

Method for Rapid Recovery Path Computation on Mesh IP Network

30/09/2008

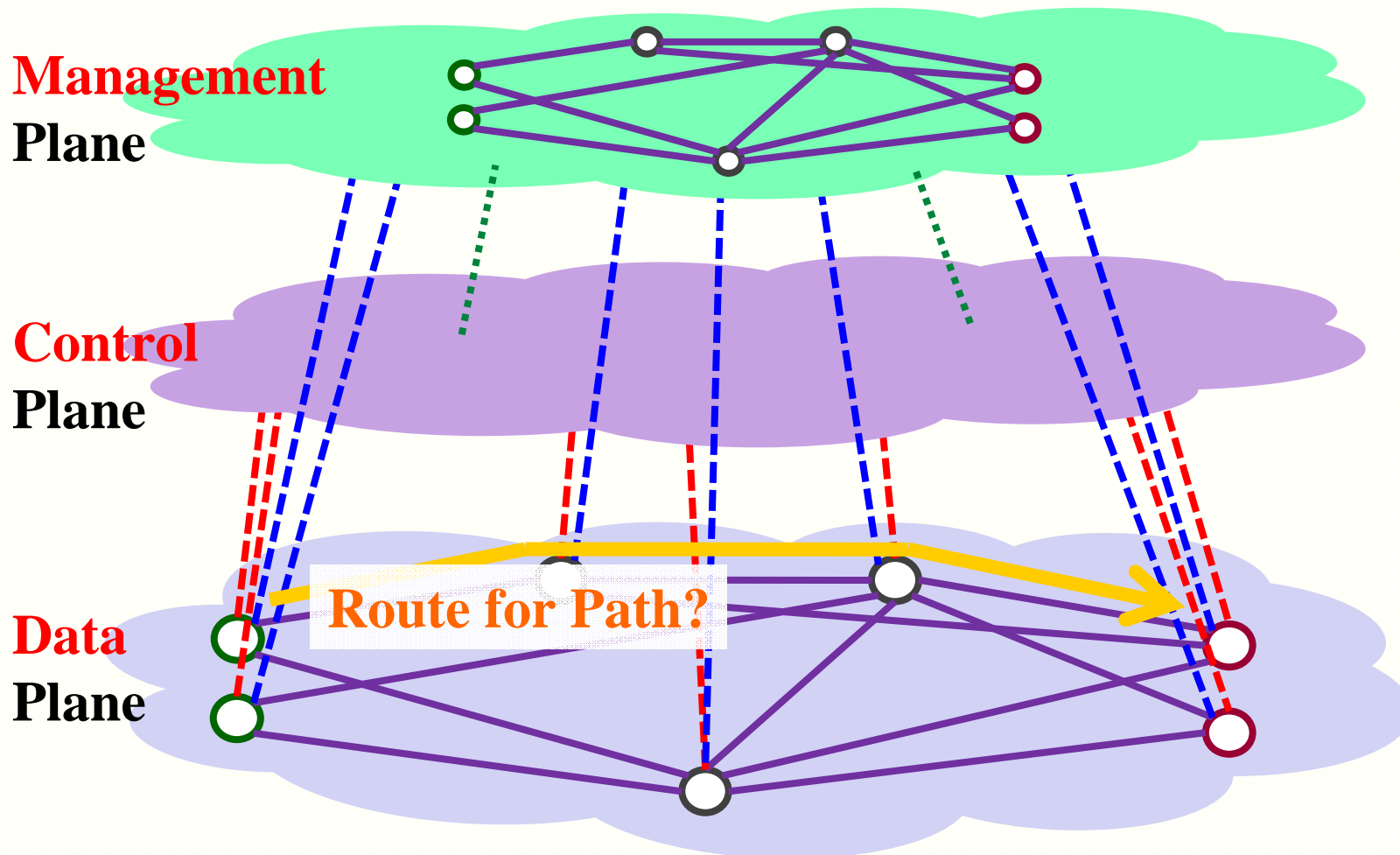
NTT Service Integration Laboratories

TSUJINO Masayuki

* This study was partly supported by the Ministry of Internal Affairs and Communications of Japan.

