Biased random-key genetic algorithms with applications to optimization problems in telecommunications

Tutorial given at the Spring School in Advances in Operations Research, Higher School of Economics
Nizhny Novgorod, Russia ✤ May 3, 2011

Mauricio G. C. Resende
AT&T Labs Research
Florham Park, New Jersey

mgcr@research.att.com

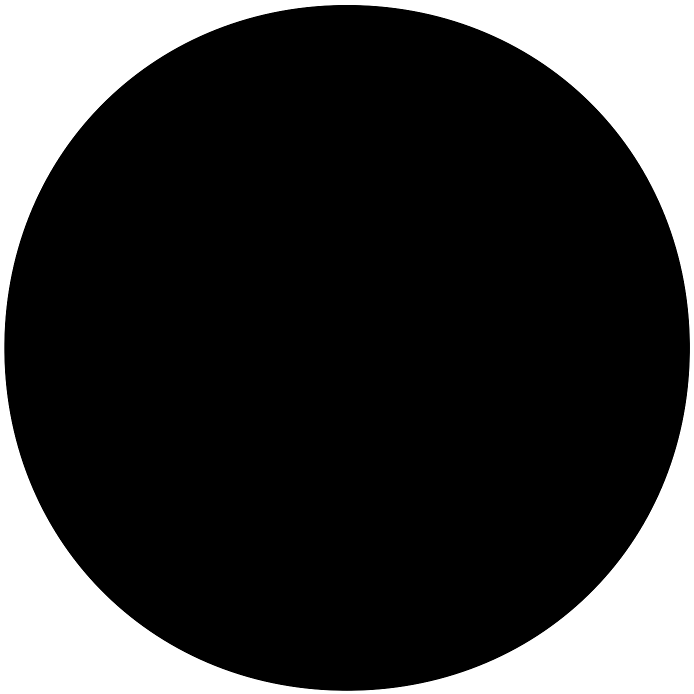
Summary

- Biased random-key genetic algorithms
- Applications in telecommunications
  - Routing in IP networks
  - Designing survivable IP networks with composite links
  - Three-layer metropolitan network design
  - Redundant server location for content distribution
  - Routing & wavelength assignment in optical networks
- Concluding remarks

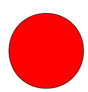
Biased random-key genetic algorithms

Genetic algorithms

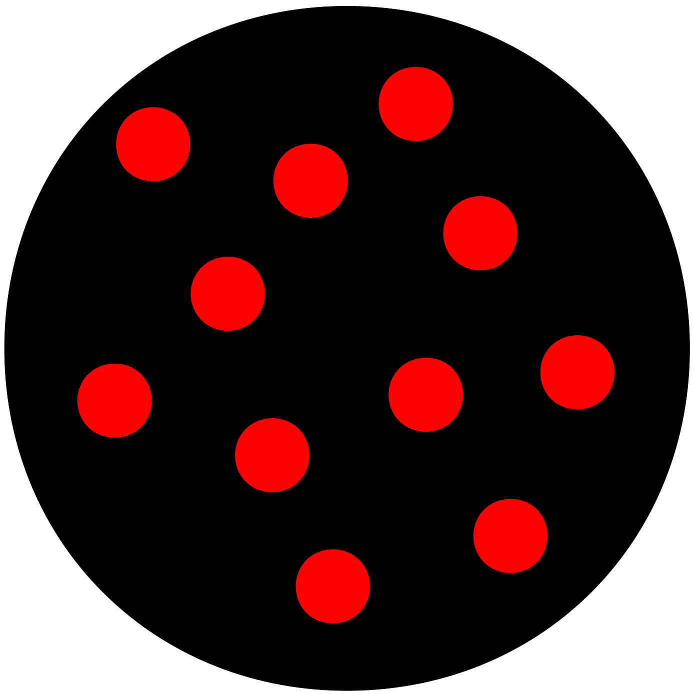
Holland (1975)



Adaptive methods that are used to solve search and optimization problems.

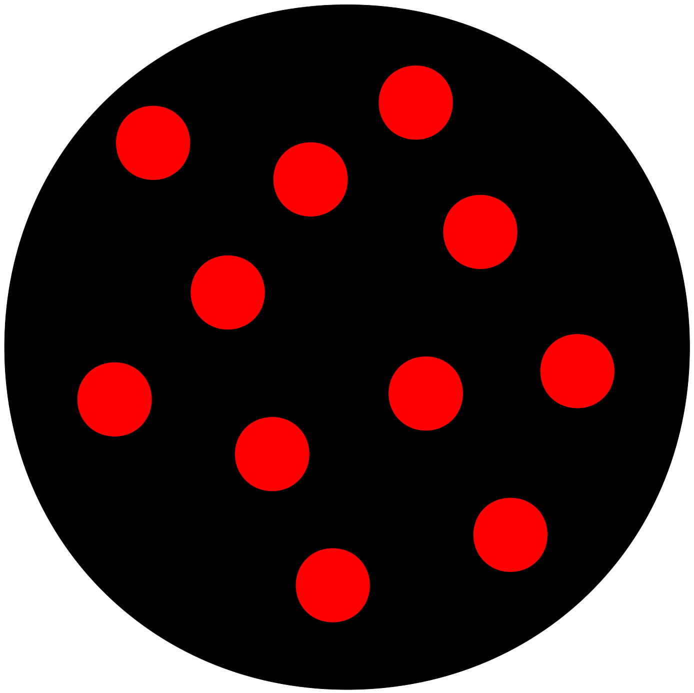
Individual: solution 

Genetic algorithms



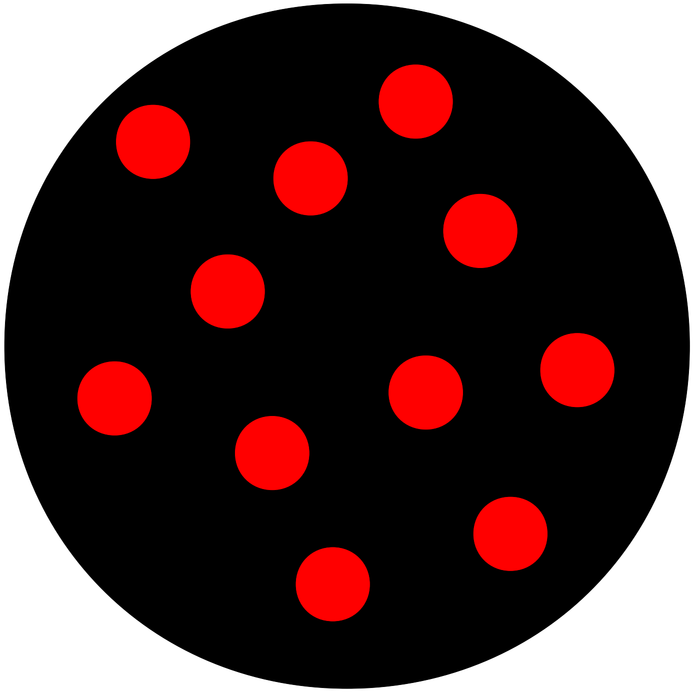
Individual: solution
Population: set of fixed number of individuals

Genetic algorithms



Genetic algorithms evolve population applying the principle of survival of the fittest.

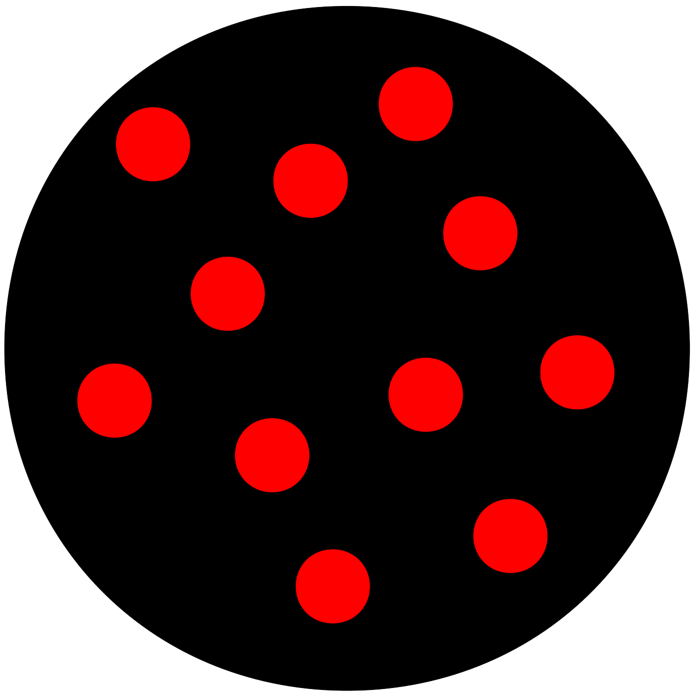
Genetic algorithms



Genetic algorithms evolve population applying the principle of survival of the fittest.

A series of generations are produced by the algorithm. The most fit individual of last generation is the solution.

Genetic algorithms

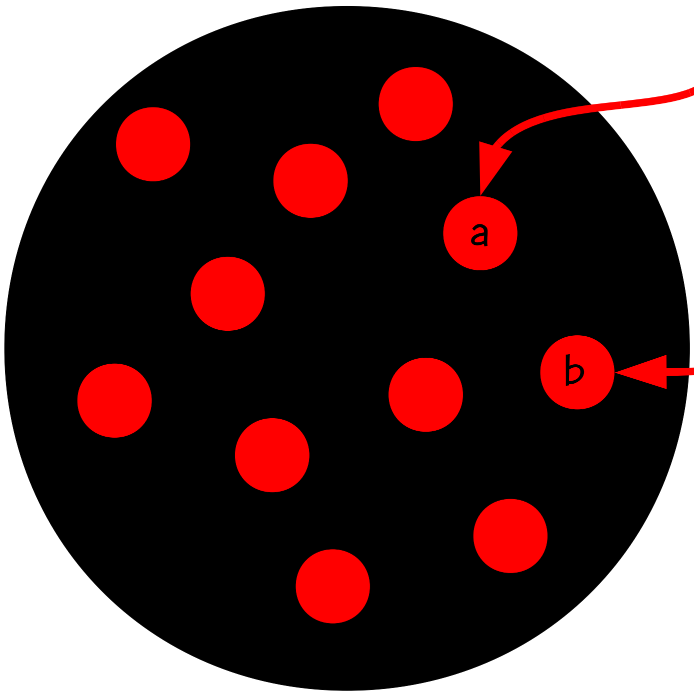


Genetic algorithms evolve population applying the principle of survival of the fittest.

A series of generations are produced by the algorithm. The most fit individual of last generation is the solution.

Individuals from one generation are combined to produce offspring that make up next generation.

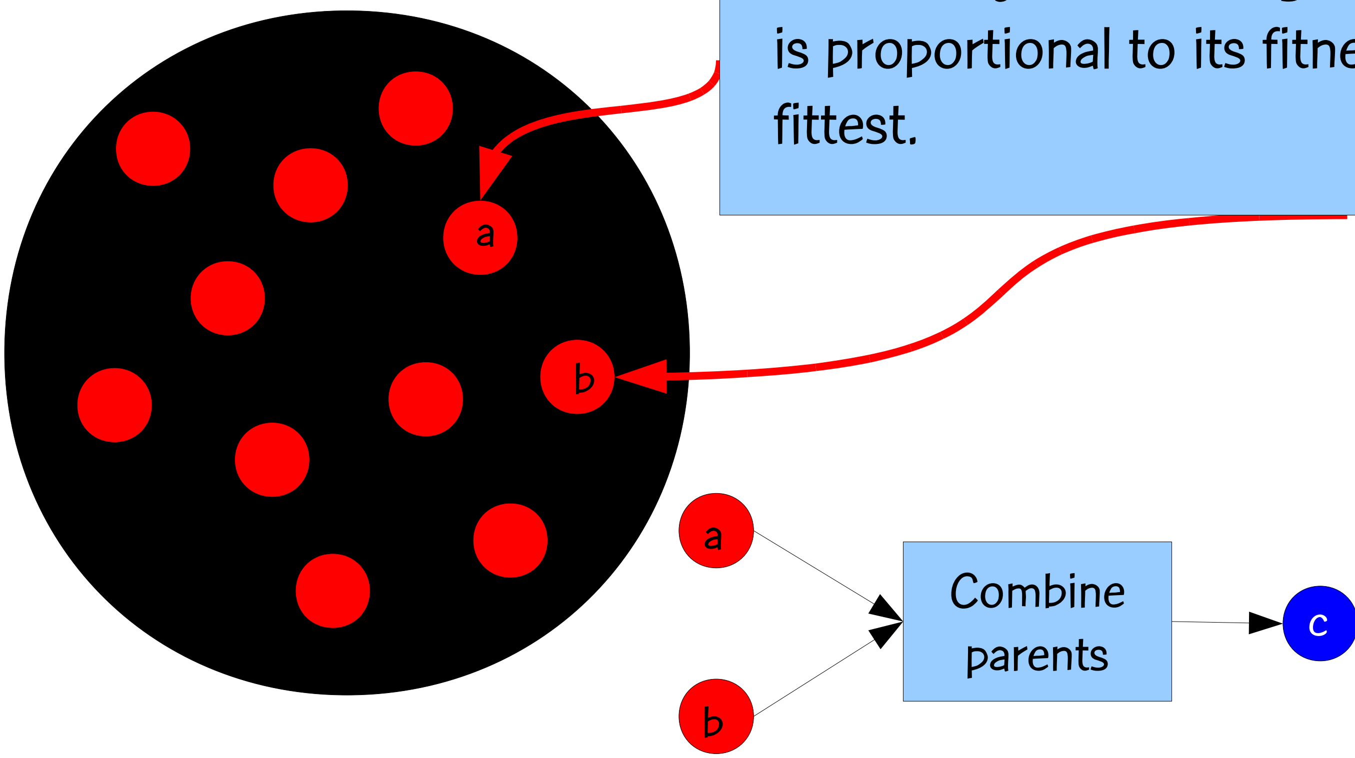
Genetic algorithms



Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.

Genetic algorithms

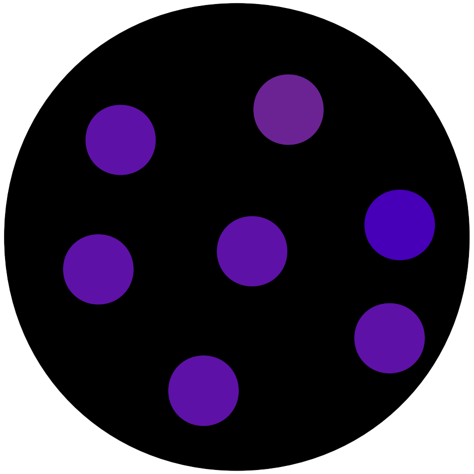
Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.



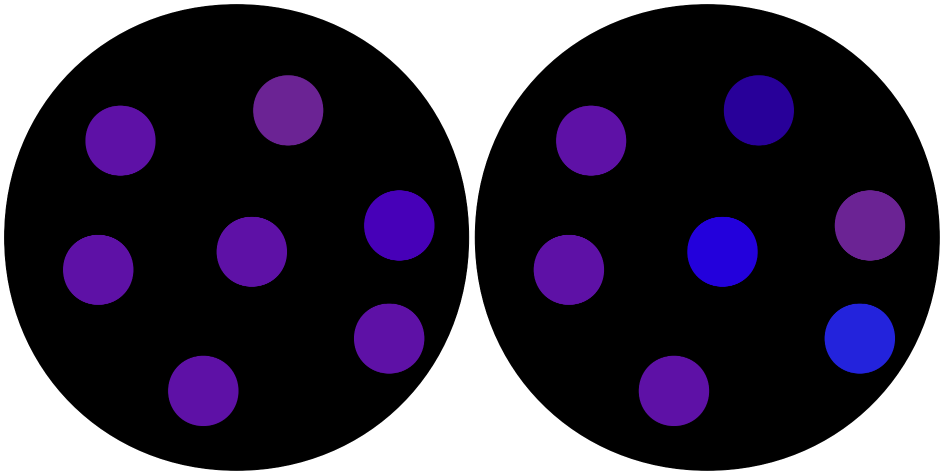
Child in generation K+1

Parents drawn from generation K

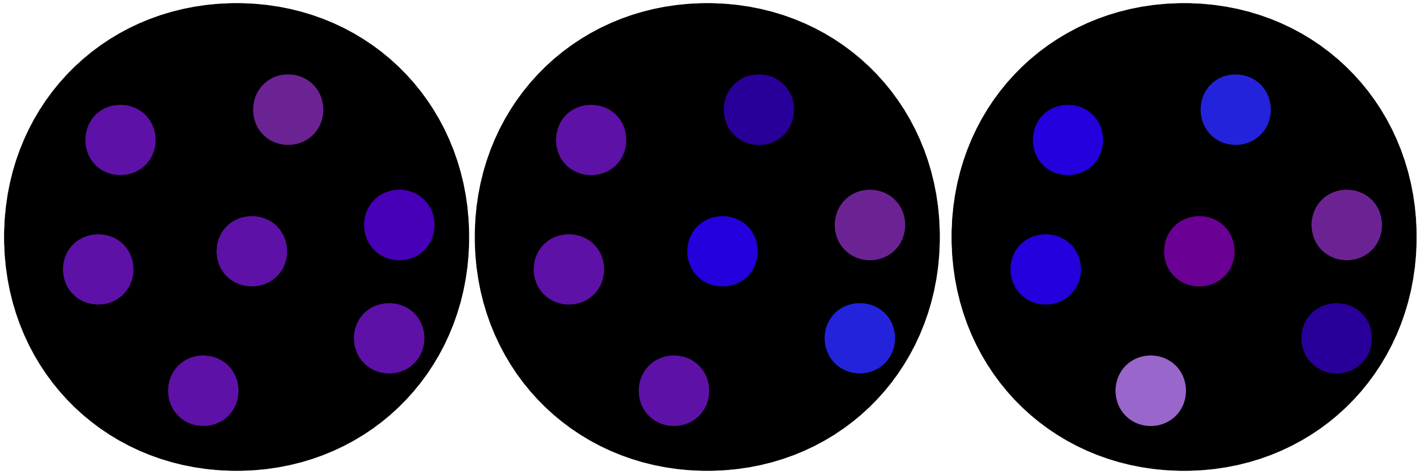
Evolution of solutions



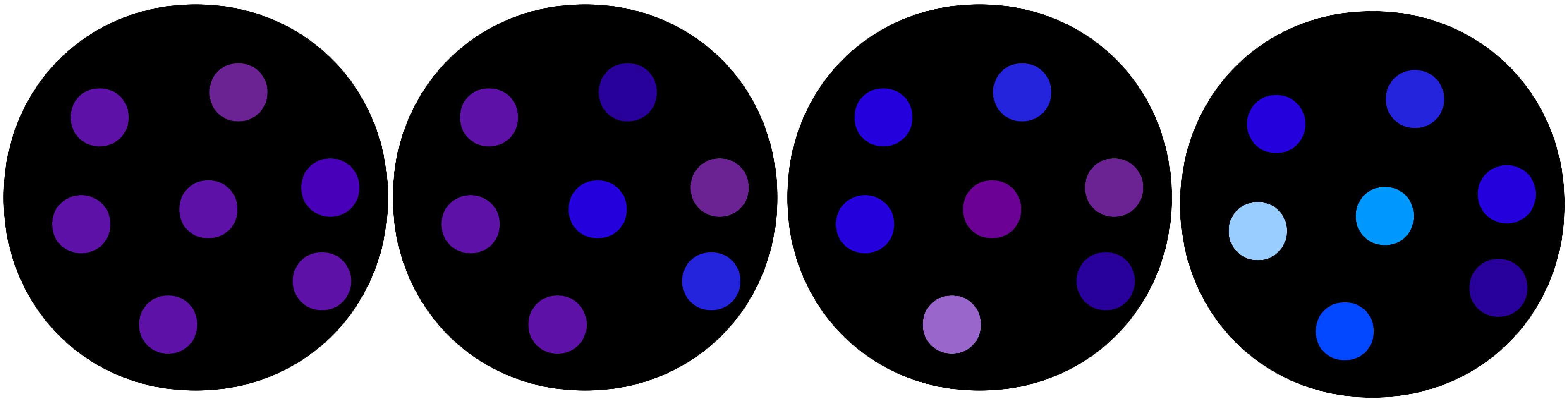
Evolution of solutions



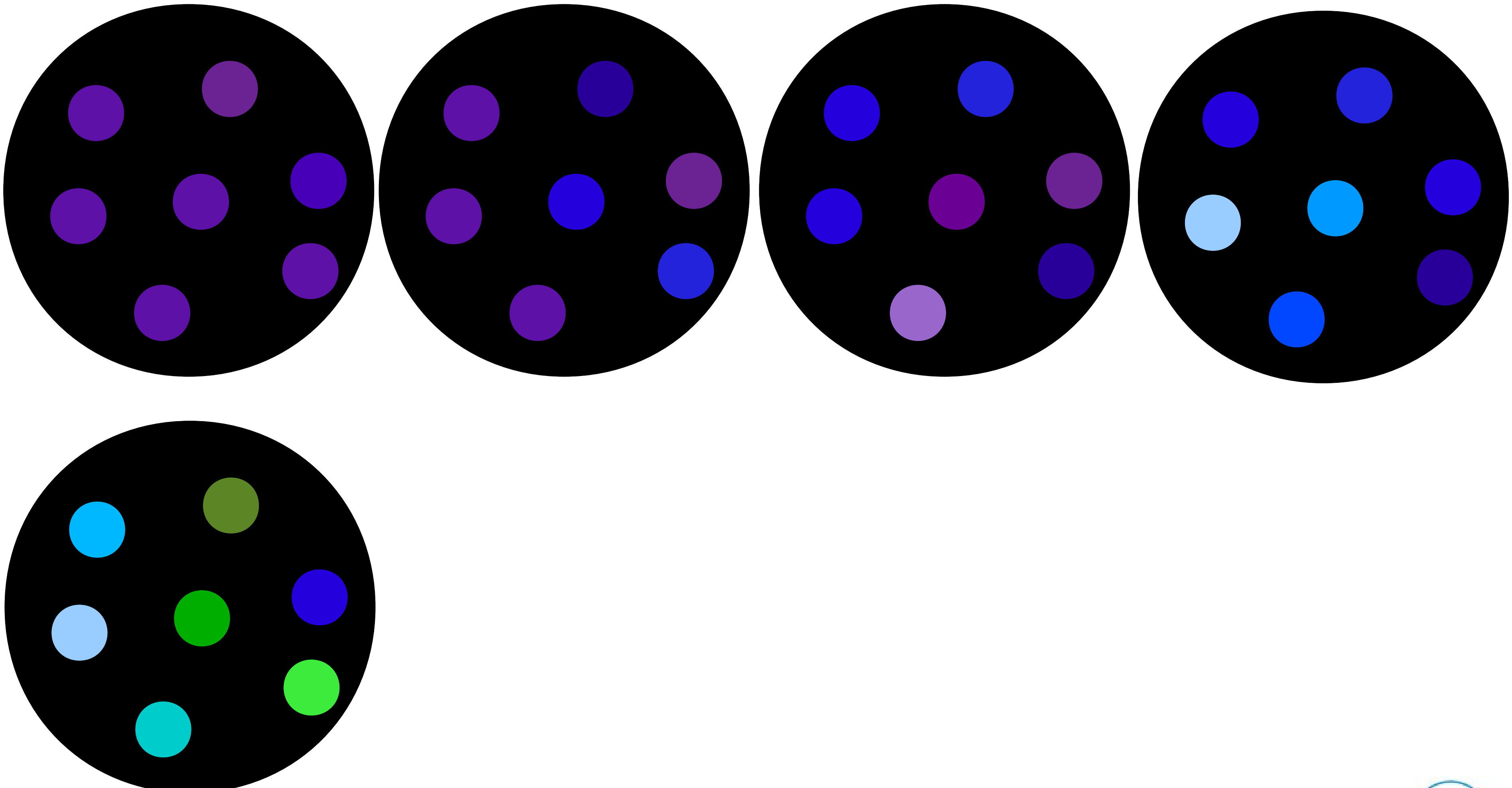
Evolution of solutions



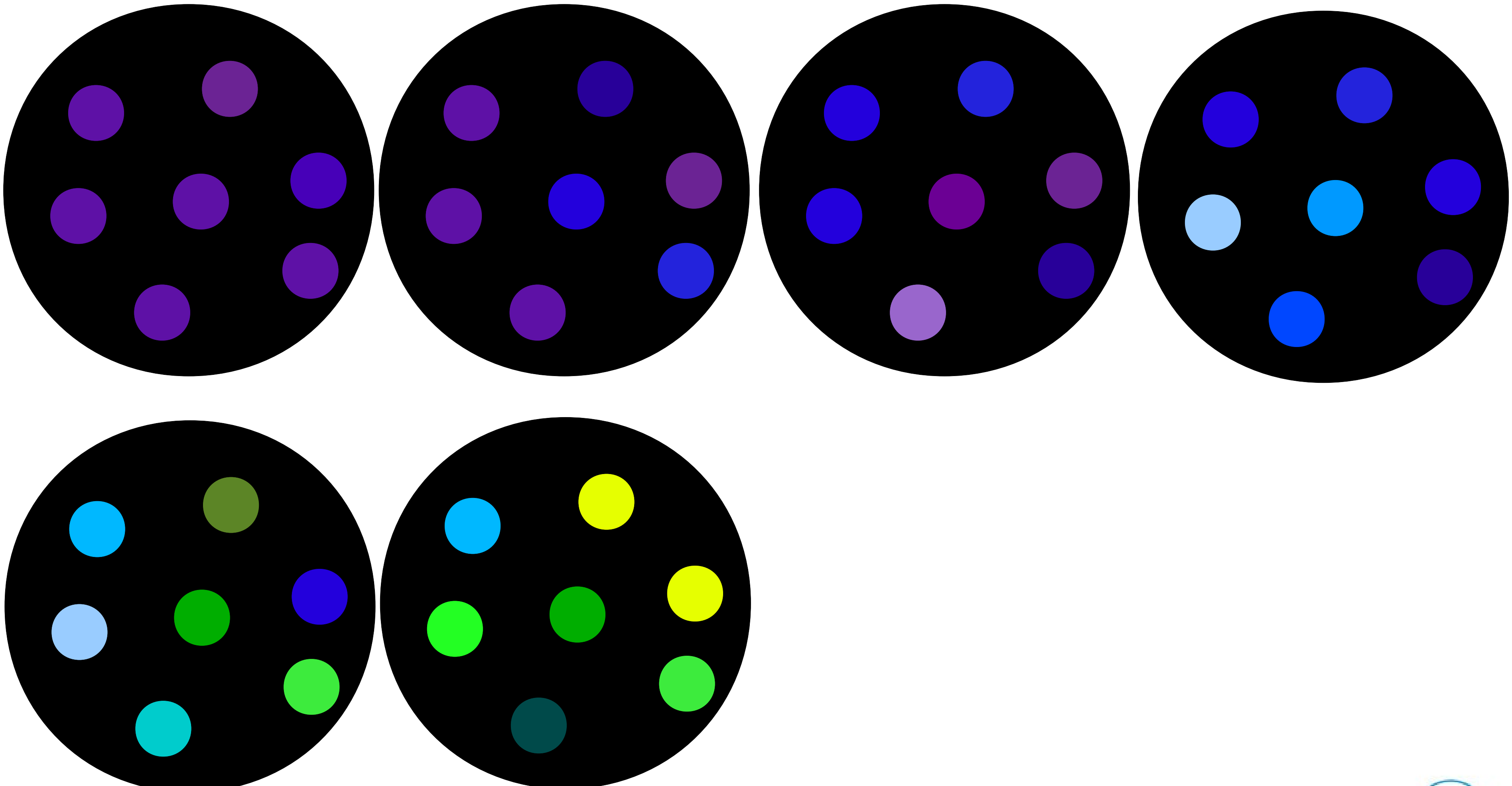
Evolution of solutions



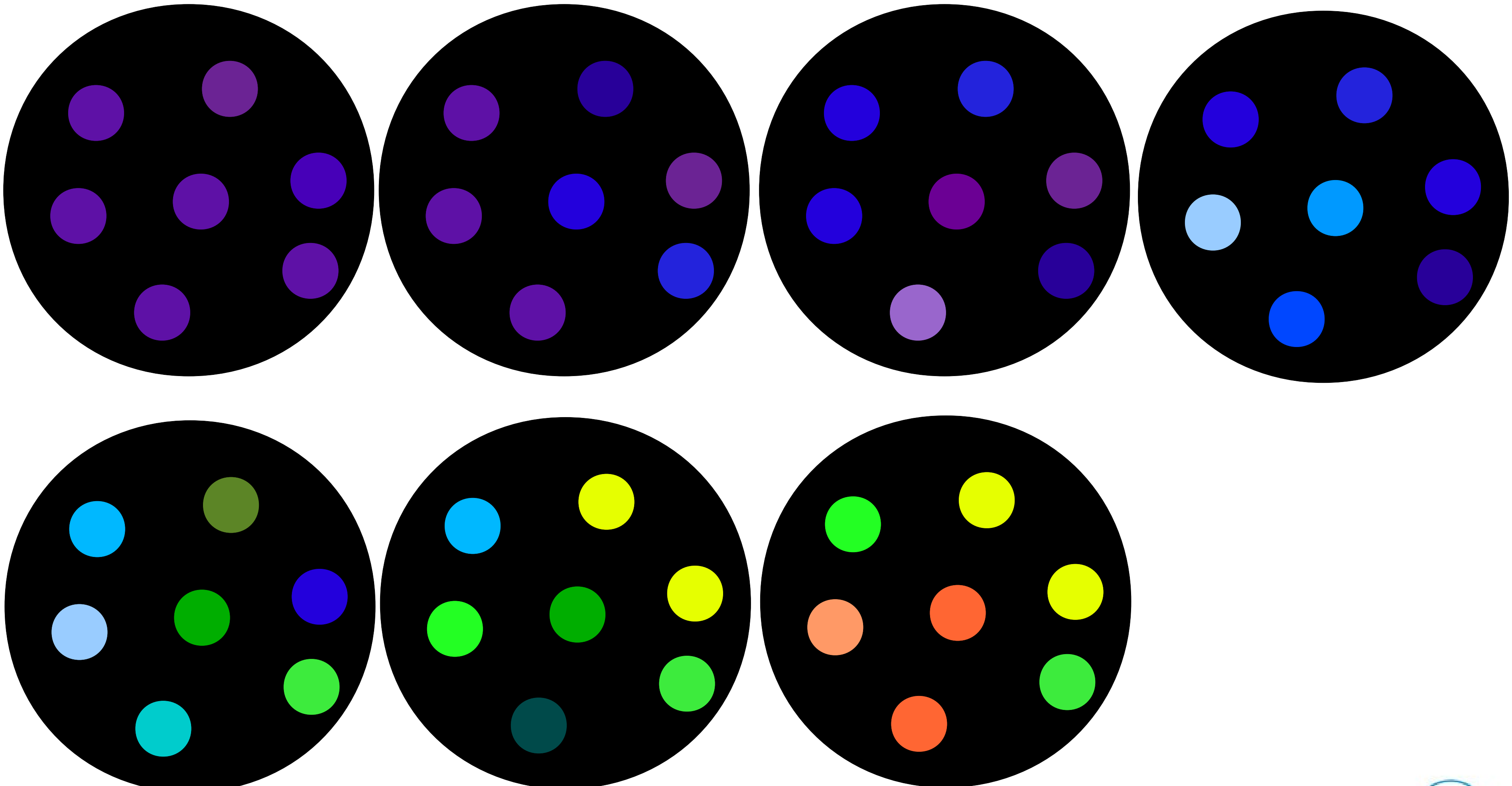
Evolution of solutions



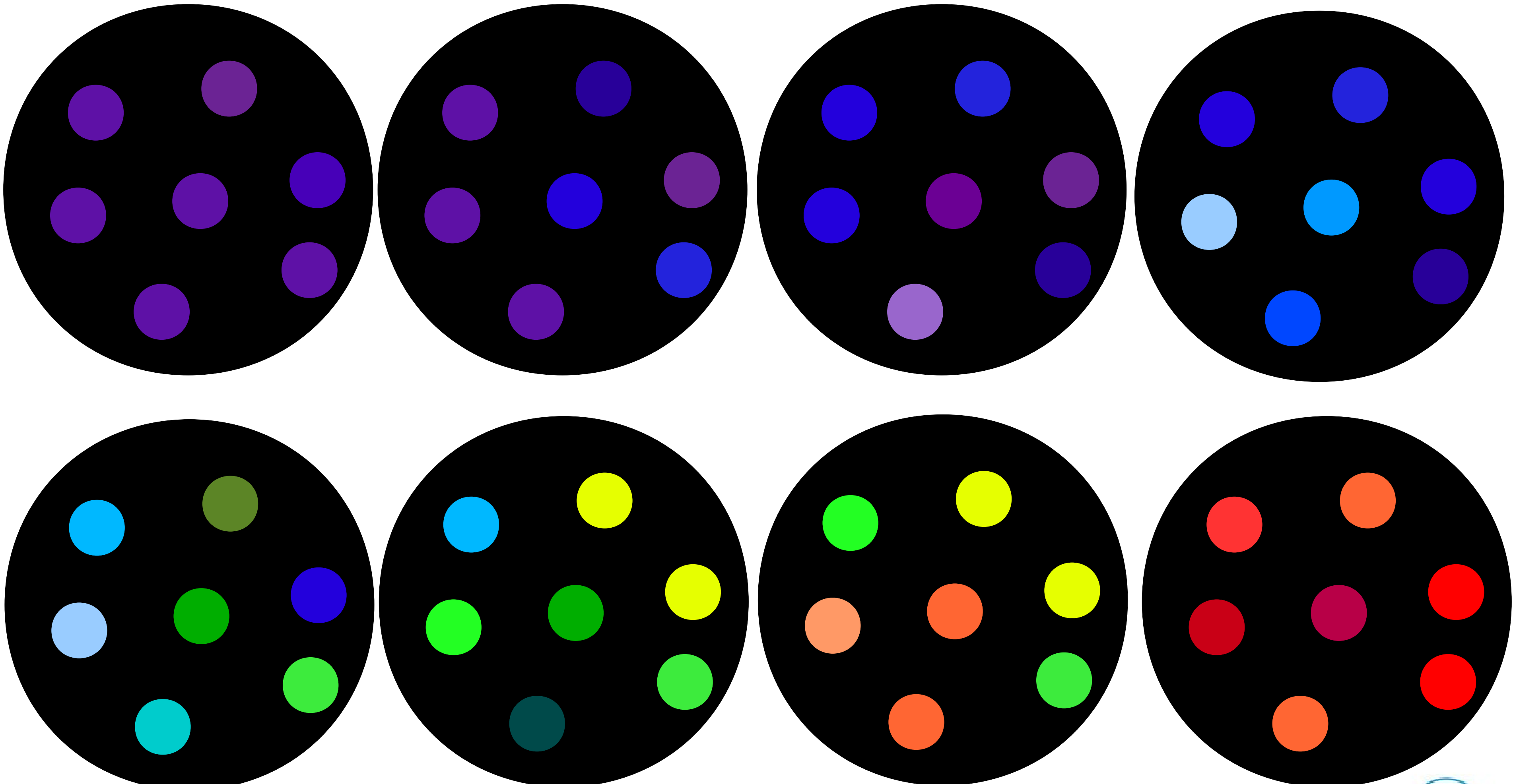
Evolution of solutions



Evolution of solutions



Evolution of solutions



Genetic algorithms with random keys

GAs and random keys

- Introduced by Bean (1994) for sequencing problems.

GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- Individuals are strings of real-valued numbers (random keys) in the interval [0,1].

$$S = (\ 0.25,\ 0.19,\ 0.67,\ 0.05,\ 0.89\ )$$

s(1) s(2) s(3) s(4) s(5)

GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- Individuals are strings of real-valued numbers (random keys) in the interval [0,1].
- Sorting random keys results in a sequencing order.

$$S = (\,0.25,\ 0.19,\ 0.67,\ 0.05,\ 0.89\,)$$
$$\quad s(1)\quad s(2)\quad s(3)\quad s(4)\quad s(5)$$

$$S' = (\,0.05,\ 0.19,\ 0.25,\ 0.67,\ 0.89\,)$$
$$\quad s(4)\quad s(2)\quad s(1)\quad s(3)\quad s(5)$$

Sequence: 4 – 2 – 1 – 3 – 5

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)

$$a = (0.25, 0.19, 0.67, 0.05, 0.89)$$
$$b = (0.63, 0.90, 0.76, 0.93, 0.08)$$

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)

b = (0.63, 0.90, 0.76, 0.93, 0.08)

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = ( )

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25)

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90)

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90, 0.76)

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90, 0.76, 0.05)

GAs and random keys

- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90, 0.76, 0.05, 0.89)

GAs and random keys

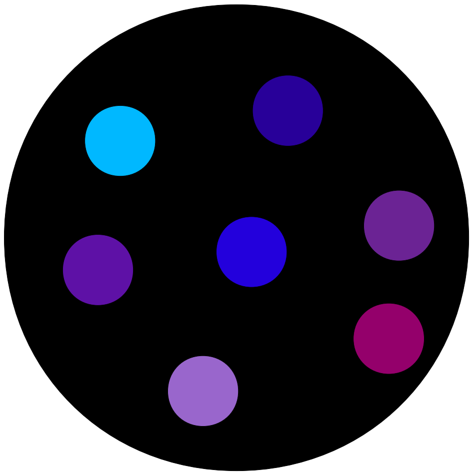
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90, 0.76, 0.05, 0.89)

If every random-key array corresponds to a feasible solution: Mating always produces feasible offspring.

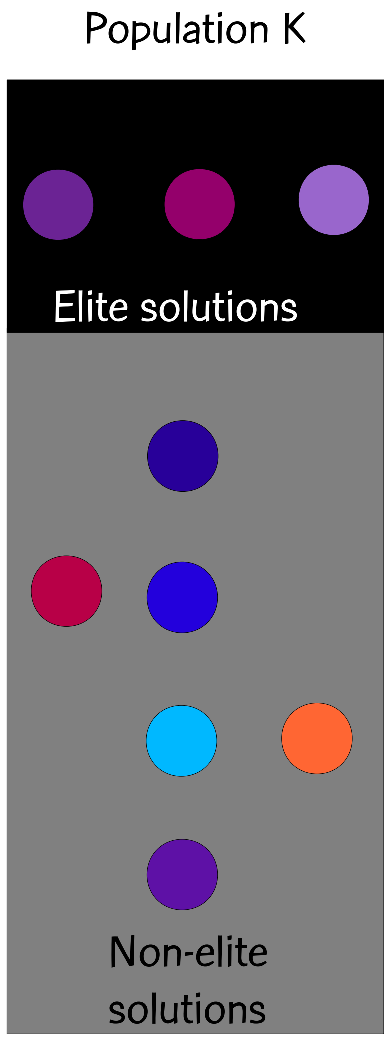
GAs and random keys

Initial population is made up of P chromosomes, each with N genes, each having a value (allele) generated uniformly at random in the interval [0,1].



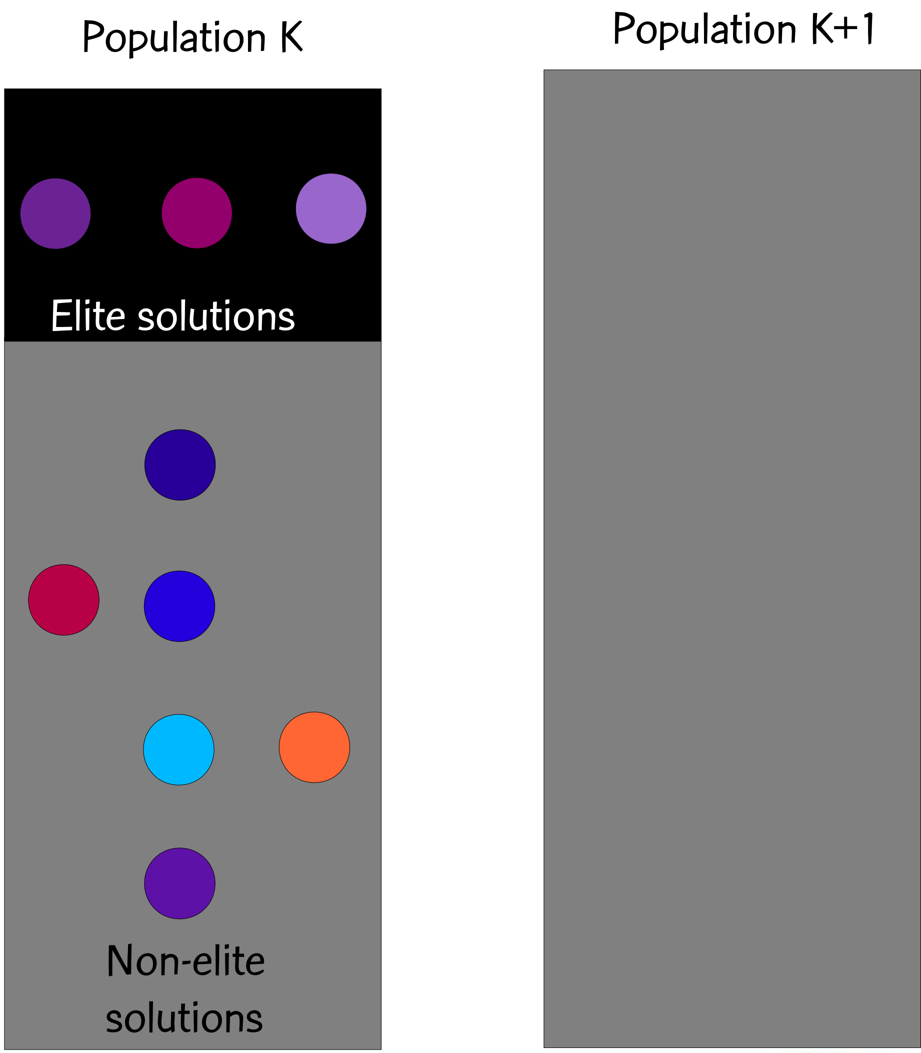
GAs and random keys

At the K-th generation, compute the cost of each solution and partition the solutions into two sets: elite solutions, non-elite solutions. Elite set should be smaller of the two sets and contain best solutions.



GAs and random keys

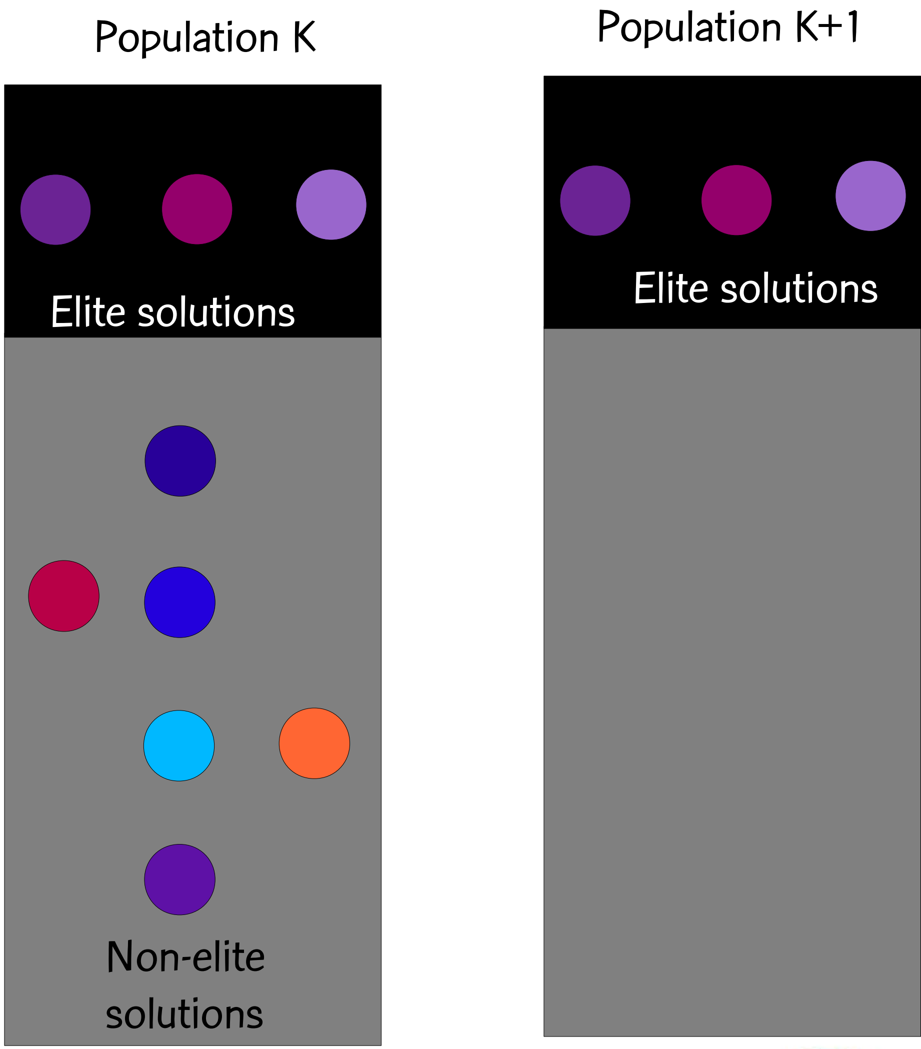
Evolutionary dynamics



GAs and random keys

Evolutionary dynamics

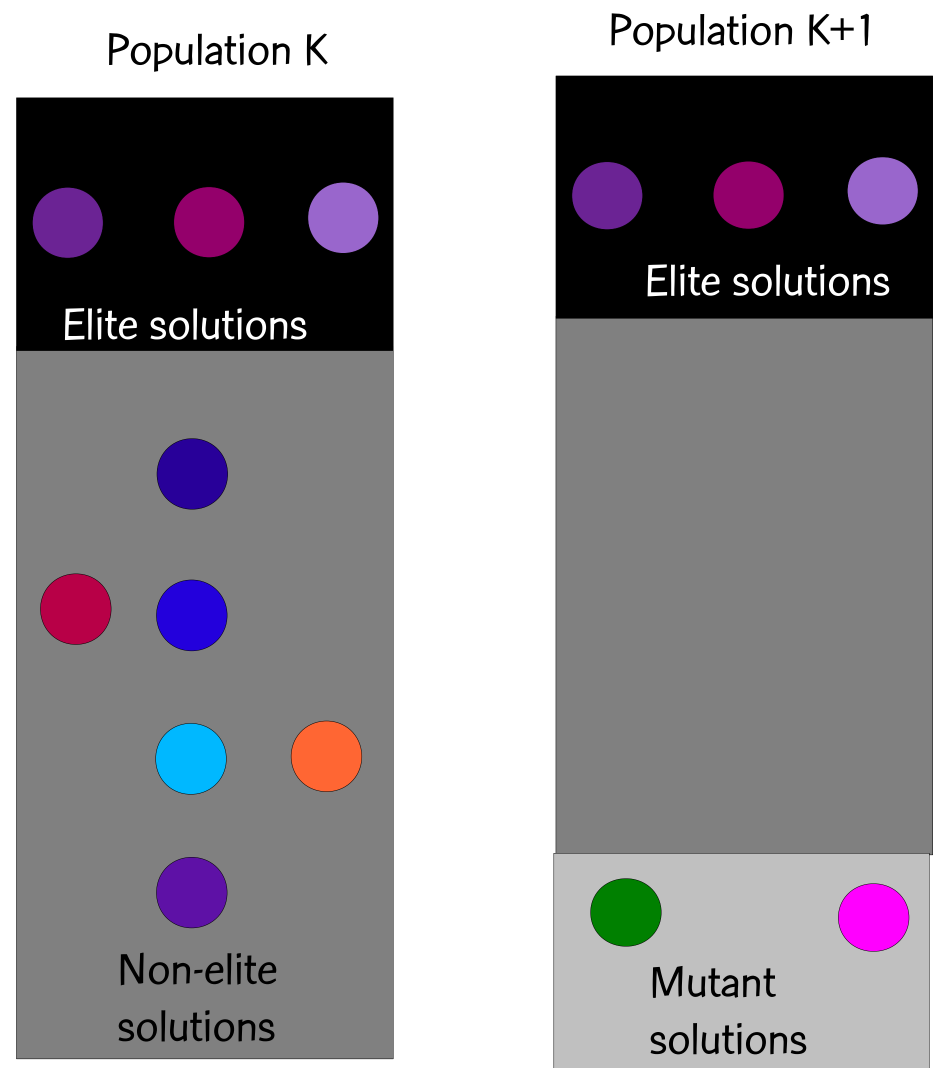
- Copy elite solutions from population K to population K+1



GAs and random keys

Evolutionary dynamics

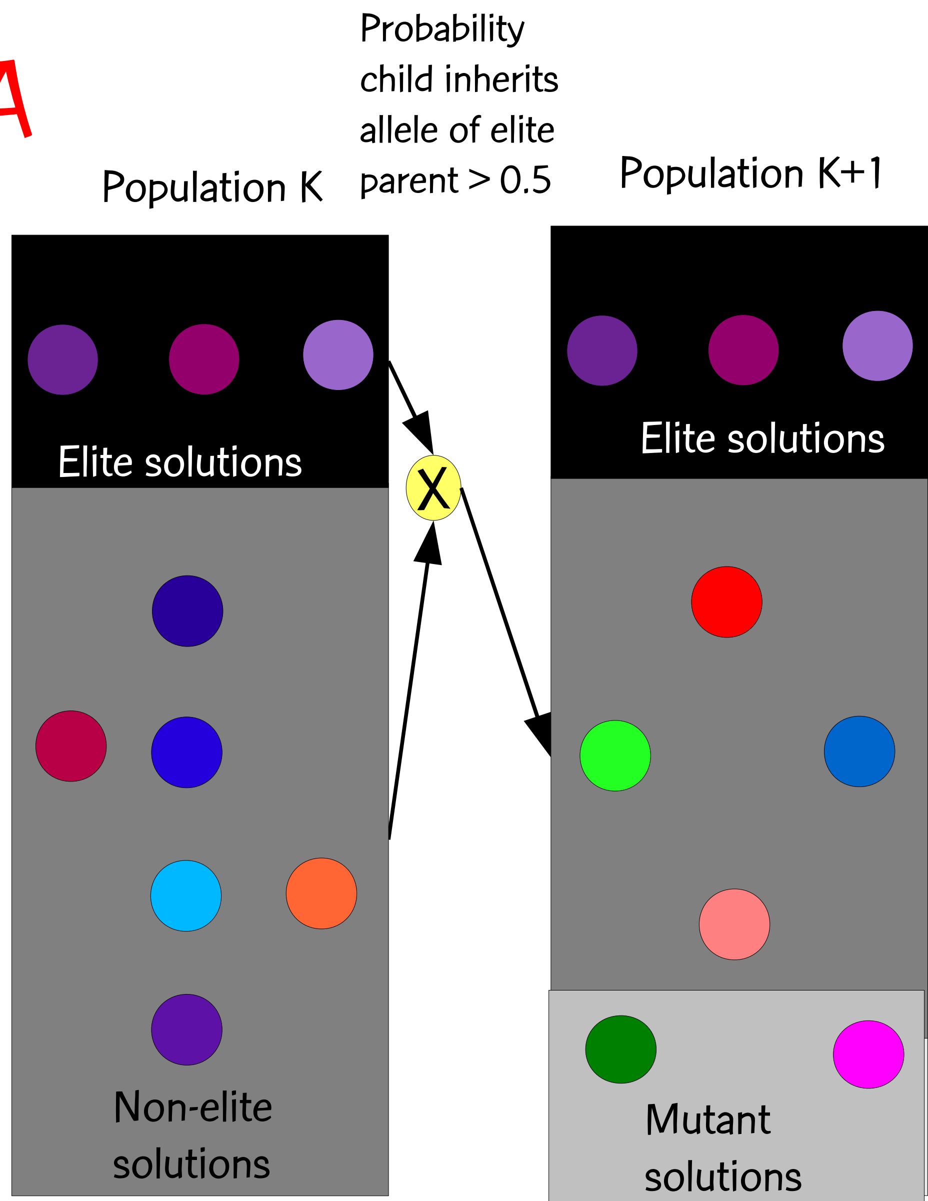
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1



Biased random key GA

Evolutionary dynamics

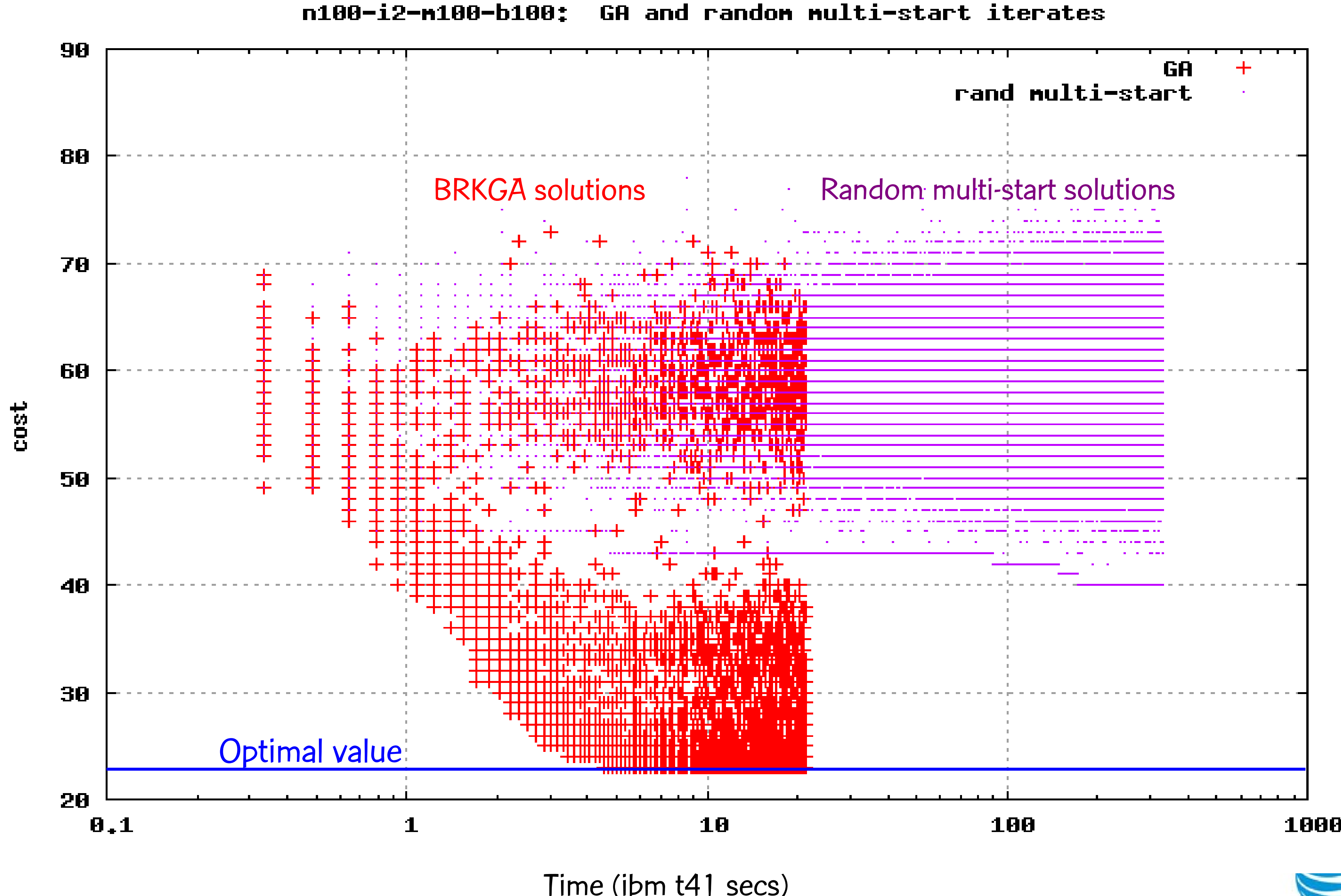
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1
- While K+1-th population < P
  - BIASED RANDOM KEY GA: Mate elite solution with non elite to produce child in population K+1. Mates are chosen at random.



Observations

- Random method: keys are randomly generated so solutions are always random vectors
- Elitist strategy: best solutions are passed without change from one generation to the next
- Child inherits more characteristics of elite parent: one parent is always selected (with replacement) from the small elite set and probability that child inherits key of elite parent > 0.5
- No mutation in crossover: mutants are used instead

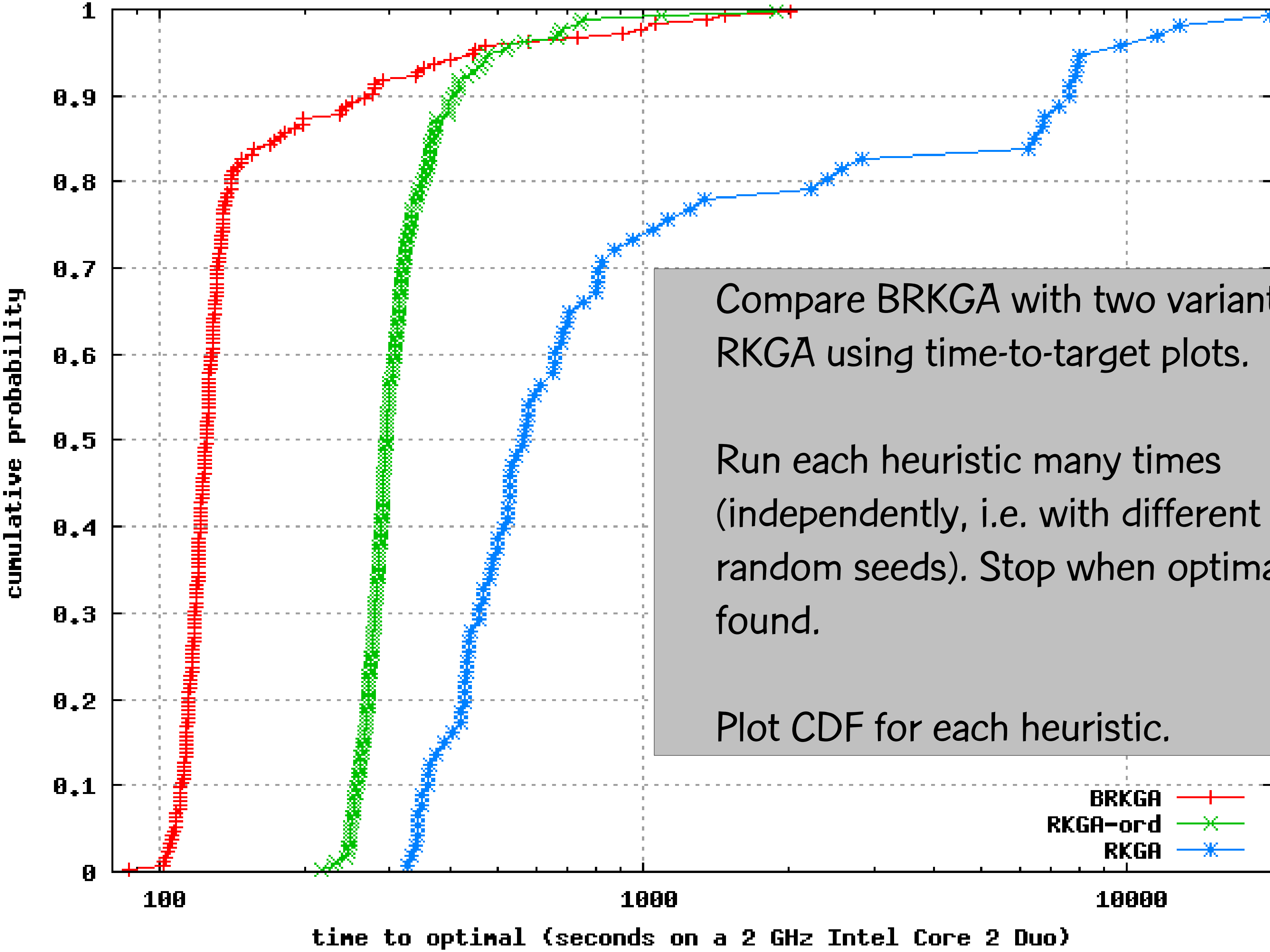
solution



n100-i2-m100-b100: GA and random multi-start iterates
GA
rand multi-start
BRKGA solutions
Random multi-start solutions
Optimal value
cost
90
80
70
60
50
40
30
20
0.1
1
10
100
1000
Time (ibm t41 secs)

Random-keys vs biased random-keys

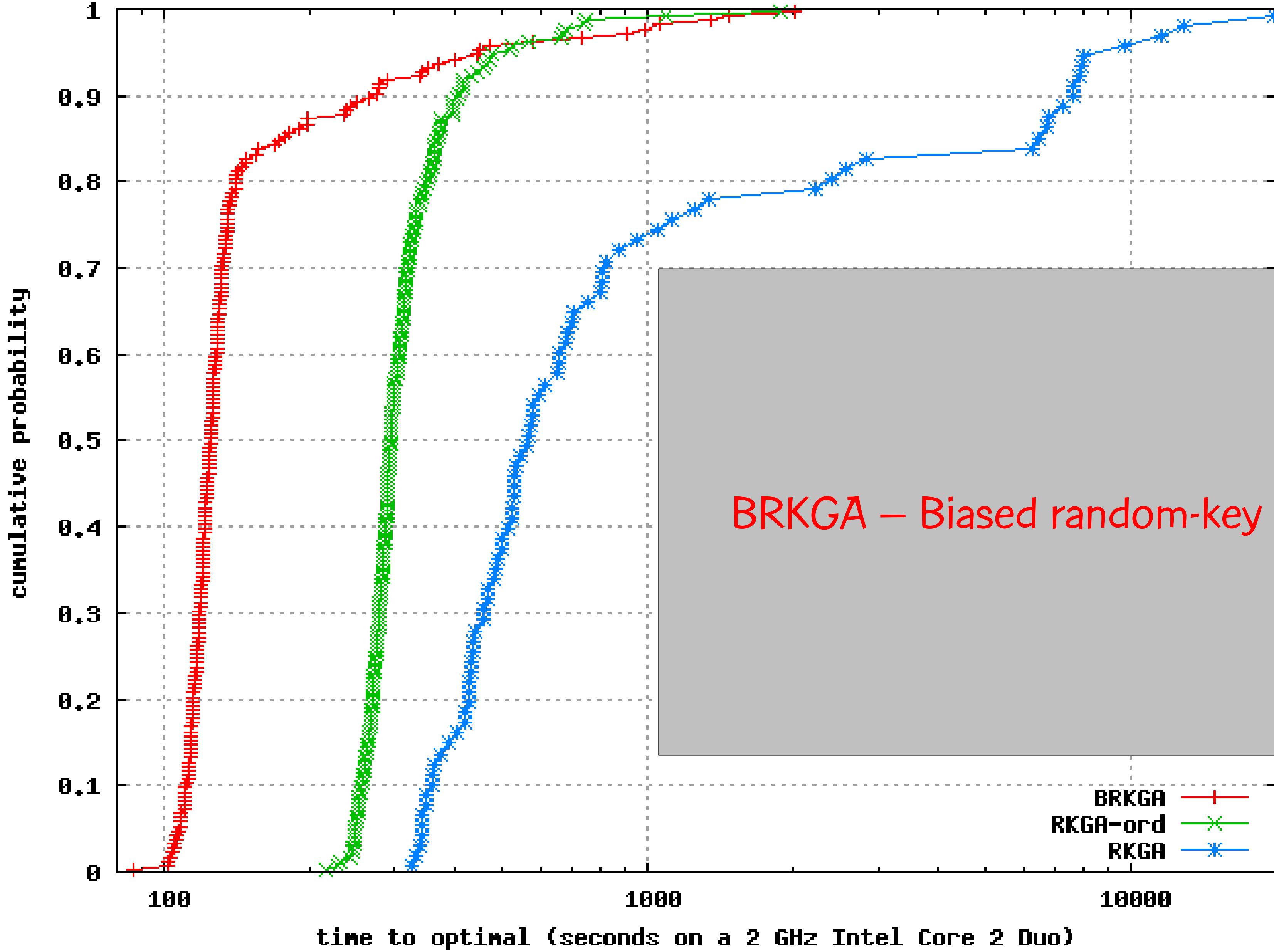
- How do random-key GAs (Bean, 1994) and biased random-key GAs differ?
  - A random-key GA selects both parents at random from the entire population for crossover: some pairs may not have any elite solution
  - A biased random-key GA always has an elite parent during crossover
  - Parametrized uniform crossover makes it more likely that child inherits characteristics of elite parent in biased random-key GA while it does not in random-key GA (survival of the fittest)



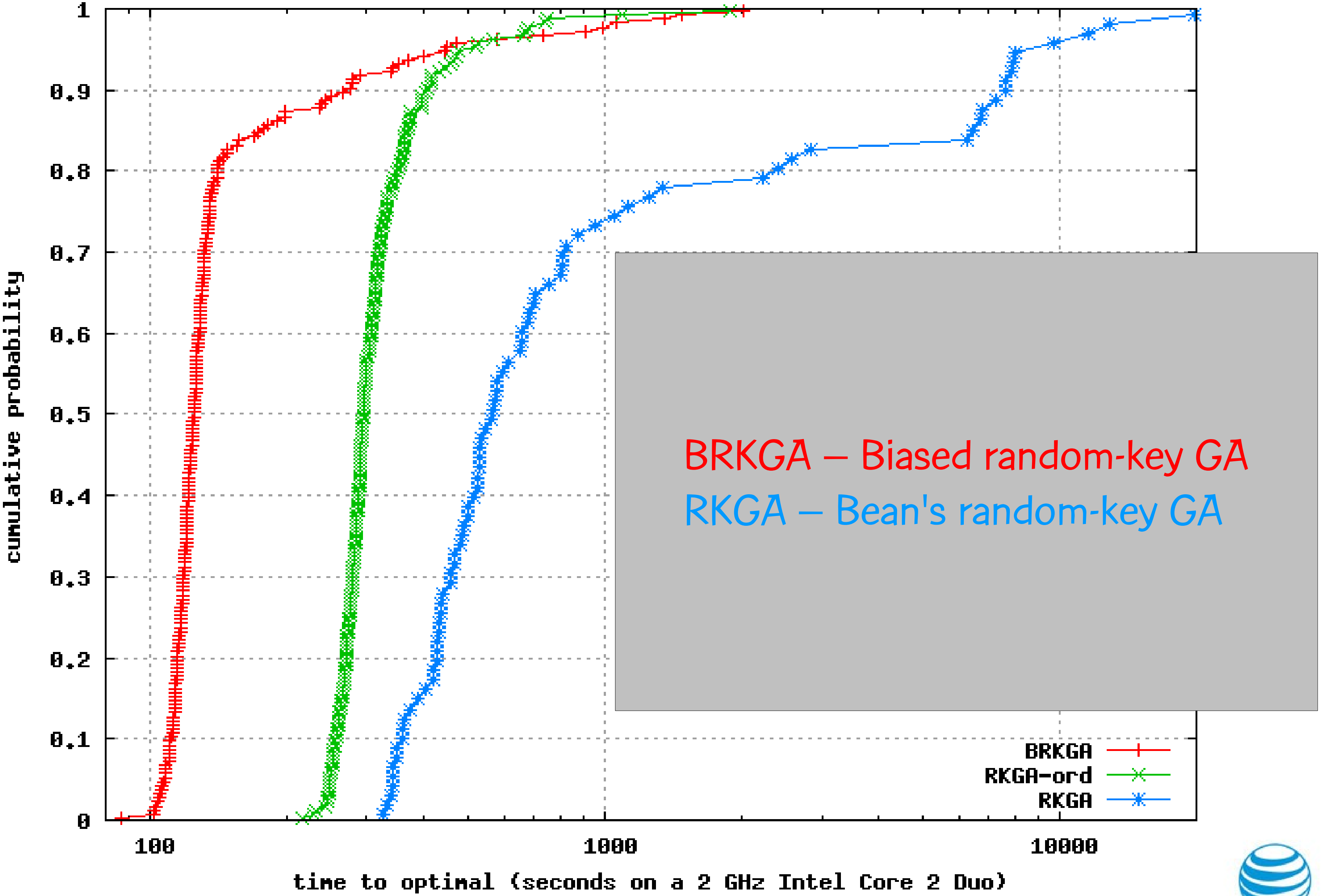
Compare BRKGA with two variants of RKGA using time-to-target plots.

