An hybrid approach to solve boolean formulations of routing problems

William Aufort

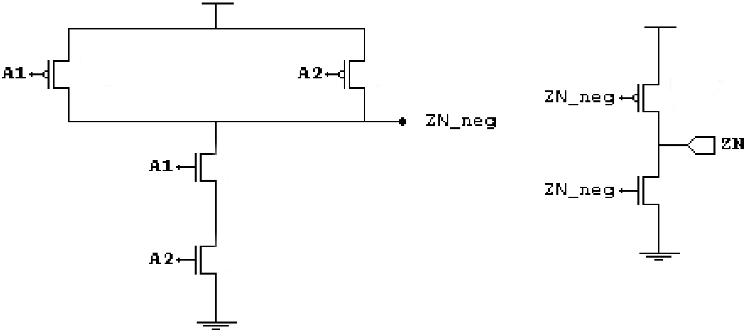
September, 9th 2015

ENS DE LYON

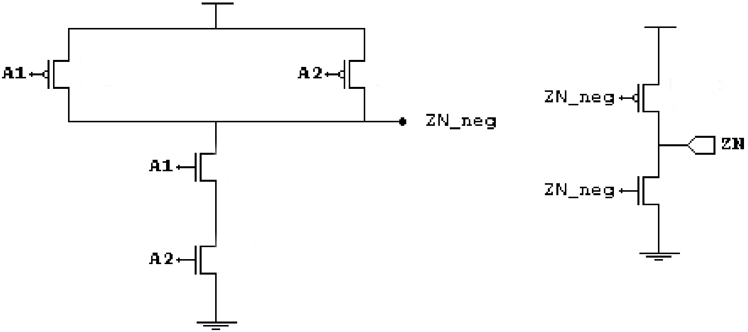
UPC

Plan

Design the circuit

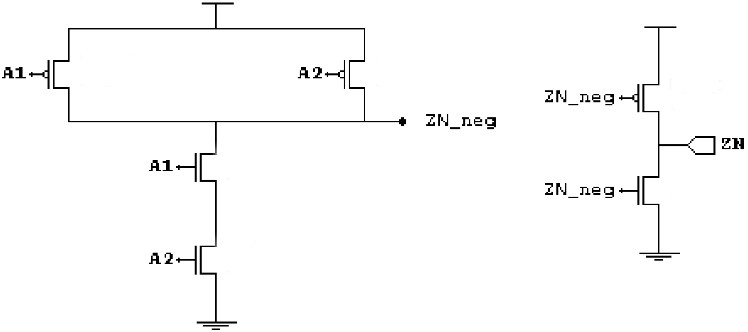


Design the circuit



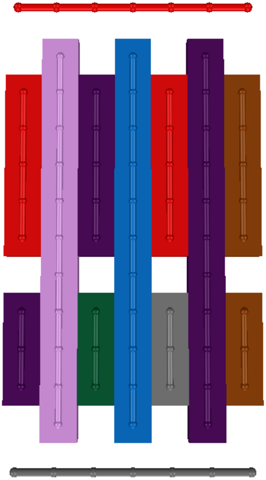
$$ZN = ?$$

Design the circuit

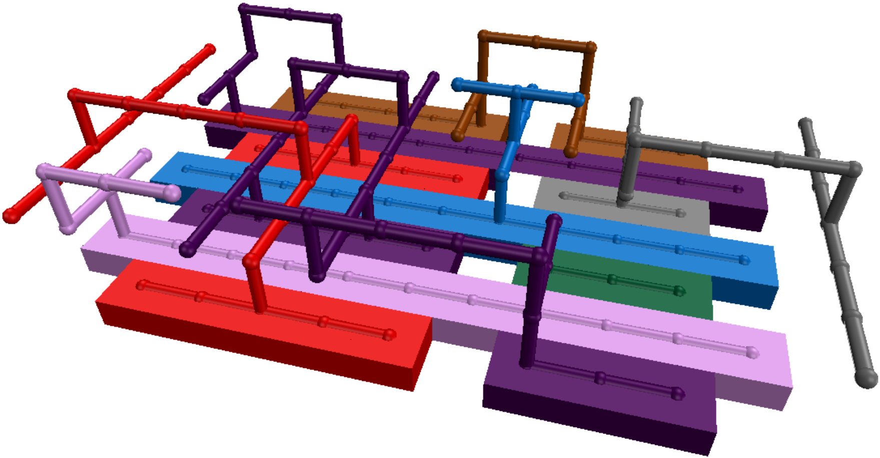


$$ZN = A_1 \wedge A_2$$

Place the transistors



And route !

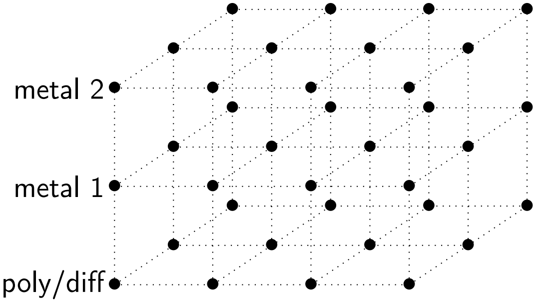


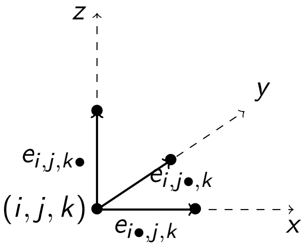
A difficult step...

