

Approximating optimal spare capacity allocation by successive survivable routing

Yu Liu

Joint work with *David Tipper* and *Peerapon Siripongwutikorn*

Dept. of Information Science and Telecommunications

University of Pittsburgh



Thanks for travel grants from *IEEE ComSoc* and *Corporate patrons*

To be presented at INFOCOM 2001– *Capacity Allocation*

Wednesday April 25, 2001, 10:30am – 12:00pm



1. Introduction

- Growing interests in network survivability
- Survivable network techniques
 - Failure prevention
 - Topology design
 - Spare capacity allocation for a given network management/restoration procedure
 - *Goal:* maintain service for certain failures at reasonable cost



Spare capacity allocation problem

- Given network topology, traffic demand, working paths
- Find backup paths and spare capacities
- Minimize total spare capacity

- Two multi-commodity flow models
 - Path-flow model
 - Arc-flow model



Challenging problems

- **SCA was** known as an NP-hard problem
- Many approximation algorithms exist
 - Most of them are centralized algorithms, slow
 - Recent distributed algorithms are not near optimal
 - Some of them cannot scale to larger networks
- Fast approximation algorithms for near optimal solution are needed



2. A two-pronged attack

A. Matrix-based method

- Simplify the SCA structure
- Aggregate per-flow information

B. Approximation algo. – successive survivable routing

- Partitioning multi-commodity to single flow
- Using shortest path algorithm with special link cost
- Updating backup paths iteratively

Considering following case

- Use path restoration with disjoint backup routes
- Guarantee 100% protection for all single link failures



A. Matrix-based presentation

- Given path link incident matrices \mathbf{A} and \mathbf{B} for working and backup paths, diagonal matrix \mathbf{M} with elements for bandwidth request of flows, find spare provision matrix \mathbf{C} , and spare capacity reservation \mathbf{s}
 - $\mathbf{C} = \mathbf{B}^T \mathbf{M} \mathbf{A}$, element C_{ij} gives required spare capacity on link i when link j fails
 - or $\mathbf{C} = \sum_r \mathbf{C}_r$, where $\mathbf{C}_r = \mathbf{b}_r^T \mathbf{a}_r$, \mathbf{a}_r and \mathbf{b}_r are vectors for working and backup paths of flow r
 - $\mathbf{s} = \text{rowmax}(\mathbf{C})$, or $\mathbf{s} \geq \mathbf{C}$, since spare capacity reservations are the maximum one among failures to guarantee protections for any single link failures



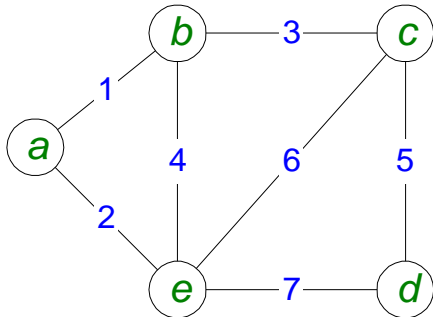
Example

From B^T and A , get C

From C , get s

For $r=2$, find

$$C_2 = b_r^T a_r$$



Link i		1	2	3	4	5	6	7	Total working	Spare path and link incident matrix B^T																					
	$w_i = \sum A^{*,i}$	2	2	3	1	2	1	2	13																						
	$s = \text{rowmax}(C)$	$C = B^T \cdot A$																													
1	2	0	2	1	1	1	0	1		0	0	1	1	0	1	1	0	0	0	0											
2	2	2	0	2	1	1	0	0		1	1	0	0	0	1	1	0	0	0	0											
3	1	0	1	0	0	0	1	1		0	0	1	0	0	0	0	0	1	0	0											
4	1	1	1	1	0	0	1	0		1	0	0	1	1	0	0	0	1	0	0											
5	2	1	1	1	0	0	0	2		0	1	1	0	0	0	0	0	0	0	1											
6	1	0	0	1	0	1	0	1		0	0	0	0	1	0	0	1	0	1	0											
7	2	1	0	2	0	2	0	0		0	1	0	0	0	1	0	1	0	0	0											
Total spare	11																														
									Flows																						
									src	dst	m	1	2	3	4	5	6	7	8	9	10										
									1	a	b	1																			
									2	a	c	1																			
									3	a	d	1																			
									4	a	e	1																			
									5	b	c	1																			
									6	b	d	1																			
									7	b	e	1																			
									8	c	d	1																			
									9	c	e	1																			
									10	d	e	1																			
Working path and link incident matrix A																															
									1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
									2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
									3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
									4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
									5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
									6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
									7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
									8	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									10	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Arc-flow model