Run each heuristic many times (independently, i.e. with different random seeds). Stop when optimal is found.

Plot CDF for each heuristic.

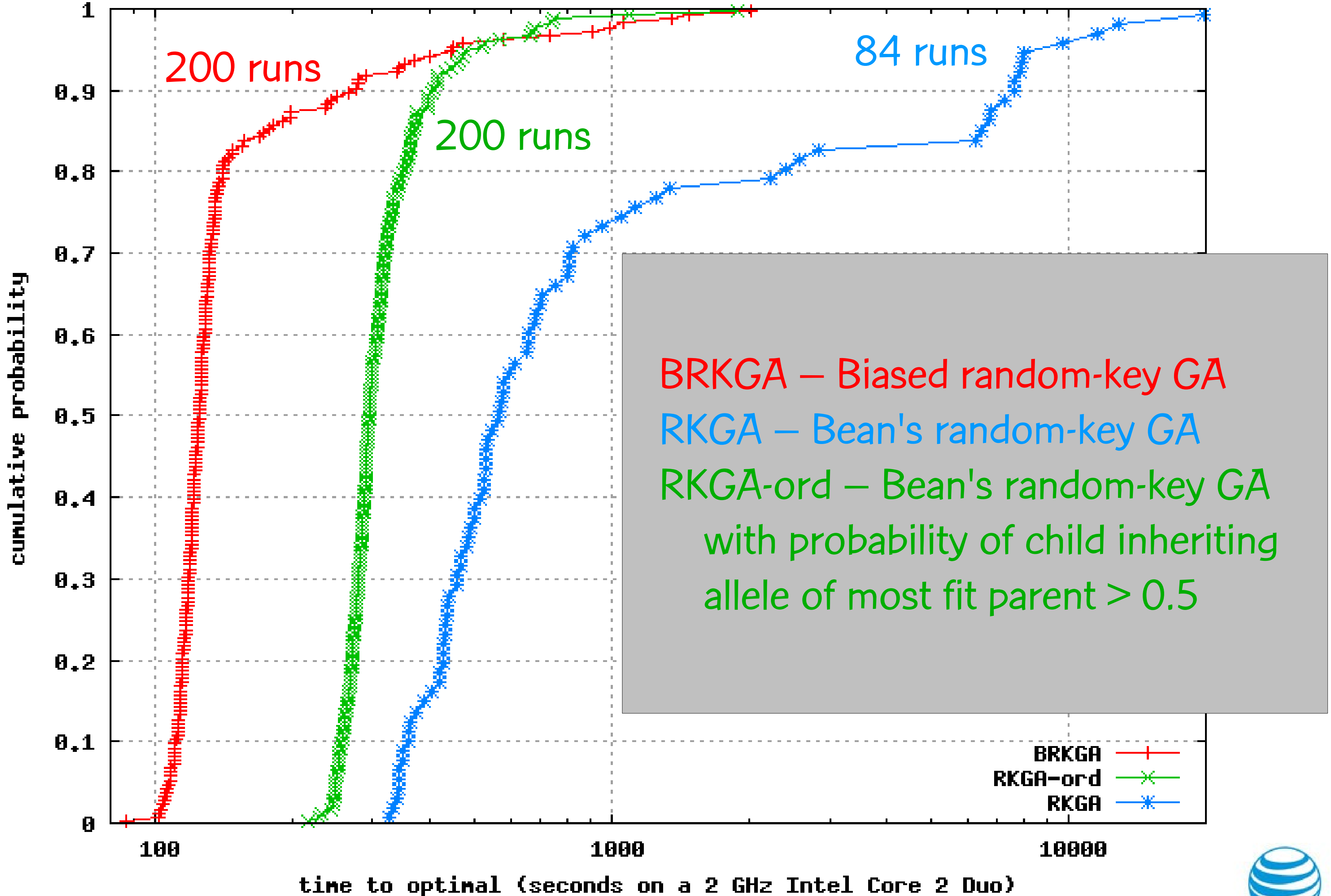


BRKGA – Biased random-key GA



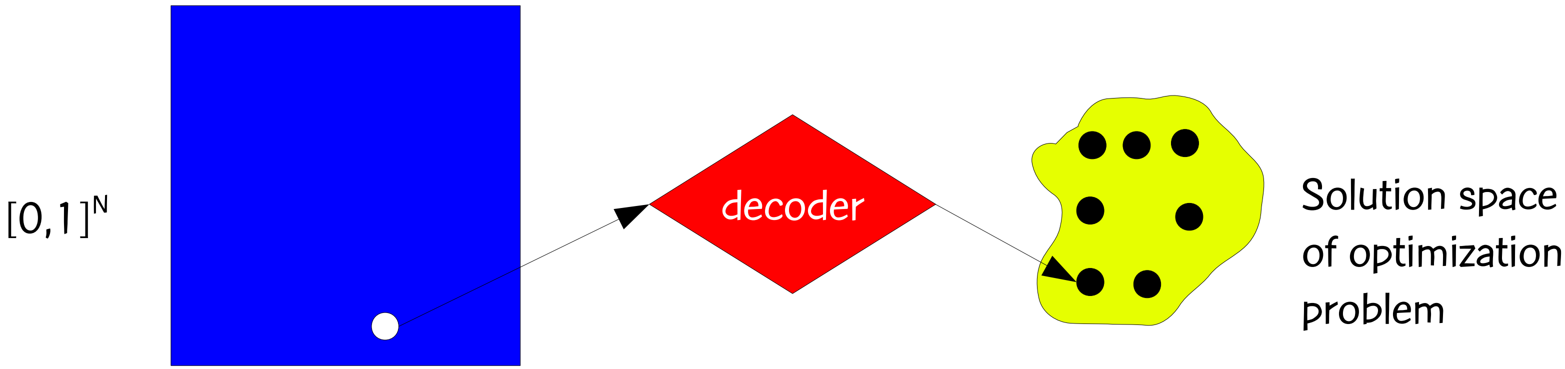
cumulative probability
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
1000
10000
time to optimal (seconds on a 2 GHz Intel Core 2 Duo)
BRKGA
RKGA-ord
RKGA
BRKGA – Biased random-key GA
RKGA – Bean's random-key GA

220 node network monitor location example



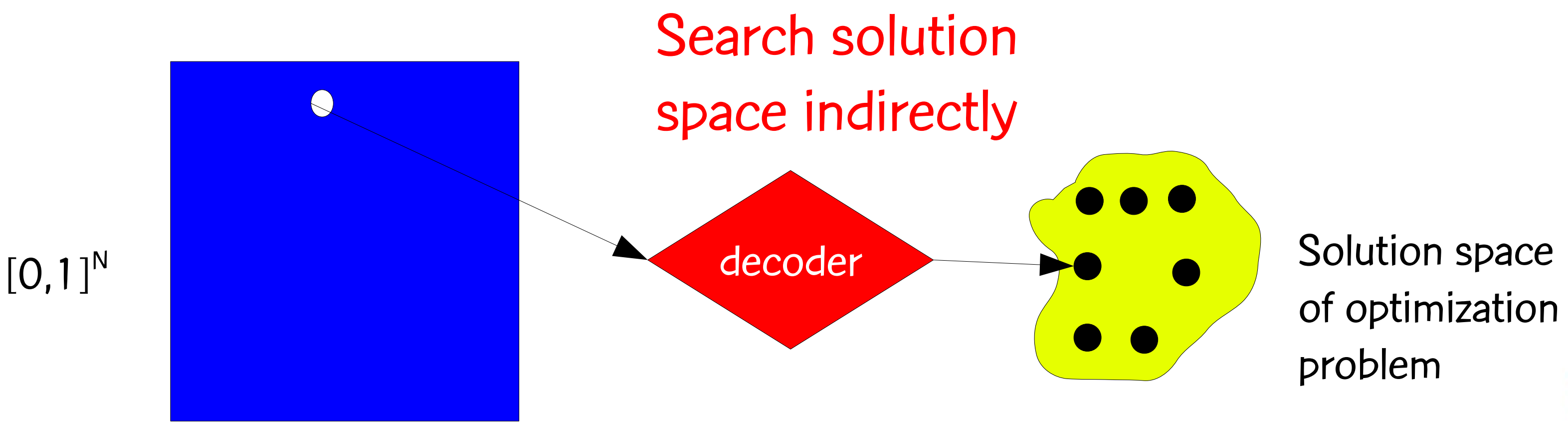
Decoders

- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly by searching the space of random keys and using the decoder to evaluate fitness of the random key.



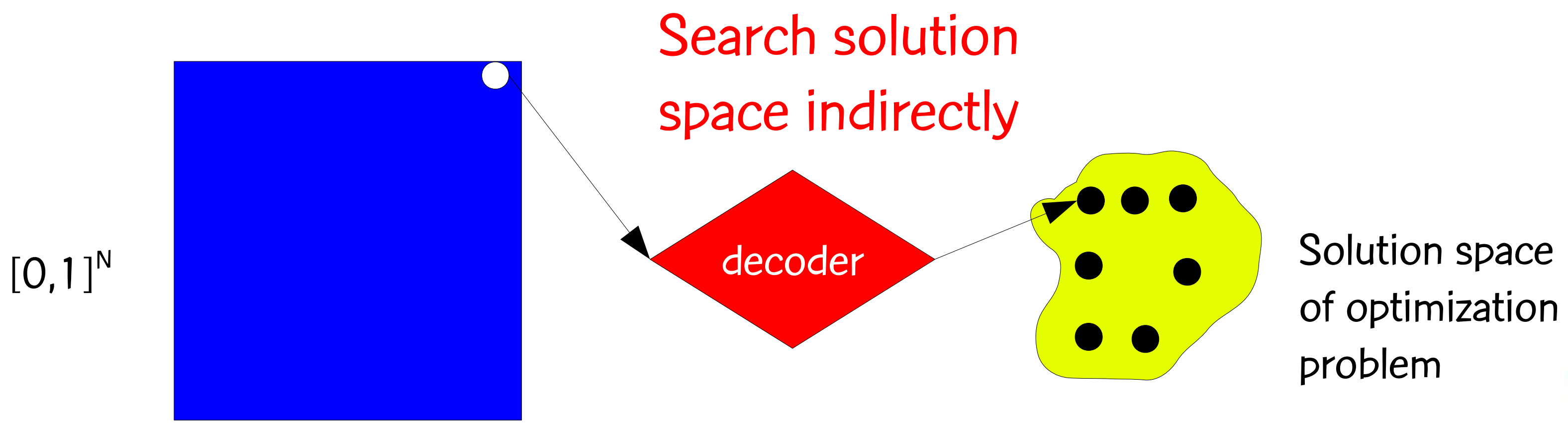
Decoders

- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly by searching the space of random keys and using the decoder to evaluate fitness of the random key.



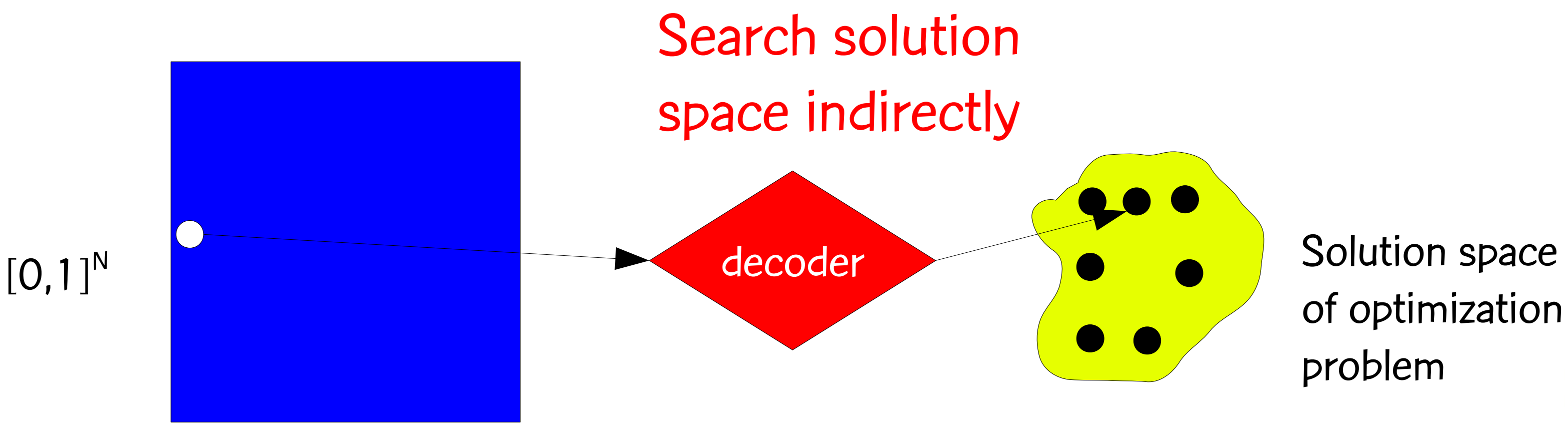
Decoders

- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly by searching the space of random keys and using the decoder to evaluate fitness of the random key.

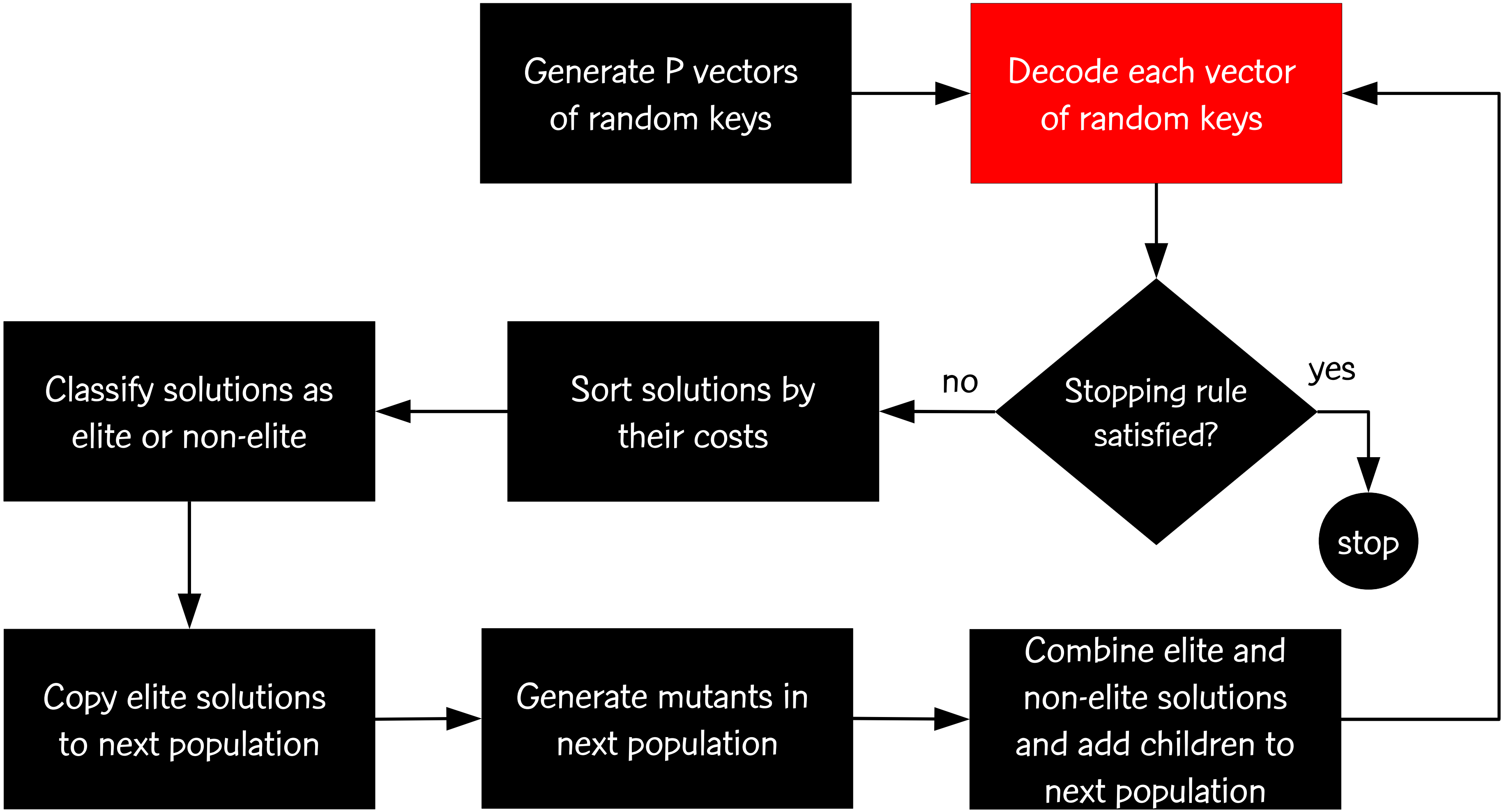


Decoders

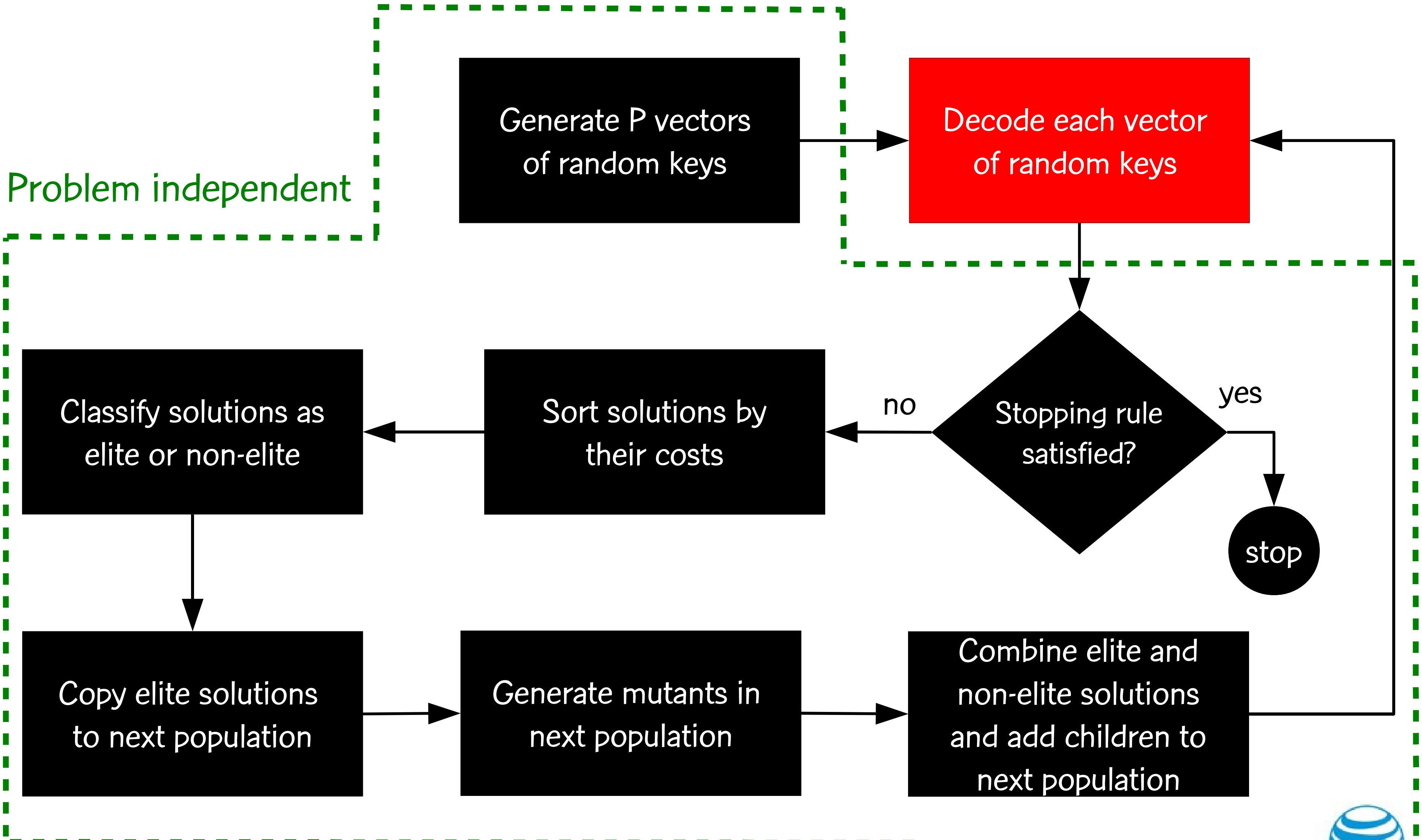
- A **decoder** is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on **sorting** the random-key vector to produce a **sequence**.
- A random-key GA **searches the solution space indirectly** by searching the space of random keys and using the decoder to evaluate fitness of the random key.



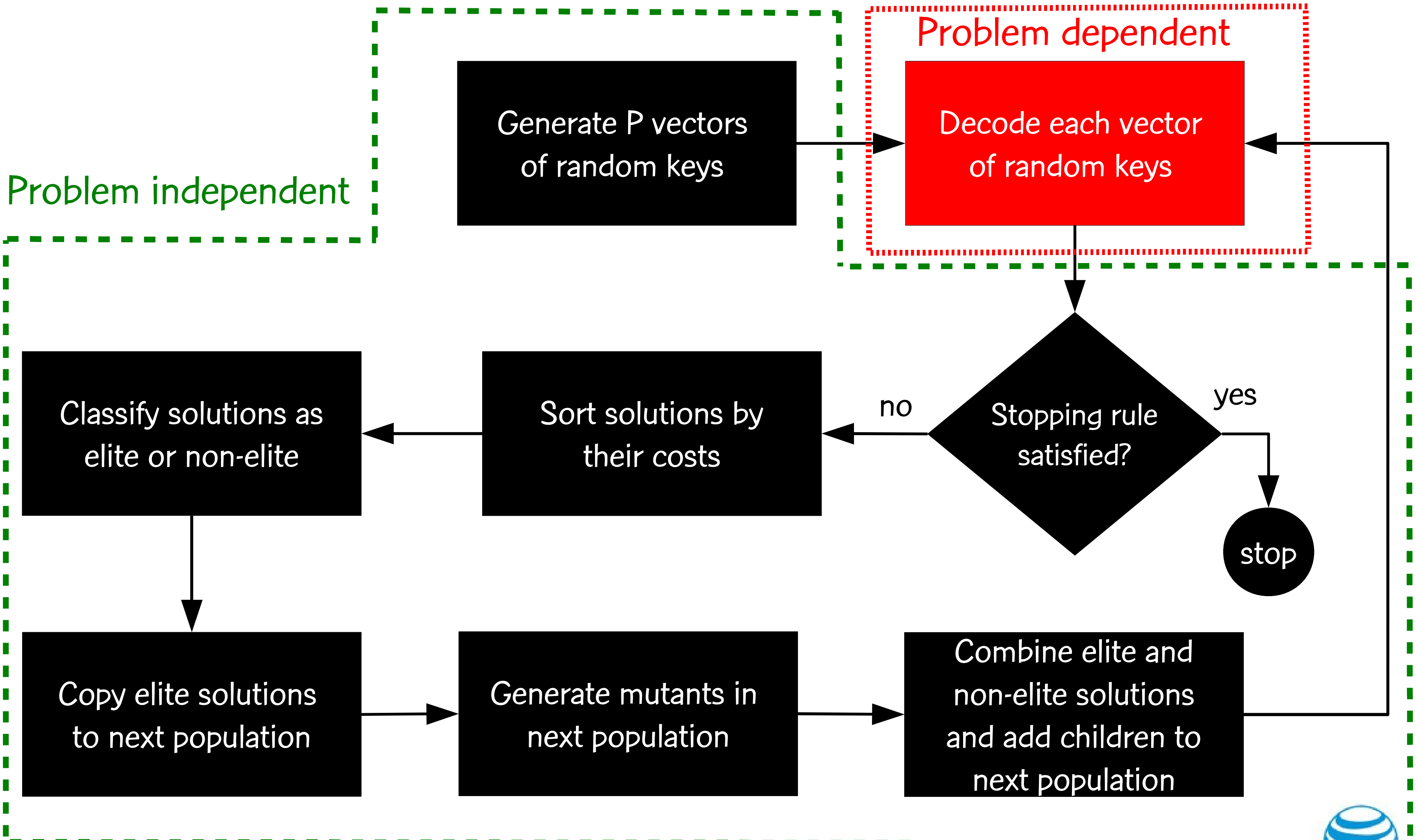
Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Specifying a biased random-key GA

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population
  - Size of elite partition
  - Size of mutant set
  - Child inheritance probability
  - Stopping criterion

Specifying a biased random-key algorithm

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of N, say N or 2N
  - Size of elite partition
  - Size of mutant set
  - Child inheritance probability
  - Stopping criterion

Specifying a biased random-key algorithm

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of N, say N or 2N
  - Size of elite partition: 15-25% of population
  - Size of mutant set
  - Child inheritance probability
  - Stopping criterion

Specifying a biased random-key algorithm

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of N, say N or 2N
  - Size of elite partition: 15-25% of population
  - Size of mutant set: 5-15% of population
  - Child inheritance probability
  - Stopping criterion

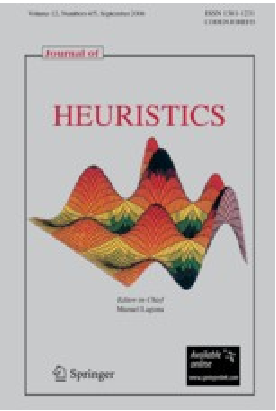
Specifying a biased random-key algorithm

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of N, say N or 2N
  - Size of elite partition: 15-25% of population
  - Size of mutant set: 5-15% of population
  - Child inheritance probability: > 0.5, say 0.7
  - Stopping criterion

Specifying a biased random-key algorithm

- Encoding is always done the same way, i.e. with a vector of N random-keys (parameter N must be specified)
- Decoder that takes as input a vector of N random-keys and outputs the corresponding solution of the combinatorial optimization problem and its cost (this is usually a heuristic)
- Parameters:
  - Size of population: a function of N, say N or 2N
  - Size of elite partition: 15-25% of population
  - Size of mutant set: 5-15% of population
  - Child inheritance probability: > 0.5, say 0.7
  - Stopping criterion: e.g. time, # generations, solution quality, # generations without improvement

Reference



J.F. Gonçalves and M.G.C.R., "Biased random-key genetic algorithms for combinatorial optimization," J. of Heuristics, published online 31 August 2010.

Tech report version:

http://www2.research.att.com/~mgcr/doc/srkga.pdf

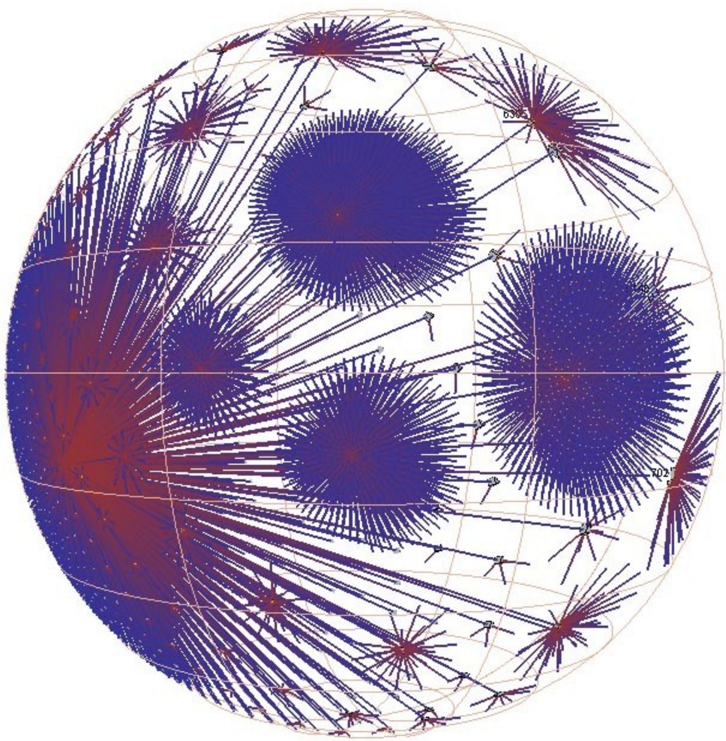
Applications in telecom

Applications in telecom

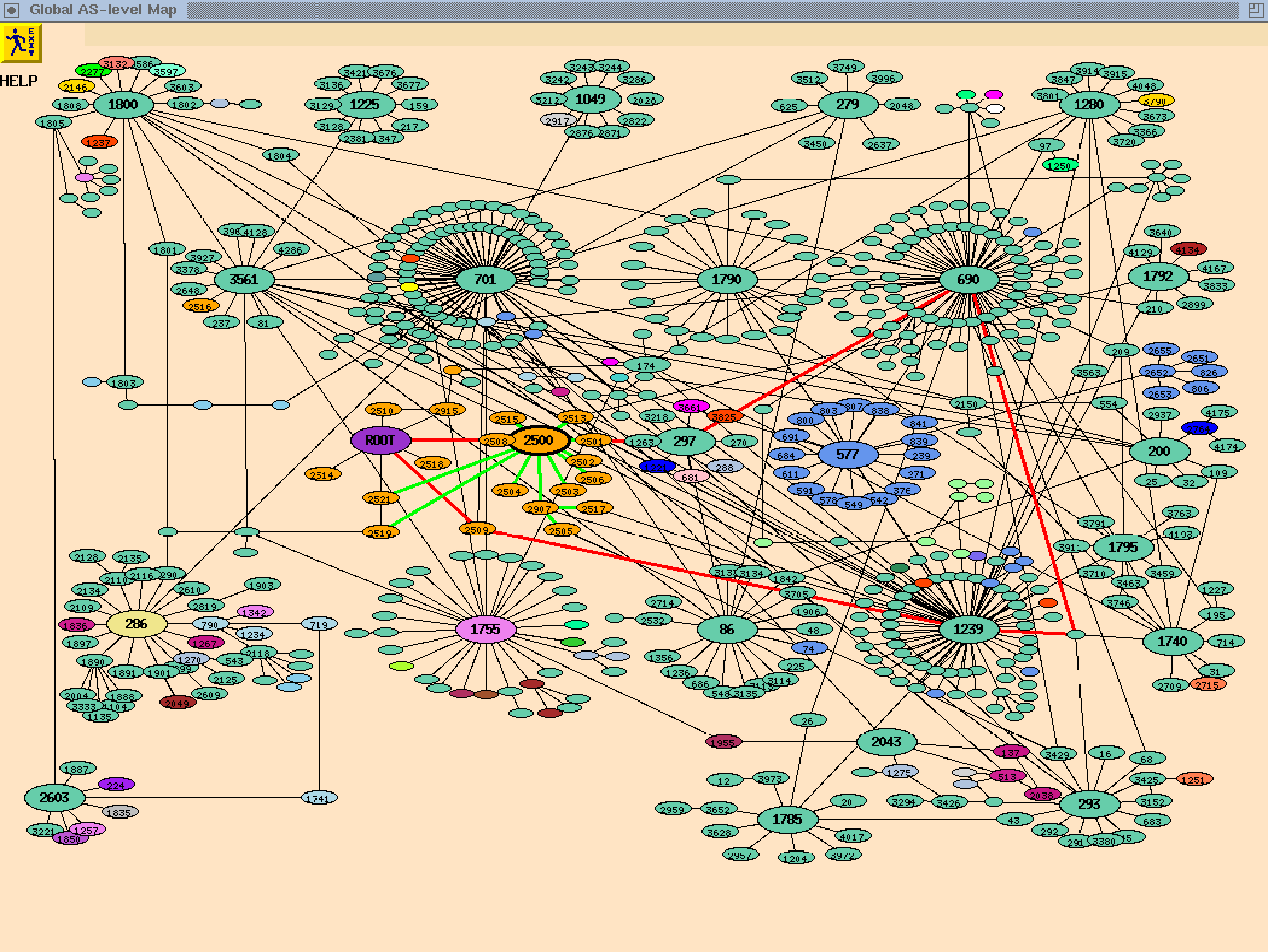
- Routing in IP networks
- Design of survivable IP networks
- Host placement for path-disjoint monitoring
- Routing and wavelength assignment in optical networks

OSPF routing in IP networks

The Internet



- The Internet is composed of many (inter-connected) autonomous systems (AS).
- An AS is a network controlled by a single entity, e.g. ISP, university, corporation, country, ...


