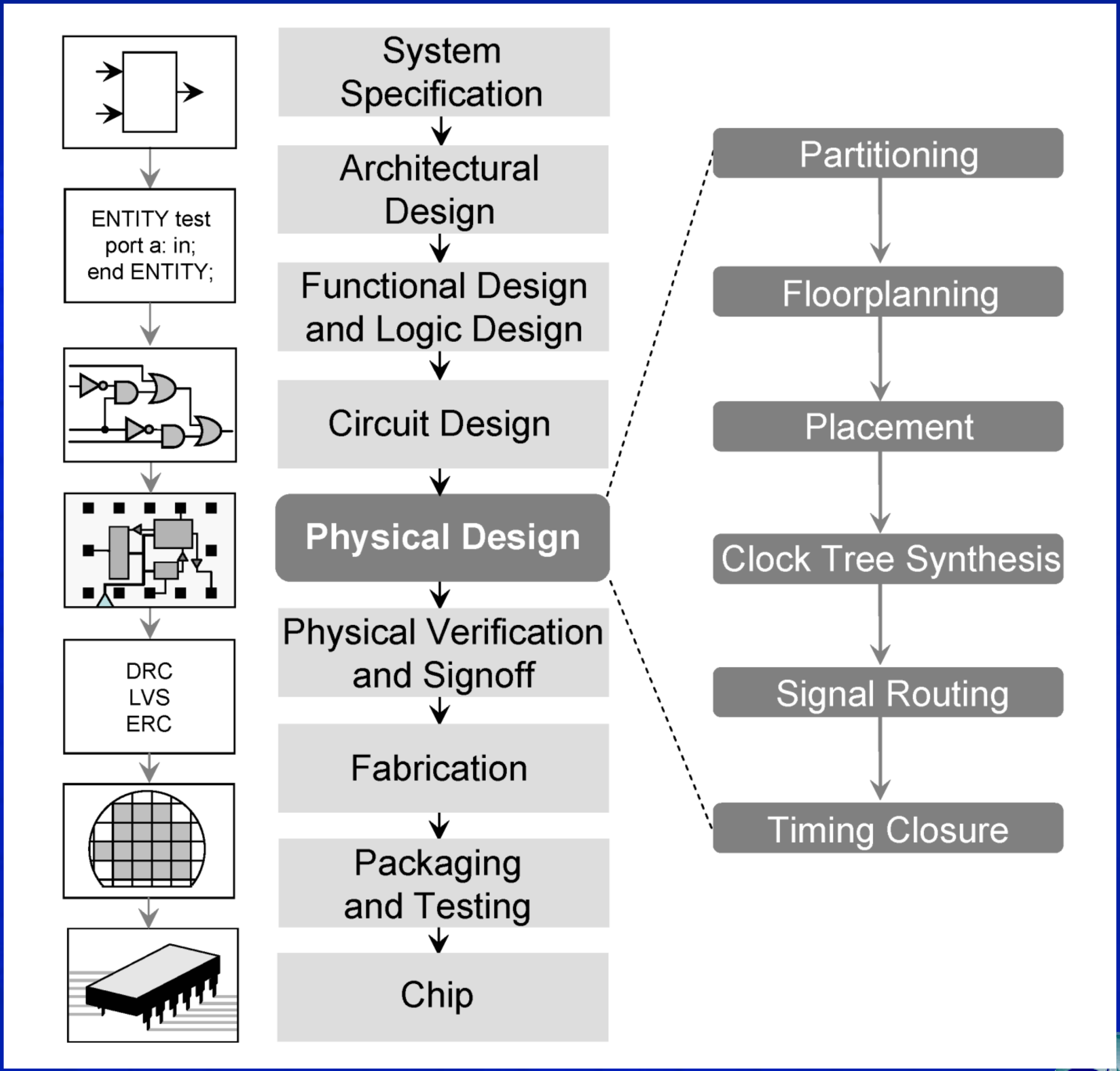
Routing under Constraints

Alexander Nadel
Intel, Israel

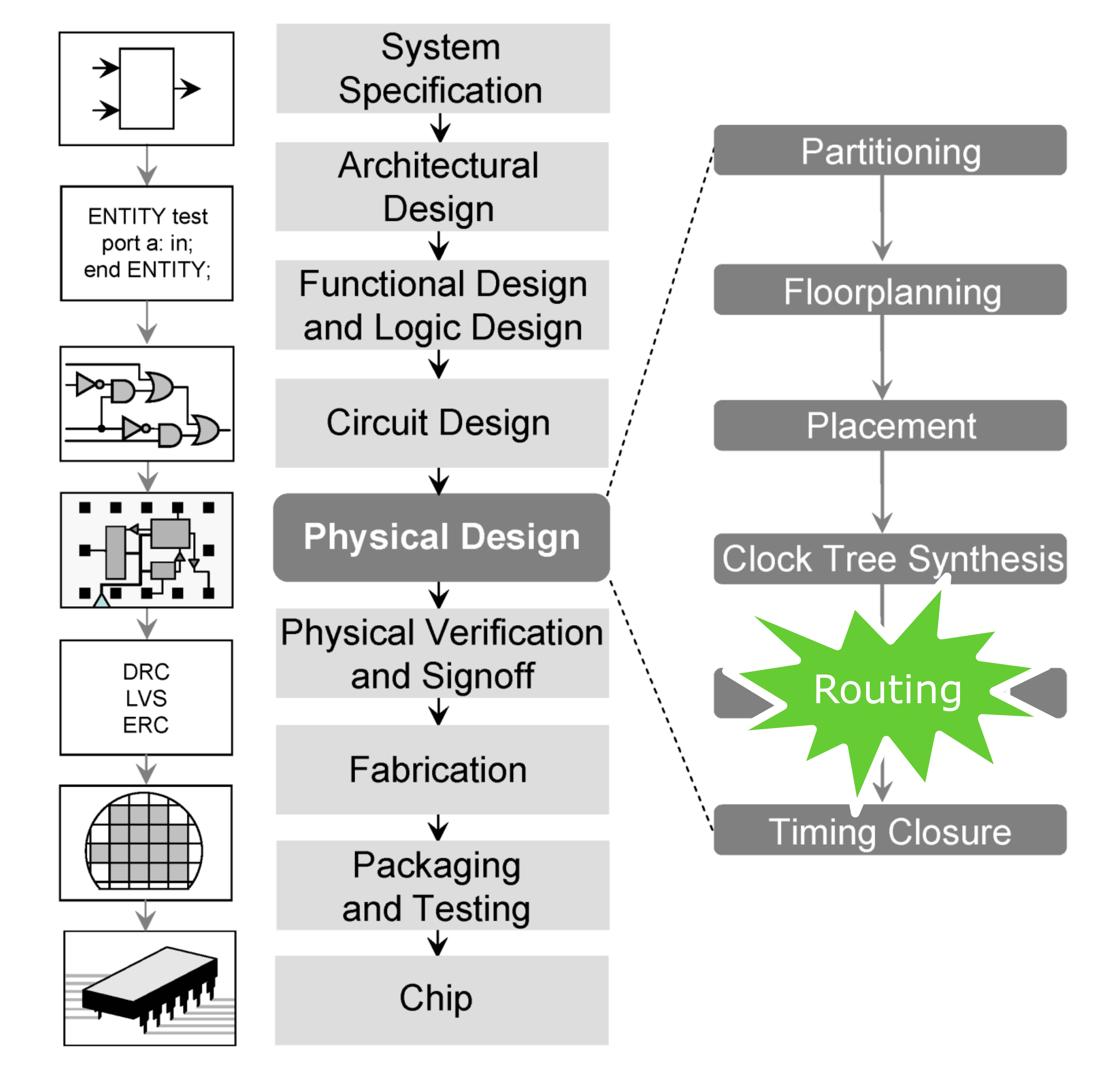
FMCAD
Mountain View CA, USA
October 4, 2016

Leap ahead™

Design and Technology Solutions



"PhysicalDesign" by Linear77 - Own work. Licensed under CC BY 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:PhysicalDesign.png#/media/File:PhysicalDesign.png



"PhysicalDesign" by Linear77 - Own work. Licensed under CC BY 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:PhysicalDesign.png#/media/File:PhysicalDesign.png

Leap ahead™

Design and Technology Solutions

Outline

Goal: Design a Scalable Design Rule-aware Router

Routing under Constraints (RUC): Problem Formalization

Bit-Vector / SAT Encoding

Doesn't scale

DRouter through SAT Solver Surgery

A*-based decision strategy (emulates constraints!)	Graph conflict analysis	Net restarting & net swapping

Unsolved crafted and industrial RUC instances are routed!

Leap ahead™
Design and Technology Solutions

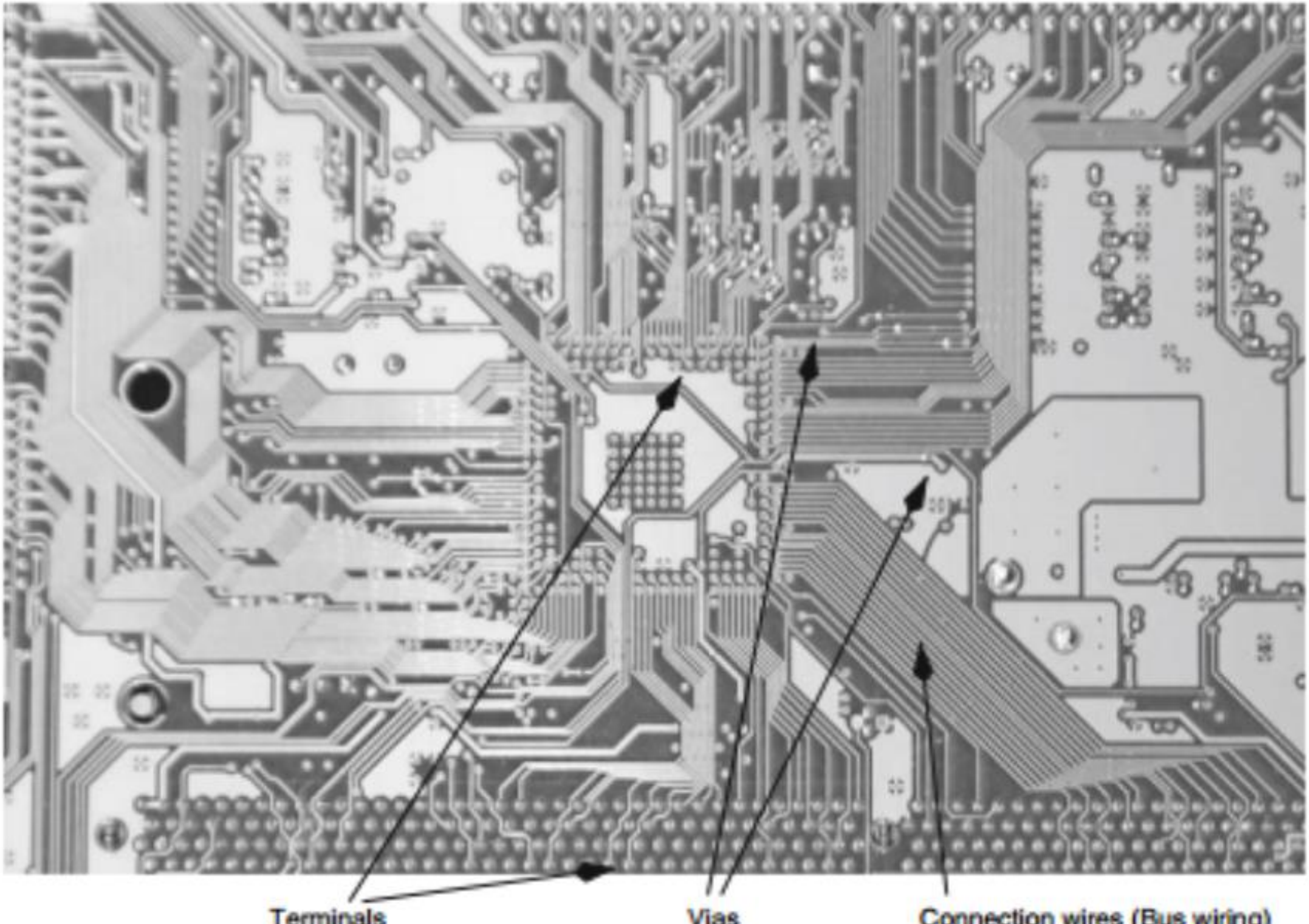
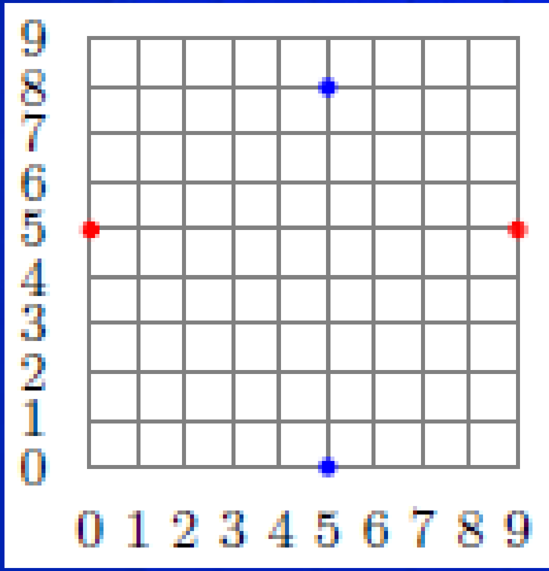


Fig. 3 Bottom side of a printed circuit board

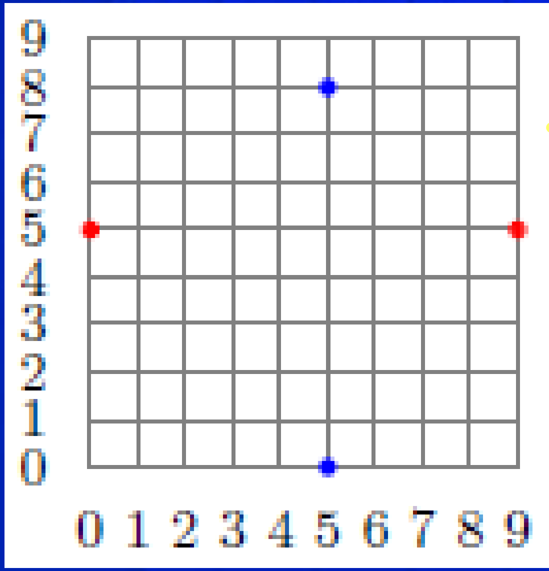
Routing: Input
(AKA Steiner Tree Packing Problem)

Input



Routing: Input
(AKA Steiner Tree Packing Problem)

Input



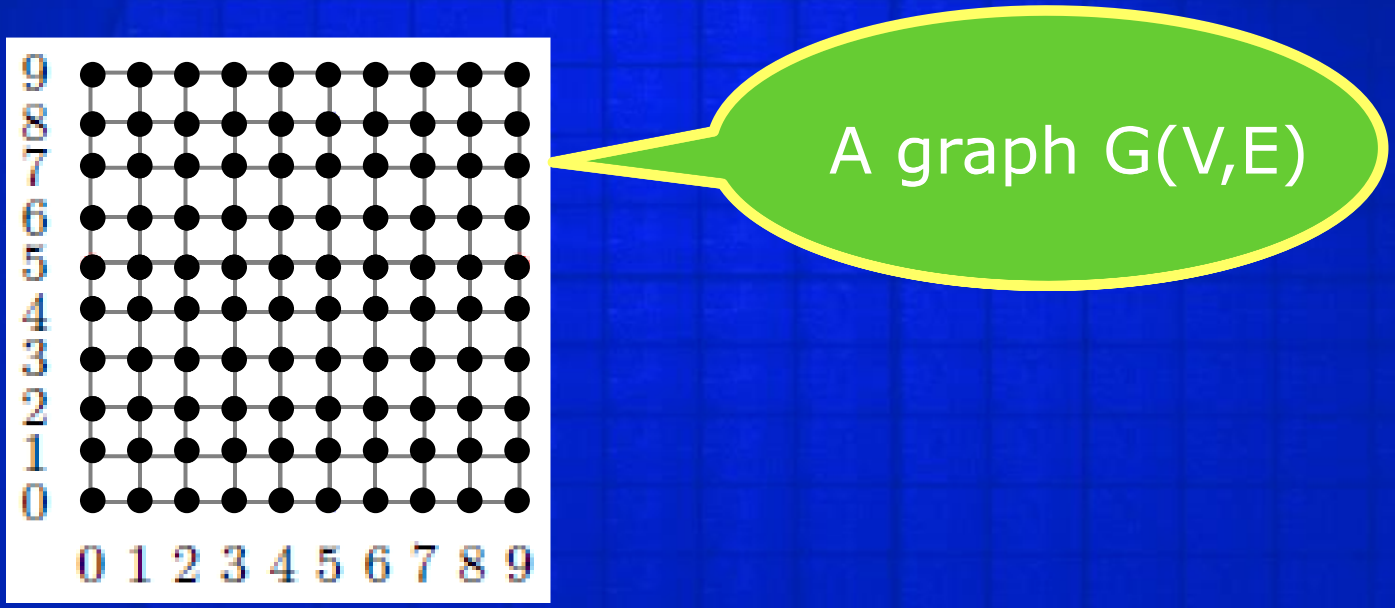
A graph G(V,E)

Design and Technology Solutions

Routing: Input

(AKA Steiner Tree Packing Problem)

Input

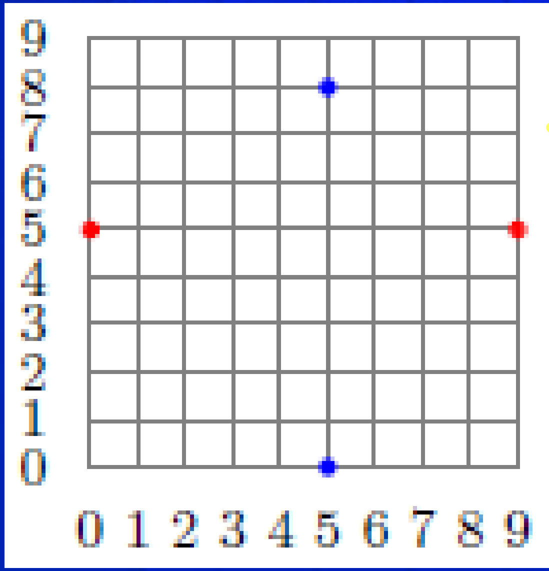


Design and Technology Solutions

Leap ahead™

Routing: Input
(AKA Steiner Tree Packing Problem)

Input

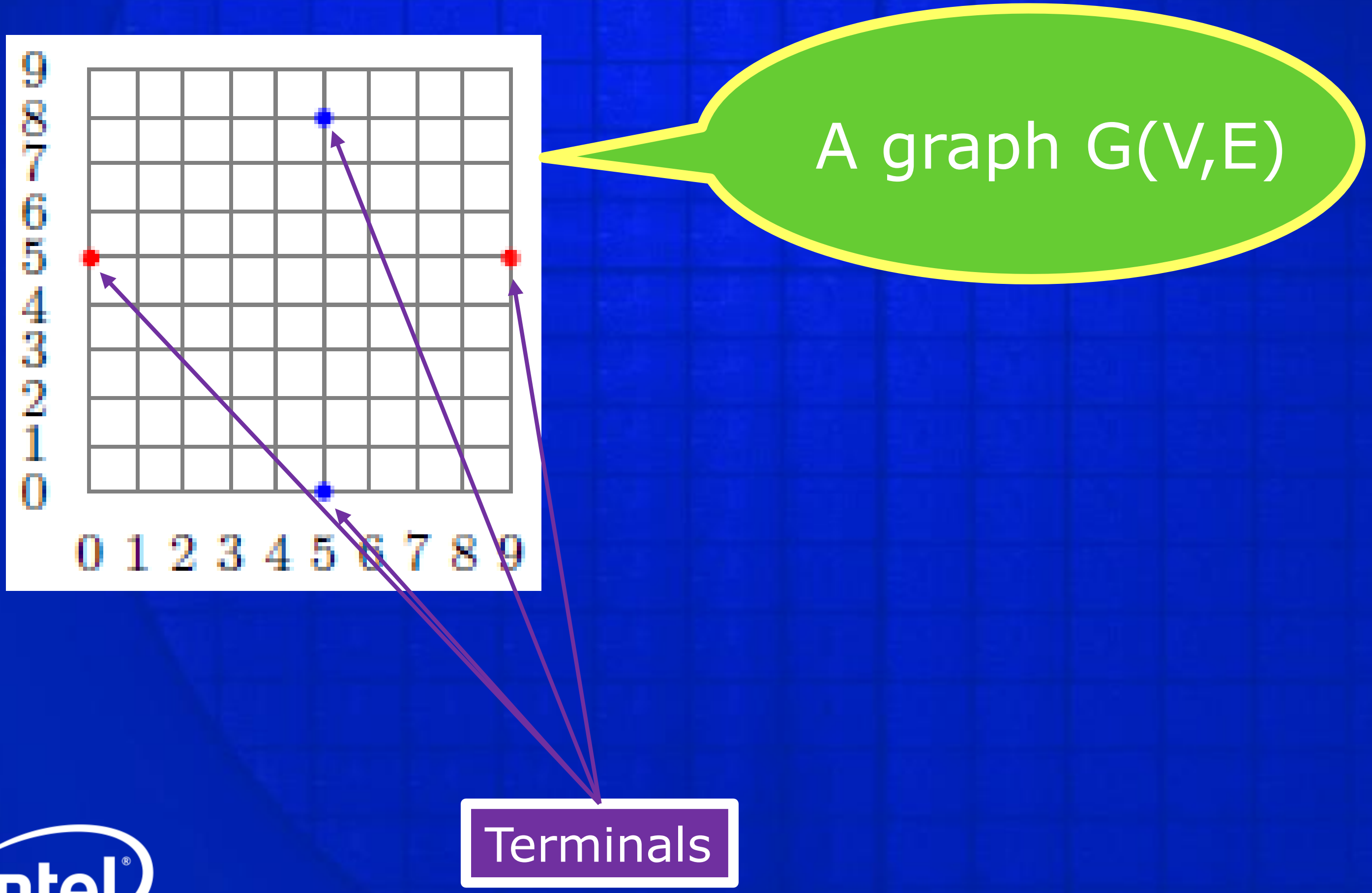


A graph G(V,E)

intel Leap ahead™
Design and Technology Solutions

Routing: Input
(AKA Steiner Tree Packing Problem)

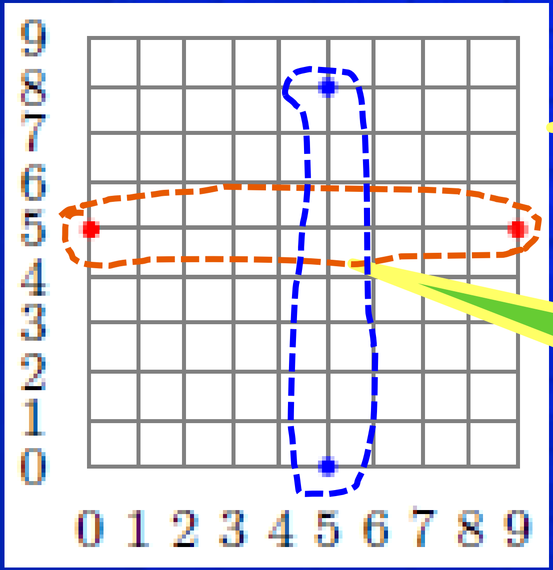
Input



intel Leap ahead™
Design and Technology Solutions

Routing: Input
(AKA Steiner Tree Packing Problem)

Input



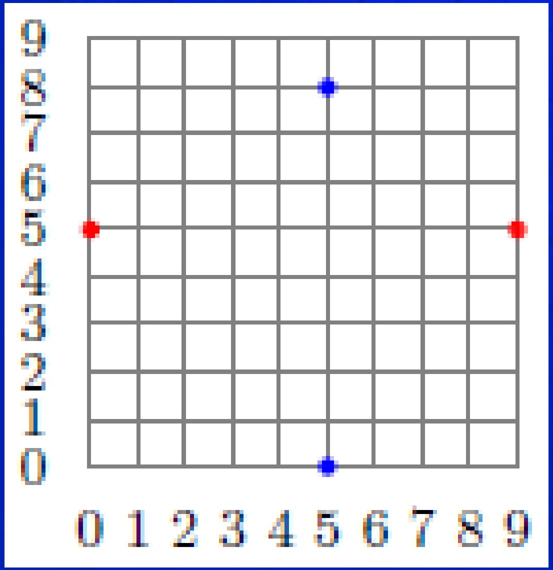
A graph G(V,E)

Disjoint Nets $N_i \subseteq V$

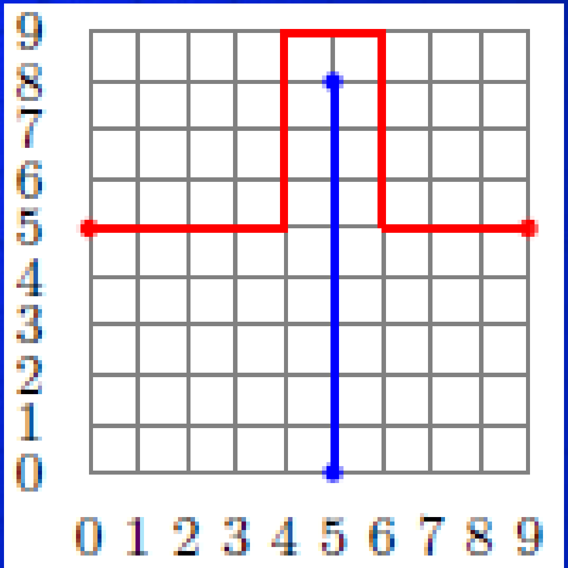
intel Leap ahead™
Design and Technology Solutions

Routing: Output

Input



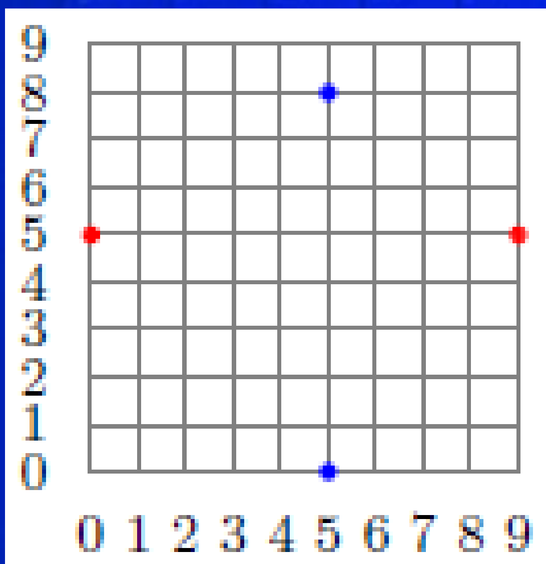
Output



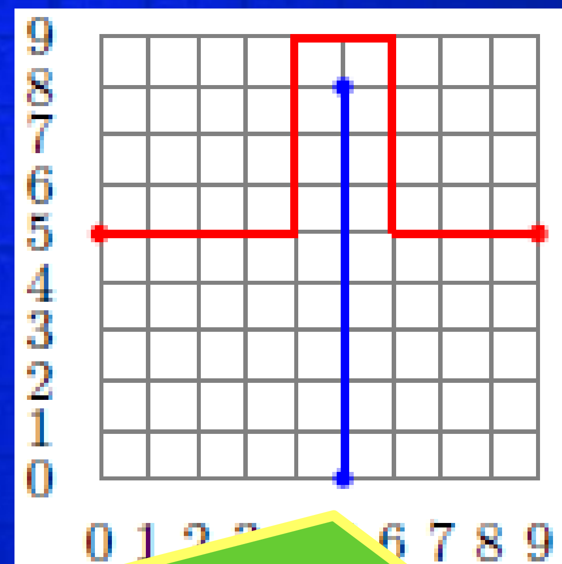
intel Leap ahead™
Design and Technology Solutions

Routing: Output

Input



Output



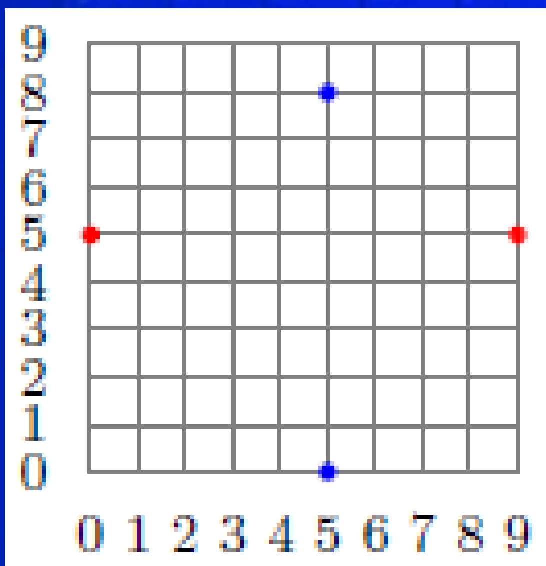
1. Each net is spanned by a tree, called the **net routing**
2. Net routings can’t intersect
3. Optimization: minimize the total routing length

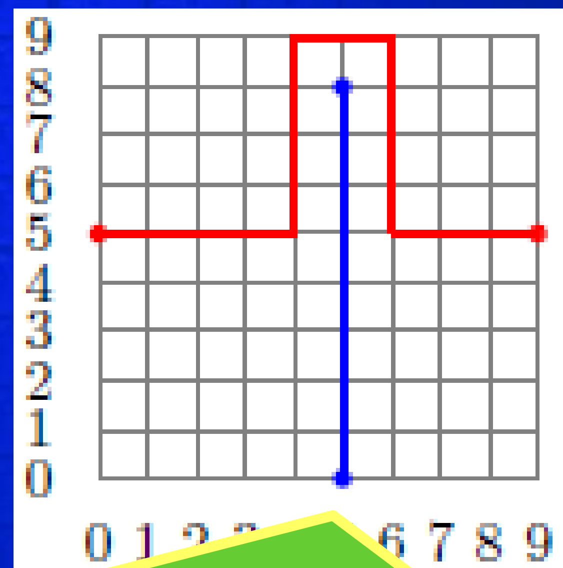
intel Leap ahead™

Design and Technology Solutions

Routing: Output

It is **NP-hard** to find:
1. **Shortest** solution for **one** multi-terminal net (Steiner tree problem)
2. **Any** solution for **many** multi-terminal nets





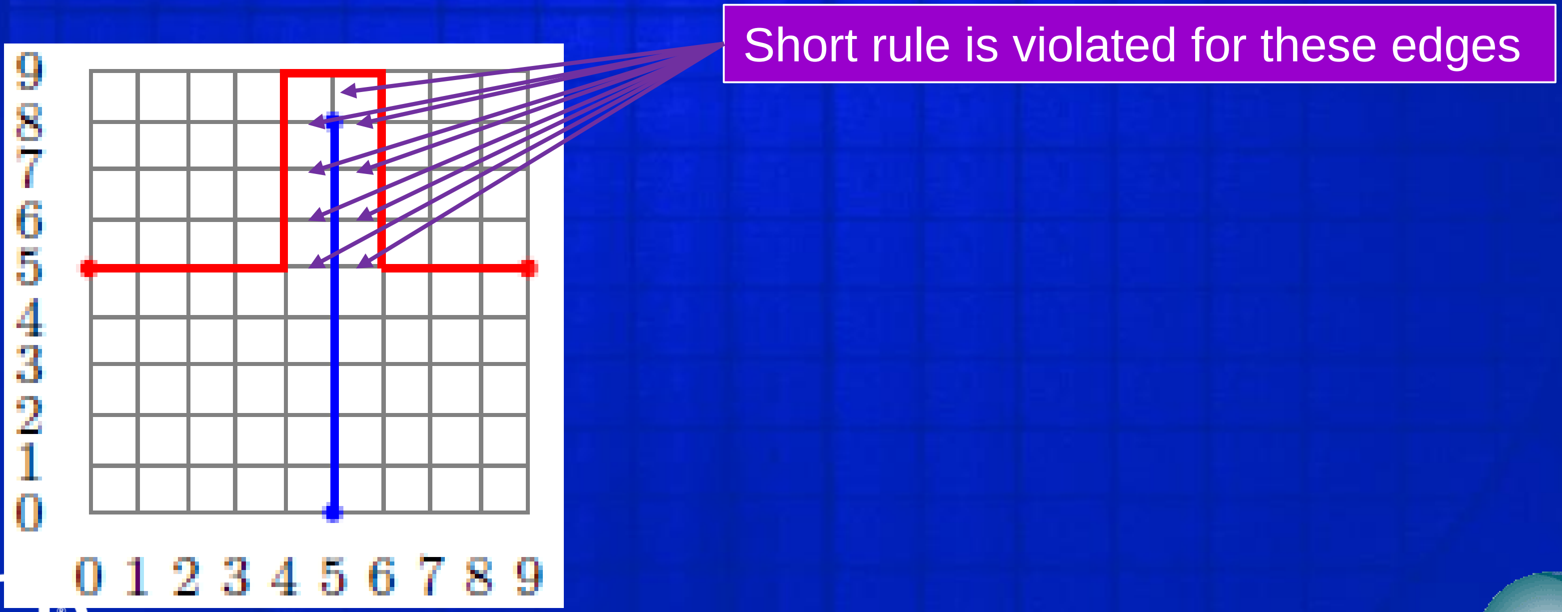
1. Each net is spanned by a tree, called the **net routing**
2. Net routings can't intersect
3. Optimization: minimize the total routing length

Design Rules

- Routing is to satisfy design rules
  - Originating in the manufacturing requirement

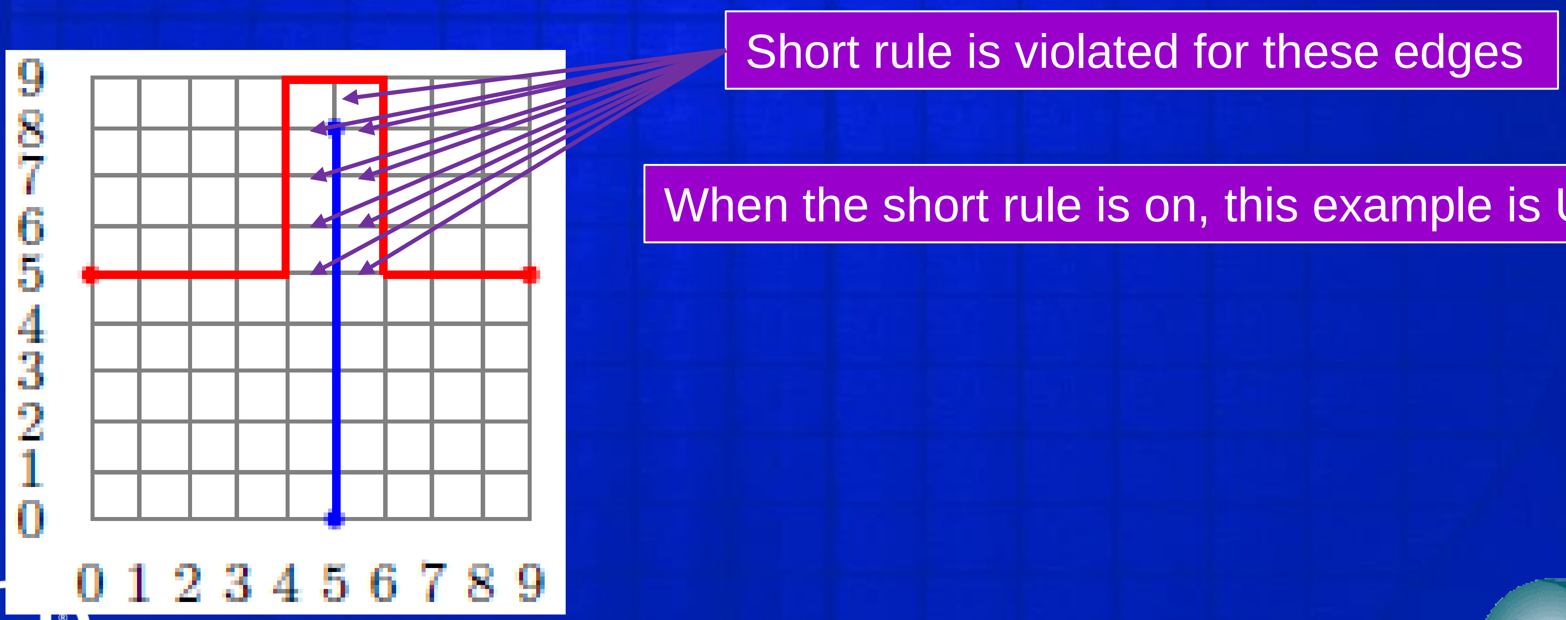
Design Rules

- Routing is to satisfy design rules
  - Originating in the manufacturing requirement
- Example "short" rule:
  - The 2 vertices of any edge can't belong to two distinct net routings



Design Rules

- Routing is to satisfy design rules
  - Originating in the manufacturing requirement
- Example "short" rule:
  - The 2 vertices of any edge can't belong to two distinct net routings



When the short rule is on, this example is UNSAT

intel Leap ahead™
Design and Technology Solutions

Industrial Approach: Rip-Up and Reroute

- Nets are routed one-by-one
  - Using A*
    - s-t shortest-path given costs' under-approximation
    - A*≡Dijkstra if no costs' under-approximation is provided
  - Trying to heuristically obey design rules
- Violations are allowed, hence the initial solution might be problematic
  - Net routings might intersect
  - Design rules might be violated
- Clean-up is applied
  - Rip-up: problematic net routings are removed
  - Reroute: un-routed nets are attempted again

intel Leap ahead™

Design and Technology Solutions

The Problem with the Current Solution

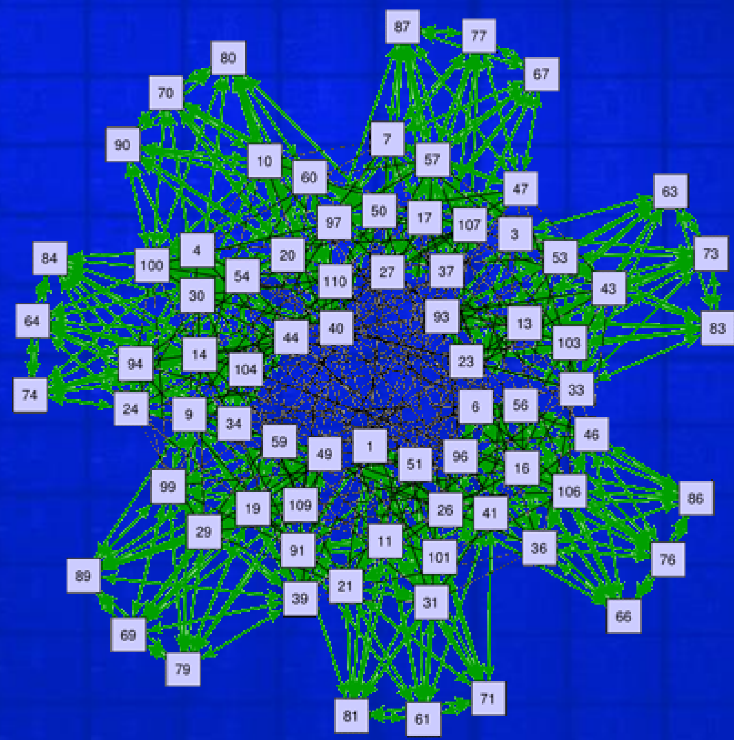
- Design rule violations persist
  - Manual clean-up is carried out

