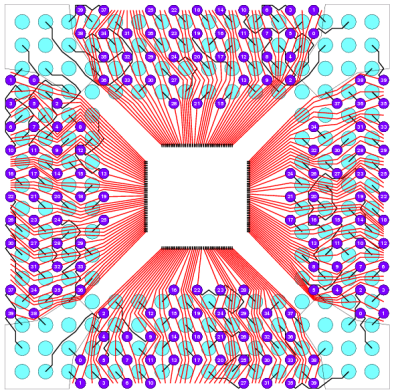
VLSI Layout Design
Routing (1) Overview

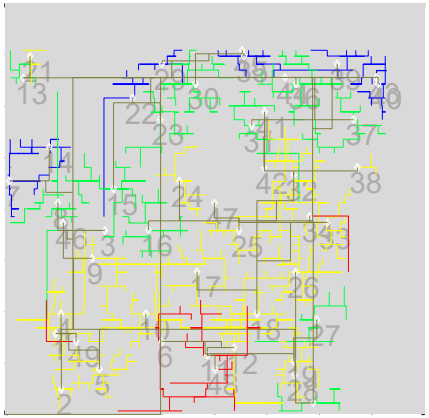
Atsushi Takahashi
Department of Information and Communications Engineering
School of Engineering
Tokyo Institute of Technology
atsushi@ict.e.titech.ac.jp

ICT.I419 VLSI Layout Design

Routing

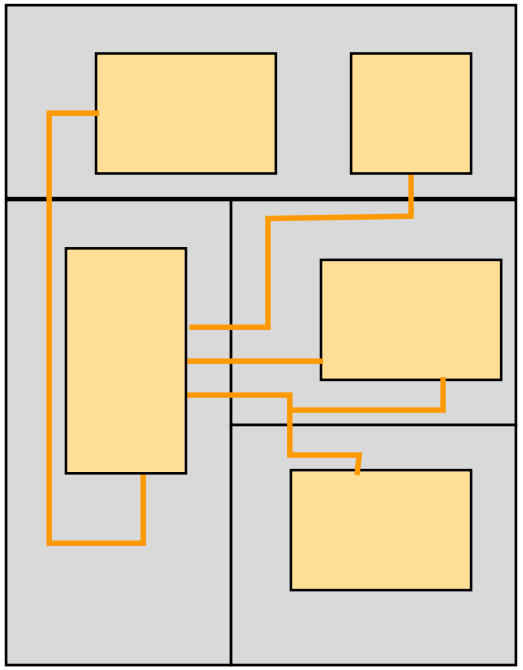
- Connect pins under the design rule
  - Many nets (many connection requests)
  - 100% completion ratio
    - 100% without manual correction
    - Near 100% + manual correction
  - Various design rules
    - # of layers
    - Obstacles
  - Various properties of instances
    - Pin distribution
  - Various objectives
    - Total length, delay, power, shape





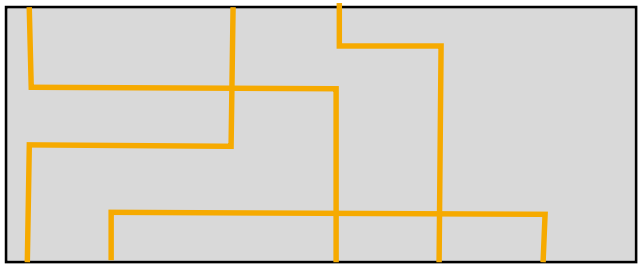
Hierarchical Design

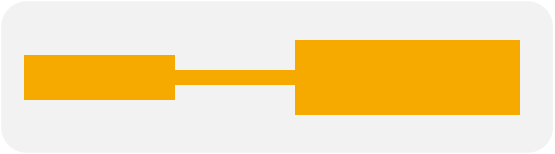
- Global routing
  - Design rule insensitive
  - Routing area is divided into subareas
  - Balance the congestion of subareas
    - Greedy approach
      - Shortest Path (Two terminal)
      - Steiner Tree (Three or more terminal)
    - Rip-up and Reroute
      - needed usually for multiple nets
      - find a compromise among nets
      - update cost-map by the history of rip-up

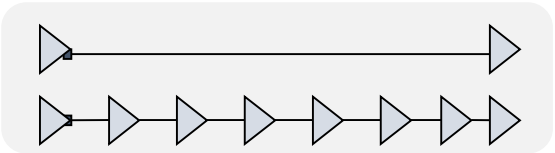


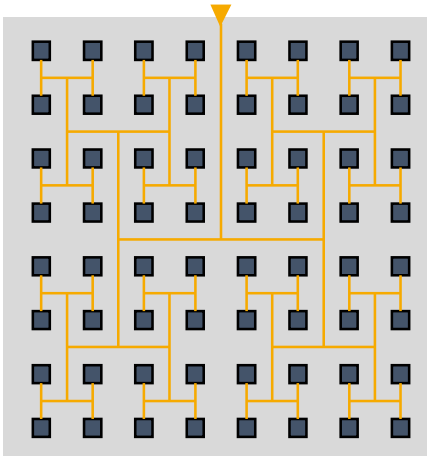
Hierarchical Design

- Detailed routing
  - Design rule sensitive
  - Multiple Nets
    - Channel routing (Two or Three layer)
    - Switch box routing (Two or Three layer)
    - Area routing (Three or more layer)
  - Single Nets
    - Wire sizing
    - Buffer Insertion
    - Signal integrity
    - Via planning
    - Clock routing



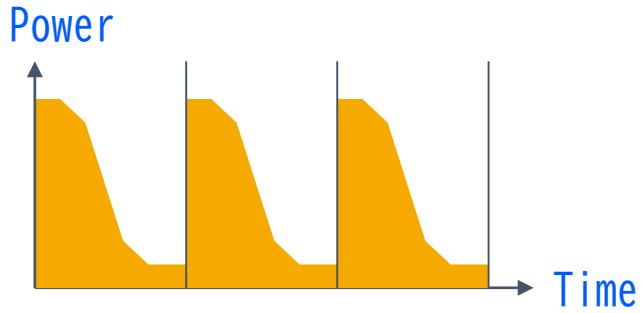


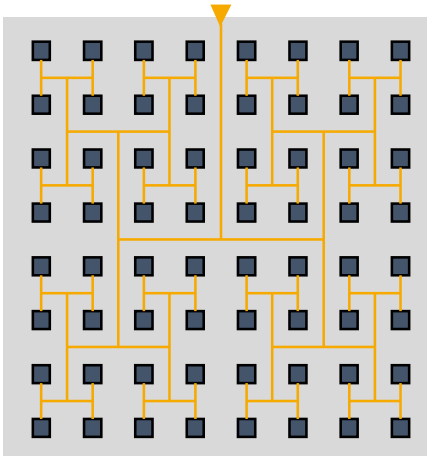




Objectives

- Objectives are changing depending on technological environments
  - 100% routing
  - Area (total length) minimization
  - Delay minimization
  - Skew minimization
  - Power minimization
  - Noise minimization
  - Delay control





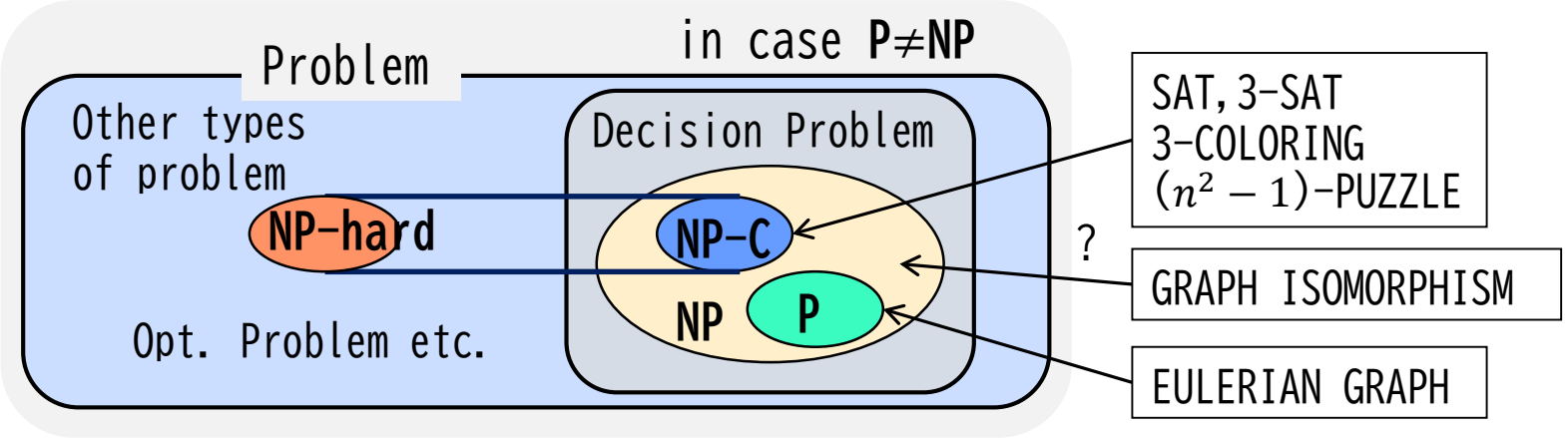
Theoretical Aspect

- Background of Algorithm Design
- Computational Complexity
  - **P** and **NP**
  - **NP**-complete
  - **NP**-hard
  - Nondeterministic Polynomial Time Algorithm
  - (Deterministic) Polynomial Time Algorithm
  - Polynomial Time Reduction

[Garey and Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness", Freeman and Co., 1979]

NP-hardness

- A problem is NP-hard if the decision problem associated with the problem is NP-complete
- Optimization problem is
  - neither in NP nor in NP-C
  - not said to be NP-complete
  - said to be NP-hard if a related decision problem is NP-complete
- If P≠NP, then
  - No polynomial time algorithm for NP-hard problem
- If a problem is NP-hard, then
  - Approximation algorithm or Heuristic algorithm are pursued

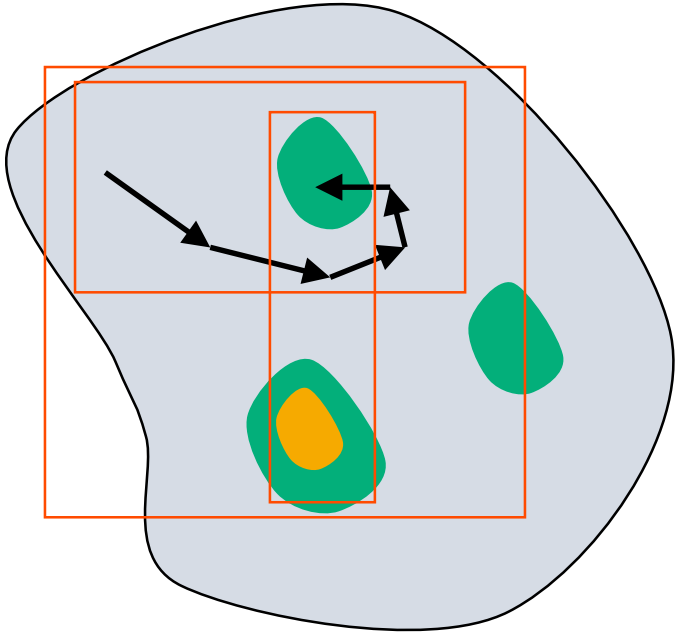


First Step of Algorithm Design

- Check whether problem is easy or not?
  - ✓ Assuming **P** ≠ **NP**
    - Difficult = **NP**-hard, **NP**-complete
      - Design heuristic
    - Easy = **P** (or decision version is in **P**)
      - Design exact polynomial time algorithm
      - Reduce time and space complexity
- Most of practical problems are difficult
  - **NP**-hardness seems trivial ....
    but proof of **NP**-hardness is not easy
  - So, proof is often skipped, recently
- In the following
  - **P** = problem solvable in polynomial time

Digital Integrated Circuits Synthesis

- Exploration of Huge Design Space
- Increase of computation power enable us to use computation power rich algorithms
  - Iterative improvement
  - Stochastic search
  - Analytical method
- Solution space design
  - Abandon useless area
  - Focus on promising area
  - Efficiency

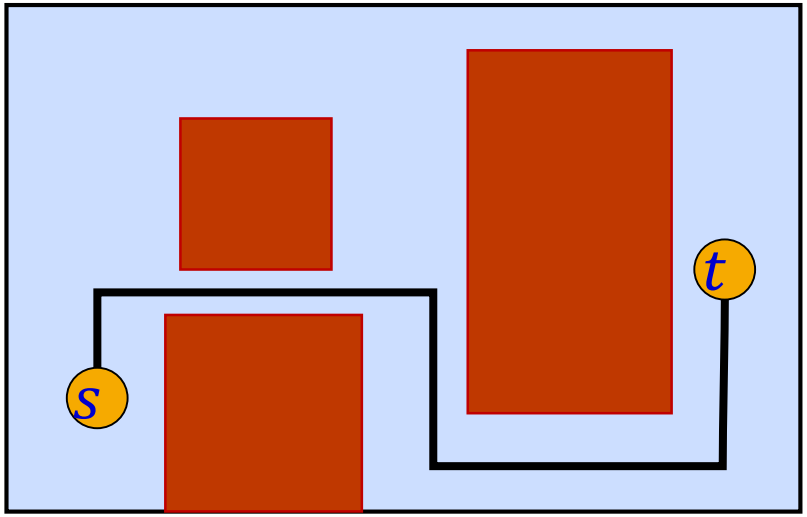


Automated vs. Manual Routing

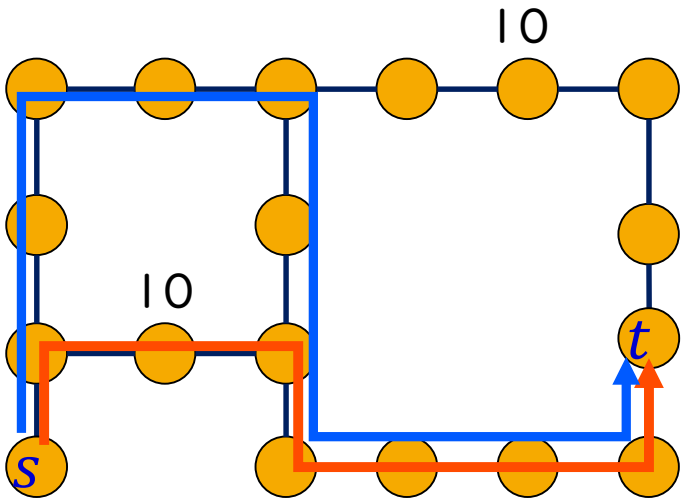
- Good routing tools have been developed so far
- For large chips
  - Huge number of nets and enough routing resources
  - Looser constraint
  - Too many nets to design manually
  - Lower quality is affordable
  - Routing tools are essential in recent design
- For packages, RDL, and PCBs
  - Medium number of nets and limited routing resources
  - Tighter constraint
  - Time consuming, but designer can handle
  - Higher quality is essential
  - Routing tools are still not popular in high-end designs

Shortest Path Problem

- Basic Routing Problem
- Find a shortest path between two nodes
  - Decision version of shortest path problem is in **P**
    - Assumption : weight is nonnegative



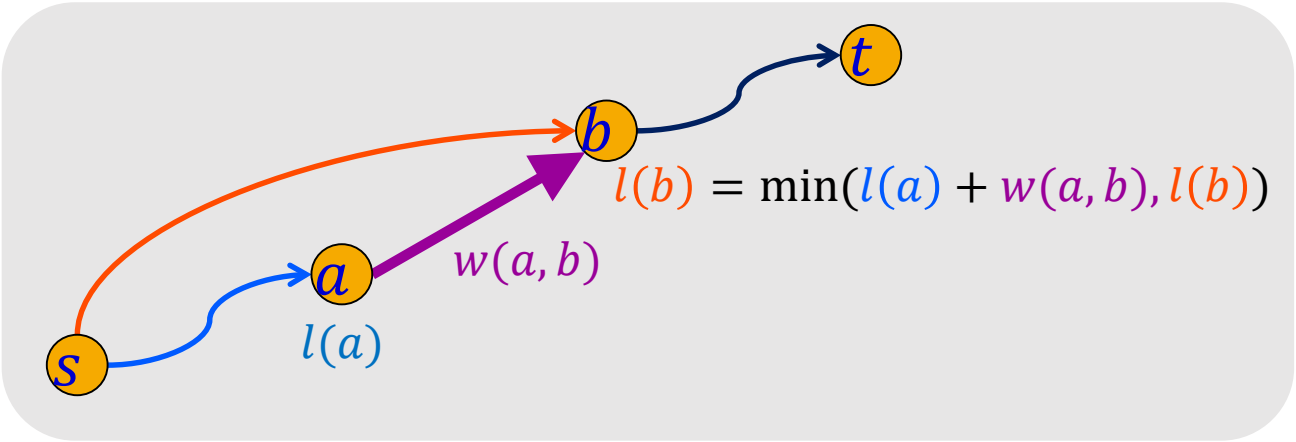
Routing Plane

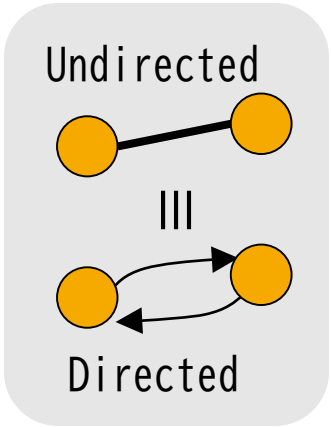


Routing Graph

Shortest Path Algorithm

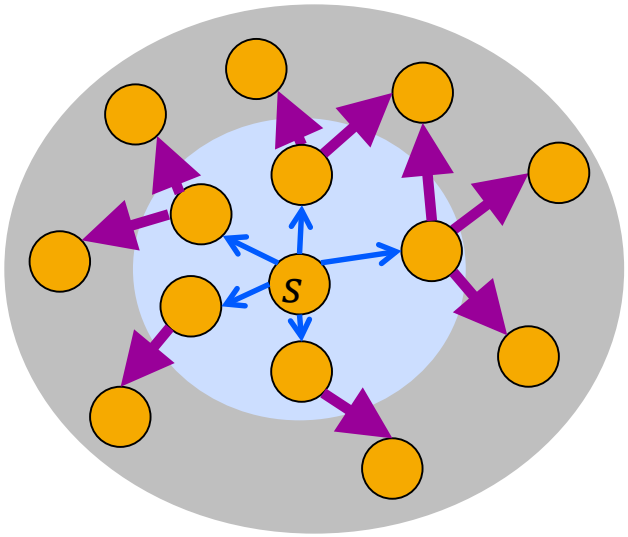
- Find minimum $l(v)$
  - $l(v)$: the length of a best route from source s to v found so far
  - $l(a) + w(a,b) \geq l(b)$ for any edge (a,b)
- Common procedure of maze like algorithm
  - Explore adjacent vertex
    - check whether $l(a) + w(a,b) < l(b)$
  - Update best route found so far if a better route is found
    - If $l(a) + w(a,b) < l(b)$, then update $l(b) := l(a) + w(a,b)$



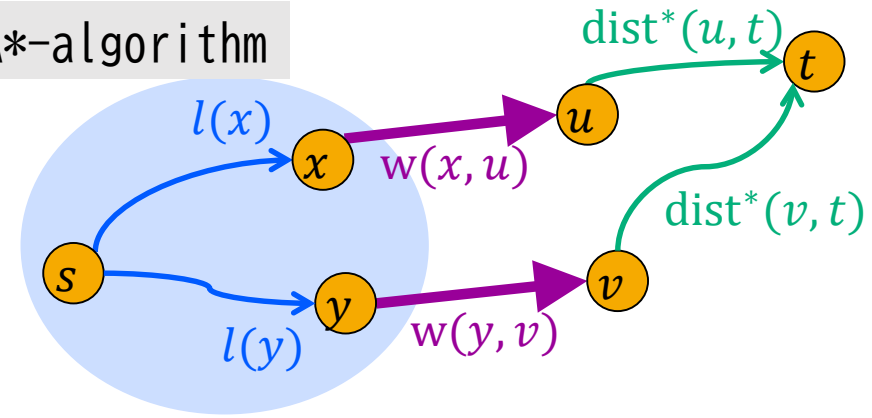


Variety of Exploration

- Priority of exploration
  - Queue (first-update first-explore) [#edge from source]
    - Breadth-First Search, Wave Front (for unit weight)
    - Bellman-Ford (for non-negative weight cycle)
      - Explore all adjacent vertices whenever the label is updated
  - Minimum length from source $[l(x) + w(x,u)]$
    - Dijkstra (for non-negative weight)
  - Minimum expected length from source to sink $[l(x) + w(x,u) + \mathrm{dist}^*(u,t)]$
    - A*-algorithm (for non-negative weight and consistent prediction dist^*)