Global AS-level Map
EXIT
HELP
ROOT
2500
701
3561
1800
1225
1849
279
1280
1790
690
1792
297
577
200
1795
1755
286
86
1239
1740
2043
2603
1785
293

Routing

- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS:
  - different ASes:

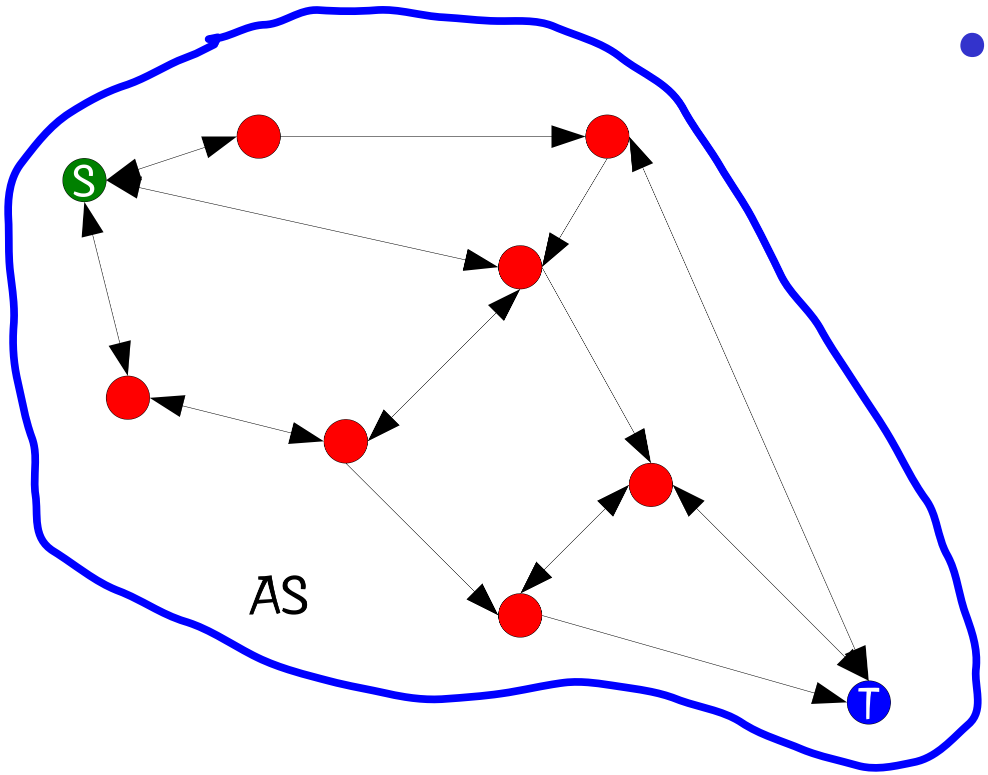
Routing

- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS: IGP routing
  - different ASes:

Routing

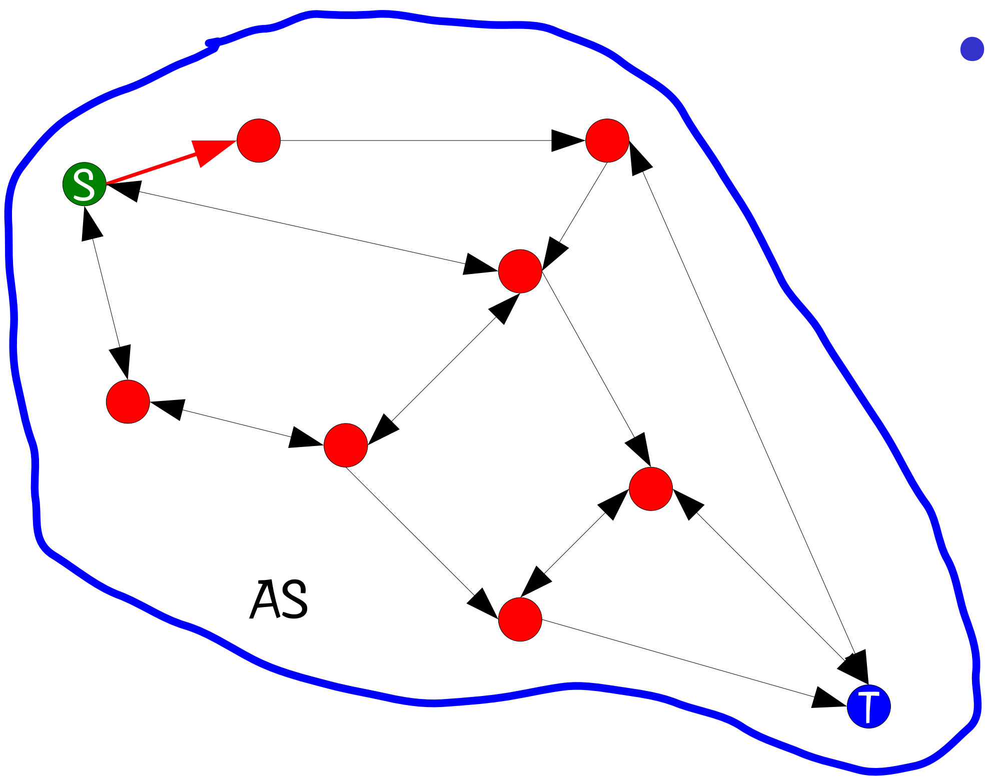
- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS: IGP routing
  - different ASes: BGP routing

IGP Routing



- IGP (interior gateway protocol) routing is concerned with routing within an AS.

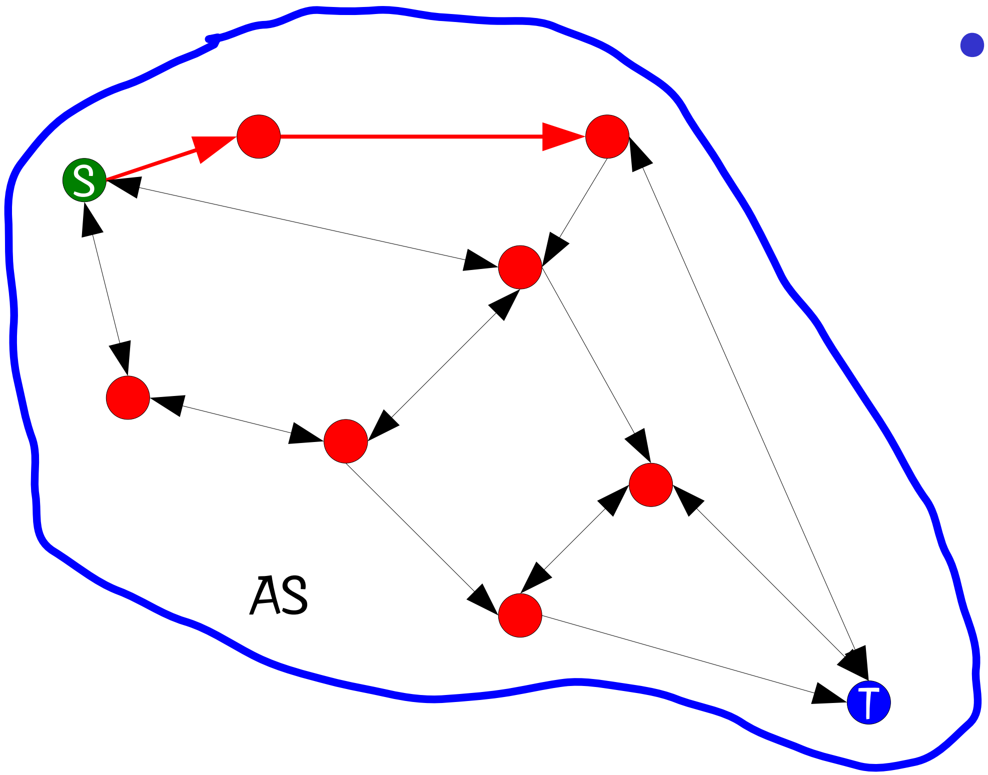
IGP Routing



- IGP (interior gateway protocol) routing is concerned with routing within an AS.

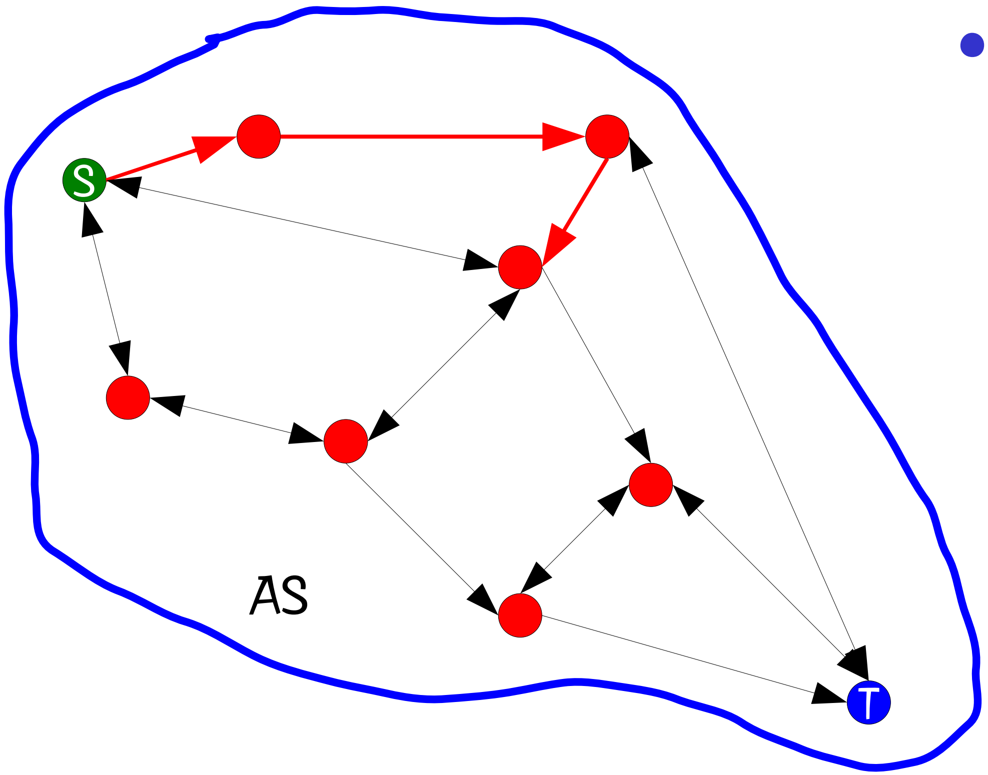
at&t Your world. Delivered.

IGP Routing



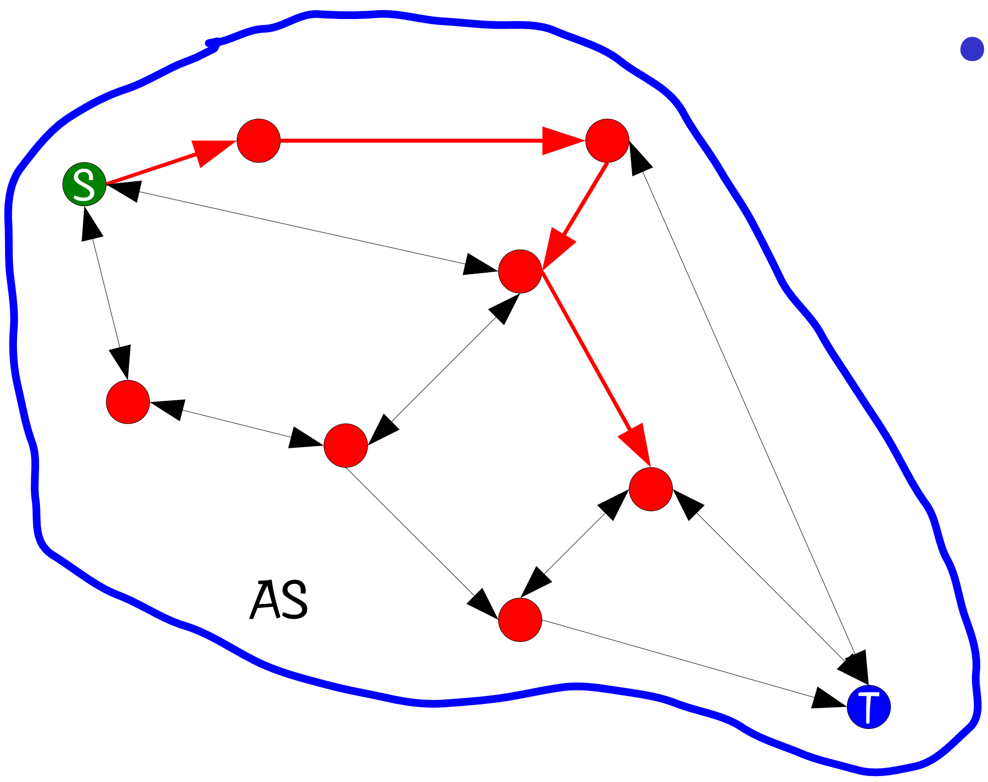
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



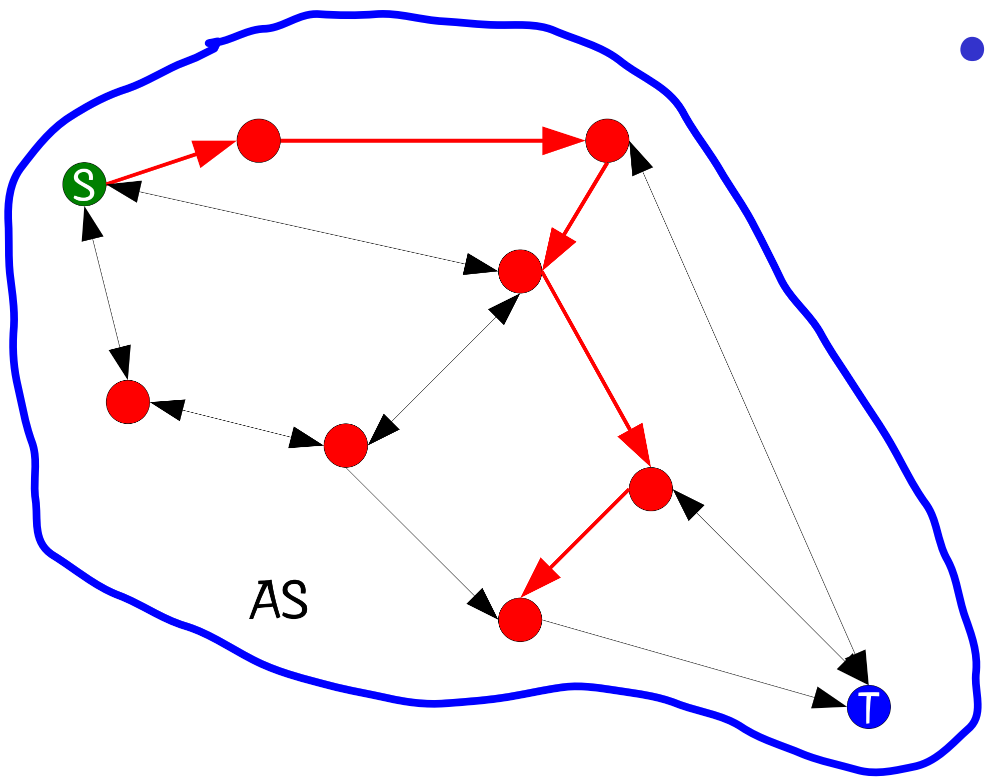
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



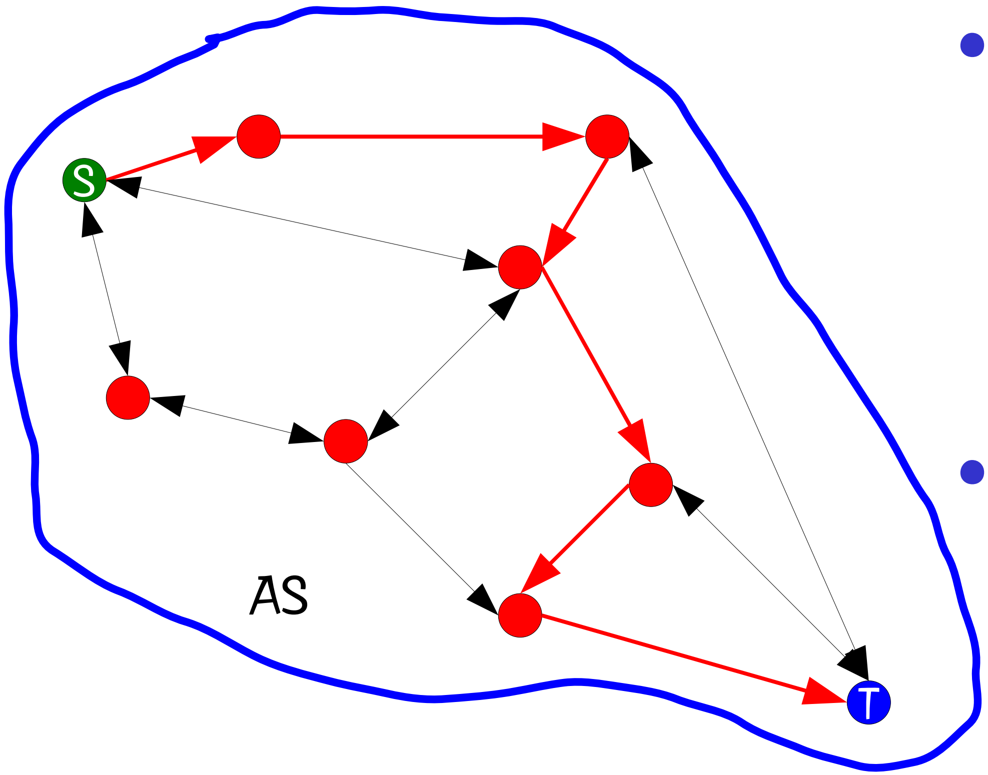
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



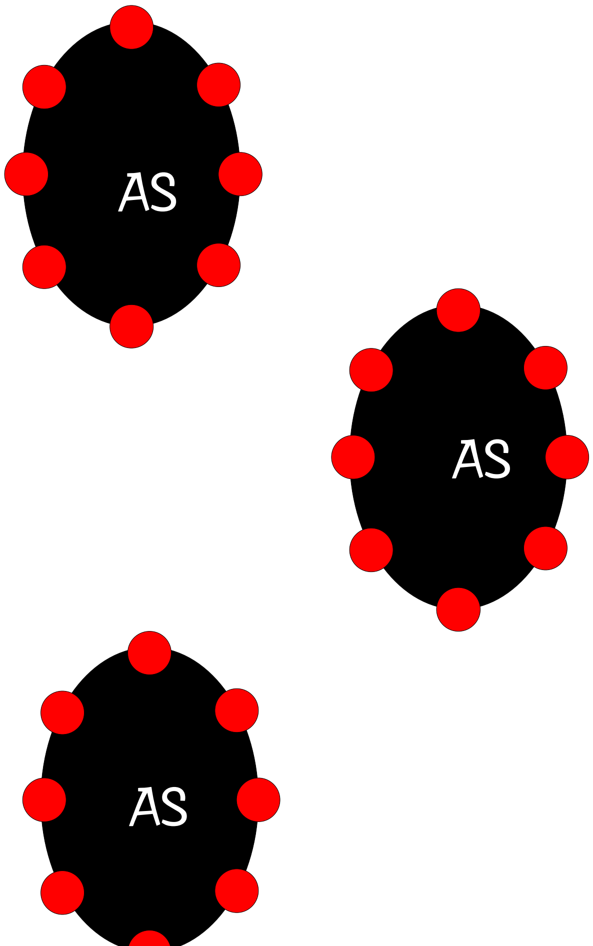
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



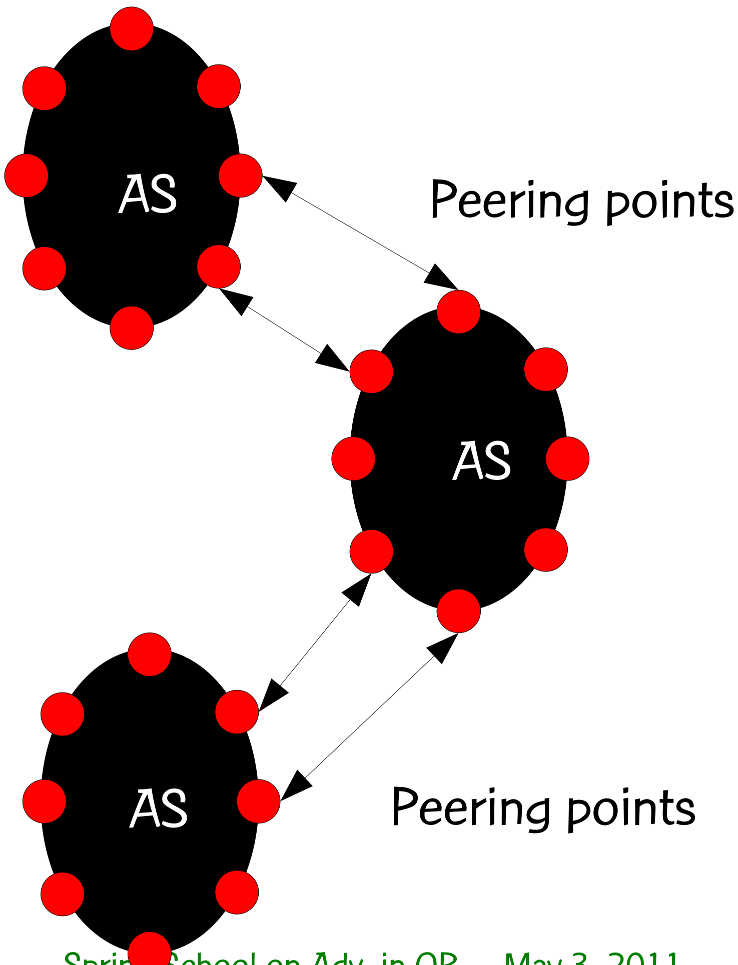
- IGP (interior gateway protocol) routing is concerned with routing within an AS.
- Routing decisions are made by AS operator.

BGP Routing



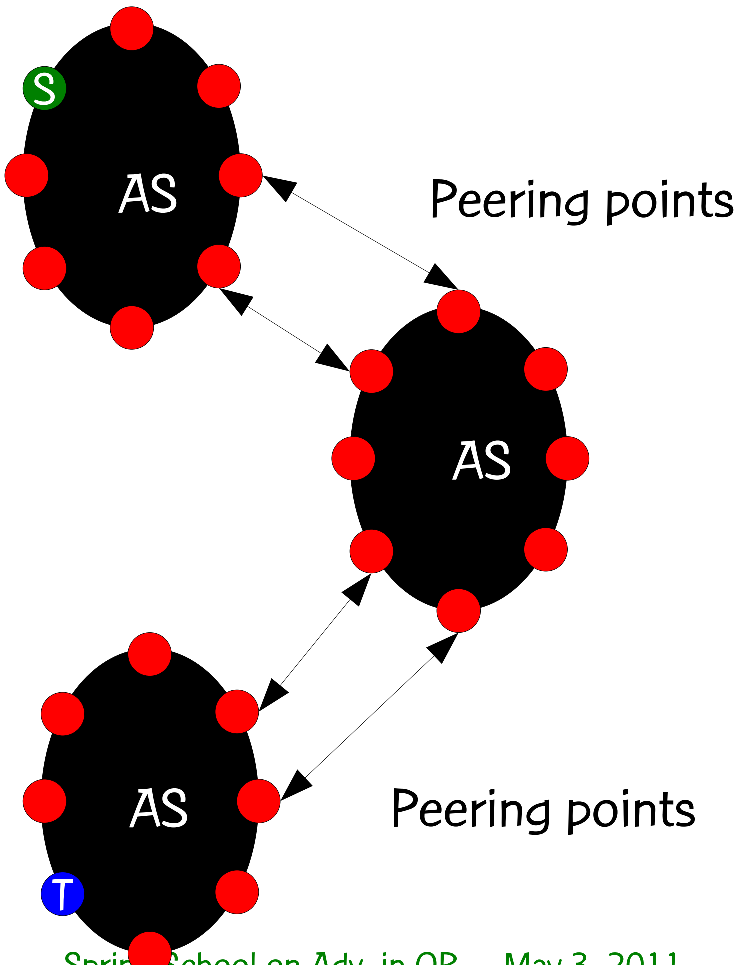
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



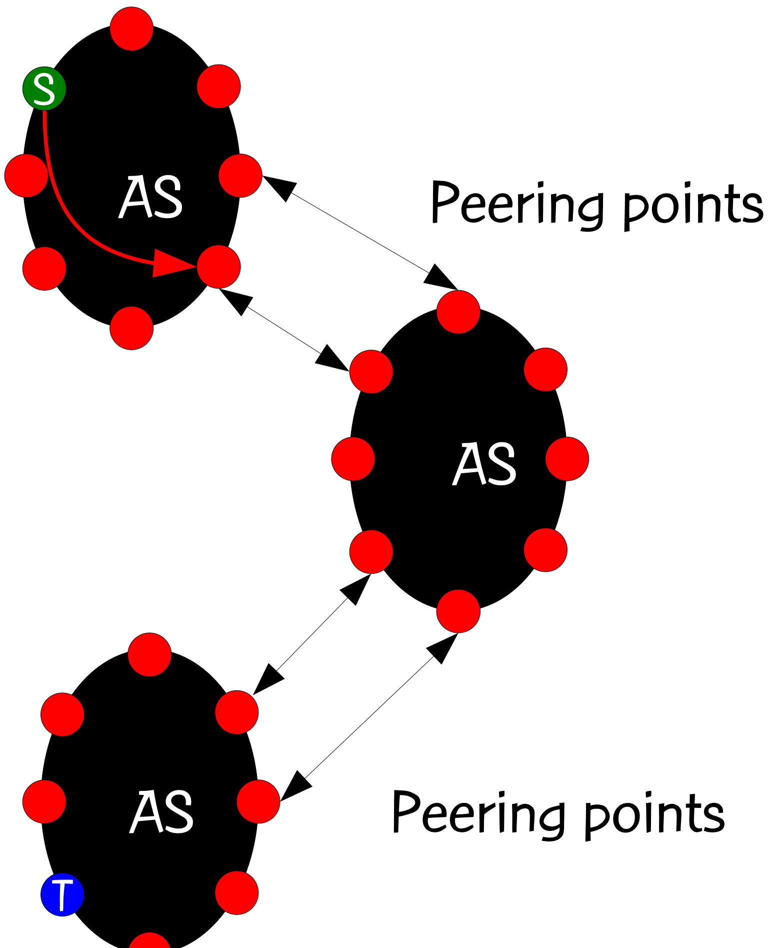
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



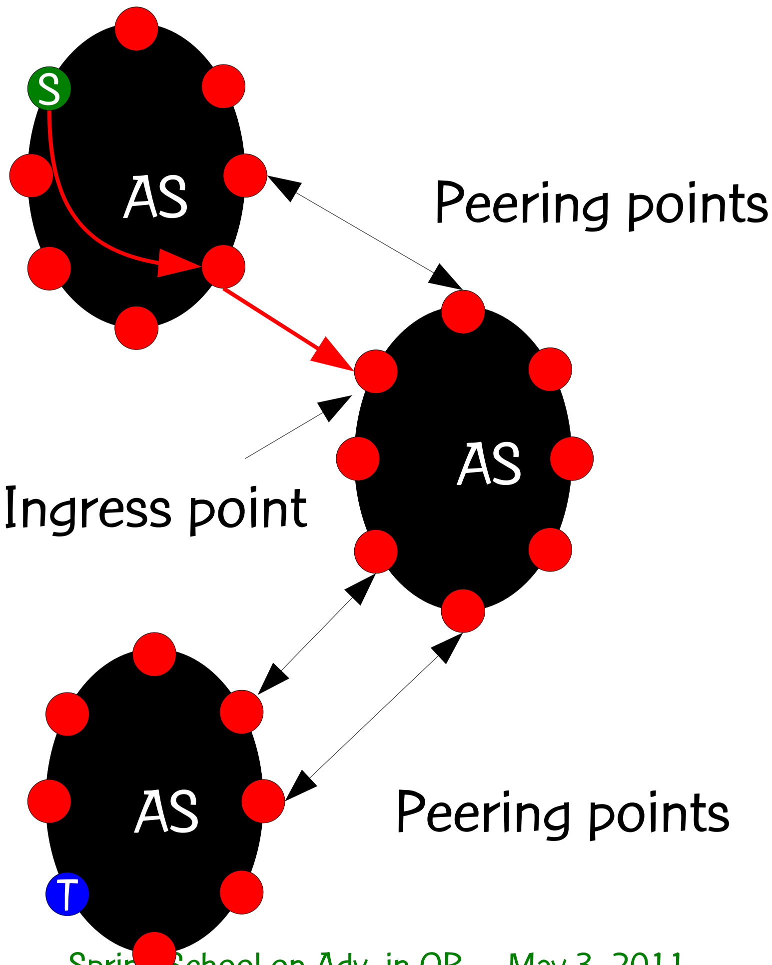
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



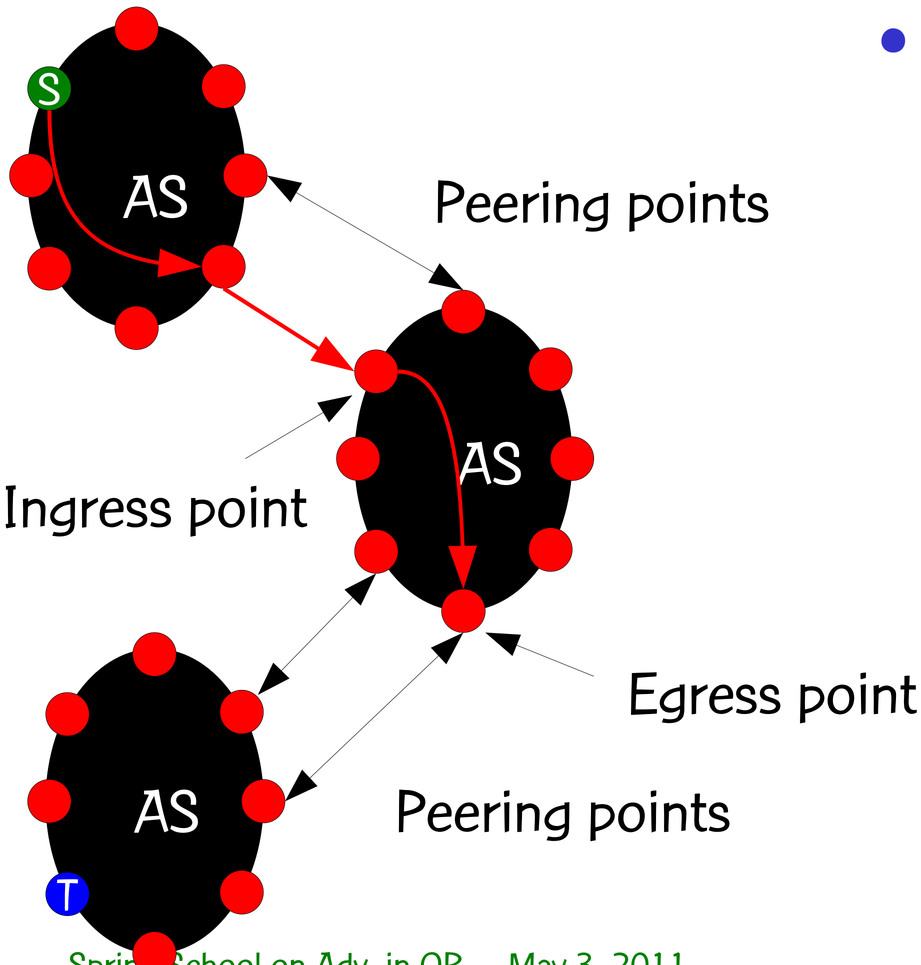
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