Some violations still persist

Time-to-market is impacted

Potential Solution

Constraint Solving

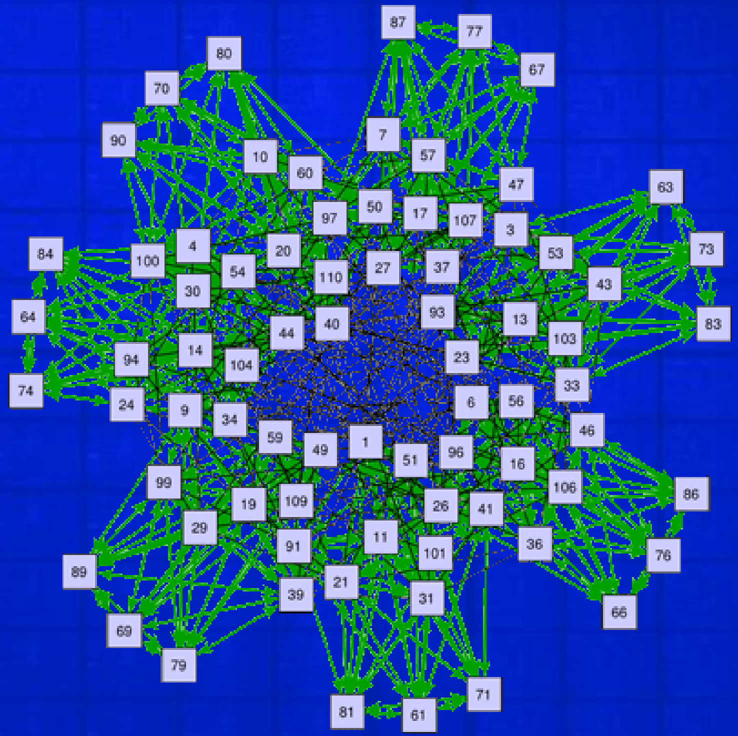


intel Leap ahead™

Design and Technology Solutions

Potential Solution

Constraint Solving



Next: formalizing Routing under Constraints

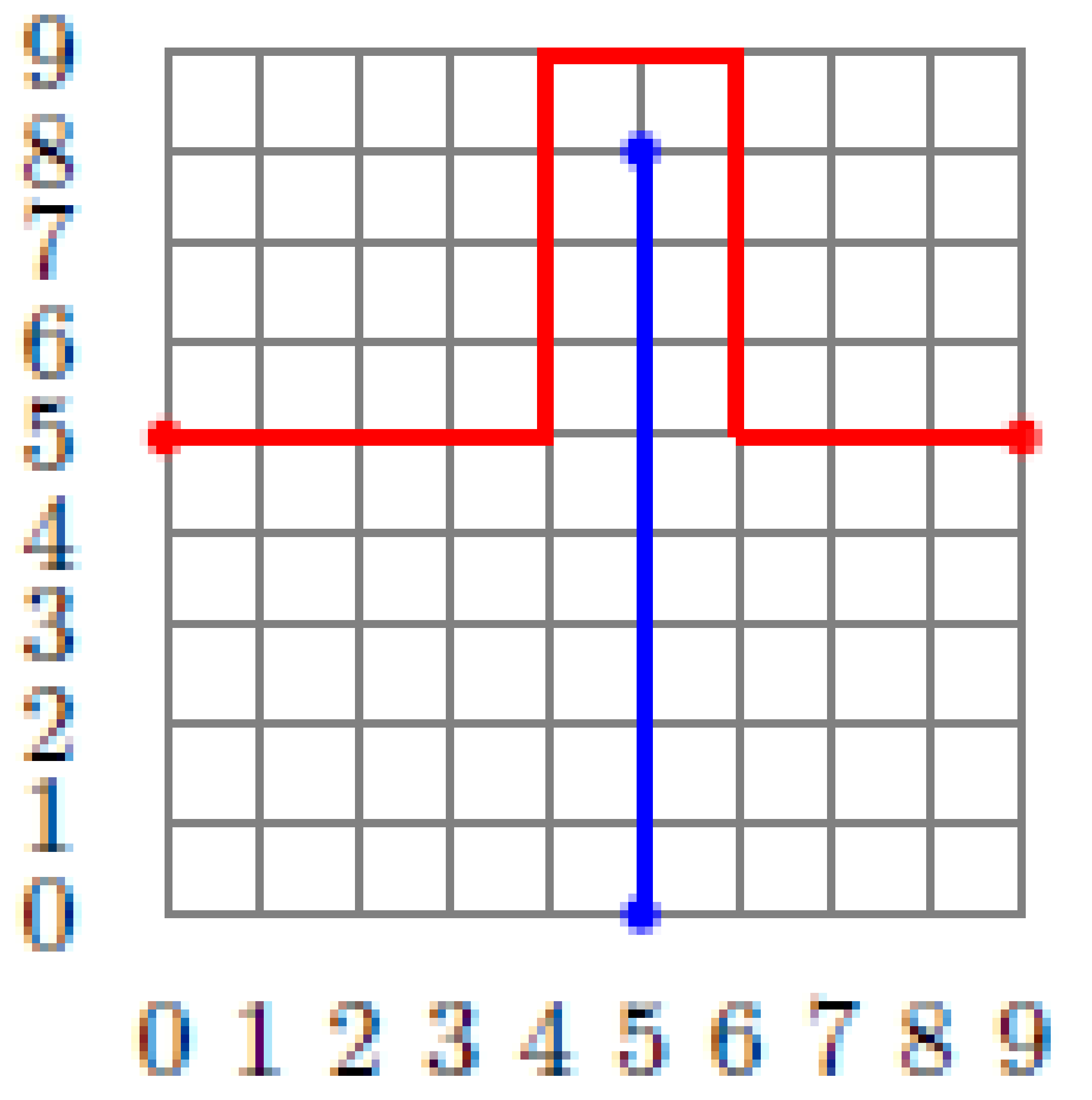
Leap ahead™

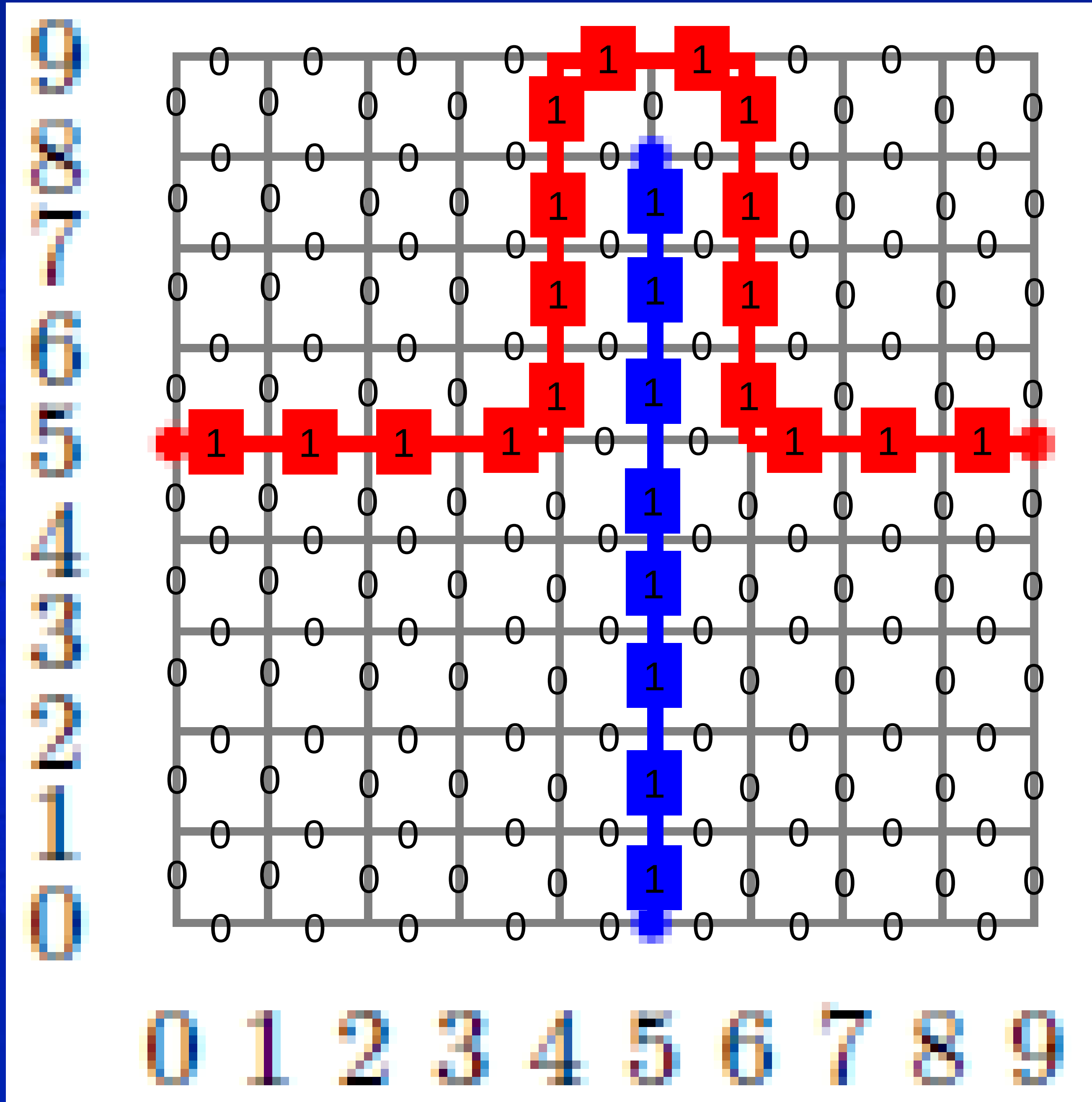
Design and Technology Solutions

Routing Induces Assignment

Edge variables

Bool **e**: edge activity


9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9

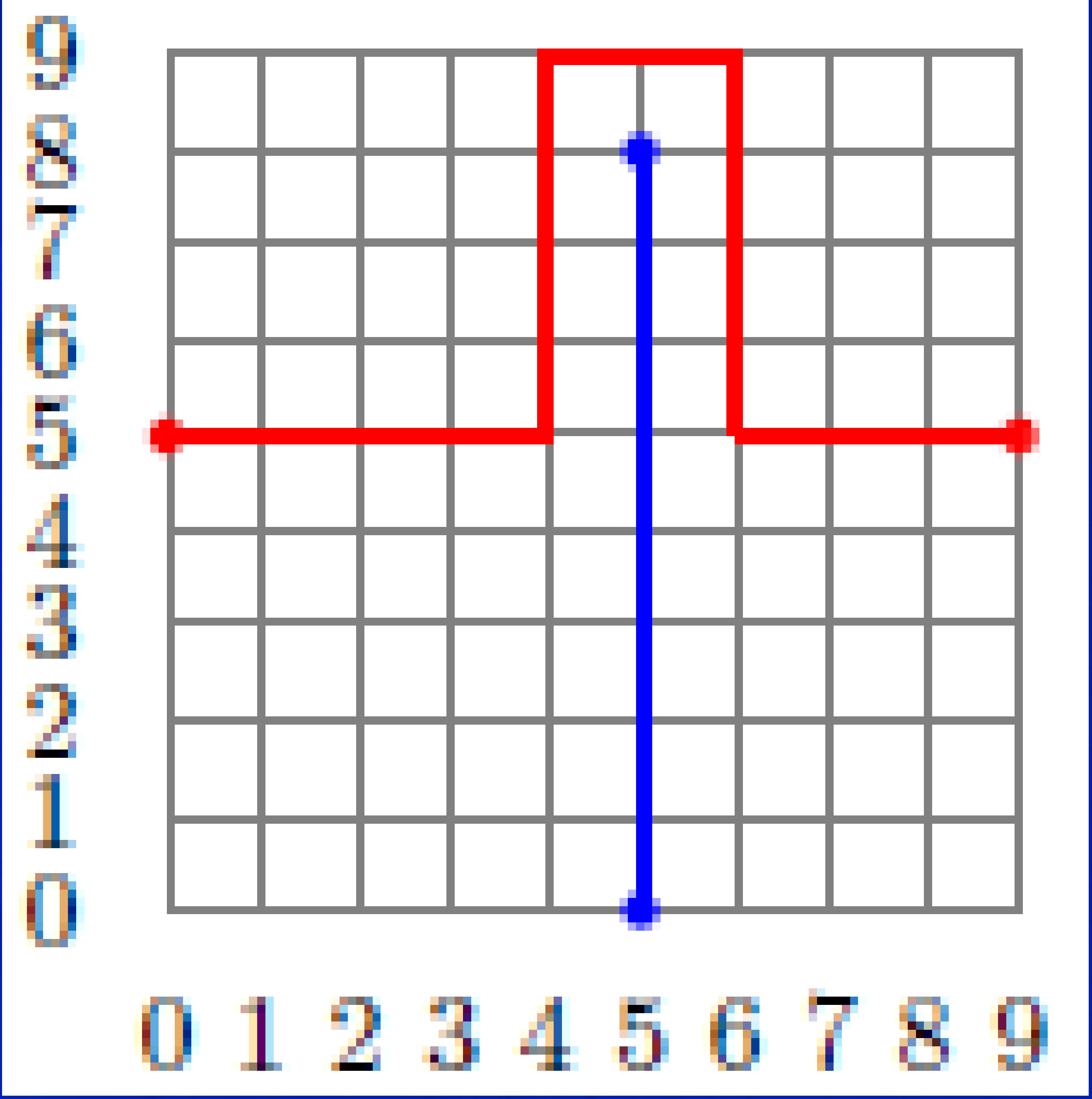

9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9

Routing Induces Assignment

Vertex variables

Bool **v**: activity status
Bit-vector **n**: net id
(∅ for inactive vertices)

intel Leap ahead™
Design and Technology Solutions
14



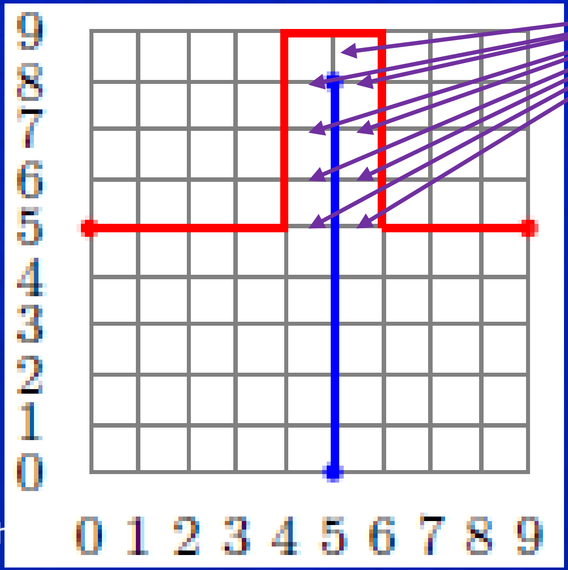
intel® Leap ahead™

Design and Technology Solutions

9	0,∅	0,∅	0,∅	0,∅	1,0	1,0	1,0	0,∅	0,∅	0,∅
8	0,∅	0,∅	0,∅	0,∅	1,0	1,1	1,0	0,∅	0,∅	0,∅
7	0,∅	0,∅	0,∅	0,∅	1,0	1,1	1,0	0,∅	0,∅	0,∅
6	0,∅	0,∅	0,∅	0,∅	1,0	1,1	1,0	0,∅	0,∅	0,∅
5	1,0	1,0	1,0	1,0	1,0	1,1	1,0	1,0	1,0	1,0
4	0,∅	0,∅	0,∅	0,∅	0,∅	1,1	0,∅	0,∅	0,∅	0,∅
3	0,∅	0,∅	0,∅	0,∅	0,∅	1,1	0,∅	0,∅	0,∅	0,∅
2	0,∅	0,∅	0,∅	0,∅	0,∅	1,1	0,∅	0,∅	0,∅	0,∅
1	0,∅	0,∅	0,∅	0,∅	0,∅	1,1	0,∅	0,∅	0,∅	0,∅
0	0,∅	0,∅	0,∅	0,∅	0,∅	1,1	0,∅	0,∅	0,∅	0,∅
	0	1	2	3	4	5	6	7	8	9

Modeling Routing under Constraints

- Design rules can be easily expressed in BV logic
  - Variables:
    - Edge & vertex activities
    - Vertex nids
    - Any auxiliary variables
- "Short" rule example
  - For every edge e=(v,u): $\neg v \vee \neg u \vee nid(v)=nid(u)$



Short rule is violated for these edges

intel Leap ahead

Design and Technology Solutions

Routing under Constraints (RUC): Problem Formulation

Routing under Constraints (RUC): Problem Formulation

Leap ahead™

Design and Technology Solutions

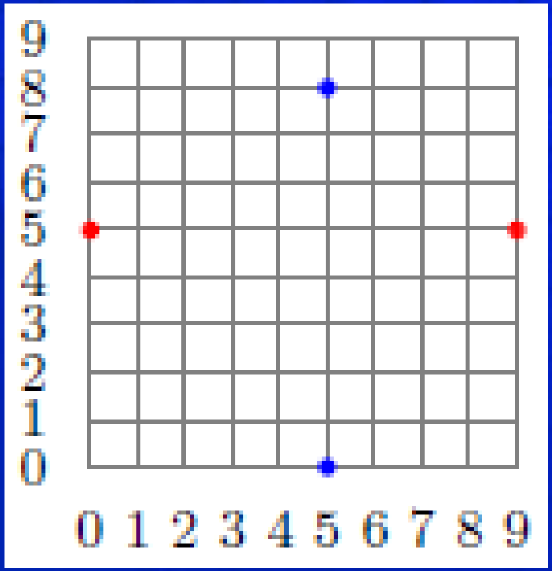
Routing under Constraints (RUC): Problem Formulation

Input

Routing under Constraints (RUC): Problem Formulation

Input

1. Graph G(V,E)
2. Disjoint Nets $N_i \subseteq V$

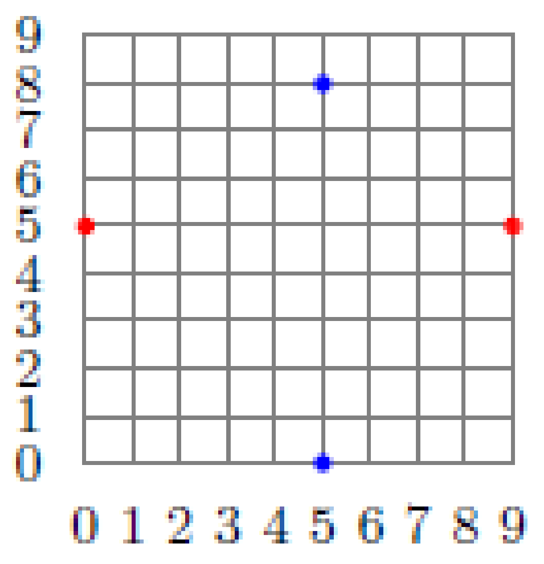


Design and Technology Solutions

Routing under Constraints (RUC): Problem Formulation

Input

1. Graph G(V,E)
2. Disjoint Nets $N_i \subseteq V$



A quantifier-free bit-vector formula $F(V \cup E \cup N \cup A)$

- V : vertex activity
- E : edge activity
- N : vertex net id
- A : any auxiliary variables

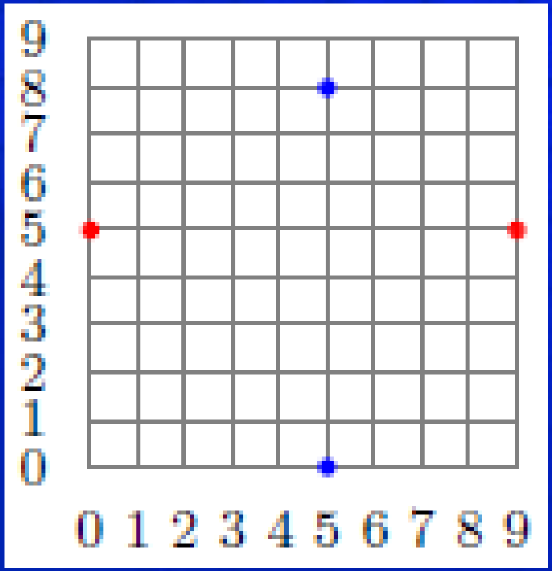
(represents the design rules)

Design and Technology Solutions

Routing under Constraints (RUC): Problem Formulation

Input

1. Graph G(V,E)
2. Disjoint Nets $N_i \subseteq V$



A quantifier-free bit-vector formula $F(V \cup E \cup N \cup A)$

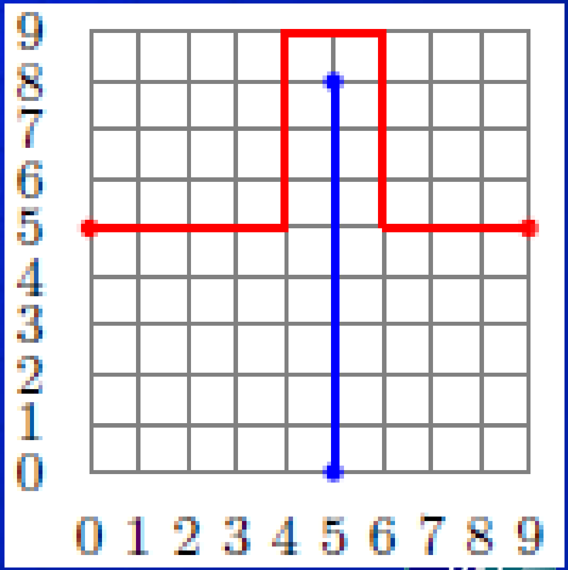
- V : vertex activity
- E : edge activity
- N : vertex net id
- A : any auxiliary variables

(represents the design rules)

Output: a model to F, which induces a routing:

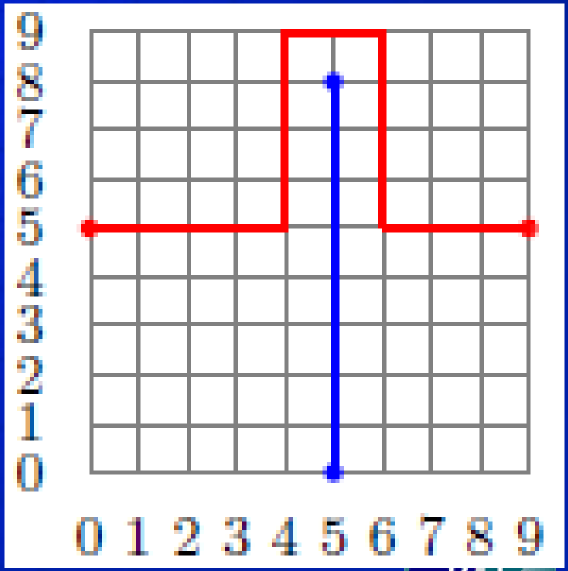
- e=(v,u) is active →
  - v and u are active, and
  - nid(v) = nid(u)
- For each net i: active vertices with nid i and active edges span the net's terminals
- *Optional optimization requirement*: the overall weight of active edges is as small as possible

Solving Attempt: Encoding into Bitvector Logic / SAT



Solving Attempt: Encoding into Bitvector Logic / SAT

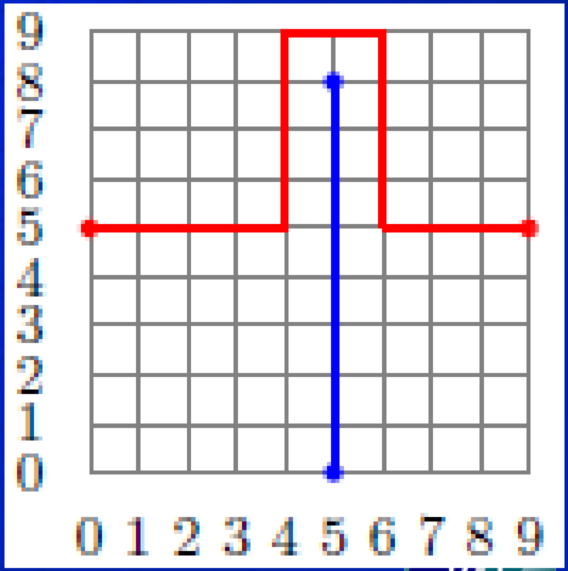
- For 2-terminal nets:



Design and Technology Solutions

Solving Attempt: Encoding into Bitvector Logic / SAT

- For 2-terminal nets:
  - e=(v,u) is active →
    - v and u are active, and
    - nid(v) = nid(u)

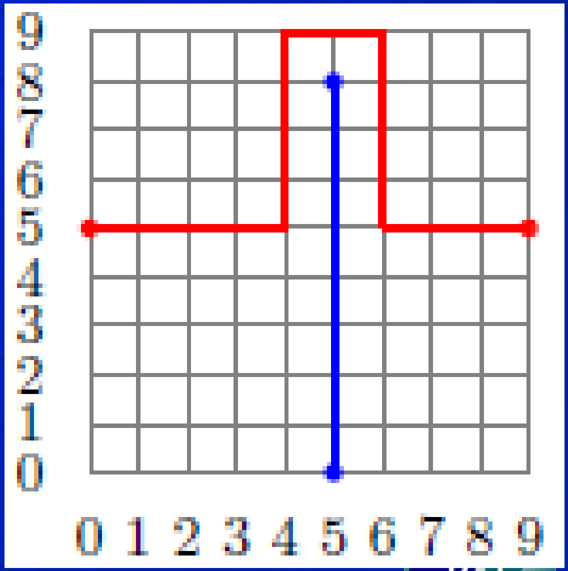


intel Leap ahead™

Design and Technology Solutions

Solving Attempt: Encoding into Bitvector Logic / SAT

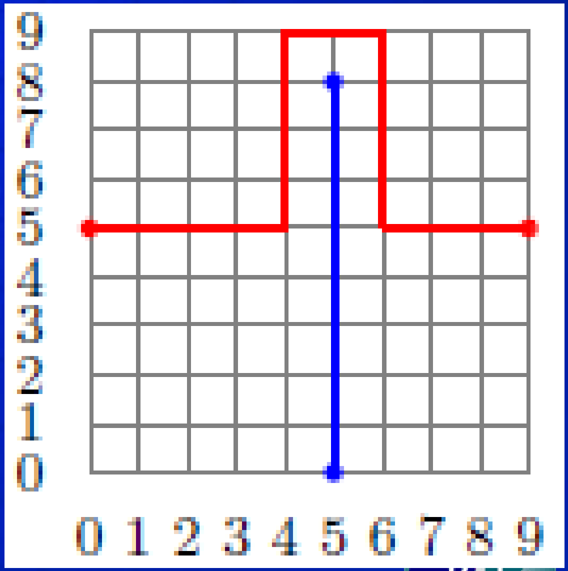
- For 2-terminal nets:
  - e=(v,u) is active →
    - v and u are active, and
    - nid(v) = nid(u)
  - A terminal has one active neighbor edge



Design and Technology Solutions

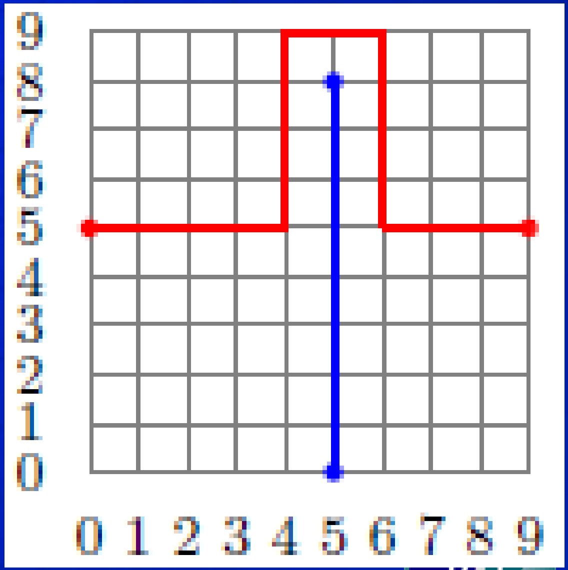
Solving Attempt: Encoding into Bitvector Logic / SAT

- For 2-terminal nets:
  - e=(v,u) is active →
    - v and u are active, and
    - nid(v) = nid(u)
  - A terminal has one active neighbor edge
  - An active non-terminal has two active neighbor edges



Solving Attempt: Encoding into Bitvector Logic / SAT

- For 2-terminal nets:
  - e=(v,u) is active →
    - v and u are active, and
    - nid(v) = nid(u)
  - A terminal has one active neighbor edge
  - An active non-terminal has two active neighbor edges
- For n-terminal nets:
  - Encode directed trees
    - Using edge directions



Design and Technology Solutions

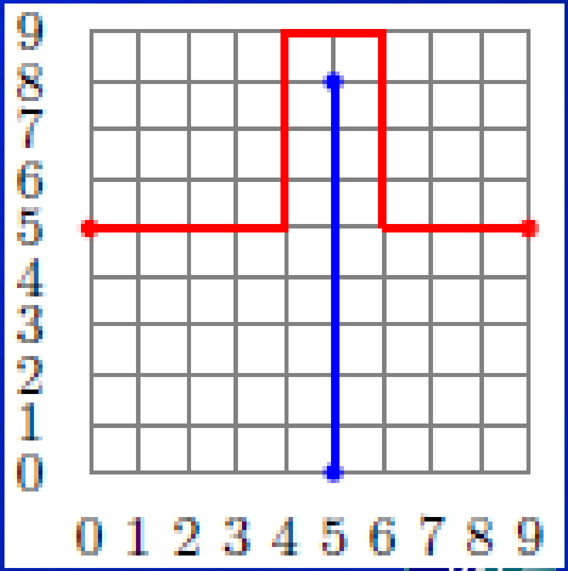
Solving Attempt: Encoding into Bitvector Logic / SAT

- For 2-terminal nets:
  - e=(v,u) is active →
  - A
  - A

eighbor edge

two active neighbor edges

- For
  - E



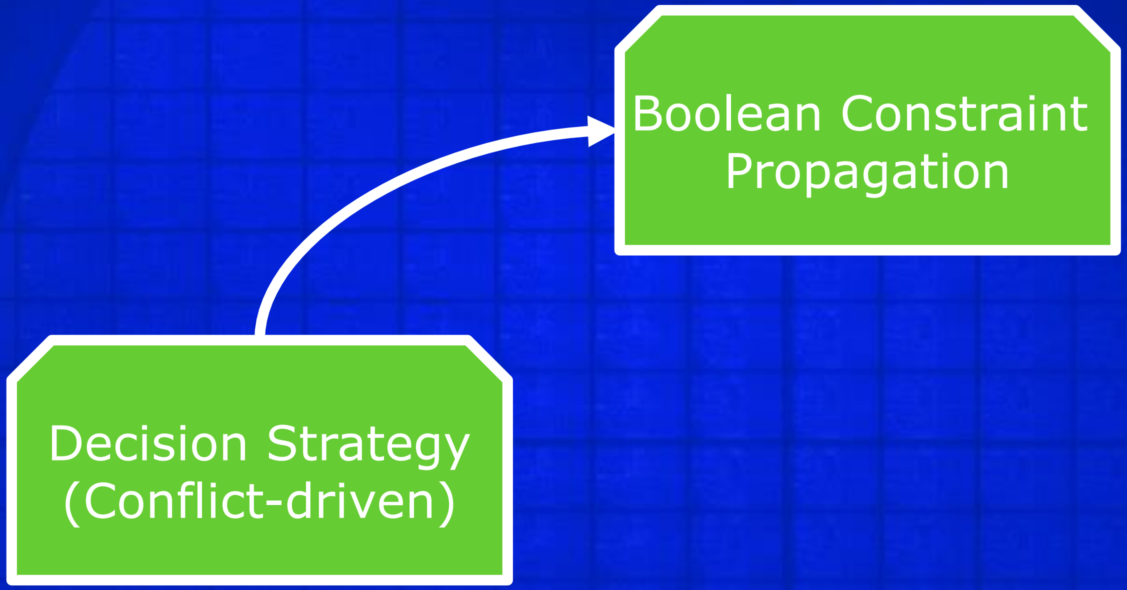
SAT Solver's Internals

SAT Solver's Internals

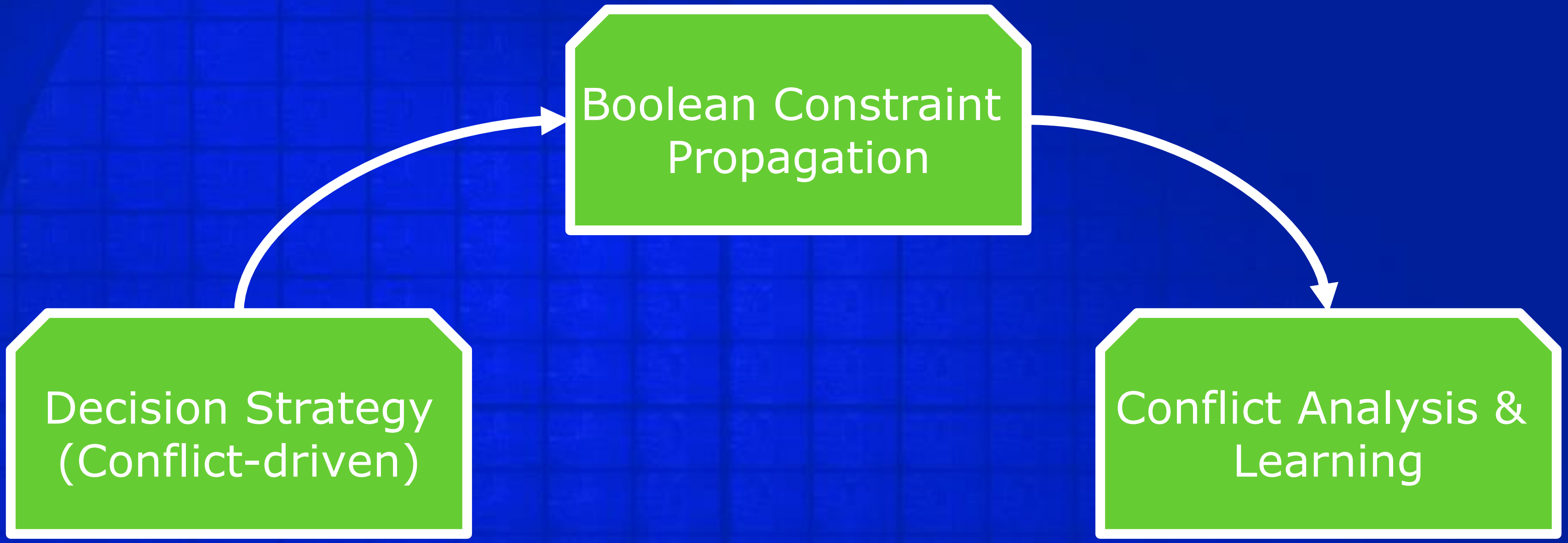
Decision Strategy
(Conflict-driven)

