State of the Art

November 18, 2015

1 Recurring notions

1.1 Complexity

1.1.1 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.

1.1.2 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.

1.1.3 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.

1.1.4 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

1.1.5 P=NP

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

1.2 Optimization

1.2.1 Linear Programming

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

1.2.2 Semidefinite Programming

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

2 Graph Partitioning

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

2.1 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 [1]. 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.

2.2 Algorithms

[2]

2.2.1 Kernighan-Lin Algorithm

2.2.2 Fiduccia-Mattheyses Algorithm

2.2.3 Hagen-Kahng EIG Partitioning

2.3 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 [3] 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).

2.3.1 Simulated Annealing

[3] 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.

2.3.2 Tabu Search

A Tabu Search, as its name and [3] 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.

2.3.3 Genetic Algorithms

3 Electronic Design Automation

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

3.1 Conferences

3.1.1 Design Automation Conference

San Diego, Anaheim and San Francisco in June.

3.1.2 International Conference on computer-Aided Design

3.1.3 Design Automation an Test in Europe

3.1.4 Asia and South Pacific Design Automation Conference

3.1.5 International Symposium on Quality Electronic Design

Glossary

SA Simulated Annealing.

TS Tabu Search.

References

[1] Andreev, K., and Räcke, H. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures* (New York, NY, USA, 2004), SPAA '04, ACM, pp. 120–124.

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

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