$$\min_{\mathbf{B}=\{b_{rj}\}} S = \mathbf{e}^T \mathbf{s}$$

Total spare capacity

$$\mathbf{B}=\{b_{rj}\}$$

Enough spare capacity on each link

$$\text{s.t. } \mathbf{s} \geq \mathbf{C}_{*j}, 1 \leq j \leq L$$

$$\mathbf{C} = \mathbf{B}^T \mathbf{M} \mathbf{A}$$

Calculation of spare provision matrix

$$\mathbf{A} + \mathbf{B} \leq \mathbf{1}$$

Link-disjointed backup paths

$$\mathbf{B} \mathbf{R}^T = \mathbf{D} \pmod{2}$$

Flow conservative for backup

\mathbf{B} is a binary matrix

Integer programming



Arc-flow model: explanations

- $\mathbf{R}=\{r_{jn}\}$ is the binary node-link incidence matrix
 - $r_{jn} = 1$ when node n is origin/destination node of link j , ,
otherwise $r_{jn} = 0$
- $\mathbf{D}=\{d_{rn}\}$ is the binary demand node-pair matrix
 - $d_{rn} = 1$ when node n is origin/destination node of flow r ,
otherwise $d_{rn} = 0$
- Flow conservative constraints might introduce loops
 - Do not influence the optimal solution
 - Trivial to find an eligible route by removing these loops



Benefits

- $O(L^2)$ state information – spare provision matrix \mathbf{C}
- Hide per-flow based information
 - Better scalability
 - More privacy and security
- SCA is NP-complete [proved in another paper]
- Hints to find an approximate algorithm

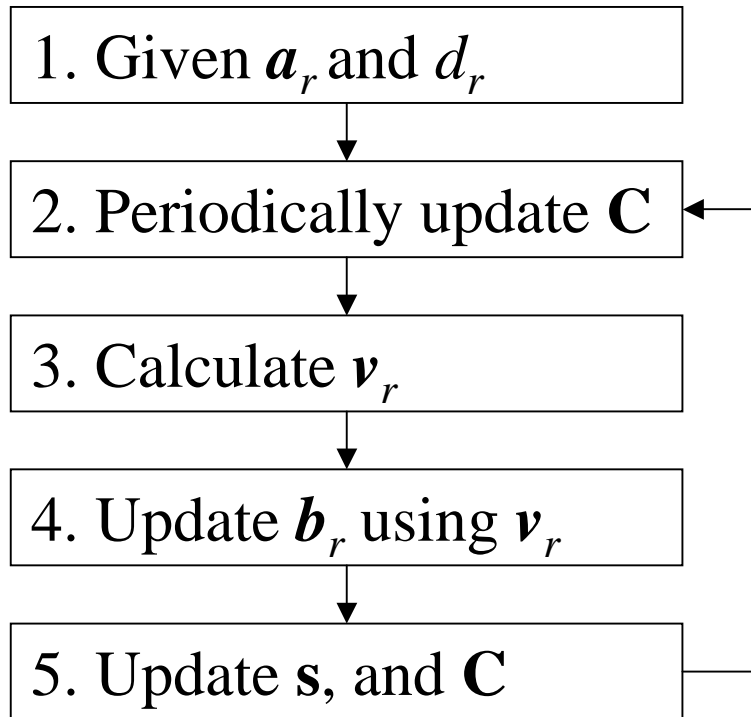


B. An approximation algorithm

- Decompose multi-commodity flow to smaller single-flow problems
 - Using shortest path algorithm for each flow to
 - route backup paths, link-disjointed from their working paths;
 - using spare provision matrix \mathbf{C} to calculate
 - **link cost** – incremental spare reservations \mathbf{v}_r ;
 - increase spare capacity reservation \mathbf{s} when necessary
 - Source nodes **successively** update the backup paths whenever \mathbf{C} changes
- successive survivable routing (SSR)



SSR flowchart



- On source node of flow r :
 - a_r, b_r : working and backup path vectors
 - d_r : destination node
 - C, s : spare provision matrix and spare reservation vector
 - v_r : incremental spare reservations as link cost
- Stop after no backup path update on the network