Design and
Technology
Solutions

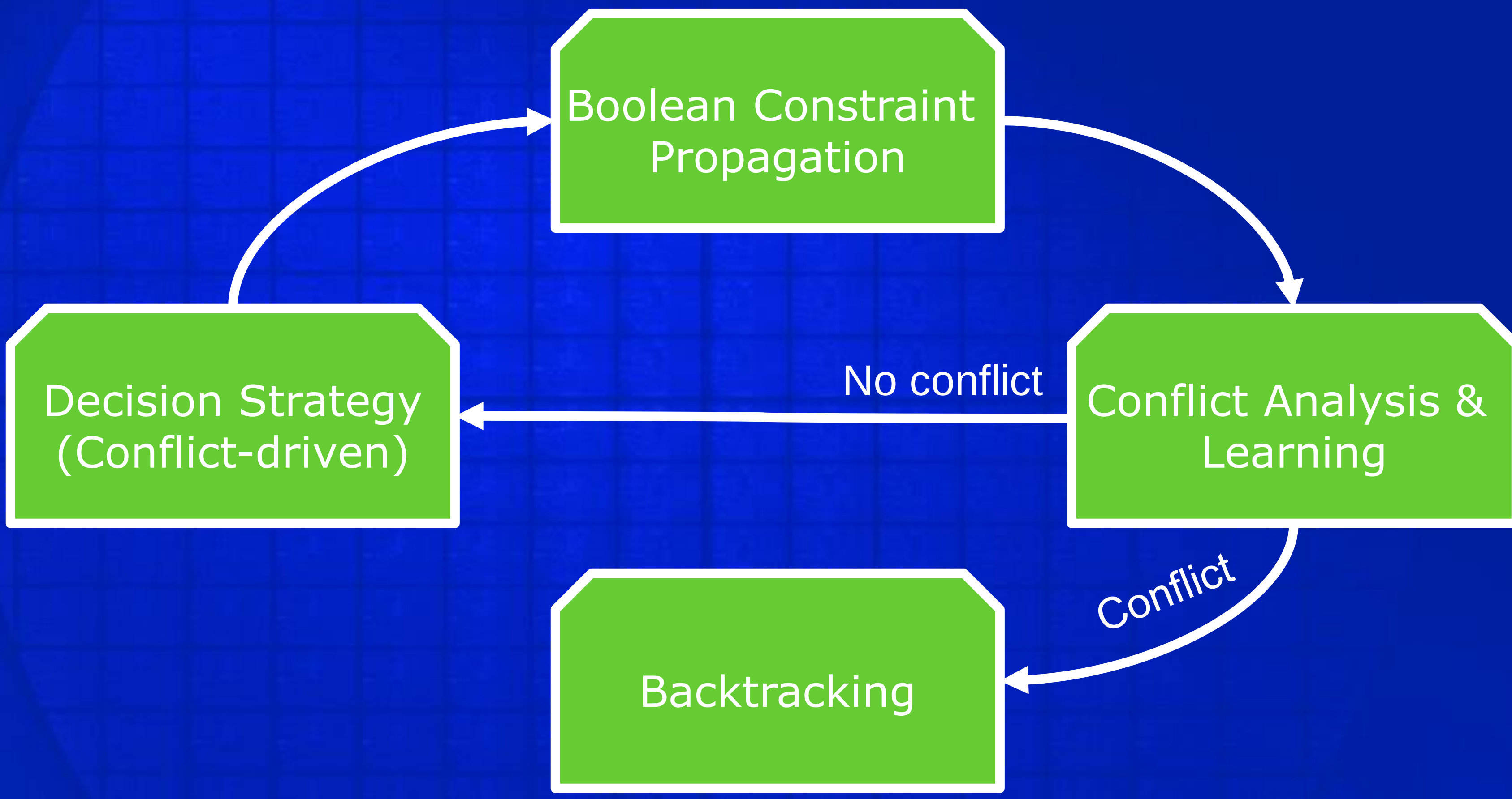
SAT Solver’s Internals



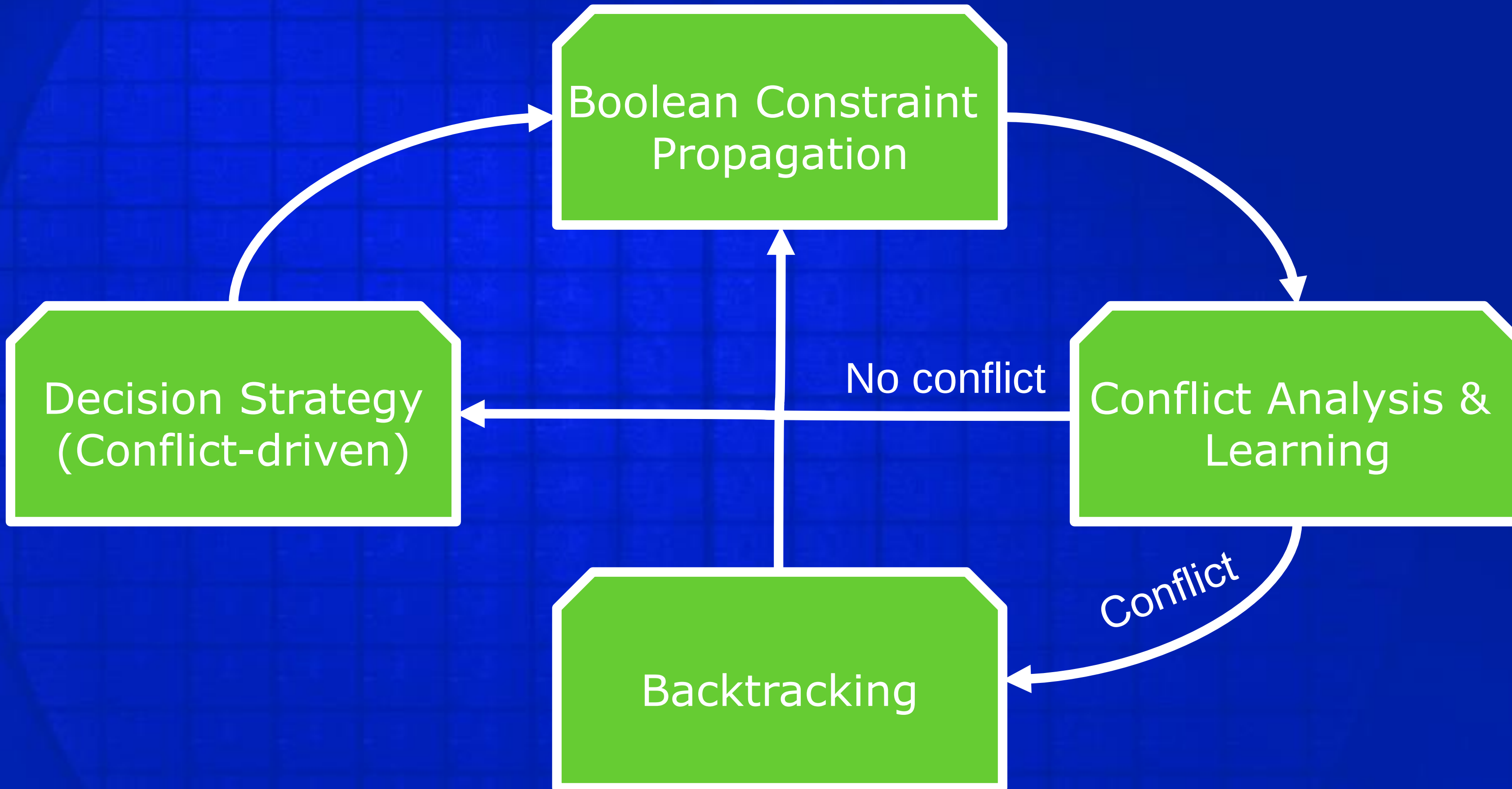
SAT Solver's Internals



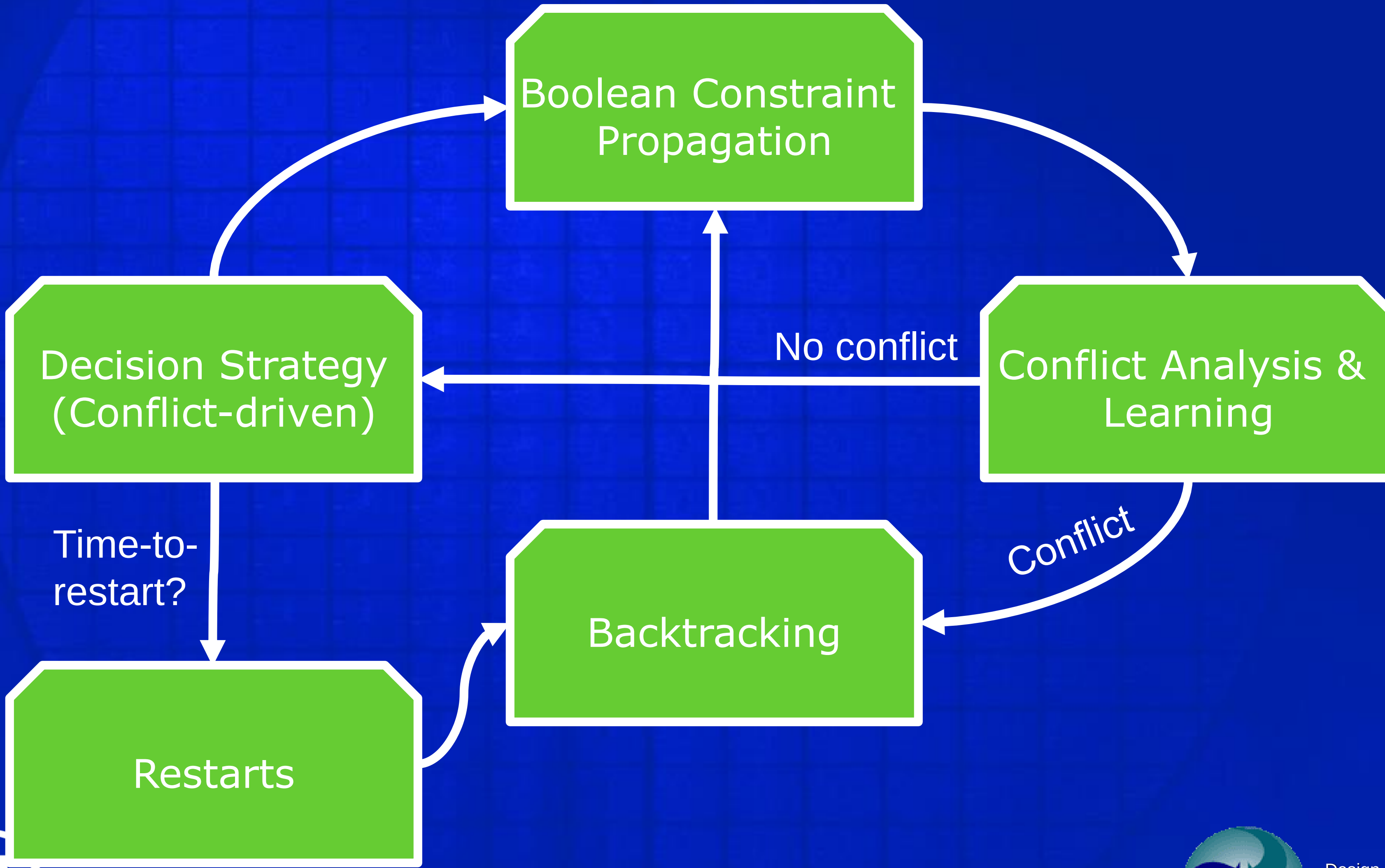
SAT Solver’s Internals



SAT Solver's Internals



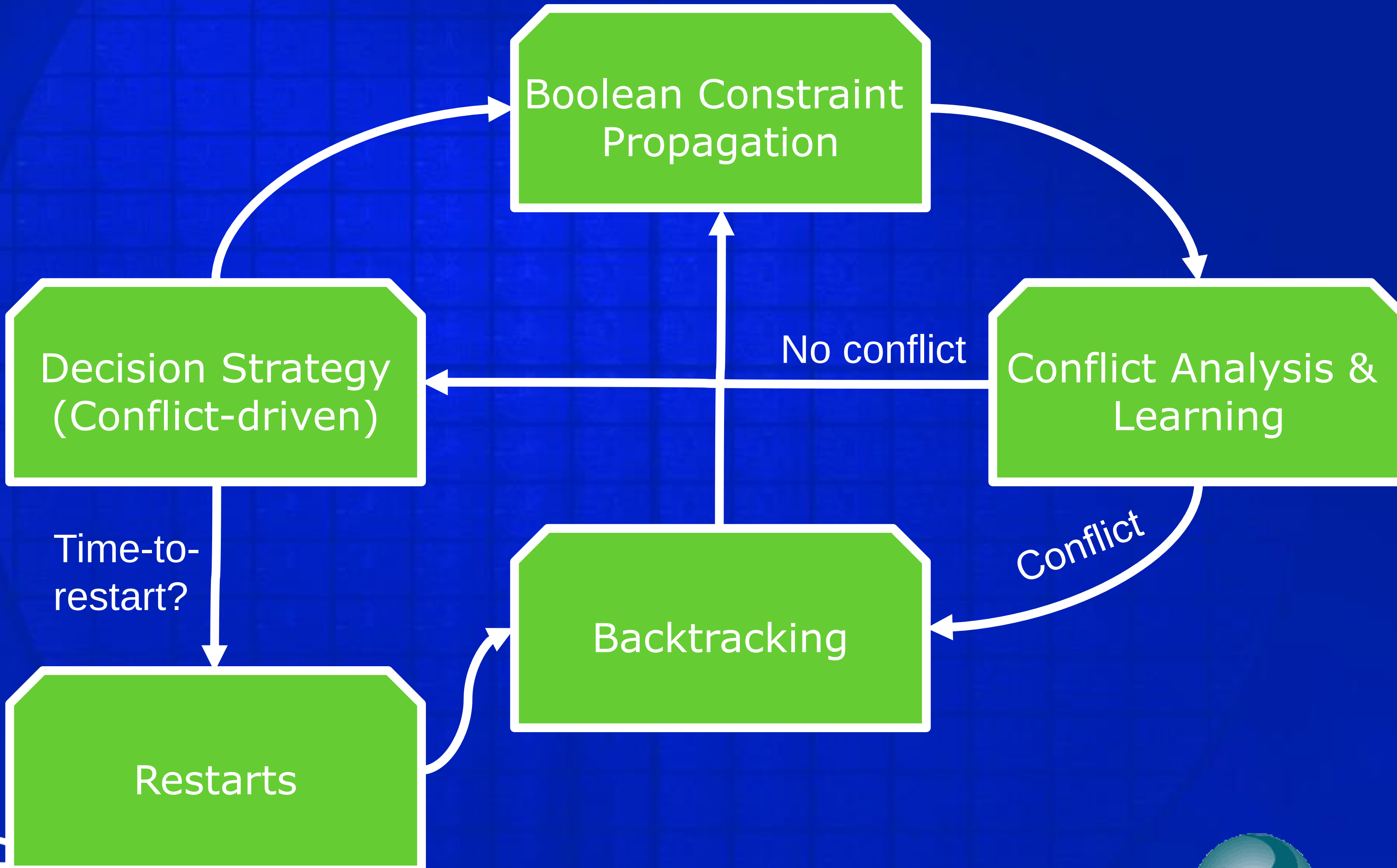
SAT Solver's Internals



intel Leap ahead™
Design and Technology Solutions

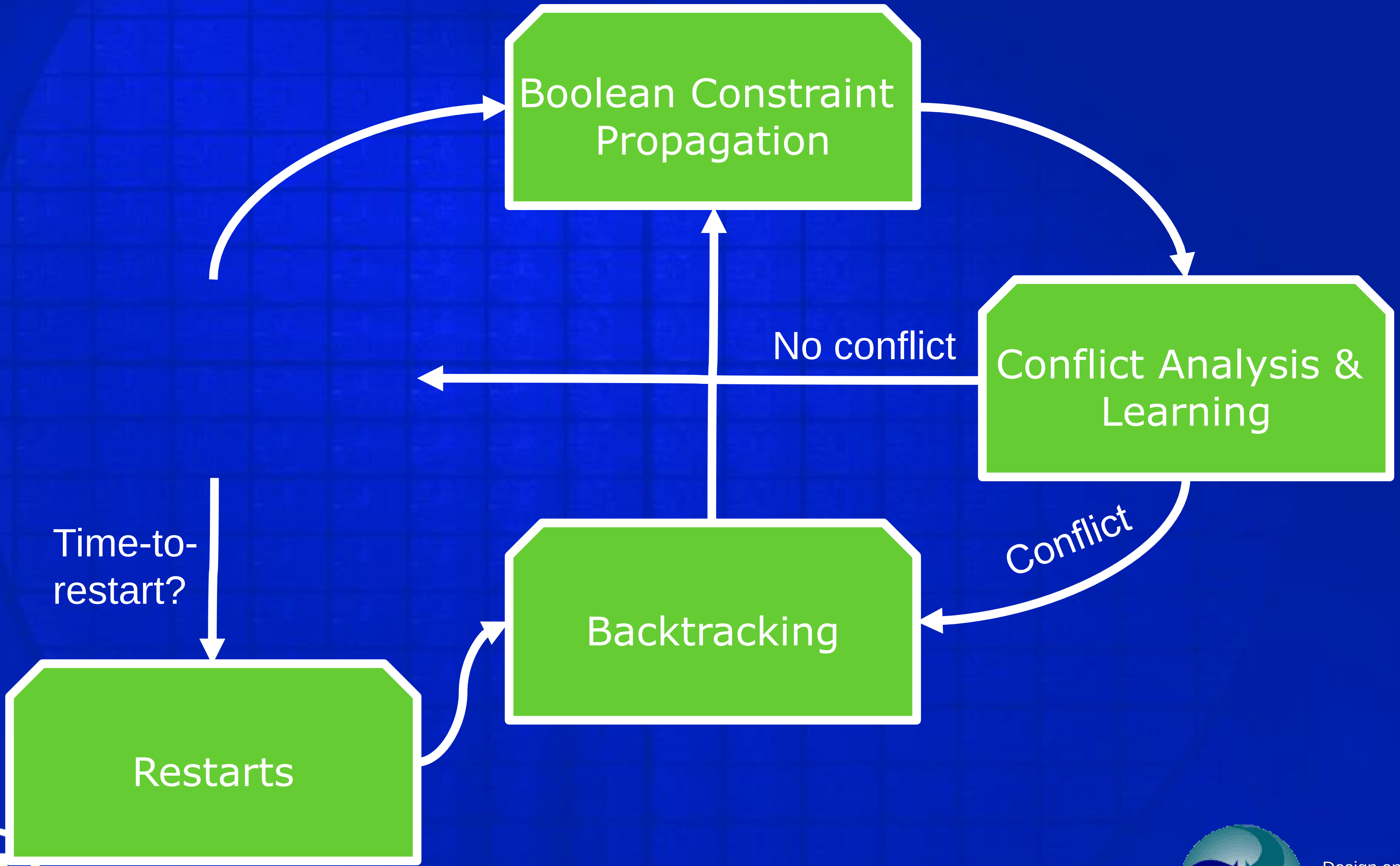
SAT → DRouter through **Surgery**

SAT → DRouter



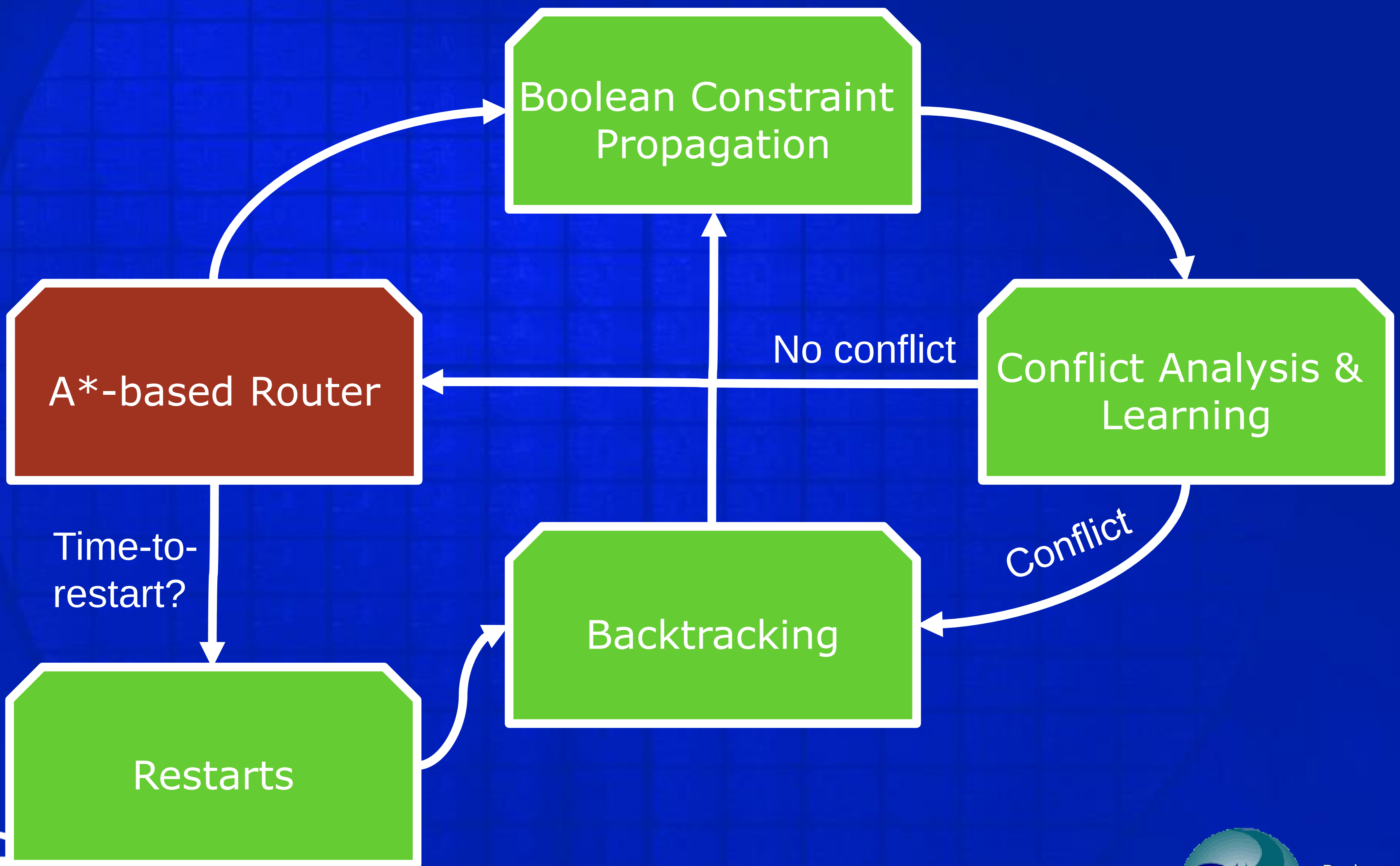
intel Leap ahead™
Design and Technology Solutions

SAT → DRouter



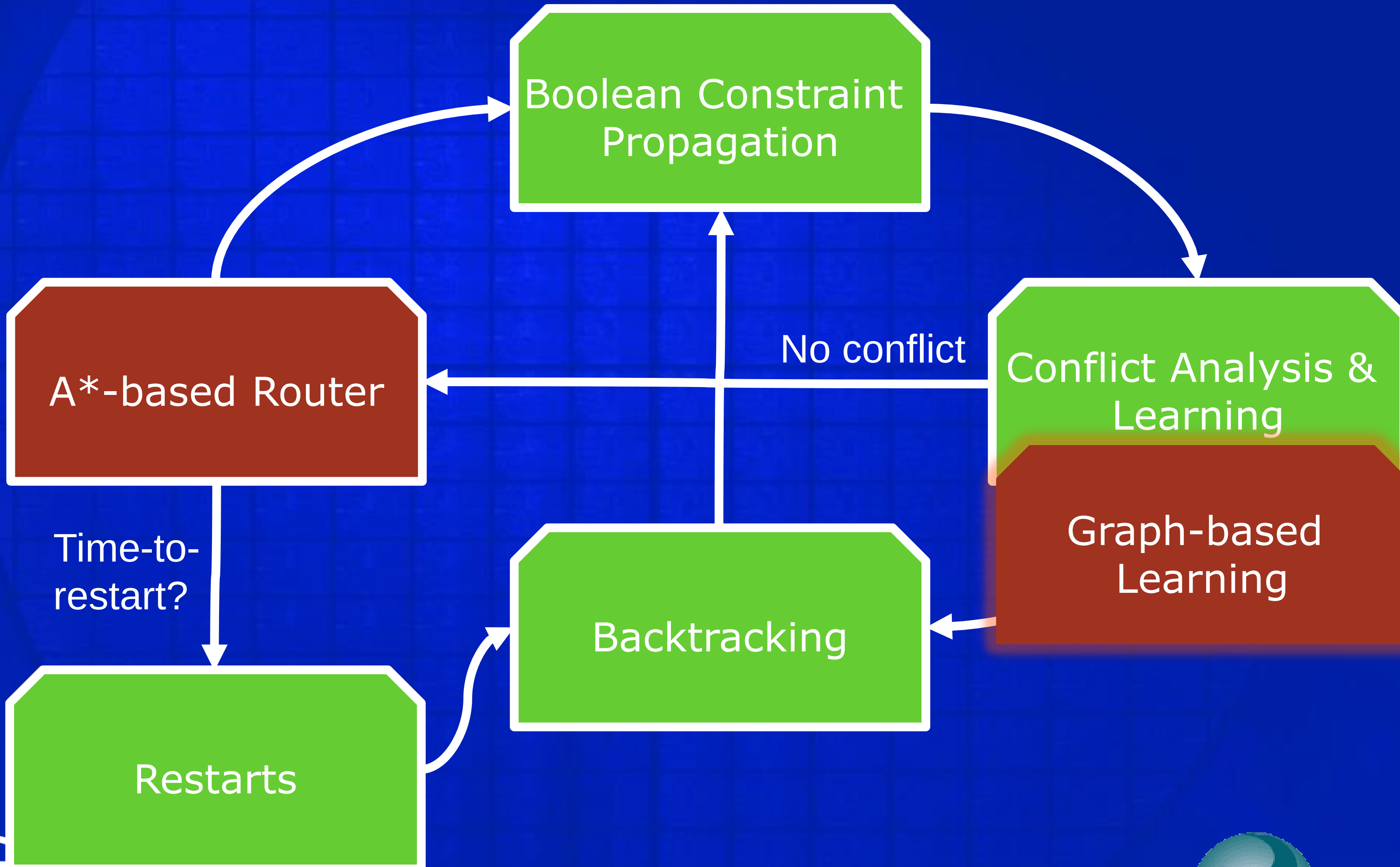
Boolean Constraint Propagation
No conflict
Conflict Analysis & Learning
Conflict
Backtracking
Time-to-restart?
Restarts

SAT → DRouter

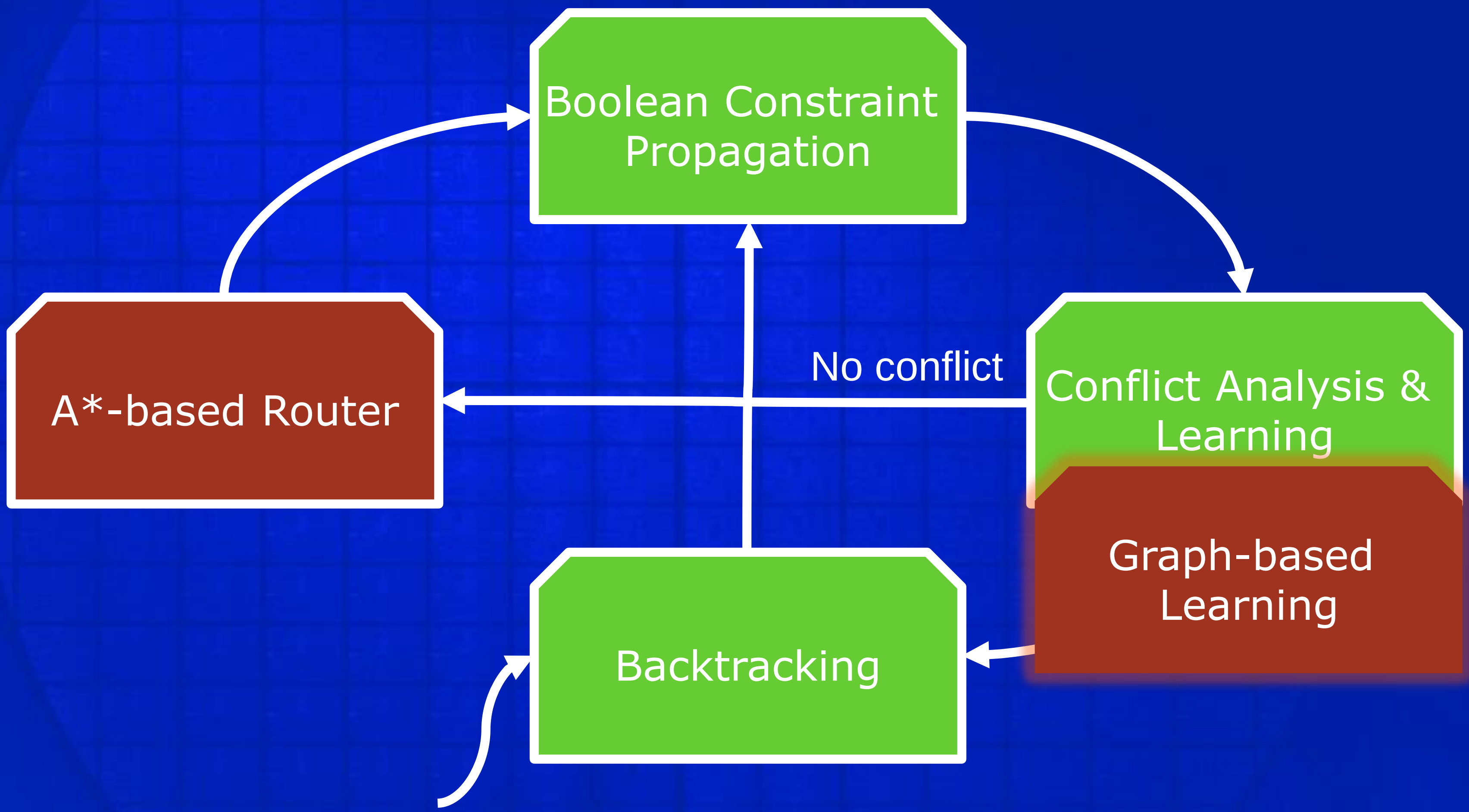


intel Leap ahead™
Design and Technology Solutions

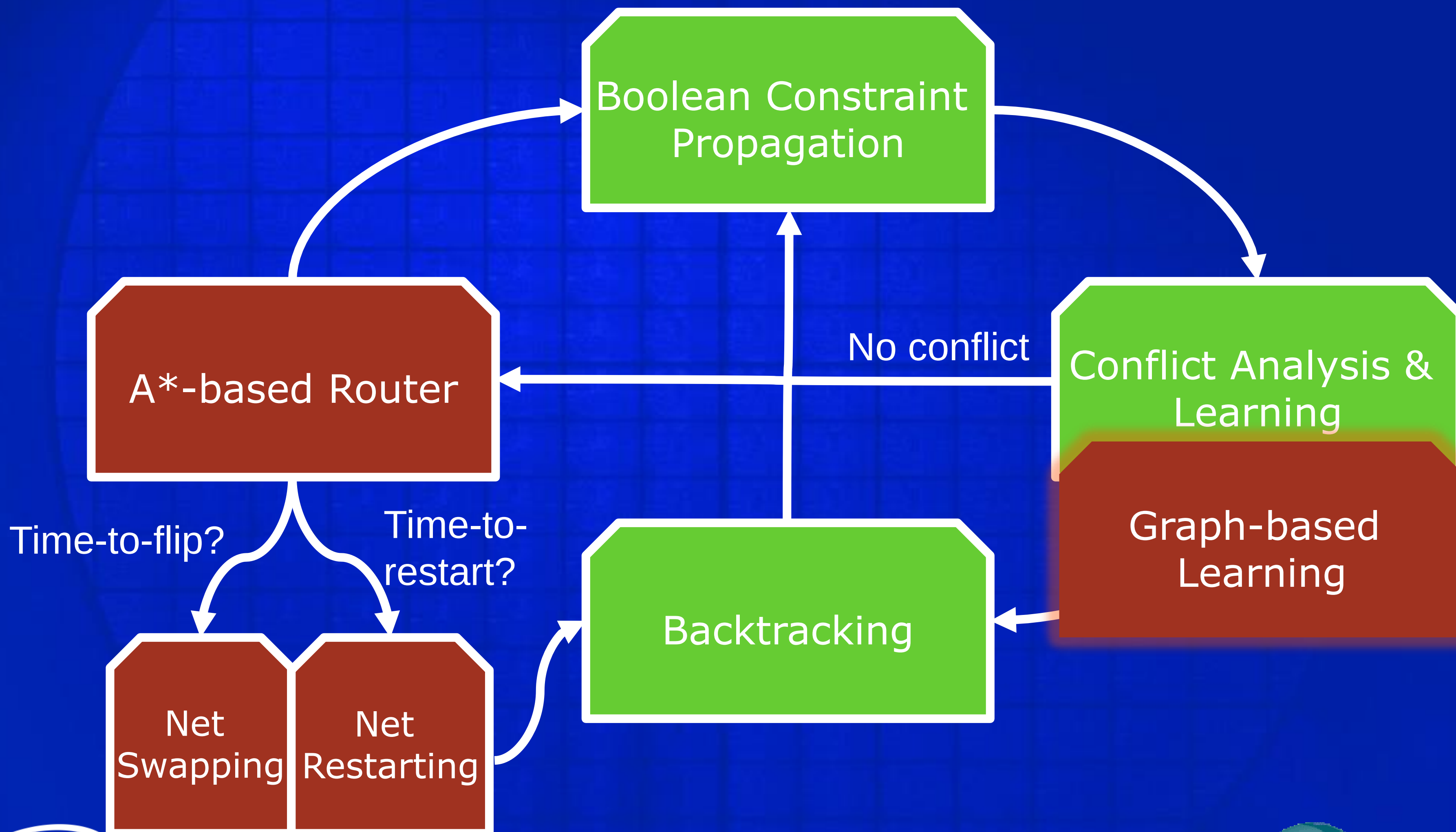
SAT → DRouter


Boolean Constraint Propagation
No conflict
A*-based Router
Conflict Analysis & Learning
Graph-based Learning
Time-to-restart?
Backtracking
Restarts

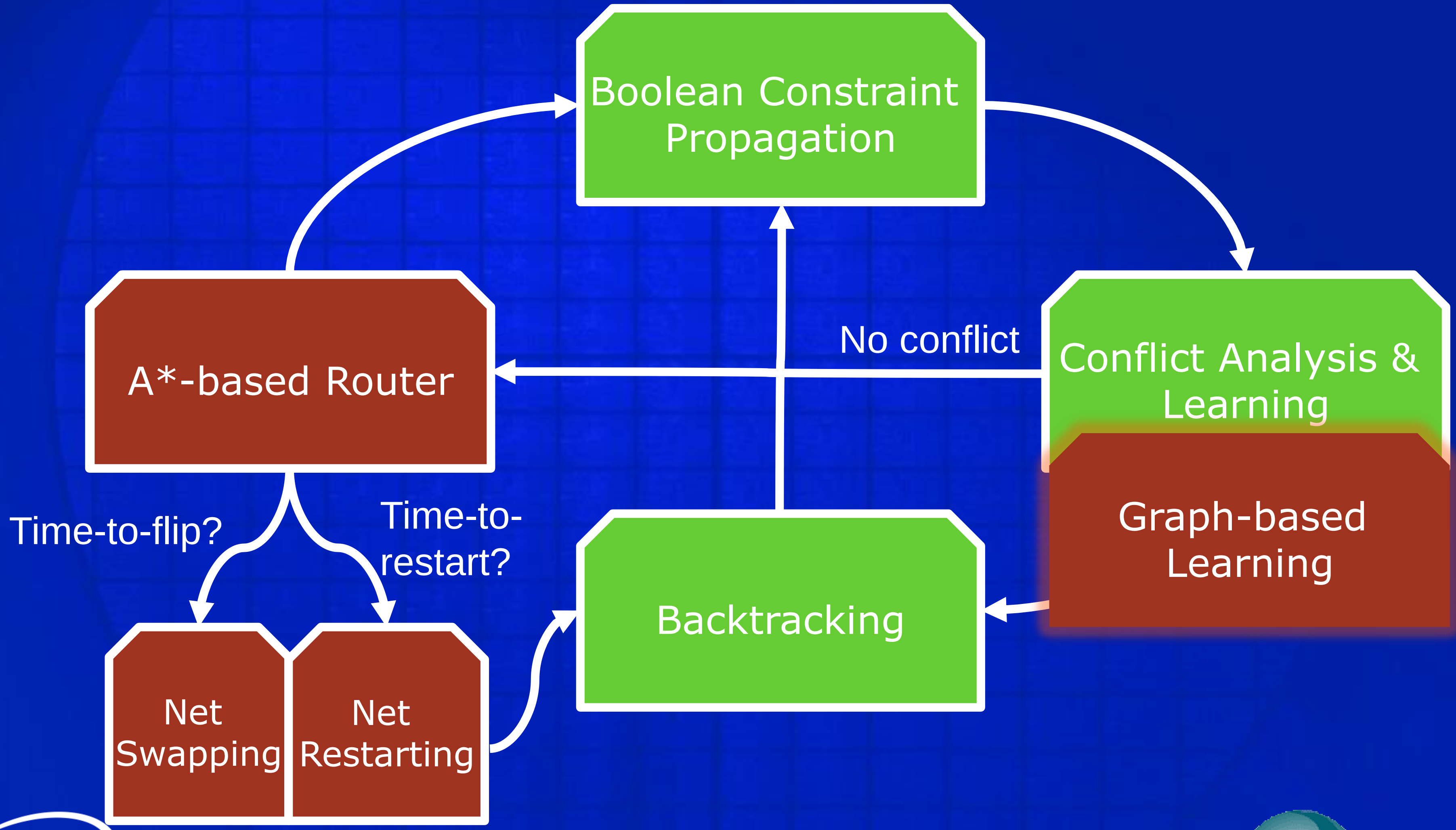
SAT → DRouter


Boolean Constraint Propagation
No conflict
A*-based Router
Conflict Analysis & Learning
Graph-based Learning
Backtracking

SAT → DRouter


Boolean Constraint Propagation
No conflict
A*-based Router
Conflict Analysis & Learning
Graph-based Learning
Time-to-flip?
Time-to-restart?
Backtracking
Net Swapping
Net Restarting

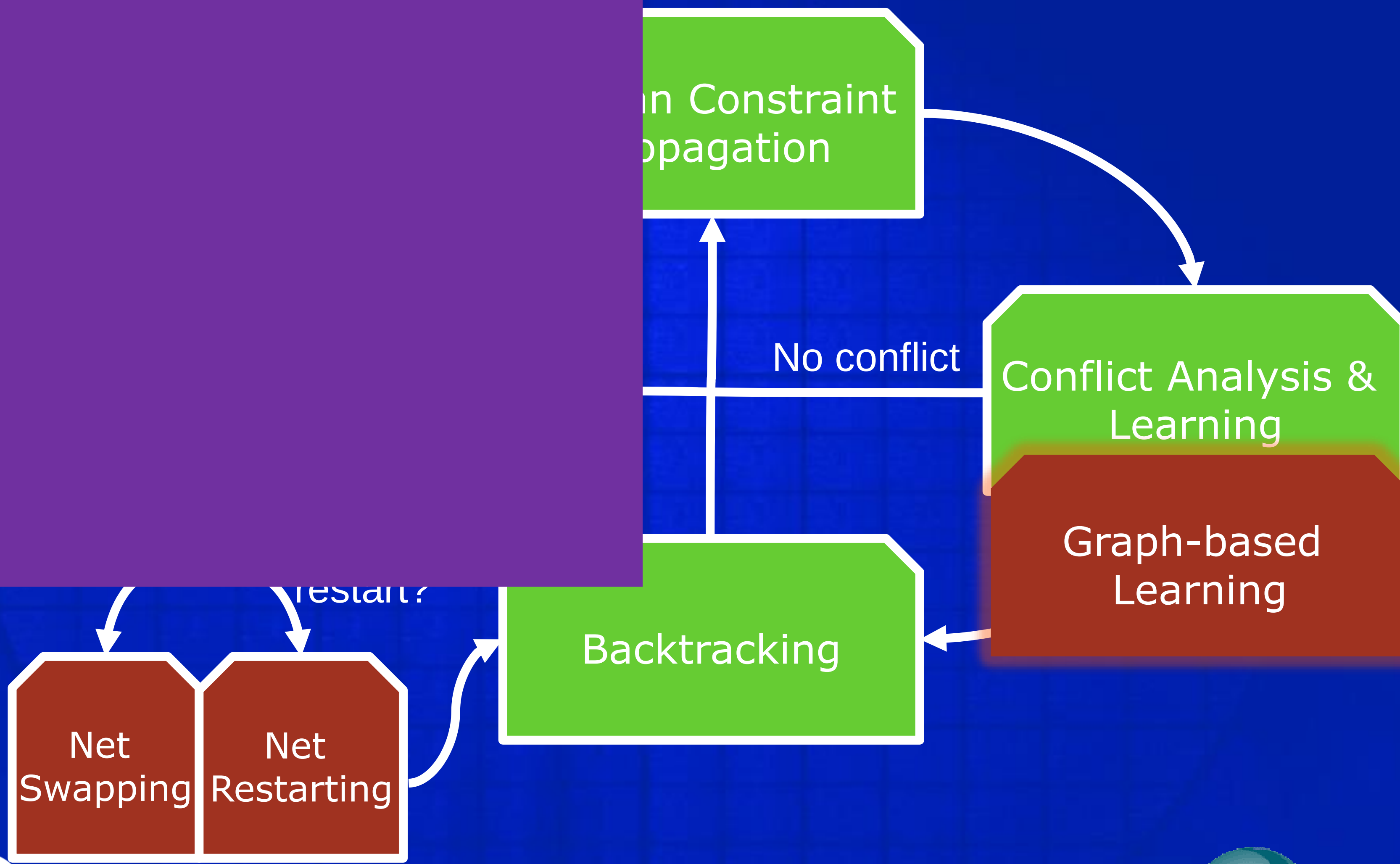
DRouter



intel Leap ahead™
Design and Technology Solutions

Router

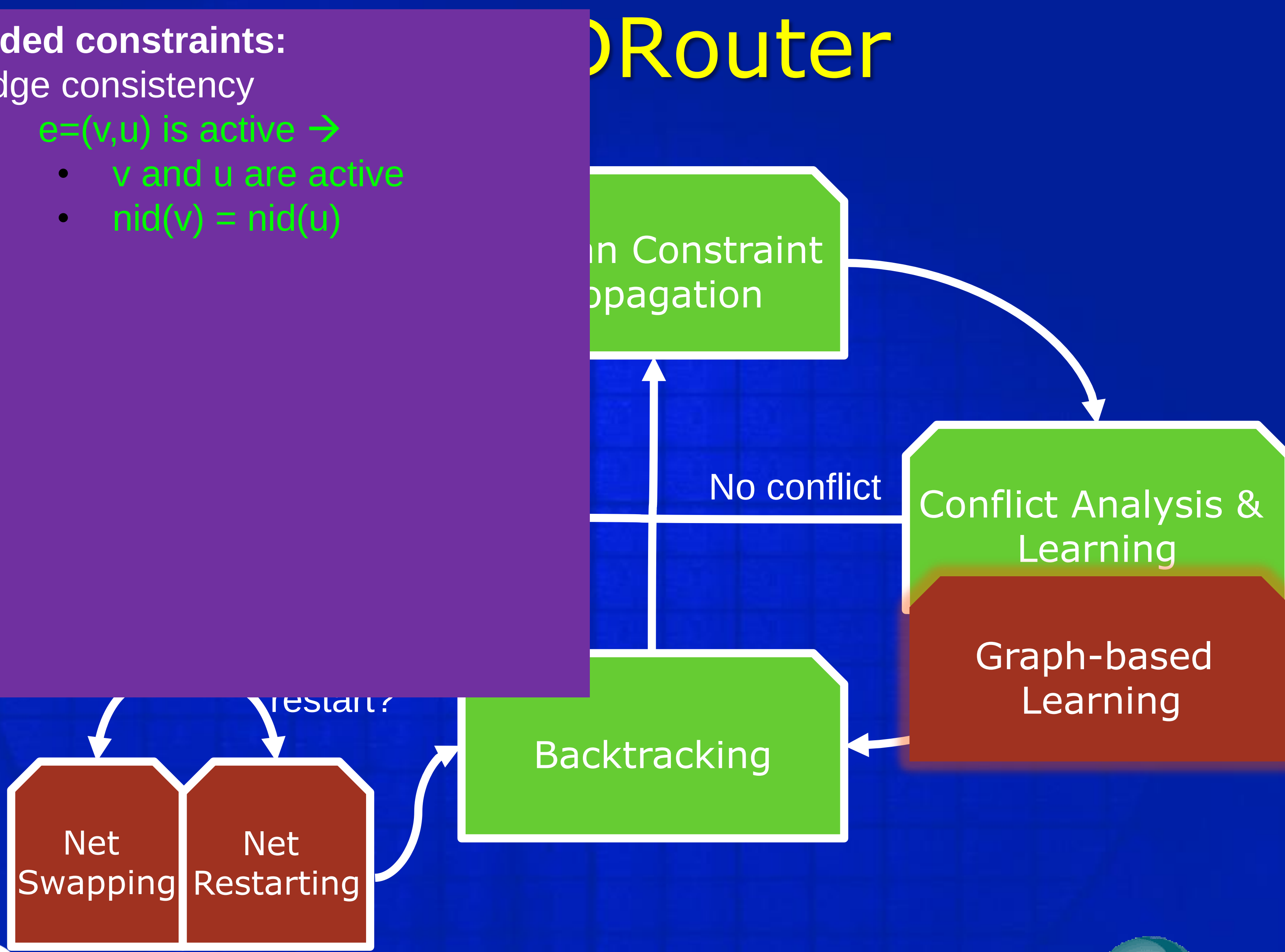
Encoded constraints:



Leap ahead™
Design and
Technology
Solutions

Encoded constraints:

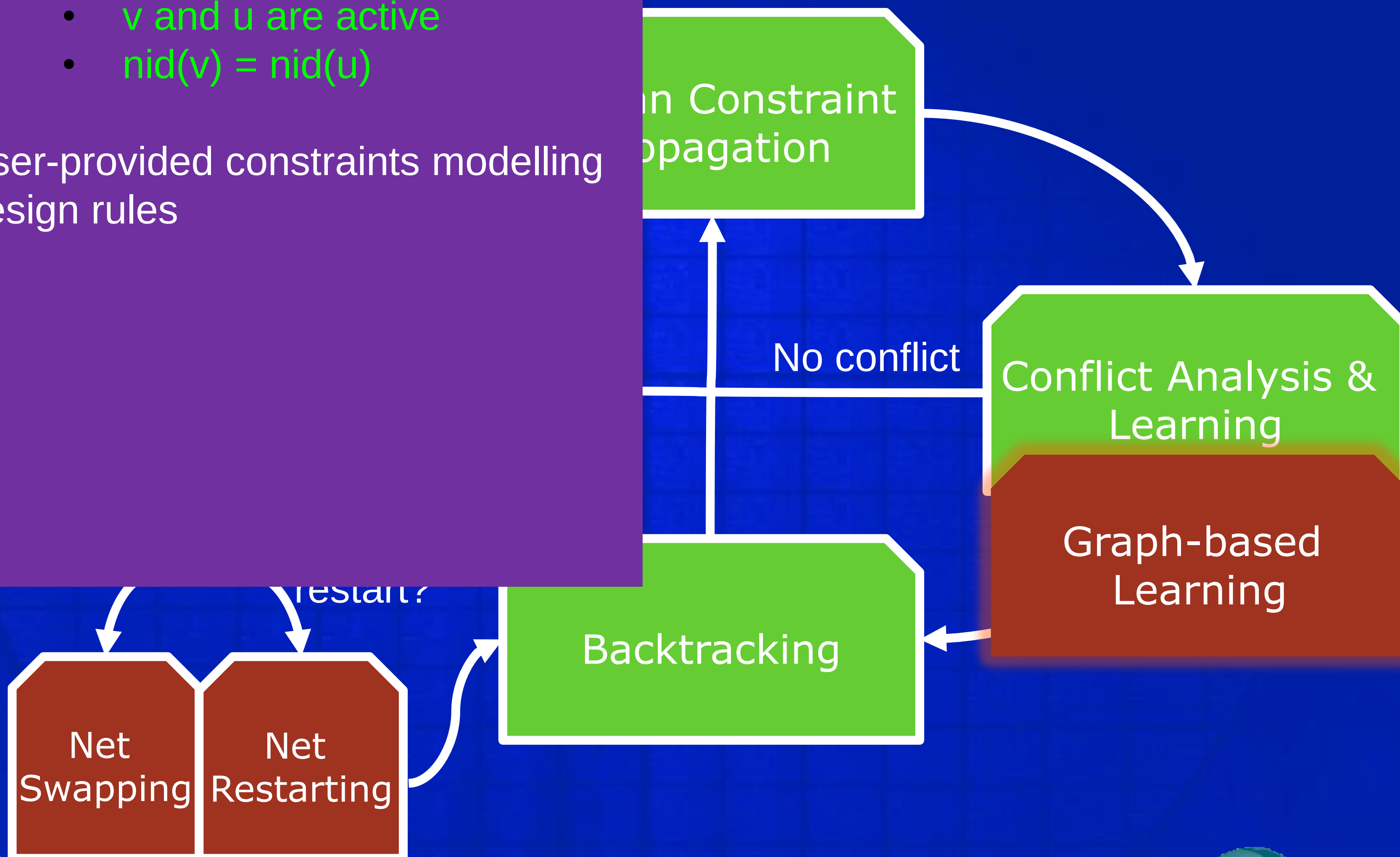
1. Edge consistency
   - e=(v,u) is active →
     - v and u are active
     - nid(v) = nid(u)



Encoded constraints:

1. Edge consistency
   - e=(v,u) is active →
     - v and u are active
     - nid(v) = nid(u)

2. User-provided constraints modelling design rules



Design and Technology Solutions

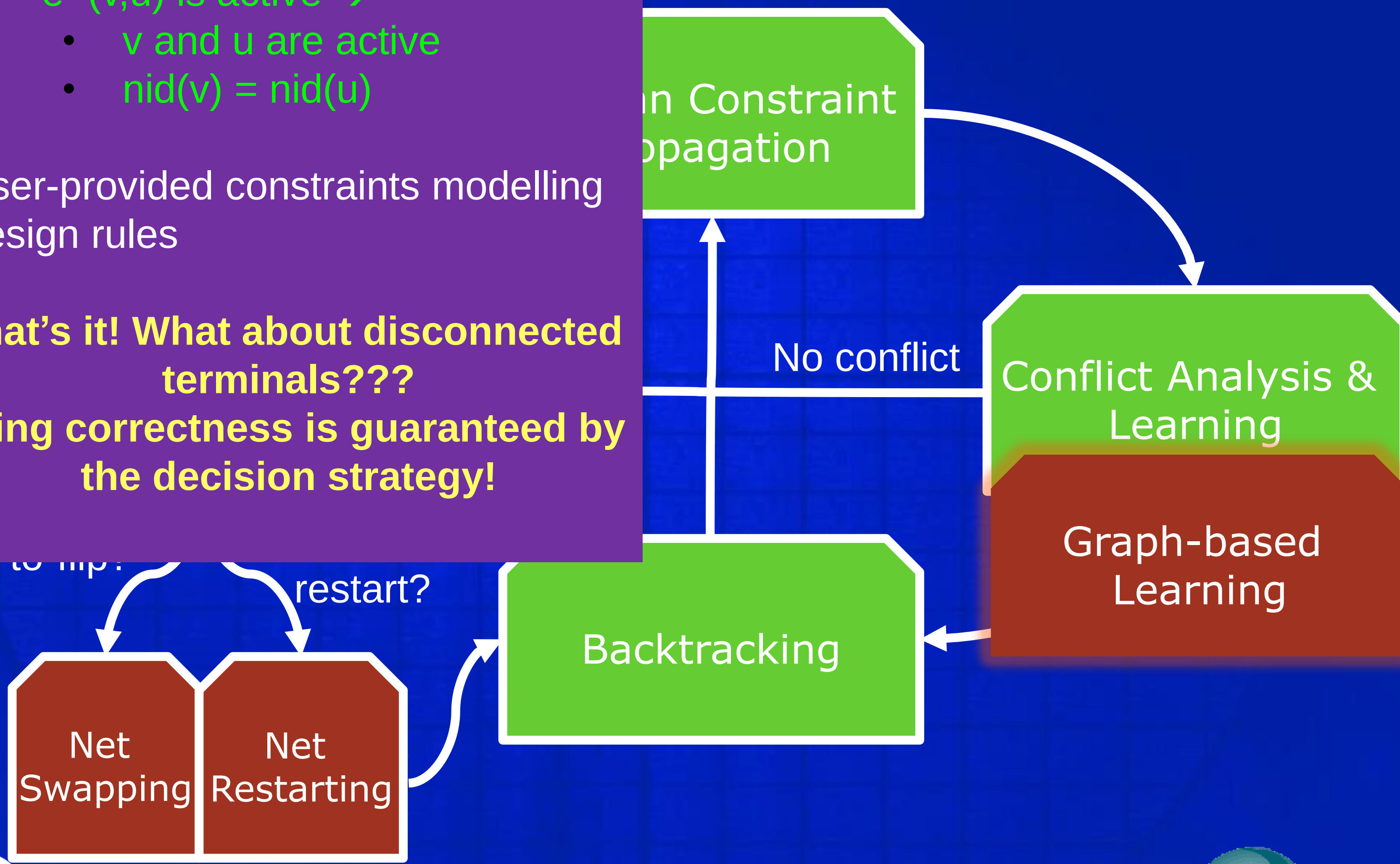
Router

Encoded constraints:

1. Edge consistency
   - e=(v,u) is active →
     - v and u are active
     - nid(v) = nid(u)

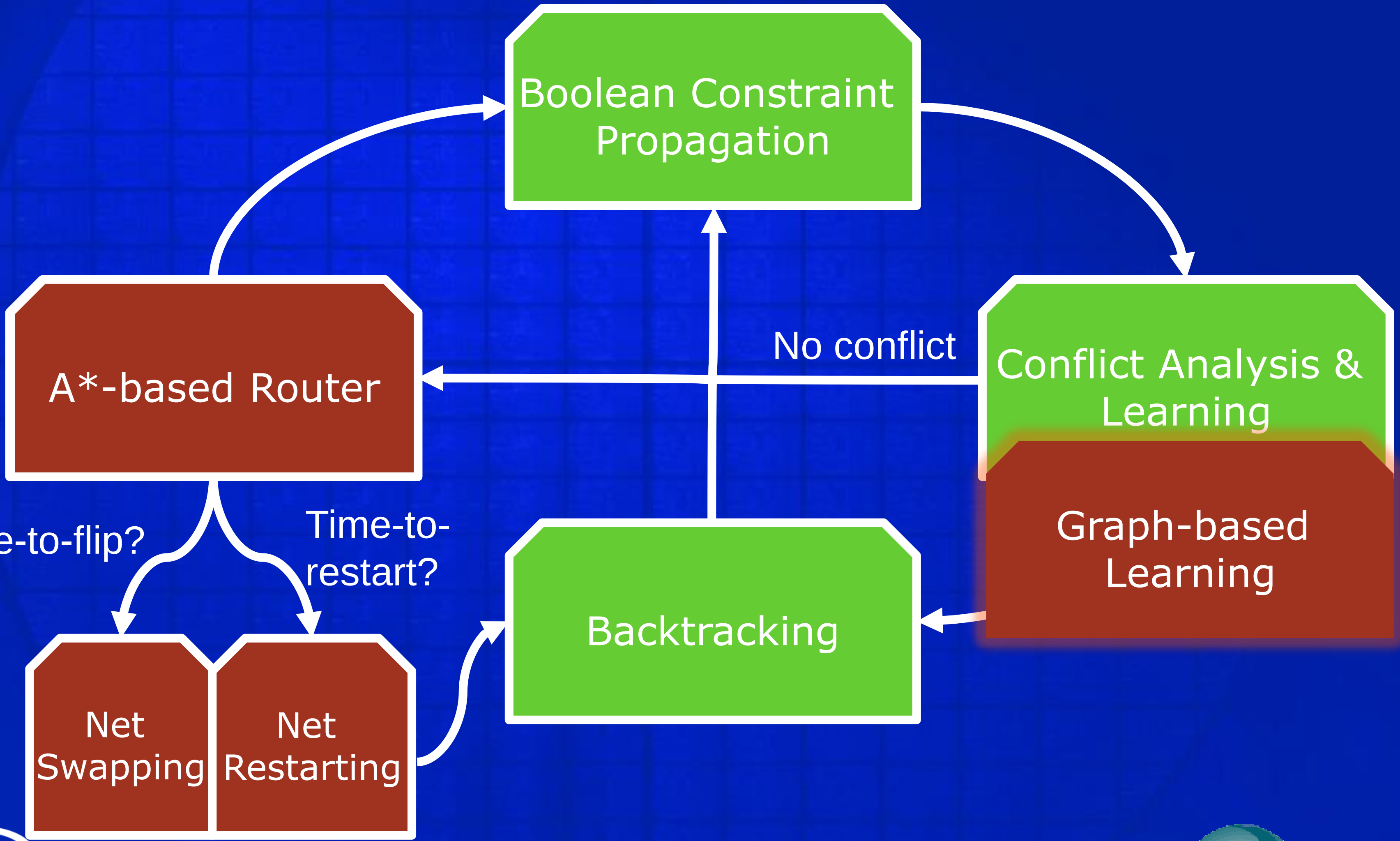
2. User-provided constraints modelling design rules

That's it! What about disconnected terminals???
Routing correctness is guaranteed by the decision strategy!