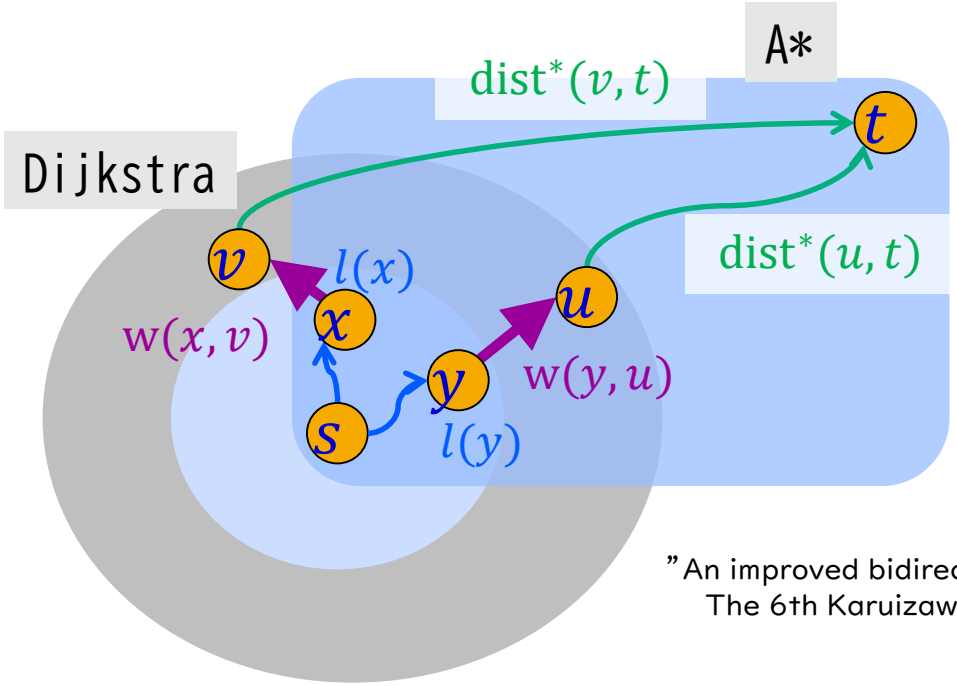
A*-algorithm

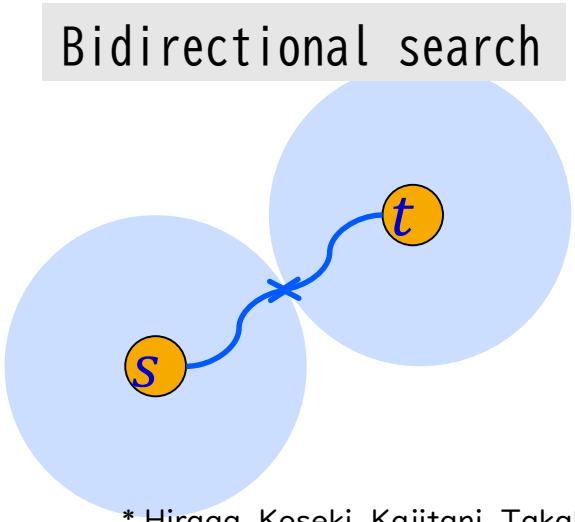


$\mathrm{dist}^*(.,t)$: expected length to sink t

Speed-up Techniques

- Reduce redundant exploration as much as possible
- Expand the region that distance from the source is known
  - Dijkstra: $l(x) + w(x,v)$ is minimum
  - A* : $l(x) + w(x,v) + \mathrm{dist}^*(b,t)$ is minimum
    - Utilize geometrical information
- Bidirectional search (from source and from sink)
  - One big radius circle versus Two small radius circles

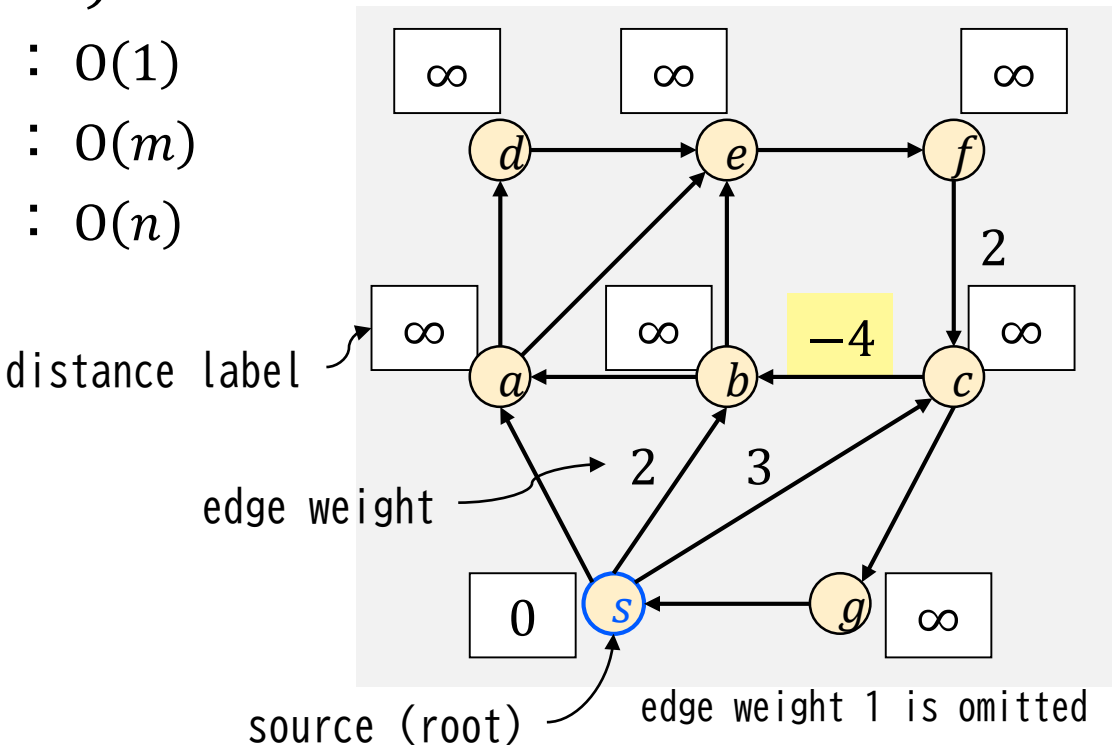




* Hiraga, Koseki, Kajitani, Takahashi.
"An improved bidirectional search algorithm for the 2 terminal shortest path"
The 6th Karuizawa Workshop on Circuits and Systems, 1993 (in Japanese)

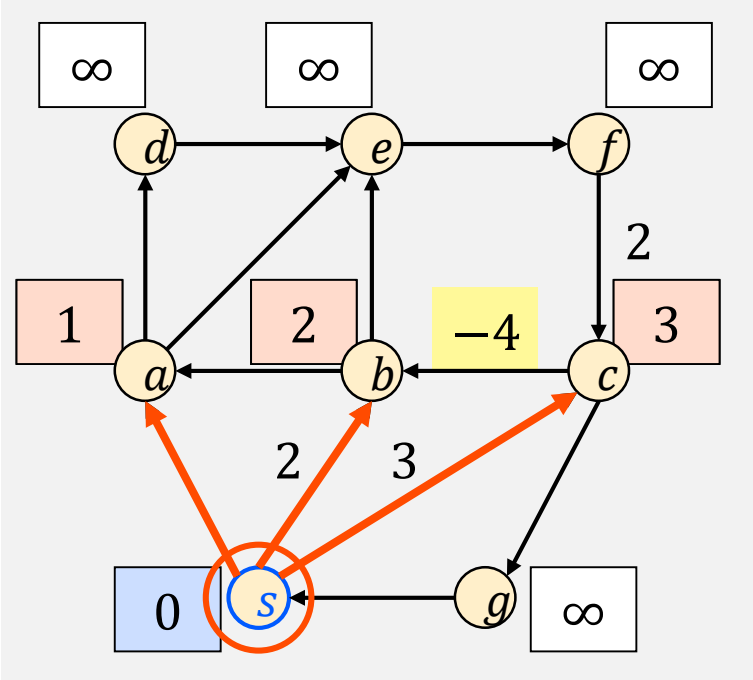
Shortest Path with Negative Weight

- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$



Shortest Path with Negative Weight

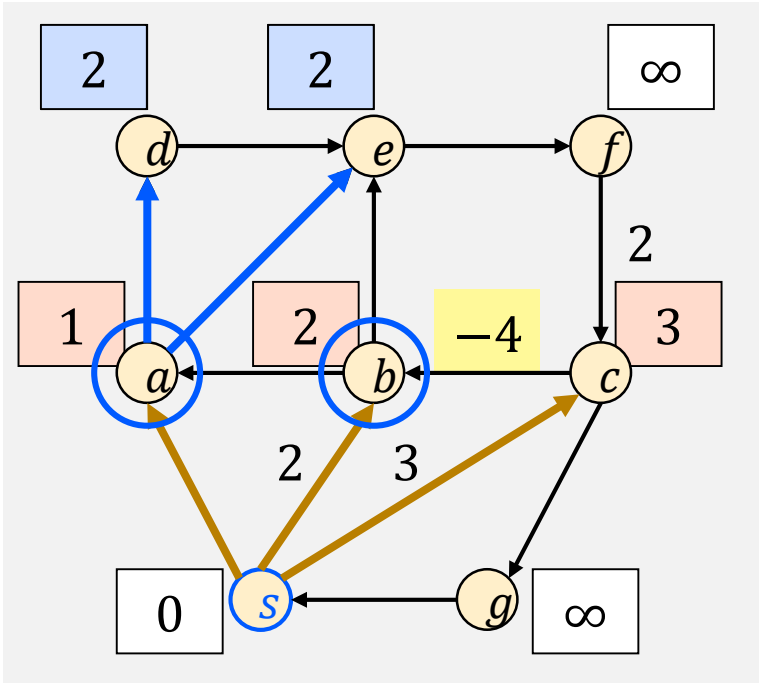
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round



edge weight 1 is omitted

Shortest Path with Negative Weight

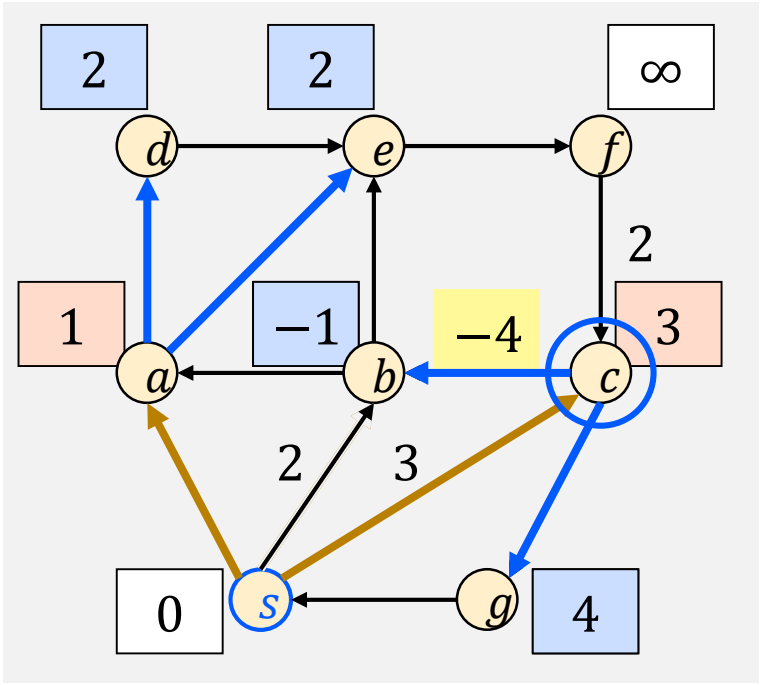
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round



edge weight 1 is omitted

Shortest Path with Negative Weight

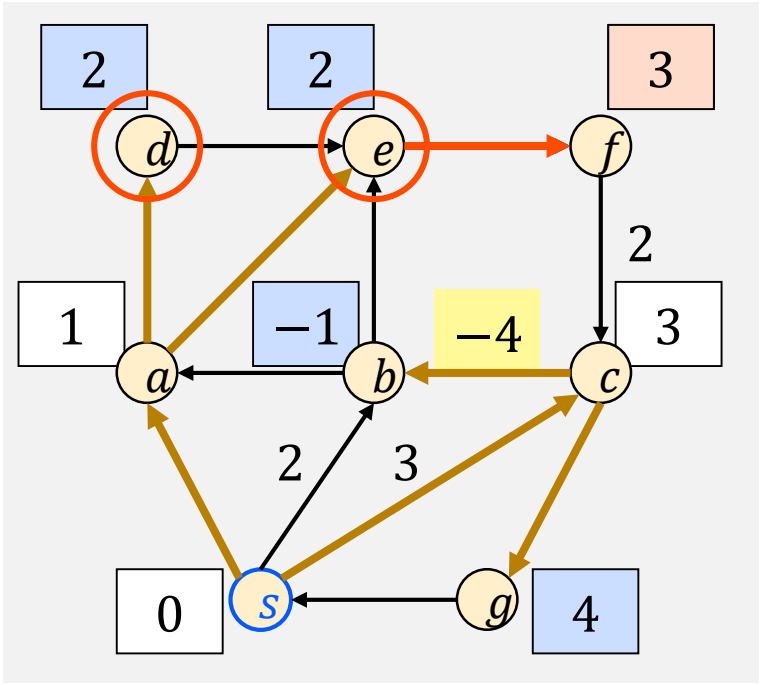
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round



edge weight 1 is omitted

Shortest Path with Negative Weight

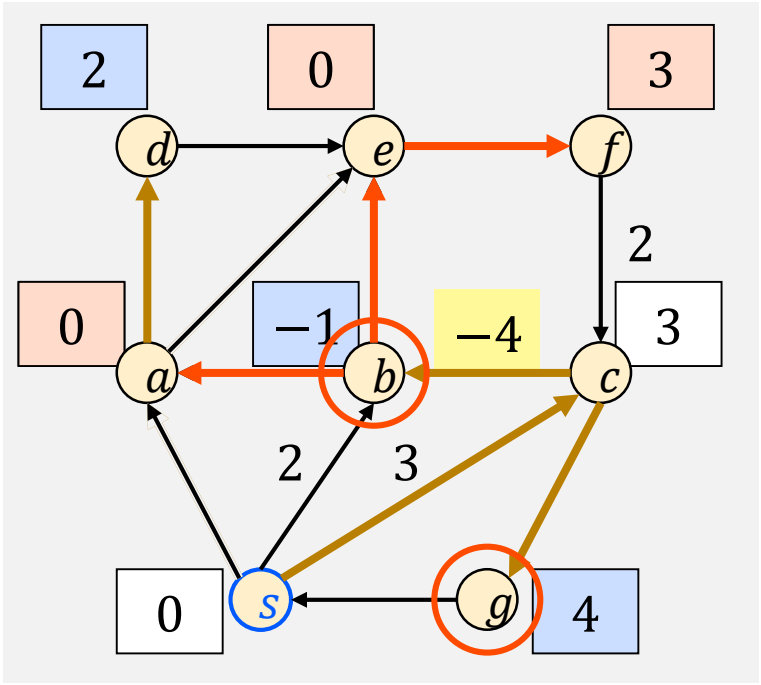
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round
  - 3rd Round



edge weight 1 is omitted

Shortest Path with Negative Weight

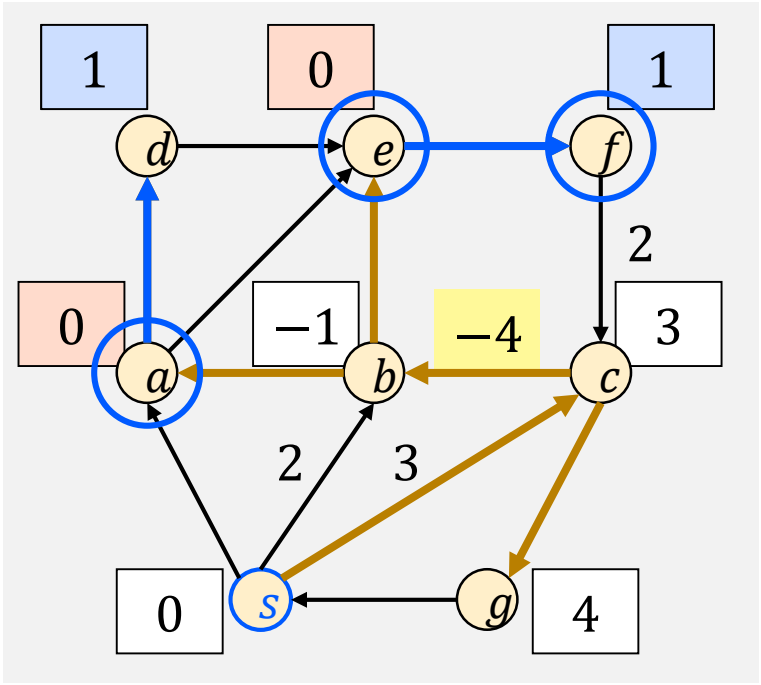
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round
  - 3rd Round



edge weight 1 is omitted

Shortest Path with Negative Weight

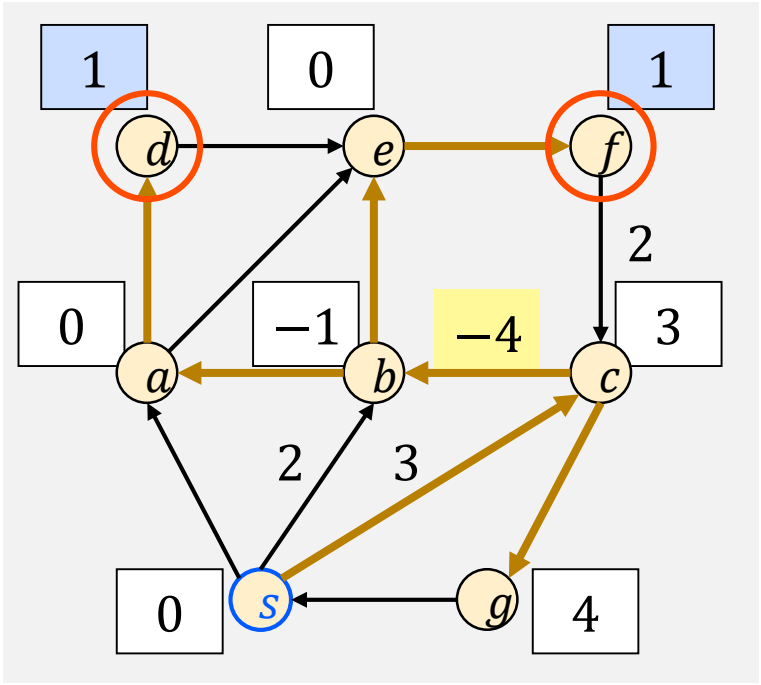
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round
  - 3rd Round
  - 4th Round



edge weight 1 is omitted

Shortest Path with Negative Weight

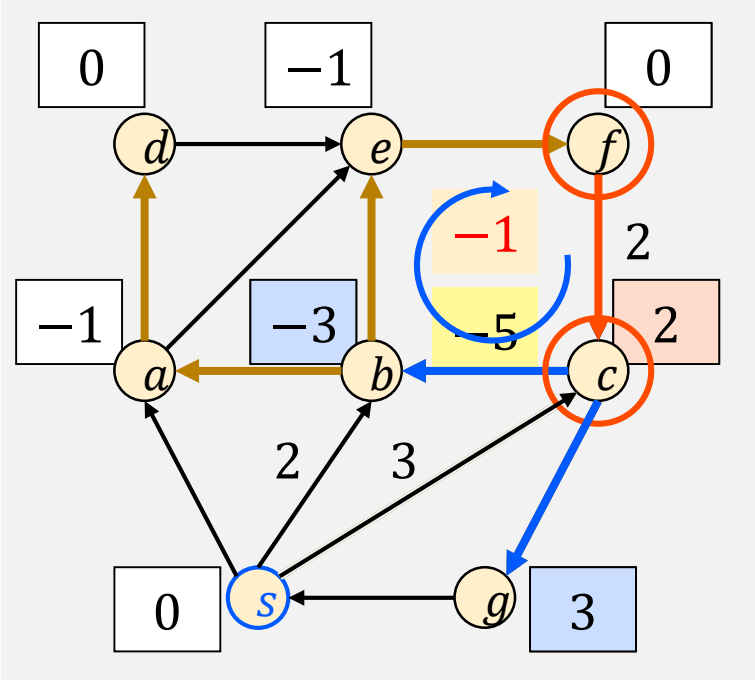
- Bellman-Ford Algorithm
  - Update adjacent labels in the next round if updated
  - Complexity : $O(nm)$
    - Labeling : $O(1)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$
  - 1st Round
  - 2nd Round
  - 3rd Round
  - 4th Round
  - 5th Round
    - ✓ No label update, and finish