- Complex design rules due to the decreasing size of the transistors
- Much more complicated to have a technology-independent algorithm
- Basis of my work : an algorithm based on a boolean approach
- $\Rightarrow$ need to model the problem

Plan

Grid and wires



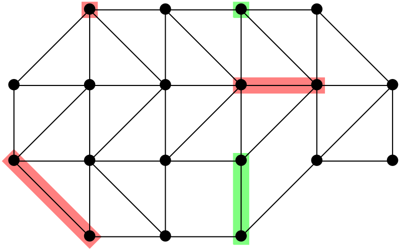


General Definitions

Definition

- A **terminal** is a set of grid points.
- A **net** $n \subset V$ is a set of terminals that must be connected. A **n-terminal** is a terminal associated to the net n.
- A **subnet** s is a pair of terminals of the same net.

Goal : Connect the terminals of each net respecting the design rules

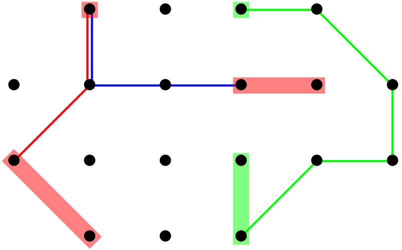


General Definitions

Definition

- A **terminal** is a set of grid points.
- A **net** $n \subset V$ is a set of terminals that must be connected. A **n-terminal** is a terminal associated to the net n.
- A **subnet** s is a pair of terminals of the same net.

Goal : Connect the terminals of each net respecting the design rules



Boolean variables and consistency constraints

$$\rho(e) \qquad \rho(e, n) \qquad \rho(e, n, s)$$

Boolean variables and consistency constraints

$$\begin{array}{ccc} \rho(e) & \rho(e,n) & \rho(e,n,s) \\ \downarrow & \downarrow & \downarrow \\ \text{wire} & \text{net} & \text{subnet} \end{array}$$

Boolean variables and consistency constraints

$$
\begin{array}{ccccc}
\rho(e) & \Leftarrow & \rho(e,n) & \Leftarrow & \rho(e,n,s) \\
\downarrow & & \downarrow & & \downarrow \\
\text{wire} & & \text{net} & & \text{subnet}
\end{array}
$$

Definition (Consistency constraints)

$$\forall e, \left(\bigvee_{n \in \mathcal{N}} \rho(e,n) \right) \Rightarrow \rho(e)$$

$$\forall e, \forall n, \left(\bigvee_{s \in \mathrm{subnets}(n)} \rho(e,n,s) \right) \Rightarrow \rho(e,n)$$

Boolean variables and consistency constraints

$$\begin{array}{ccccc} \rho(e) & \Leftarrow & \rho(e,n) & \Leftarrow & \rho(e,n,s) \\ \downarrow & & \downarrow & & \downarrow \\ \text{wire} & & \text{net} & & \text{subnet} \end{array}$$

Definition (Consistency constraints)

$$\forall e, \left(\bigvee_{n \in \mathcal{N}} \rho(e,n) \right) \Rightarrow \rho(e)$$

$$\forall e, \forall n, \left(\bigvee_{s \in \text{subnets}(n)} \rho(e,n,s) \right) \Rightarrow \rho(e,n)$$

$$\forall e, \forall n, \forall n' \neq n, \quad \rho(e,n) \Rightarrow \neg\rho(e,n')$$

$$\forall n, \forall e, \forall e' \text{adjacent to } e, \quad (\rho(e,n) \wedge \rho(e')) \Rightarrow \rho(e',n)$$

Routability constraints

- External edges $\mathrm{Ext}(S) = \{(u, v)/u \in S \wedge v \notin S\}$
- $\mathrm{Terminal}(S, n, s) = \sum_{e \in \mathrm{Ext}(S)} \rho(e, n, s) = 1$
- $\mathrm{Nadj}(v, s, n) = \sum_{e \in adv(v)} \rho(e, n, s)$
- $\mathrm{Degree0or2}(v, n, s) = \mathrm{Nadj}(v, s, n) = 0 \vee \mathrm{Nadj}(v, s, n) = 2$

Routability constraints

- External edges $\text{Ext}(S) = \{(u, v)/u \in S \wedge v \notin S\}$
- $\text{Terminal}(S, n, s) = \sum_{e \in \text{Ext}(S)} \rho(e, n, s) = 1$
- $\text{Nadj}(v, s, n) = \sum_{e \in adv(v)} \rho(e, n, s)$
- $\text{Degree0or2}(v, n, s) = \text{Nadj}(v, s, n) = 0 \vee \text{Nadj}(v, s, n) = 2$

Theorem

We can express the routability constraints only with local properties :

$$Terminal(S, n, s) \wedge \bigwedge_{v \notin S \cup T} Degree0or2(v, n, s)$$

Boolean variables and formulas

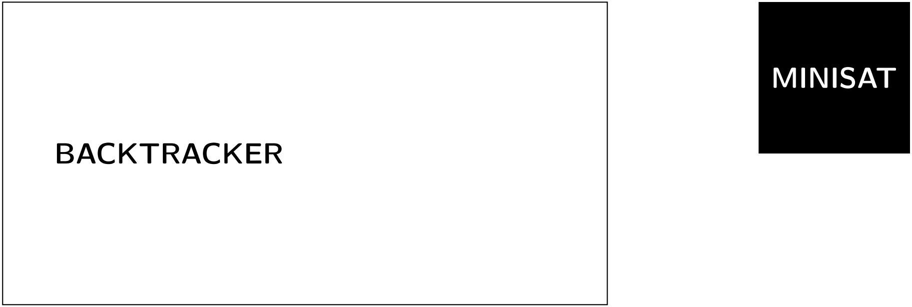
- Add the design rules constraints and we get the whole formula
- Solve it using a SAT solver $\Rightarrow$ a particular solution
- Find the "best" one
  - Best ? $\Rightarrow$ wirelength
  - Incremental approach to optimize : Rerouting using integer linear programming

Motivations ?