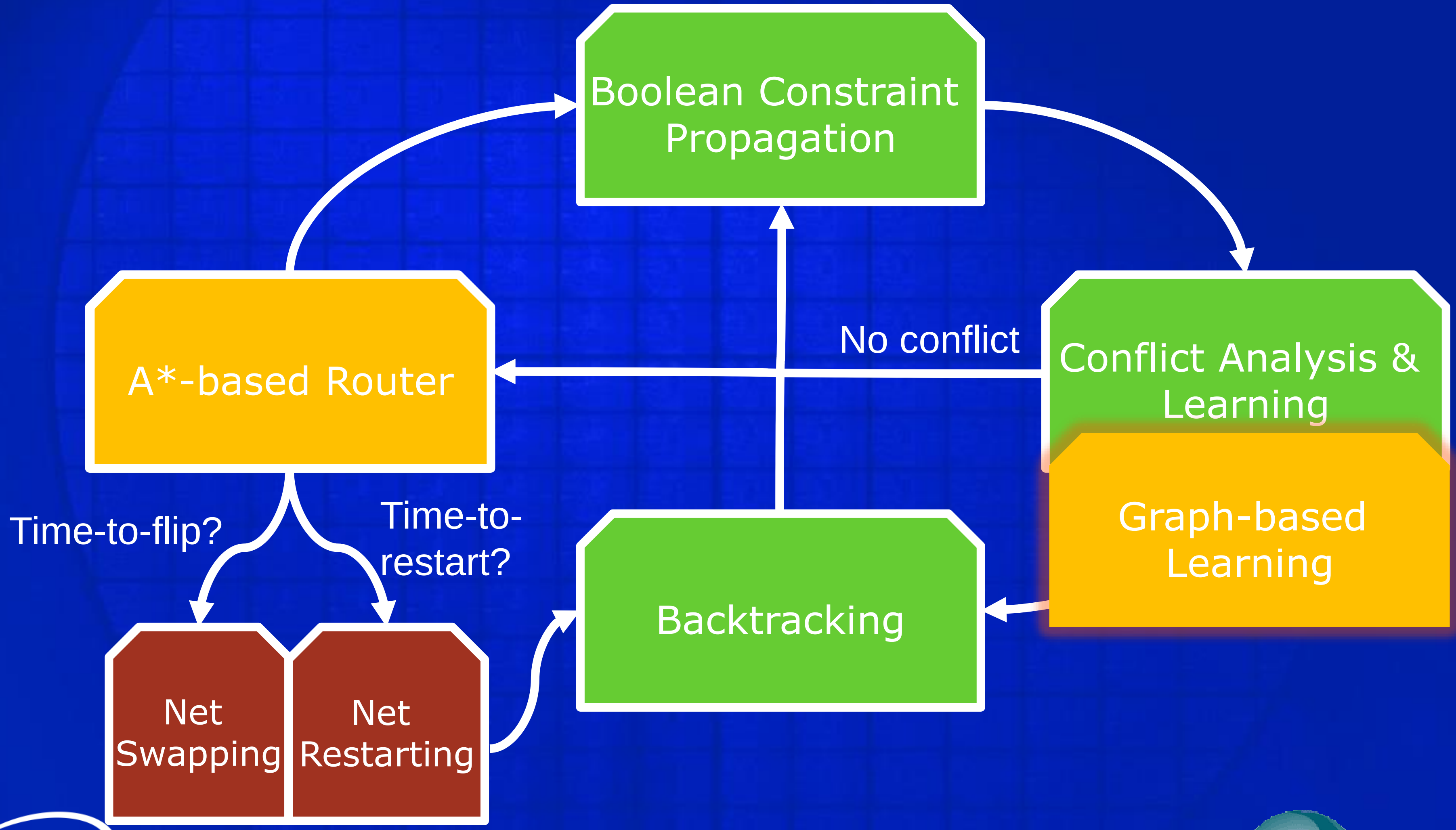


Design and Technology Solutions

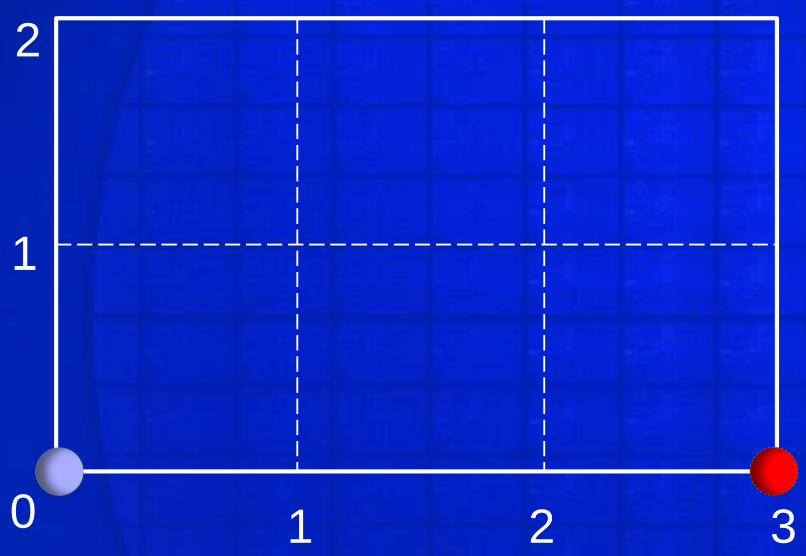
1-Net Example



1-Net Example



1-Net Example

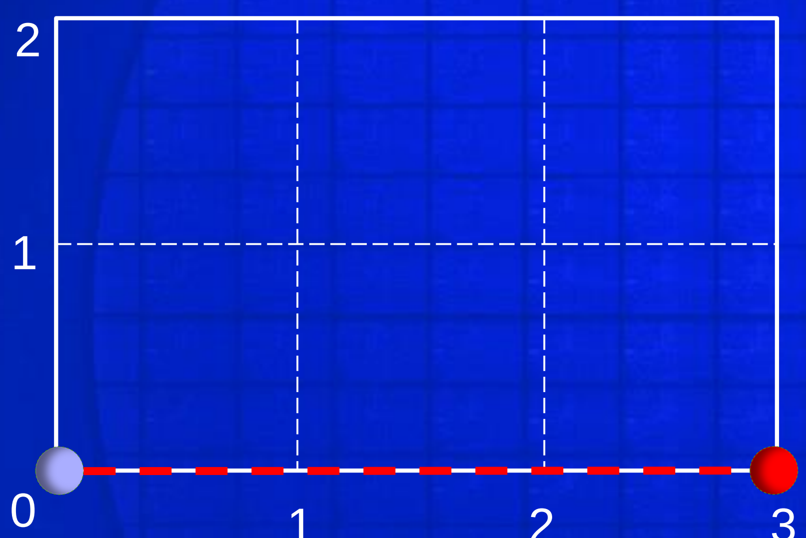


s: (0, 0)
t: (3, 0)

$$\left.\begin{array}{l}\neg(1,0) \vee \neg(2,0) \\ \neg(1,0) \vee \neg(1,1) \\ \neg(3,2) \vee \neg(3,1)\end{array}\right\}\text{ Design rules}$$

intel Leap ahead™
Design and Technology Solutions

1-Net Example



Path Suggestion
(not an actual SAT decision)

s: (0, 0)
t: (3, 0)

σ (sugg.)

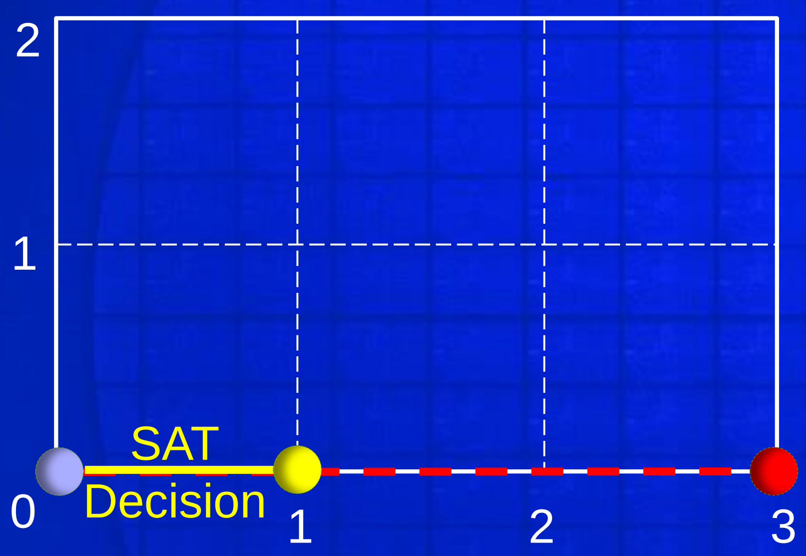
¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)

Initial path:
A* from s->t

Leap ahead™

Design and Technology Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

¬(1,0) ∨ ¬(2,0)

¬(1,0) ∨ ¬(1,1)

¬(3,2) ∨ ¬(3,1)

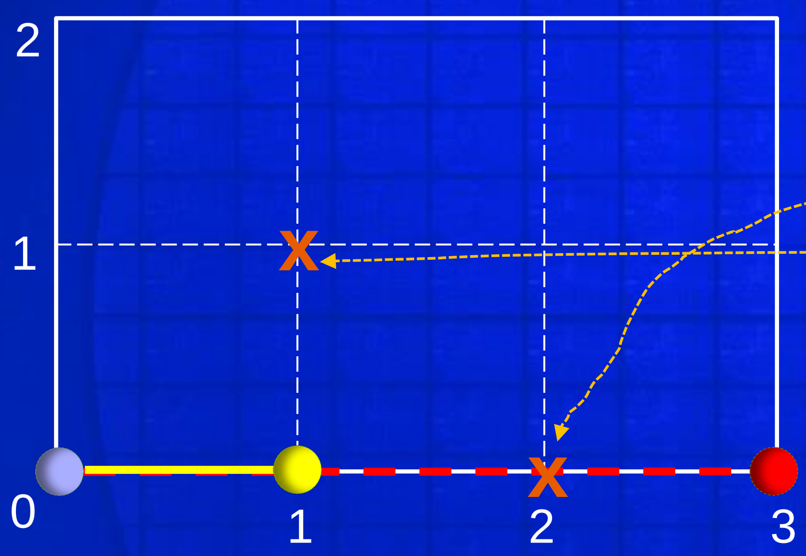
Initial path:

A* from s->t

Activate edge in sugg.

Design and Technology Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\boxed{\neg(1,0) \vee \neg(2,0)}$

$\neg(1,0) \vee \neg(1,1)$

$\neg(3,2) \vee \neg(3,1)$

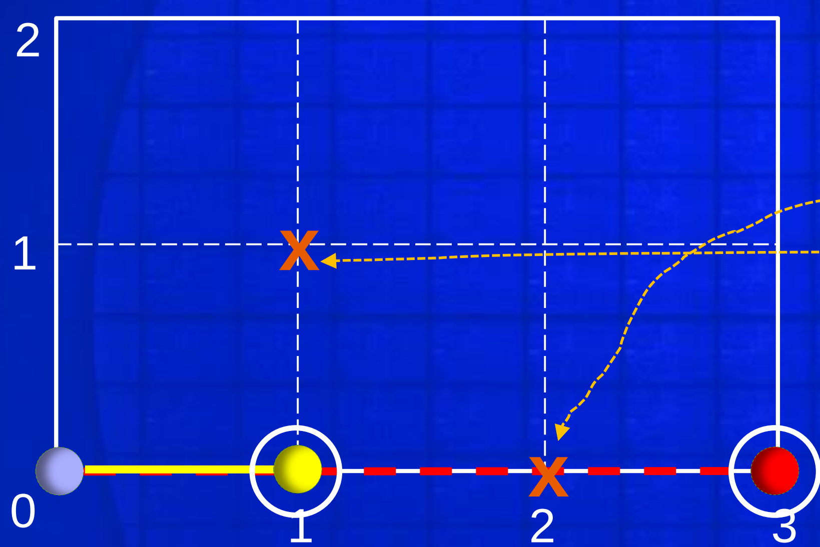
Initial path:
A* from s->t

Activate edge in sugg.

σ-violation

intel Leap ahead™
Design and Technology Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\boxed{\neg(1,0) \vee \neg(2,0)}$

$\neg(1,0) \vee \neg(1,1)$

$\neg(3,2) \vee \neg(3,1)$

Initial path:
A* from s->t

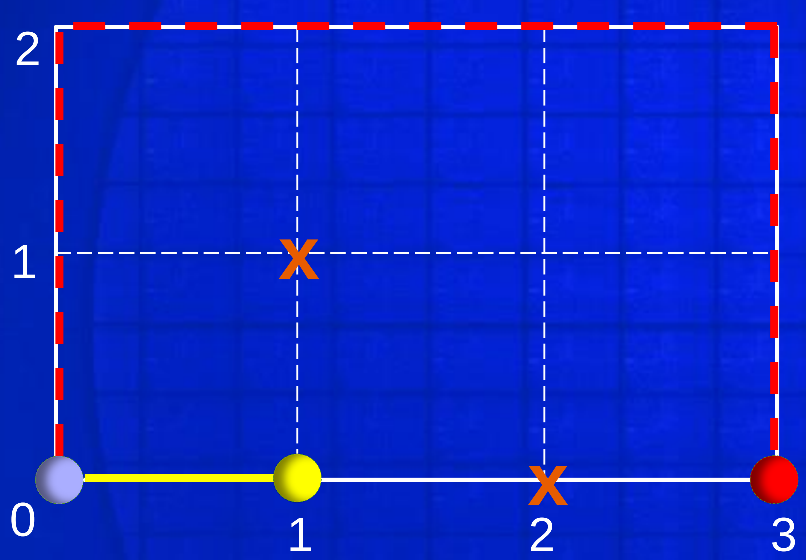
Activate edge in sugg.

σ-violation

A* search
for new σ

Design and
Technology
Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)

Initial path:
A* from s->t

Activate edge in sugg.

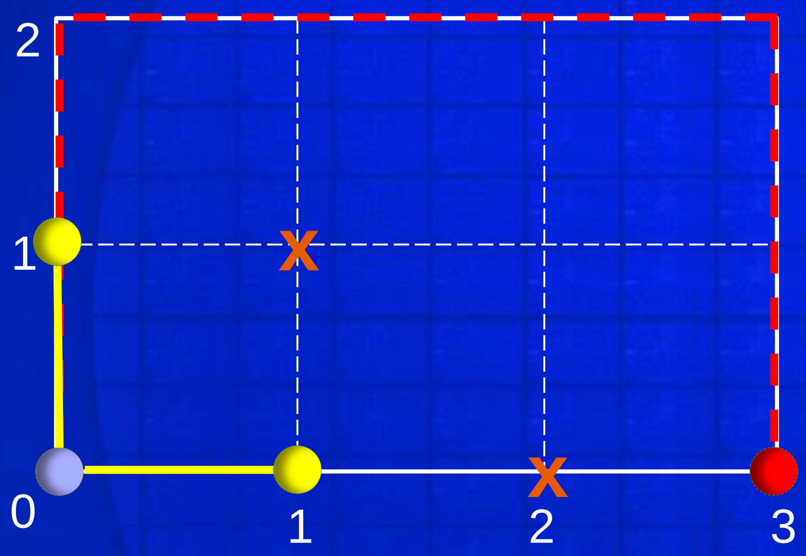
Path found

A* search
for new σ

intel Leap ahead™

Design and
Technology
Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\neg(1,0) \vee \neg(2,0)$

$\neg(1,0) \vee \neg(1,1)$

$\neg(3,2) \vee \neg(3,1)$

Initial path:
A* from s->t

Activate edge in sugg.

No σ-violation

Target is part of path?

no

Repeat

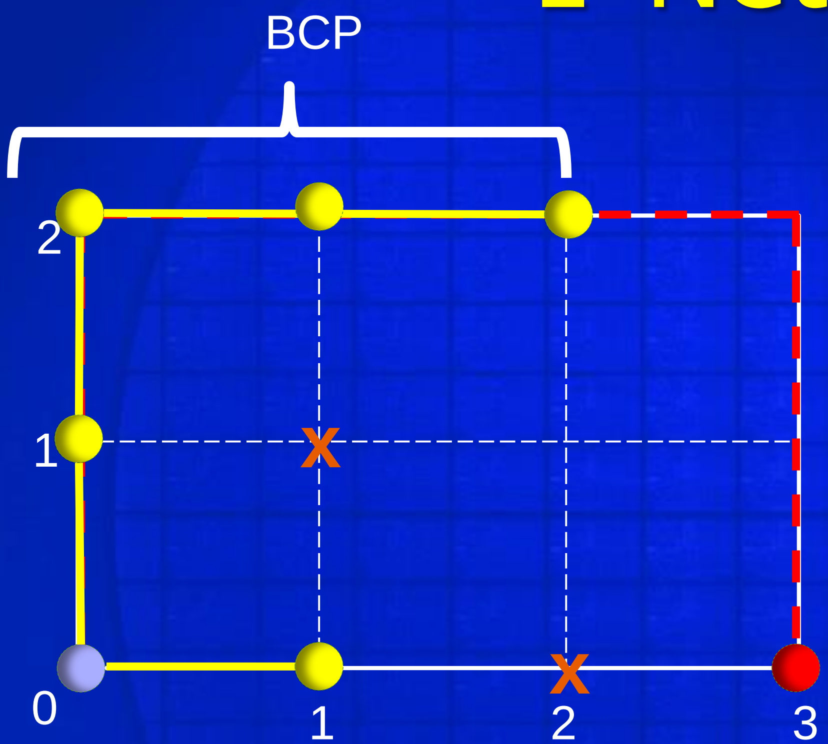
A* search
for new σ

Path found

Leap ahead™

Design and
Technology
Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

No σ-violation

no

Target is part of path?

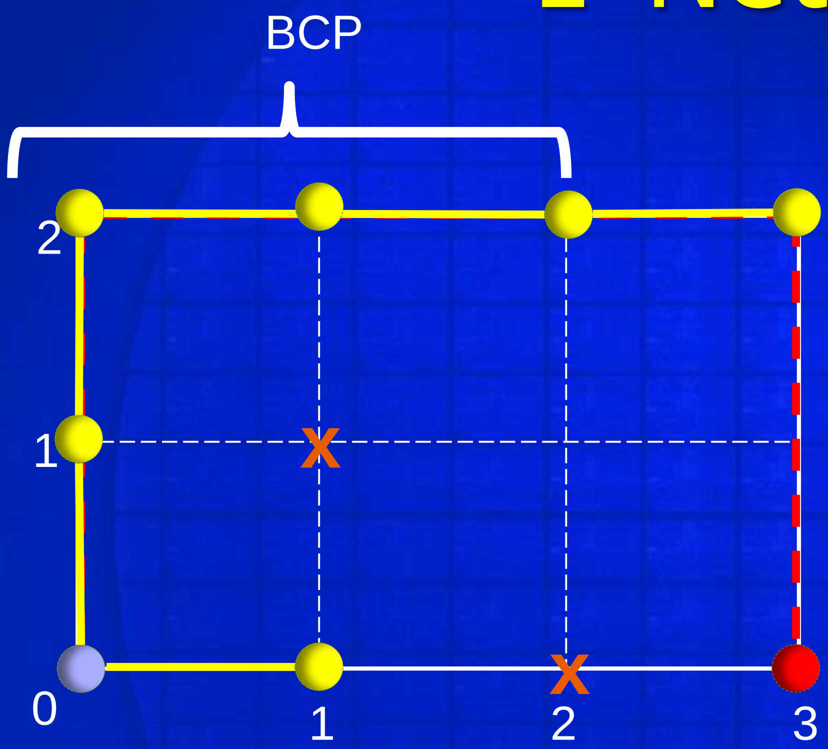
Path found

A* search
for new σ

intel. Leap ahead™

Design and Technology Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)

Initial path:
A* from s->t

Activate edge in sugg.

No σ-violation

Target is part of path?

no

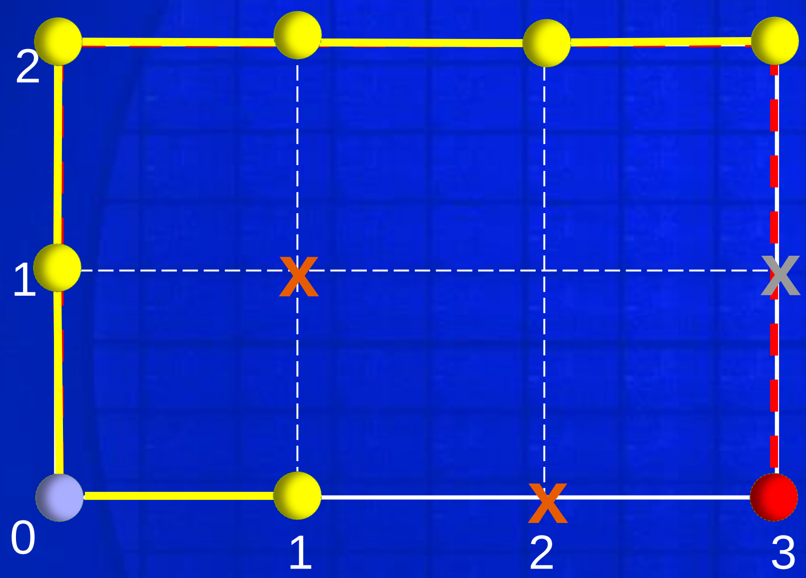
Repeat

Path found

A* search
for new σ

Design and
Technology
Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

¬(1,0) ∨ ¬(2,0)

¬(1,0) ∨ ¬(1,1)

$\boxed{¬(3,2) ∨ ¬(3,1)}$

Initial path:
A* from s->t

Repeat

Activate edge in sugg.

σ-violation

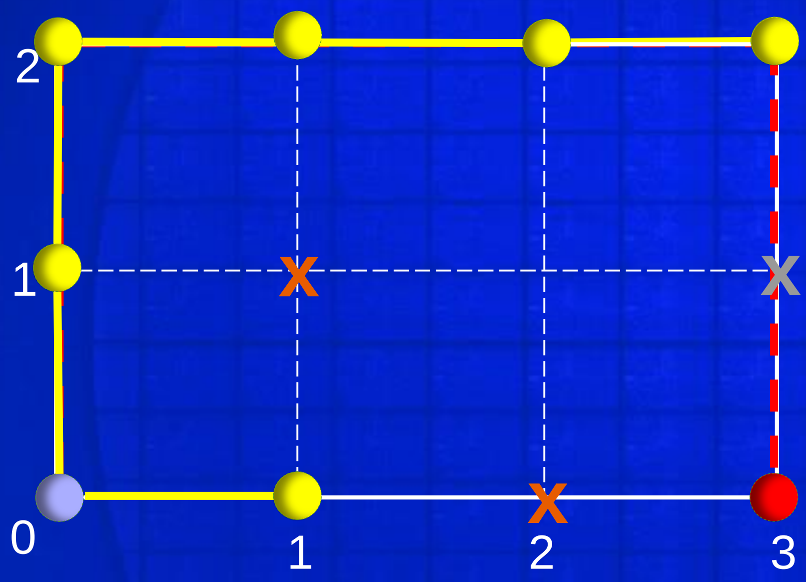
No σ-violation

no

A* search
for new σ

Target is part of path?

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\neg(1,0) \vee \neg(2,0)$

$\neg(1,0) \vee \neg(1,1)$

$\boxed{\neg(3,2) \vee \neg(3,1)}$

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

σ-violation

No σ-violation

no

A* search
for new σ

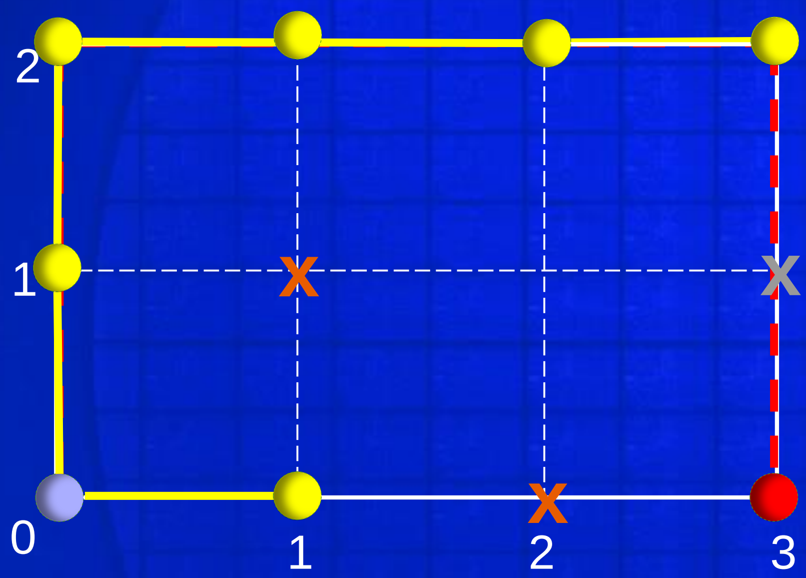
Target is part of path?

Graph conflict (s and t can't be connected)

intel Leap ahead™

Design and Technology Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

σ-violation

No σ-violation

no

Add conflicting clause: vertex cut (2,0) ∨ (3,1)

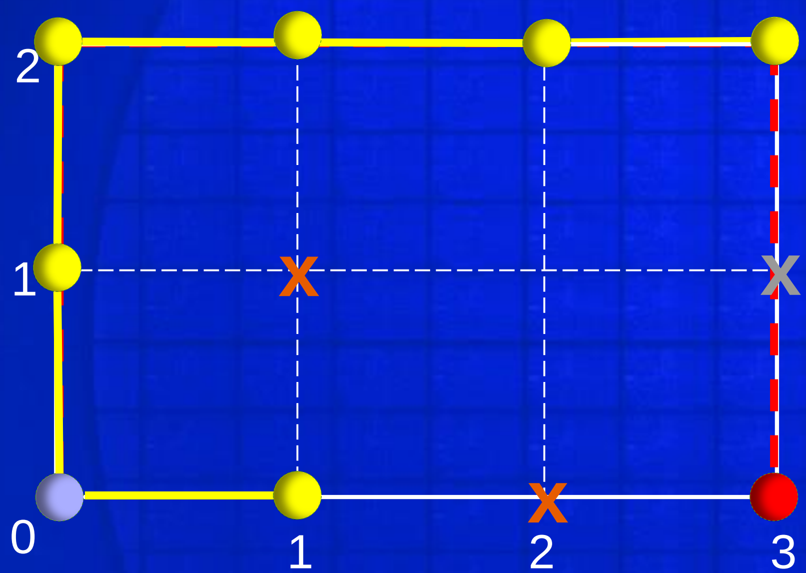
A* search
for new σ

Target is part of path?

Graph conflict (s and t can't be connected)

intel Leap ahead™
Design and Technology Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
$\boxed{\neg(3,2) \vee \neg(3,1)}$

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

σ-violation

No σ-violation

no

A* search
for new σ

Target is part of path?

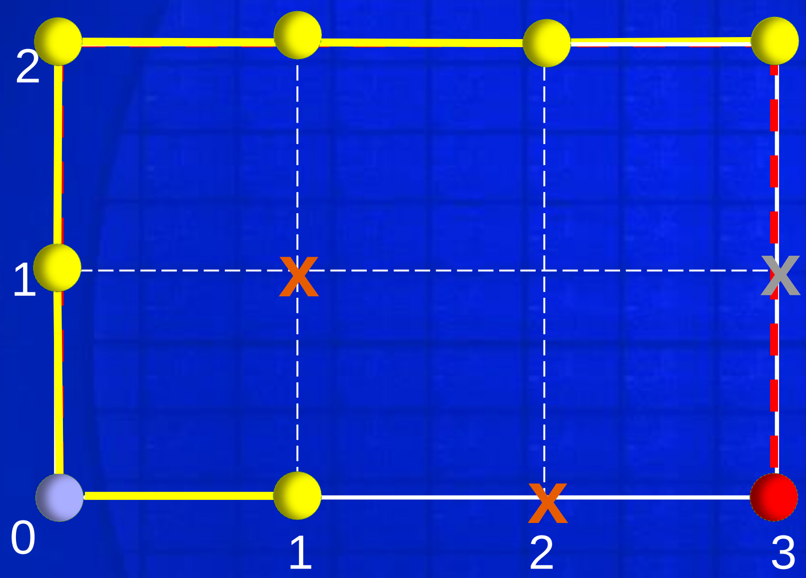
1UIP conflict clause: (2,0) ∨ ¬(3,2)
Add conflicting clause: vertex cut (2,0) ∨ (3,1)

Graph conflict (s and t can't be connected)

Leap ahead™

Design and
Technology
Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
$\boxed{\neg(3,2) \vee \neg(3,1)}$

Initial path:
A* from s->t

Activate edge in sugg.

σ-violation

No σ-violation

A* search
for new σ

Target is part of path?

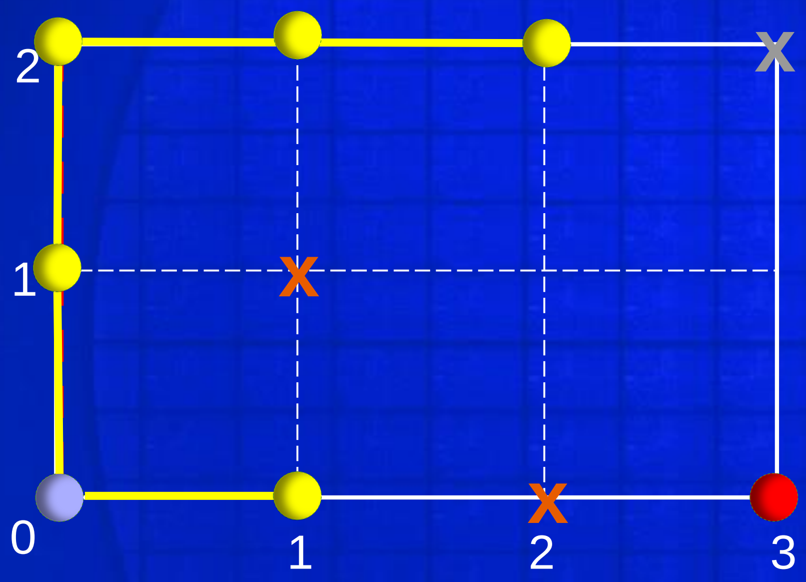
no

Repeat

Graph conflict (s and t can't be connected)

1UIP conflict clause: (2,0) ∨ ¬(3,2)
Add conflicting clause: vertex cut (2,0) ∨ (3,1)

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)
(2,0) ∨ ¬(3,2)

Initial path:
A* from s->t

Activate edge in sugg.

σ-violation

No σ-violation

A* search
for new σ

Target is part of path?

no

Repeat

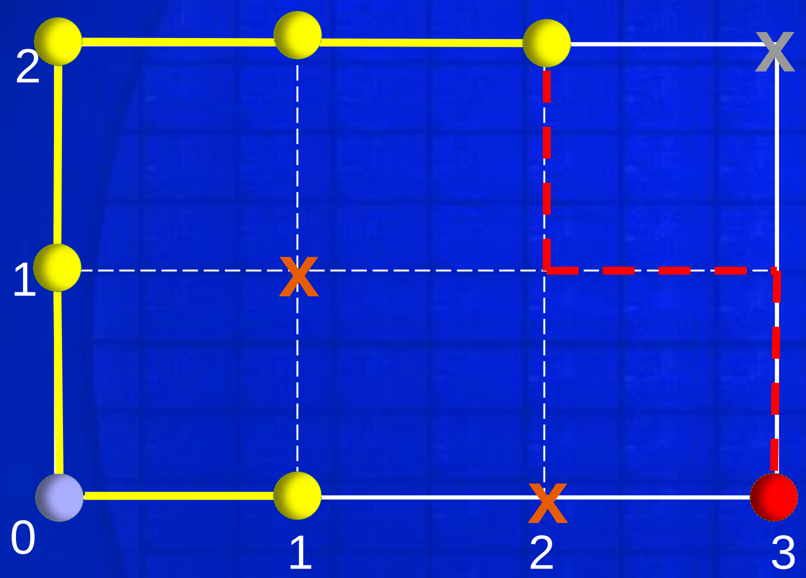
1UIP conflict clause: (2,0) ∨ ¬(3,2)
Add conflicting clause: vertex cut (2,0) ∨ (3,1)

Learn & Backtrack

Graph conflict (s and t can't be connected)

intel Leap ahead
Design and Technology Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\neg(1,0) \vee \neg(2,0)$

$\neg(1,0) \vee \neg(1,1)$

$\neg(3,2) \vee \neg(3,1)$

$(2,0) \vee \neg(3,2)$

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

σ-violation

No σ-violation

no

A* search
for new σ

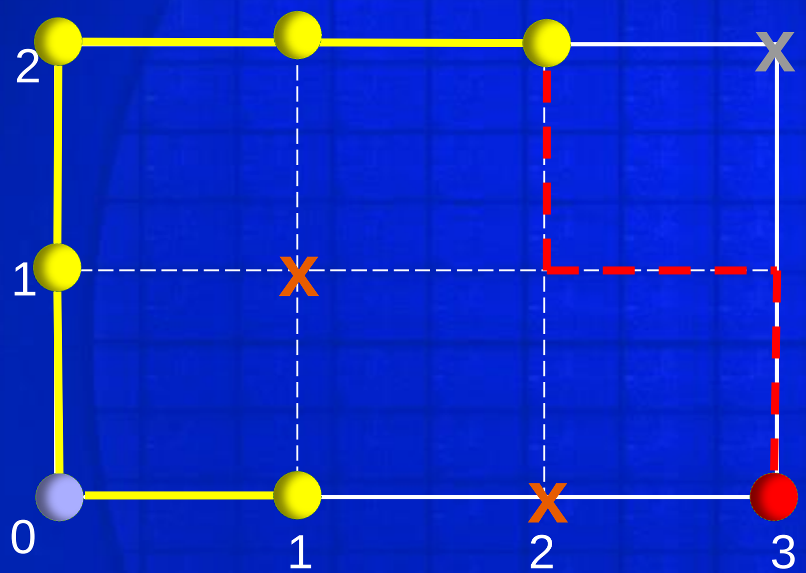
Target is part of path?

Learn & Backtrack

Graph conflict

Design and Technology Solutions

1-Net Example



s: (0, 0)

t: (3, 0)

σ (sugg.)

Real path

$\neg(1,0) \vee \neg(2,0)$

$\neg(1,0) \vee \neg(1,1)$

$\neg(3,2) \vee \neg(3,1)$

$(2,0) \vee \neg(3,2)$

Initial path:
A* from s->t

Activate edge in sugg.

σ-violation

No σ-violation

Target is part of path?

no

Repeat

A* search
for new σ

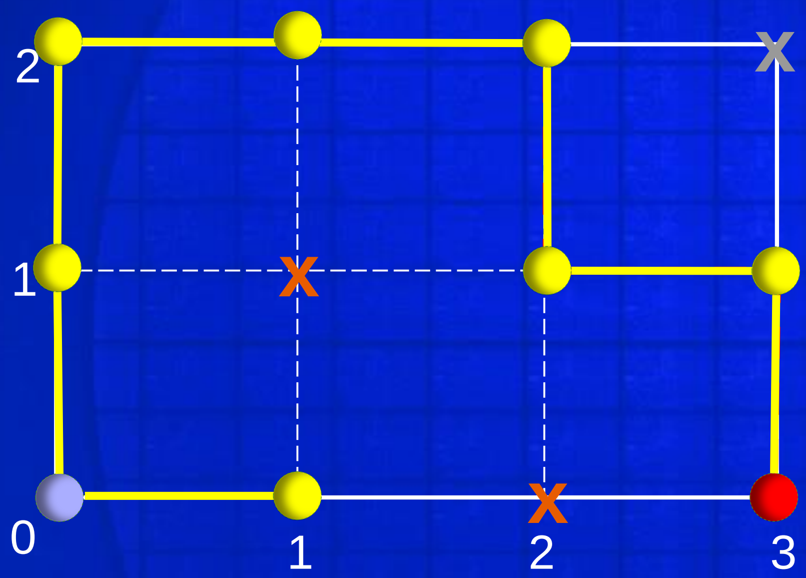
Learn & Backtrack

Graph conflict

intel Leap ahead™

Design and Technology Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)
(2,0) ∨ ¬(3,2)

Initial path:
A* from s->t

Activate edge in sugg.

σ-violation

No σ-violation

Repeat

no

A* search
for new σ

Target is part of path?

(yes!)

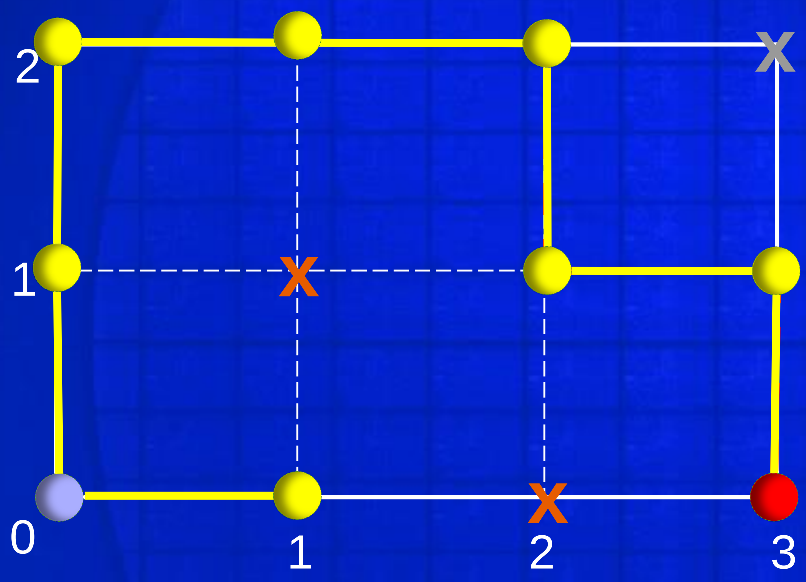
Learn & Backtrack

Graph conflict

DONE!

intel Leap ahead™
Design and Technology Solutions

1-Net Example



s: (0, 0)
t: (3, 0)

σ (sugg.)
Real path

¬(1,0) ∨ ¬(2,0)
¬(1,0) ∨ ¬(1,1)
¬(3,2) ∨ ¬(3,1)
(2,0) ∨ ¬(3,2)

Initial path:
A* from s->t

Activate edge in sugg.