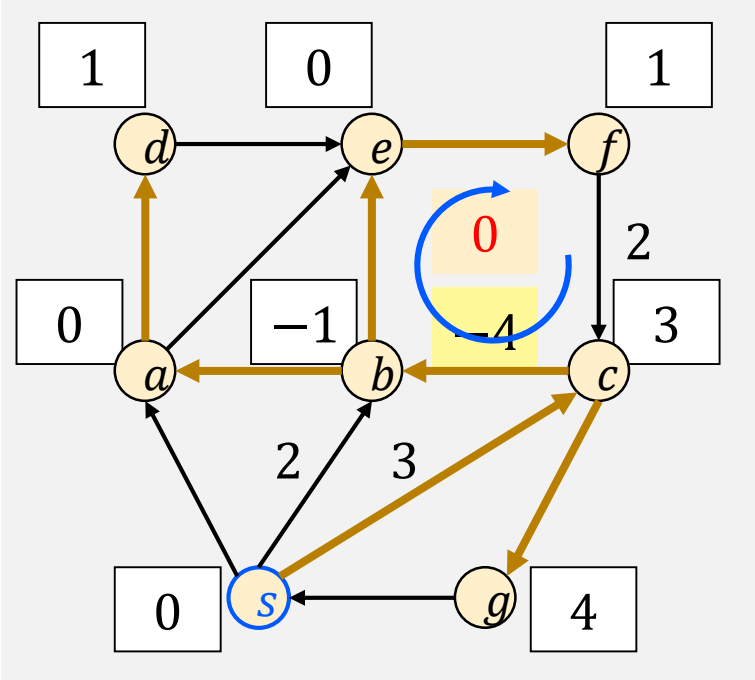


edge weight 1 is omitted

Negative Weight Cycle

- Bellman-Ford Algorithm
  - ✓ Converge when no negative cycle exists
  - ✓ Updates never converge when negative cycle exists





edge weight 1 is omitted

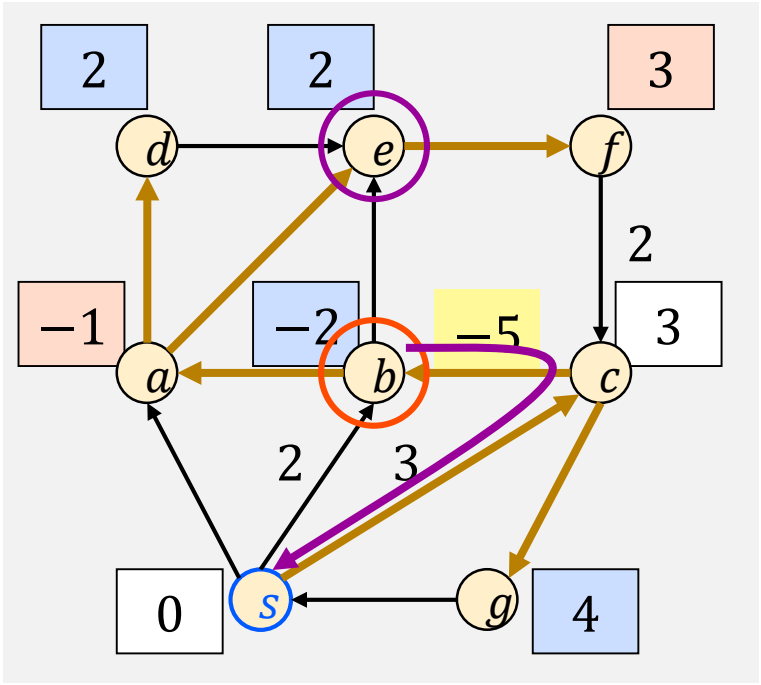
- Can stop at #node-round as negative weight cycle is detected

Negative Cycle Detection Techniques

- Time Out
  - Stop after a certain number of updates are performed
- Distance Lower Bound
  - Stop if the distance label becomes below the predefined lower bound
- Walk to the Root before labeling u from v
  - Follow parent pointer from v in search tree
    - If it reaches u, update forms a cycle in search tree, that is, negative cycle found
    - If it reaches root without passing u, update forms no cycle in search tree
- Walk to the Root after labeling adjacent nodes v
  - Follow parent pointer from v in search tree.
    - If it reaches v, negative cycle is found in search tree
    - Otherwise, it reaches root, no cycle is formed so far in search tree

Negative Cycle Detection (Pre-Check)

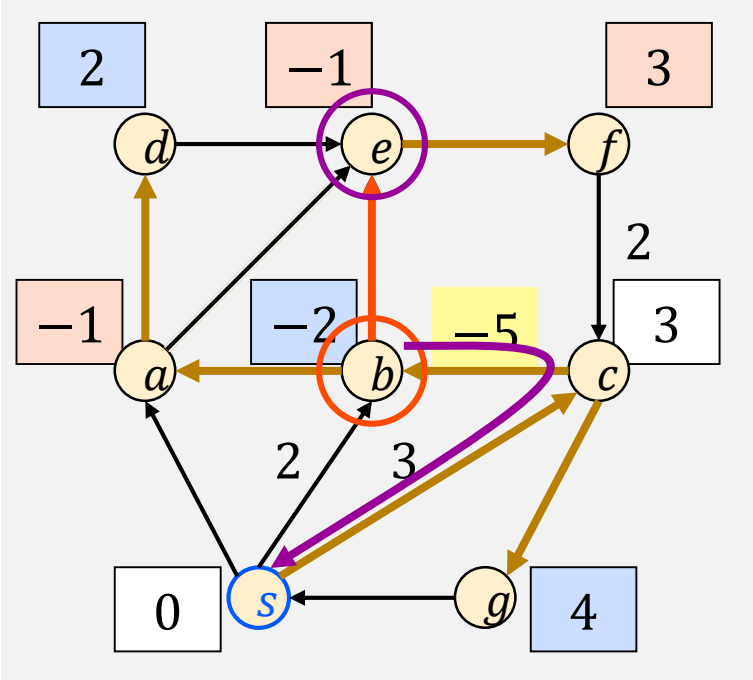
- Bellman-Ford + Walk to the Root before
  - 3rd Round
    - Before update label e by b
      - Walk to the root from b
      - e is not on the path to s



edge weight 1 is omitted

Negative Cycle Detection (Pre-Check)

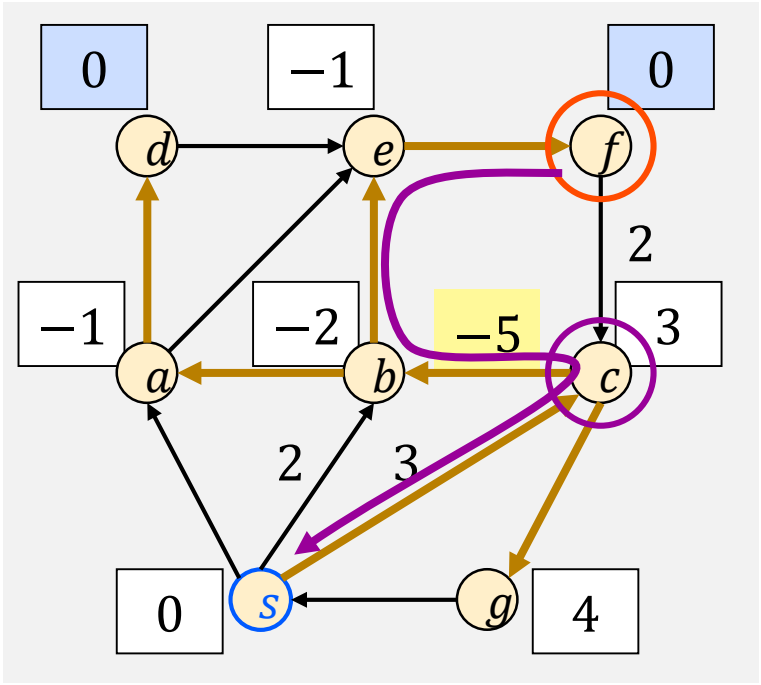
- Bellman-Ford + Walk to the Root before
  - 3rd Round
    - Before update label e by b
      - Walk to the root from b
      - e is not on the path to s
    - No cycle will be formed



edge weight 1 is omitted

Negative Cycle Detection (Pre-Check)

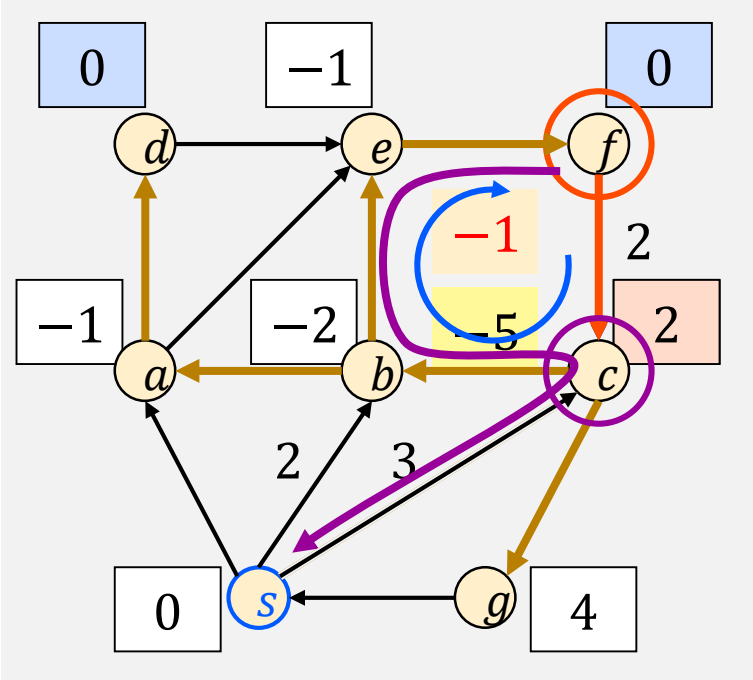
- Bellman-Ford + Walk to the Root before
  - 5th Round
    - Before update label c by f
      - Walk to the root from f
      - c is on the path to s



edge weight 1 is omitted

Negative Cycle Detection (Pre-Check)

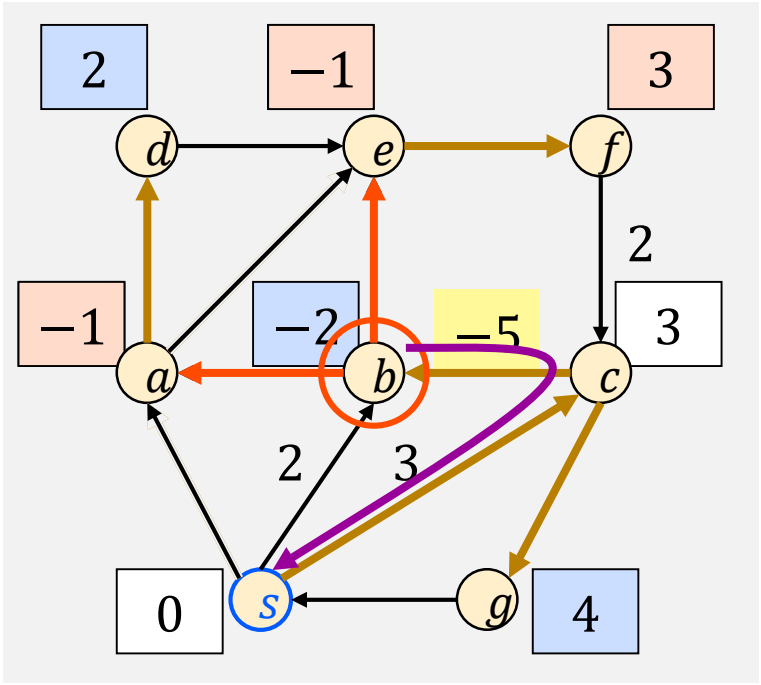
- Bellman-Ford + Walk to the Root before
  - ➢ 5th Round
    - Before update label c by f
      - Walk to the root from f
      - c is on the path to s
  - ✓ Cycle will be formed if the destination is on the walk to the root
  - Complexity : $O(n^2m)$
    - Labeling : $O(n)$
    - Labeling/ #round : $O(m)$
    - #round : $O(n)$



edge weight 1 is omitted

Negative Cycle Detection (Post-Check)

- Bellman-Ford + Walk to the Root after
  - 3rd Round
    - Labeling the neighbors of b
      - Walk to the root from b
      - It reaches the root s
      - Confirm that no cycle is formed

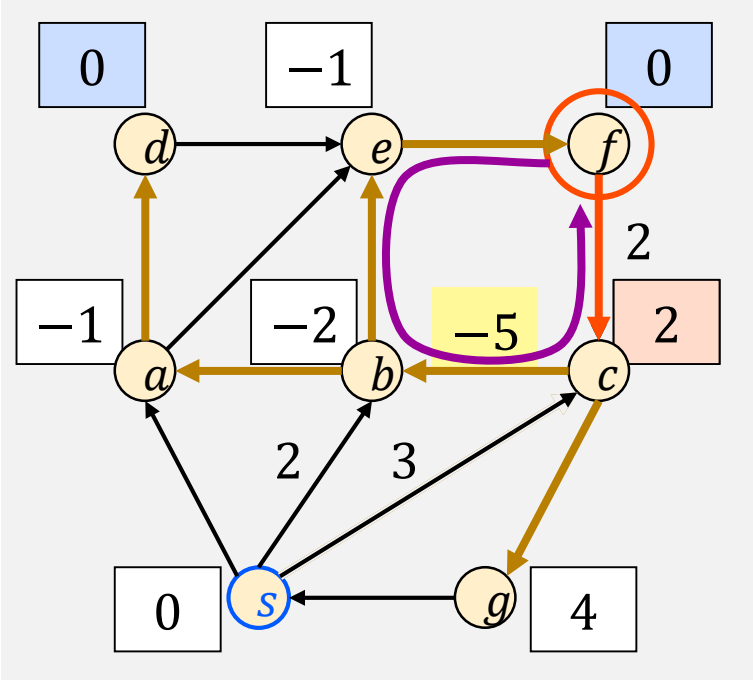


edge weight 1 is omitted

Negative Cycle Detection (Post-Check)

- Bellman-Ford + Walk to the Root after
  - 5th Round
    - Labeling the neighbors of f
      - Walk to the root from f
      - It cannot reach the root s, but reaches f
      - Confirm that cycle is formed
  - Complexity: $O(km + k^2 n)$
    - Labeling : $O(1)$
    - Labeling/#round : $O(m)$
    - Walk to the root: $O(k)$
    - Walk/#round : $O(n)$
    - #round : $O(k)$

k : diameter of graph



edge weight 1 is omitted

Experiments on Negative Cycle Detection

	t-cp	Ans.	BF[s]	BFn[s]
b1	4000	No	26.66	0.0023
	8000	No	12.99	0.0170
	8300	No	10.22	0.0149
	8322	No	0.8762	0.0148
	8323	Yes	0.1255	0.1259
	8400	Yes	0.1268	0.1271
	11569	Yes	0.1248	0.1250
	16000	Yes	0.1247	0.1250

#reg:1654 #path:11697

BF : Bellman-Ford
BFn: Bellman-Ford with Post-Check
No: Negative cycle exists (infeasible)
Yes: No negative cycle (feasible)

	t-cp	Ans.	BF[s]	BFn[s]
b3	9000	No	795.25	0.019
	9552	No	32.78	1.912
	9553	Yes	3.029	3.033
	10000	Yes	2.876	2.882

#reg:7973 #path:104136

	t-cp	Ans.	BF[s]	BFn[s]
b5	8763	No	11669.6	0.253
	9664	No	1096.11	0.246
	9665	Yes	7.256	7.313
	10692	Yes	7.130	7.151

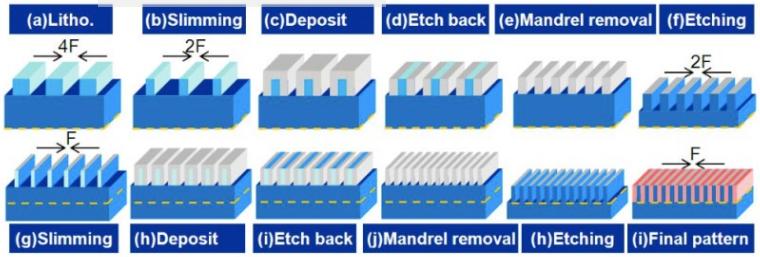
#reg:12460 #path:947082

* Atsushi Takahashi. “Practical Fast Clock-Schedule Design Algorithms.” IEICE Trans. Fundamentals, Vol.E89-A, No.4, pp.1005-1011, 2006

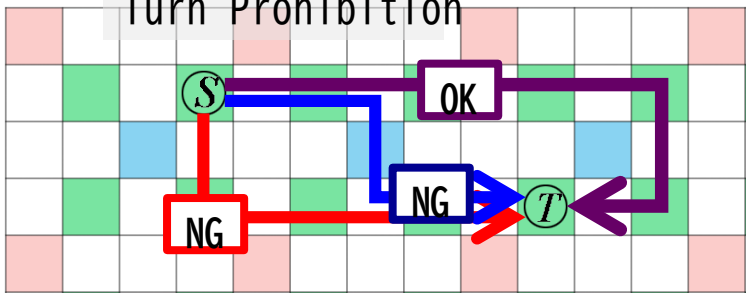
Grid-Based Routing for SAQP

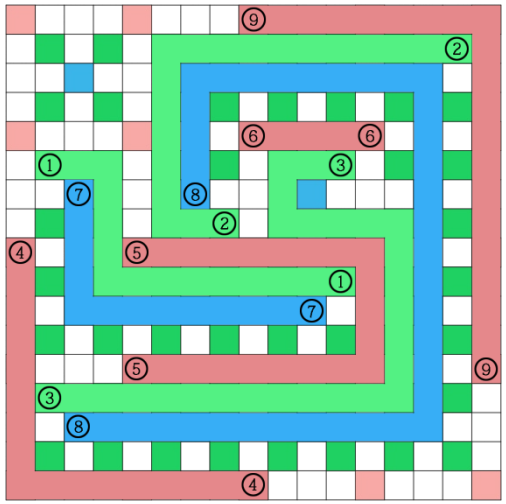
- Sometime, several turns are prohibited in Path
  - Vehicle Navigation
  - Tertiary Pattern of Self-Aligned Quadruple Patterning (SAQP) in Kodama Grid [ASP-DAC2013, *1]
- Existence of Path under Turn Prohibition: **NP**-complete [*2]