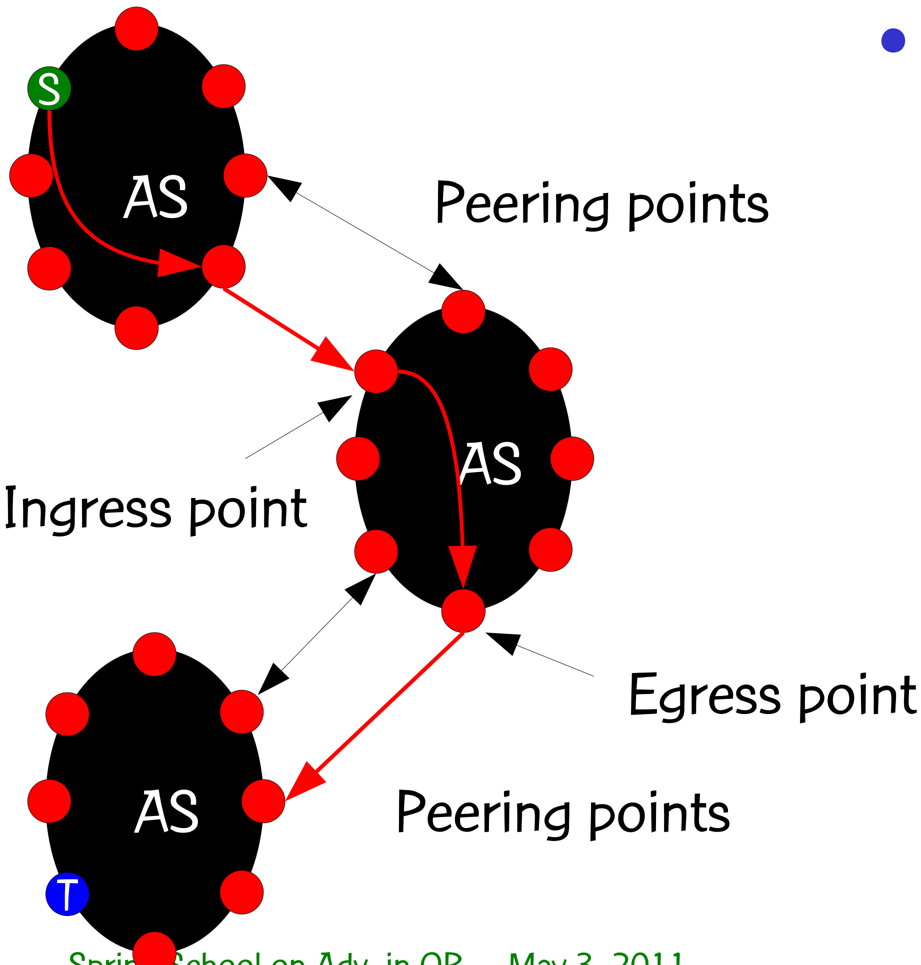
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



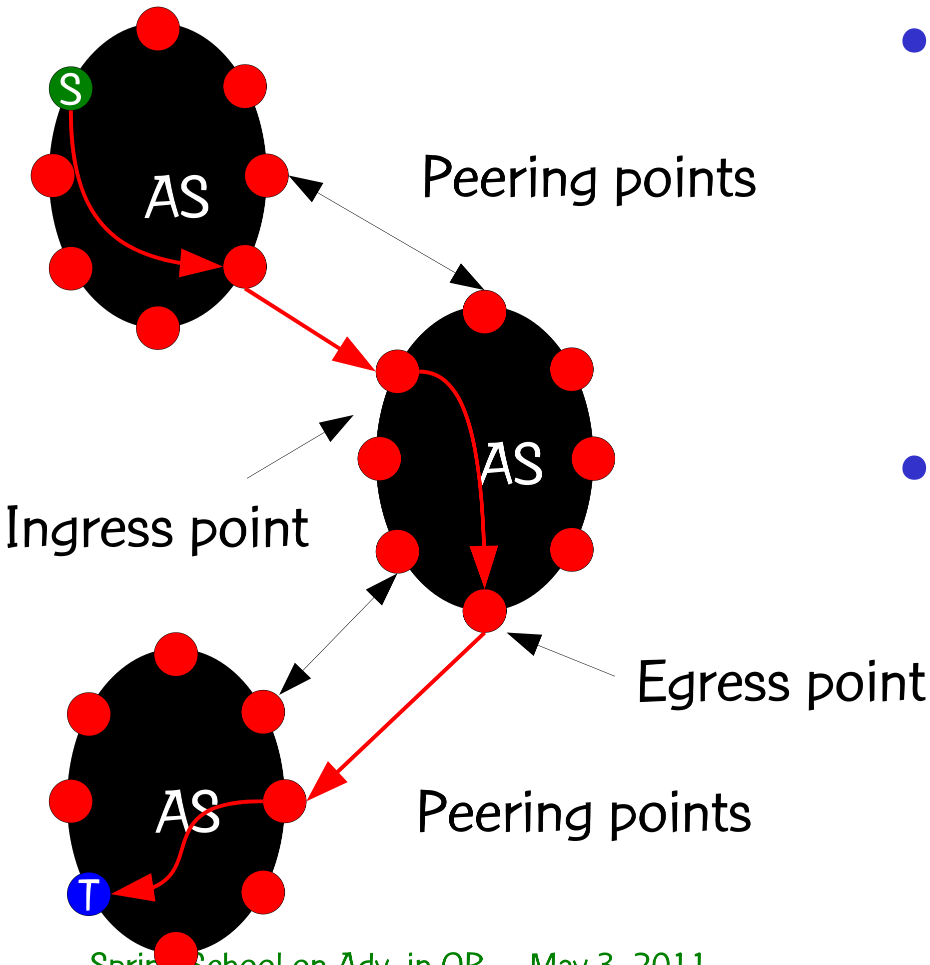
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.
- AS operators choose egress point and route in AS from ingress point to egress point.

IGP Routing

OSPF routing

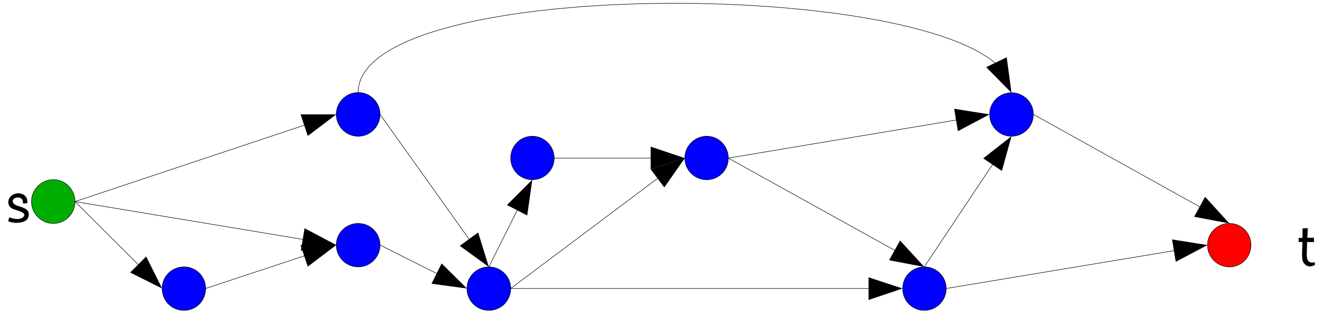
- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.

OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight $w(a)$ assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.

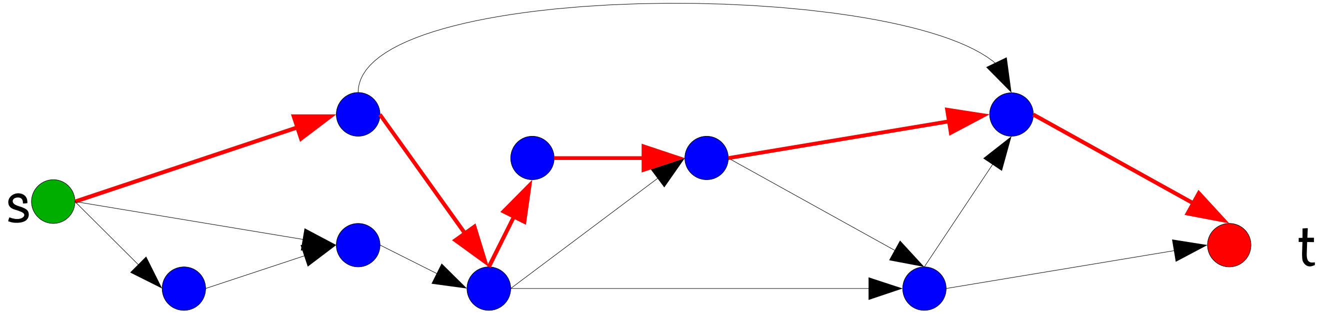
OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



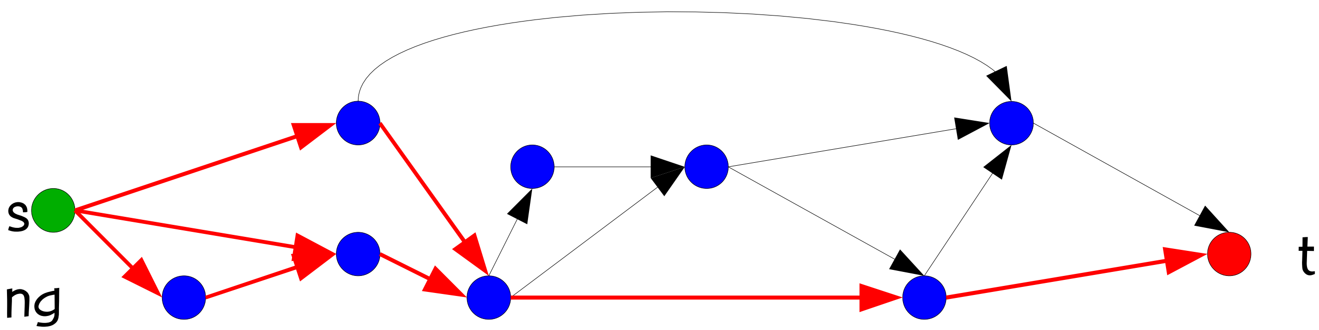
OSPF routing

- Given a network G = (N,A), where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



Traffic splitting

OSPF routing

- By setting OSPF weights appropriately, one can do traffic engineering, i.e. route traffic so as to optimize some objective (e.g. minimize congestion, maximize throughput, etc.).
- Some recent papers on this topic:
  - Fortz & Thorup (2000, 2004)
  - Ramakrishnan & Rodrigues (2001)
  - Sridharan, Guérin, & Diot (2002)
  - Fortz, Rexford, & Thorup (2002)
  - Ericsson, Resende, & Pardalos (2002)
  - Buriol, Resende, Ribeiro, & Thorup (2002, 2005)
  - Reis, Ritt, Buriol, & Resende (2011)

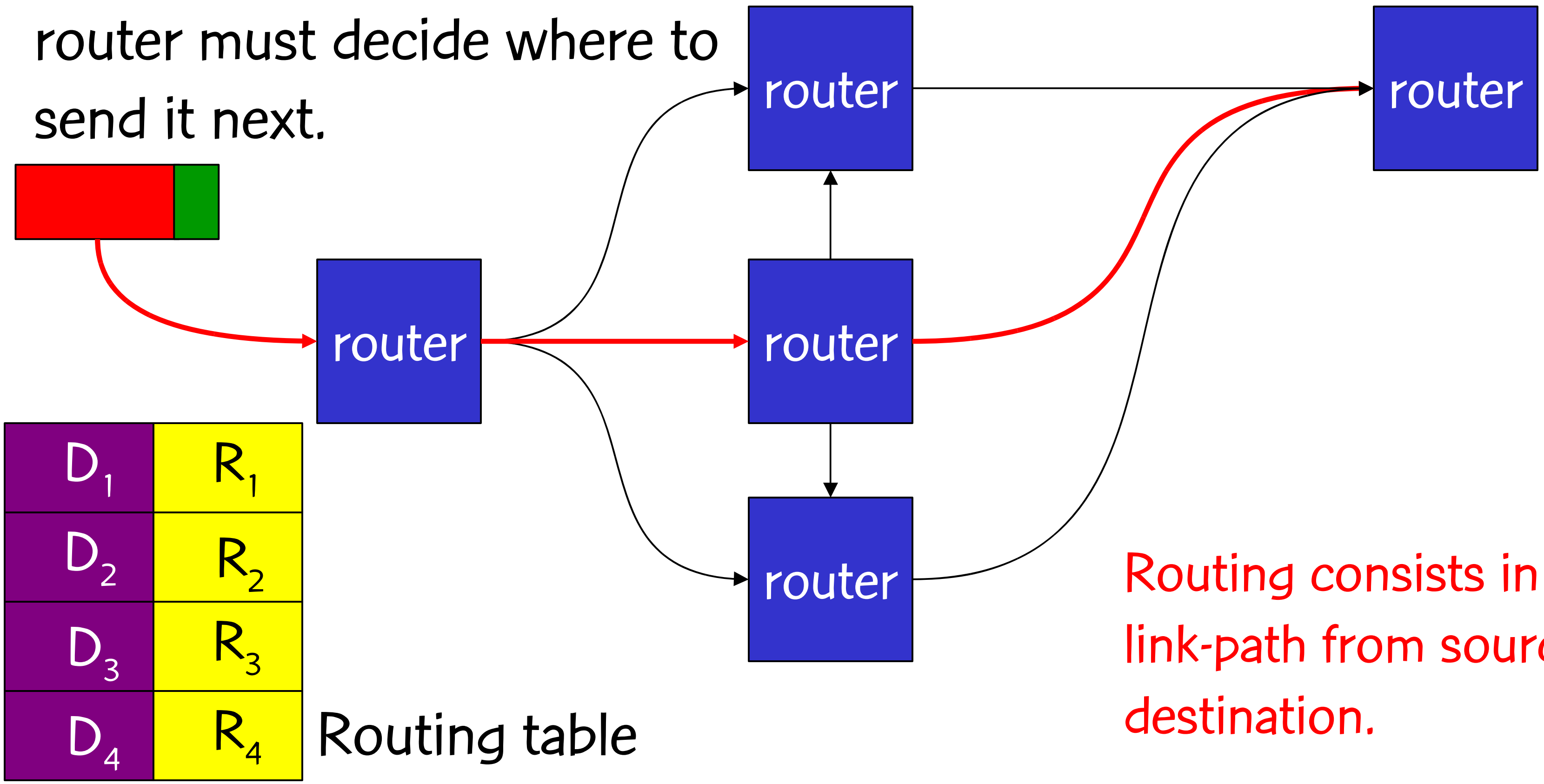
OSPF routing

- By setting OSPF weights appropriately, one can do traffic engineering, i.e. route traffic so as to optimize some objective (e.g. minimize congestion, maximize throughput, etc.).
- Some recent papers on this topic:
  - Fortz & Thorup (2000, 2004)
  - Ramakrishnan & Rodrigues (2001)
  - Sridharan, Guérin, & Diot (2002)
  - Fortz, Rexford, & Thorup (2002)
  - Ericsson, Resende, & Pardalos (2002)
  - Buriol, Resende, Ribeiro, & Thorup (2002, 2005)
  - Reis, Ritt, Buriol & Resende (2011)

Packet routing

When packet arrives at router, router must decide where to send it next.

Packet's final destination.



Routing table

Routing consists in finding a link-path from source to destination.

OSPF routing

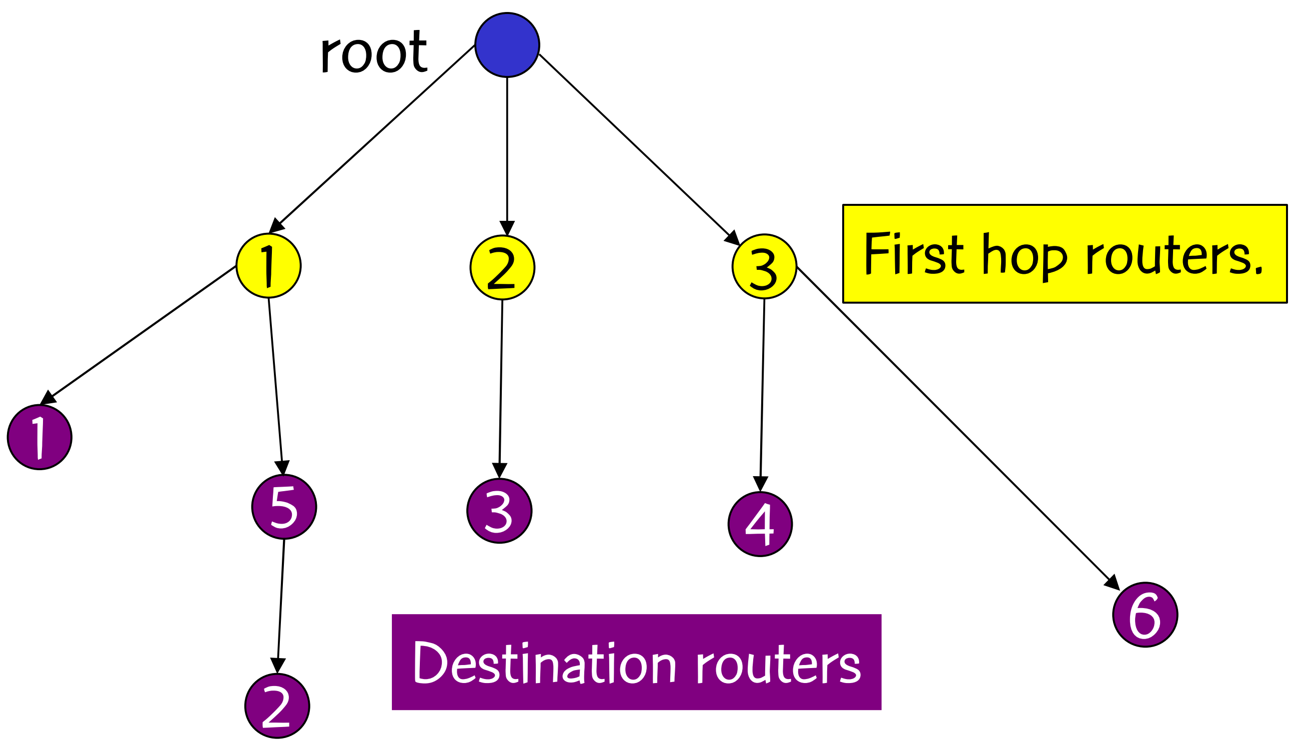
- Assign an integer weight $\in [1, w_{max}]$ to each link in AS. In general, $w_{max} = 65535 = 2^{16} - 1$.
- Each router computes tree of shortest weight paths to all other routers in the AS, with itself as the root, using Dijkstra's algorithm.

OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3

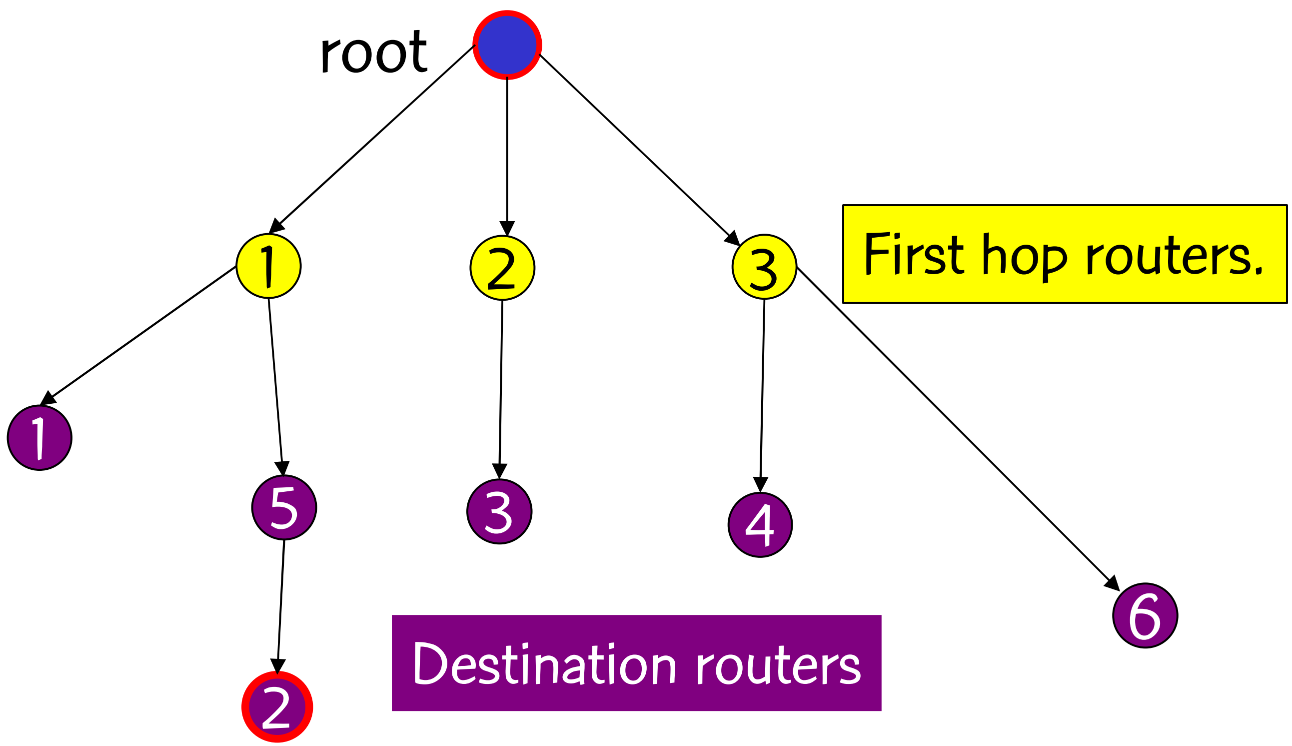


OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3

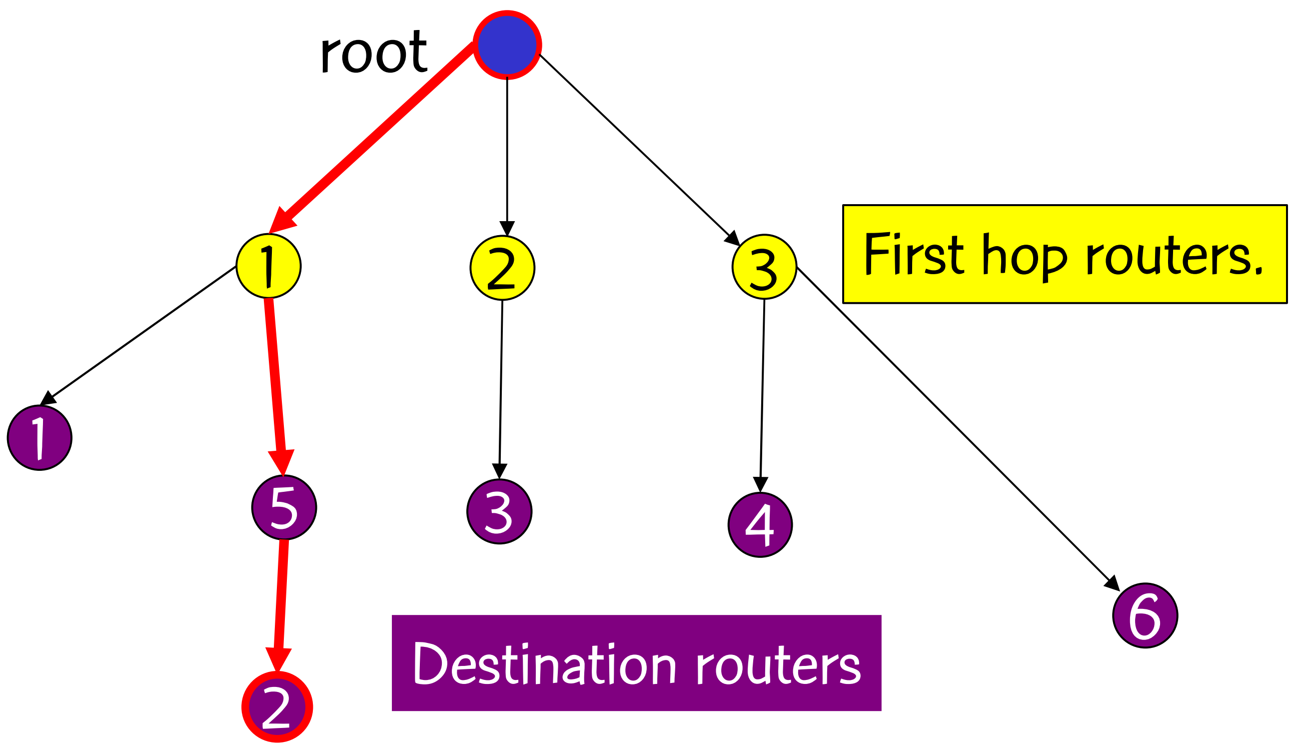


OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3

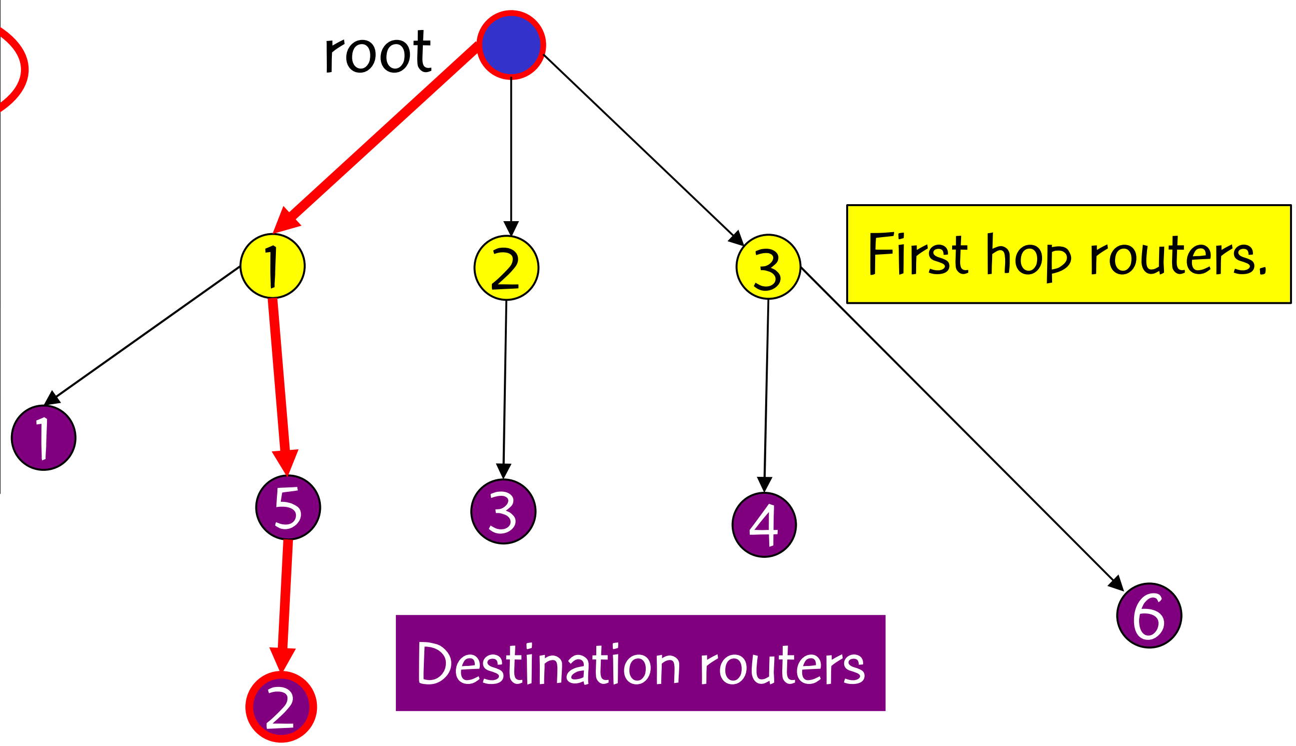


OSPF routing

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3

Routing table is filled
with first hop routers
for each possible destination.

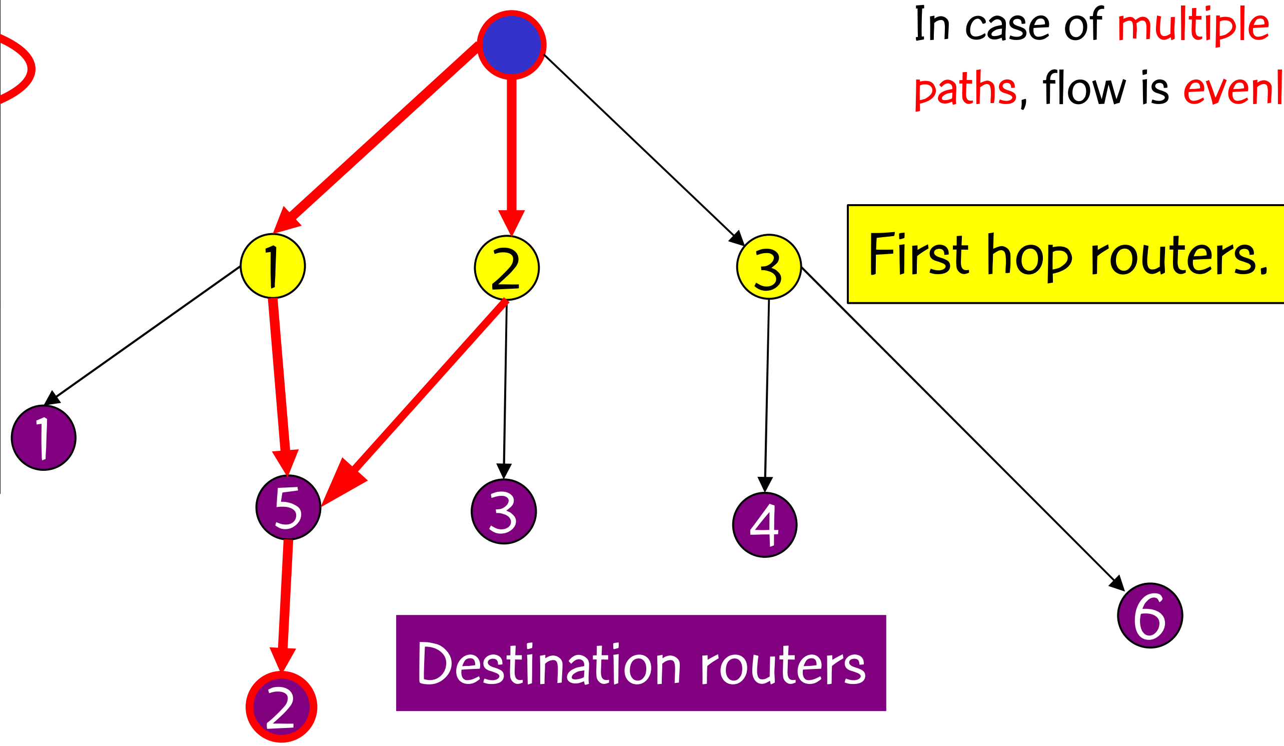


OSPF routing

Routing table is filled with first hop routers for each possible destination. In case of multiple shortest paths, flow is evenly split.

Routing table

D_1	R_1
D_2	R_1, R_2
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



OSPF weight setting

- OSPF weights are assigned by network operator.
  - CISCO assigns, by default, a weight proportional to the inverse of the link bandwidth (Inv Cap).
  - If all weights are unit, the weight of a path is the number of hops in the path.
- We propose two BRKGA to find good OSPF weights.