Repeat

σ-violation

No σ-violation

no

A* search
for new σ

Path found

Target is part of path?

(yes!)

DONE!

Learn & Backtrack

Graph conflict

Result:
Path that follows constraints!

Leap ahead™
Design and Technology Solutions

Multiple Nets Handling

Leap ahead™
Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!

intel Leap ahead™
Design and Technology Solutions

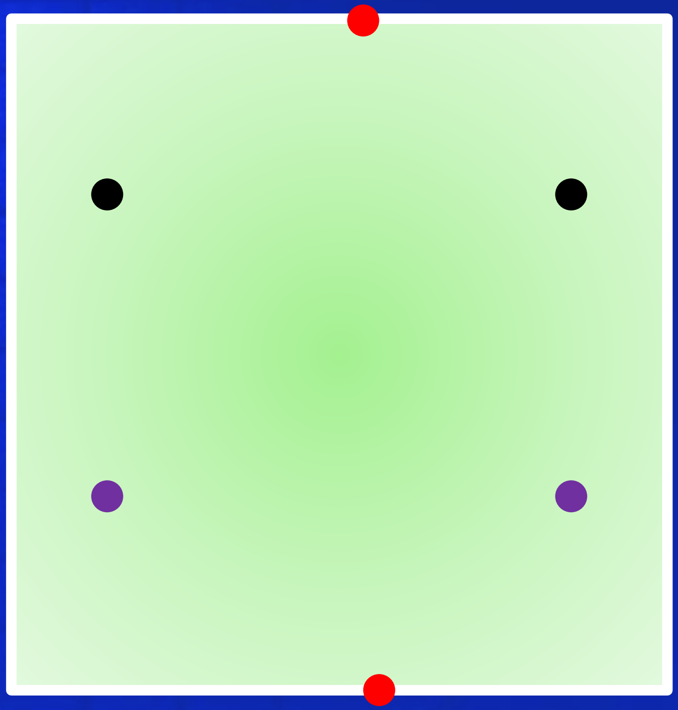
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!

Design and Technology Solutions

Multiple Nets Handling

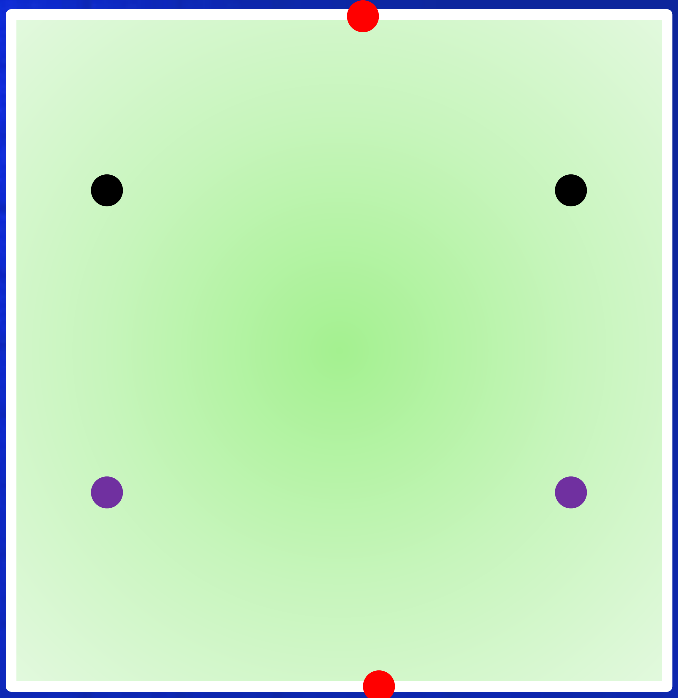
- Route the nets one-by-one
  - Order is critical!



Design and Technology Solutions

Multiple Nets Handling

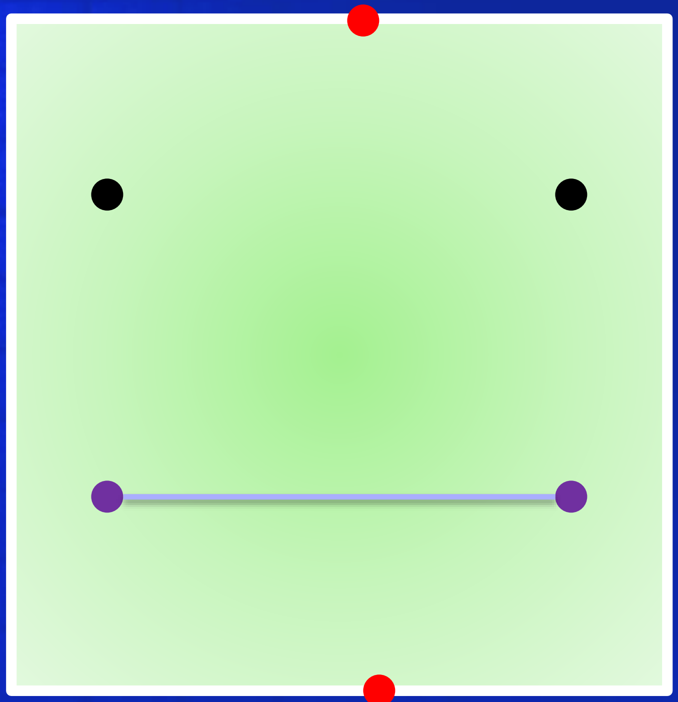
- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red



intel Leap ahead™
Design and Technology Solutions

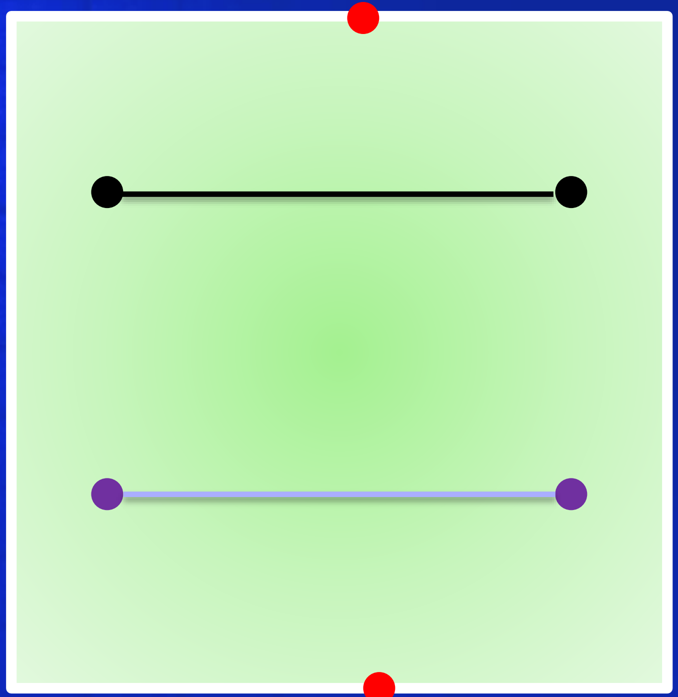
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red



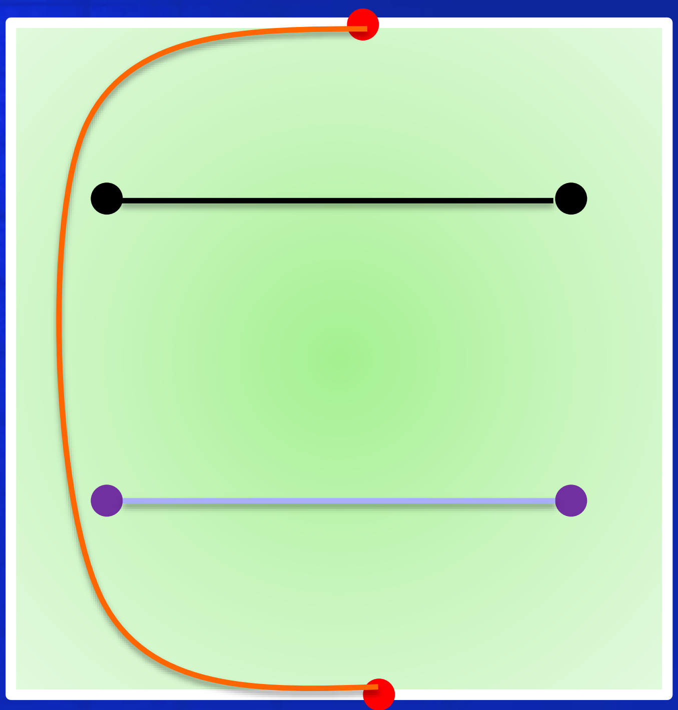
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red



Multiple Nets Handling

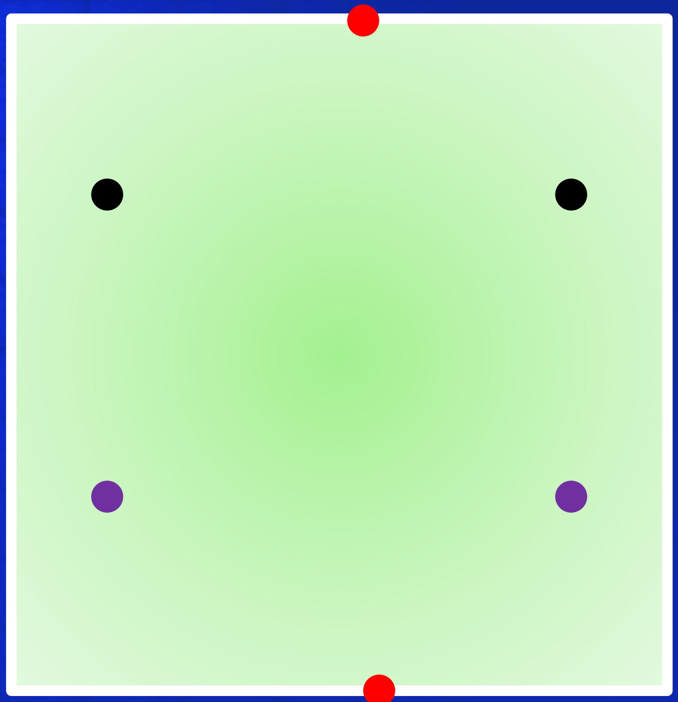
- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red



intel Leap ahead™
Design and Technology Solutions

Multiple Nets Handling

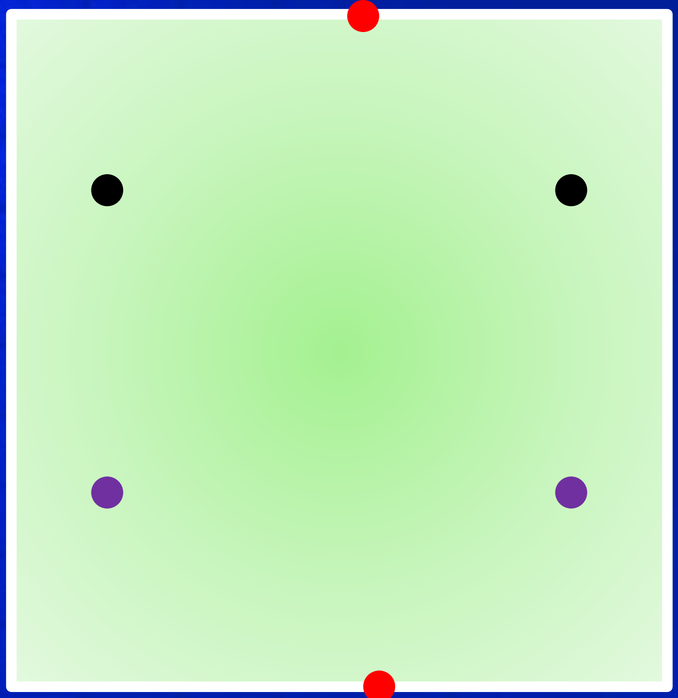
- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



intel Leap ahead™
Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

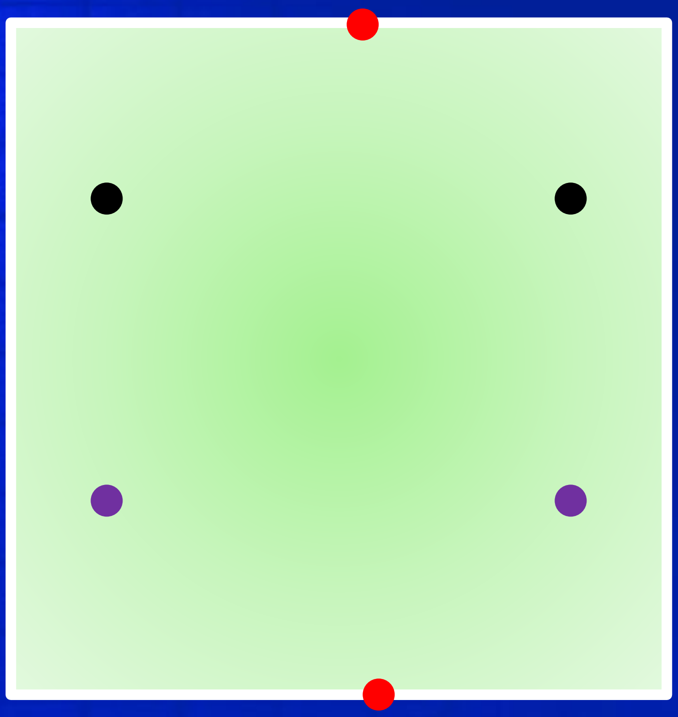


Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

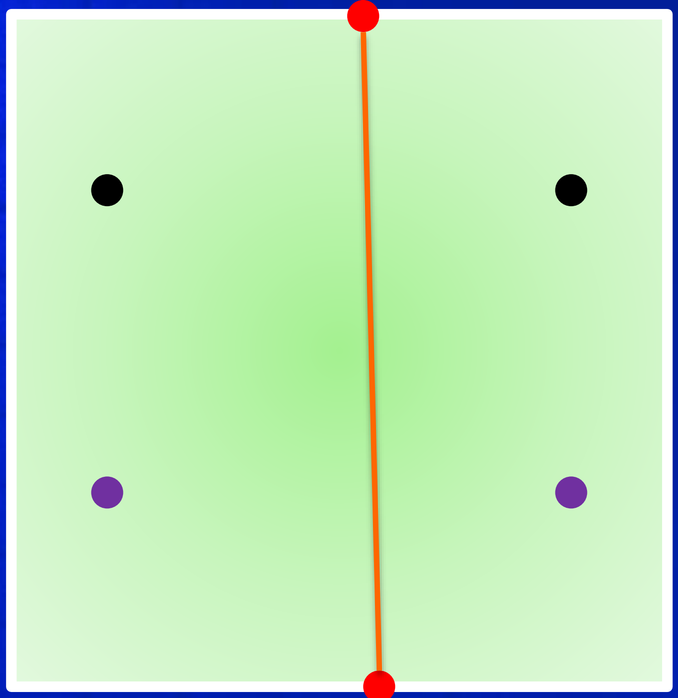


intel Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

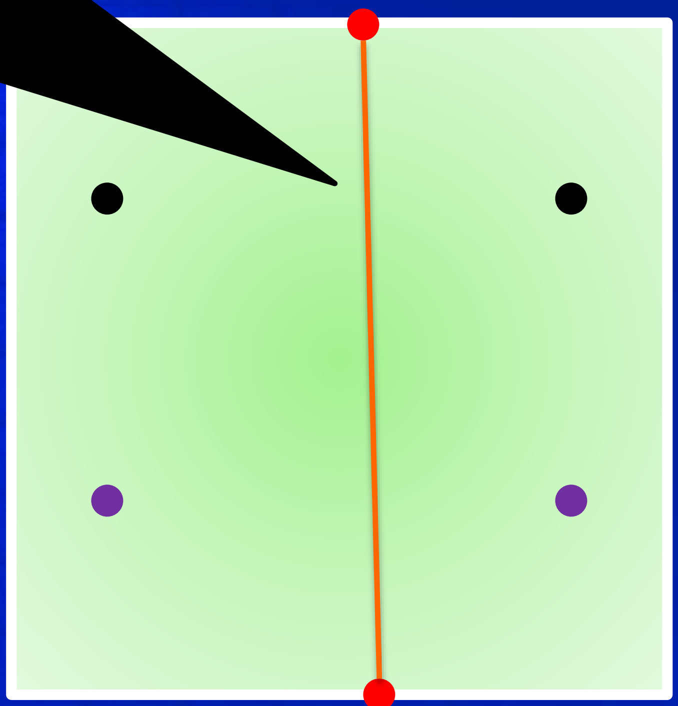


Design and Technology Solutions

- **Graph conflict**
  - black is blocked
- **Early conflict detection**
  - Check for graph conflicts after routing each terminal
- **Learn a conflict clause & re-route**

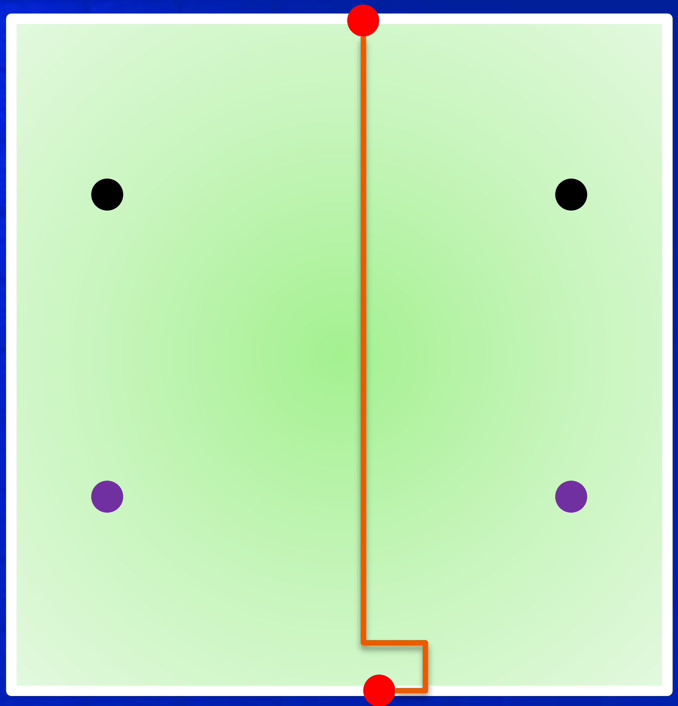
...andling

- F...
  - ...
- Exa...
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



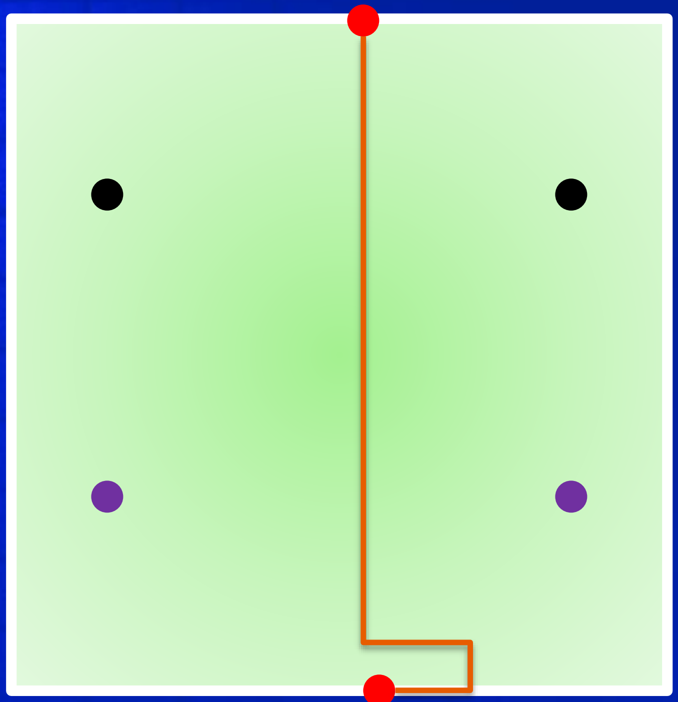
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



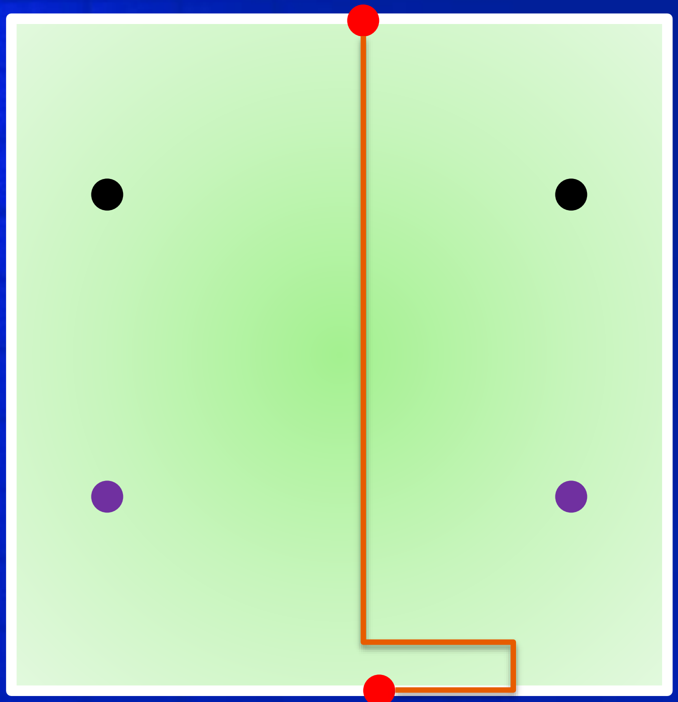
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



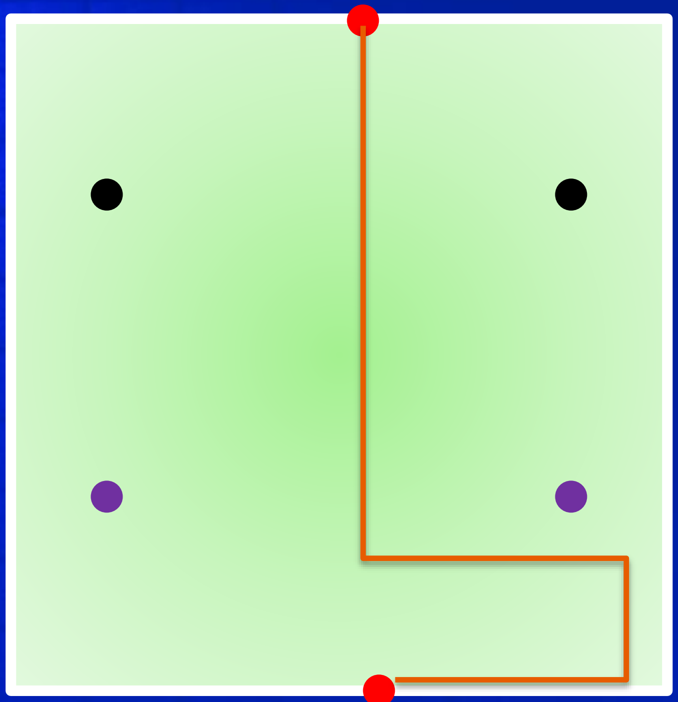
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

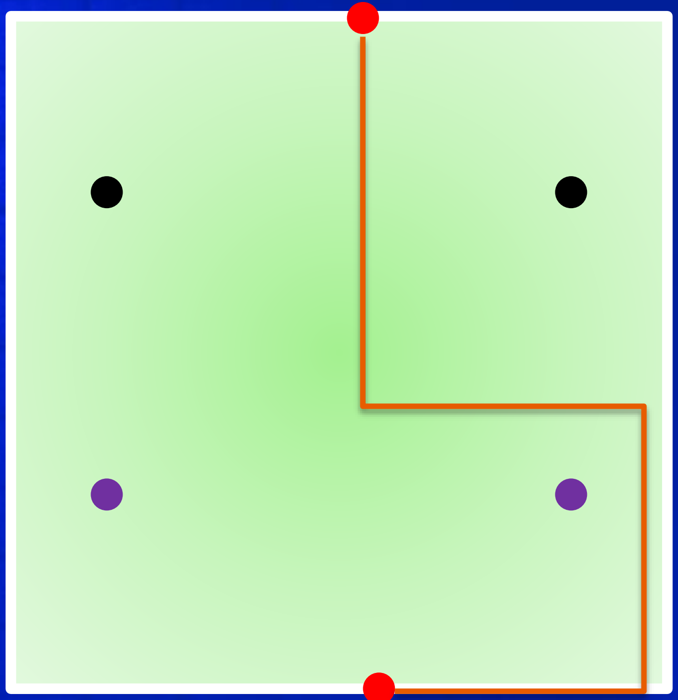


intel Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

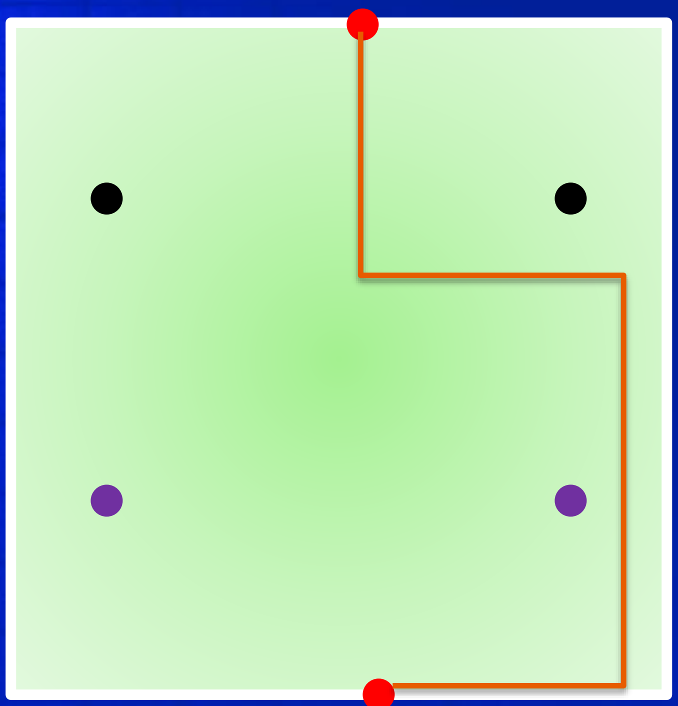


intel Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

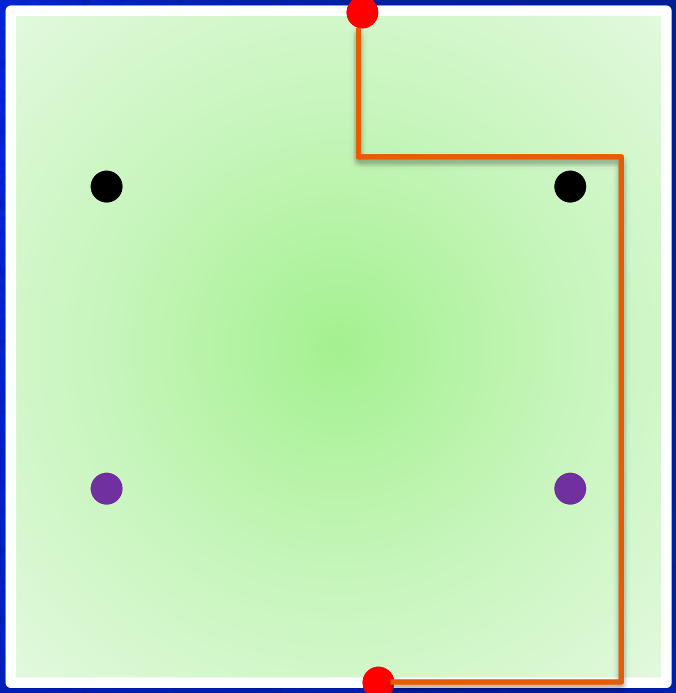


intel Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

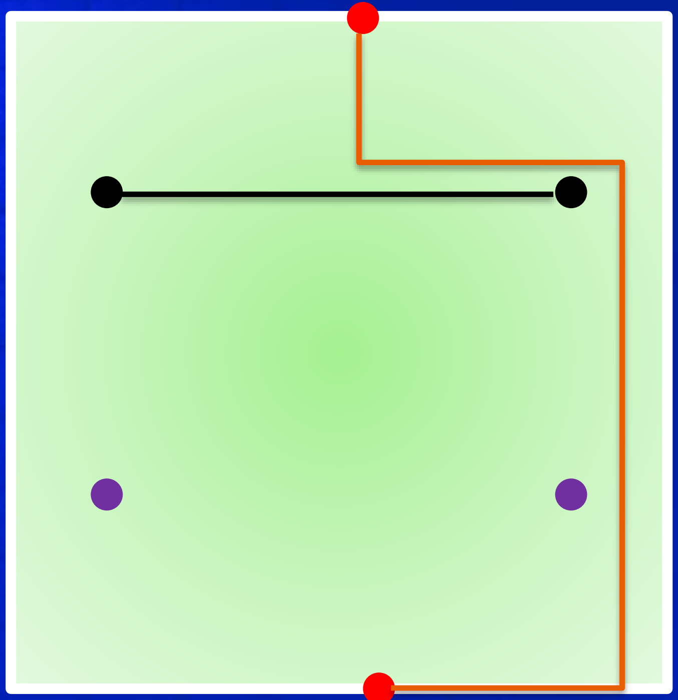
- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet



Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

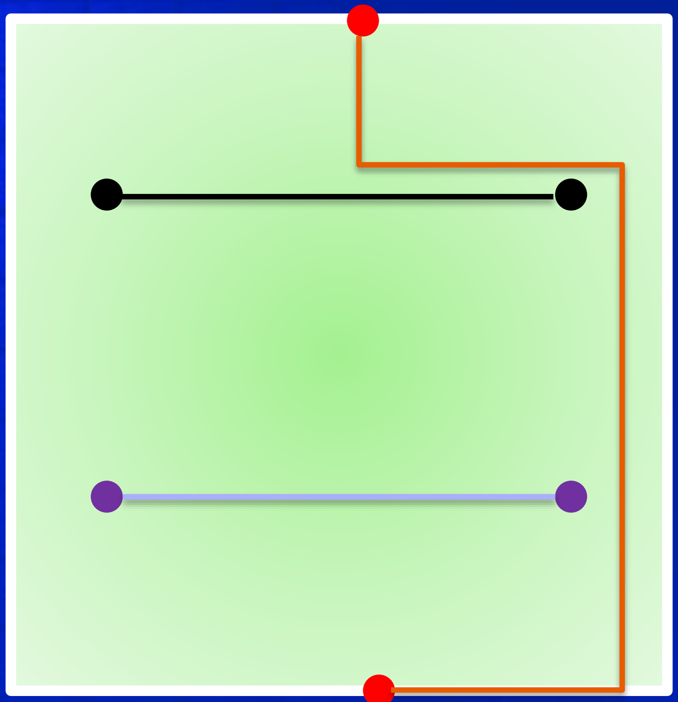


Leap ahead™

Design and Technology Solutions

Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet

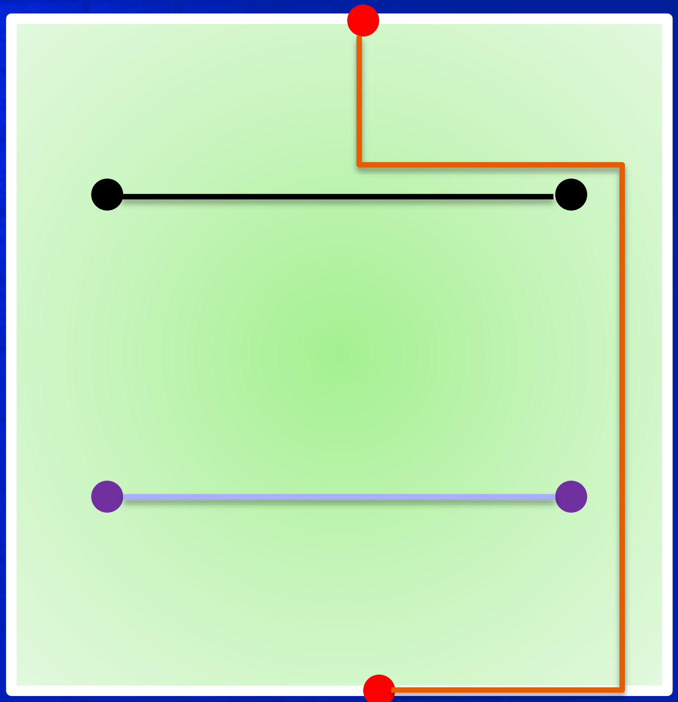


intel Leap ahead™

DTS Design and Technology Solutions

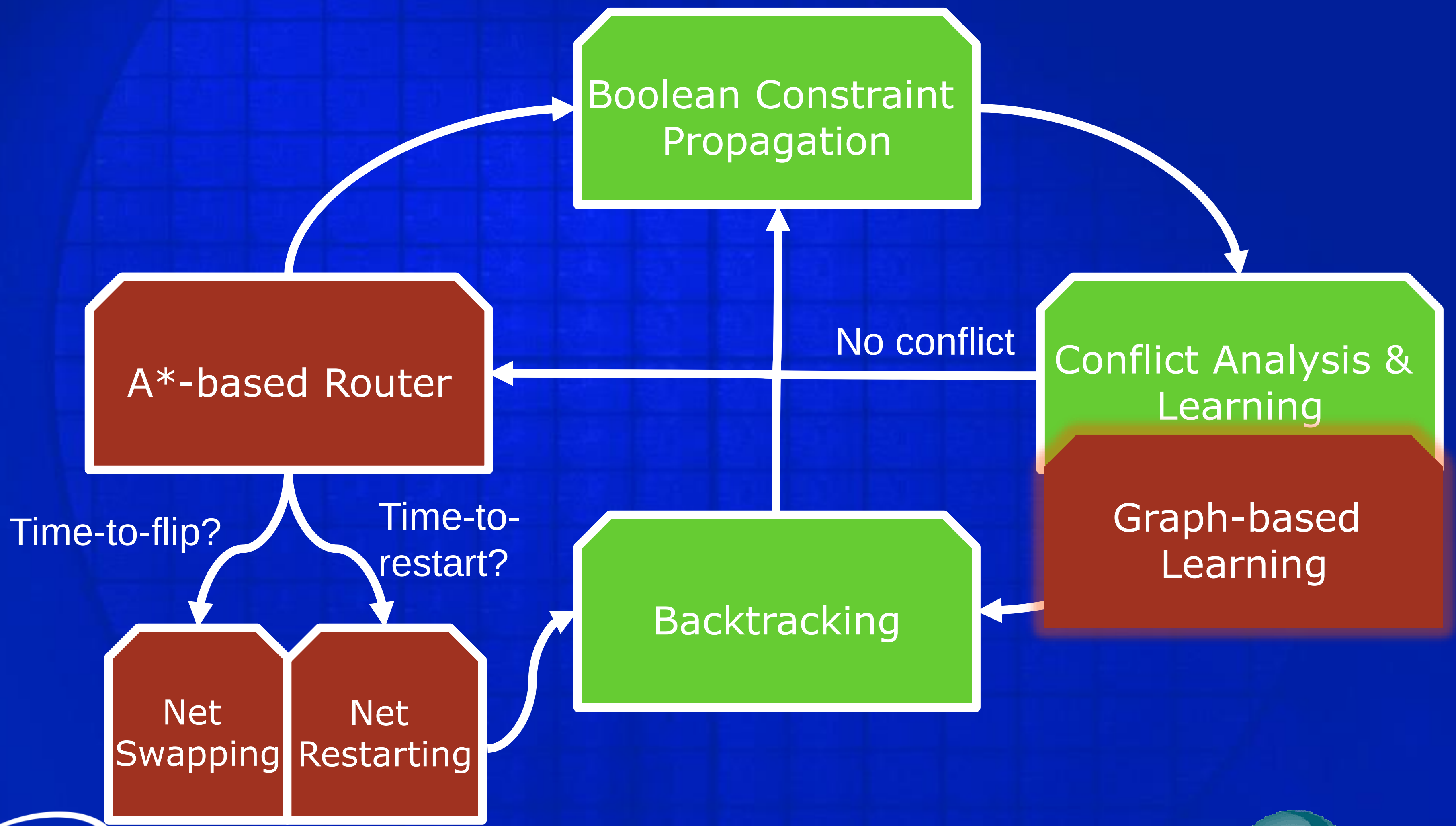
Multiple Nets Handling

- Route the nets one-by-one
  - Order is critical!
- Example Order 1:
  - Violet
  - Black
  - Red
- Example Order 2:
  - Red
  - Black
  - Violet
- Too slow! Solution: dynamic net reordering!