SAQP Process

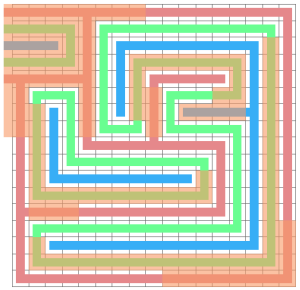


Turn Prohibition





Kodama Grid



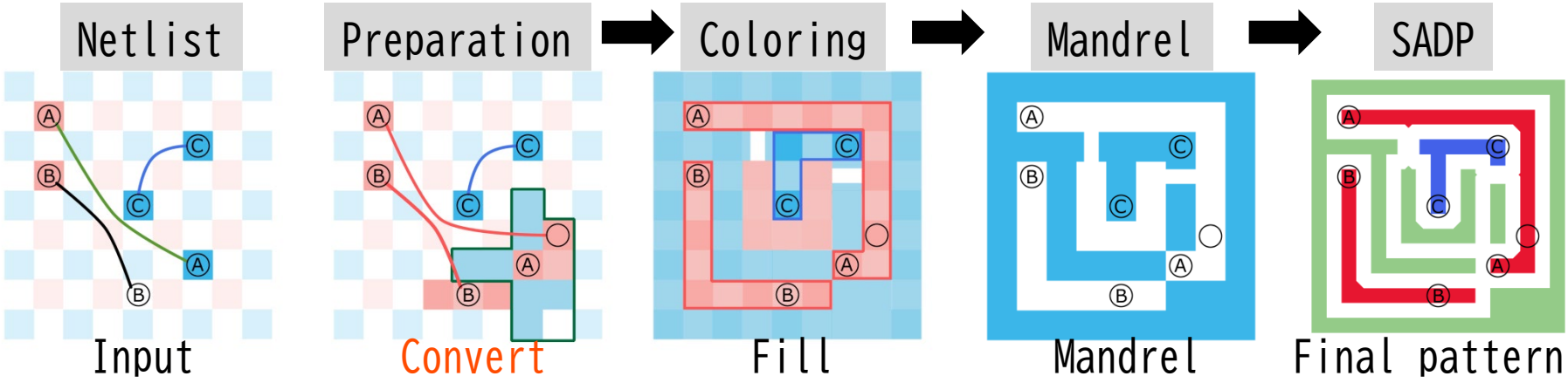
SAQP Pattern

*1 Kodama et.al. "Self-Aligned Double and Quadruple Patterning Aware Grid Routing Method." IEEE TCAD, Vol.34, No. 5, pp.753-765, 2015

*2 Ihara,Hongo,Takahashi,Kodama. "Grid-based Self-Aligned Quadruple Patterning Aware Two Dimensional Routing Pattern." DATE 2016

Grid-Based Routing for SADP

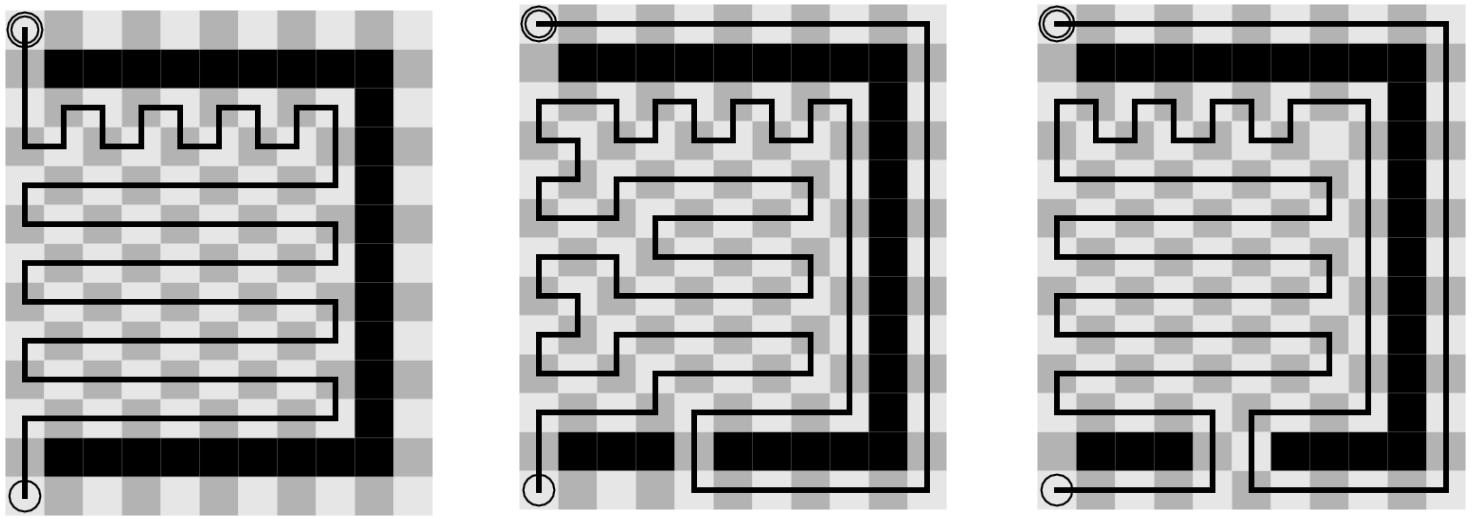
- Self-Aligned Double Patterning (SADP)
  - ✓ "Partially Pre-colored Two-color grid" (Kodama Grid) [ASP-DAC2013]
- Enhancement of Kodama Grid based Method
  - ✓ Routing pattern generation method for SADP
    - Manufacturing by SADP process is guaranteed if routing on the grid is completed
    - ✓ No Limitation on Pin Location in Netlist
      - ✓ Convert [recoloring grid, pin extension] into single-colored net



* Ihara, Takahashi, Kodama. "Rip-up and Reroute based Routing Algorithm for Self-Aligned Double Patterning." Proc. SASIMI 2015.
* Ihara, Takahashi, Kodama. "Effective Two-Dimensional Pattern Generation for Self-Aligned Double Patterning." Proc. ISCAS, 2015.

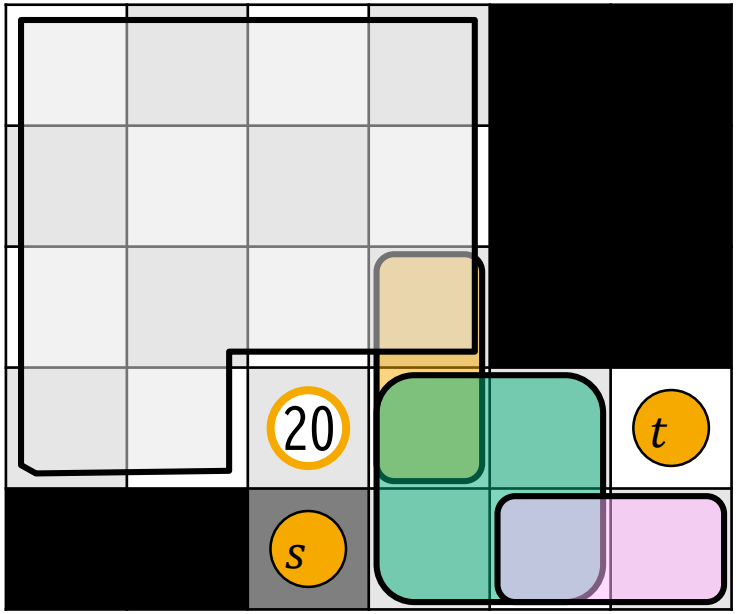
Longest Path Problem

- Not basic but required some cases
  - Length control in restricted routing resources
- Find a longest path between two nodes
  - NP-hard



Longer Path Algorithm

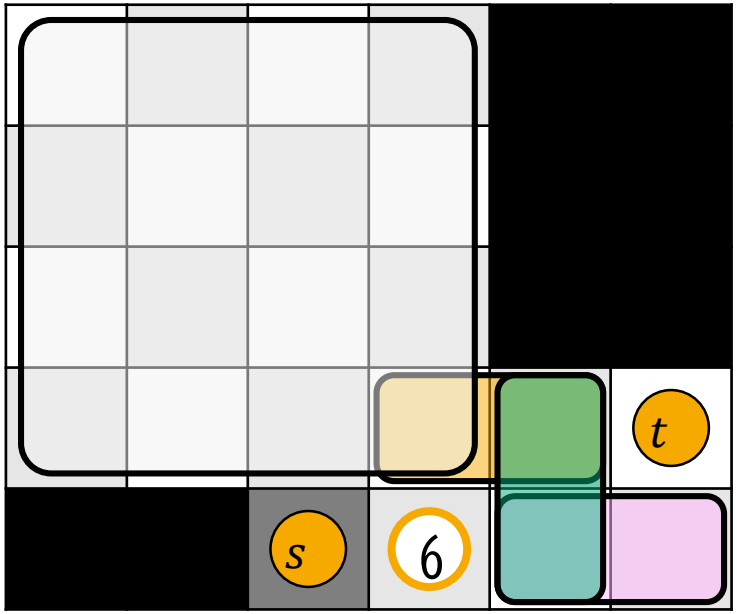
- Heuristic Algorithm
  - US (Upper bound based Seed) Routing
- Select direction that has higher potential extend
- Bi-connectivity based length upper bound



- Kohira, Suehiro, Takahashi. “A Fast Longer Path Algorithm for Routing Grid with Obstacles using Biconnectivity based Length Upper Bound.” IEICE Trans. Fundamentals, Vol.E92-A, No.12, pp.2971-2978, December 2009.

Longer Path Algorithm

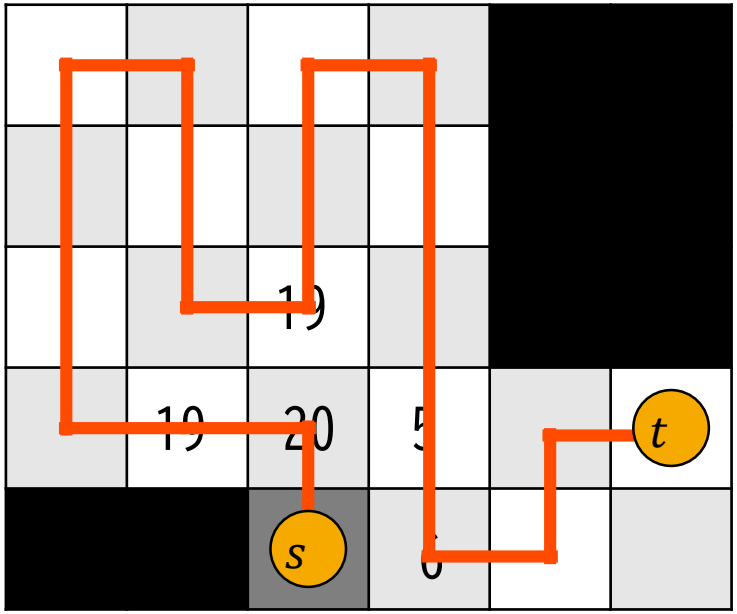
- Heuristic Algorithm
  - US (Upper bound based Seed) Routing
- Select direction that has higher potential extend
- Bi-connectivity based length upper bound



- Kohira, Suehiro, Takahashi. “A Fast Longer Path Algorithm for Routing Grid with Obstacles using Biconnectivity based Length Upper Bound.” IEICE Trans. Fundamentals, Vol.E92-A, No.12, pp.2971-2978, December 2009.

Longer Path Algorithm

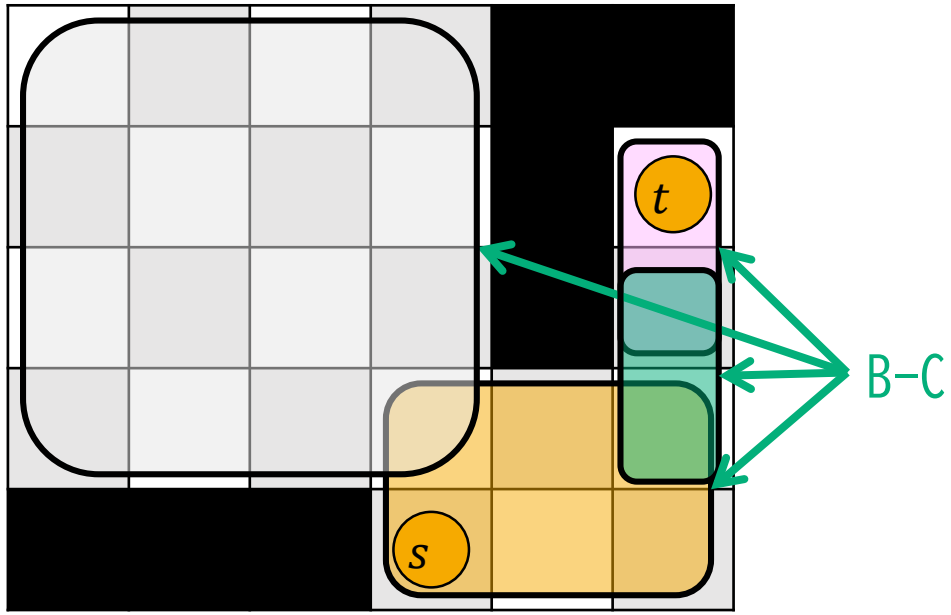
- Heuristic Algorithm
  - US (Upper bound based Seed) Routing
- Select direction that has higher potential extend
- Bi-connectivity based length upper bound



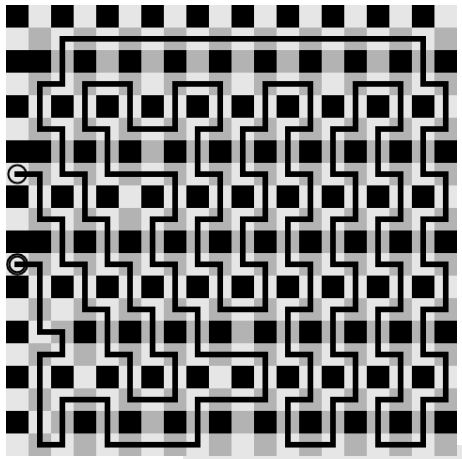
- Kohira, Suehiro, Takahashi. “A Fast Longer Path Algorithm for Routing Grid with Obstacles using Biconnectivity based Length Upper Bound.” IEICE Trans. Fundamentals, Vol.E92-A, No.12, pp.2971-2978, December 2009.

Bi-connectivity based Upper Bound

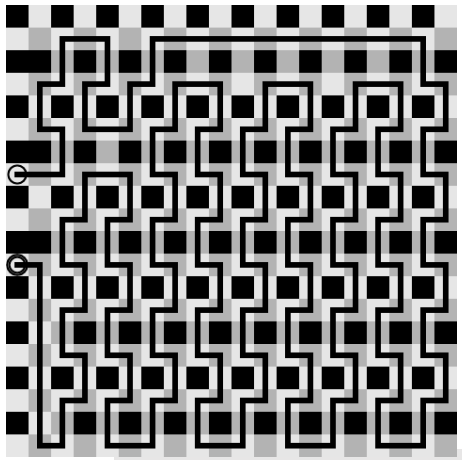
- B-C Sequence from start to goal is unique
  - ✓ biconnected
    - connected graph even if a vertex is removed
  - ✓ biconnected component (B-C)
    - maximal biconnected subgraph



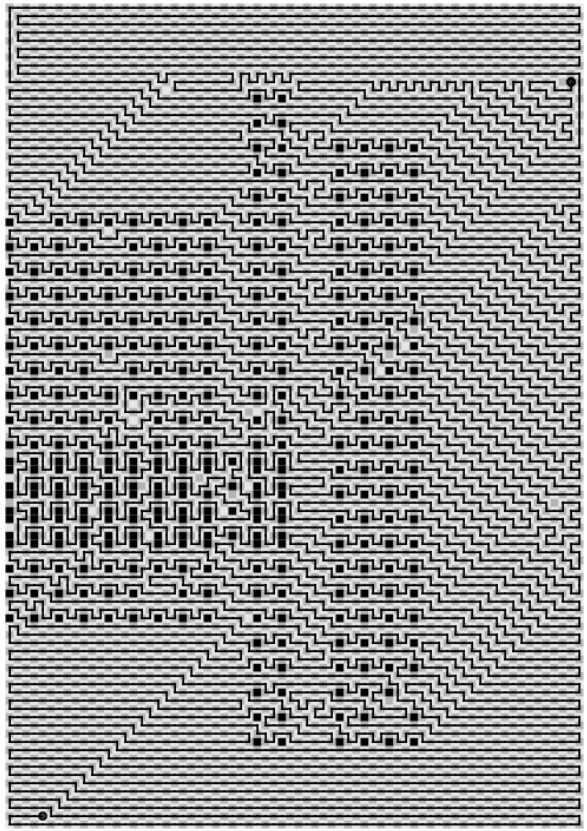
Longer Path (US) Experiments



US: 251



Furthest + C-Flip: 261



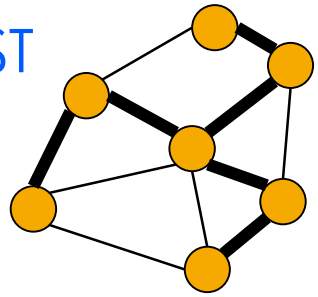
US: 6626

U_p=6654
U_c=6650
U_a=6650

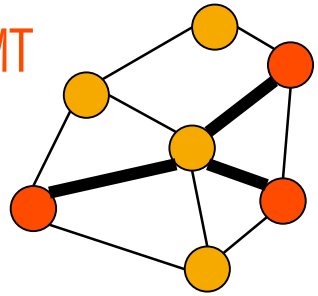
Tree Problems

- Minimum Spanning tree (MST)
  - Connect all vertices by minimum total weight
  - **P** problem
  - Kruskal Algorithm, Prim Algorithm
- Minimum Steiner tree (SMT: Steiner Minimum Tree)
  - Connect specified vertices by minimum total weight
  - **NP**-hard
  - Rectilinear SMT
    - Hannan Grid

MST

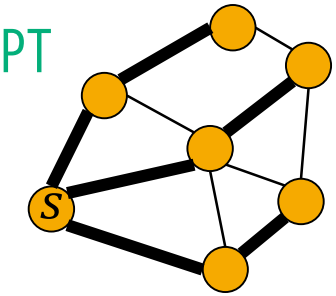


SMT



- Shortest Path Tree (SPT)
  - Connect sinks by shortest path from source
  - **P** problem
  - Dijkstra Algorithm

SPT



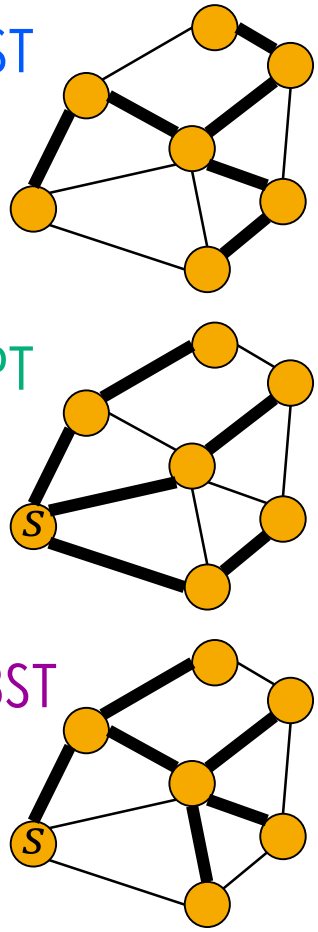
- Minimum Shortest Path Tree (MSPT)
  - Connect sinks by shortest path from source and by minimum total weight
  - **NP**-hard ???
    - There were several incorrect proofs

Cost and Radius in Tree

- Minimum Spanning tree (MST) : P problem
  - Cost is minimum, but Radius might be large
  - Prim Algorithm: $0 \times l(x) + w(x,v)$

- Shortest Path Tree (SPT) : P problem
  - Radius is minimum, but Cost might be large
  - Dijkstra Algorithm: $1 \times l(x) + w(x,v)$

- Cost/Radius Balanced (Steiner) Tree (CRBST) [*]
  - Balance Cost and Radius
  - CRBST Algorithm: $\frac{\mathrm{dist}(s,v)}{B} \times l(x) + w(x,v)$
    - v is near : $w(x,v)$ = cost first
      far : $l(x) + w(x,v)$ = radius first
  - Radius is bounded by B