- A technology-independant algorithm
- Solving part given to a solver $\Rightarrow$ **BLIND**
- The main idea of my internship : If the solver "knows" the problem :
  - Speed up the search of a particular routing
  - A better initial solution
  - And so, a reduced optimization time

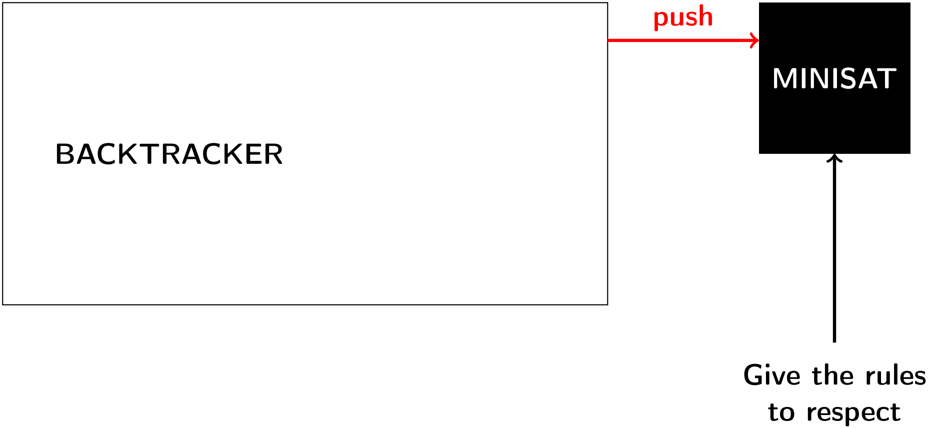
Plan

Hybrid approach : general idea

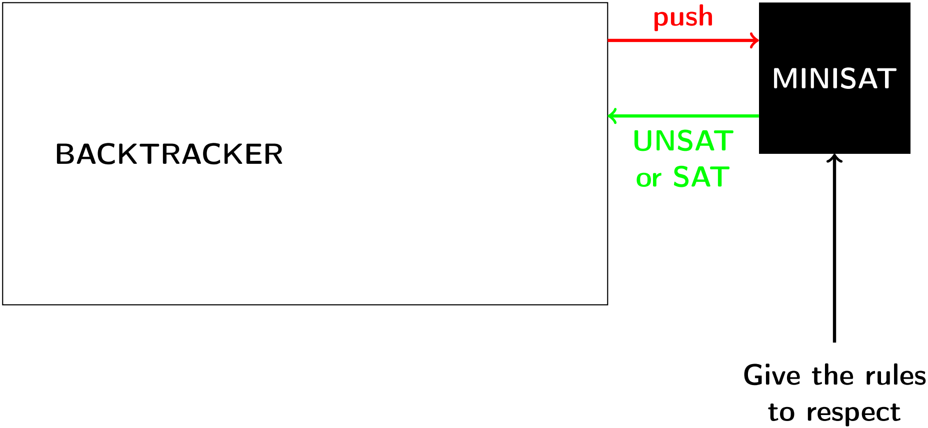


Hybrid approach : general idea