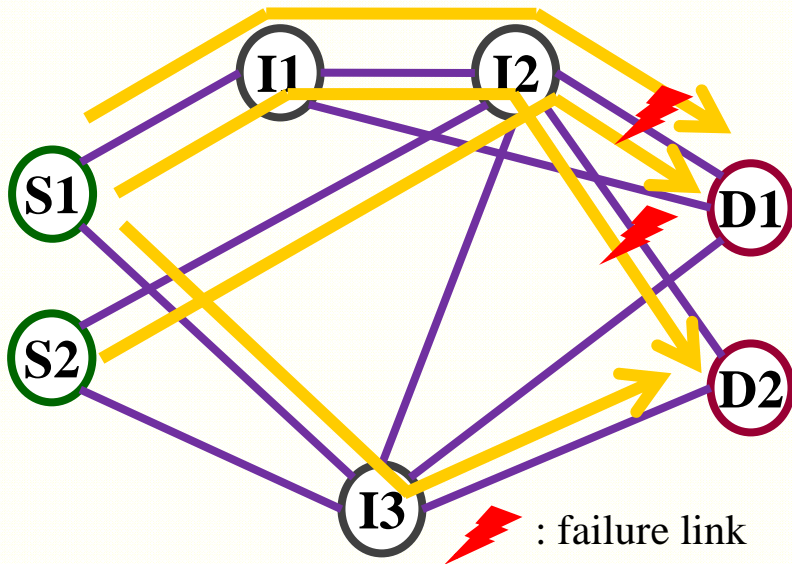
Centralized Management Architecture in MPLS

- Management plane obtains information about whole network conditions.
- Management plane establishes label switched paths (LSPs) and chooses LSP routes while considering obtained network conditions.

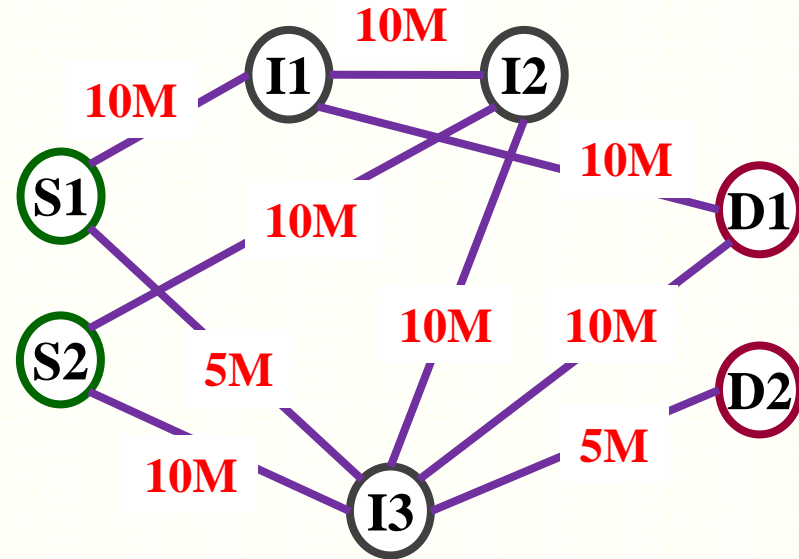


Path Computation under Restoration Scheme

Restoration Scheme: Computing recovery paths after breakdown



Residual Network



Note:

All link capacities are 100 Mbps
 Demands for all S-D pair: 50 Mbps

Paths for S1-D1, S1-D2, and S2-D1 are under failure.

Legend:

*M: Spare link capacity [Mbps]

Compute desirable path routes for S1-D1, S2-D2, and S2-D1 on residual network.

Recovery by Descending Order of Priority

- Differentiate demands depending on their priority level
 - e.g., mission critical VoIP traffic > best-effort IP traffic
- “Path-group” : a set of paths whose demands have the same priority level
- Establishment of a path-group with a higher priority level precedes that of a lower-priority path-group

Requirements for Path-Group Computation

- Scalability
 - Short time
 - 10 minutes at most for path-group with specific priority
 - Nationwide networks
 - about 50 nodes
 - Many demands
- Effectiveness
 - Effective use of link resources
 - Having sufficient available resources leads to accommodating as many paths at a lower priority level as possible.