* Mitsubayashi, Takahashi, Kajitani. "Cost-Radius Balanced Spanning/Steiner Trees." IEICE Trans. Fundamentals, Vol. E80-A, No. 4, pp. 689-694, 1997

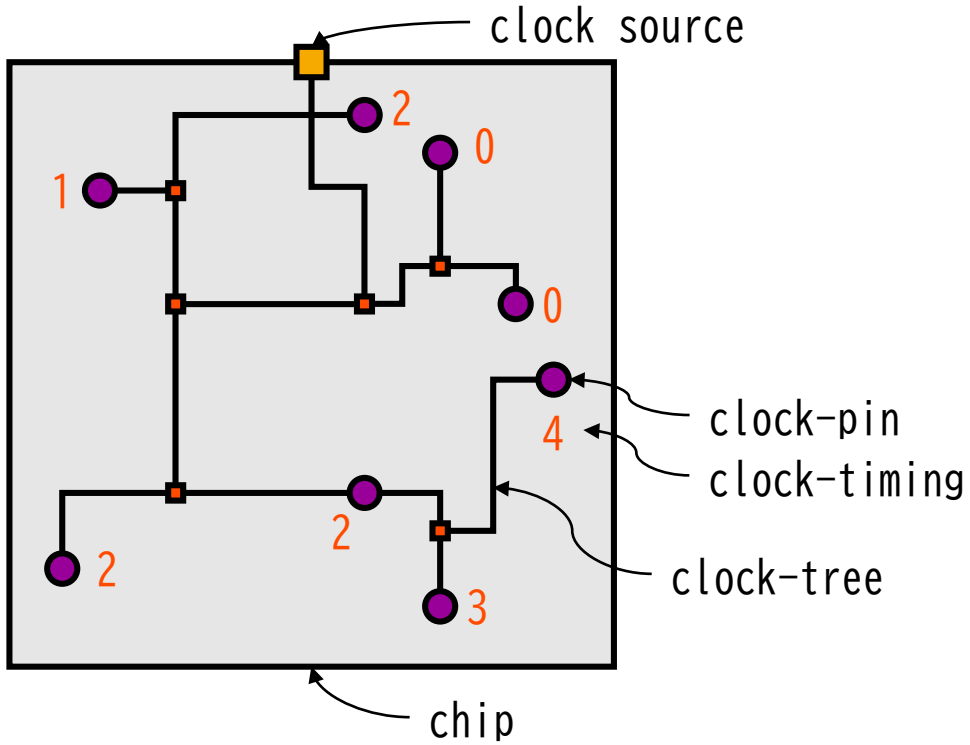
Clock Distribution Network

- Tree or Mesh or Others?
- Clock Skew
  - Zero Clock Skew or Clock Schedule?
- Total Wire Length
- Power Consumption
- Buffer Insertion
- …

- Clock Tree
  - Deferred-Merge Embedding
    - Zero-skew clock-tree [Chao et al 92, Edahiro 93, ...]
    - Bounded-skew clock-tree [Huang et al. 95, ...]
    - Useful-skew clock-tree [Xi&Dai 96]

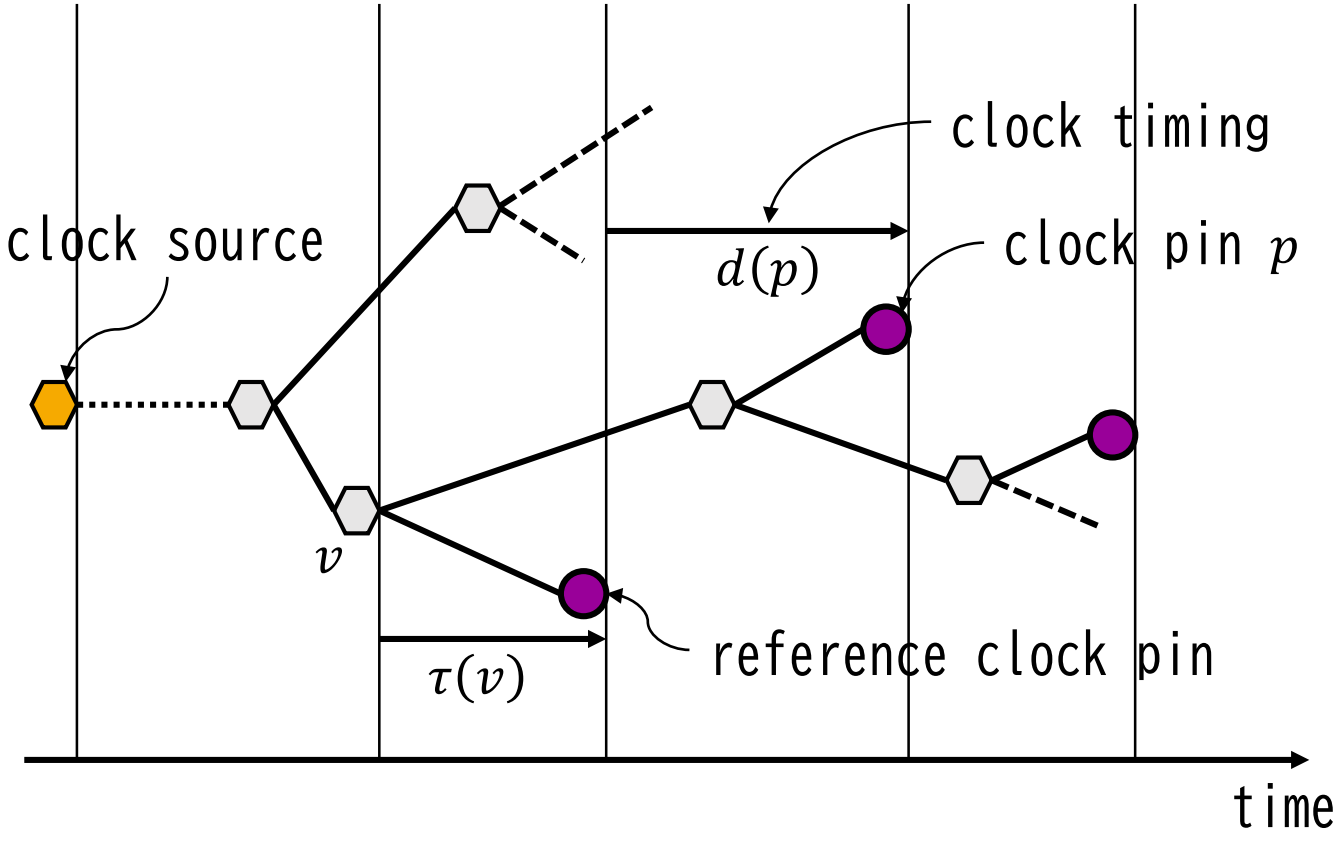
Schedule-Clock-Tree

- Schedule-clock-tree routing (SCT-routing)
- Input: clock-schedule
- Output: clock-tree that realizes the clock-schedule



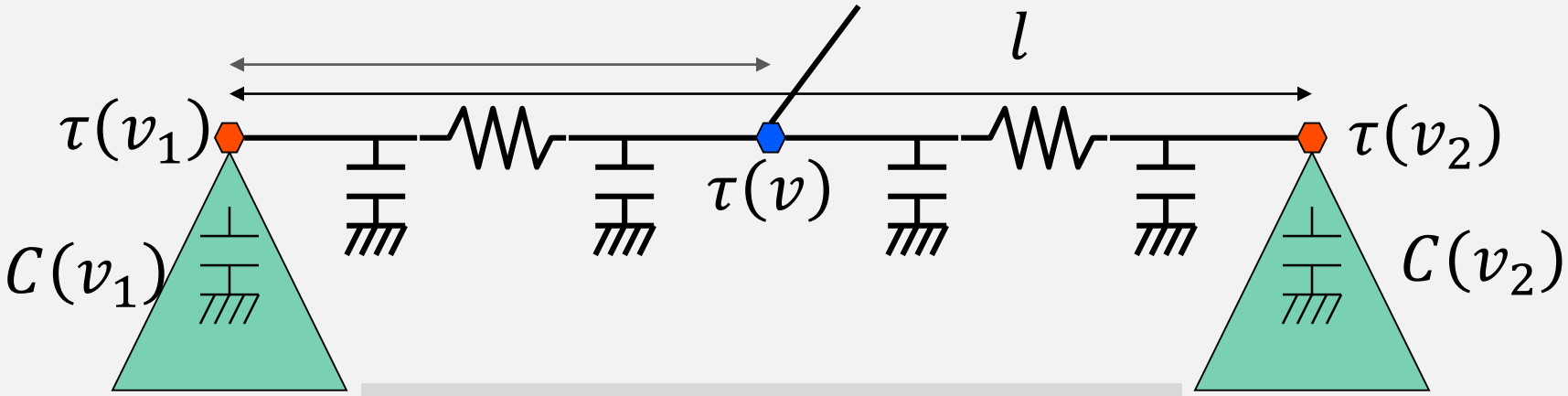
* Inoue, Takahashi, Takahashi, Kajitani. "Schedule-Clock-Tree Routing for Semi-Synchronous Circuits." IEICE Transactions on Fundamentals, Vol.E82-A, No.11, pp.2431-2439, Nov. 1999

Delays in Schedule-Clock-Tree



Delay-Balance-Point

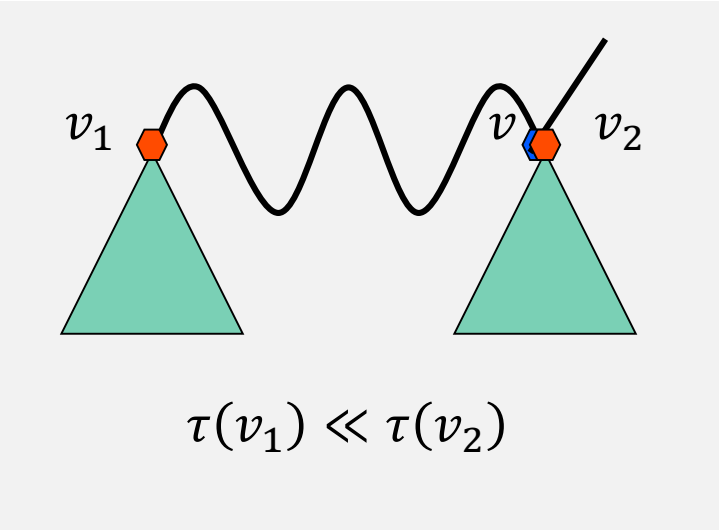
- Equation in a schedule-clock-tree
  - $\tau(v) = rl_1\left(\frac{cl_1}{2} + C(v_1)\right) + \tau(v_1) = r(l - l_1)\left(\frac{c(l-l_1)}{2} + C(v_2)\right) + \tau(v_2)$

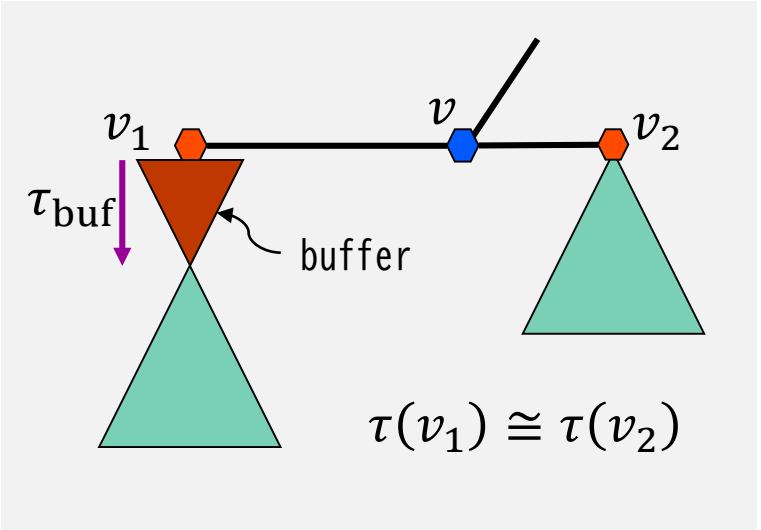


$$l_1 = \frac{\tau(v_2) - \tau(v_1) + rl\left(C(v_2) + \frac{cl}{2}\right)}{r\left(cl + C(v_1) + C(v_2)\right)}$$

Buffer Insertion

- To reduce the connection length
  - If the detour occurs





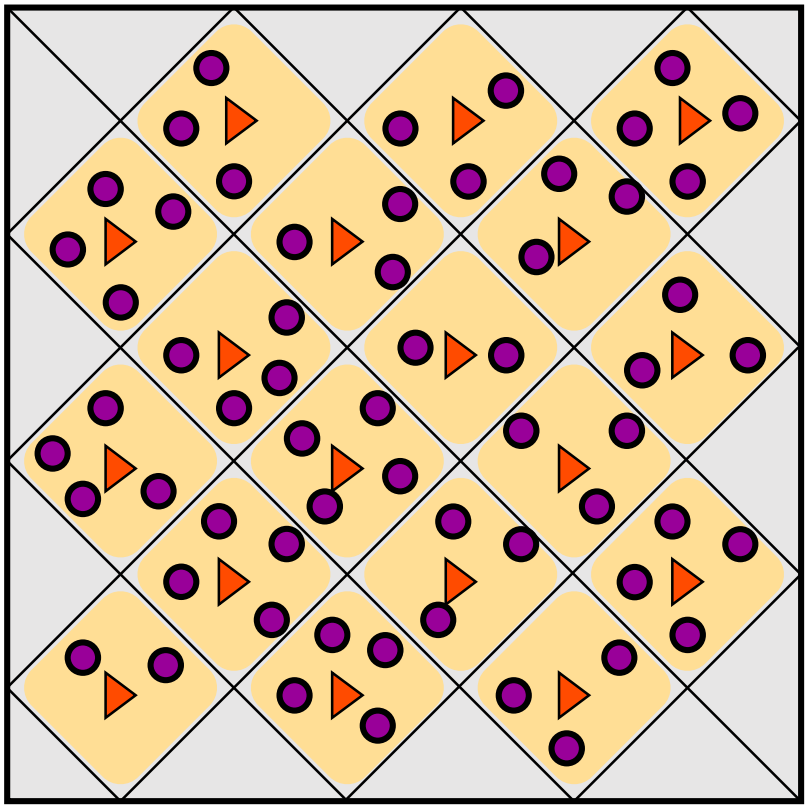
Lower cost solution is selected

SCT-Routing

- Topology generation
  - Find a pair with the minimum merging cost
  - Insert buffer, if necessary
  - Compute the merging-segment
    consisting of delay-balance-points
  - Repeat until complete topology is obtained

- Topology embedding
  - Determine the exact location of each node
    from the clock-source to the clock-pins

Clustering Based Light Clock-Tree



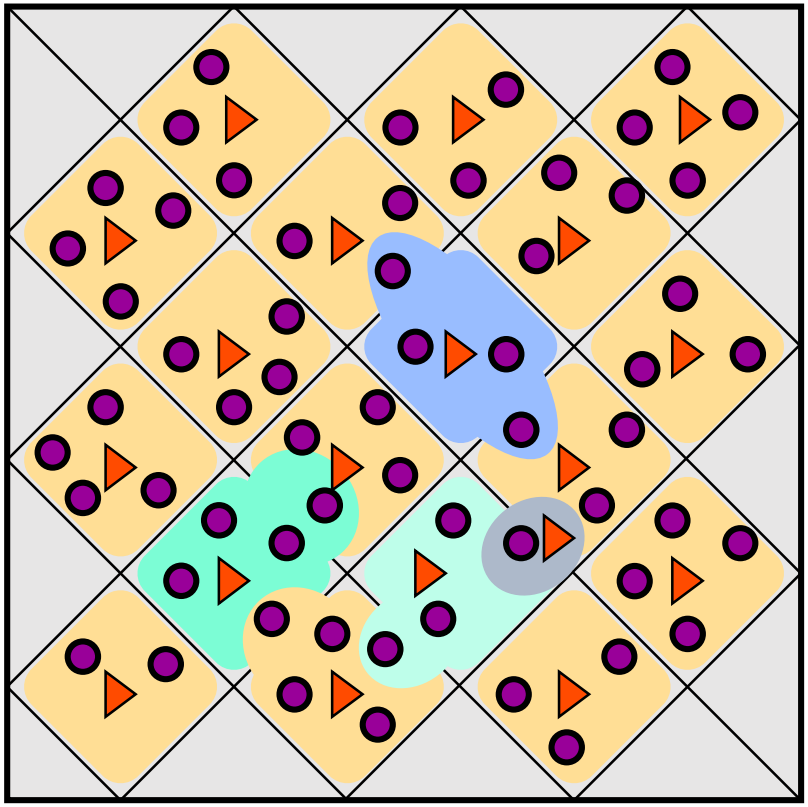
register
clock buffer

1. Construct initial clusters
2. Modify clusters

- Objective
  - Clock period
- Constraints
  - Timing constraint aware
  - Radius of cluster
  - Number of clusters

* Saitoh, Azuma, Takahashi.
"A Clustering Based Fast Clock Schedule Algorithm for Light Clock-Trees"
IEICE Trans. Fundamentals, Vol.E85-A, No.12, pp.2756-2763, 2002

Clustering Based Light Clock-Tree



register
clock buffer

1. Construct initial clusters
2. Modify clusters

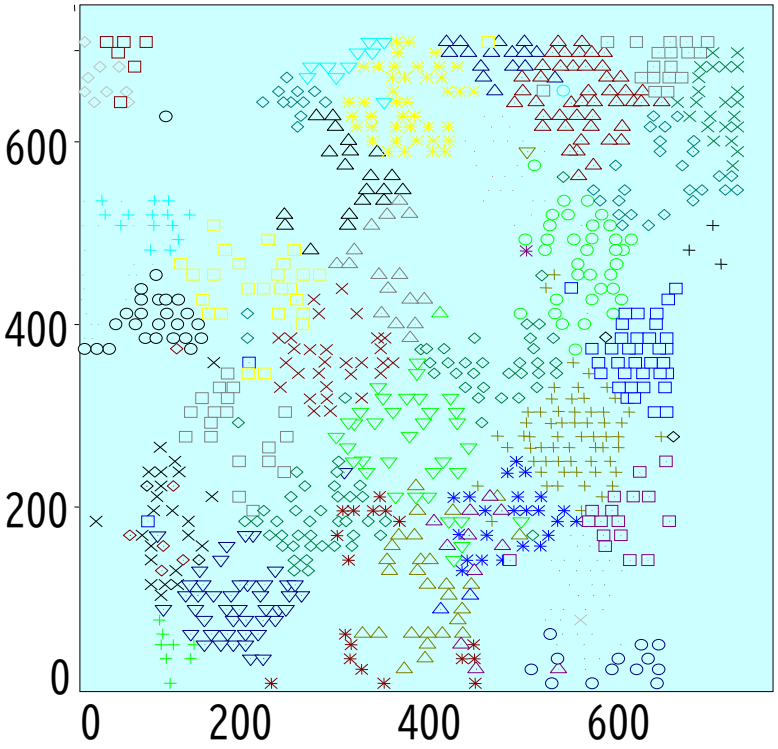
■ Objective
- Clock period

■ Constraints
- Timing constraint aware
- Radius of cluster
- Number of clusters

* Saitoh, Azuma, Takahashi.
"A Clustering Based Fast Clock Schedule Algorithm for Light Clock-Trees"
IEICE Trans. Fundamentals, Vol.E85-A, No.12, pp.2756-2763, 2002