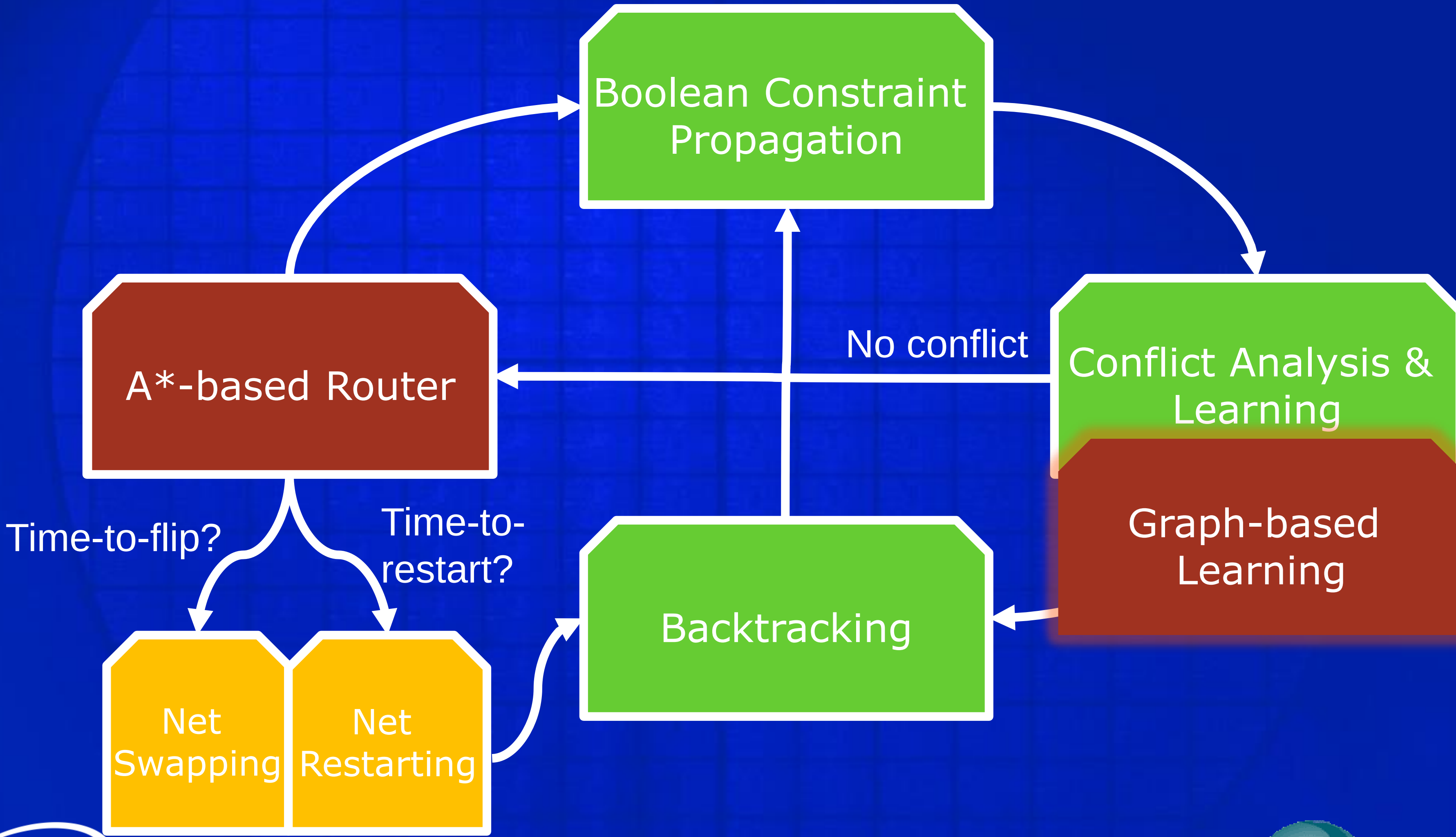
Design and Technology Solutions

DRouter



intel Leap ahead
Design and Technology Solutions
52

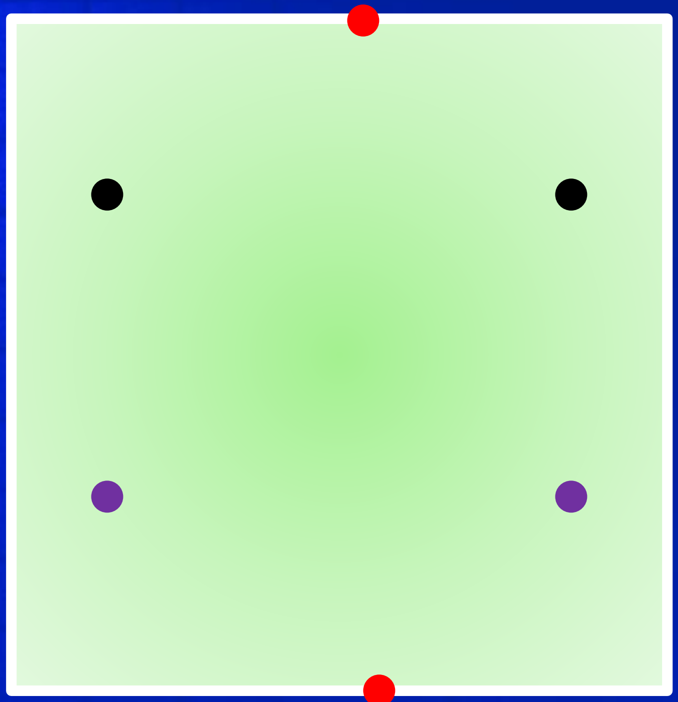
DRouter



intel Leap ahead
Design and Technology Solutions

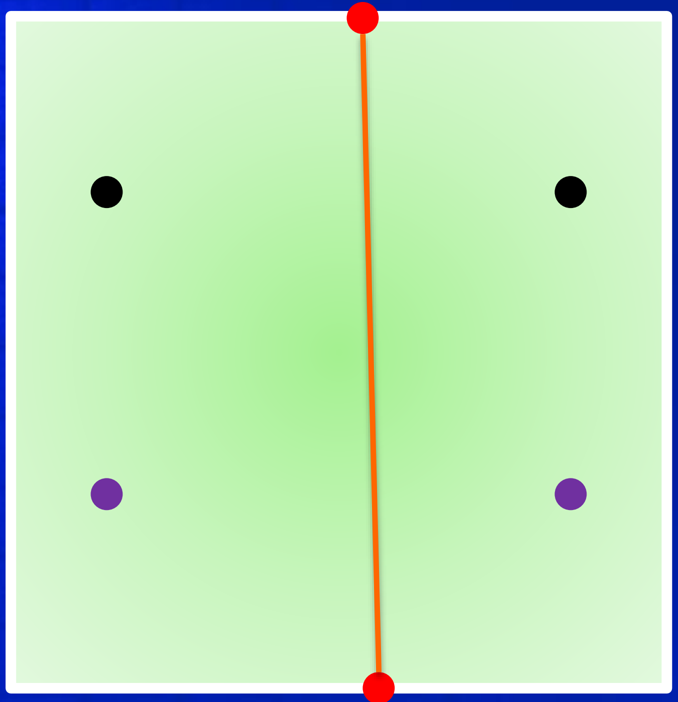
Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet



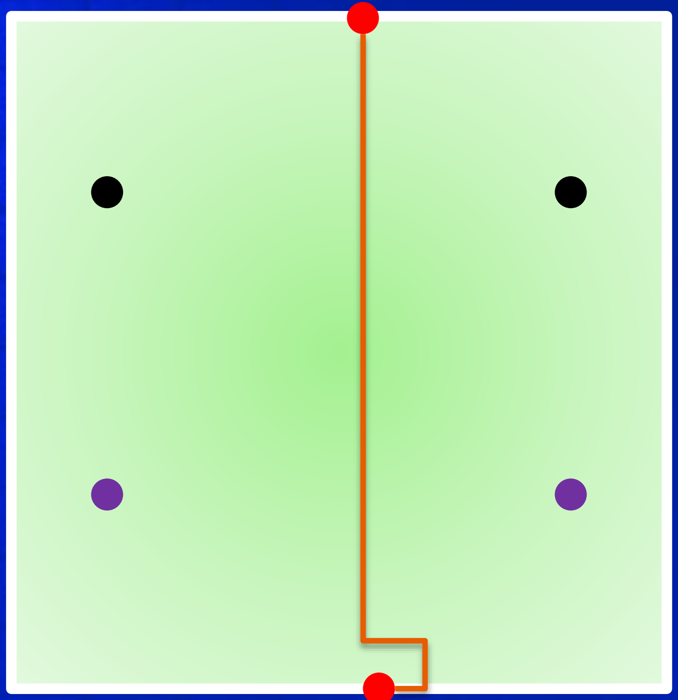
Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet



Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet

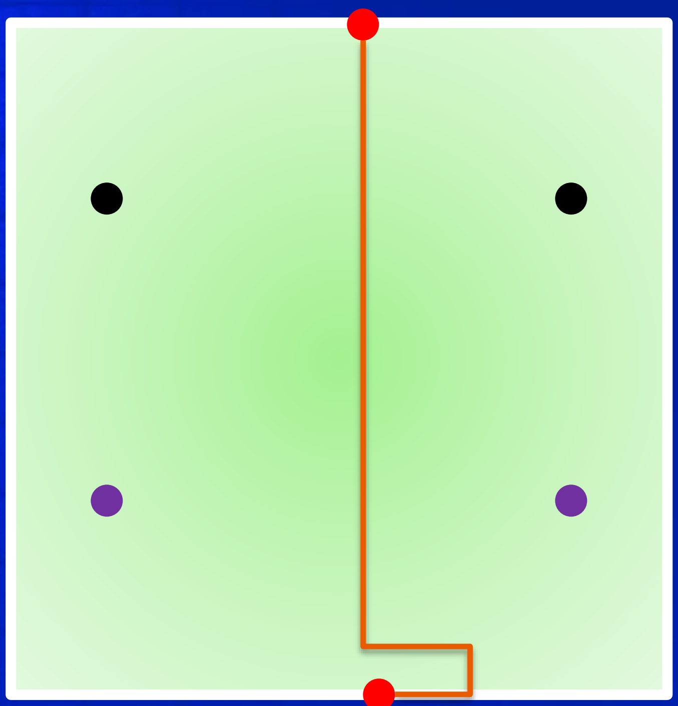


intel Leap ahead™

Design and Technology Solutions

Net Swapping

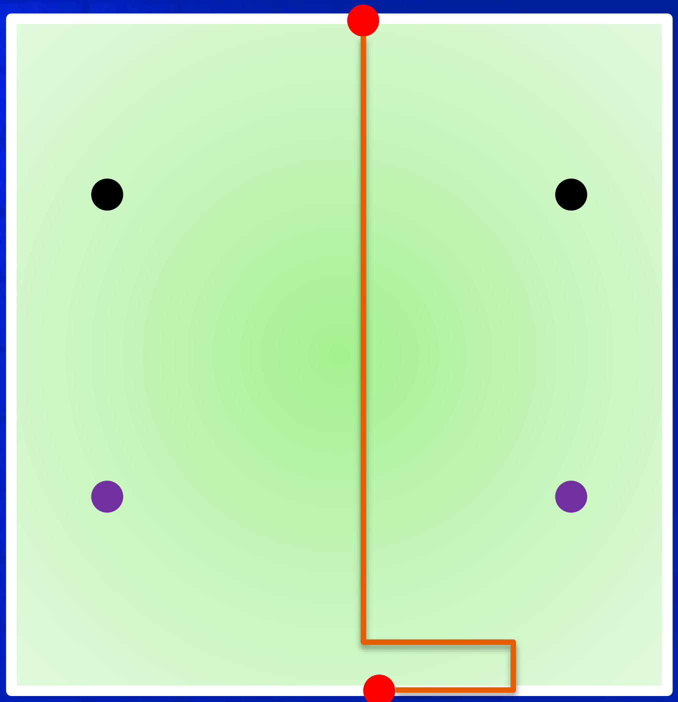
- Example Order 2:
  - Red
  - Black
  - Violet



intel Leap ahead™
Design and Technology Solutions

Net Swapping

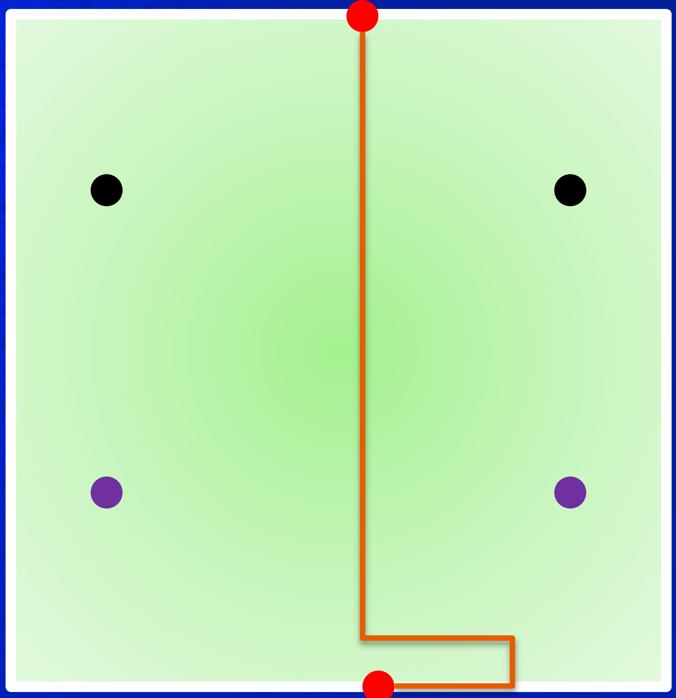
- Example Order 2:
  - Red
  - Black
  - Violet



Design and Technology Solutions

Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet



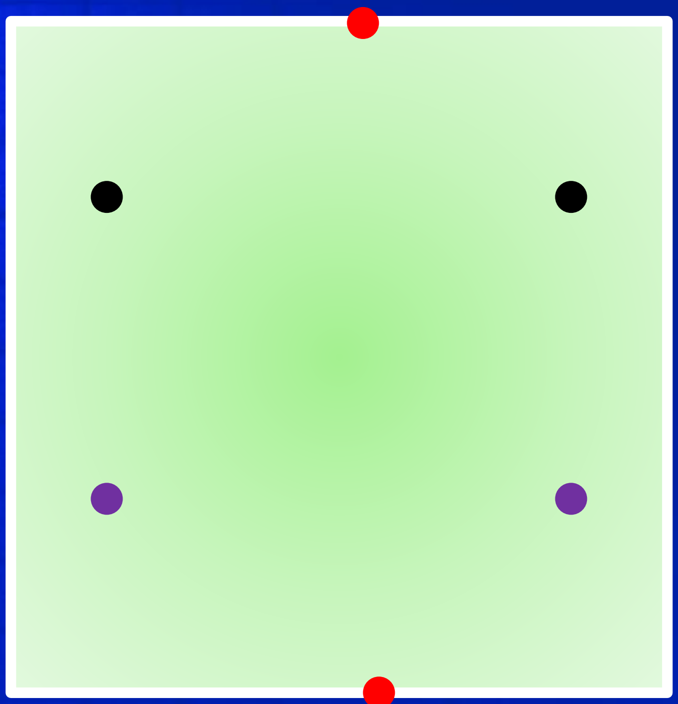
Net Swapping:
After N conflicts, swap the order between:
the first blocked net i
the blocking net j
{A,j,B,i,C} → {A,i,j,B,C}

intel Leap ahead™
Design and Technology Solutions

Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black
  - Red
  - Violet

Swapped



Net Swapping:

After N conflicts, swap the order between:

the first blocked net i

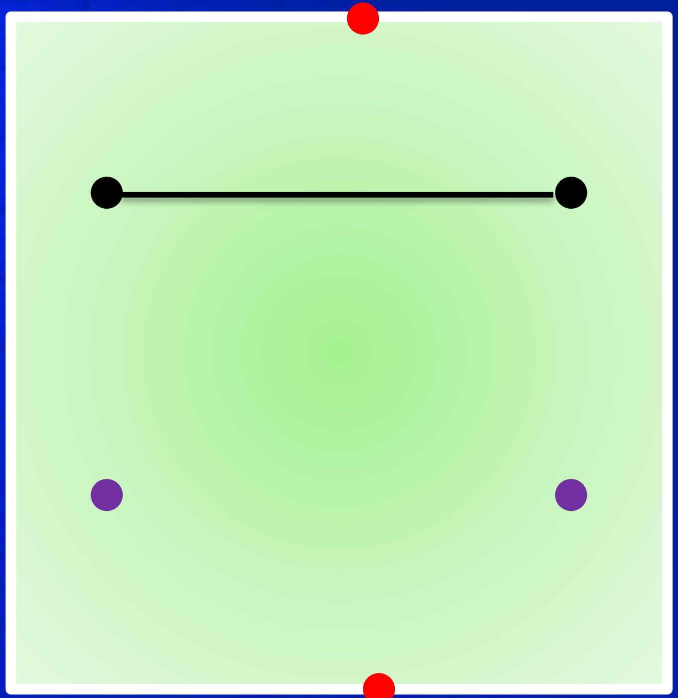
the blocking net j

{A,j,B,i,C} → {A,i,j,B,C}

Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black
  - Red
  - Violet

Swapped

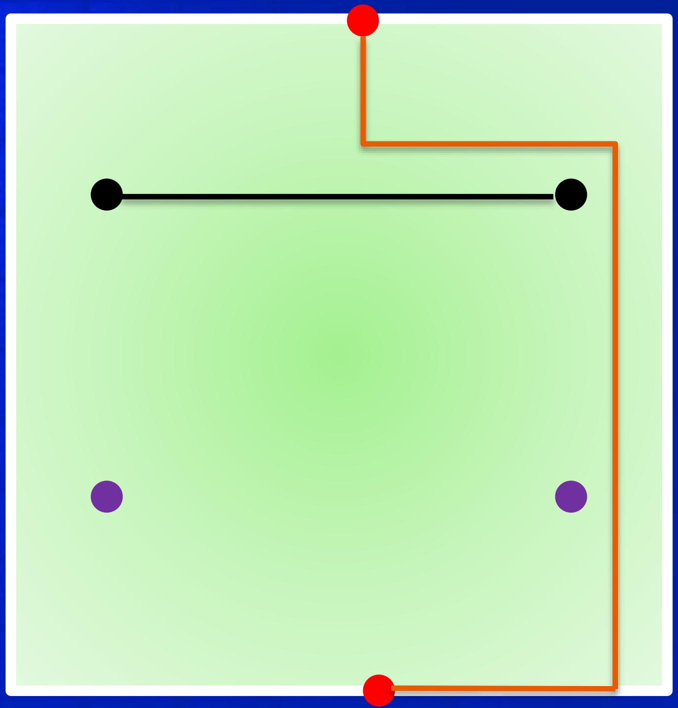


Net Swapping:
After N conflicts, swap the order between:
the first blocked net i
the blocking net j
{A,j,B,i,C} → {A,i,j,B,C}

Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black
  - Red
  - Violet

Swapped



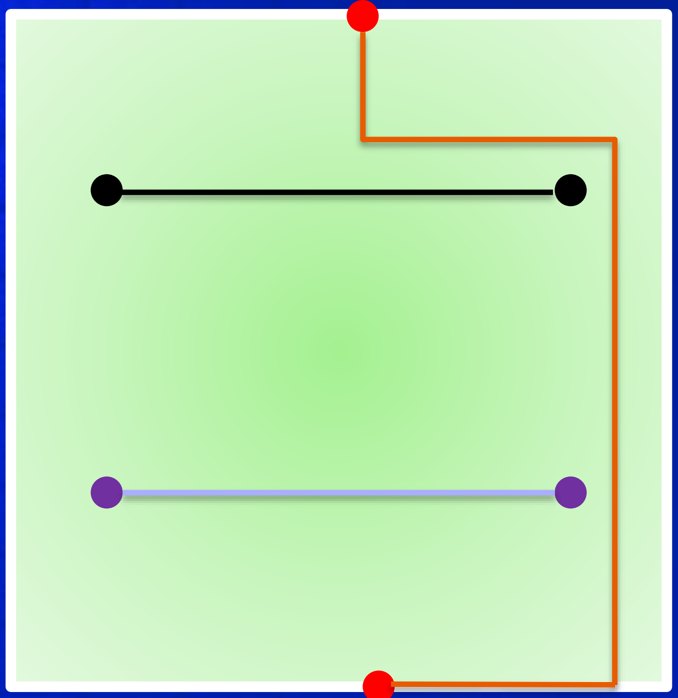
Net Swapping:
After N conflicts, swap the order between:
the first blocked net i
the blocking net j
{A,j,B,i,C} → {A,i,j,B,C}

intel Leap ahead™
Design and Technology Solutions

Net Swapping

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black
  - Red
  - Violet

Swapped



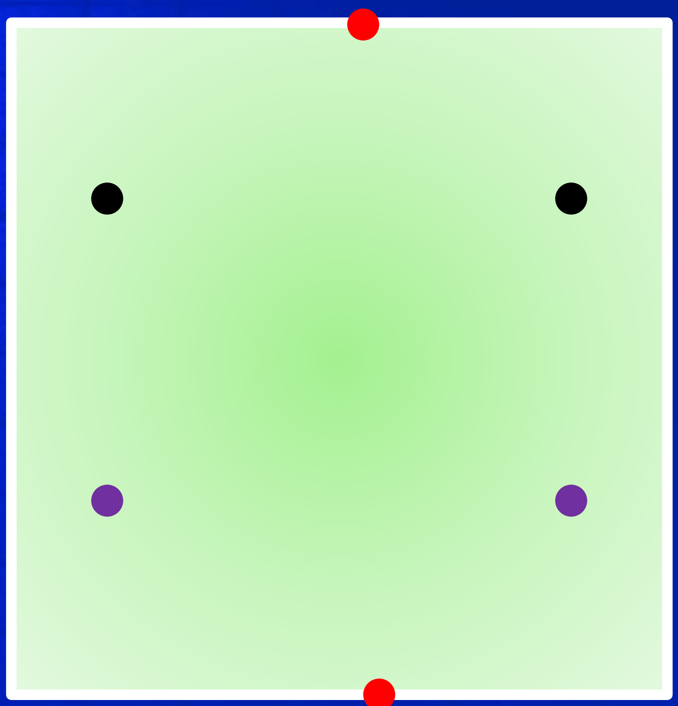
Net Swapping:
After N conflicts, swap the order between:
the first blocked net i
the blocking net j
{A,j,B,i,C} → {A,i,j,B,C}

Leap ahead™

Design and Technology Solutions

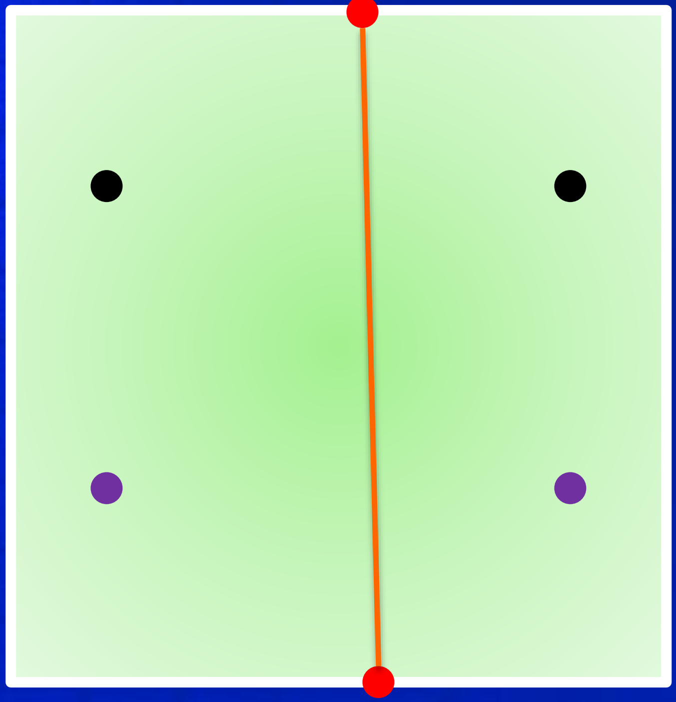
Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet



Net Restarting

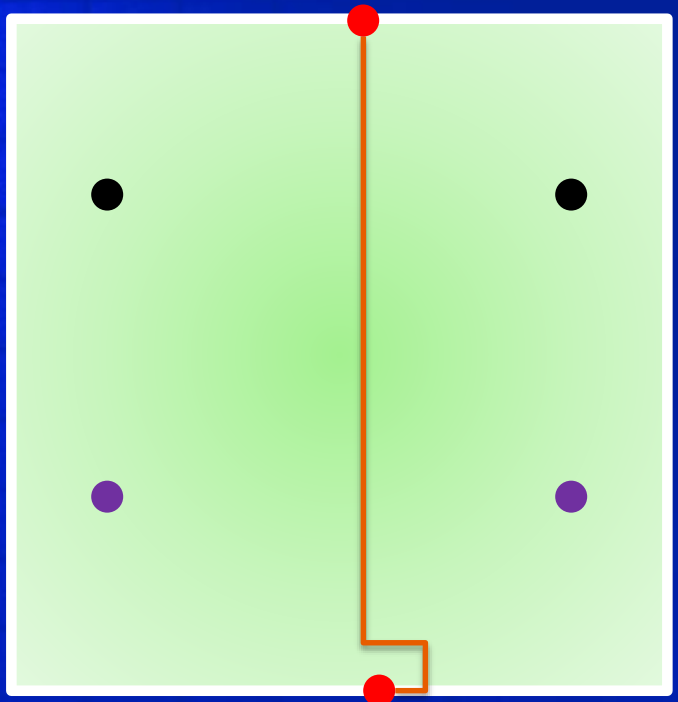
- Example Order 2:
  - Red
  - Black
  - Violet



Design and Technology Solutions

Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet

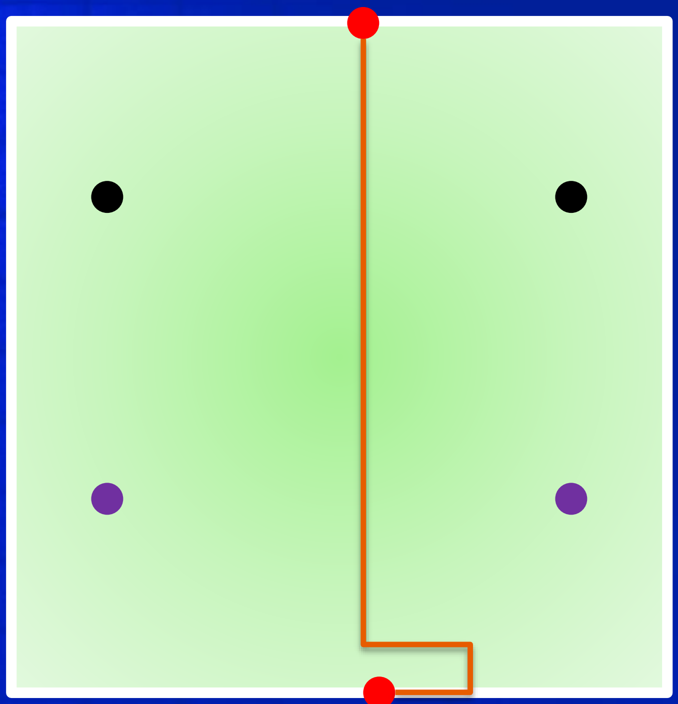


intel Leap ahead™

Design and Technology Solutions

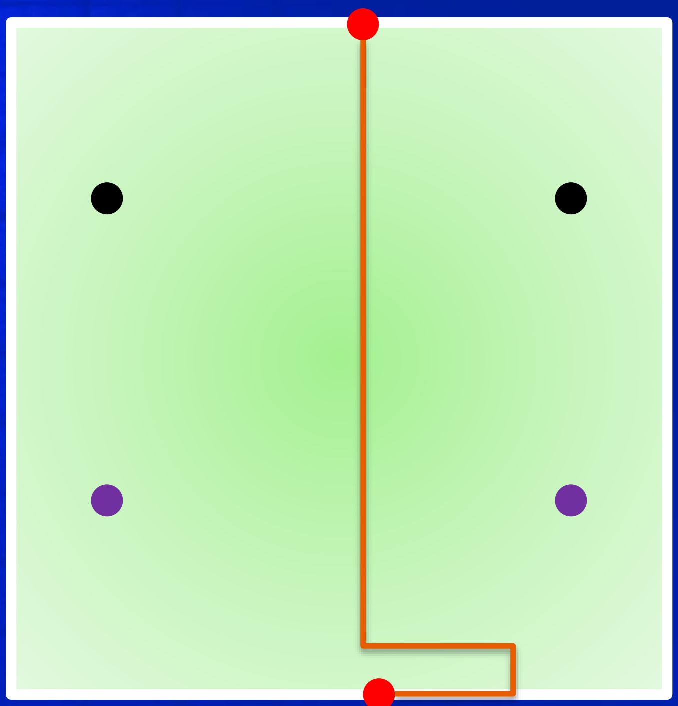
Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet



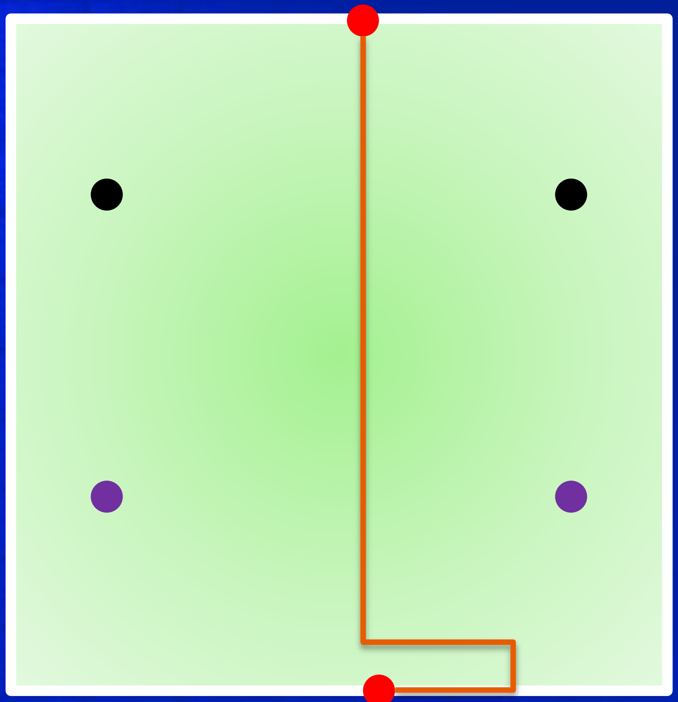
Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet



Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet



Net Restarting

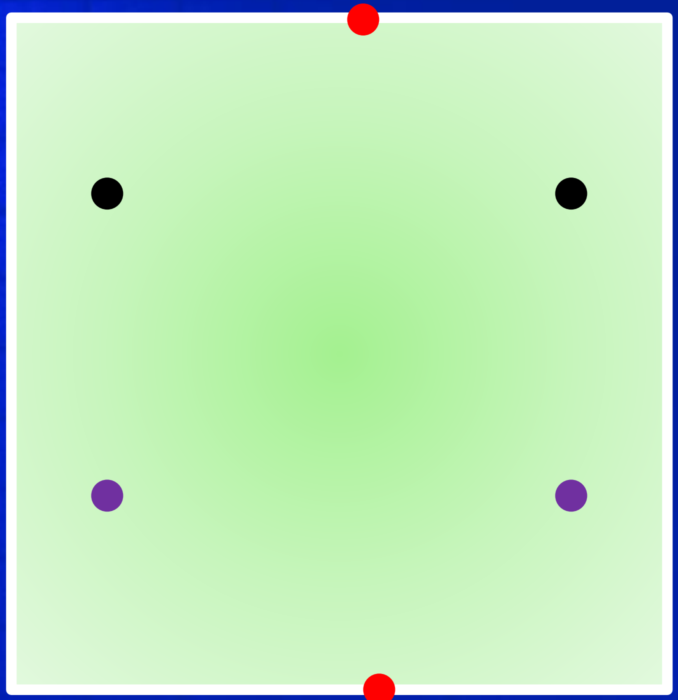
Restart and move the blocked net to the top (after M conflicts for that net)

Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black ← Moved to the top
  - Red
  - Violet

Net Restarting

Restart and move the blocked net to the top (after M conflicts for that net)

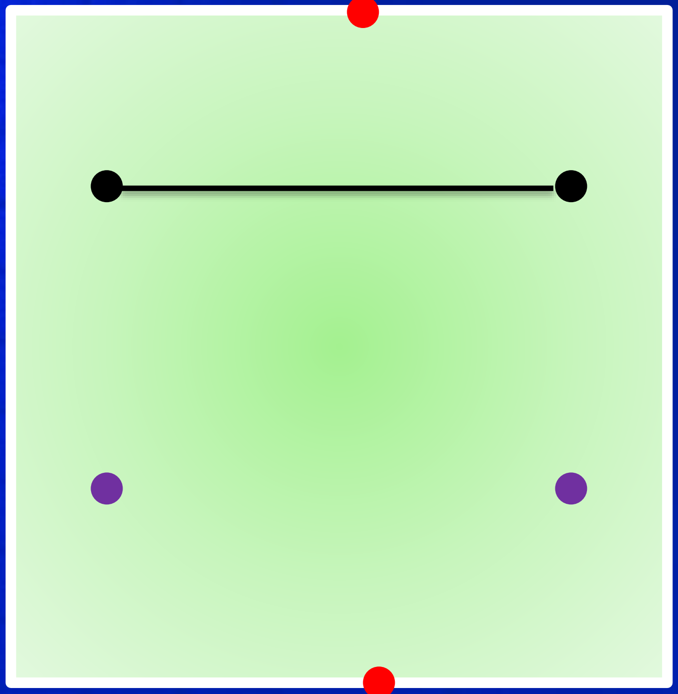


Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black ← Moved to the top
  - Red
  - Violet

Net Restarting

Restart and move the blocked net to the top (after M conflicts for that net)



Design and Technology Solutions

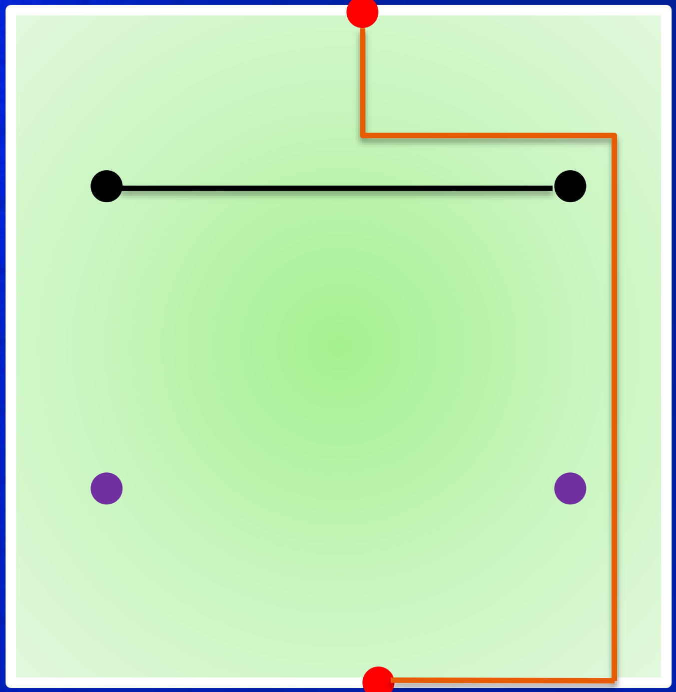
Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black
  - Red
  - Violet

Moved to the top

Net Restarting

Restart and move the blocked net to the top (after M conflicts for that net)



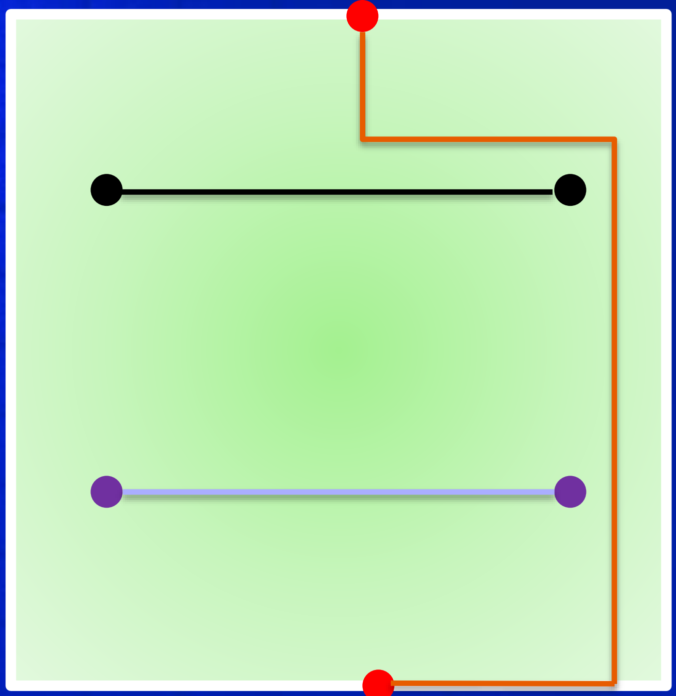
intel Leap ahead™

Design and Technology Solutions

Net Restarting

- Example Order 2:
  - Red
  - Black
  - Violet
- Flip:
  - Black ← Moved to the top
  - Red
  - Violet

Net Restarting
Restart and move the blocked net to the top (after M conflicts for that net)



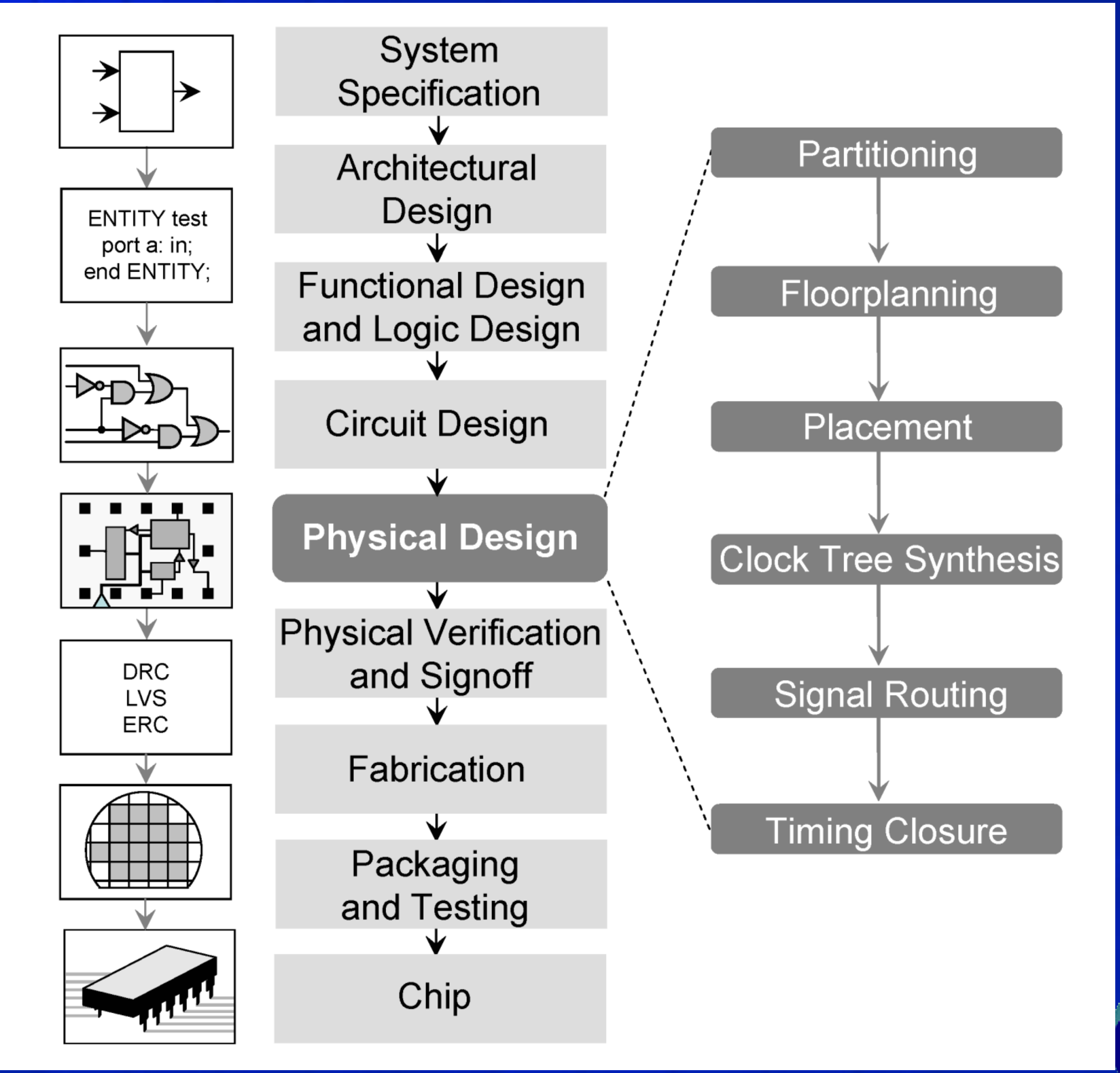
Design and Technology Solutions

Net Swapping vs. Net Restarting

- Swapping is local
- Restarting is global
- In practice both techniques are crucial
- Strategy:
  - Swap for some time
  - If it doesn't work, restart

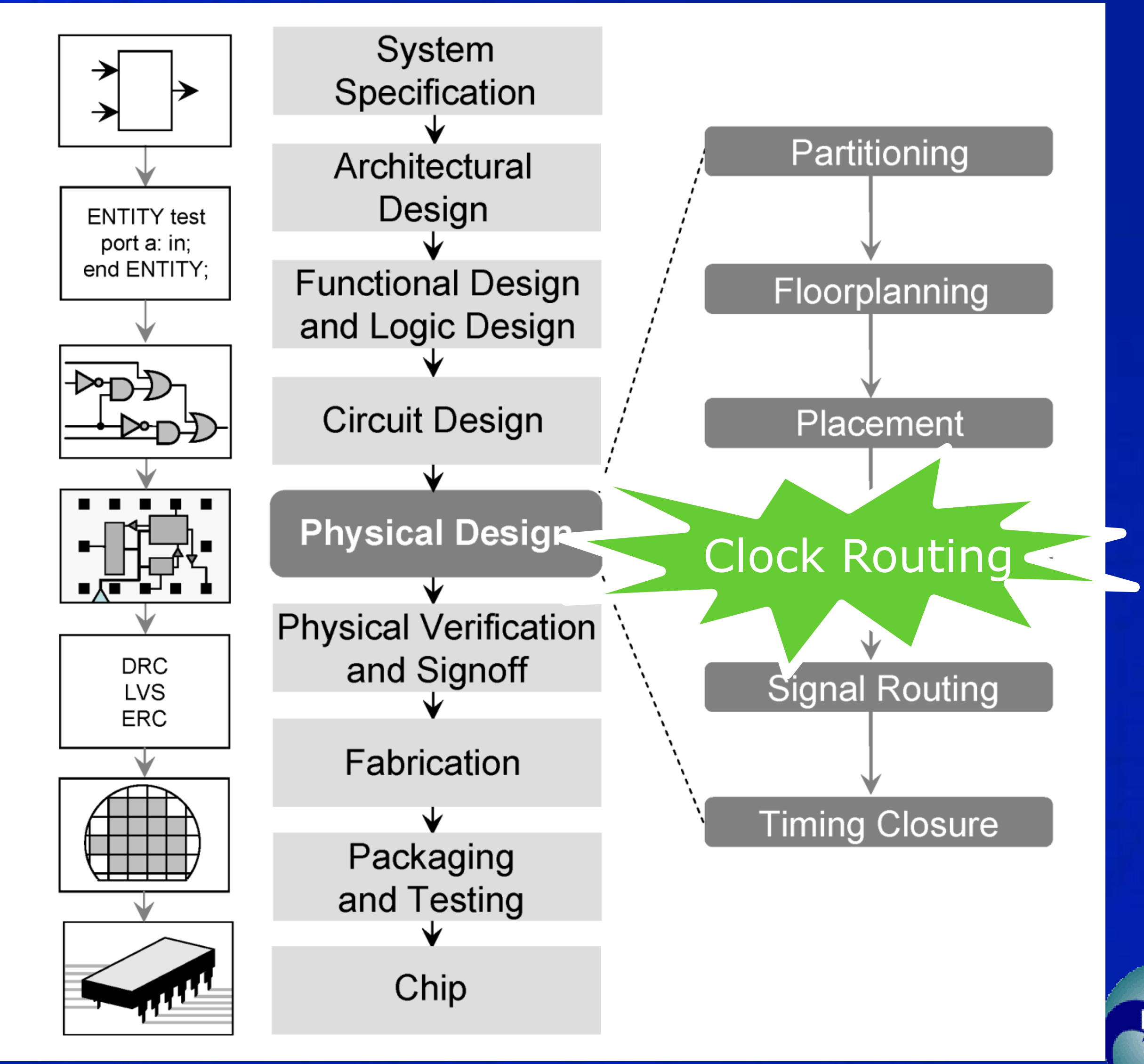
Related Work 1: Clock Routing
Erez & Nadel, CAV'15

Related Work 1: Clock Routing
Erez & Nadel, CAV'15



Design and Technology Solutions

Related Work 1: Clock Routing
Erez & Nadel, CAV'15



intel Leap ahead
Design and Technology Solutions

Related Work 1: Clock Routing
Erez & Nadel, CAV'15

Related Work 1: Clock Routing
Erez & Nadel, CAV'15

- Reduction to finding bounded-path in graph

intel Leap ahead™
Design and Technology Solutions

Related Work 1: Clock Routing
Erez & Nadel, CAV'15

- Reduction to finding bounded-path in graph
- SAT solver surgery: graph-aware decision strategy & graph conflict analysis

Design and Technology Solutions

Related Work 1: Clock Routing
Erez & Nadel, CAV'15

- Reduction to finding bounded-path in graph
- SAT solver surgery: graph-aware decision strategy & graph conflict analysis
- The decision strategy:
  - Emulates constraints!
  - Guides the solver towards the solution
  - Considers additional optimization requirements

Related Work 2: Monosat Solver
Bayless & Bayless & Hoos & Hu, AAAI'15

Related Work 2: Monosat Solver
Bayless & Bayless & Hoos & Hu, AAAI'15

- Can reason about graph predicates & SAT/BV

Related Work 2: Monosat Solver
Bayless & Bayless & Hoos & Hu, AAAI'15

- Can reason about graph predicates & SAT/BV
- Graph conflict analysis

intel Leap ahead™
Design and Technology Solutions

Related Work 2: Monosat Solver
Bayless & Bayless & Hoos & Hu, AAAI'15

- Can reason about graph predicates & SAT/BV
- Graph conflict analysis
- Shortest-path decision heuristic can be optionally applied

Related Work 2: Monosat Solver
Bayless & Bayless & Hoos & Hu, AAAI'15

- Can reason about graph predicates & SAT/BV
- Graph conflict analysis
- Shortest-path decision heuristic can be optionally applied
- Path-finding (routing for one 2-terminal net) is conceptually similar in Monosat and DRouter
  - Main difference:
    - Lazy A* in DRouter vs.
    - Eager incremental Ramalingam-Reps in Monosat
- RUC can be easily expressed in Monosat language

Monosat vs. DRouter for Routing under Constraints

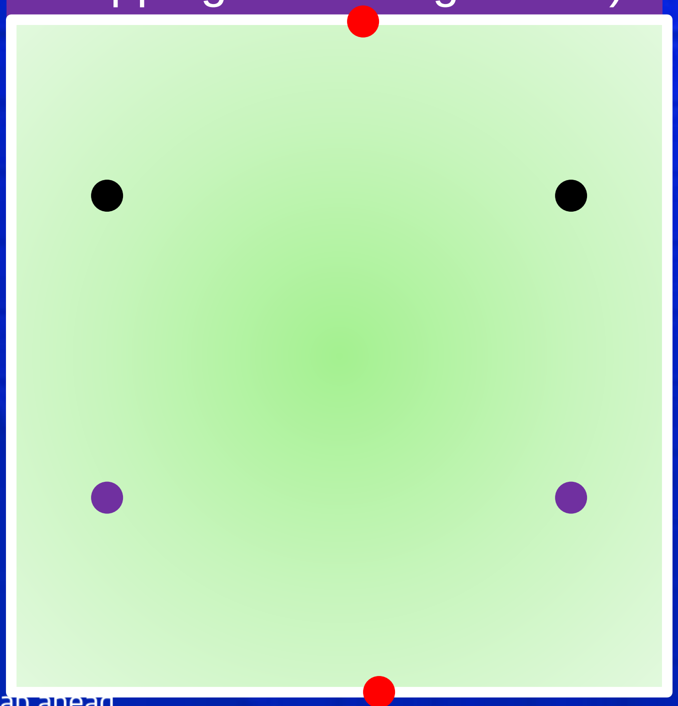
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)