Formulate Path-Group computation as a minimum cost problem
NP-hard problem → Enormous number of candidate paths

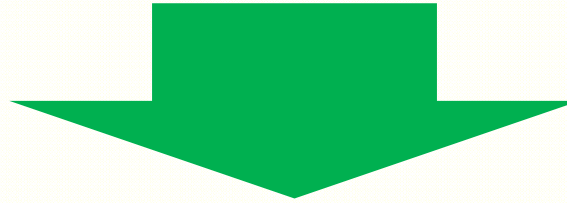
Alternatives

	Greedy Approach	Heuristic Approach
Contents	Compute minimum cost route for each path in sequence, where tentative link costs based on free link capacity are applied	Enable to improve the link resource usage
Scalability	High	Low
Effectiveness	Low	High

Trade off

Objectives

Evaluating the greedy and the heuristic algorithm using numerical simulation from viewpoints of computation time [Scalability] and optimality [Effectiveness]



Guideline for feasibility of path-group computation algorithm

1. Two alternate algorithms for computing route of path-group
2. Simulation result

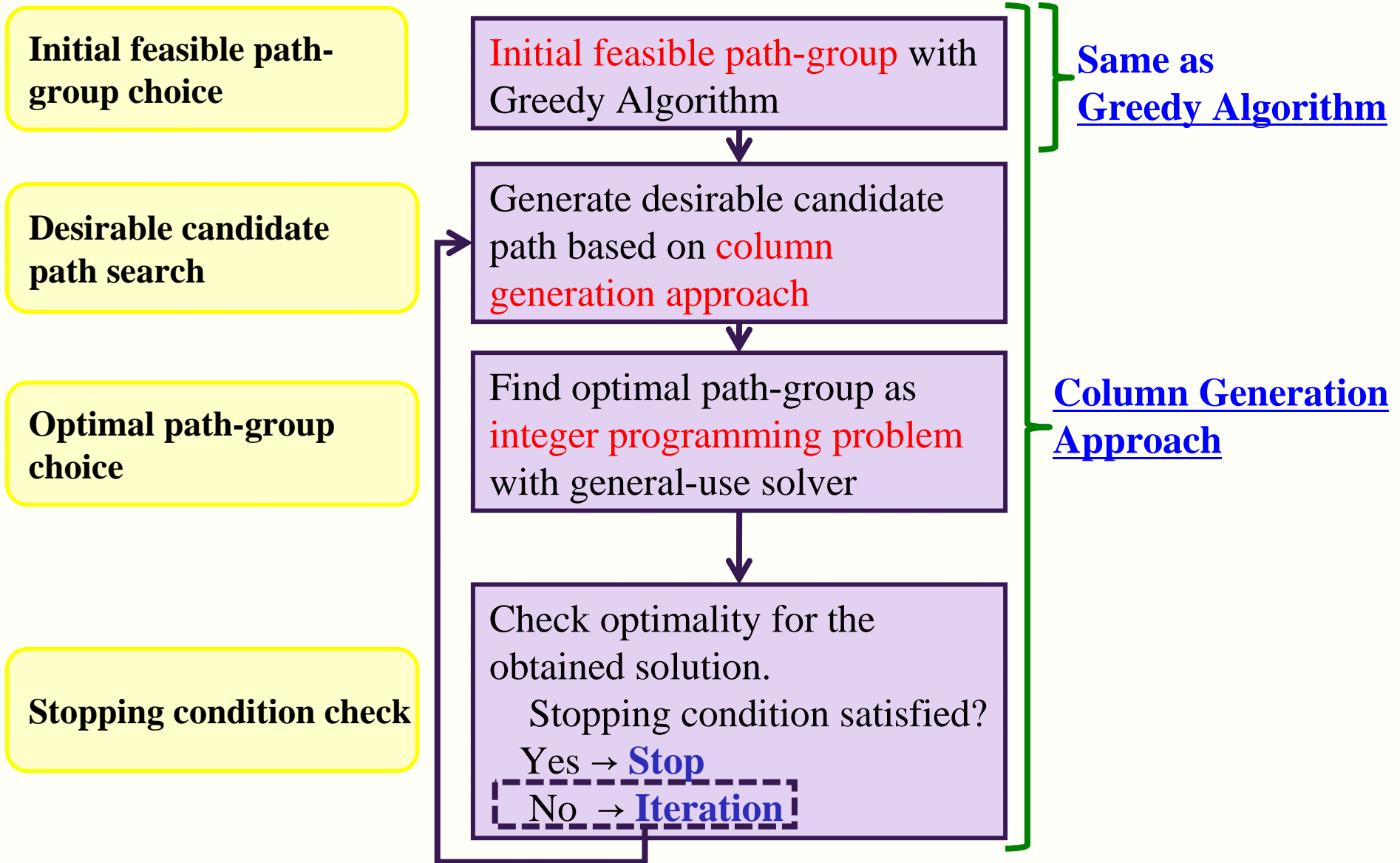