Clustering Based Light Clock-Tree

■ Experimental Result



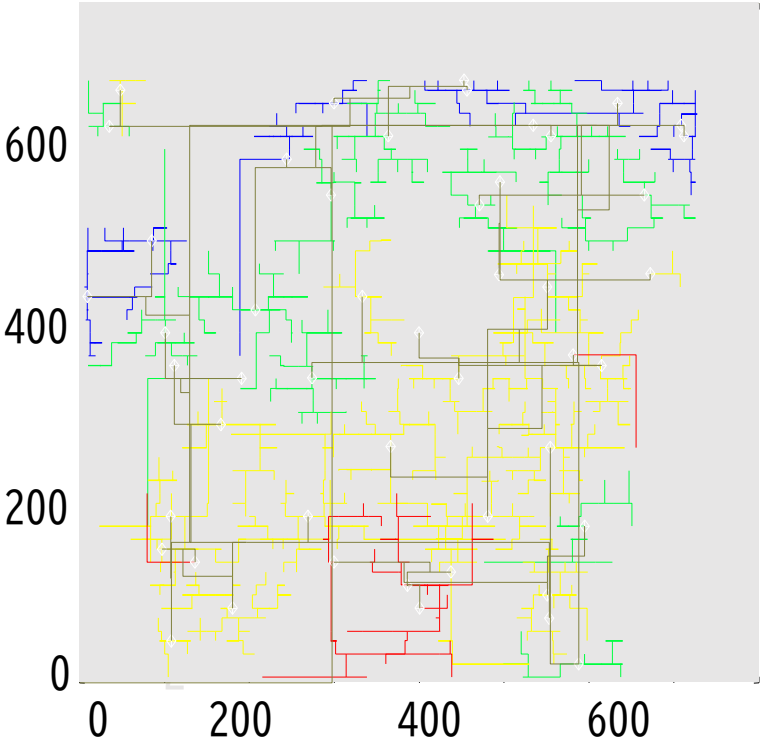
Process : 0.25 (μm)
Chip area: 728×710 (μm^2)
#register: 888
Max delay: 11.6 (ns)

⇩ Computation time : 24.9 s

Max cluster radius: 300 (μm)
Number of cluster: 49

Clustering Based Light Clock-Tree

■ Experimental Result

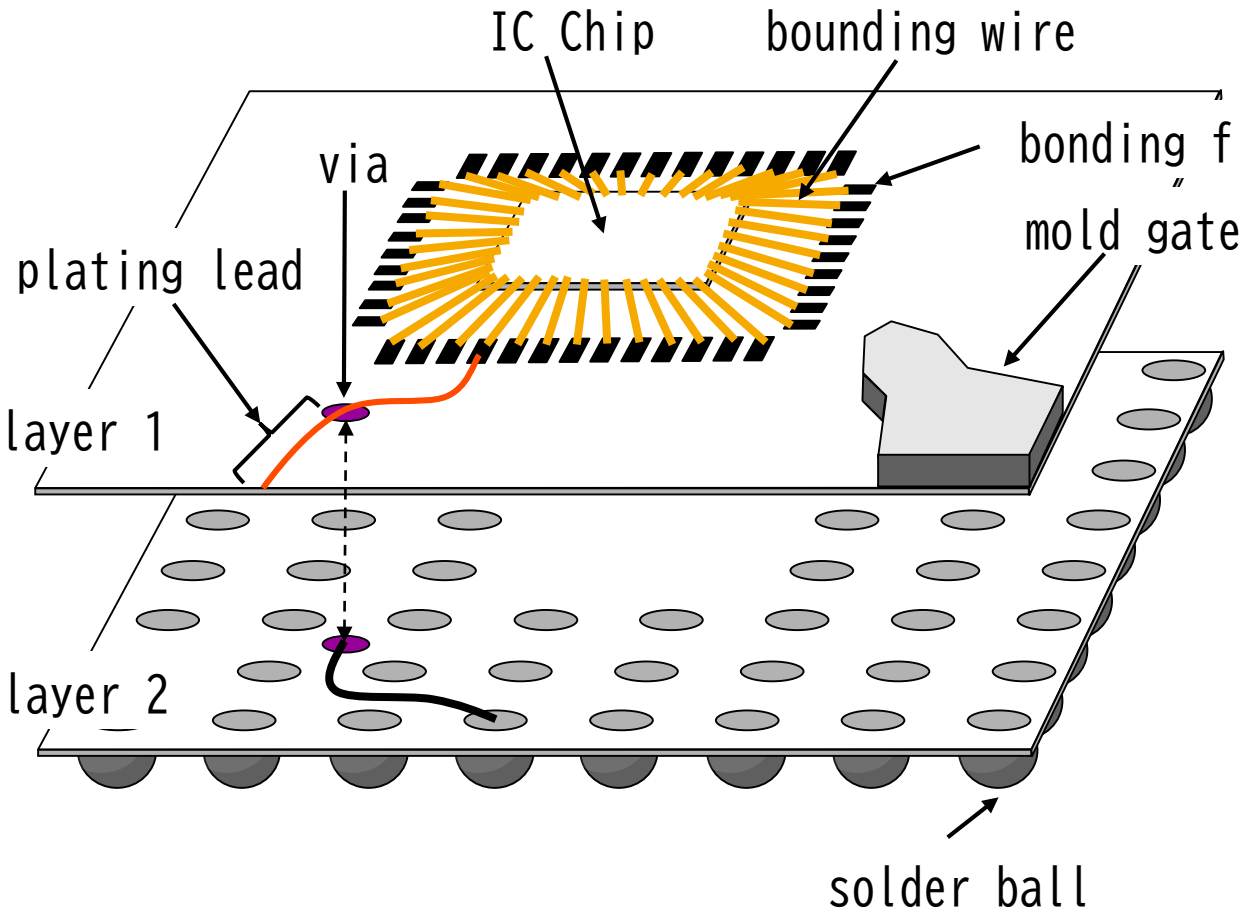


Process : 0.25 (μm)
Chip area: 728×710 (μm^2)
#register: 888
Max delay: 11.6 (ns)

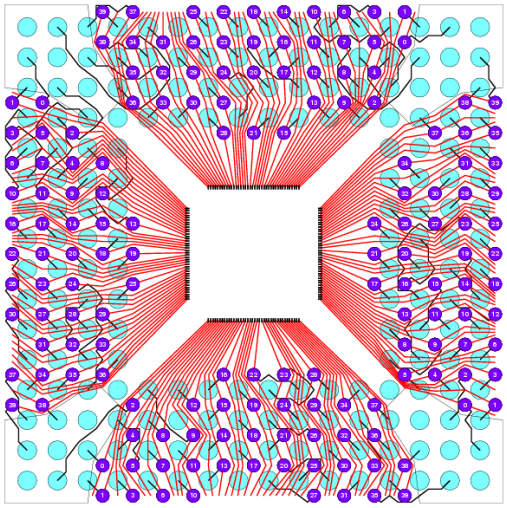
⇩ Computation time : 24.9 s

Max cluster radius: 300 (μm)
Number of cluster: 49
Clock Period: 8.4 (ns)
Wire Length: 30317 (μm)

2-layer BGA Package

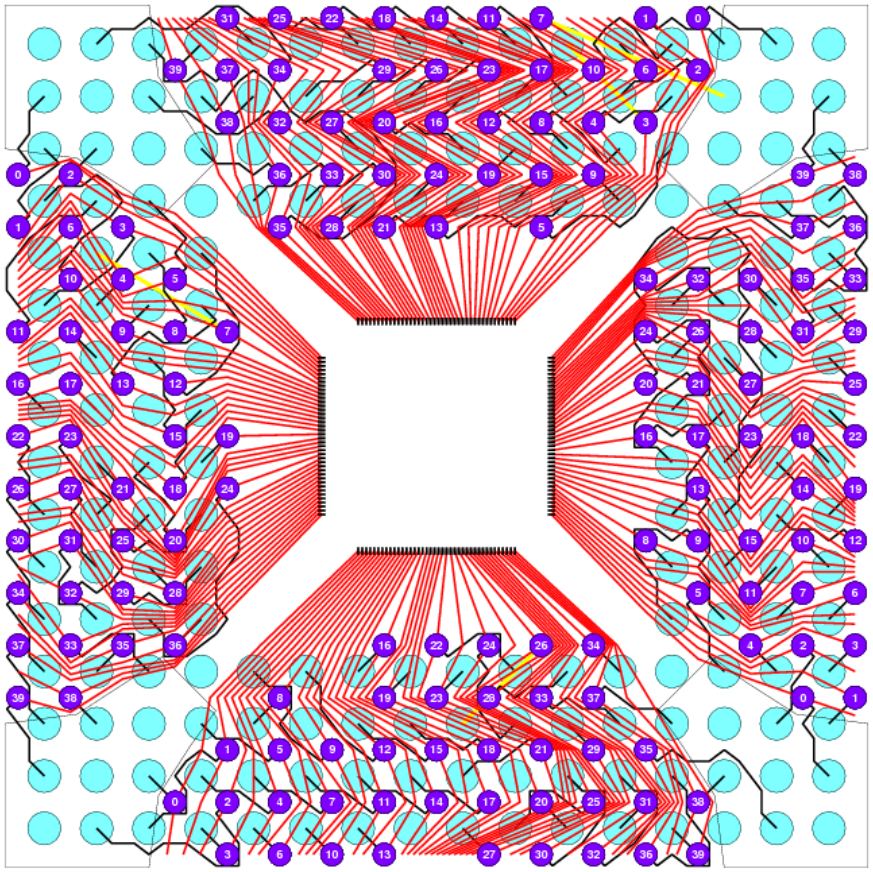


Routing Design

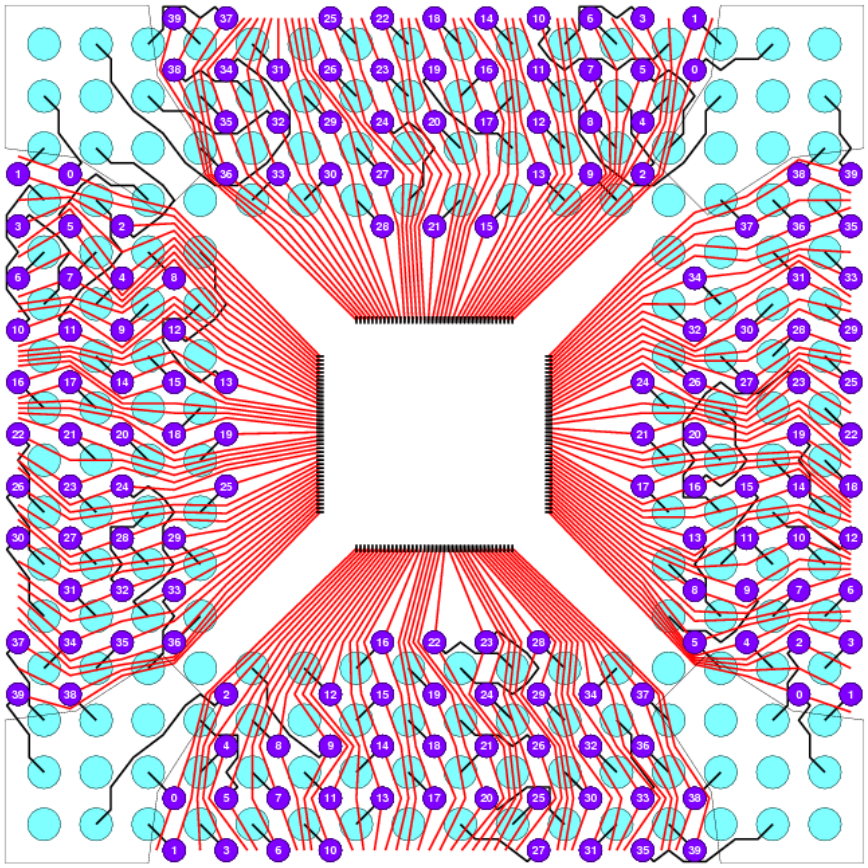


BGA Routing

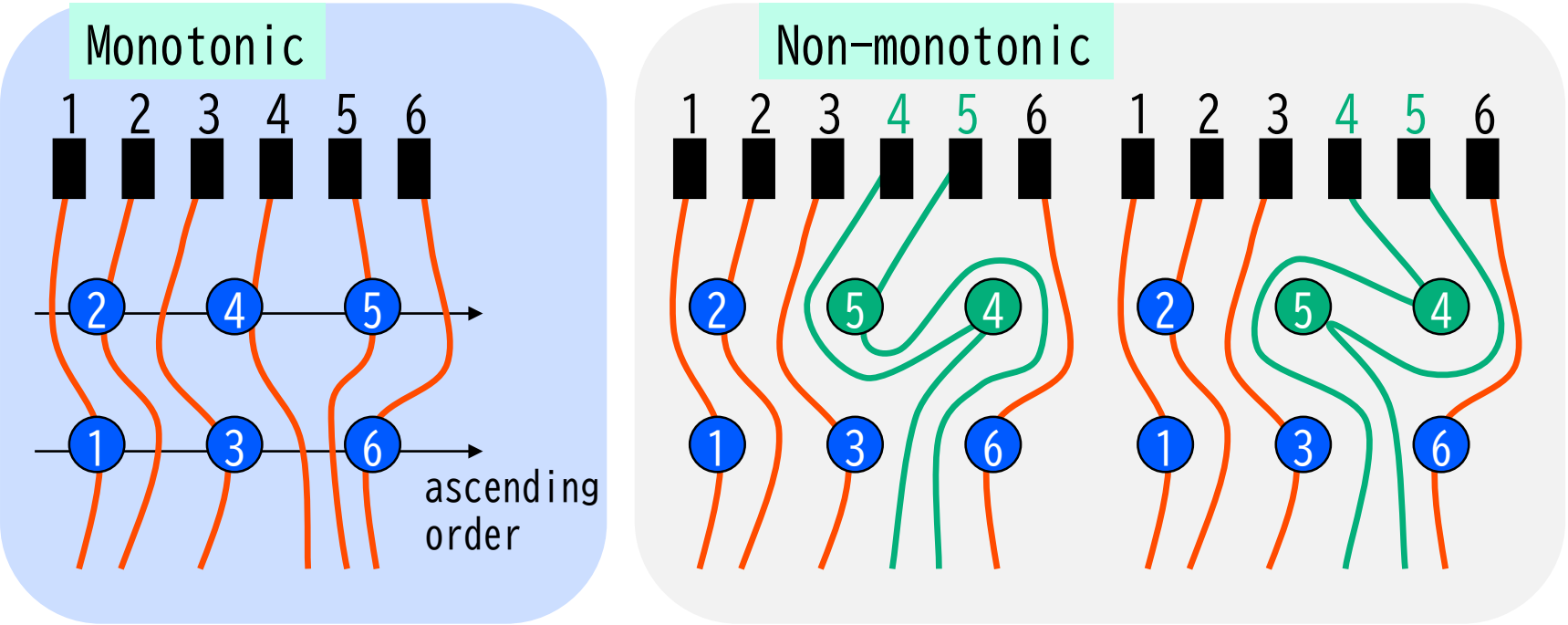
Bad Routing



Better Routing



Monotonic Via Assignment



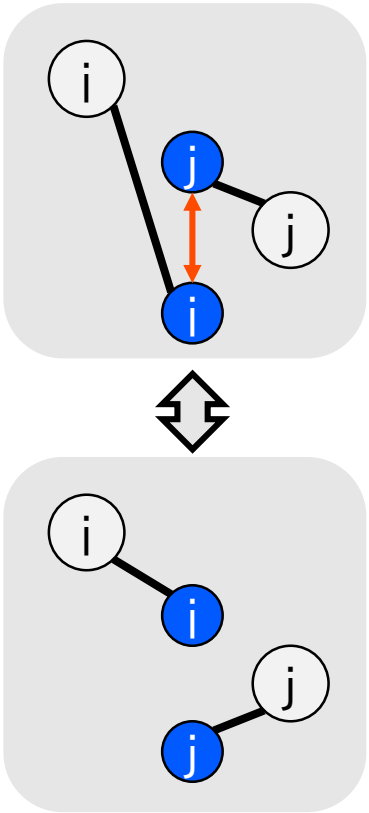
Monotonic Via Assignment

- ✓ All routes on layer 1 can be monotonic
- ✓ Monotonic routing pattern is unique

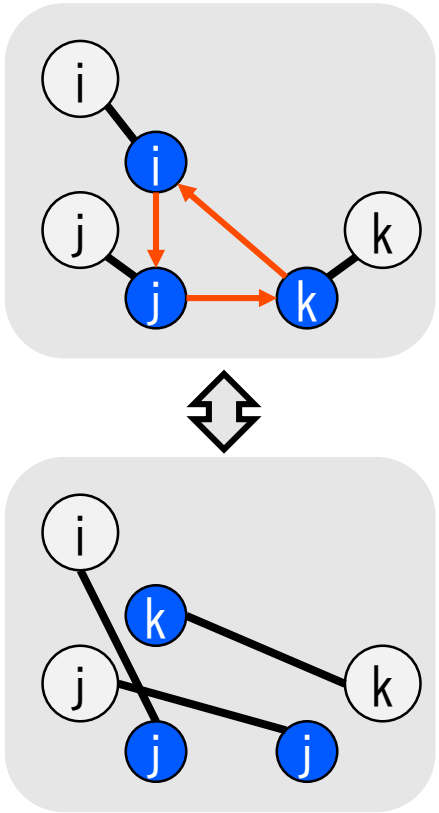
* Tomioka, Takahashi. "Routing of Monotonic Parallel and Orthogonal Netlists for Single-Layer Ball Grid Array Packages. IEICE Trans. Fundamentals, Vol.E89-A, No.12, pp.3551-3559, December 2006.

Modifications

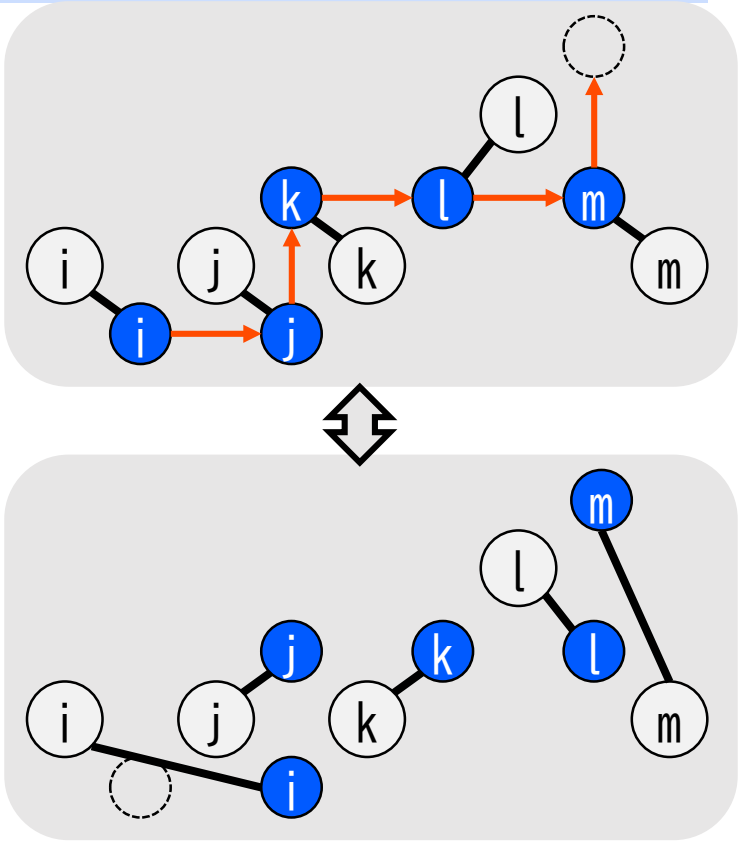
Exchange (EXC)



Rotation (ROT)



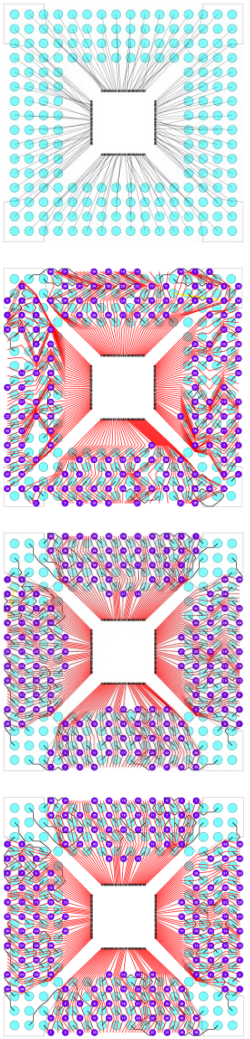
Monotonic Sequence (MSEQ)



* Kubo, Takahashi. "A Via Assignment and Global Routing Method for 2-Layer Ball Grid Array Packages." IEICE Trans. Fundamentals, Vol.E88-A, No.5, pp.1283-1289, May 2005.
* Kubo, Takahashi. "Global Routing by Iterative Improvements for 2-Layer Ball Grid Array Packages." IEEE TCAD, Vol.25, No.4, pp.725-733, April 2006.