Link cost and local objective

- Link cost calculation
 - Let $\mathbf{C}^+ = (\mathbf{e} - \mathbf{a}_r)^T \mathbf{a}_r$, and $\mathbf{s}^+ = \text{rowmax}(\mathbf{C}^+ + \mathbf{C})$
 - $(\mathbf{e} - \mathbf{a}_r)$ assumes that a backup path uses all possible links
 - \mathbf{s}^+ is a temporary spare capacity reservation vector
 - Incremental spare reservation $\mathbf{v}_r = \mathbf{s}^+ - \mathbf{s}$
 - \mathbf{v}_r tells how much additional spare capacities are needed if a link is used on the new backup path
- *Goal*: Each flow seeks a new backup path with minimal total additional spare capacity reservation



Complexity

- Polynomial running time
 - shortest path algorithm for each flow, $O(N^2)$
 - Limited backup path update iterations for each flow
- Polynomial space complexity
 - Advertised information in $O(L^2)$
 - No **per-flow** based information



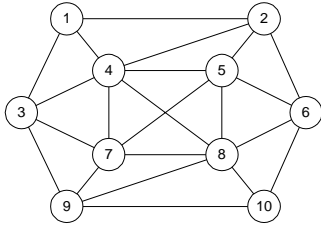
4. Numerical comparison

- Compare different algorithms and bounds
 - **RAFT**: Resource aggregation fault tolerance
 - **SPI**: Sharing with partial information
 - **SR**: Survivable routing (without any backup path update)
 - **SSR** : Successive survivable routing
 - **SA**: Simulated annealing
 - **BB**: Branch and bound on a path-flow model – optimal
 - **LP**: Linear programming – infeasible lower bound