Minimization of congestion

- Consider the directed capacitated network $G = (N, A, c)$, where N are routers, A are links, and c_a is the capacity of link $a \in A$.
- We use the measure of Fortz & Thorup (2000) to compute congestion:

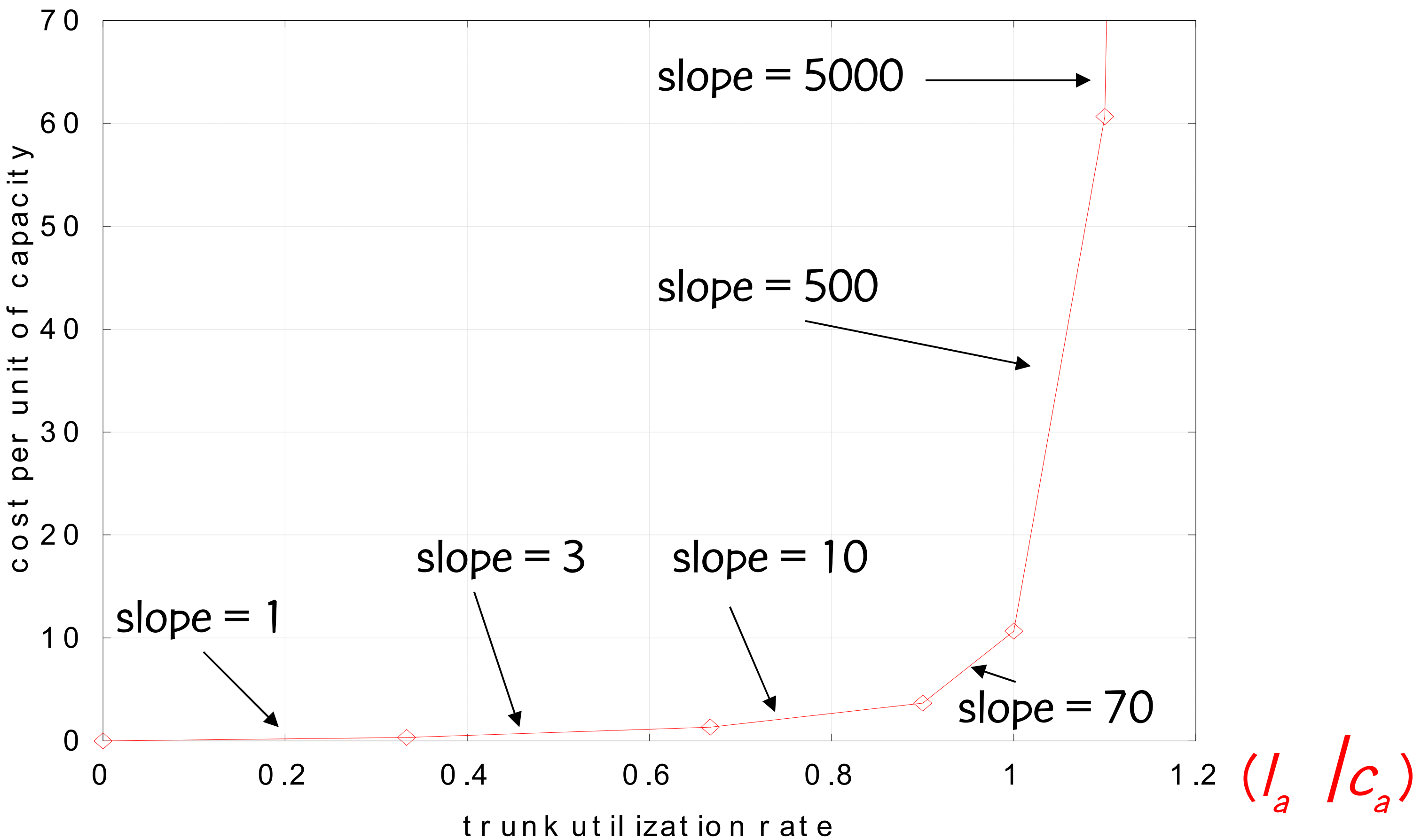
$$\Phi = \Phi_1(l_1) + \Phi_2(l_2) + \ldots + \Phi_{|A|}(l_{|A|})$$

where l_a is the load on link $a \in A$,

$\Phi_a(l_a)$ is piecewise linear and convex,

$\Phi_a(0) = 0$, for all $a \in A$.

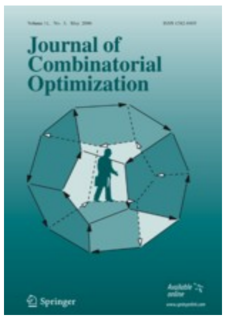
Piecewise linear and convex $\Phi_a(l_a)$ link congestion measure



OSPF weight setting problem

- Given a directed network $G = (N, A)$ with link capacities $c_a \in A$ and demand matrix $D = (d_{s,t})$ specifying a demand to be sent from node s to node t:
  - Assign weights $w_a \in [1, w_{max}]$ to each link $a \in A$, such that the objective function Φ is minimized when demand is routed according to the OSPF protocol.

BRKGA for OSPF routing in IP networks



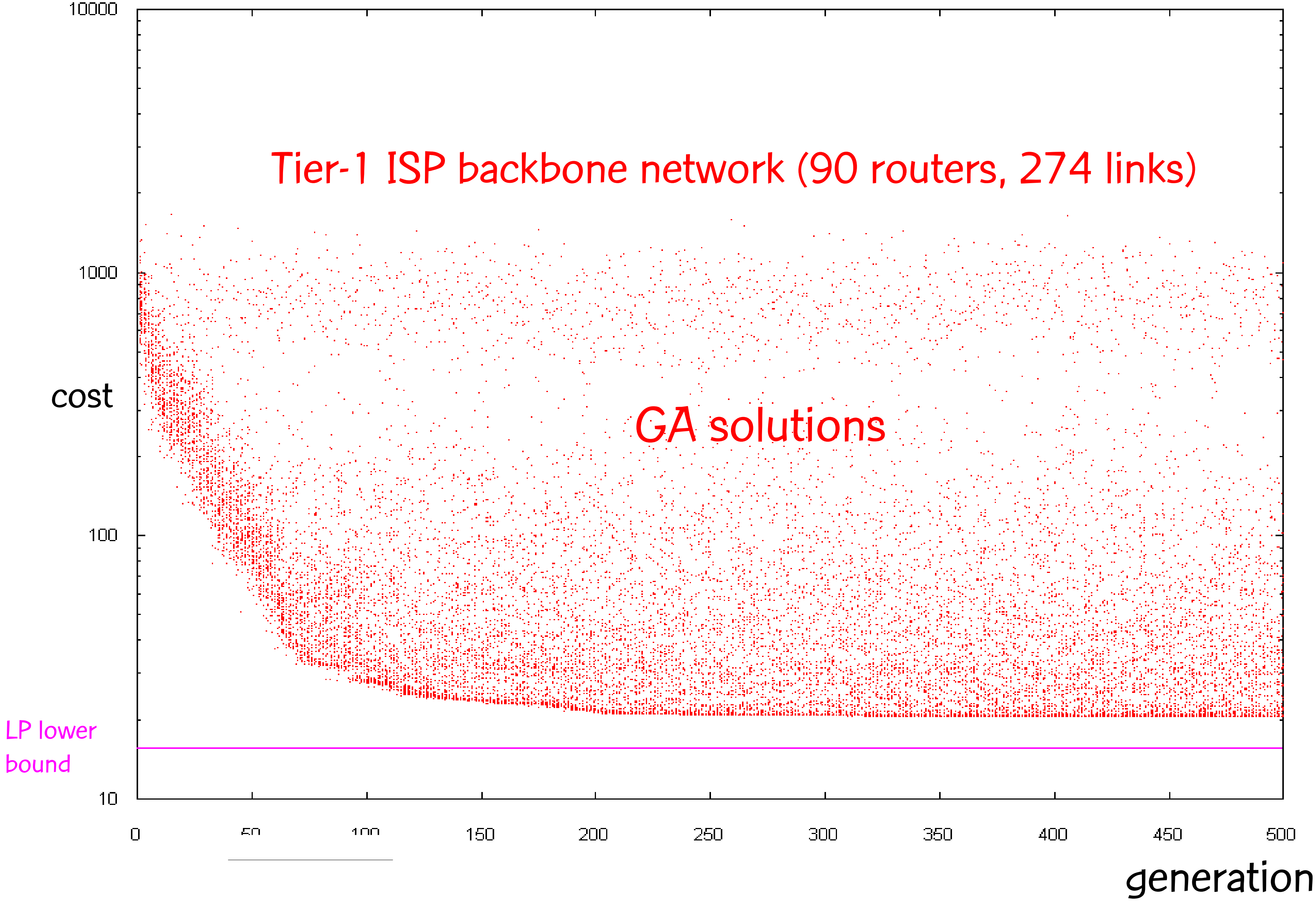
M. Ericsson, M.G.C.R., & P.M. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," J. of Combinatorial Optimization, vol. 6, pp. 299-333, 2002.

http://www2.research.att.com/~mgcr/doc/gaospf.pdf

BRKGA for OSPF routing in IP networks

Ericsson, R., & Pardalos (J. Comb. Opt., 2002)

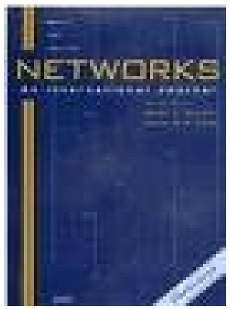
- Chromosome:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoder:
  - For i = 1,N: set $w(i) = \text{ceil}\,(X(i) \times w_{max})$
  - Compute shortest paths and route traffic according to OSPF.
  - Compute load on each link, compute link congestion, add up all link congestions to compute network congestion.


10000
1000
100
10
cost
Tier-1 ISP backbone network (90 routers, 274 links)
GA solutions
LP lower bound
0
50
100
150
200
250
300
350
400
450
500
generation

Tier-1 ISP backbone network (90 routers, 274 links)

Weight setting with GA permits a 50% increase in traffic volume w.r.t. weight setting with the Inverse Capacity rule.

Improved BRKGA for OSPF routing in IP networks



L.S. Buriol, M.G.C.R., C.C. Ribeiro, and M. Thorup, "A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing," Networks, vol. 46, no. 1, pp. 36-56, 2005.

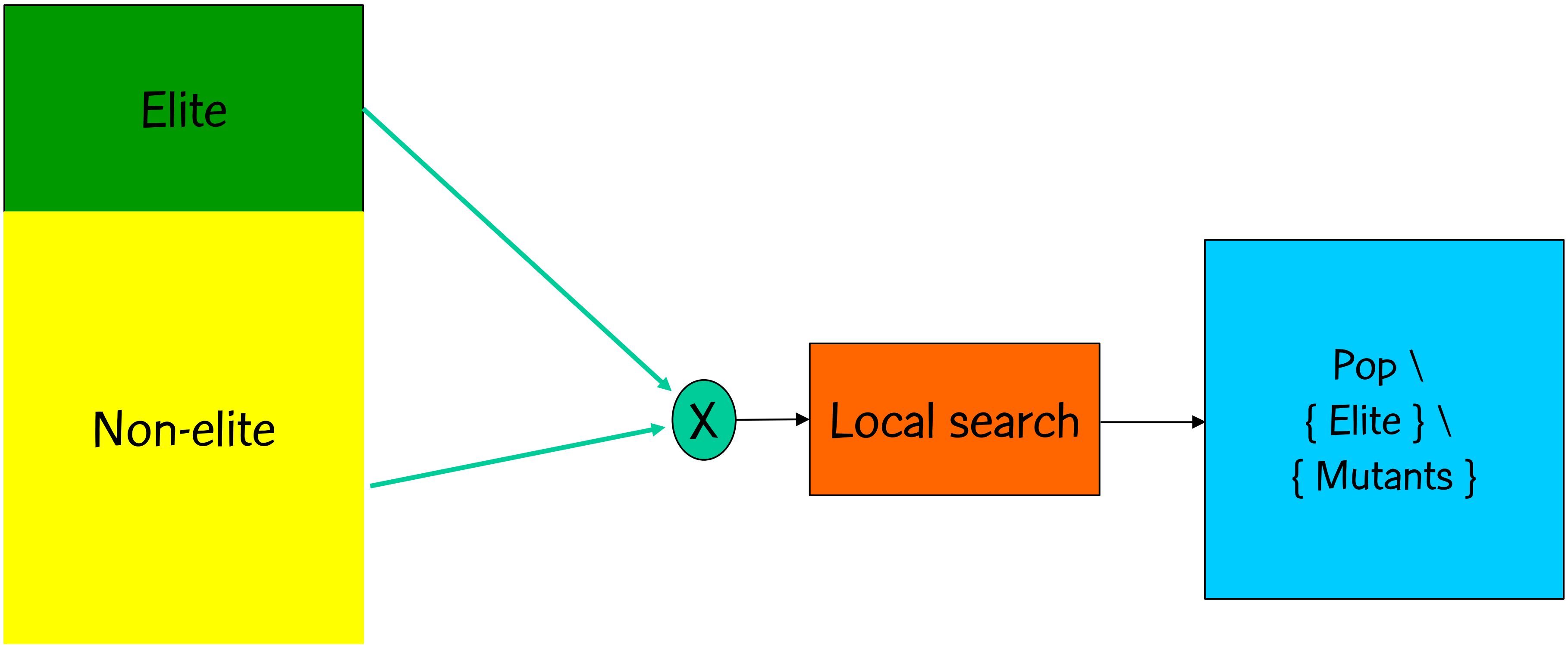
http://www2.research.att.com/~mgcr/doc/hgaospf.pdf

Improved BRKGA for OSPF routing in IP networks

Buriol, R., Ribeiro, and Thorup (Networks, 2005)

- Chromosome:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoder:
  - For i = 1,N: set $w(i) = \text{ceil}\,(\,X(i) \times w_{max}\,)$
  - Compute shortest paths and route traffic according to OSPF.
  - Compute load on each link, compute link congestion, add up all link congestions to compute network congestion.
  - Apply fast local search to improve weights.

Decoder has a local search phase



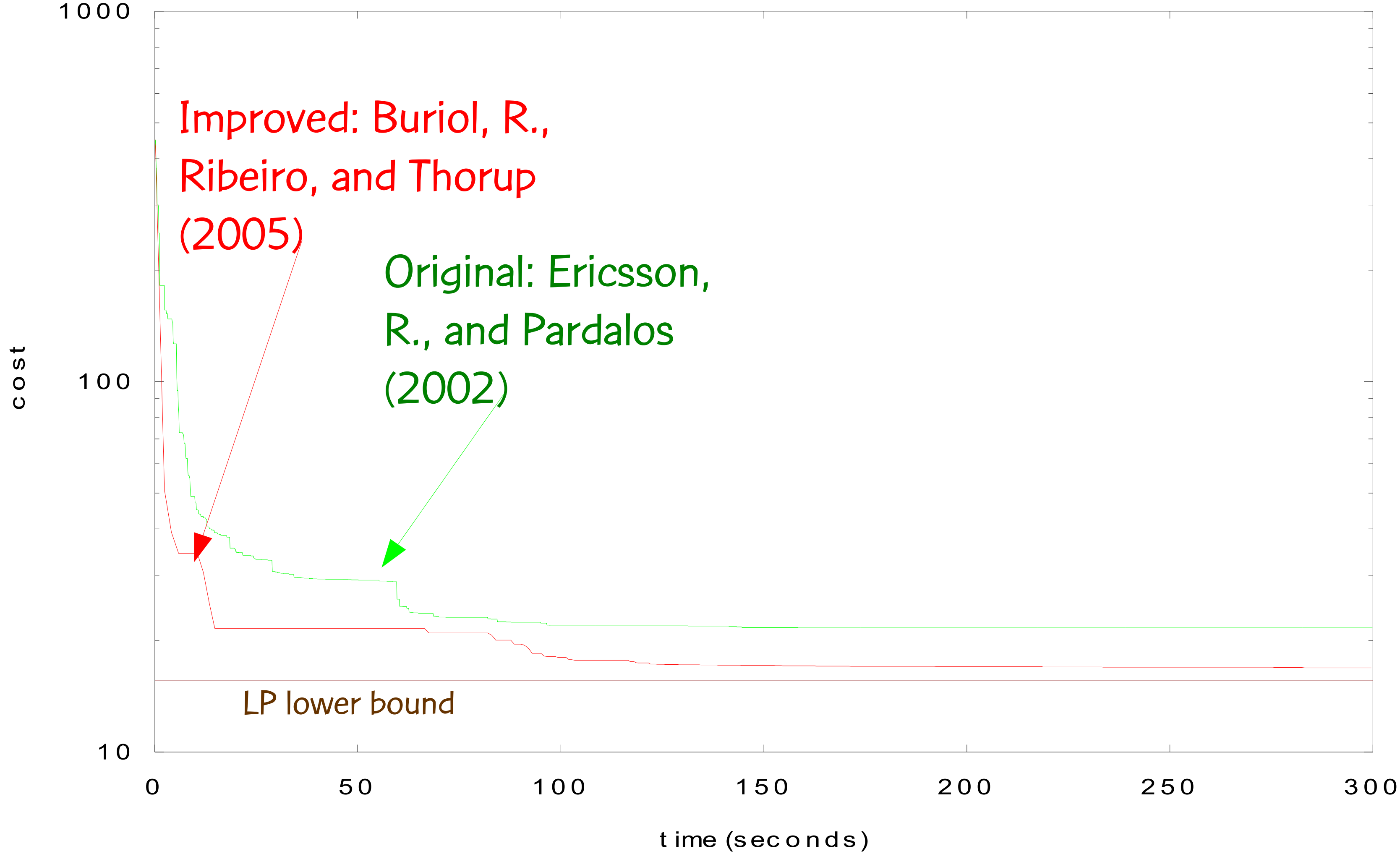
Fast local search

- Let A^* be the set of five arcs $a \in A$ having largest Φ_a values.
- Scan arcs $a \in A^*$ from largest to smallest Φ_a:
  - Increase arc weight, one unit at a time, in the range

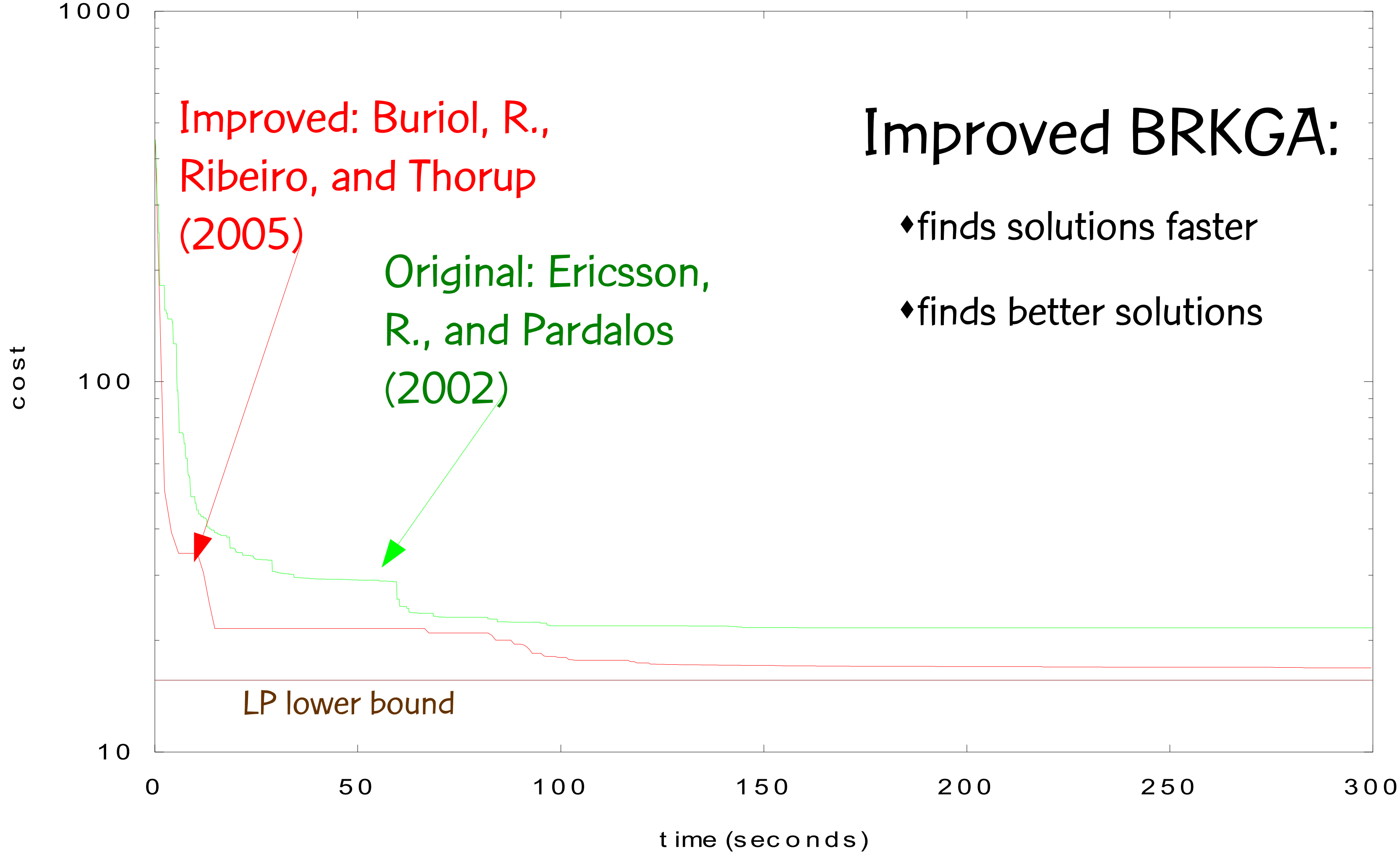
    $$[w_a, w_a + \lceil (w_{max} - w_a)/4 \rceil]$$

  - If total cost Φ is reduced, restart local search.

Effect of decoder with fast local search



Effect of decoder with fast local search



Improved BRKGA:

- finds solutions faster
- finds better solutions

DEFT routing in IP networks

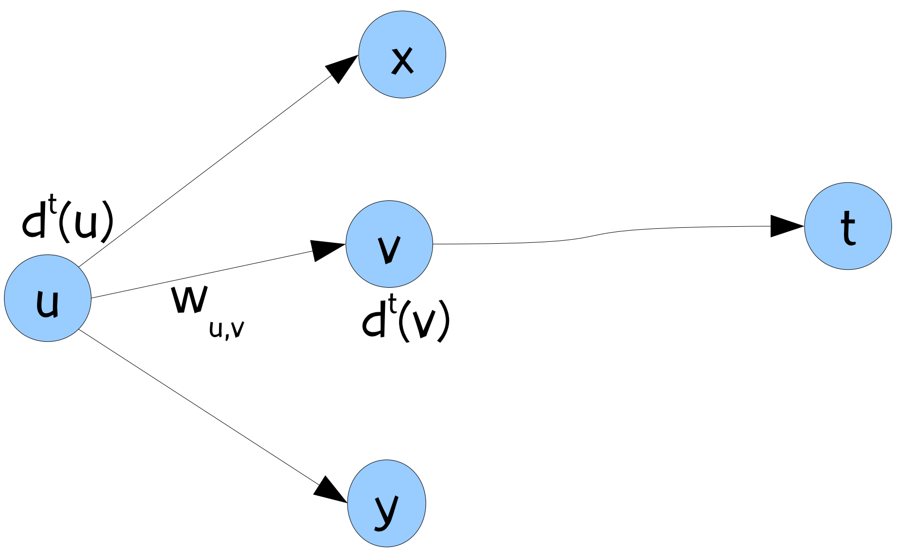
DEFT routing

- Proposed by Dahai Xu, Mung Chiang, and Jennifer Rexford, DEFT: Distributed Exponentially-weighted Flow spliTting, INFOCOM 2007
- Flow is routed on all links that lead to the destination. An exponential penalty is used to assign less flow to links that are on longer paths.

DEFT routing

- Consider each forward link (u,v) outgoing a given node u.
- Denote by $w_{u,v}$ the real-valued weight of link (u,v) and $d^t(u)$ as the distance of node u from target t.
- The gap $h^t(u,v)$ between u and v is calculated as:

$$h^t(u,v) = d^t(v) + w_{u,v} - d^t(u)$$



DEFT routing

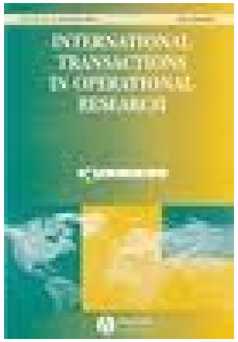
- Exponential function:

$$\text{if } d^t(u) > d^t(v) \text{ then } \Gamma[h^t(u,v)] = \exp[-h^t(u,v)],$$
$$\text{otherwise } \Gamma[h^t(u,v)] = 0$$

- The total flow $f^t(u)$ out of node u and destined to node t is split according to:

$$f^t(u,v) = f^t(u)\, \Gamma[h^t(u,v)] / \Sigma_{(u,j)\in E} \Gamma[h^t(u,j)]$$

BRKGA for DEFT weight setting



R. Reis, M. Ritt, L.S. Buriol, and M.G.C.R., "A biased random-key genetic algorithm for OSPF and DEFT routing to minimize network congestion," International Transactions in Operational Research, vol. 18, pp. 401-423, 2011.

Tech report version:

http://www.research.att.com/~mgcr/doc/brkga-deft-ospf.pdf

BRKGA for DEFT weight setting

Reis, Ritt, Buriol, and R. (ITOR, 2011)

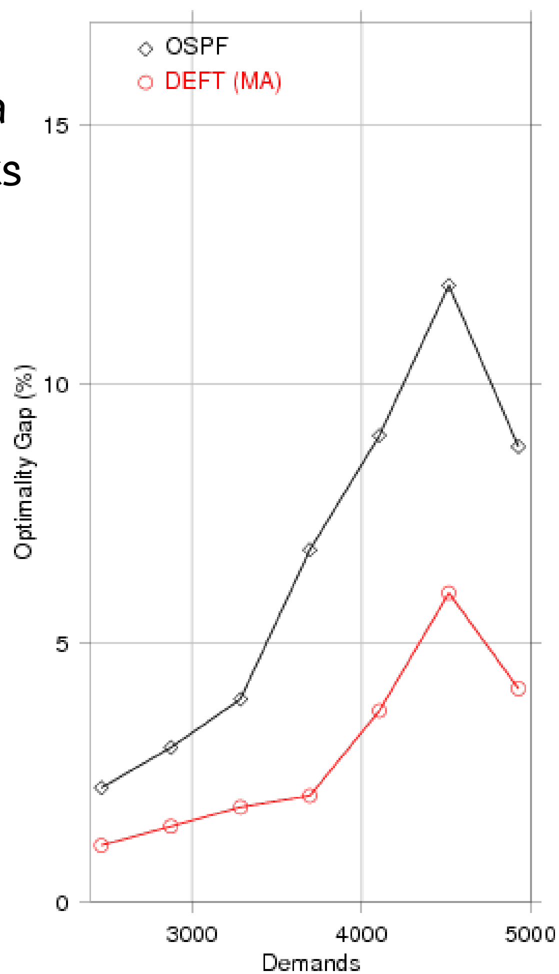
- Similar to improved BRKGA for OSPF weight setting
  - Decoder with fast local search
- Decoder is the only difference
  - weights are set as in improved BRKGA for OSPF
  - shortest paths and gaps are determined, penalties defined, and flows computed
  - fast local search is adapted for DEFT

Experiments

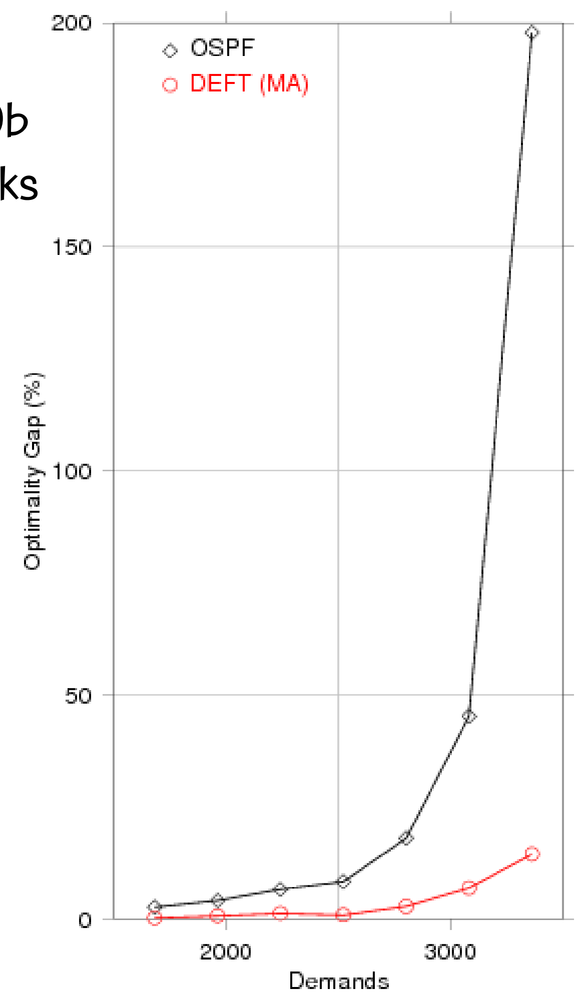
- 6 instances with 7 different demand matrices
- Results are averages over 3 random seeds
- Stopping criterion: 2000 generations or 500 generations without improvement
- Each run takes about 1 hour on a SGI Altrix (1.6Ghz Itanium 2 processor)

OSPF vs. DEFT
Two level hierarchy with 50 nodes

hier50a
148 links

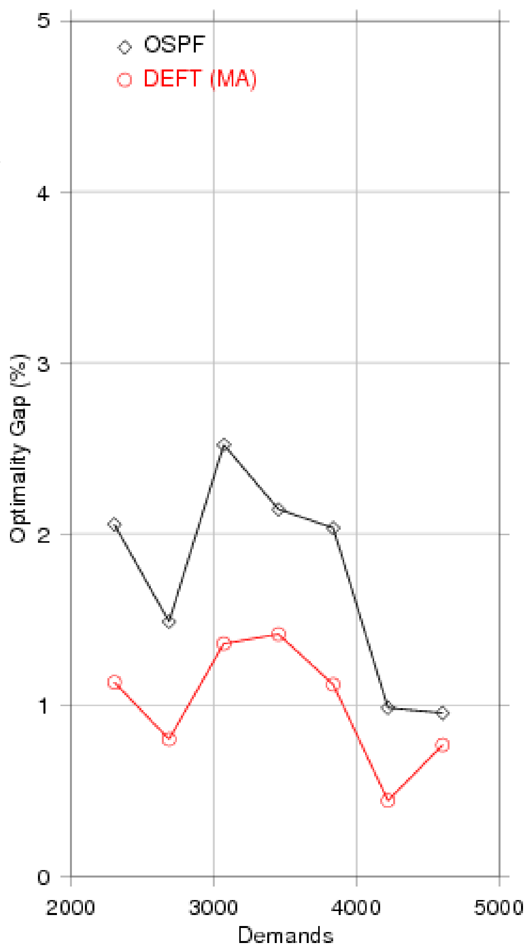


Hier50b
212 links

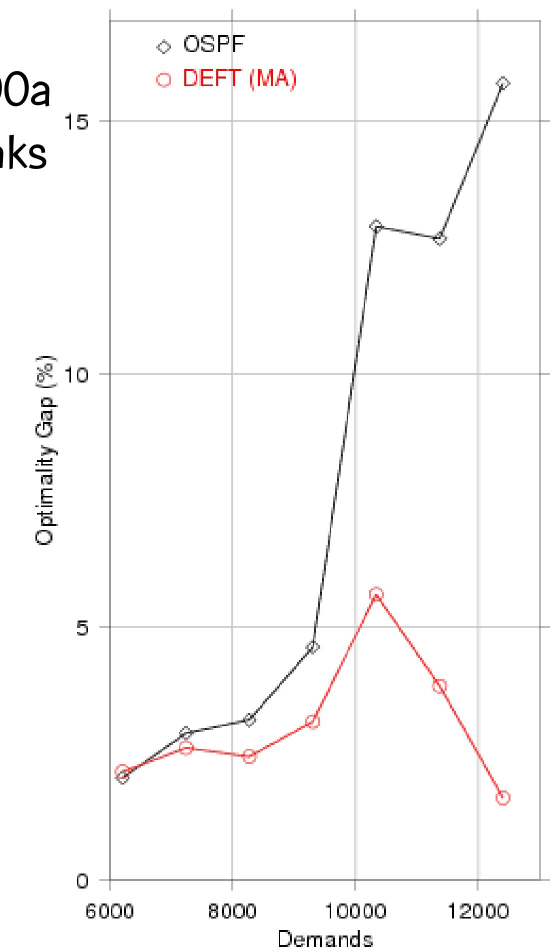


OSPF vs. DEFT
Two level hierarchy with 100 nodes

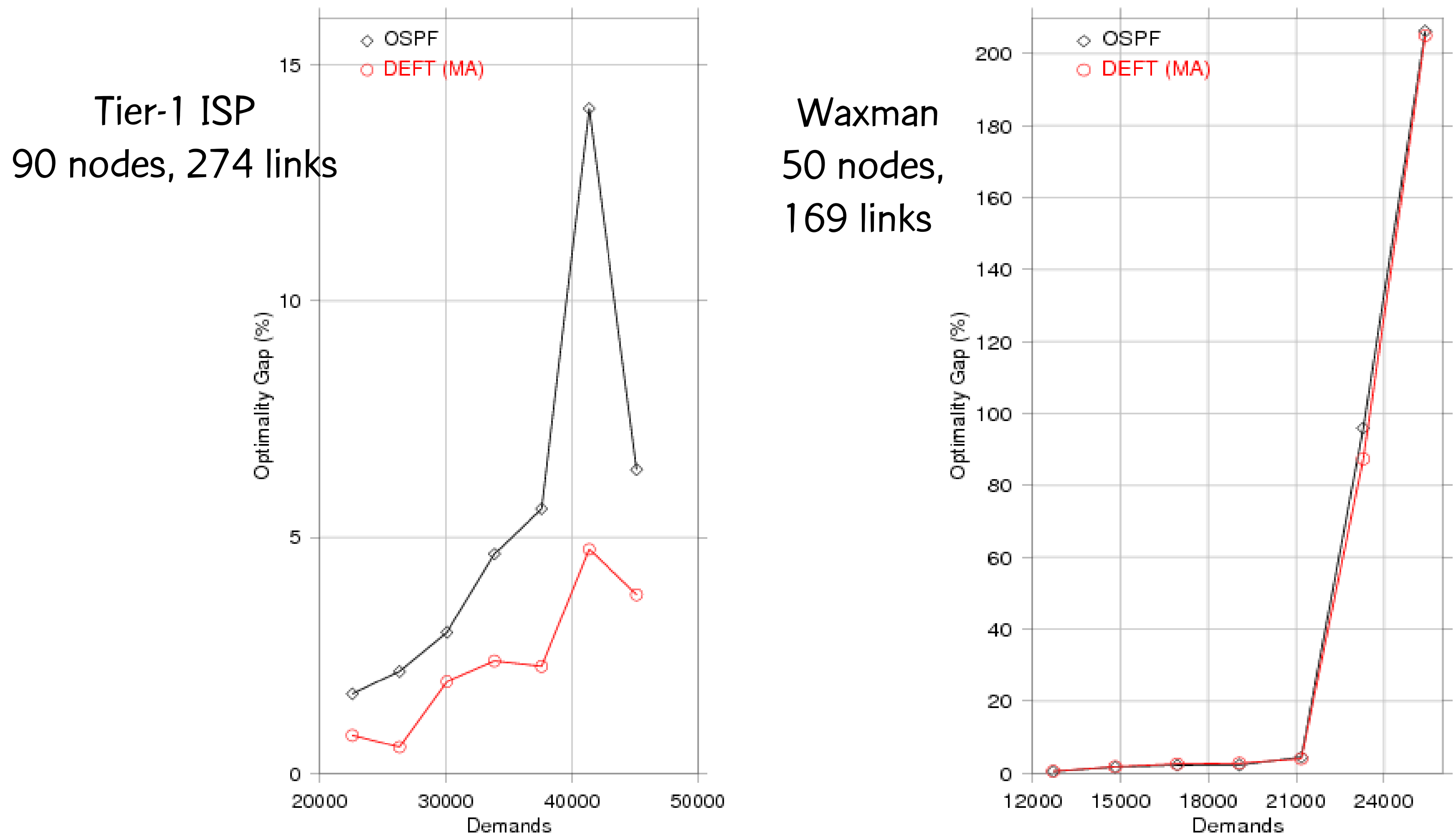
Hier100
280 links



Hier100a
360 links



OSPF vs. DEFT


Tier-1 ISP
90 nodes, 274 links
OSPF
DEFT (MA)
Optimality Gap (%)
0
5
10
15
20000
30000
40000
50000
Demands
Waxman
50 nodes,
169 links
OSPF
DEFT (MA)
Optimality Gap (%)
0
20
40
60
80
100
120
140
160
180
200
12000
15000
18000
21000
24000
Demands

Survivable IP network design

Survivable IP network design

L.S. Buriol, M.G.C.R., and M. Thorup, "Survivable IP network design with OSPF routing," Networks, vol. 49, pp. 51-64, 2007.