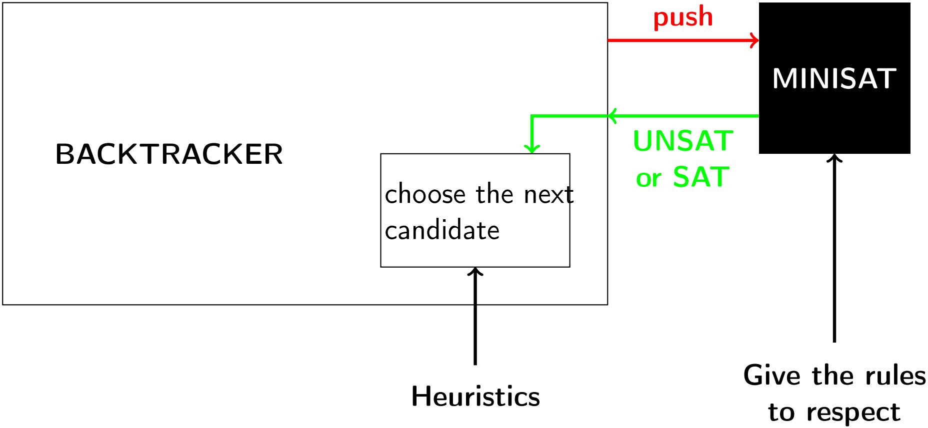
Hybrid approach : general idea



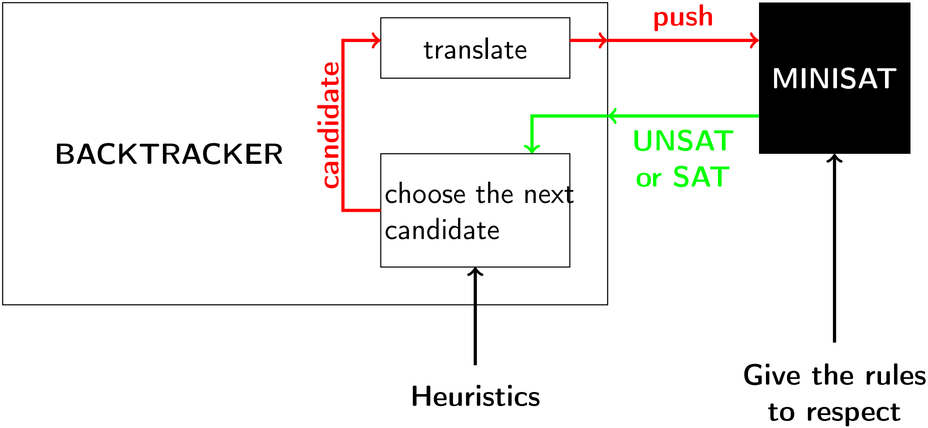
Hybrid approach : general idea



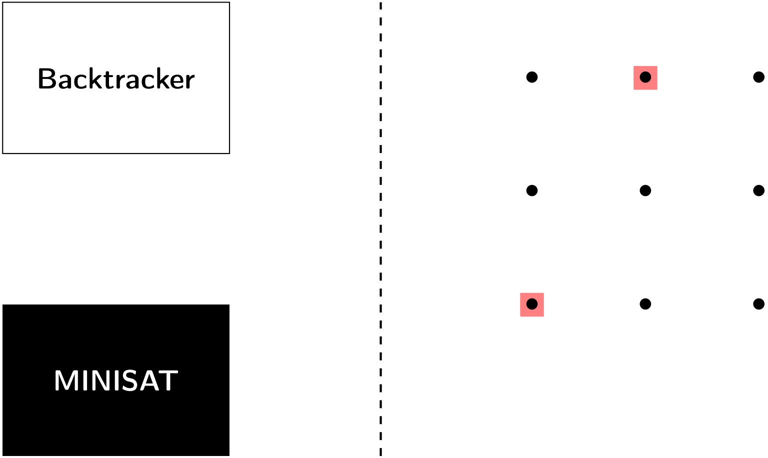
Hybrid approach : general idea



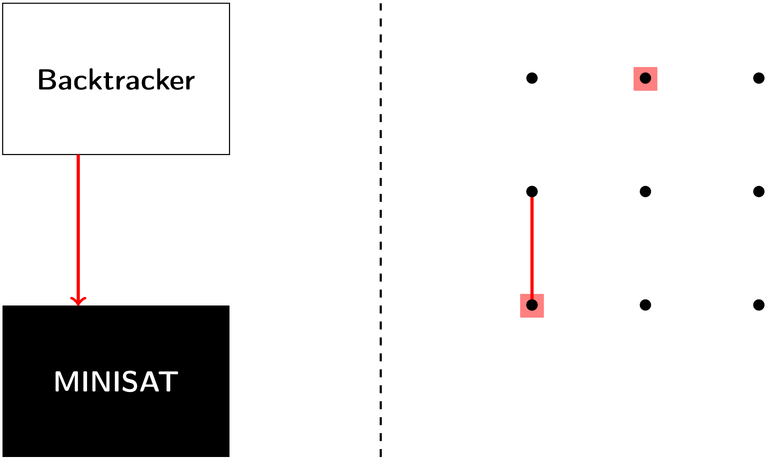
Hybrid approach : general idea



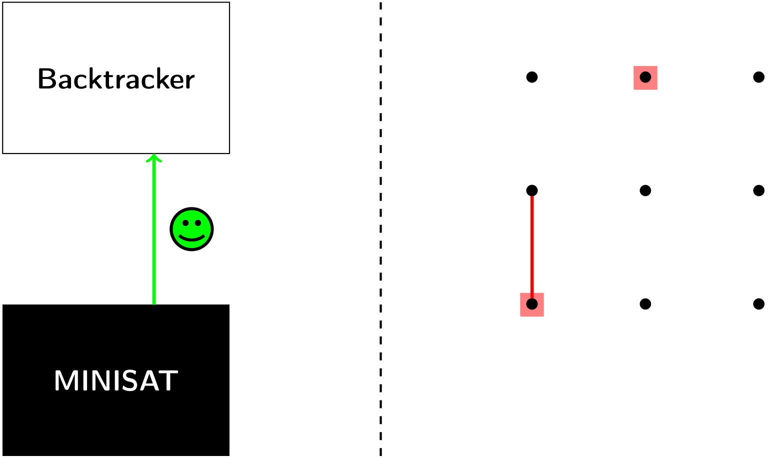
An example !



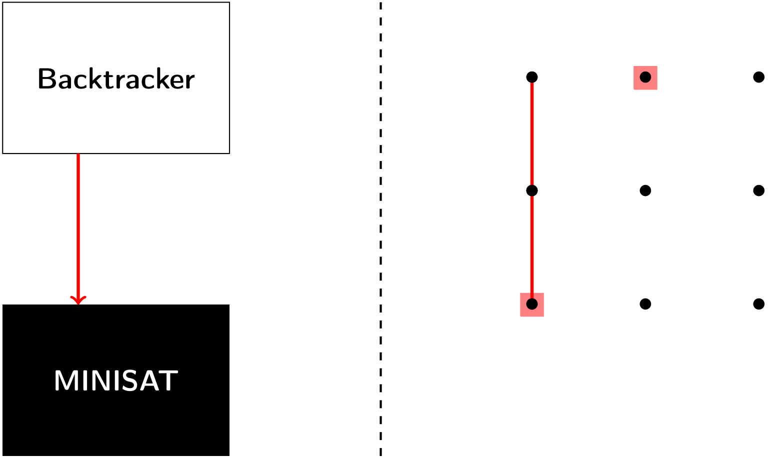
An example !



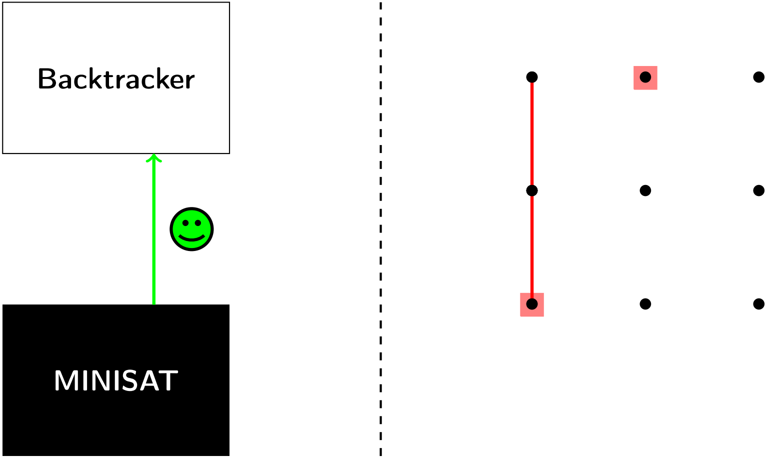
An example !