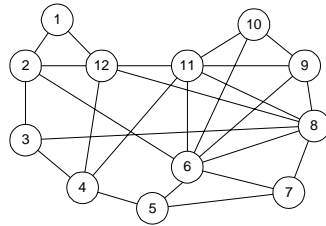


Experiment networks

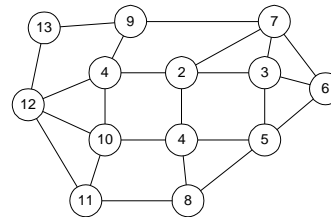
1



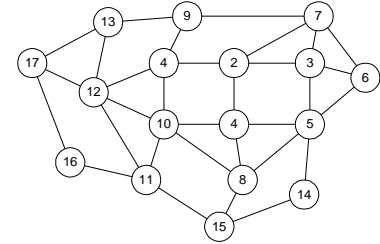
2



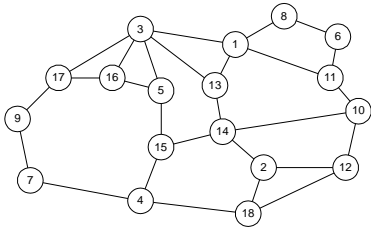
3



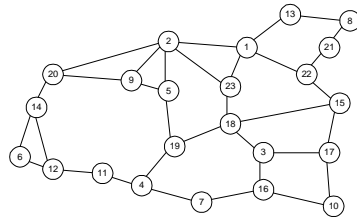
4



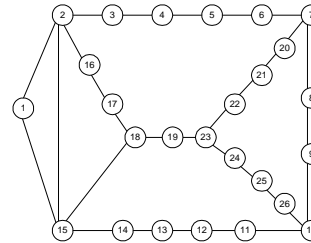
5



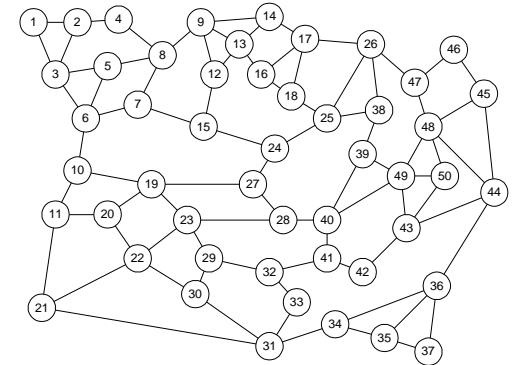
6



7



8





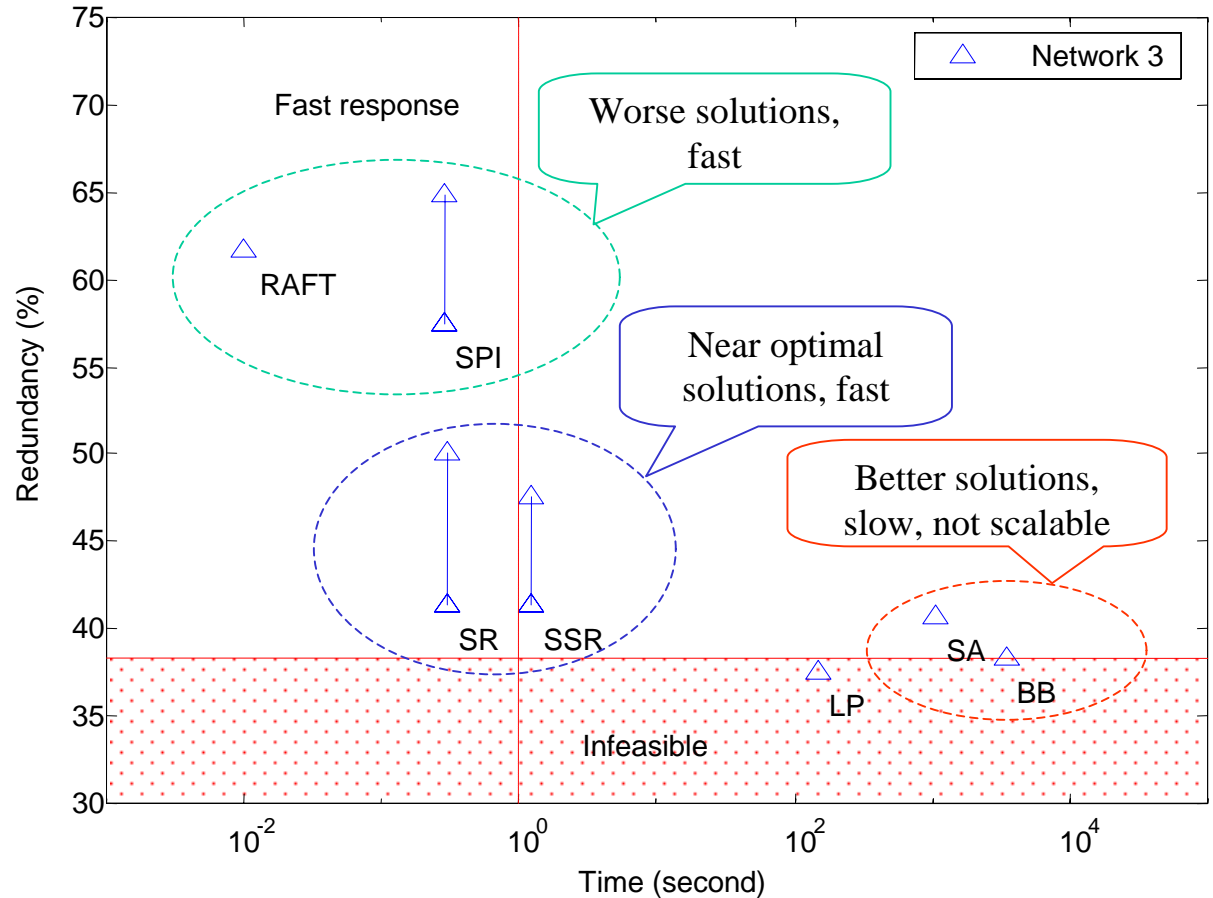
Redundancy versus Time on network 3

Redundancy is the ratio of the total spare and work reservations

$$\eta = S/W$$

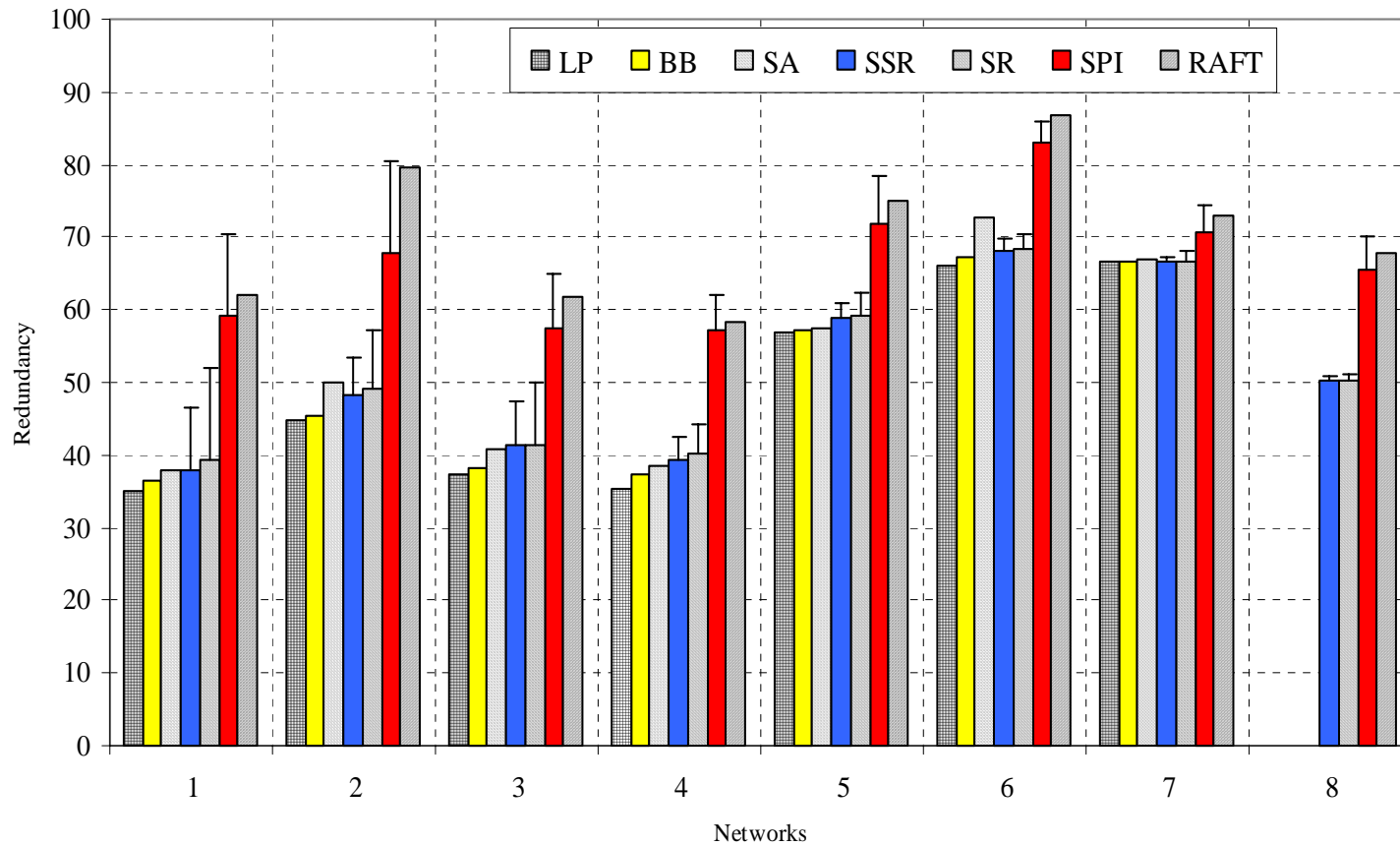
SSR, SR, SPI

have 64 random cases with different flow sequences





Typical SSR results





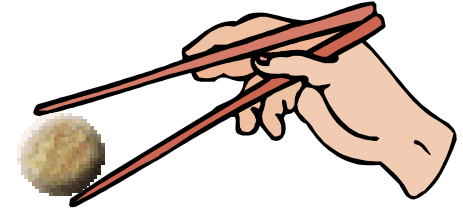
Conclusions

- Redundancy
 - $LP \leq BB \leq SA \leq SSR \leq SR \ll SPI \cong RAFT$
- Time
 - $BB > (LP, SA) \gg SSR \geq SR \geq SPI \geq RAFT$
- SSR has best trade-off
 - Near optimal
 - Fast



5. Summary

- Two-pronged attack
 - A. Matrix-based method
 - Simplify the SCA structure
 - Aggregate per-flow information into $O(L^2)$
 - B. Successive survivable routing algorithm
 - Partitioning multi-commodity to single flow
 - Using shortest path algorithm with special link cost
 - Updating backup paths iteratively
 - Fast computation find near optimal solution
- US Patent pending





Roadmap and Questions?

<http://www2.sis.pitt.edu/~yliu/>



Matrix and SSR

Multi-layer topology design

SCA is NP-complete

Nonlinear link cost

Node failures

Failure dependent path restoration

Capacitated network

Multi-layer spare capacity allocation