Cell Routing Using SAT

Sergio Rodríguez Guasch

Facultat d'Informàtica de Barcelona

January 17, 2018

Outline

A brief intro to SAT

The satisfiability problem

Given a CNF formula $F(x_1, ..., x_n)$ is there a satisfying assignment for F?

- A wide variety of problems can be encoded as a CNF formula
- Although NP-Complete, professional SAT-Solvers are capable to deal with huge formulas coming from the real world really fast
- It can be used for optimization

Example

$$F = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1})$$

The assignment $\{\overline{x_1}, x_2, x_3\}$ satisfies F, while the assignment $\{x_1, x_2, x_3\}$ does not.

A brief intro to SAT

Encoding problems with SAT

Very similar to 0-1 Integer Linear Programming

- A set of decisional variables X must be defined
- Logic and cardinality constraints can be implemented *efficiently* in both terms of time and space

Examples

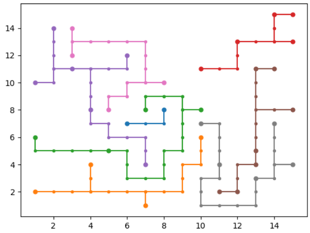
- $x_1 \implies \overline{x_2} \equiv \overline{x_1} \vee \overline{x_2}$
- $x_1 + x_2 + x_3 \leq 1 \equiv \neg(x_1 \wedge x_2 \wedge \overline{x_3}) \vee \neg(x_1 \wedge \overline{x_2} \wedge x_3) \vee \neg(\overline{x_1} \wedge x_2 \wedge x_3)$
  $\equiv (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x1 \vee \overline{x_2} \vee \overline{x_3})$

The cell routing problem

Formulation

Given a graph $G = (V, E)$, and a set of mutually disjoint subsets of vertices $S_1, ..., S_k$ (*nets*), find a subset of edges such that

- A vertex $v \in S_i$ can reach any other vertex in S_i using these edges
- A vertex $v \in S_i$ cannot reach any vertex from any other S_j
- Paths of vertices from different sets do not intersect



Making the problem easier: subnets

Problem

It is not trivial at all to encode this problem as a SAT formula

- The constraint *this set of vertices is connected* is very hard to implement
- It would require too many variables/constraints

A possible solution: subnets

This problem becomes easier if only pairs of points are considered at the same time

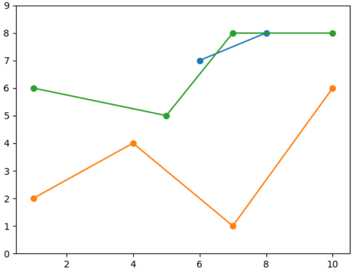
- Given a set of vertices, we can guarantee that they are connected by connecting *some pairs* of points (reachability is a transitive property)
- It is very easy to encode that two vertices must be connected

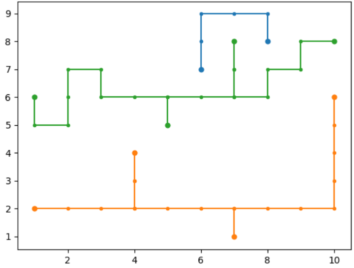
Euclidean Minimum Spanning Tree (EMST)

EMST

Consider the vertices as points, and fictional edges with the euclidean distance between endpoints as their weight, then compute the Minimum Spanning Tree. We will call the endpoints of the edges *subnets*

- Although this may make the problem unsolvable on some solvable instances, it is still preferable to intractable instances





Basic SAT Encoding

Variables

- X_e = Edge e is used by some net
- $X_{e,n}$ = Edge e is used by net n
- $X_{e,n,s}$ = Edge e is used by subnet s from net n

$$X_{e,n,s} \implies X_{e,n} \implies X_e$$

The other direction of implications can be ignored

Basic SAT Encoding

Constraints

- An edge can be used by at most one net

$$\sum_{n \in N} X_{e,n} \leq 1 \quad \forall e \in E$$

- Subnet endpoints use exactly one $X_{e,n,s}$

$$\sum_{e \in edg(s)} X_{e,n,s} = 1 \quad \forall n \in N \quad \forall s \in subnets(n)$$

- Non-endpoint vertices have, for all nets and subnets, either zero or two $X_{e,n,s}$

$$\sum_{e \in adj(v)} X_{e,n,s} \neq 1 \wedge \sum_{e \in adj(v)} X_{e,n,s} \leq 2$$

$$\forall v \in V : \neg \mathrm{endpoint}(v), n \in N \quad \forall s \in subnets(n)$$

Basic SAT Encoding

Constraints

- At most one net can pass through any vertex

$$X_{e_1,n_1} \implies \overline{X_{e_2,n_2}}$$

$$\forall e_1, e_2 \in E \times E : e_1 \neq e_2 \wedge \mathrm{common}(e_1, e_2) \quad \forall n_1, n_2 \in N \times N : n_1 \neq n_2$$

Observation

This encoding does not prevent a solution to have random, unnecesary cycles that are not connected to any net

- It is very hard to avoid them in SAT, and can be manually deleted after getting a solution
- If we minimize the function $\sum_{e \in E} X_e$ they will implicitly disappear

Additional constraints

Bounding boxes

- Most real-life datasets are such that the endpoints of a subnet can be connected without the need of going *too outside* of their bounding box
- In our implementation we prevent all the $X_{e,n,s}$ such that the endpoints of e are too far (distance 3) to be used
- Although some potential solutions may be lost, it is not the common case

Optimizing wire length

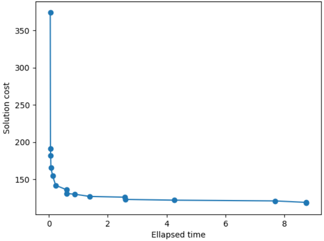
Optimizing with SAT

- Ideally, we would like to set an objective function like we do in ILP $\min \sum_{e \in E} X_e$
- Although SAT does not support optimization, we can emulate it by performing repeated calls to the solver with different constraints $\sum_{e \in E} X_e \leq k$
- We know a solution is optimal (under our EMST choice) when it has cost k and there is no solution of cost $k - 1$
- Decreasing and increasing cardinality constraints can be implemented in an incremental way, letting us to reuse already implemented formulas
- Even if it is tempting to perform a binary search, it is not always the best choice. SAT-Solvers *learn* new clauses while solving instances. These new clauses can be reused in future calls

Results

Main results

- It is not hard to get an initial solution
- Most of the CPU time is spent making minor improvements to a good enough bound



Results

Observations

- The size of the formula grows very fast
- The hardest instances seem to be the ones that are neither very sparse nor dense (somewhere in the middle)

Nets	Vars	Clauses	Time to reach optimum
1	2262	3757	0.000s
2	6966	19284	2.028s
3	11658	39932	10.461s
4	17796	69195	11.940s
5	25752	106308	36.040s
6	34716	151624	677.148s
7	39822	191448	68.465s
8	46362	239392	23.807s

Conclusions

Main conclusions

- This problem presents a very *SAT-Friendly* encoding
- This approach only works well on small instances
- Other optimizations and techniques are needed in order to solve bigger instances

Thanks

Thanks for your attention