pick one randomly. If it's better, keep it, otherwise we have a probabilty p to keep it, $(1 - p)$ to discard it. p is usually a Boltzmann-like distribution:

$$p(n) = \exp\left(-\frac{1}{T(n)}\Delta F_n\right)$$

with n the current step, $\Delta F_n = F(x) - F(x_n)$ the distance from the current solution, and $T(n)$ is what we could call the “temperature" by analogy with the annealing of steel. Each L steps, the temperature decreases, lowering the probability to accept a worse solution.

The stopping condition can be a number of steps from the begining or a number a steps without enough improvement of the solution.

4.3.2 Tabu Search

A Tabu Search, as its name and Pirlot [1996] indicate, is based on a tabu of some sort. We will proceed like for a regular local search, the difference being that we establish a tabu list that prevents from cycling through the same solutions. In this list, we could simply store the L last visited solutions, forbidding them. But this is still not good enough and performs poorly in practice. An other approach is the forbid *movements*, *e.g.* changing some coordinate from 0 to 1. However, should a potential solution forbidden by the tabu list be so good according to some "aspiration level", the tabu can be overlooked and the solution be chosen.

4.3.3 Genetic Algorithms

5 Manufacturing

5.1 Wafer to wafer

5.2 Die to die

5.3 Die to wafer

5.4 Monolithic

6 Electronic Design Automation

EDA is also known as "Electronic Computer-Aided Design" (ECAD).

6.1 Conferences

6.1.1 Design Automation Conference

San Diego, Anaheim and San Francisco in June.

Glossary

netlist File descripting the connectivity of an architecture. Such a list can either be *pin-oriented* or *net-oriented*. The former focuses on the design components or their associated nets, the later on the nets and their associated components.
A netlist can either be hierarchical or flat..

primitive In a netlist, a primitive is a definition which does not include any instance. As such, a flat netlist only contains primitives.

register transfer level Level at which are described the signals and the values they should take.

RTL Register Transfer Level.

SA Simulated Annealing.

TS Tabu Search.

References

K. Andreev and H. Räcke. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 120–124, New York, NY, USA, 2004. ACM. ISBN 1-58113-840-7. doi: 10.1145/1007912.1007931. URL `http://doi.acm.org/10.1145/1007912.1007931`.

C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. *Design Automation, 1982. 19th Conference on*, pages 175–181, 1982. ISSN 0146-7123. doi: 10.1109/DAC.1982.1585498.

L. Hagen and A. B. Kahng. New Spectral Methods for Ratio Cut Partitioning and Clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992. ISSN 19374151. doi: 10.1109/43.159993.

E. Ihler, D. Wagner, and F. Wagner. Modeling Hypergraphs by Graphs with the same Mincut Properties. *Information Processing Letters*, 45:171—-175, 1993. ISSN 0002-9572. doi: 10.1016/0020-0190(93)90115-P.

B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning of Electrical Circuits. *Bell System Technical Journal*, pages 291–307, 1970.

S. K. Lim. Partitioning. Lecture notes of ECE6133, Physical Design Automation of VLSI Systems.

S. K. Lim. *Practical Problems in VLSI Physical Design and Automation*. 2008. ISBN 9781402066269.

M. Pirlot. General local search methods. *European Journal of Operational Research*, 2217(92):493–511, 1996.

H. H. Yang and D. F. Wong. Efficient network flow based min-cut balanced partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1533–1540, 1996. ISSN 02780070. doi: 10.1109/43.552086.