Greedy Algorithm

Step 1. Set tentative link costs to be inversely proportional to the free link capacity on the target network.

Step 2. Choose a demand to compute the path route.
Compute the minimum cost path route for the demand.

Step 3. If the path routes for all demands have been computed, stop. Otherwise, assign the path computed at **Step 2** to the target network, recalculate the free link capacity on the network, and return to **Step 1**.

Heuristic Algorithm



Merits of Heuristic Algorithm

Enables efficient searching of desirable candidate paths because dual variables, used in the column generation approach, provide good information.

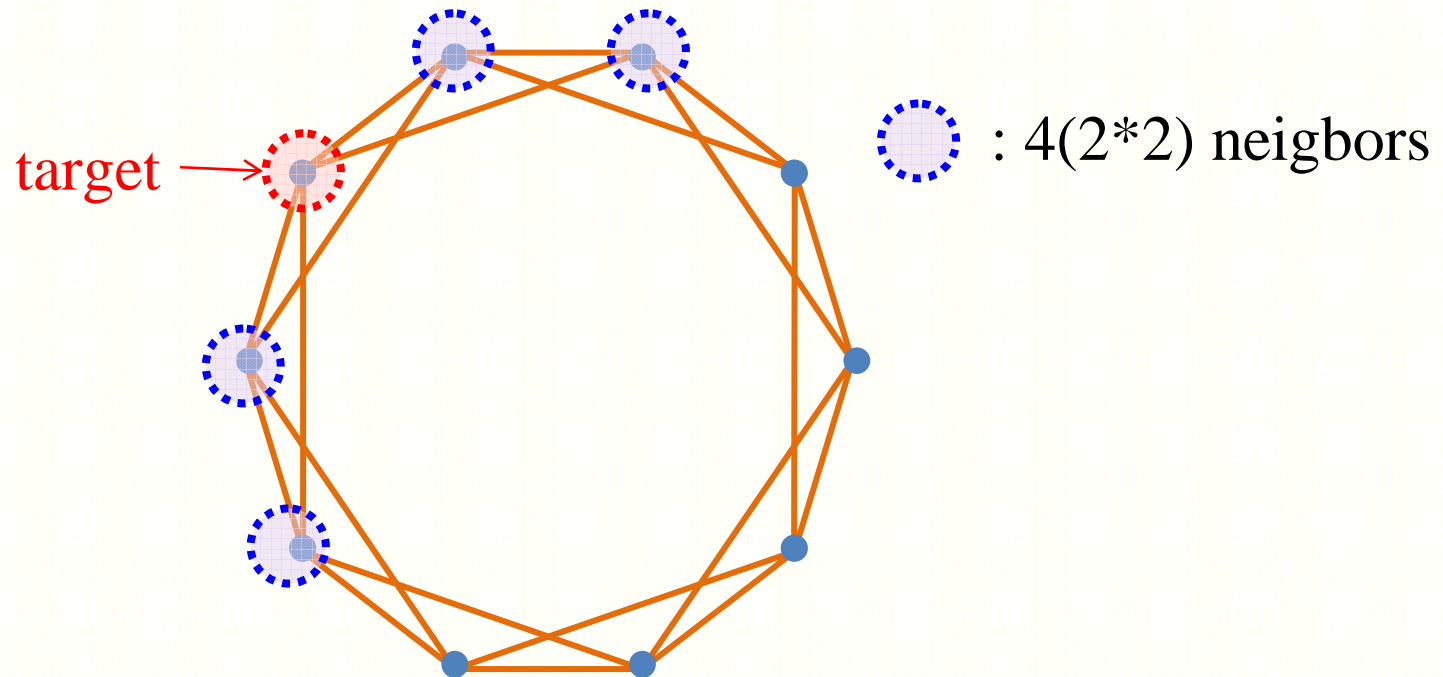
Formulates the process of choosing optimal path-group as an “relatively” small integer programming problem.