BGA Routing Method

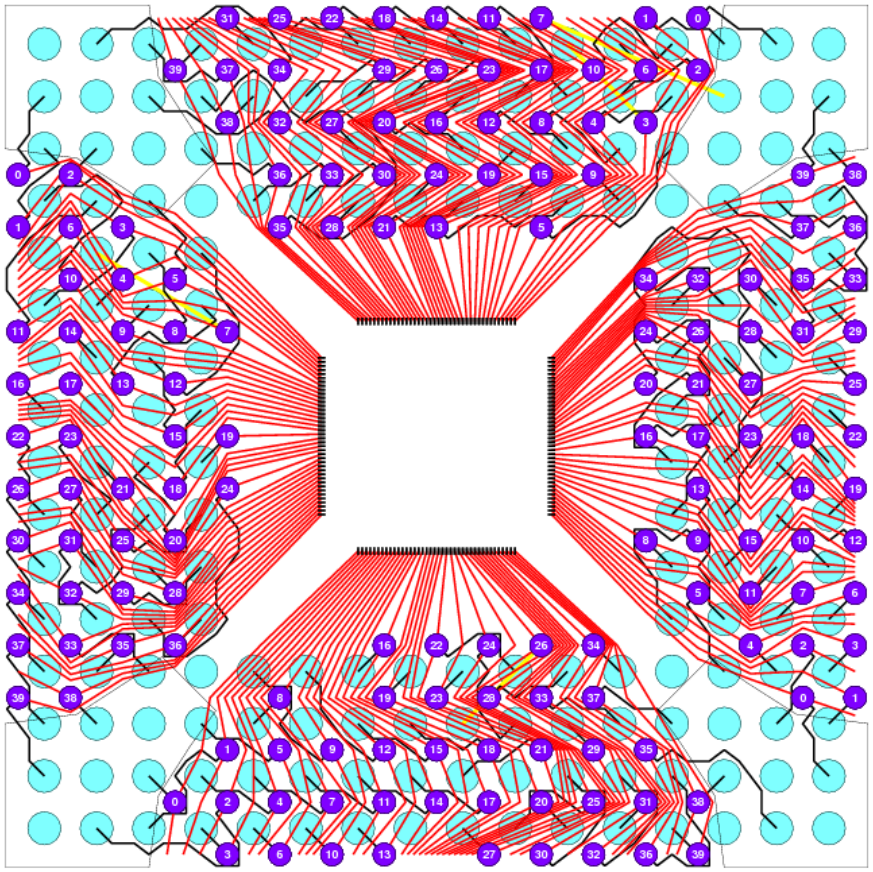
- Initial via assignment
  - Place vias near their balls under monotonic condition
- Iterative via modification
  - Phase 1
    - Improve layer 1, Keep layer 2
    - ✓ improved computation complexity
  - Phase 2
    - Improve layer 2, Keep layer 1
    - ✓ new modification introduced
- Via assignment
- Global Routing



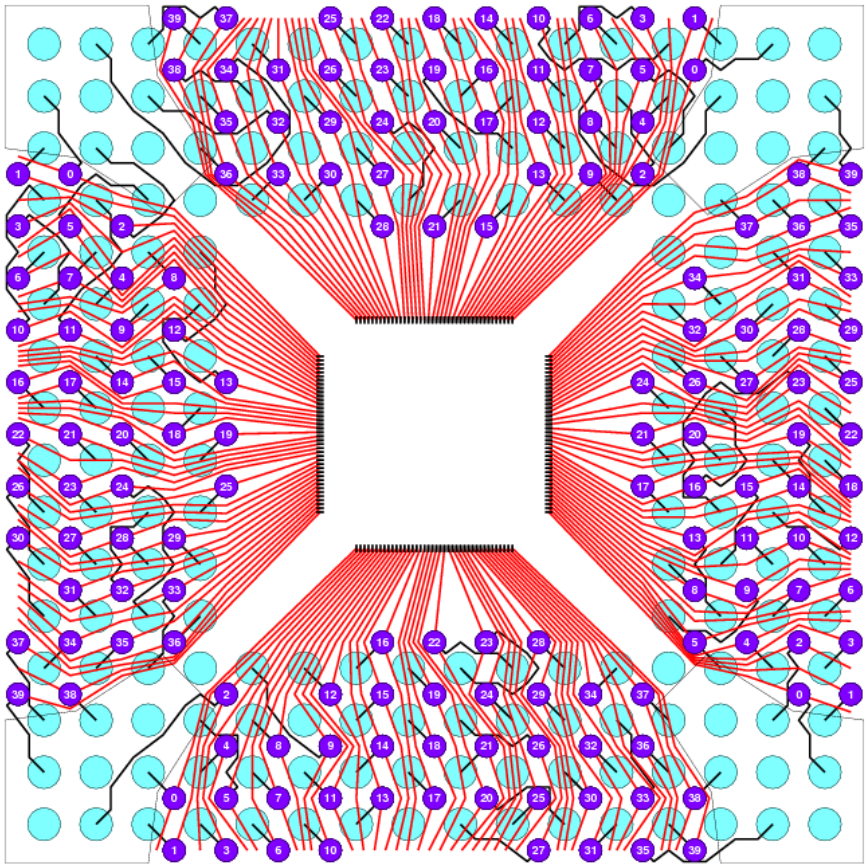
* Tomioka, Takahashi. "Routability Driven Via Assignment Method for 2-Layer Ball Grid Array Packages." IEICE Trans. Fundamentals, Vol.E92-A, No.6, pp.1433-1441, June 2009.

BGA Routing

Bad Routing

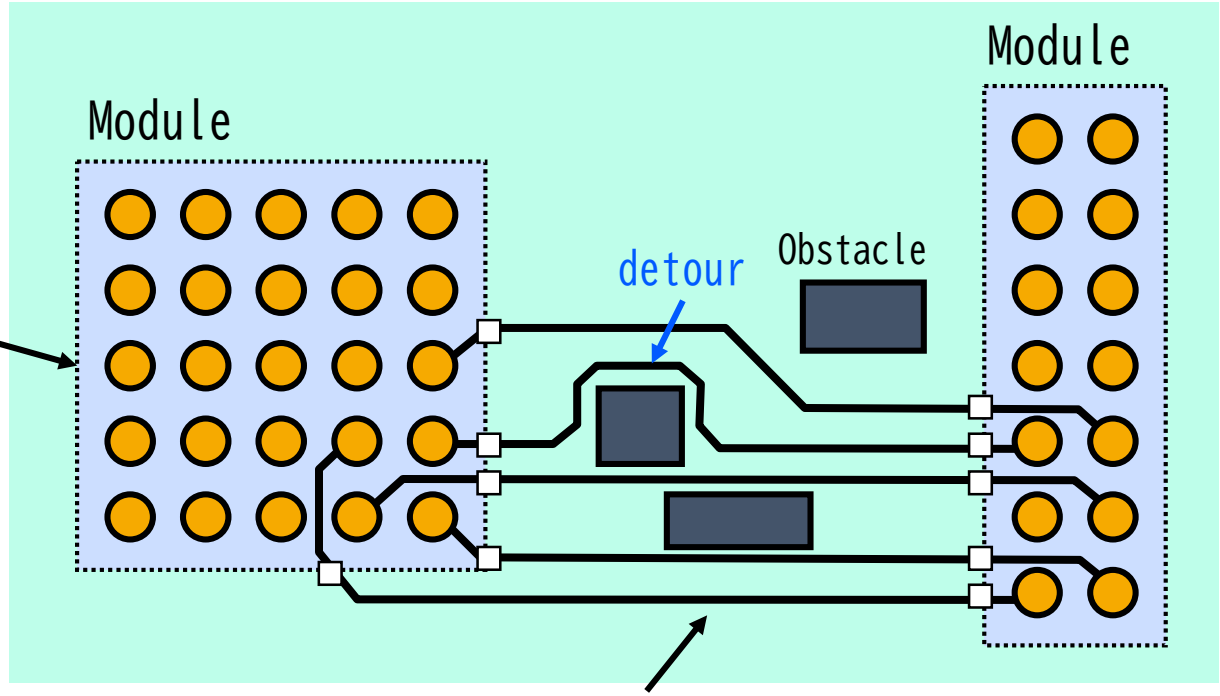


Better Routing



PCB Routing

- Divide into subareas in terms of the obstacle density

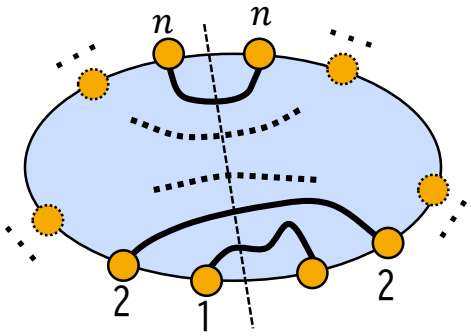


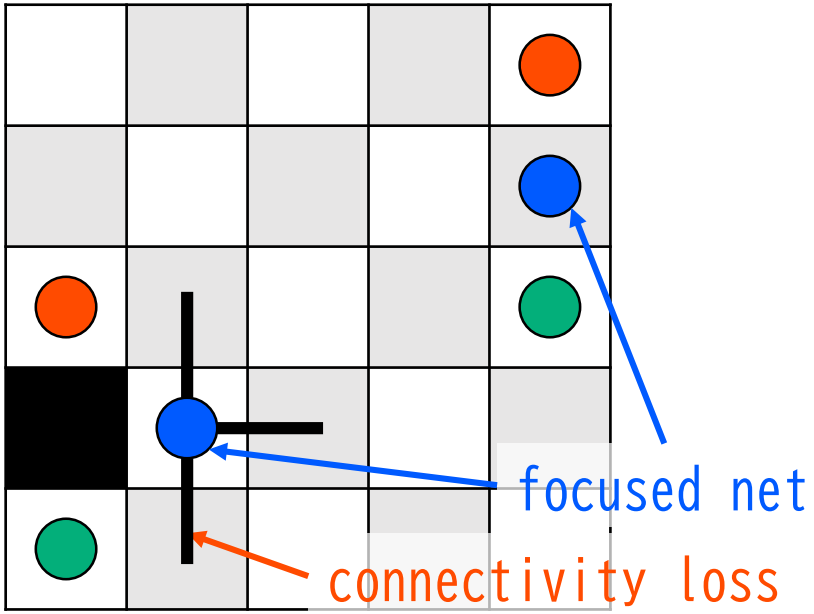
Escape routing
Escape first

River routing
Length Control etc.

CAFE router

- CAFE: Connectivity Aware Frontier Exploration
  - Until all nets are connected, following procedures are repeated
    1. Choose a focused net whose frontier is moved
    2. Check the connectivity for each adjacent grid
    3. Move the frontier to an adjacent grid

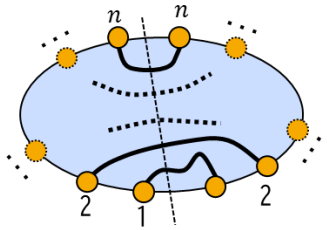


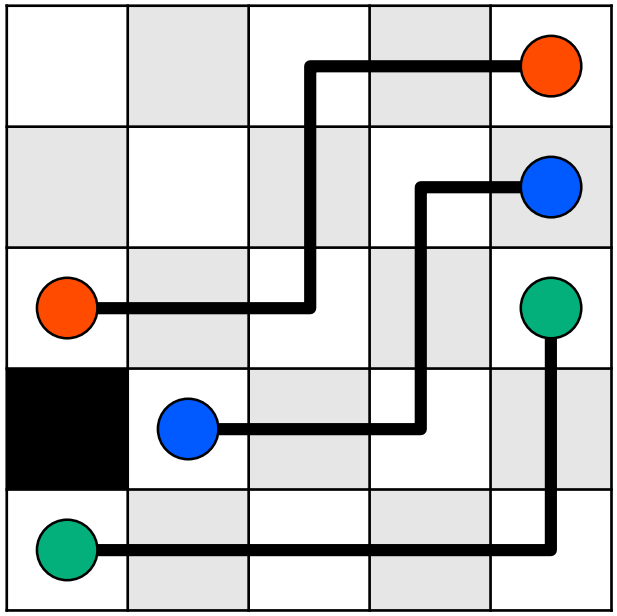


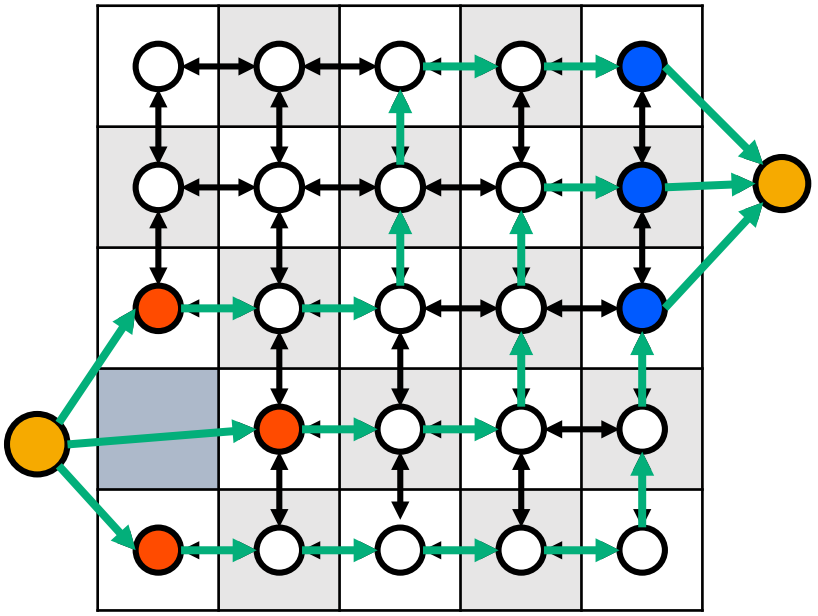
* Kohira,Takahashi.“CAFE router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles”. IEICE Trans. Fundamentals, Vol.E93-A, No.12, pp.2380-2388, 2010.

Connectivity Check

- Trunk Routing Problem
  - Flow with #nets corresponds to feasible routes
  - Max-Flow = #nets: can be connected
  - Max-Flow < #nets: cannot be connected



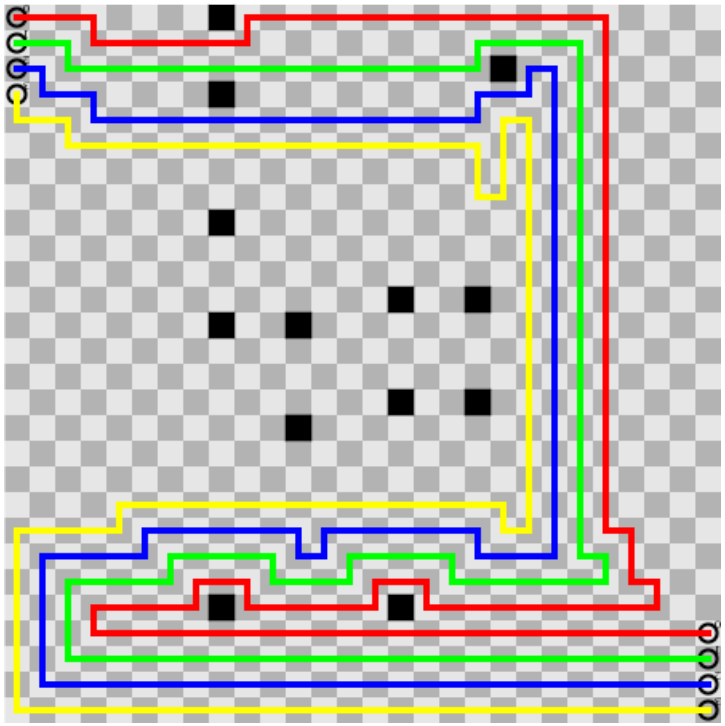




Cafe router Experiments

Sink

Target length: 100



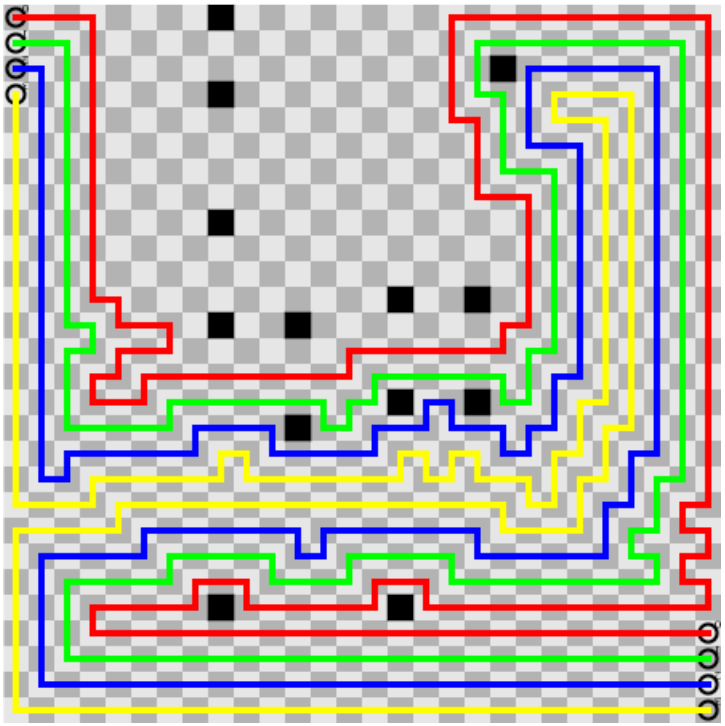
Source

error
net0= 0
net1= 0
net2= 0
net3= 0

ave. err.: 0.0
worst err.: 0
CPU time: 0.35[s]

Sink

Target length: 150



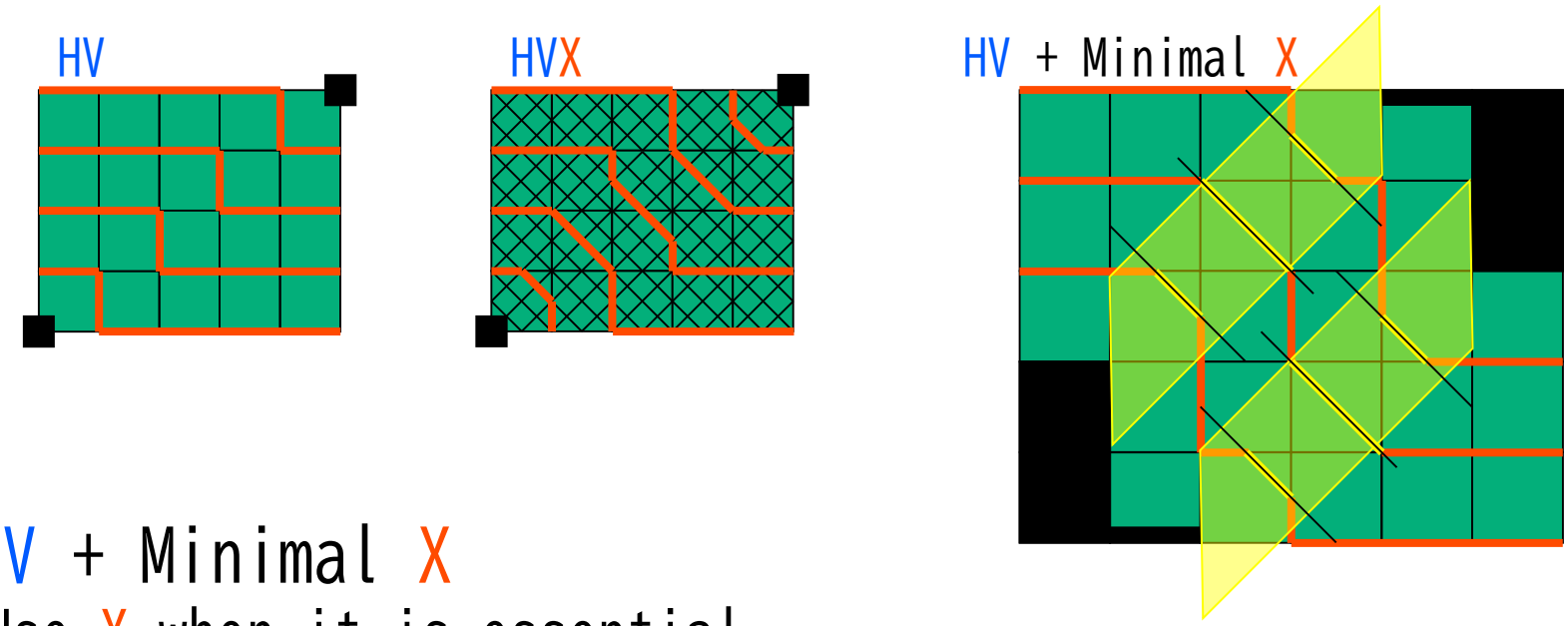
Source

error
net0= 0
net1=-2
net2=-4
net3=-4

ave. err.: 2.5
worst err.:-4
CPU time: 0.42[s]