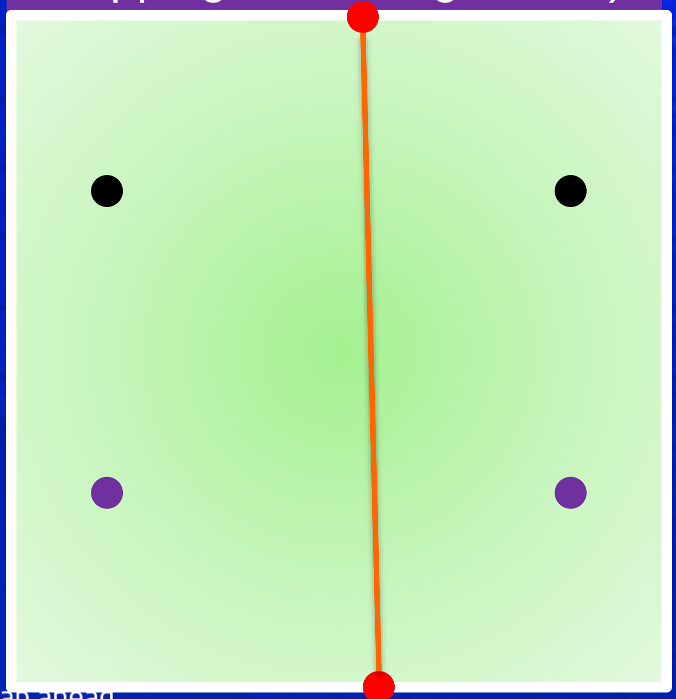


intel Leap ahead
Design and Technology Solutions

Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

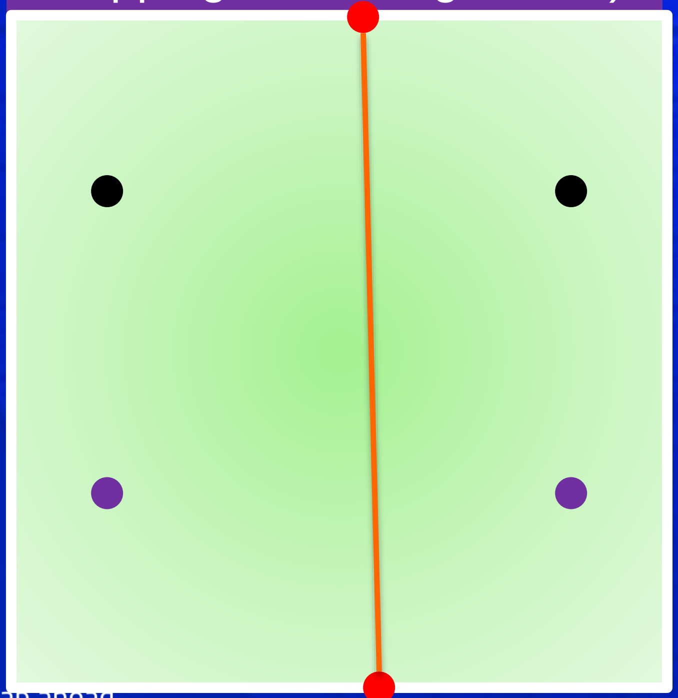
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

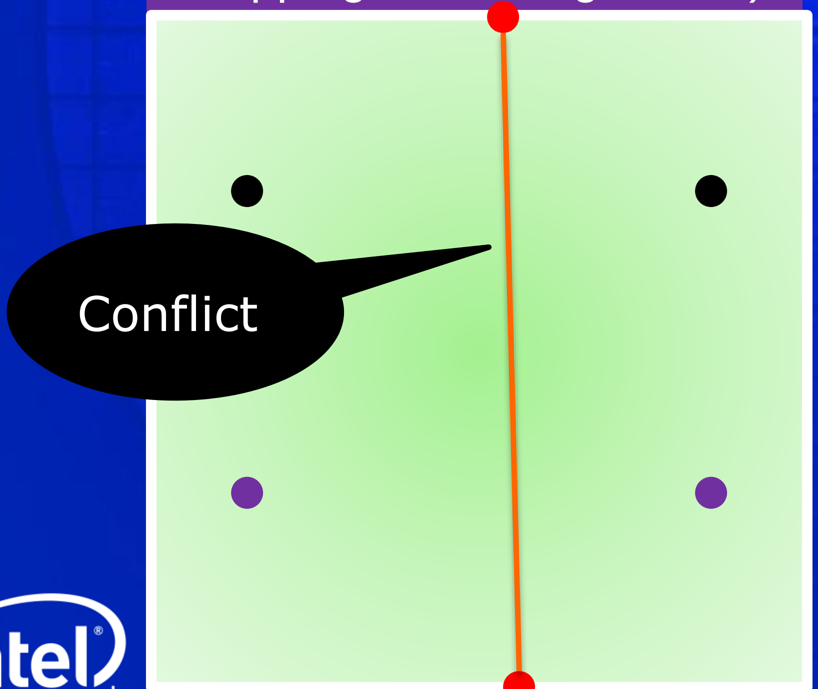
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

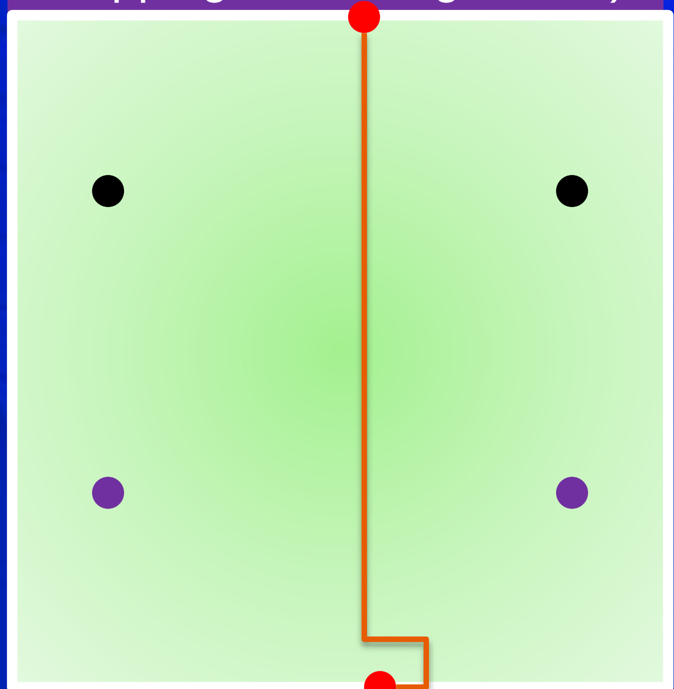
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)

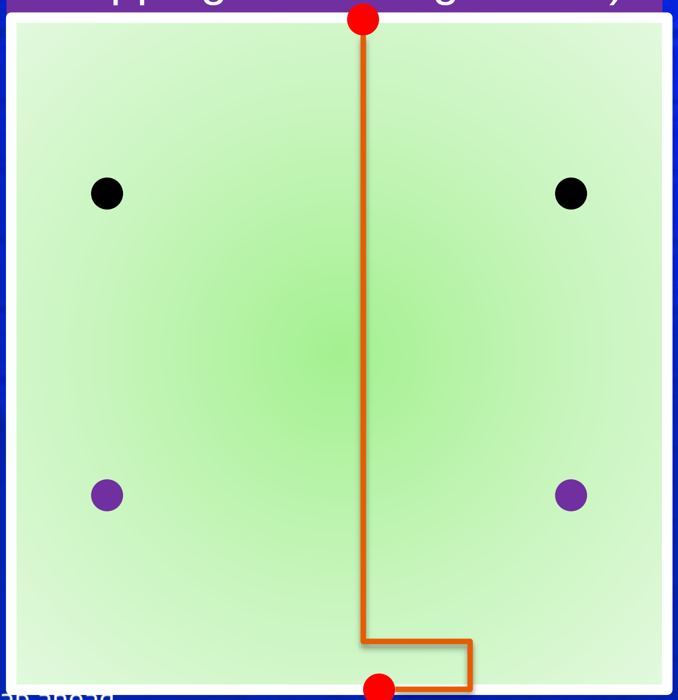


Leap ahead
Design and Technology Solutions

Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

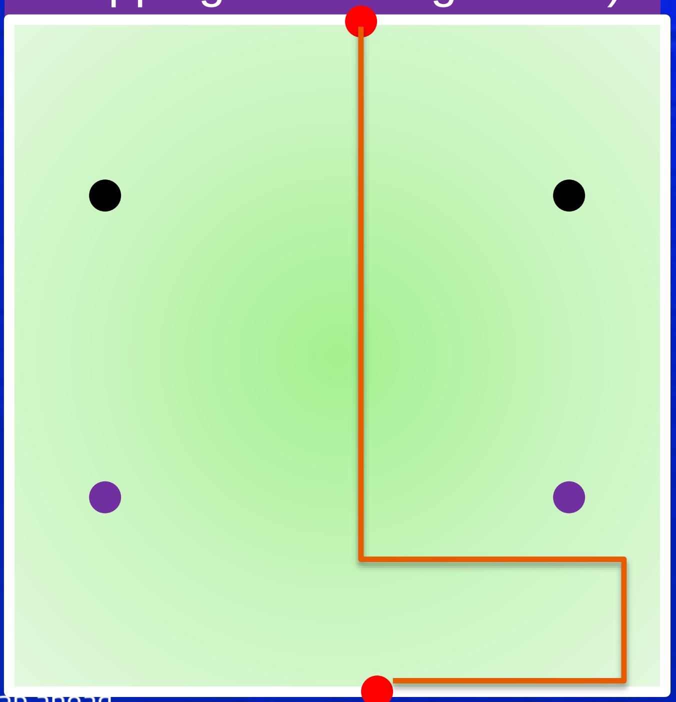
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

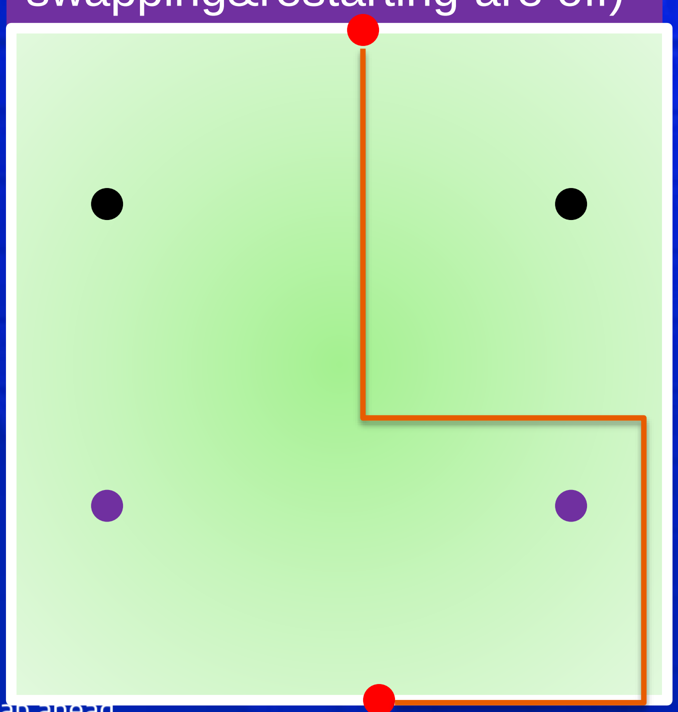
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

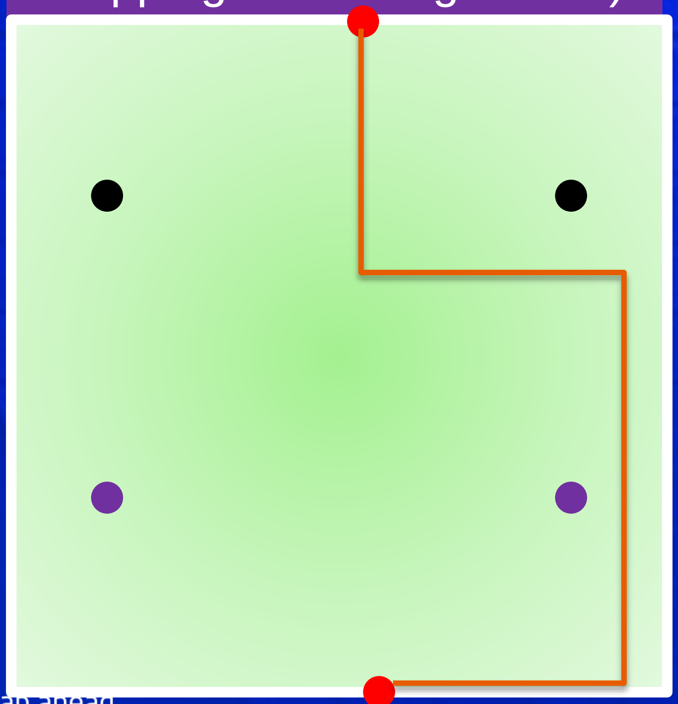
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

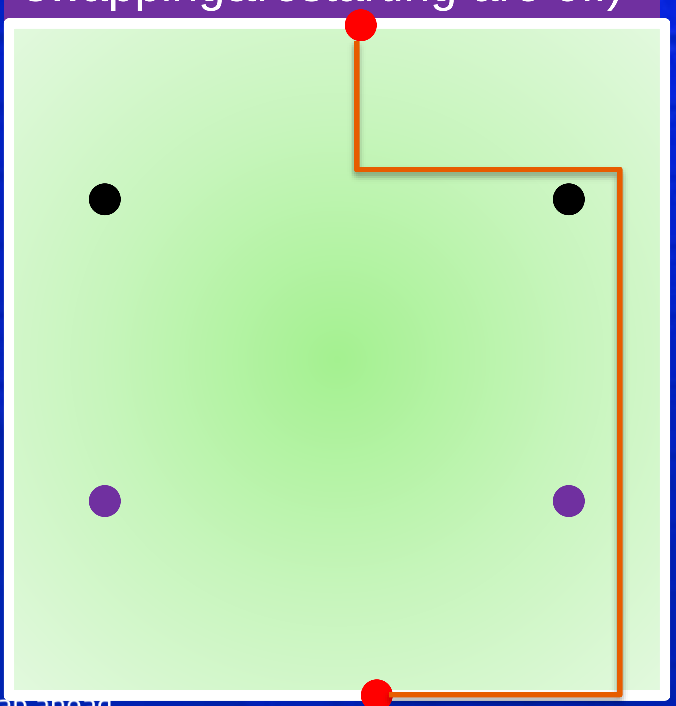
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

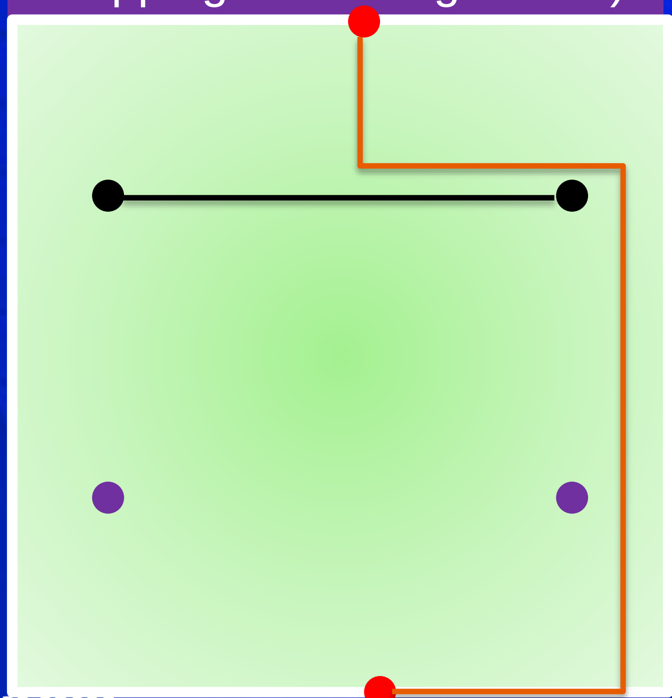
Routing in DRouter (net swapping&restarting are off)



Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)

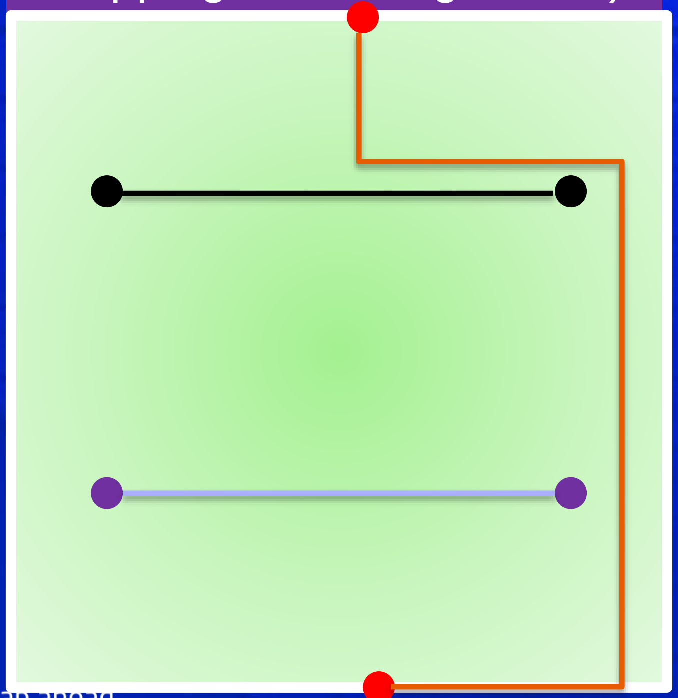


intel Leap ahead
Design and Technology Solutions

Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

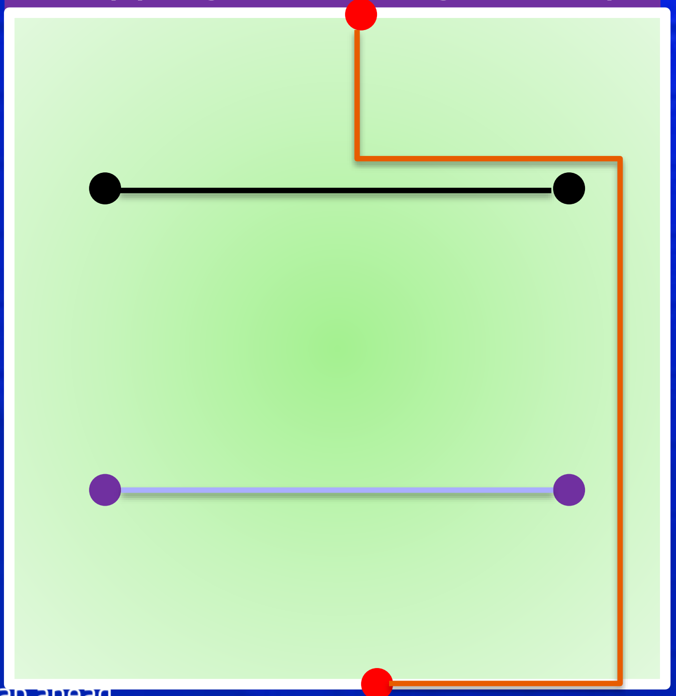
Routing in DRouter (net swapping&restarting are off)



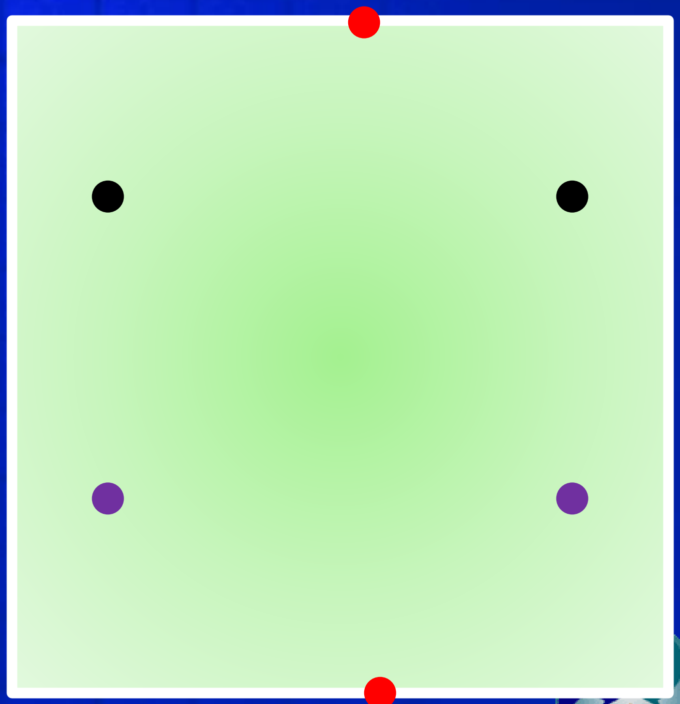
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



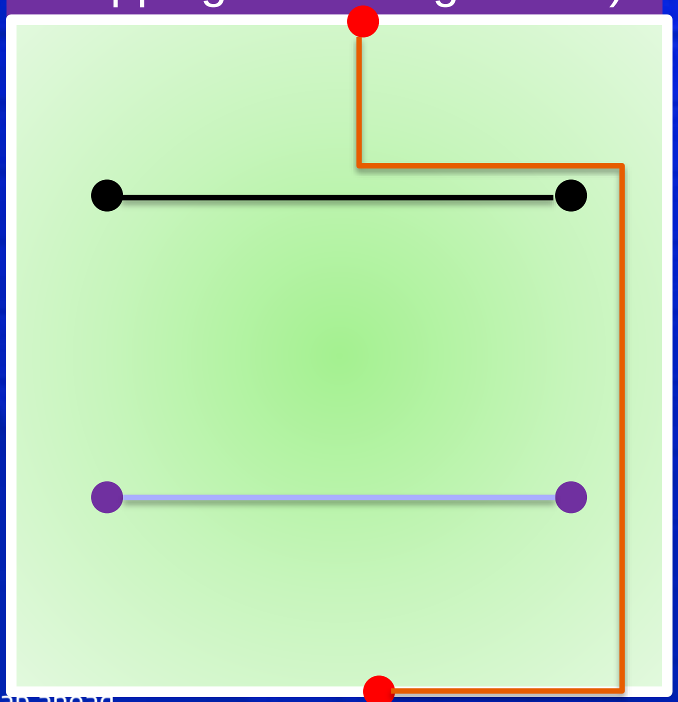
Routing in Monosat



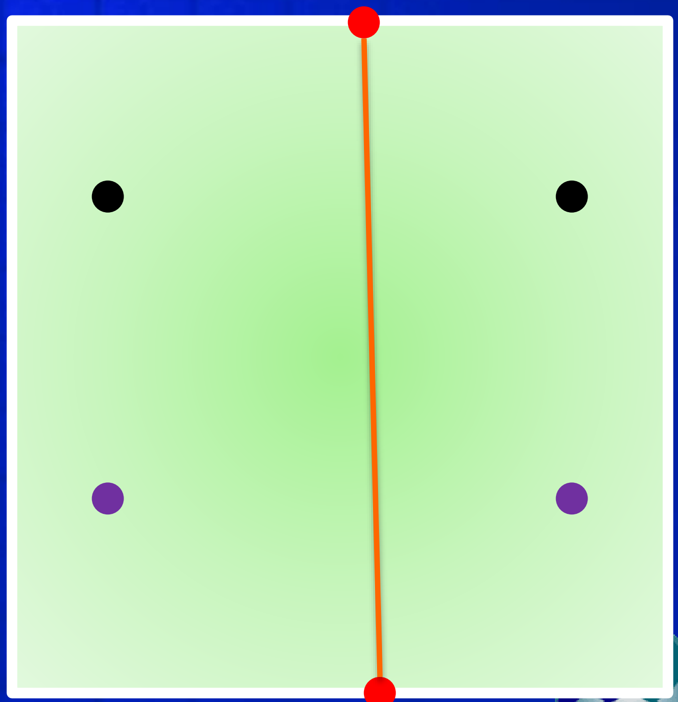
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat



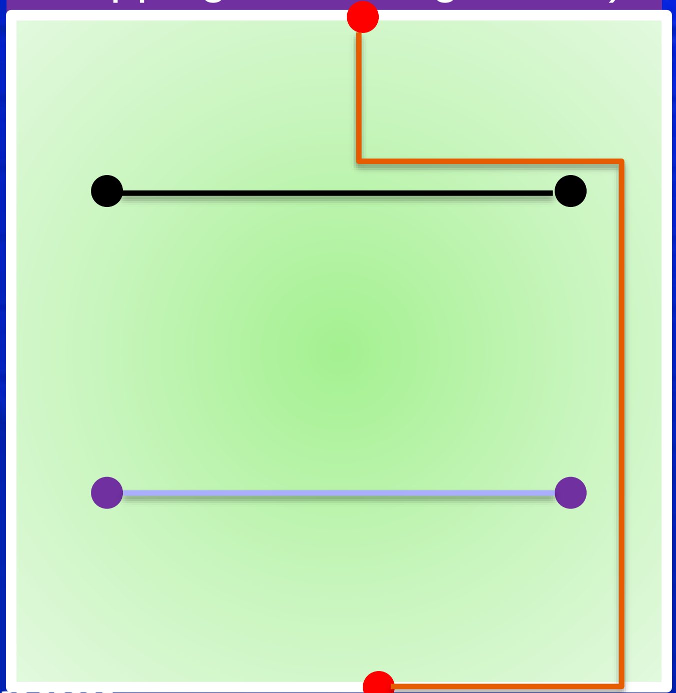
intel Leap ahead

Design and Technology Solutions

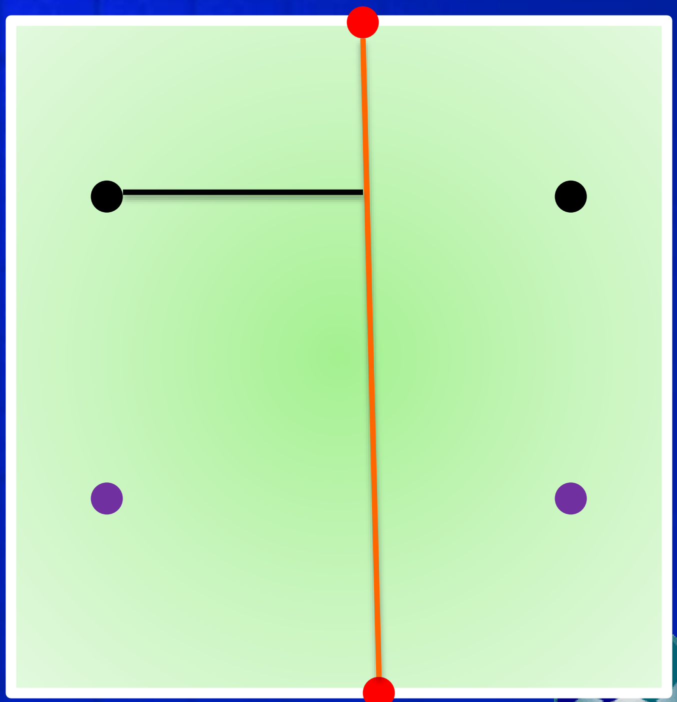
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat

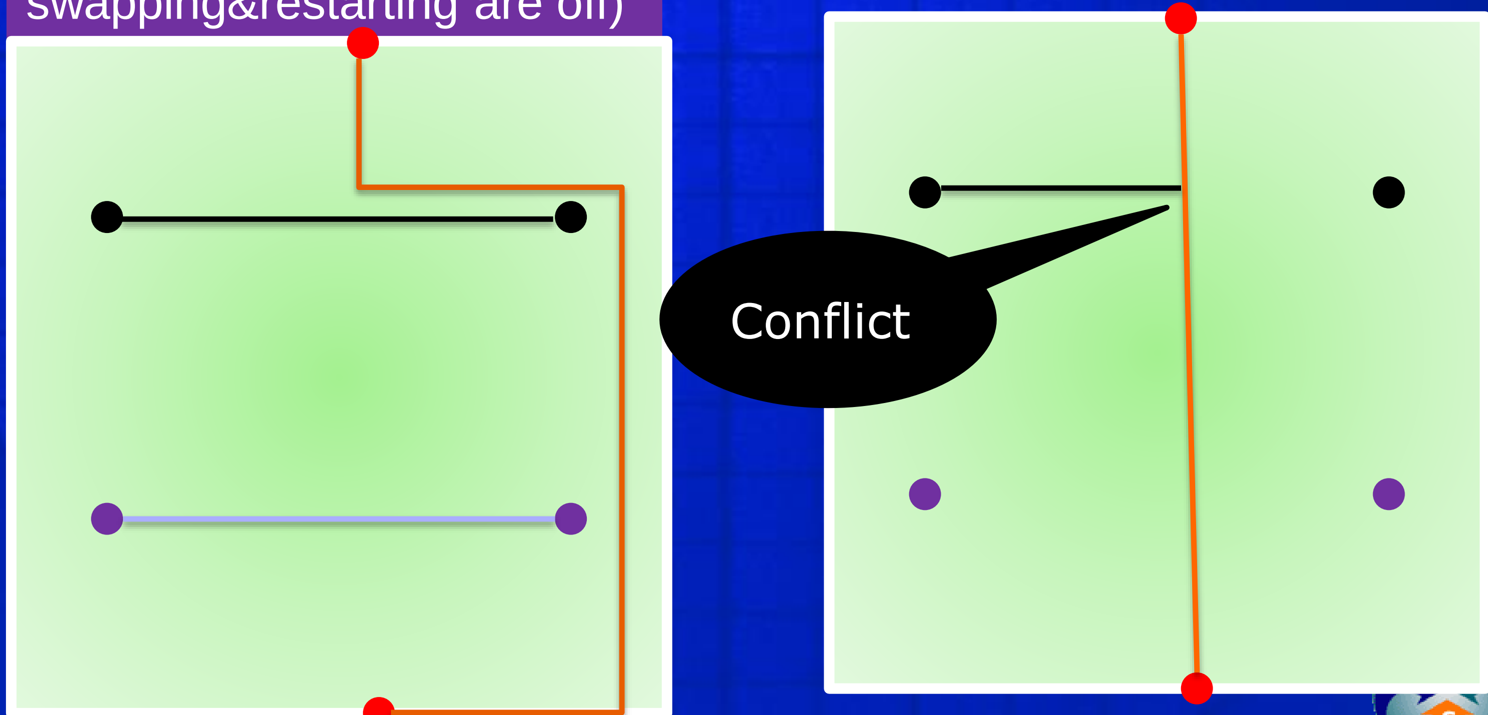


Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)

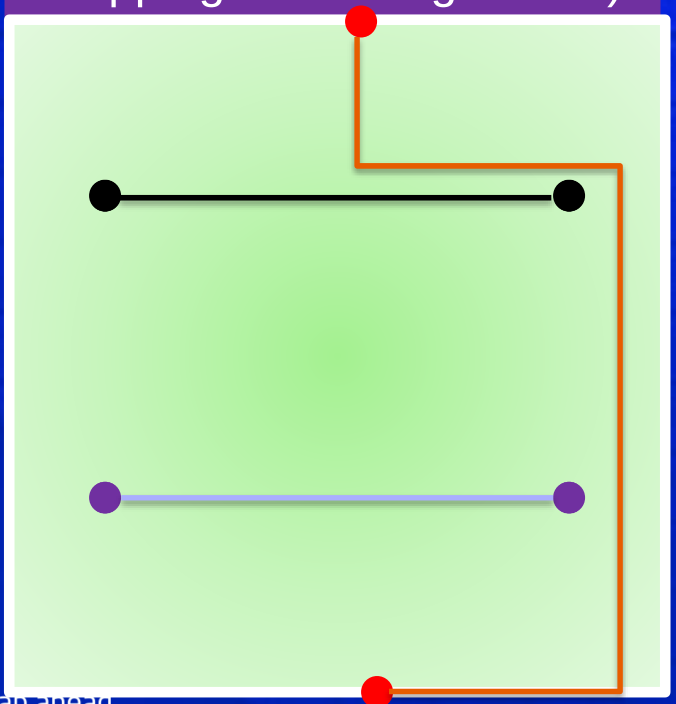
Routing in Monosat



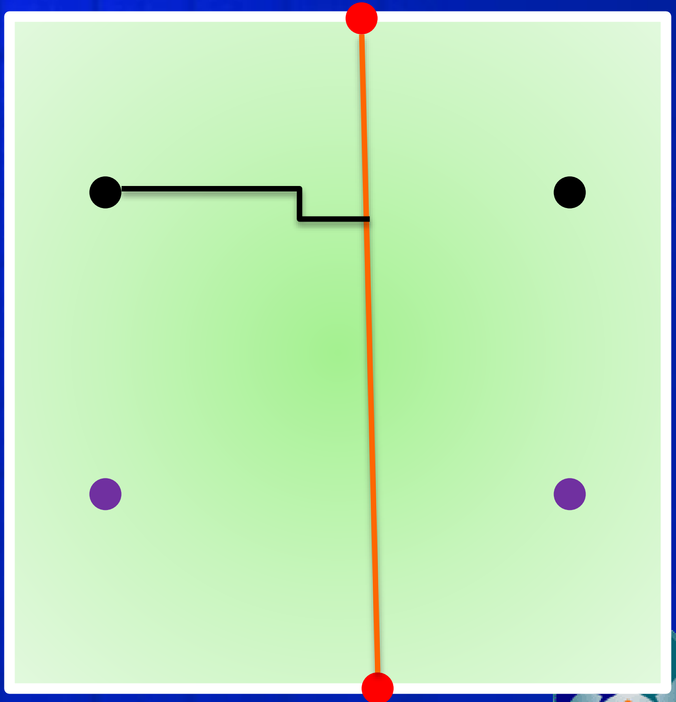
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



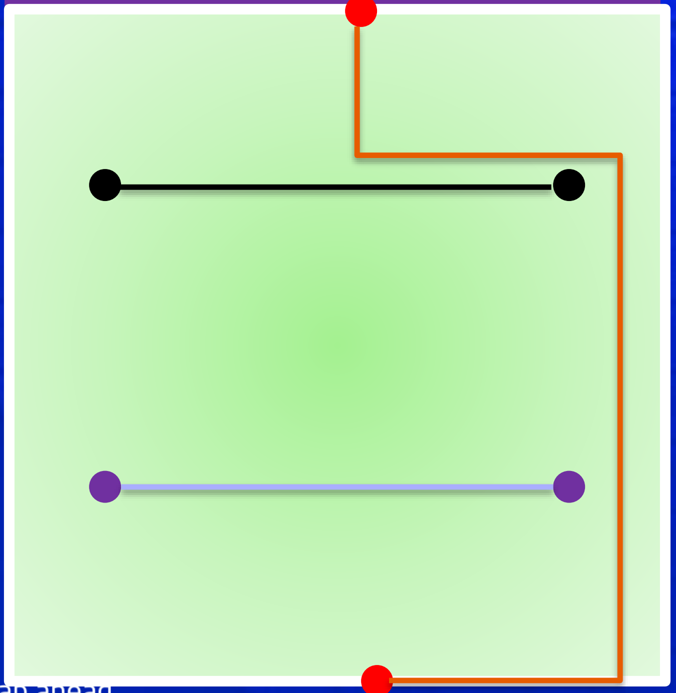
Routing in Monosat



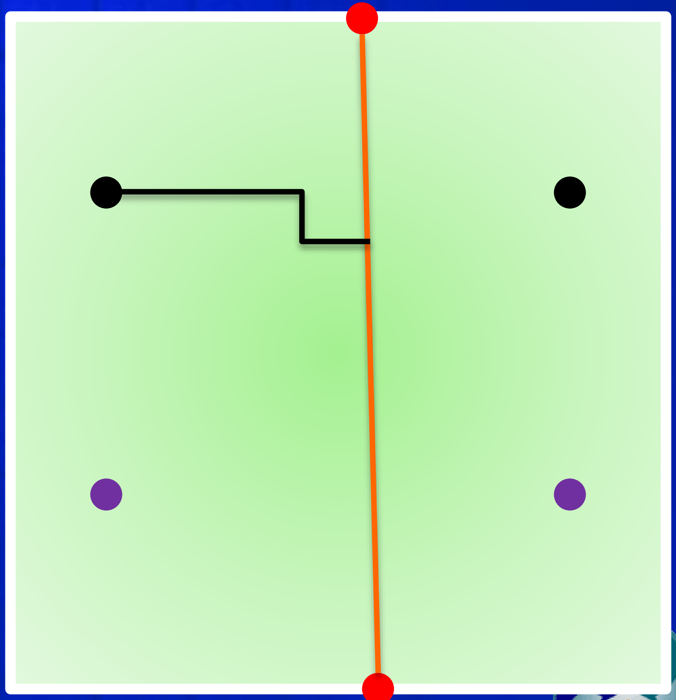
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat

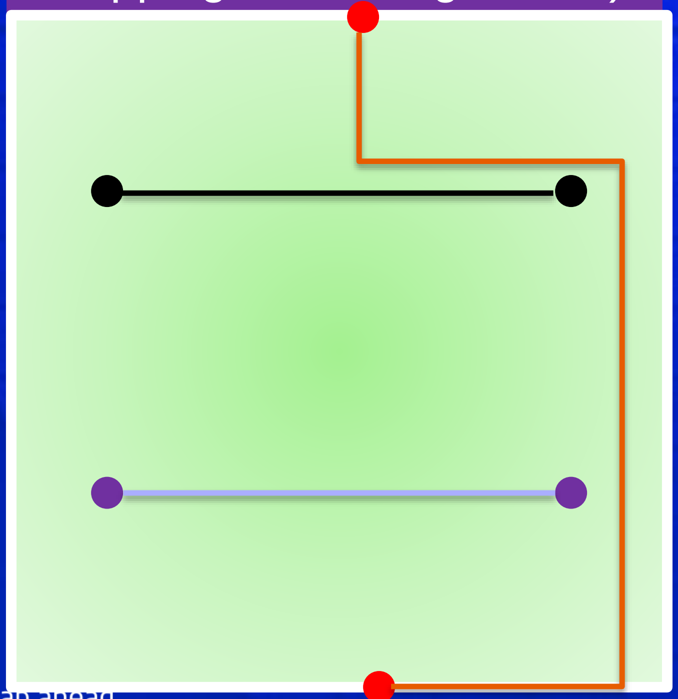


Leap ahead
Design and Technology Solutions

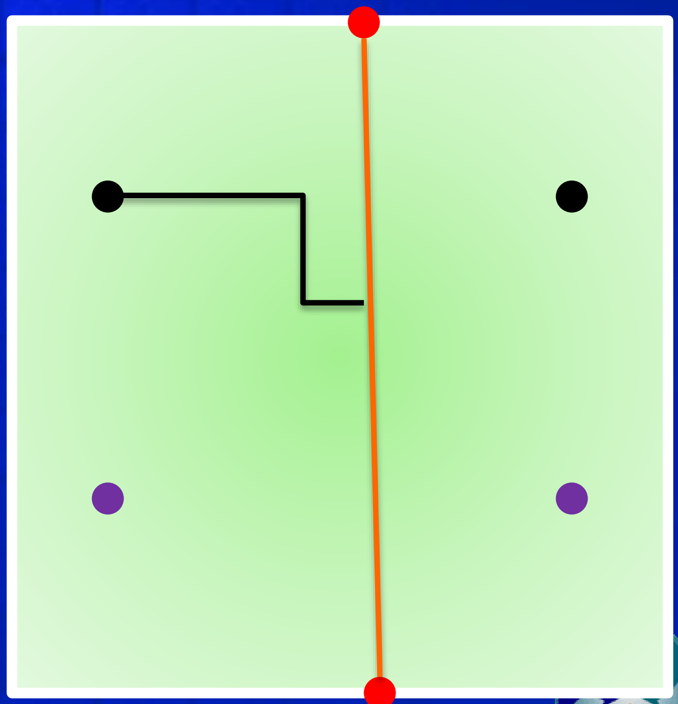
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



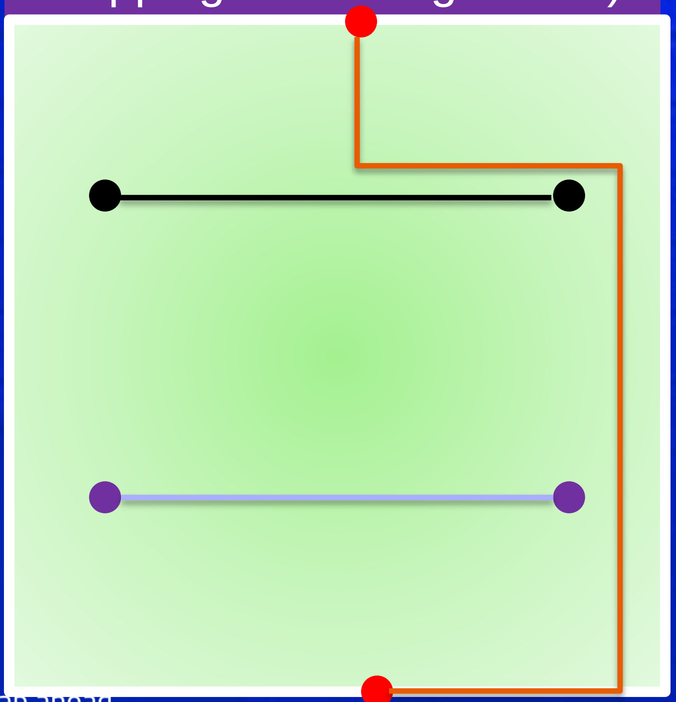
Routing in Monosat



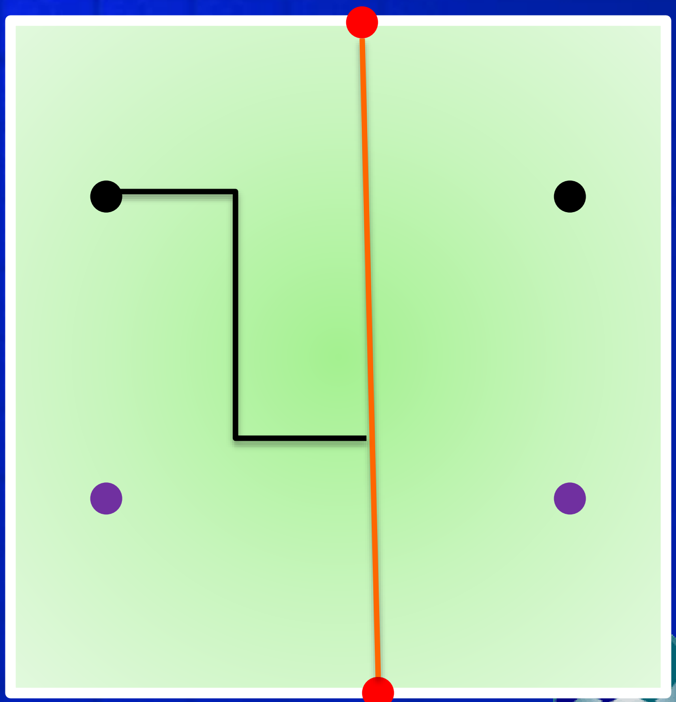
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