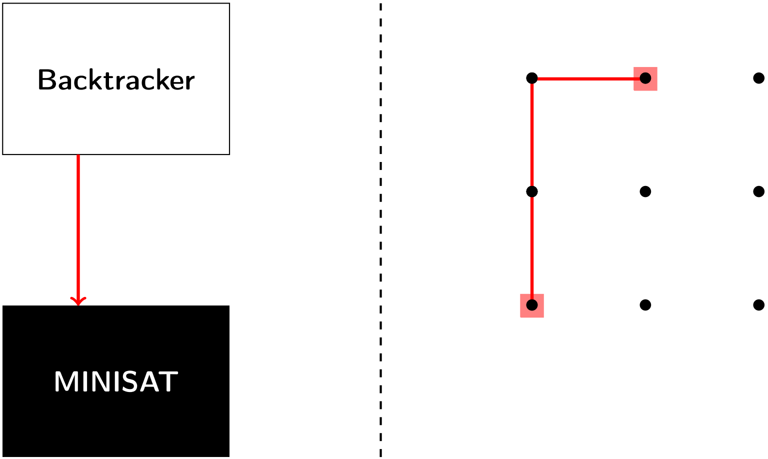
An example !



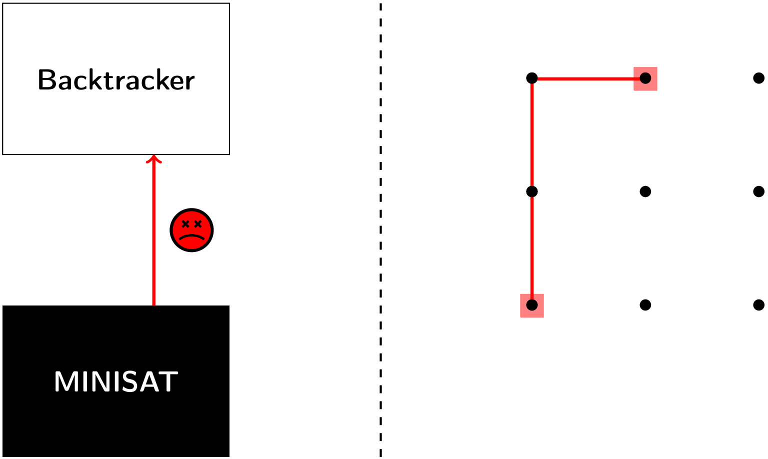
An example !



An example !



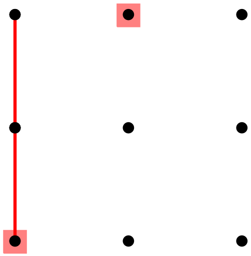
An example !



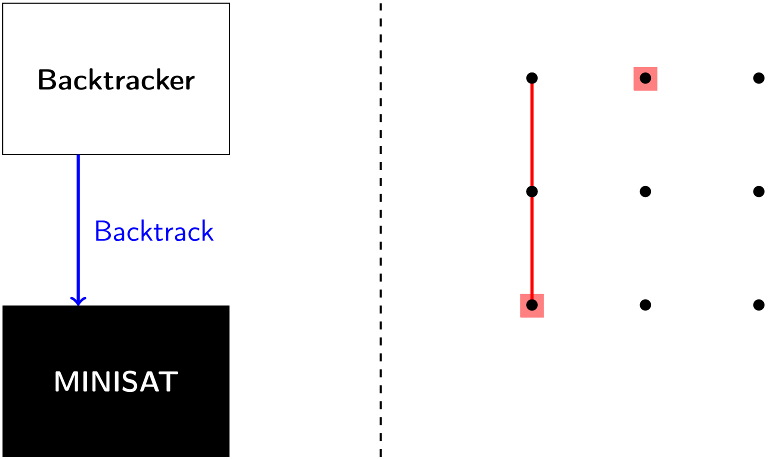
An example !

Backtracker

MINISAT



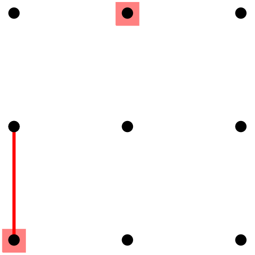
An example !



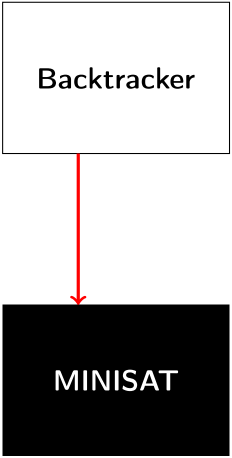
An example !

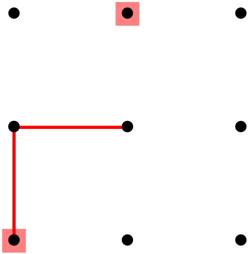
Backtracker

MINISAT

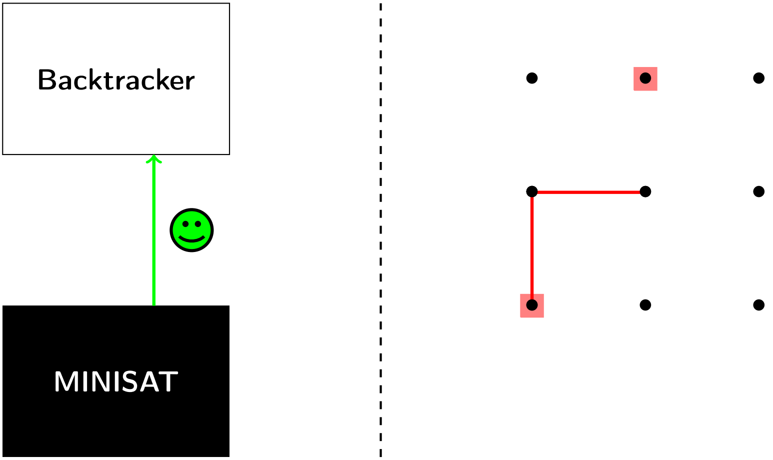


An example !