Enables getting of lower bound at each iteration by solving dual problem for partial enumerated path problem.

Network for Simulation (1) – Ring Lattice

Ring Lattice model, $RL(n,m)$:

A network that has n nodes, each of which is connected to $2 * m$ neighbors, m on each side.



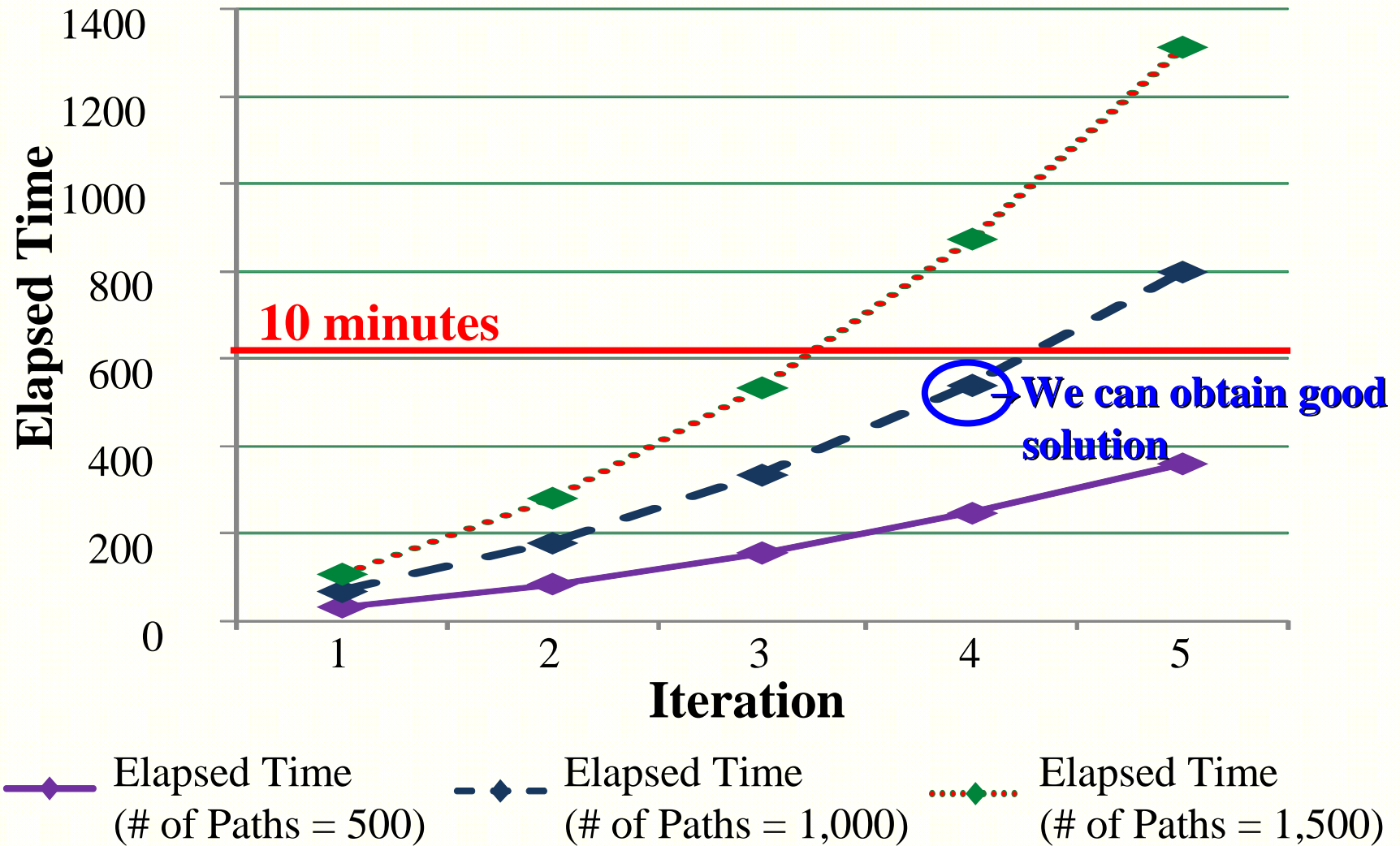
Example : $RL(10, 2)$

Test for Computation Time – Simulation Condition

- We applied the heuristic algorithm to $RL(50, 2)$ to calculate the computational capability of the heuristic algorithm on IP core networks in Japan.
- We evaluated the computation time for computing the path-group with various numbers of paths (numbers [#] of paths = 500, 1,000, and 1,500) on the network.
- CPU: Intel Core2 CPU 6600 2.40 GHz
- Software for solving optimization problems: NUOPT*

*<http://www.insightful.com/products/nuopt/default.asp>

Test for Computation Time – Result

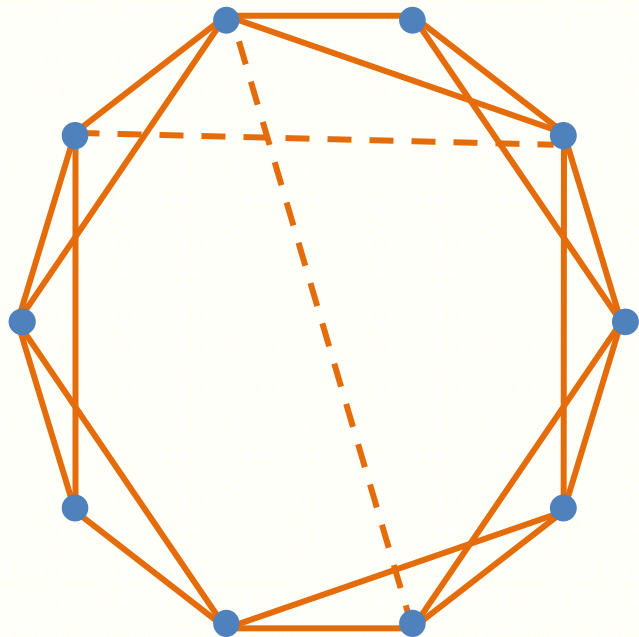


Network for Simulation (2) – Watts & Strogatz

Watts and Strogatz model, $WS(n, m, p)$:

A network that is obtained from $RL(n, m)$ by rewiring every link with probability p .