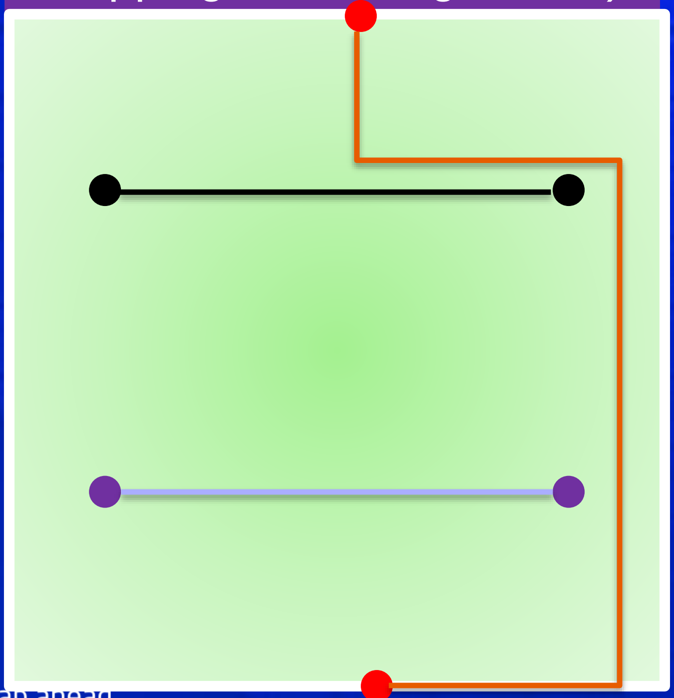
Routing in Monosat



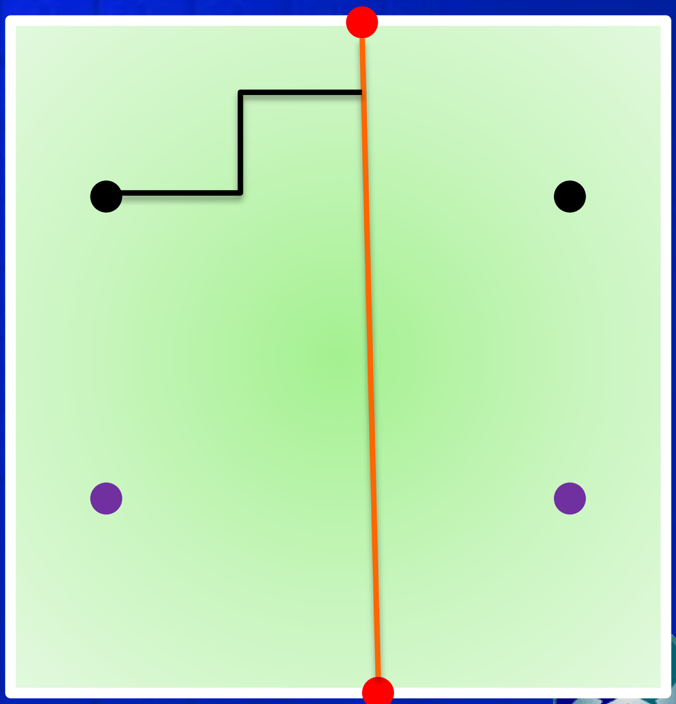
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



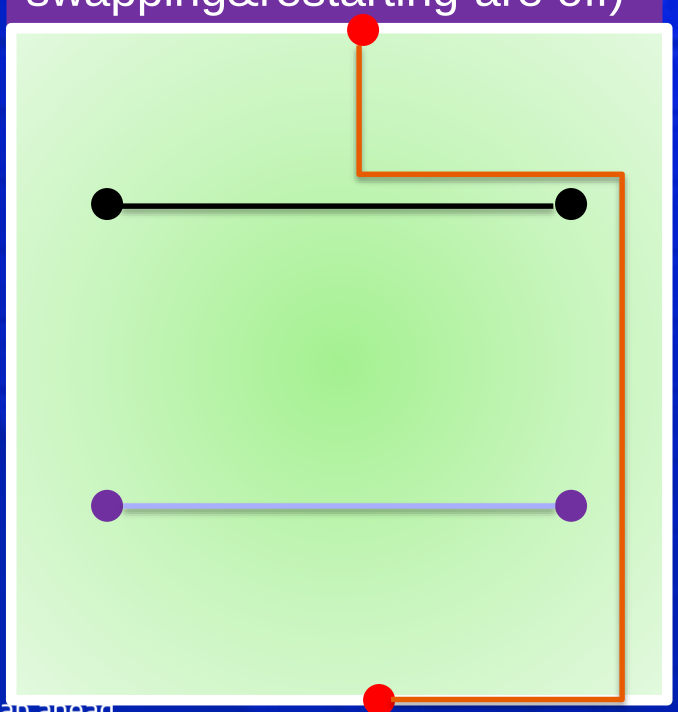
Routing in Monosat



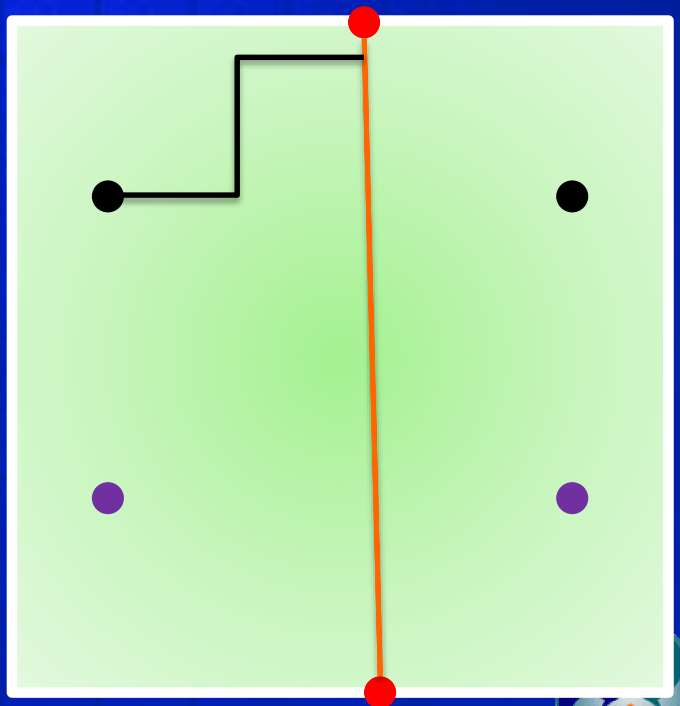
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat

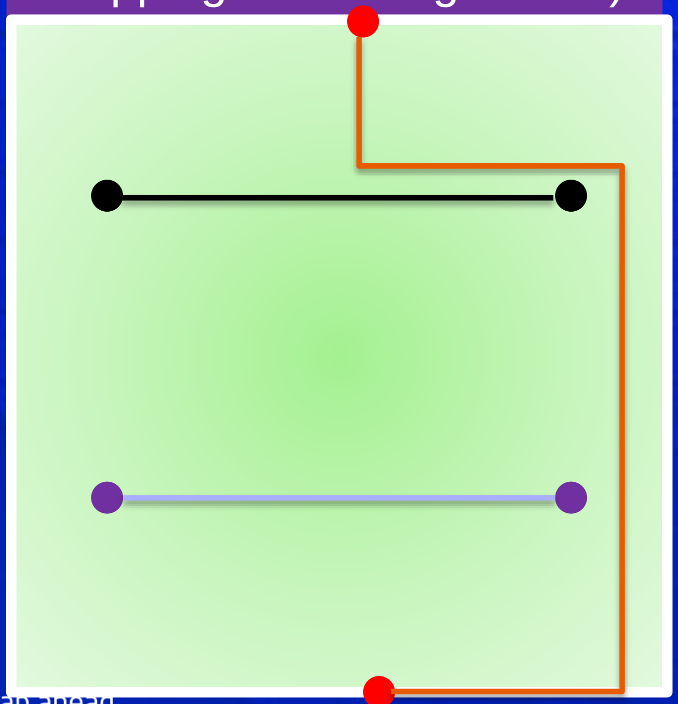


Design and Technology Solutions

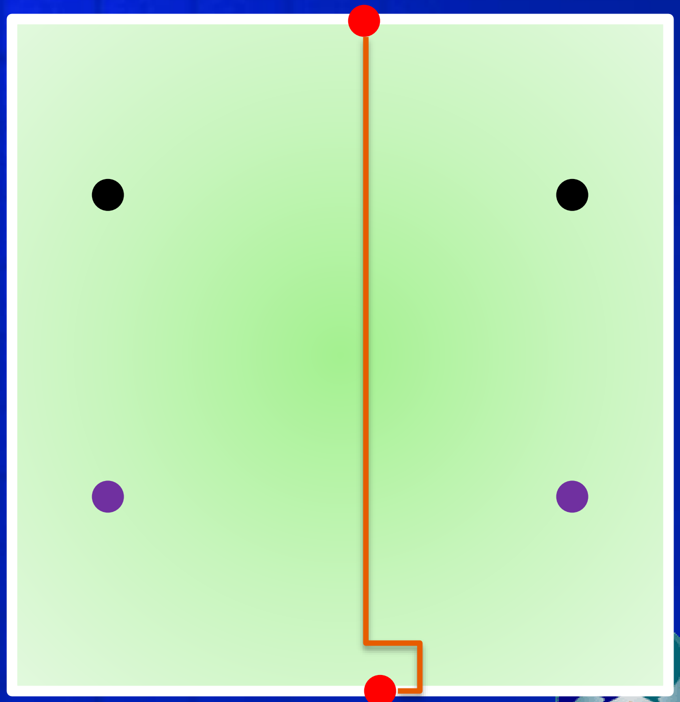
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat



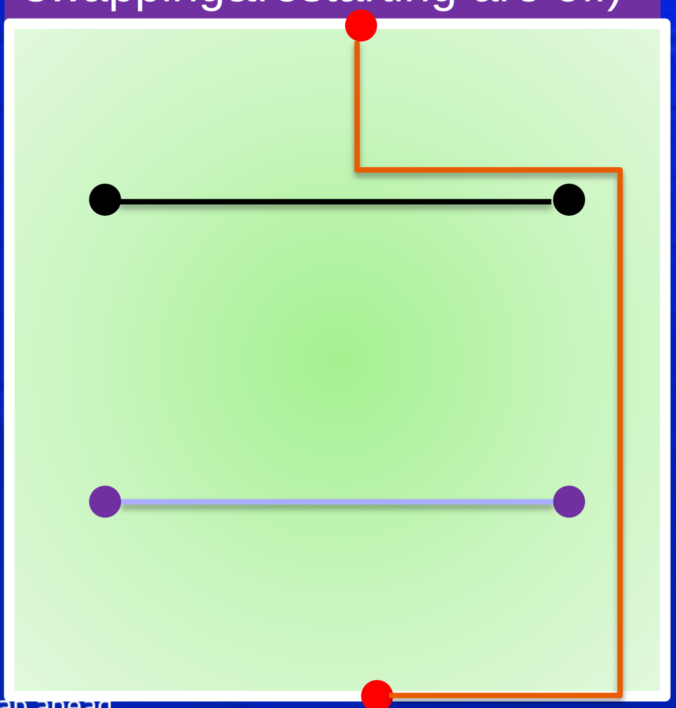
intel Leap ahead

Design and Technology Solutions

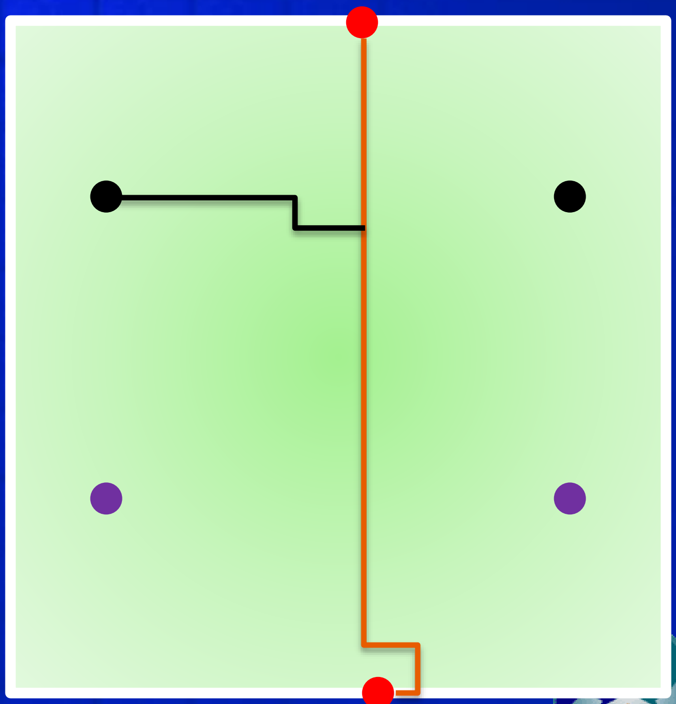
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



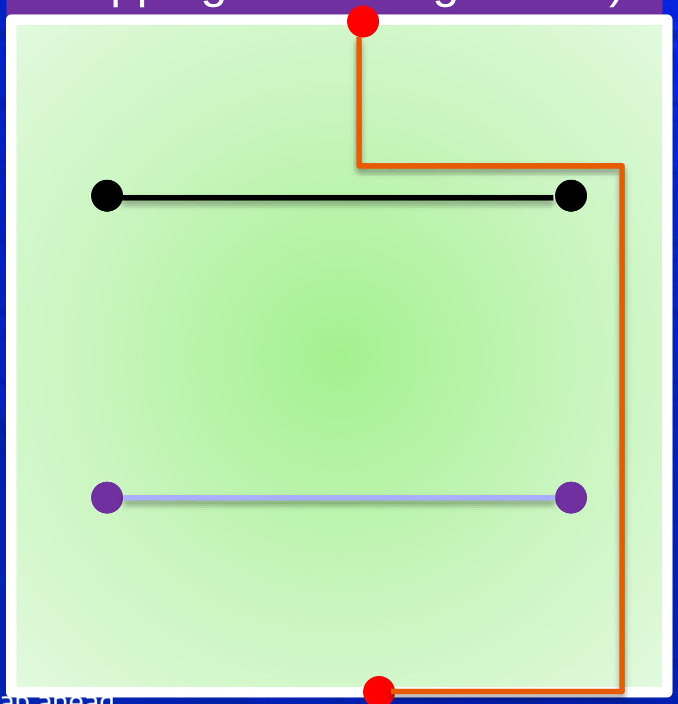
Routing in Monosat



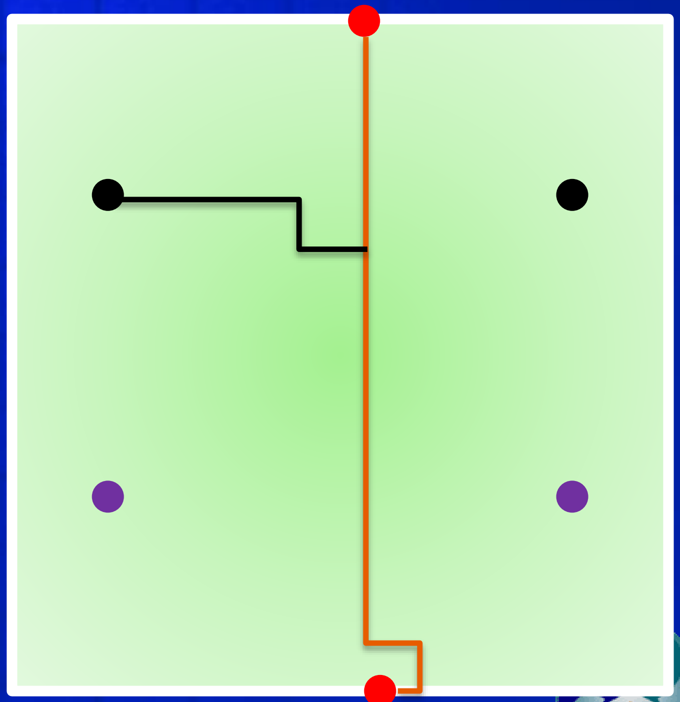
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



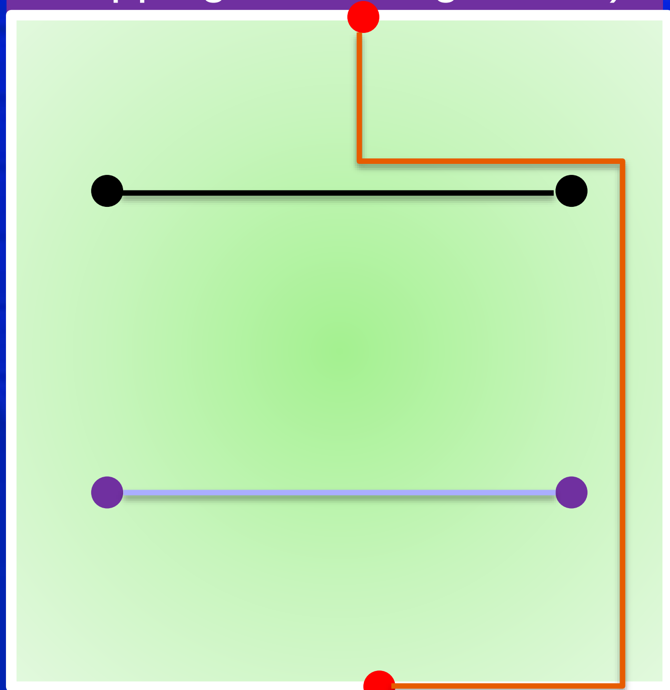
Routing in Monosat



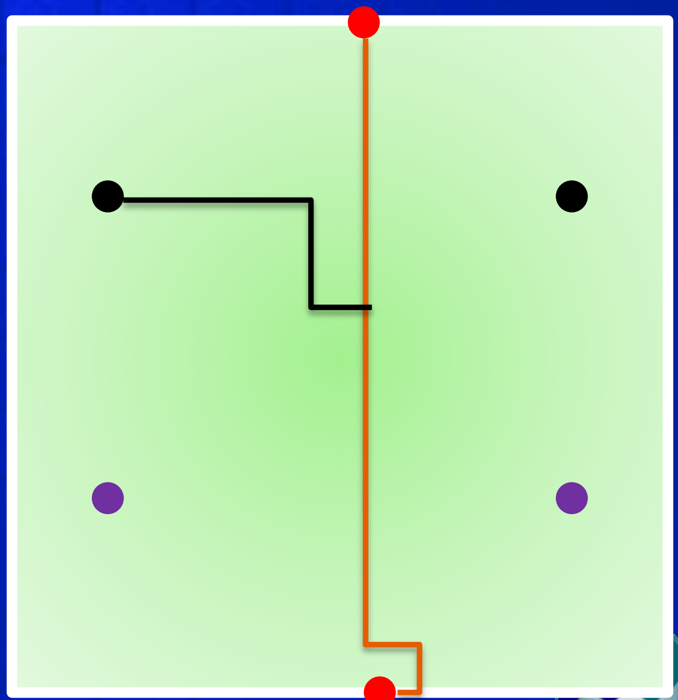
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



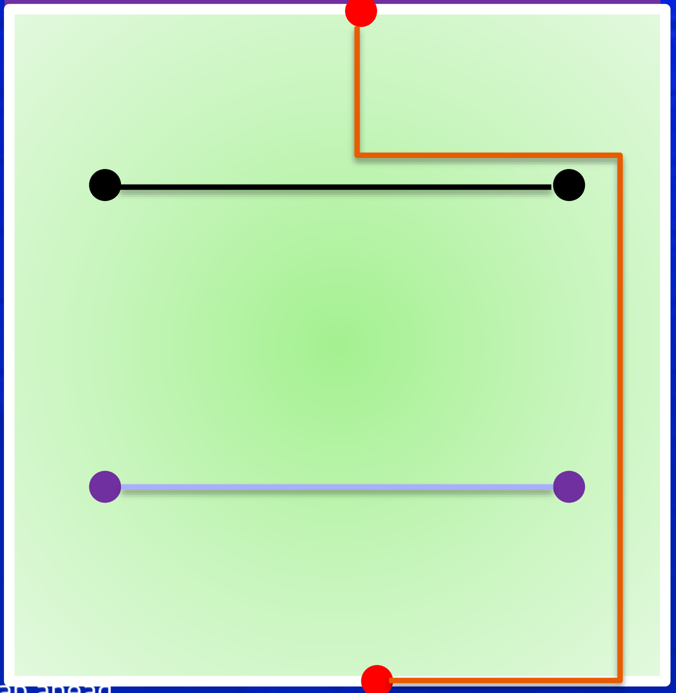
Routing in Monosat



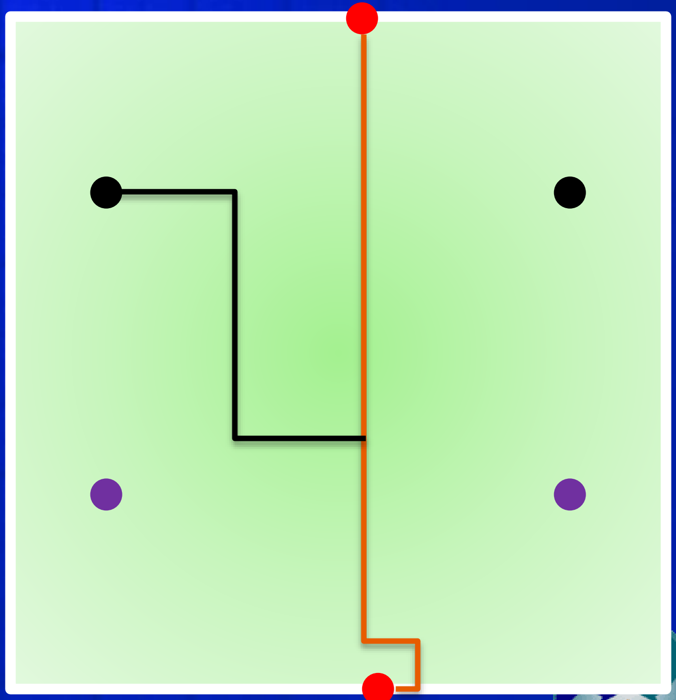
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



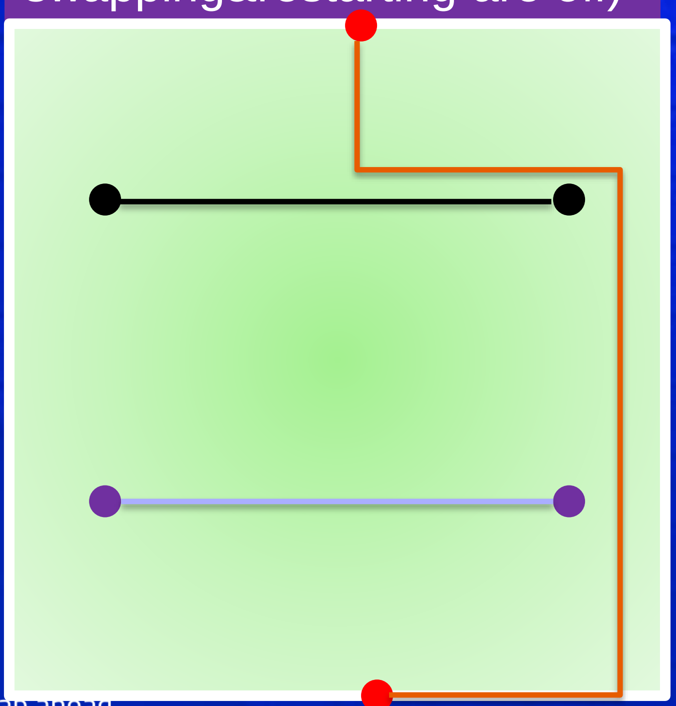
Routing in Monosat



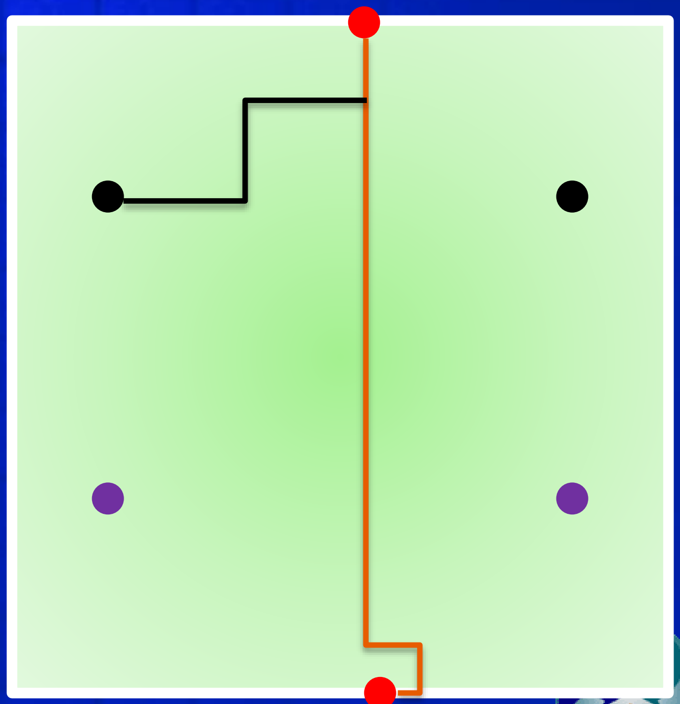
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)



Routing in Monosat



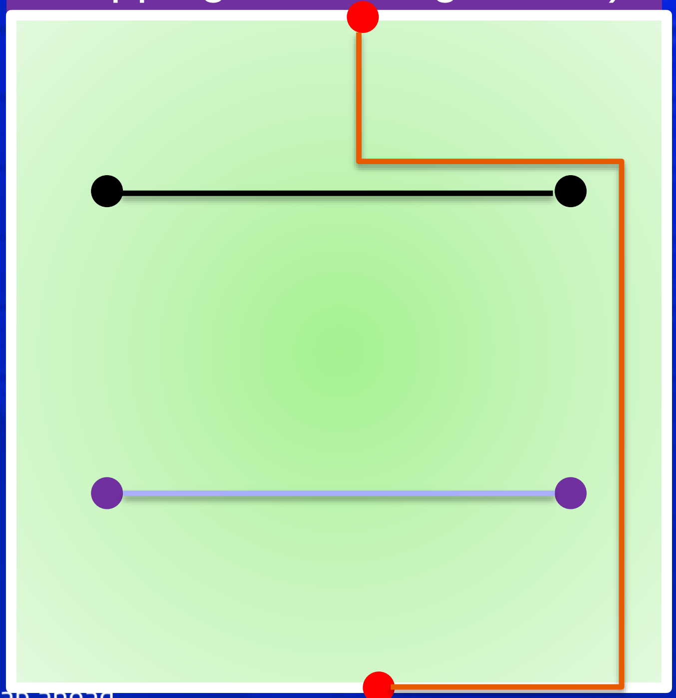
Leap ahead

Design and Technology Solutions

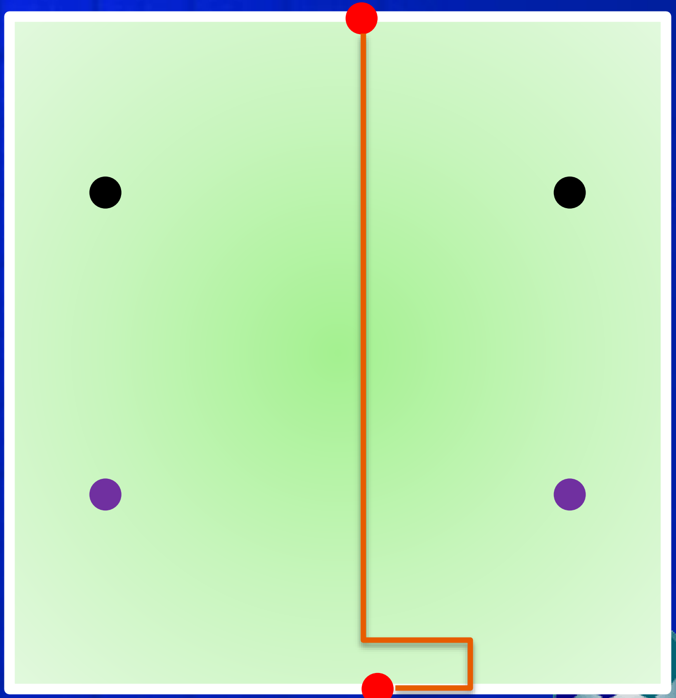
Monosat vs. DRouter for Routing under Constraints

- Monosat's algorithms are not routing-aware
  - No net re-ordering
  - Graph conflict analysis for routing is inefficient

Routing in DRouter (net swapping&restarting are off)

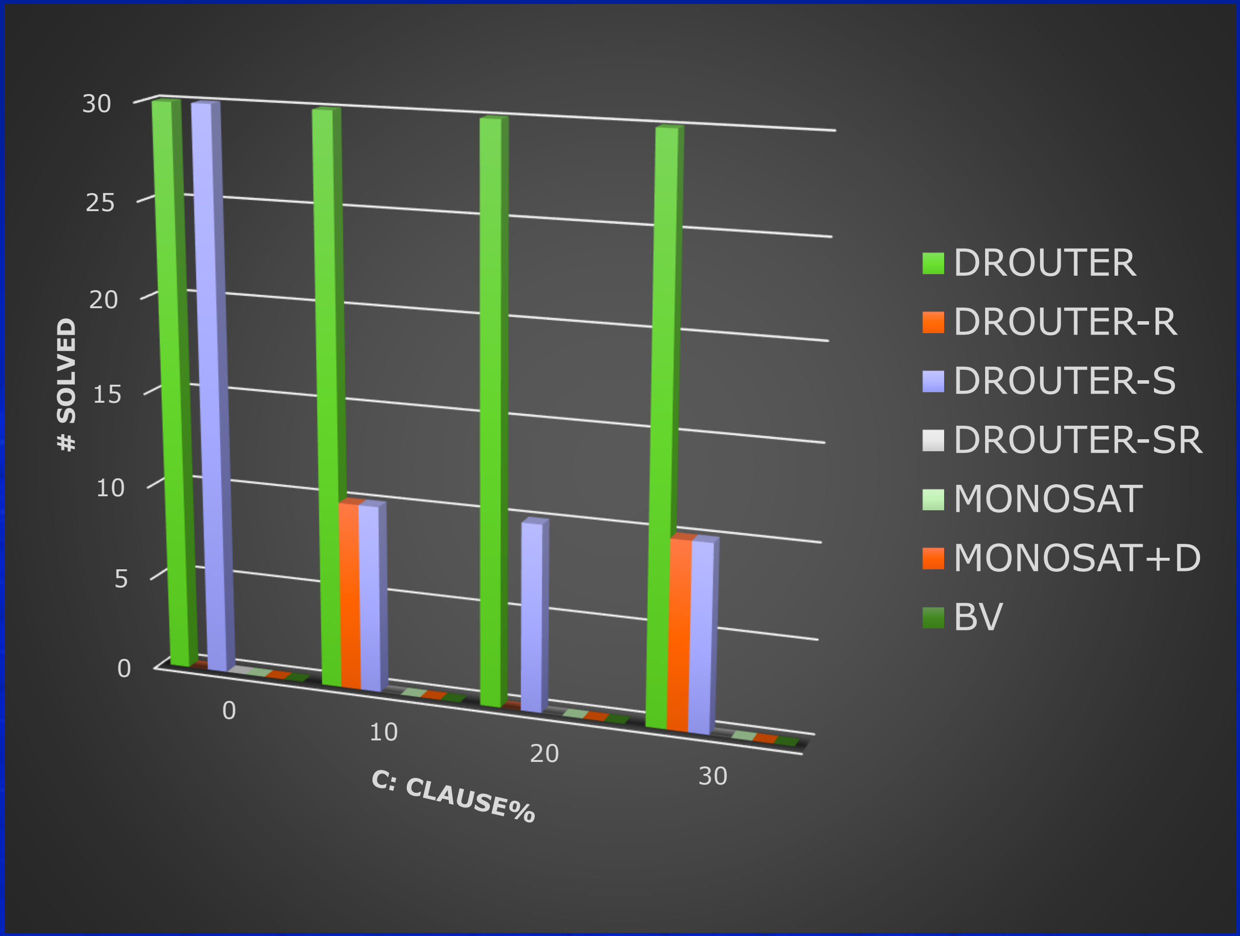


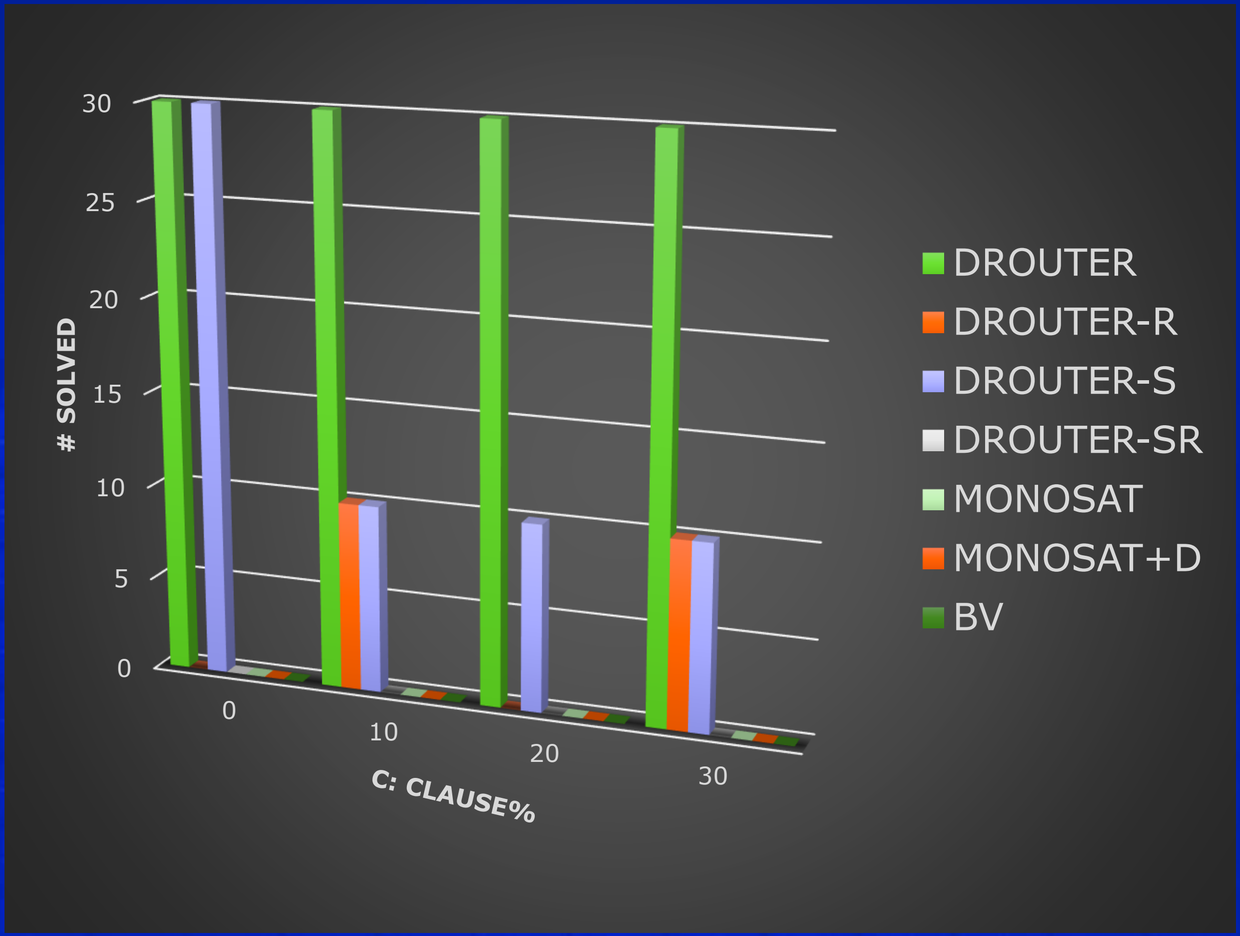
Routing in Monosat



Experimental Results on Crafted Instances

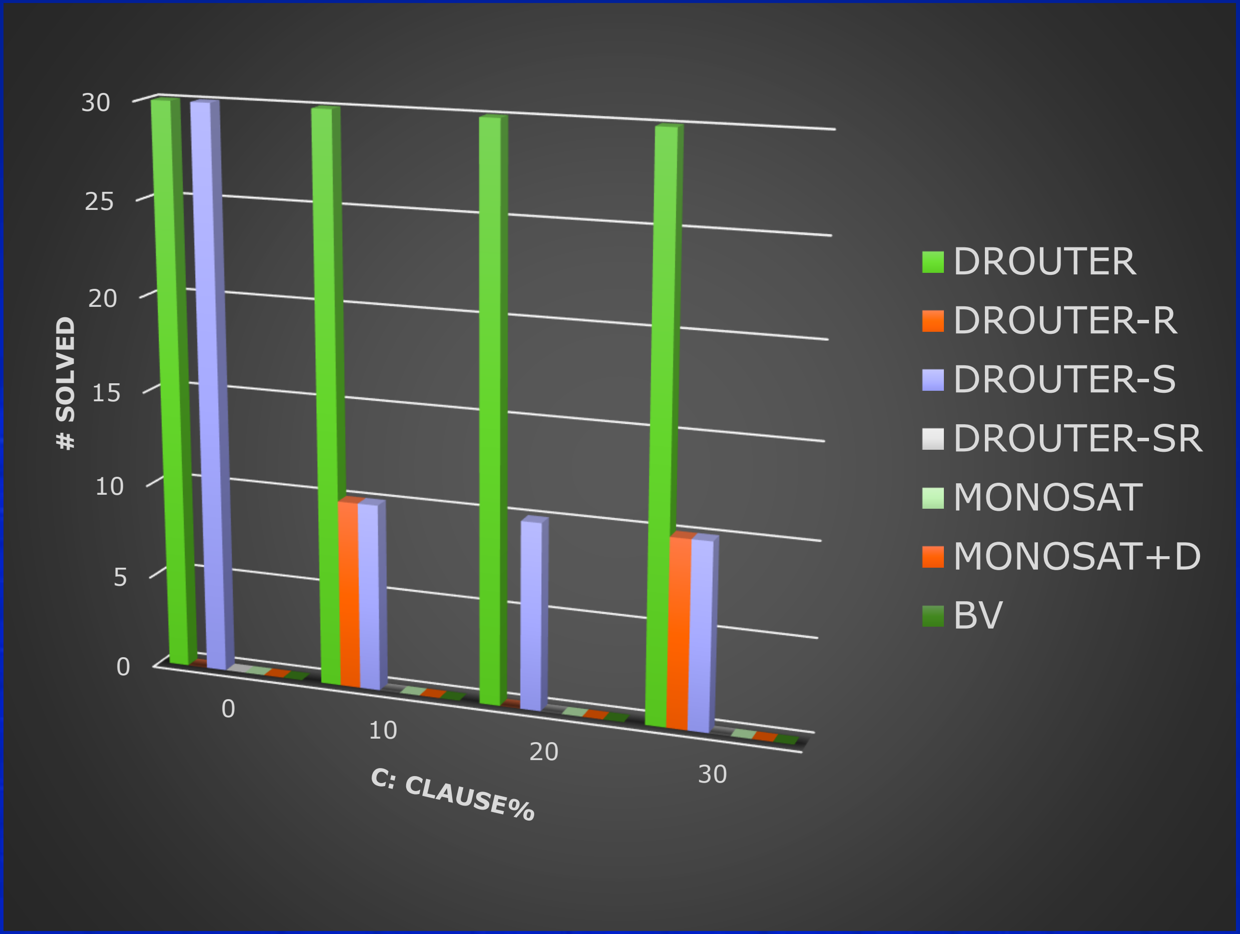
- Solvers:
  - Drouter (default)
  - Drouter – R: no net restarting
  - Drouter – S: no net swapping
  - Drouter – SR: no net swapping, no net restarting
  - Monosat (default)
  - Monosat + D: shortest-path decision strategy is on
  - BV: reduction to BV
- Instances:
  - 120 solid grid graphs of size $M \times 20$
    - $M \in \{3,5,7\}$
  - 20 random 2-terminal nets
  - Generate $C * |V|$ random binary clauses $\neg v \vee \neg u$
    - $v,u \in V$
    - $C \in \{0,0.1,0.2,0.3\}$


30
25
20
15
10
5
0
SOLVED
0
10
20
30
C: CLAUSE%
DROUTER
DROUTER-R
DROUTER-S
DROUTER-SR
MONOSAT
MONOSAT+D
BV



- Full-fledged DRouter only can solves all the instances
  - Both net restarting and net swapping are essential!

Design and Technology Solutions



- Full-fledged DRouter only can solves all the instances
    - Both net restarting and net swapping are essential!
- Monosat and BV can’t solve a single instance

Leap ahead™
Design and Technology Solutions

DRouter on Industrial Instances

- Run DRouter on difficult clips from Intel designs
  - Couldn't be routed cleanly by 2 industrial routers

DRouter on Industrial Instances

- Run DRouter on difficult clips from Intel designs
  - Couldn't be routed cleanly by 2 industrial routers

Area in μm^2	Nets	Vertices	Constraints	Time in sec.	Memory in Gb.
24	110	42,456	484,008	25	0.7
24	230	42,456	484,008	391	1.0
32	352	63,740	667,764	705	2.2
129	788	127,480	2,669,056	14,733	6.5
129	891	127,480	2,669,056	92,950	6.5

Conclusion

- DRouter: design-rule-aware router
  - SAT solver surgery:
    - Decision heuristic → A*-based router
    - Conflict analysis enhanced with graph reasoning
    - Restarts → net swapping & net restarting
- Solves instances which can't be solved by existing tools
  - Including clips from real Intel designs