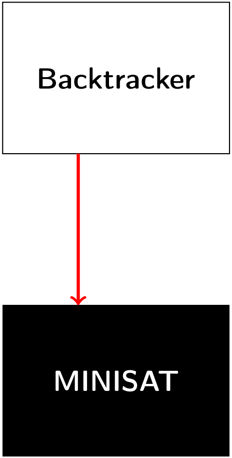


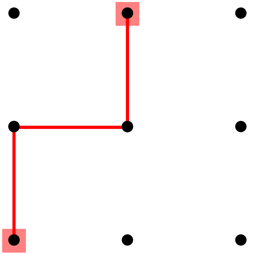


An example !

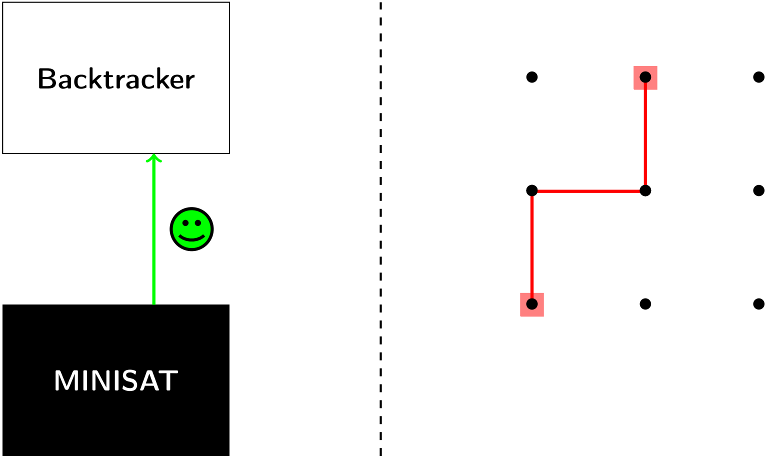


An example !

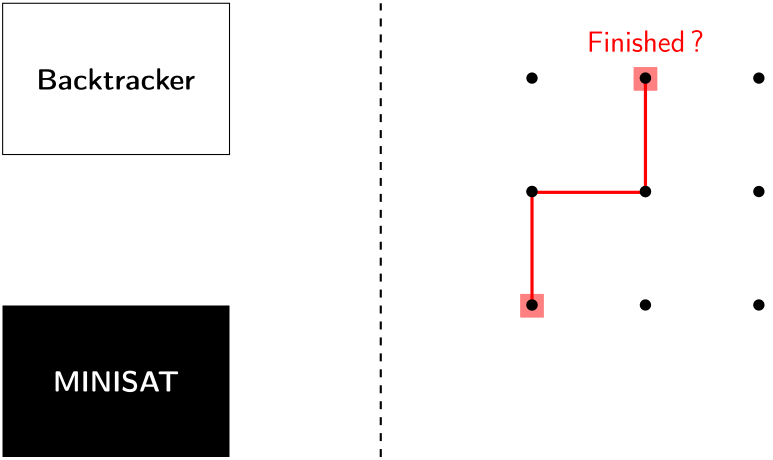




An example !



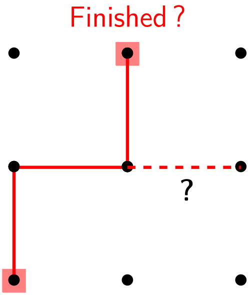
An example !



An example !

Backtracker

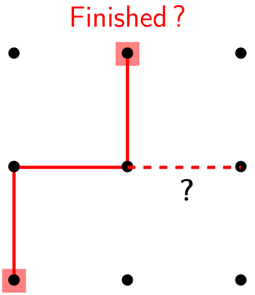
MINISAT



An example !

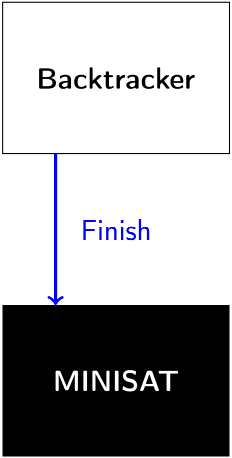
Backtracker

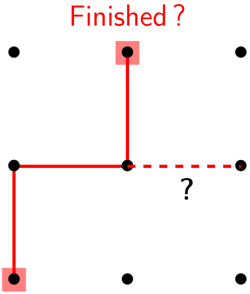
MINISAT



Depending whether there is unit propagation or not

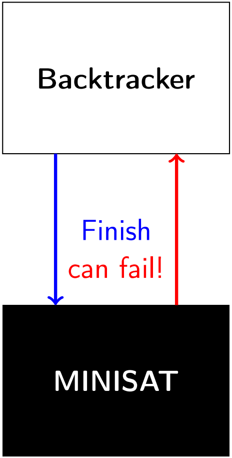
An example !

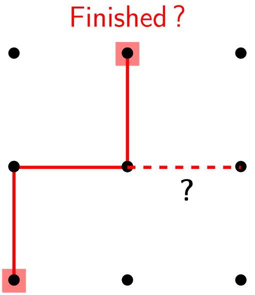




Depending whether there is unit propagation or not

An example !