We set p to 0.1 because $WS(n, m, p)$ belongs to “the small world.”



Dashed lines indicate links that are chosen with probability 0.1 for rewiring

Example: $WS(10, 2, 0.1)$

Test for Optimality – Simulation Condition

- Network conditions
 - Number of nodes: $n = 10, 15, 20, 25,$ and 30
 - Number of connected neighbors: $m = 2$
 - Number of demands: 90 (for $n = 10$), 210 (for $n = 15$), 380 (for $n = 20$), 600 (for $n = 25$), and 870 (for $n = 30$).
 - Demands were set to be between every 2 nodes.
 - Volume for each demand: Values were given from the uniform distribution at $[0,2]$.
 - Link capacity: Values were given from the uniform distribution at $[0.5 * CAP, 1.5 * CAP]$.
 - $CAP = 25$ (for $n = 10$), 46 (for $n = 15$), 71 (for $n = 20$), 99 (for $n = 25$), and 130 (for $n = 30$).
 - Cost per unit resource: 1.0 for every link
- Evaluated values
 - **$(Cost-Ratio) = (SGA - SHA) / SHA$,**
 - SGA : Solution obtained by the Greedy Algorithm
 - SHA : Solution obtained by the Heuristic Algorithm
- Ten-case simulation for each network condition.