Tech report version:

http://www.research.att.com/~mgcr/doc/gamult.pdf

Survivable IP network design

Buriol, R., & Thorup (Networks, 2007)

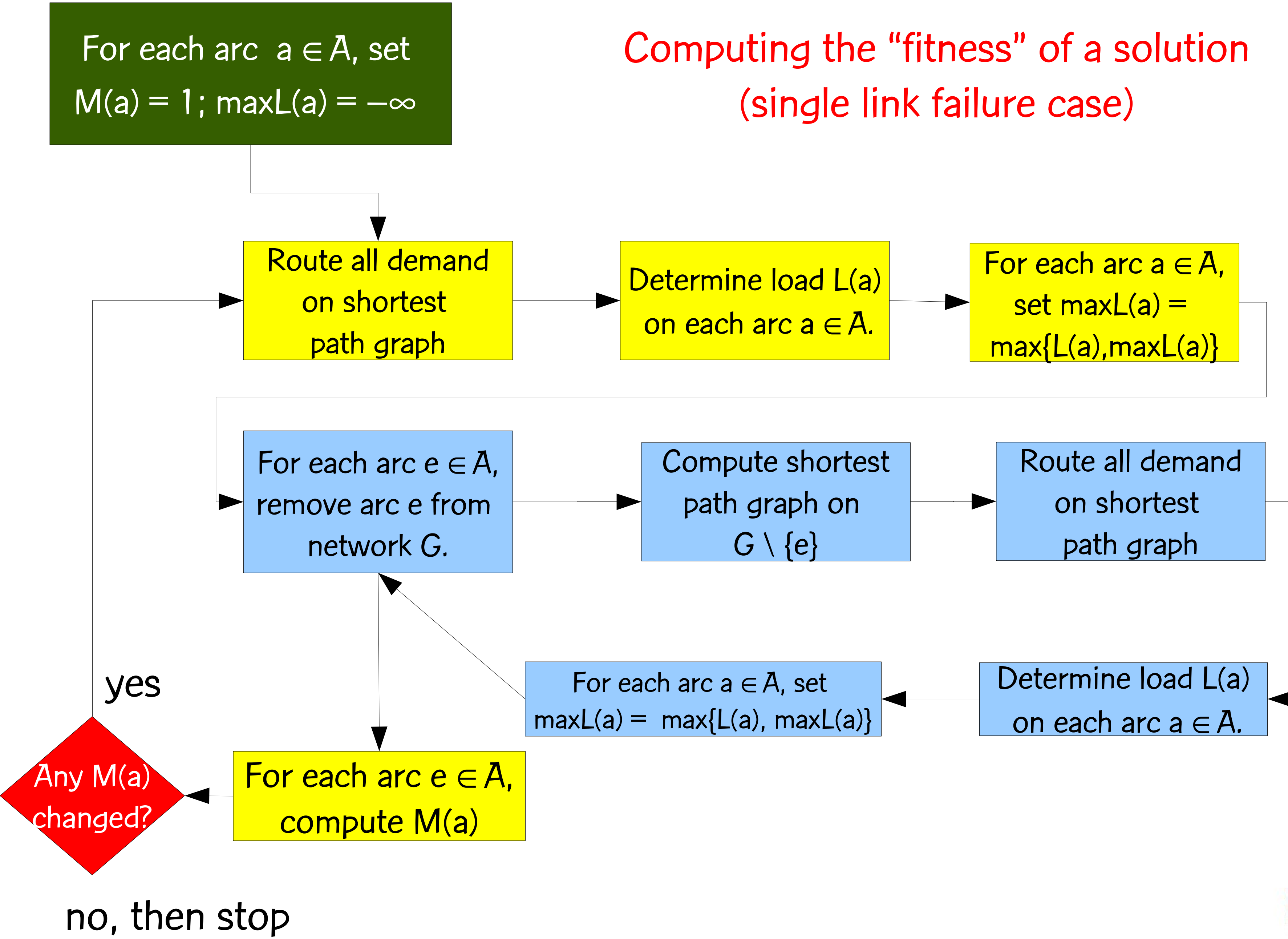
- Given
  - directed graph $G = (N,A)$, where N is the set of routers, A is the set of potential arcs where capacity can be installed,
  - a demand matrix D that for each pair $(s,t) \in N \times N$, specifies the demand $D(s,t)$ between s and t,
  - a cost $K(a)$ to lay fiber on arc a
  - a capacity increment C for the fiber.

- Determine
  - OSPF weight $w(a)$ to assign to each arc $a \in A$,
  - which arcs should be used to deploy fiber and how many units (multiplicities) $M(a)$ of capacity C should be installed on each arc $a \in A$,
- such that all the demand can be routed on the network even when any single arc fails.
- Min total design cost $= \sum_{a \in A} M(a) \times K(a)$.

Survivable IP network design

- Chromosome:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoder:
  - For i = 1,N: set $w(i) = \text{ceil}(X(i) \times w_{max})$
  - For each failure mode: route demand according to OSPF and for each arc $a \in A$ determine the load on arc a.
  - For each arc $a \in A$, determine the multiplicity M(a) using the maximum load for that arc over all failure modes.
  - Network design cost = $\sum_{a \in A} M(a) \times K(a)$

Computing the "fitness" of a solution (single link failure case)



at&t
Your world. Delivered.

Composite-link design

- In Buriol, Resende, and Thorup (2006)
  - links were all of the same type,
  - only the link multiplicity had to be determined.
- Now consider composite links. Given a load L(a) on arc a, we can compose several different link types that sum up to the needed capacity $c(a) \geq L(a)$:
  - $c(a) = \sum_{t \text{ used in arc } a} M(t) \times \gamma(t)$, where
  - M(t) is the multiplicity of link type t
  - $\gamma(t)$ is the capacity of link type t

Composite-link design

- In Buriol, Resende, and Thorup (2006)
  - links were all of the same type,
  - only the link multiplicity had to be determined.
- Now consider composite links. Given a load $L(a)$ on arc a, we can compose several different link types that sum up to the needed capacity $c(a) \geq L(a)$:
  - $c(a) = \sum_{t \text{ used in arc } a} M(t) \times \gamma(t)$, where
  - $M(t)$ is the multiplicity of link type t
  - $\gamma(t)$ is the capacity of link type t

Composite-link design

- Link types = { 1, 2, ..., T }
- Capacities = { c(1), c(2), ..., c(T) } : c(i) < c(i+1)
- Prices / unit length = { p(1), p(2), ..., p(T) }: p(i) < p(i+1)
- Assumptions:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$, i.e. price per unit of capacity is smaller for links with greater capacity
  - $c(i) = \alpha \times c(i-1)$, for $\alpha \in N$, $\alpha > 1$, i.e. capacities are multiples of each other by powers of α

Composite-link design

- Link types = $\{ 1, 2, \ldots, T \}$
- Capacities = $\{ c(1), c(2), \ldots, c(T) \} : c(i) < c(i+1)$
- Prices / unit length = $\{ p(1), p(2), \ldots, p(T) \}: p(i) < p(i+1)$
- Assumptions:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$, i.e. price per unit of capacity is smaller for links with greater capacity
  - $c(i) = \alpha \times c(i-1)$, for $\alpha \in N$, $\alpha > 1$, i.e. capacities are multiples of each other by powers of α

Composite-link design

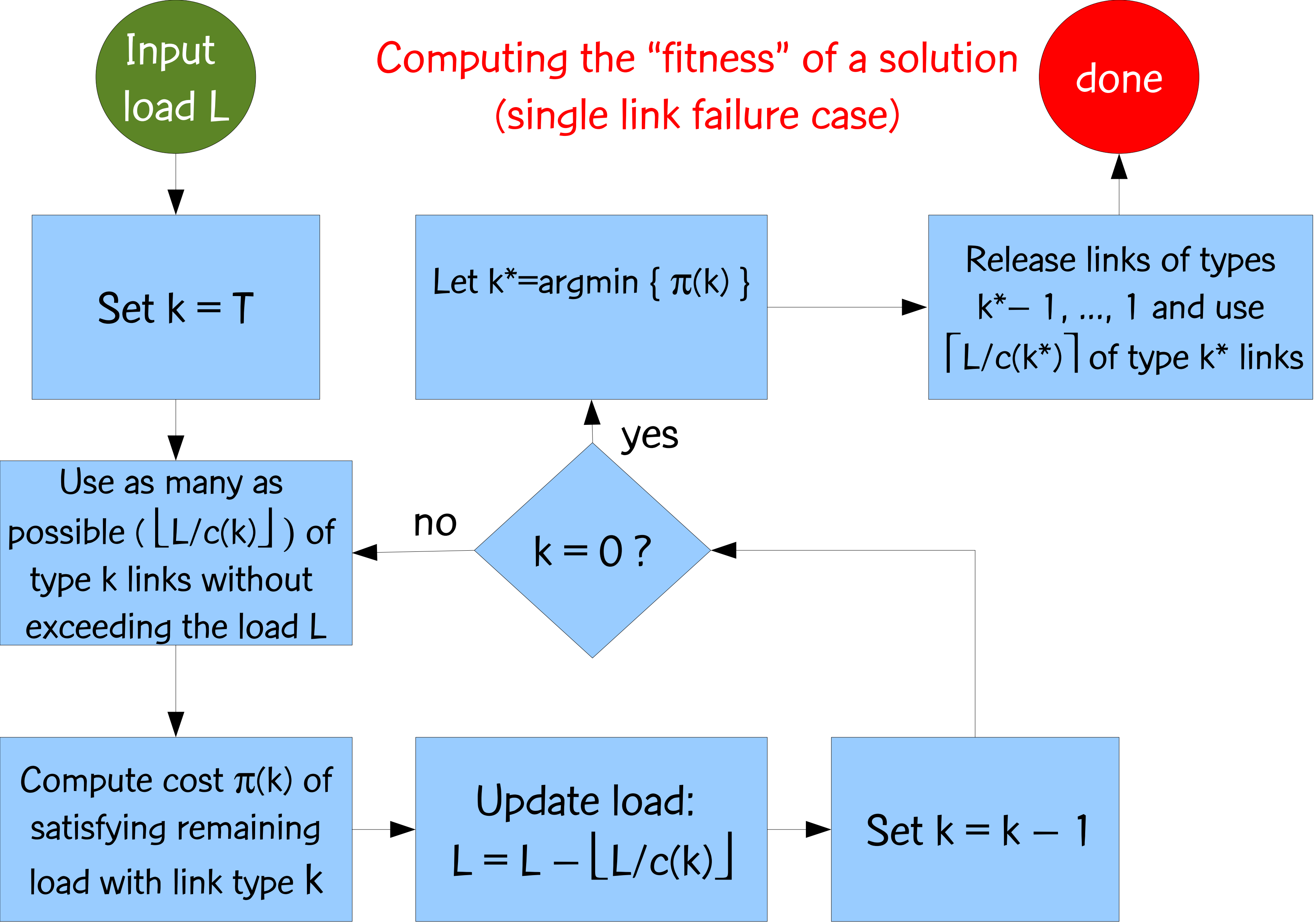
- Link types = { 1, 2, ..., T }
- Capacities = { c(1), c(2), ..., c(T) } : c(i) < c(i+1)
- Prices / unit length = { p(1), p(2), ..., p(T) }: p(i) < p(i+1)
- Assumptions:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$: economies of scale
  - $c(i) = \alpha \times c(i-1)$, for $\alpha \in N$, $\alpha > 1$, e.g. $c(OC192) = 4 \times c(OC48)$; $c(OC48) = 4 \times c(OC12)$; $c(OC12) = 4 \times c(OC3)$;

OC3	OC12	OC48	OC192	
155 Mb/s	622 Mb/s	2.5 Gb/s	10 Gb/s	$\alpha = 4$

Survivable composite link IP network design

- Chromosome:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoder:
  - For i = 1,N: set $w(i) = \text{ceil}\ (X(i) \times w_{max})$
  - For each failure mode: route demand according to OSPF and for each arc $i \in A$ determine the load on arc i.
  - For each arc $i \in A$, determine the multiplicity M(t,i) for each link type t using the maximum load for that arc over all failure modes.
  - Network design cost = $\sum_{i \in A} \sum_{t \text{ used in arc } i} M(t,i) \times p(t)$

Computing the “fitness” of a solution (single link failure case)



at&t
Your world. Delivered.

An example

- Load on link: $L = 1090$
- 3 link types: $T = \{ 1, 2, 3 \}$
- Capacities: $C = \{ 1, 4, 16 \}$
- Prices: $P = \{ 50, 90, 100 \}$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$

- $P(1) < P(2) < P(3)$
- $C(3) = 4\ C(2)$
- $C(2) = 4\ C(1)$
- $P/C = \{ 50, 22.5, 6.25 \}$
- $P(1)/C(1) > P(2)/C(2) > P(3)/C(3)$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$

- $K = |T| = 3$
- $L = 1090$
- $M(3) = \text{floor}[1090/16] =$ 68 links of type 3
- $\pi(3) = 6900$
- $L = 1090 - 1088 = 2$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$
- $\pi(3) = 6900$; $M(3) = 68$

- $K = |T| = 2$
- $L = 2$
- $M(2) = \text{floor}[2/4] = 0$ links of type 2
- $\pi(2) = 90$
- $L = 2 - 0 = 2$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$
- $\pi(3) = 6900$; $M(3) = 68$
- $\pi(2) = 90$; $M(2) = 0$
- $K = |T| = 1$
- $L = 2$
- $M(1) = \text{floor}[2/1] = 2$ links of type 1
- $\pi(1) = 100$
- $L = 2 - 2 = 0$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$
- Use $M(3) = 68$ and $M(2) = \text{ceil}\,(2/4) = 1$
- and $M(1) = 0$

- $\pi(3) = 6900$
- $\pi(2) = 90$ ::: minimum
- $\pi(1) = 100$

An example

- $L = 1090$
- $T = \{ 1, 2, 3 \}$
- $C = \{ 1, 4, 16 \}$
- $P = \{ 50, 90, 100 \}$

- $\pi(3) = 6900$
- $\pi(2) = 90$ ::: minimum
- $\pi(1) = 100$

- Use $M(3) = 68$ and $M(2) = \text{ceil}\ (2/4) = 1$
- and $M(1) = 0$

Indeed, cost of $M=(0,1,68) = 6990$ is less than cost of $M=(1,0,68) = 7000$

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1); \quad c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1); \quad p(3)/c(3) = 0.90\ p(1)/c(1$
- All four heuristics tested. Min cost k types was testec for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- c(2) = 4 c(1); c(3) = 16 c(1)
- p(2)/c(2) = 0.95 p(1)/c(1); p(3)/c(3) = 0.90 p(1)/c(1
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- c(2) = 4 c(1);   c(3) = 16 c(1)
- p(2)/c(2) = 0.95 p(1)/c(1);   p(3)/c(3) = 0.90 p(1)/c(1
- All four heuristics tested. Min cost k types was testec for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

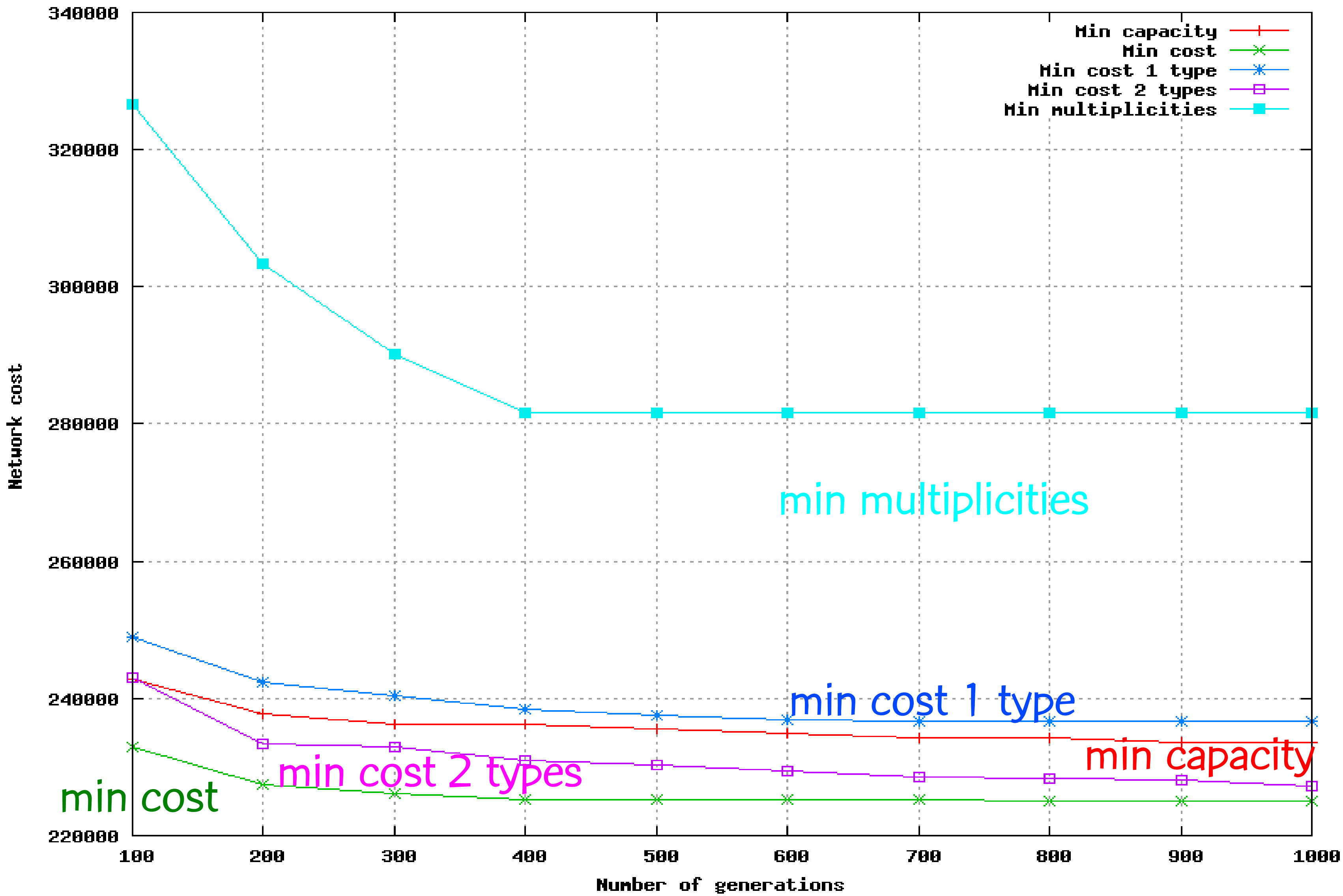
- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1$
- All four heuristics tested. Min cost k types was testec for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1);\quad c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1);\quad p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.



Min capacity
Min cost
Min cost 1 type
Min cost 2 types
Min multiplicities
340000
320000
300000
280000
260000
240000
220000
Network cost
100
200
300
400
500
600
700
800
900
1000
Number of generations
min multiplicities
min cost 1 type
min capacity
min cost 2 types
min cost

Three-layer metropolitan network design problem

Summary

- Three-layer metropolitan network design problem
- Biased random-key genetic algorithms (BRKGAs)
- BRKGA for 3-layer metro network design
- Implementation details
- An example of metropolitan network design by BRKGA
- Concluding remarks

Problem data

- Graph representing network
  - Set of nodes: central offices and demand points
  - Set of edges: (i, j) where i,j are nodes
    - Loops (i,i) are allowed
    - Parallel edges may exist
    - Two types: FIBER (1 GigE and 16 GigE) and ROADM (reconfigurable optical add-drop multiplexor)
- Matrix of peer-to-peer traffic
- Vectors of traffic to and from VPLS-PE (backbone)

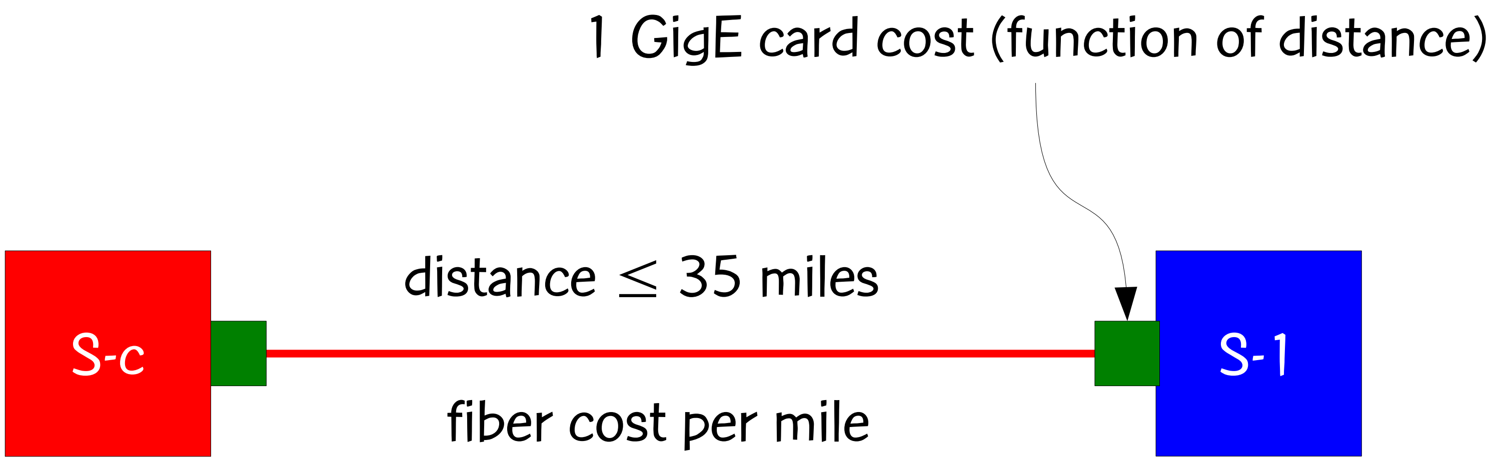
Problem data

- Previously deployed equipment
  - S-c: switch deployed on customer premises; connects to a small Ethernet switch (S-0) or medium Ethernet switch (S-1) via simple path
  - VPLS-PE (Virtual Private LAN Service – Provider Edge): gateway to IP common backbone

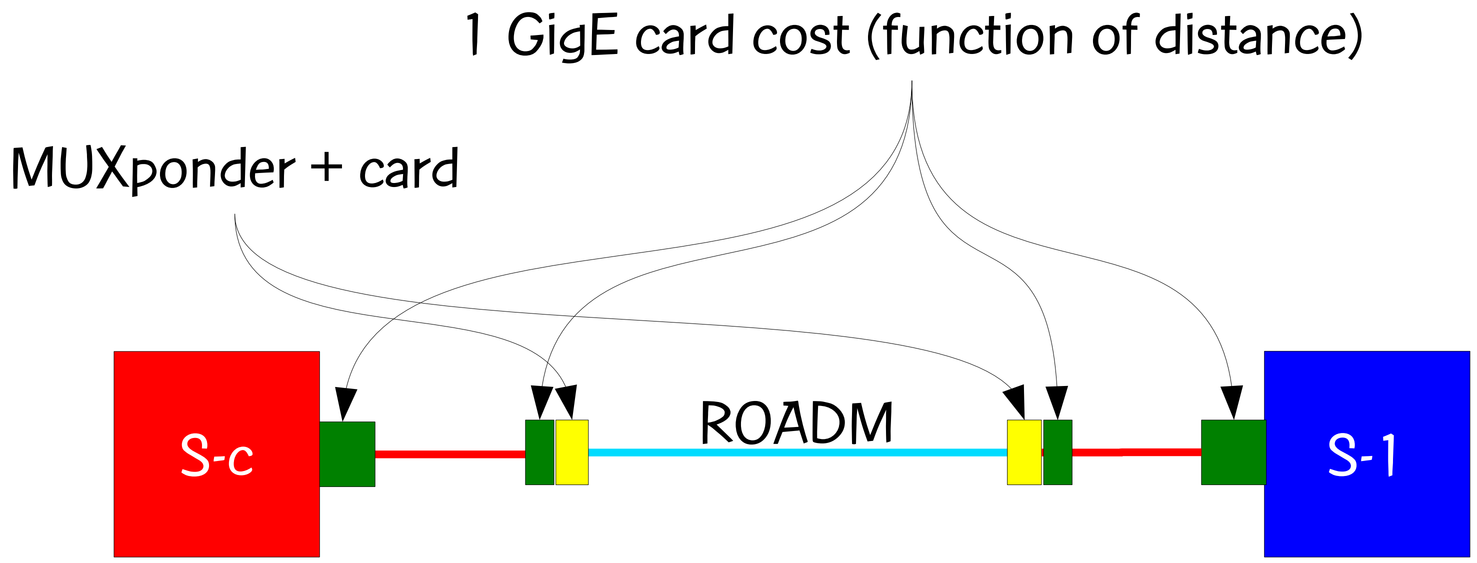
Equipment to be deployed

- S-0: aggregates up to 19 S-c switches
  - Connects to an S-1 via two node/edge disjoint paths
- S-1: aggregates up to 360 S-c and S-0 switches
  - Connects to a pair of S-2 Ethernet switches via node/edge disjoint paths
  - Two models of S-1 Ethernet switches
- S-2: aggregates up to 14 S-1 Ethernet switches
  - Connects to at least two other S-2s via disjoint paths
  - Two models of S-2 Ethernet switches

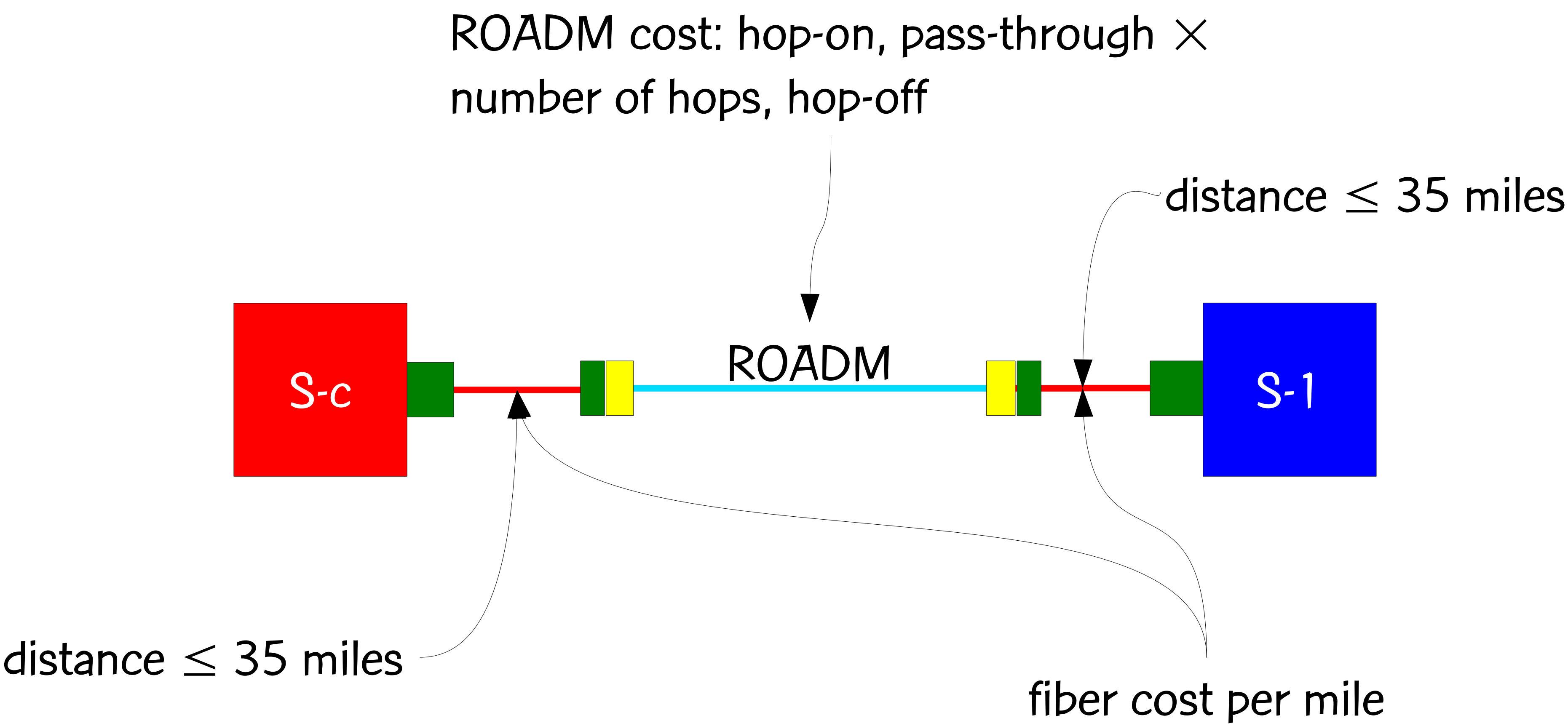
Connection cost: S-c to S-1 on fiber



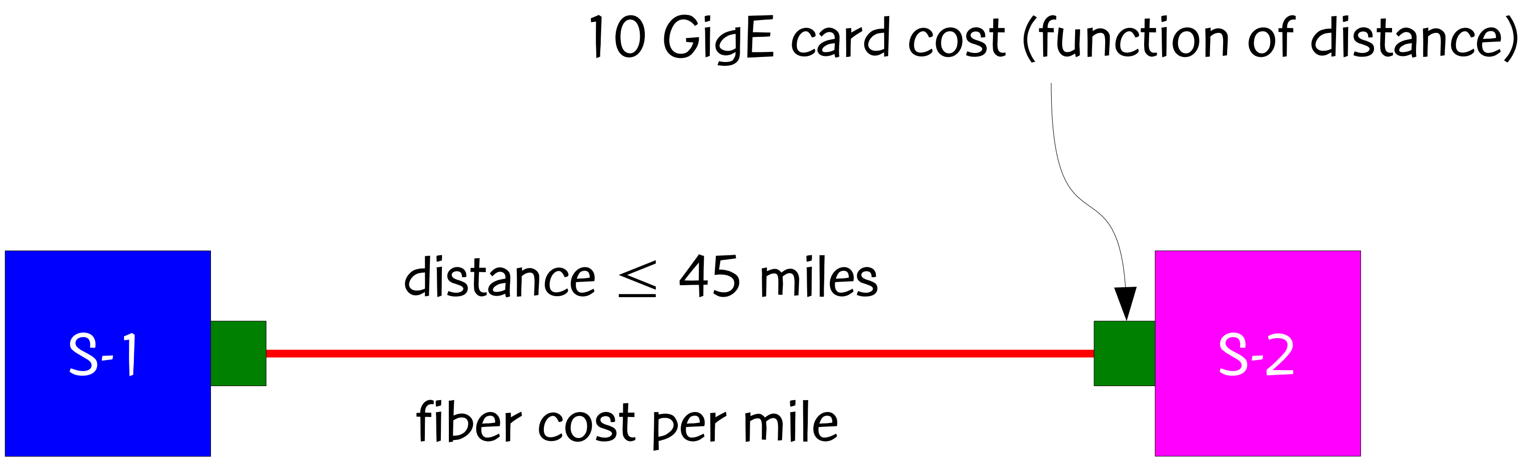
Connection cost: S-c to S-1 on fiber/ROADM