Depending whether there is unit propagation or not

Some precisions :

- Heuristics ?

Some precisions :

- Heuristics ? 3 levels :

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route ⇒ the most difficult first

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route $\Rightarrow$ the most difficult first
  - Then, choose two vertices to connect

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route ⇒ the most difficult first
  - Then, choose two vertices to connect ⇒ the closer ones

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route $\Rightarrow$ the most difficult first
  - Then, choose two vertices to connect $\Rightarrow$ the closer ones
  - Finally, build the route between the two vertices

Some precisions :

- Heuristics ? 3 levels :
  - First, choose a subnet to route $\Rightarrow$ the most difficult first
  - Then, choose two vertices to connect $\Rightarrow$ the closer ones
  - Finally, build the route between the two vertices $\Rightarrow$ shortest path
- First conclusions : termination is ensured, good for small cells but too much long with medium-sized cells.

Improvements

Improvements

- Restrictives heuristics : maximal length for the paths

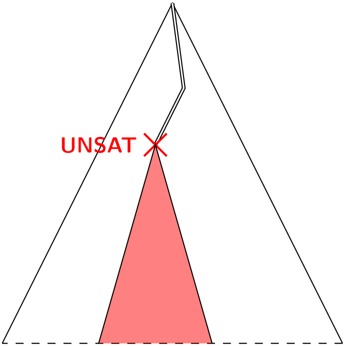
$$L = \alpha . l + \beta$$

Improvements

- Restrictives heuristics : maximal length for the paths

$$L = \alpha . l + \beta$$

- Partial routing $\Rightarrow$ allow us to skip some nets

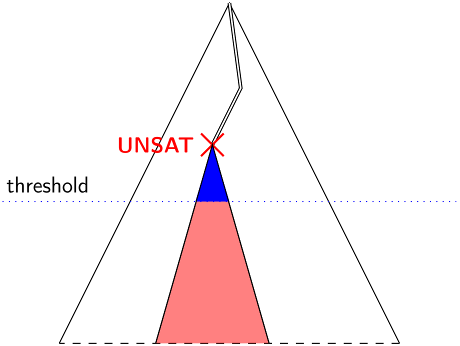


Improvements

- Restrictives heuristics : maximal length for the paths

$$L = \alpha . l + \beta$$

- Partial routing $\Rightarrow$ allow us to skip some nets

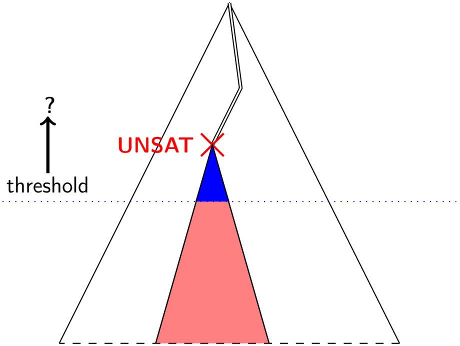


Improvements

- Restrictives heuristics : maximal length for the paths

$$L = \alpha.l + \beta$$

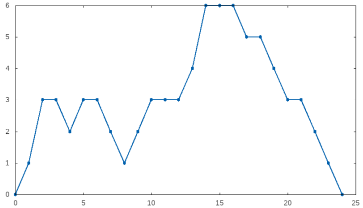
- Partial routing ⇒ allow us to skip some nets

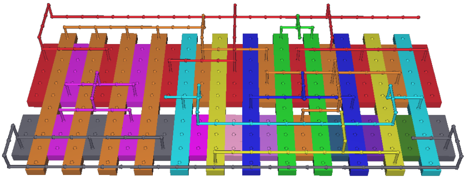


Congestion as a threshold ?

Definition

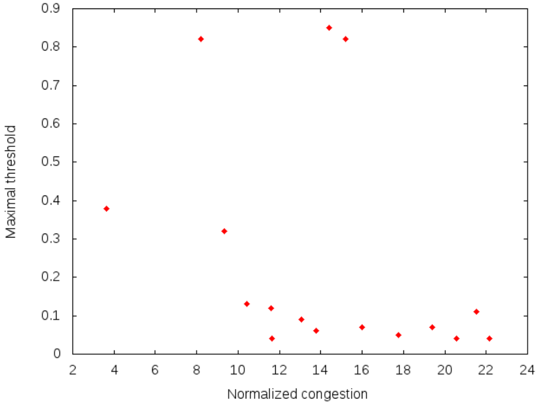
- $\text{cross}(n, s, c) =$
  $\begin{cases} 1 & \text{if } c \text{ is between the column of } S \text{ and the one of } T \\ 0 & \text{otherwise} \end{cases}$
- Congestion of the cell : $\frac{1}{\#columns} \sum_{c} \sum_{(n,s)} \text{cross}(n, s, c)$





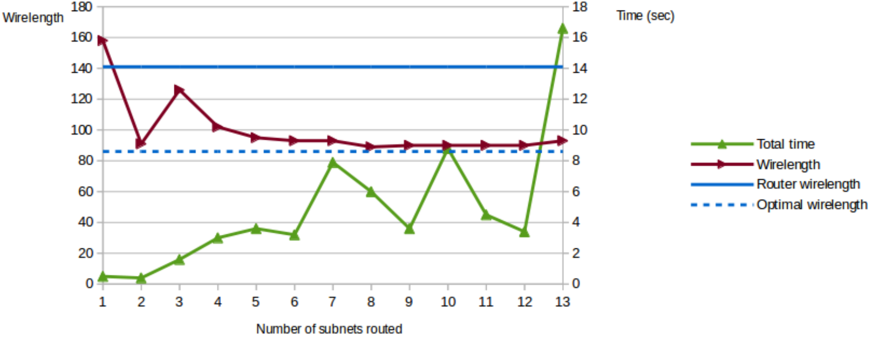
Idea : high congestion represents a high of difficulty, and conversely.