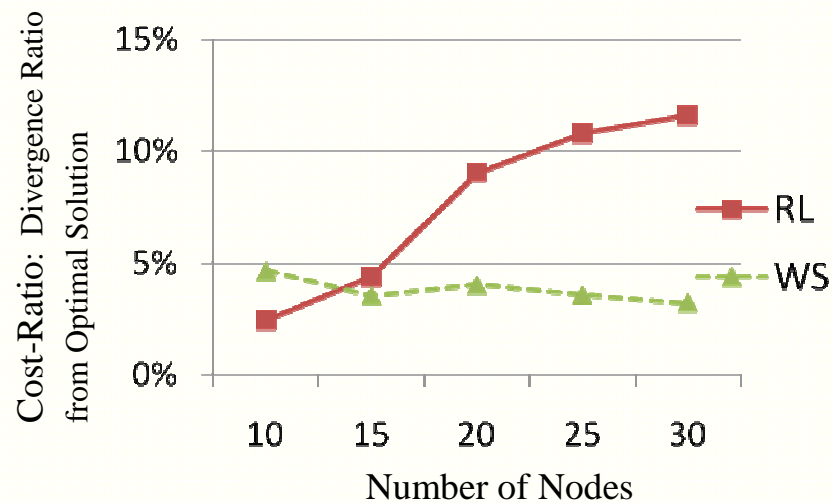
Test for Optimality – Result

Cost-Ratio

$$= (\text{SGA} - \text{SHA}) / \text{SHA}$$

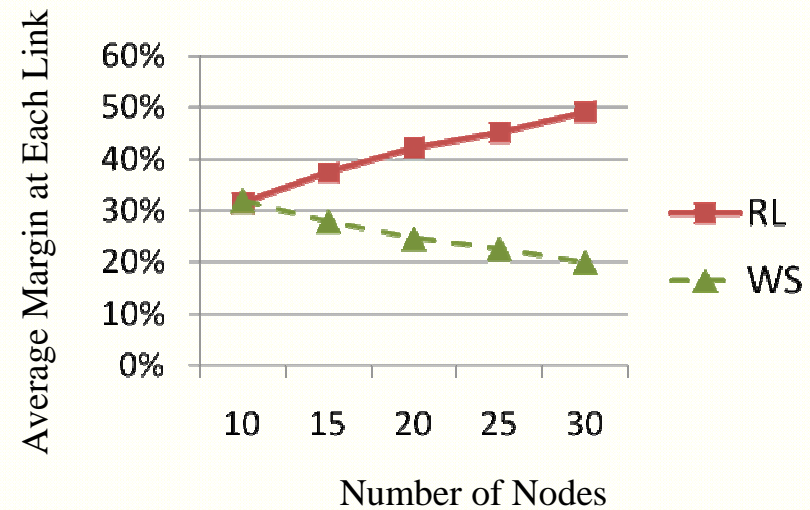
SGA : Solution for Greedy Algorithm

SHA : Solution for Heuristic Algorithm



Average Margin

$$= \text{AVE}\{(\text{Unused Resources}) / (\text{Link Capacity}), \text{ for Every Link}\}$$



Small margin at each link: The heuristic algorithm significantly improved the solution obtained by the greedy algorithm.

Conclusion

- **Computation Time**
 - We can apply the heuristic algorithm to a network of practical size.
- **Effectiveness**
 - The heuristic algorithm obtained a significantly improved path-group for networks with plenty of margin.
 - However, the heuristic algorithm did not work efficiently for networks with a small margin.
- **Current Guideline**
 - First use the greedy algorithm for premium path-groups, and then use the heuristic algorithm for other path-groups.