HVX routing

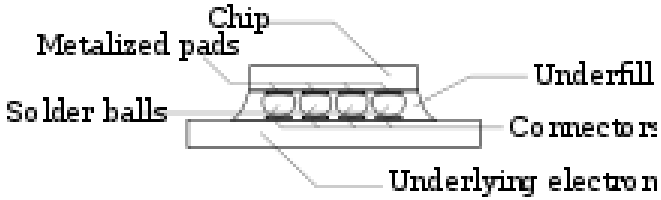
- HV might not be enough for connectivity
- HVX might be too complex

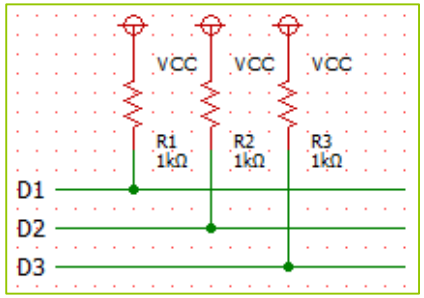


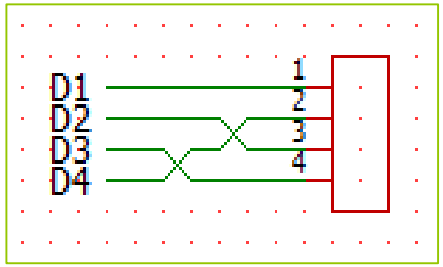
- HV + Minimal X
  - Use X when it is essential
  - Extract critical region and use X
  - After connectivity is achieved, X might be used in length control in non-critical region

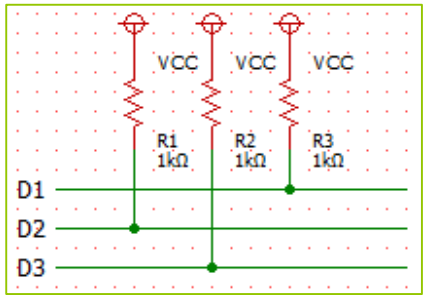
Set-Pair Routing

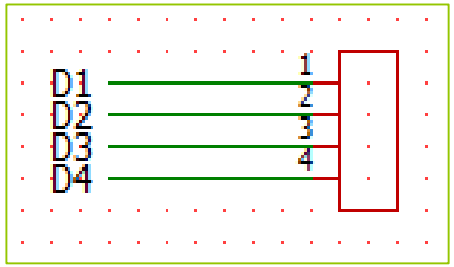
- Connection requirements are not defined on pin pair
  - Connection to equivalent passive elements
  - I/O pins of reconfigurable chip
- Network Flow algorithms work well
  - Minimum Length is easy
  - Length Control is not easy









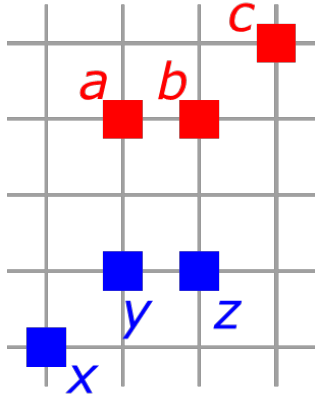


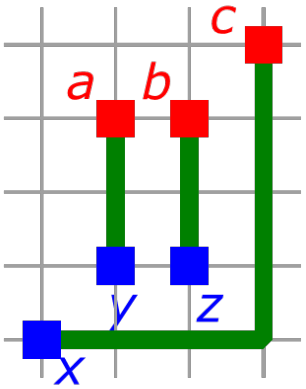
Set-Pair Routing Problem

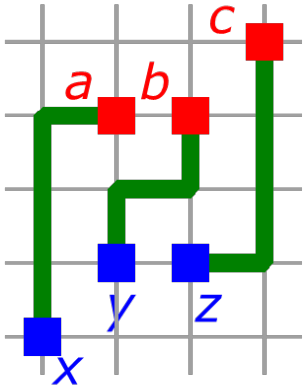
Input : Pin-set Pair(Source-pin set *S*, Sink-pin set *T*)

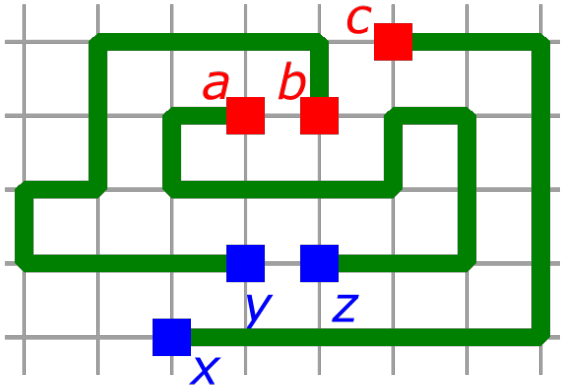
Output: Routing between *S* and *T*

- Assumptions
  - #Source-pin = #Sink-pin
  - one-to-one connection between source-pin and sink-pin
  - planar grid routing (single layer)







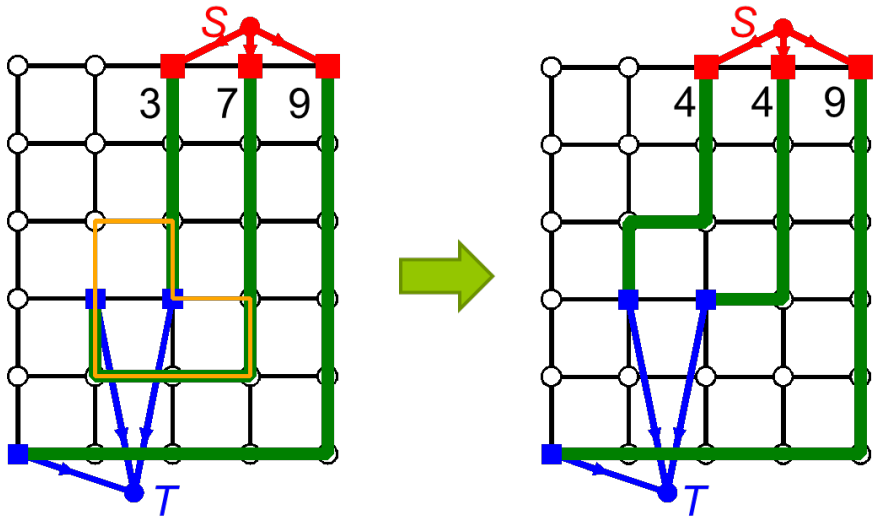


* Nakatani, Takahashi. "A Length Matching Routing Algorithm for Set-Pair Routing Problem" IEICE Trans. Fundamentals, Vol.E98-A, No.12, pp.2565-2571, 2015.

Set-Pair Routing (1)

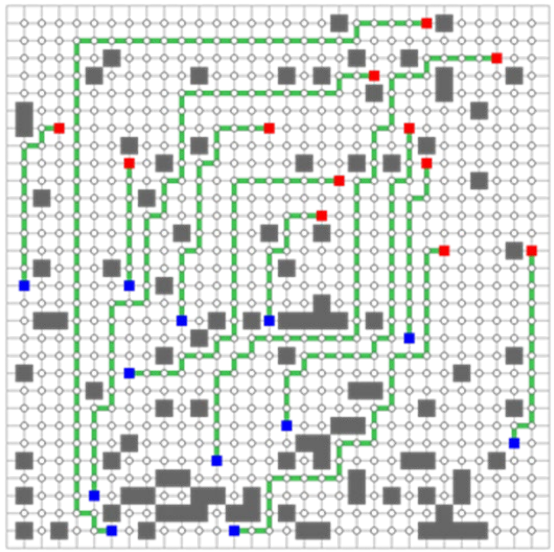
Step 1 (Total Length Reduction)

- ✓ Obtain a routing pattern that has the minimum total wire length by finding minimum cost maximum flow



Initial Max-Flow
Len-Seq (9,7,3) Total:19

Min-Cost Max-Flow
Len-Seq (9,4,4) Total:17



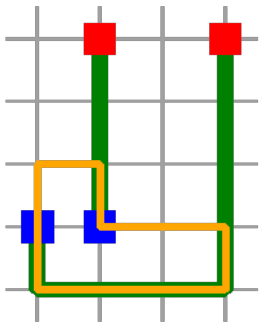
Total:271
Maximum: 51
Difference: 44

Total-Length is optimally minimized
Difference is typically large

Set-Pair Routing (1b) Negative cycle

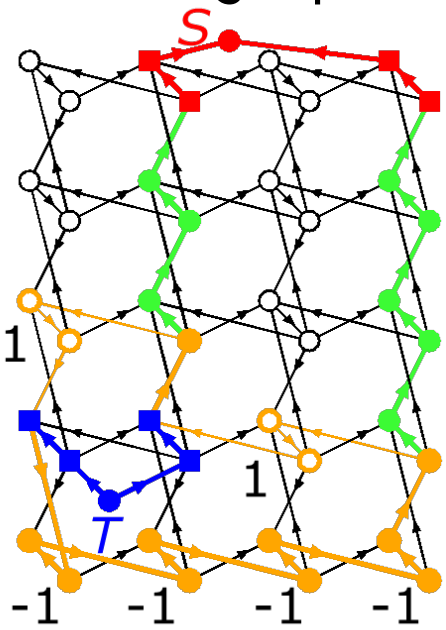
Step 1 (Total Length Reduction)

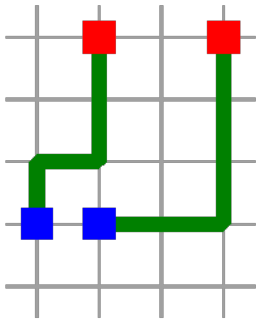
- Negative cycle detection and elimination in residue graph



Len-Seq (8,3)
Total:11 Diff:5

residue graph





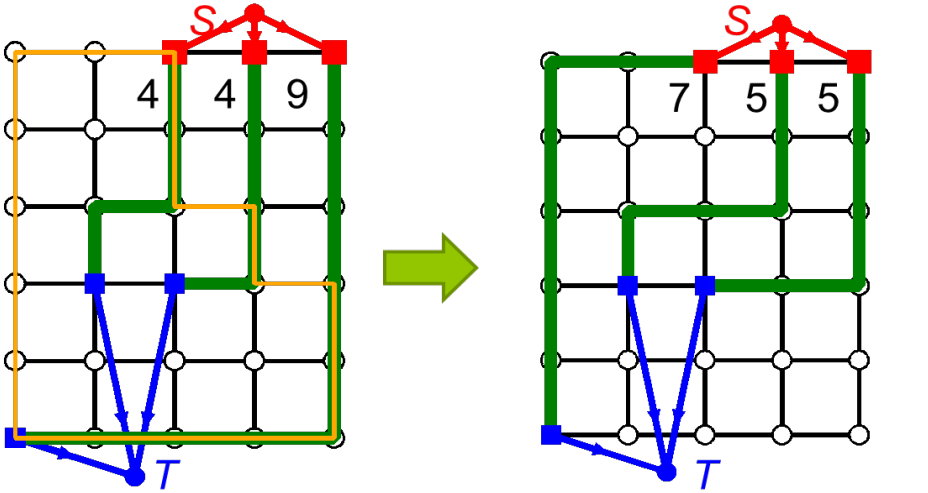
Len-Seq (5,4)
Total:9 Diff:1

No negative weight cycle ≡ Length is minimum

Set-Pair Routing (2)

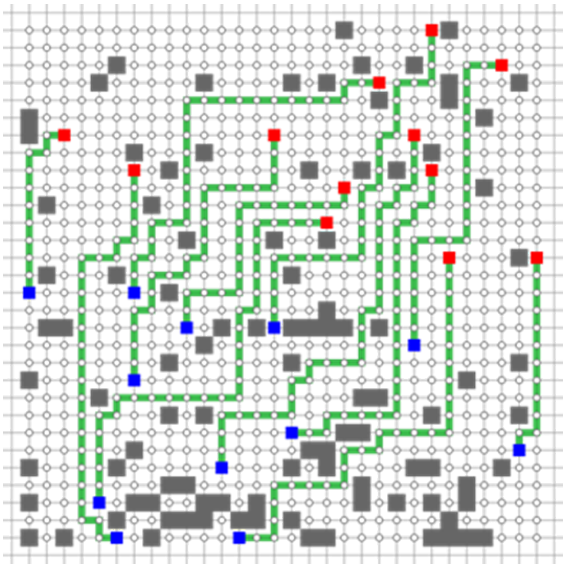
Step 2 (Maximum Length Reduction)

- ✓ Reduce the maximum wire length while keeping the total wire length minimum by greedily modifying a routing pattern so that an earlier Len-Seq is obtained



Min-Cost Max-Flow
Len-Seq (9,4,4) Total:17

Min-Cost Max Reduced Flow
Len-Seq (7,5,5) Total:17



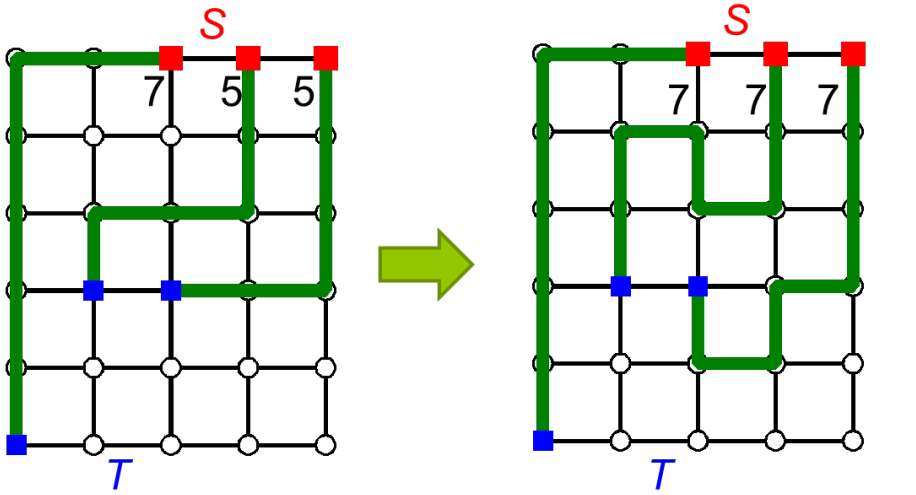
Total:271
Maximum: 30
Difference: 19

Max-Length is heuristically minimized

Set-Pair Routing (3)

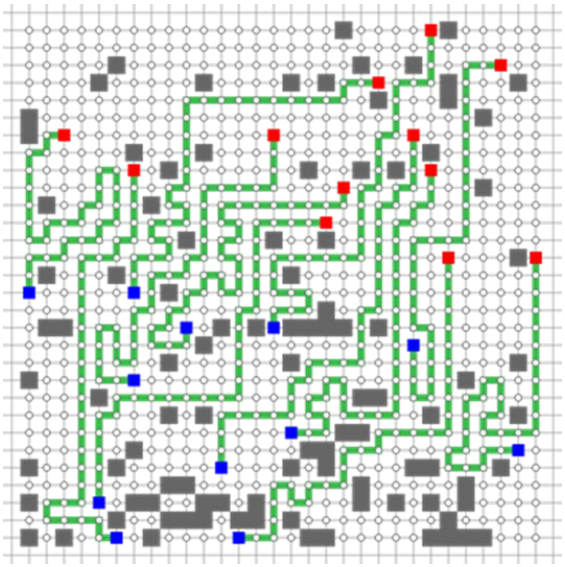
Step 3 (Minimum Length Increase)

- ✓ Lengthen a wire of the minimum length as much as possible by using R-Flip iteratively while the maximum length is kept



Min-Total Max-Reduced
Len-Seq (7,5,5) Total:17

Diff-Reduced (Min-Increased)
Len-Seq (7,7,7) Total:21

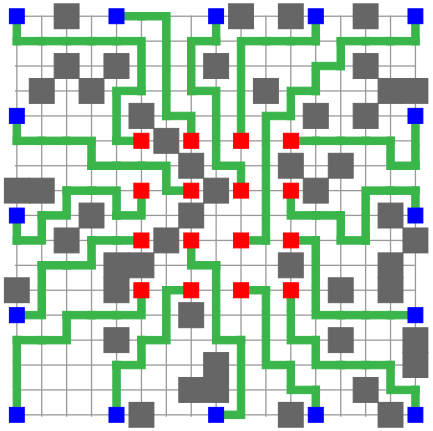


Total:355
Maximum: 30
Difference: 1

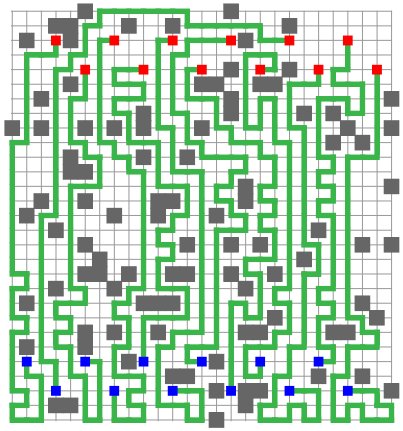
Min-Length is greedily increased

Increase of Total-Length is suppressed since Max-length is reduced in Step 2

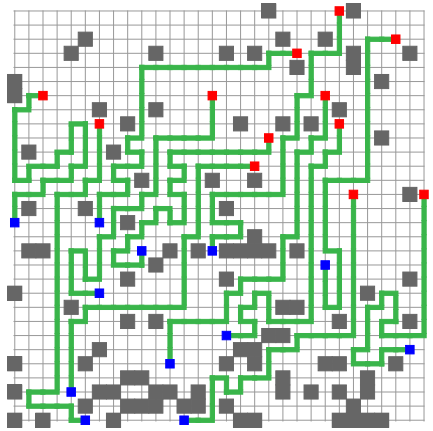
Set-Pair Routing Example



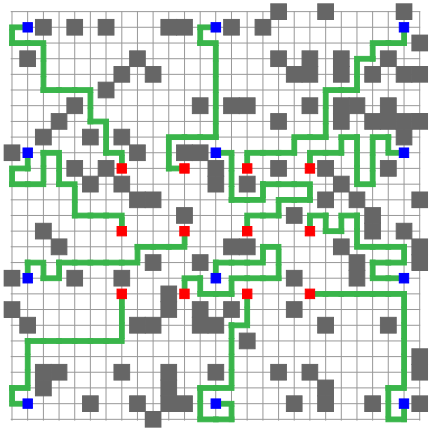
Escape-type
Size:19x19
#Net: 16
#Obst: 50
Min-Total: 146
Total: 154
Max: 16
Diff: 8



Line-type
Size:29x31
#Net: 12
#Obst: 100
Min-Total: 356
Total: 420
Max: 46
Diff: 12



Bisection-type
Size:30x30
#Net: 12
#Obst: 100
Min-Total: 271
Total: 355
Max: 30
Diff: 1



Flip-type
Size:29x29
#Net: 12
#Obst: 120
Min-Total: 132
Total: 196
Max: 19
Diff: 4

Summary

- Routing tools and algorithm for Digital LSIs tend to be matured

- Developments and improvements of application specific routing algorithms are still highly required in industry
  - Package
  - PCB
  - Design for Manufacture
  - New objectives