Congestion as a threshold ?



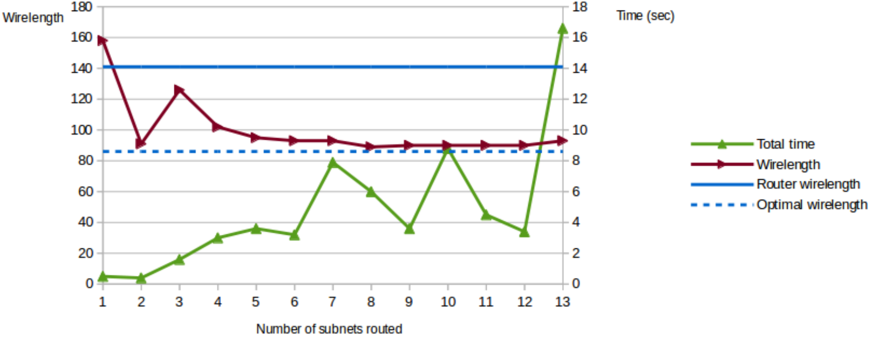
- No link between the congestion and the wanted threshold
- What's more, no satisfying minimal value for a threshold

Partial routing and skipping subnets : experiments



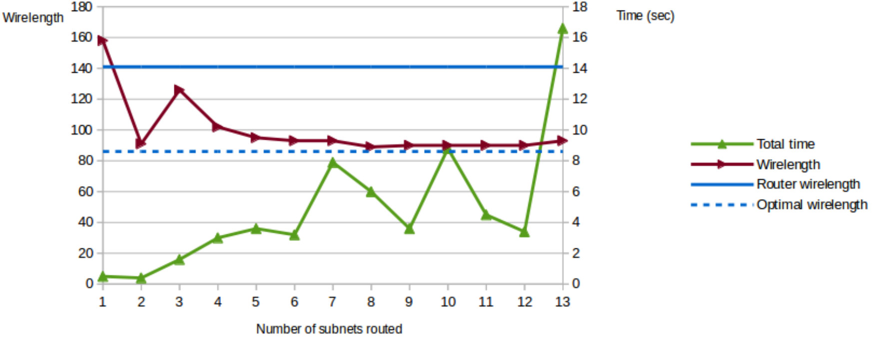
- Original router : 1 second.

Partial routing and skipping subnets : experiments



- Original router : 1 second.
- Wirelength close to the optimal

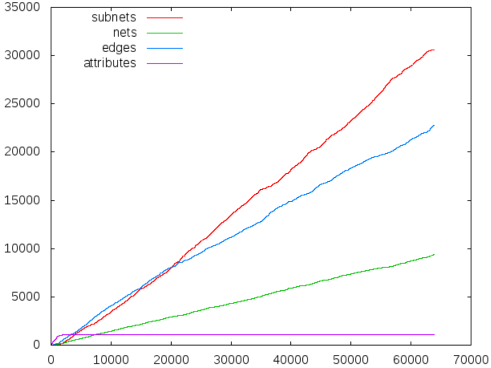
Partial routing and skipping subnets : experiments



- Original router : 1 second.
- Wirelength close to the optimal
- Too big total time, and no way to find a correct threshold

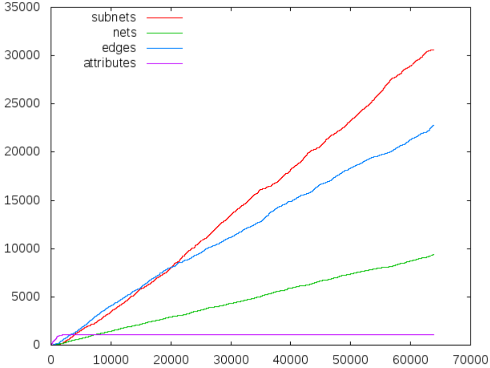
Plan

Main idea



35000
30000
25000
20000
15000
10000
5000
0
0
10000
20000
30000
40000
50000
60000
70000
subnets
nets
edges
attributes

Main idea



- We already know that Minisat is blind
- But choose the variables with heuristics seems too simple

How to act inside Minisat

- Intermediate approach : work more deeper in Minisat and let it more freedom

How to act inside Minisat

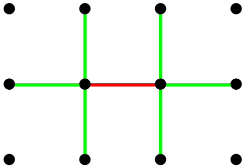
- Intermediate approach : work more deeper in Minisat and let it more freedom
- Two ways : activity and polarity (in a priority queue)

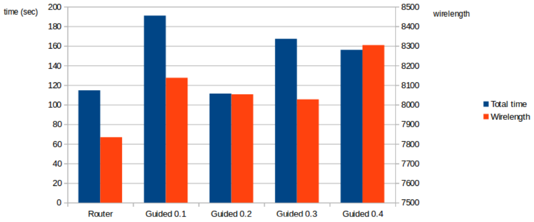
How to act inside Minisat

- Intermediate approach : work more deeper in Minisat and let it more freedom
- Two ways : activity and polarity (in a priority queue)
- On the activity (order) : More or less importance to some variables
  $\Rightarrow$ "Variable ordering is a traditional target for improving SAT-solvers"
- On the polarity : Change the default polarity

On the activity : subnet choice

Idea : give a "path building" choice of the variables.





Conclusion : values closed to the original method, but worse.

Less relevant tries

- Polarity
- Top of the queue
- Initial values
- But all these heuristics are simple !

Conclusion

- Hybrid approach : interesting but bad results.
- More deeper in Minisat : not explored a lot, only simple heuristics

Questions ?