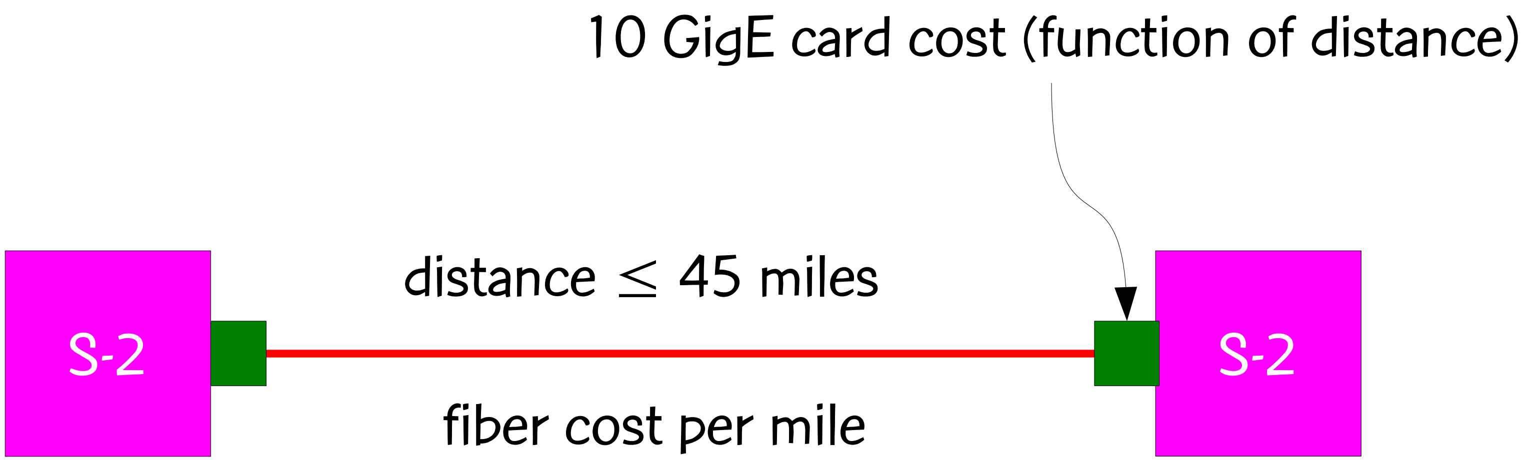
Connection cost: S-c to S-1 on fiber/ROADM



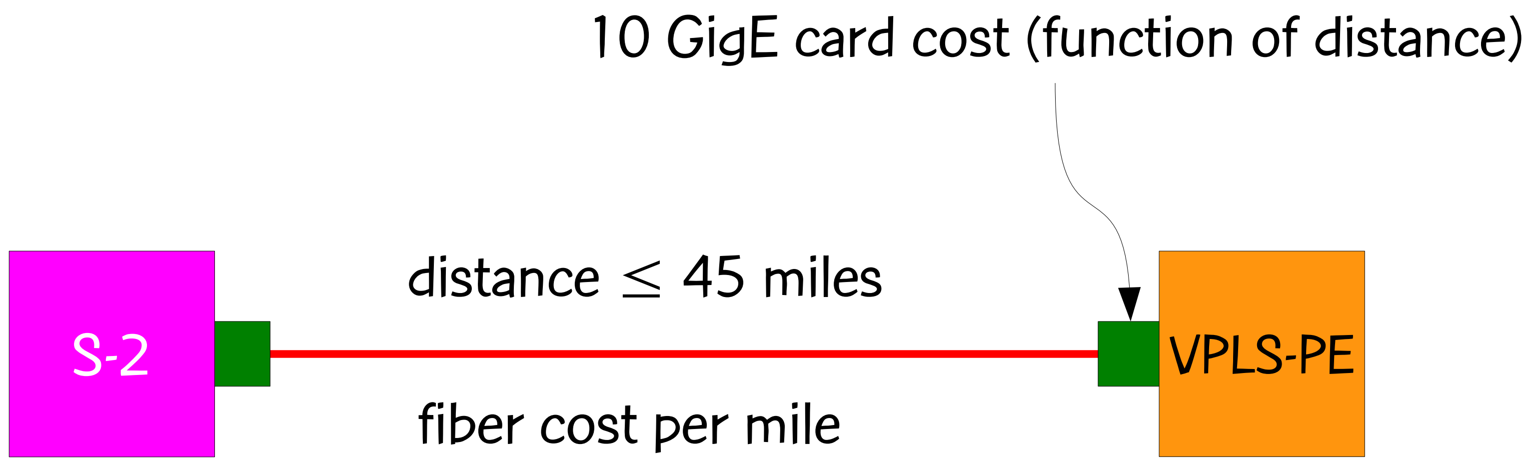
Connection cost: S-1 to S-2 on fiber



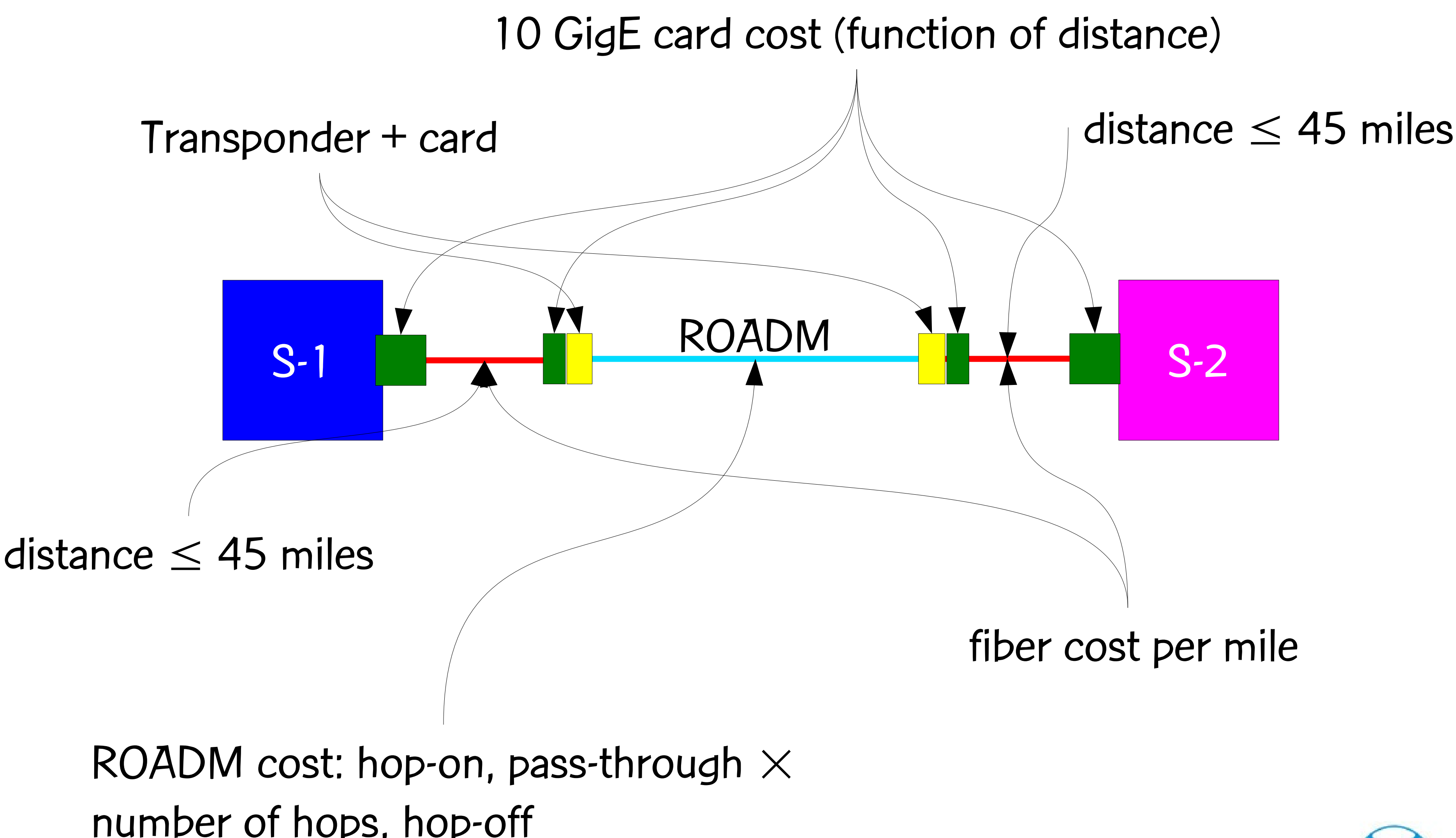
Connection cost: S-2 to S-2 on fiber



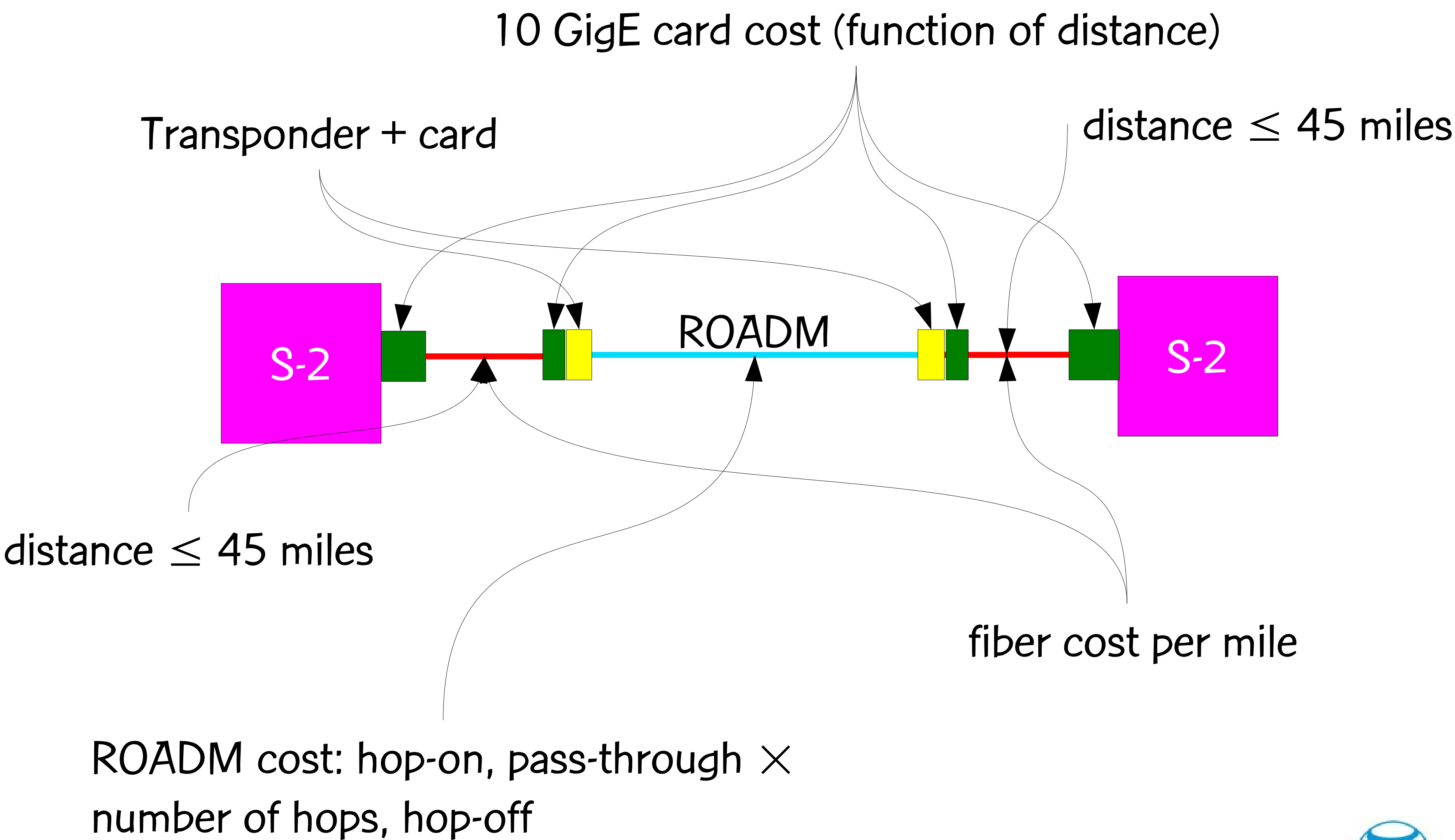
Connection cost: S-2 to VPLS-PE on fiber



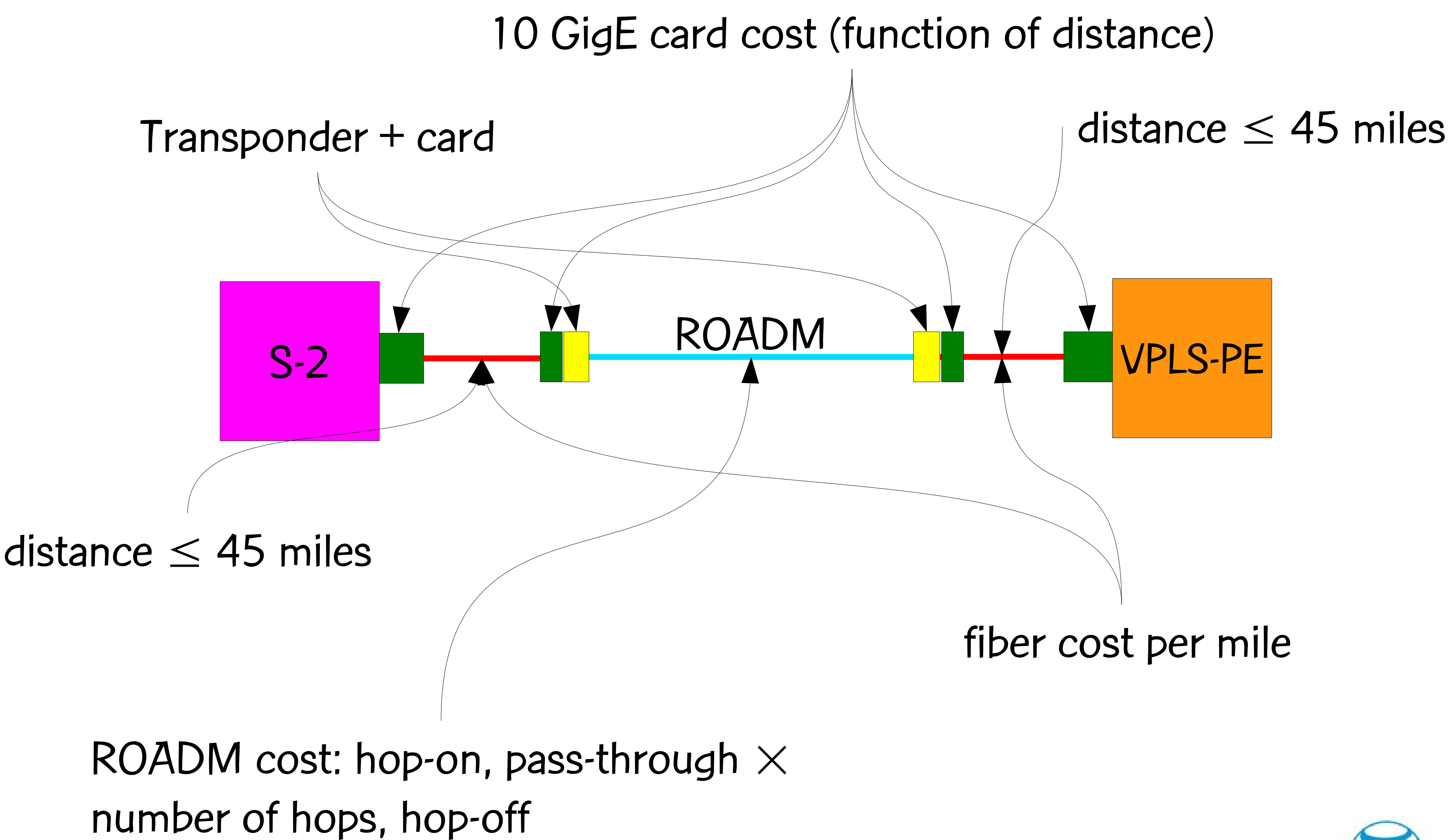
Connection cost: S-1 to S-2 on fiber/ROADM



Connection cost: S-2 to S-2 on fiber/ROADM



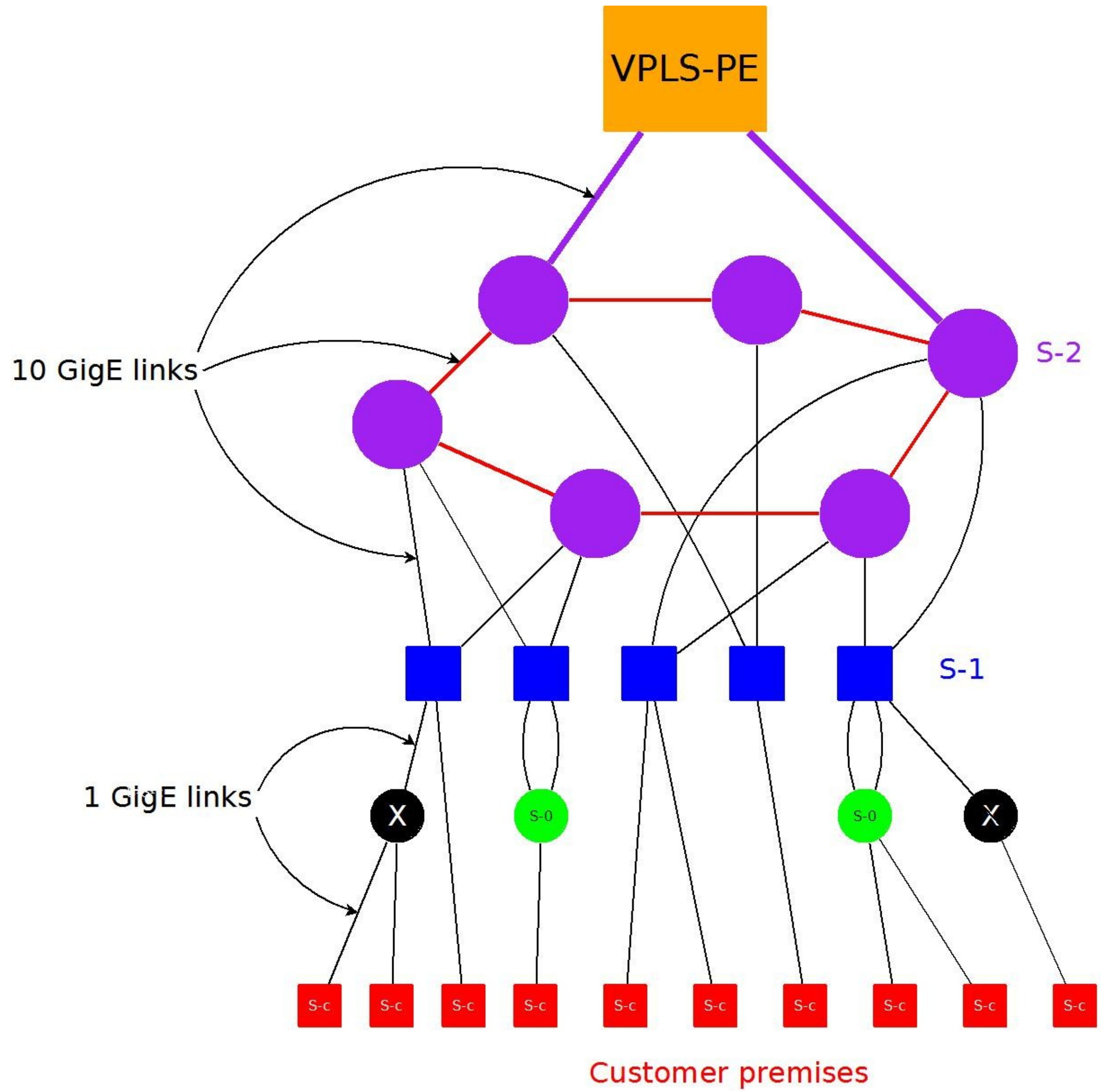
Connection cost: S-2 to VPLS-PE on fiber/ROADM



Target topology

Determine:

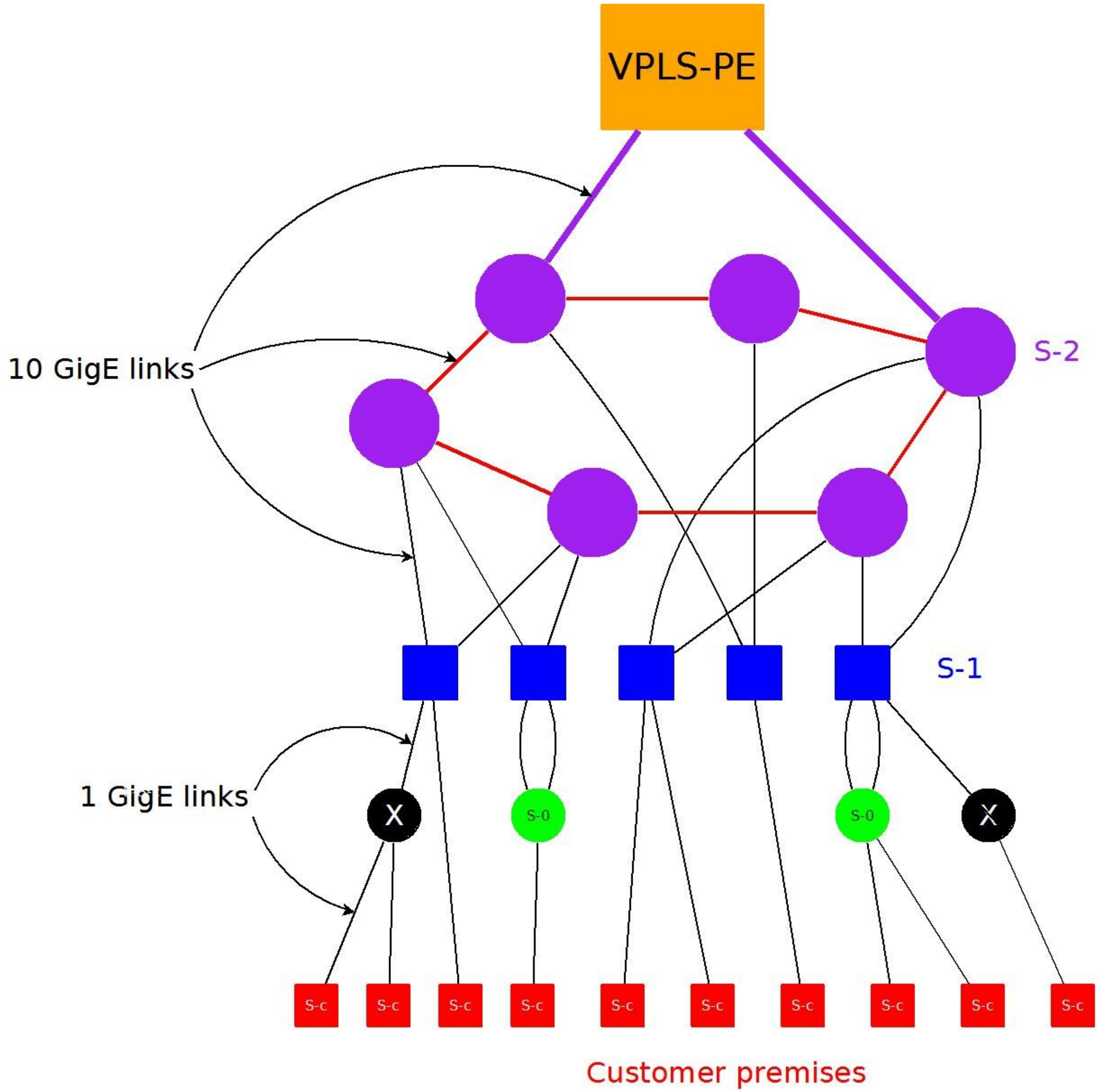
- What equipment to deploy in each central office
  - Observing limits (max S-0, S-1, S-2, ...) of each central office
- How to establish links connecting equipment
  - Obeying topology and diversity
  - Maximum length of fiber connection
  - Supporting traffic to and from demand points



Target topology

Determine:

- What equipment to deploy in each central office
  - Observing limits (max S-0, S-1, S-2, ...) of each central office
- How to establish links connecting equipment
  - Obeying topology and diversity
  - Maximum length of fiber connection
  - Supporting traffic to and from demand points



Objective:

- Minimize equipment and connection costs

Key decisions

- Nodes that will host S-1s

Key decisions

- Nodes that will host S-1s
- Nodes that will host S-2s

Key decisions

- Nodes that will host S-1s
- Nodes that will host S-2s
- Connection of S-c to S-0 or S-1s

Key decisions

- Nodes that will host S-1s
- Nodes that will host S-2s
- Connection of S-c to S-0 or S-1s
- Connection of S-0 to S-1s

Key decisions

- Nodes that will host S-1s
- Nodes that will host S-2s
- Connection of S-c to S-0 or S-1s
- Connection of S-0 to S-1s
- Connection of S-1s to S-2s

Key decisions

- Nodes that will host S-1s
- Nodes that will host S-2s
- Connection of S-c to S-0 or S-1s
- Connection of S-0 to S-1s
- Connection of S-1s to S-2s
- Interconnection of S-2s and VPLS-PE

Computational challenges

- Large scale (problems can have hundreds of nodes and links)
- Non-linearity of costs
- Solution turnaround should be minutes/hours rather than days/weeks

Computational challenges

- Large scale (problems can have hundreds of nodes and links)
- Non-linearity of costs
- Solution turnaround should be minutes/hours rather than days/weeks

Too large, non-linear, for integer programming solution.

Computational challenges

- Large scale (problems can have hundreds of nodes and links)
- Non-linearity of costs
- Solution turnaround should be minutes/hours rather than days/weeks

Too large, non-linear, for integer programming
Solution: heuristics needed.

Computational challenges

- Large scale (problems can have hundreds of nodes and links)
- Non-linearity of costs
- Solution turnaround should be minutes/hours rather than days/weeks

Too large, non-linear, for integer programming
Solution: heuristics needed: BRKGA

Encoding

X =

1	2	3	4	. . .	n-1	n	n+1	n+2	n+3	n+4	. . .	2n-1	2n

- Central offices are $V^* = \{1, \ldots, n\}$: where equipment can be located
- Solution is encoded as a real $2 \times n$-vector X of random keys in $[0,1)$, where $n = |V^*|$
- The first n elements of X correspond to S-1 locations
- The last n elements of X correspond to S-2 locations

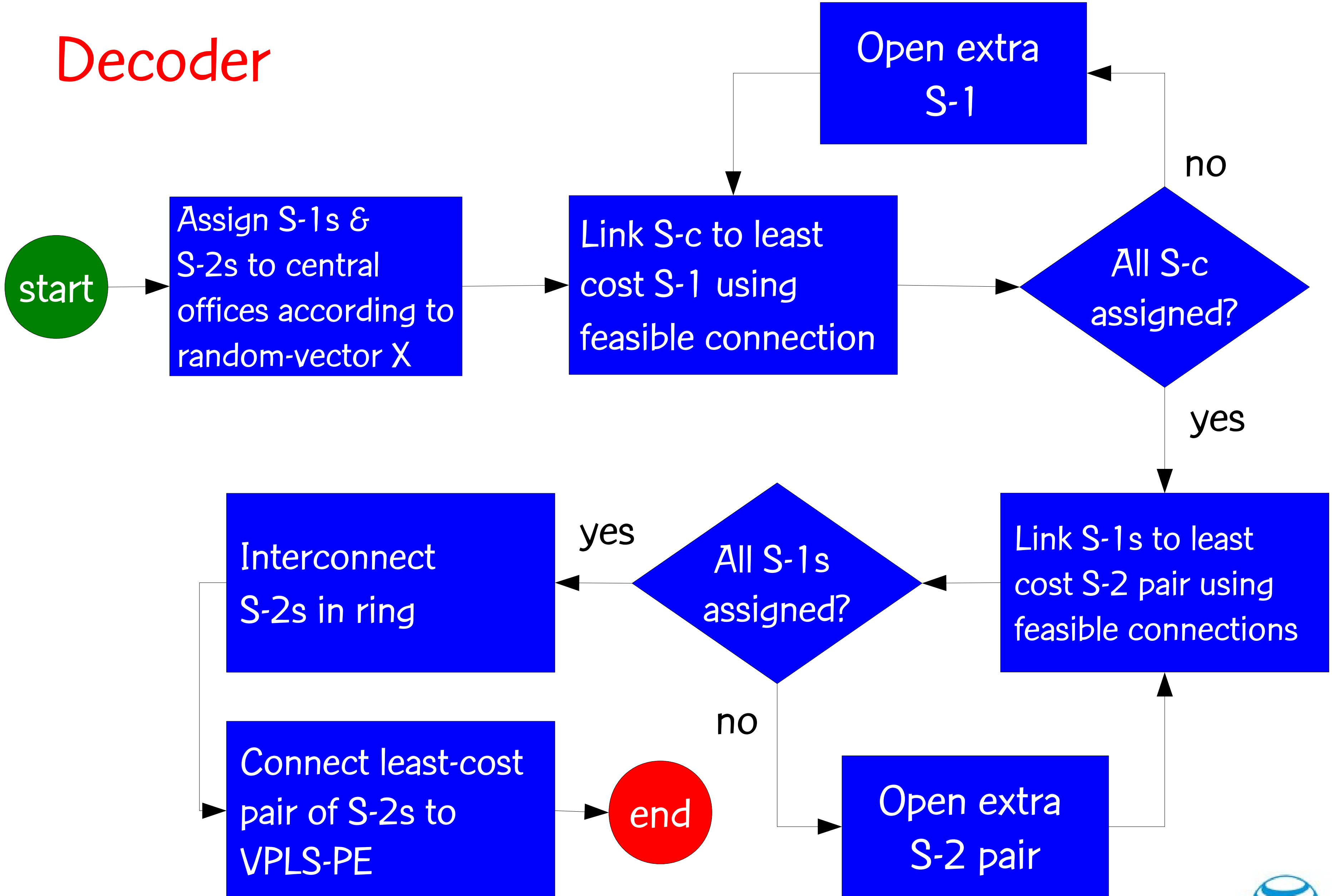
Decoding vector of random keys

X =

1	2	3	4	. . .	n-1	n	n+1	n+2	n+3	n+4	. . .	2n-1	2n

- Decoder takes as input a vector of random keys X with $2 \times n$ keys
- Initial equipment placement is done with random keys:
  - Location i in {1, ..., n} hosts an S-1 if $X[i] \geq 0.5$
  - Location j in {1, ..., n} hosts an S-2 if $X[j+n] \geq 0.5$
    - If #S-2 not even, add S-2 at $j = \text{argmax} \{ X[j]: X[j] < 0.5 \}$

Decoder


start
Assign S-1s &
S-2s to central
offices according to
random-vector X
Link S-c to least
cost S-1 using
feasible connection
All S-c
assigned?
no
Open extra
S-1
yes
Link S-1s to least
cost S-2 pair using
feasible connections
All S-1s
assigned?
no
Open extra
S-2 pair
yes
Interconnect
S-2s in ring
Connect least-cost
pair of S-2s to
VPLS-PE
end

Decoding vector of random keys: S-c to S-1

- Place preassigned S-1s in their nodes
- Place new S-1 in node i iff random key $X[i] \geq 0.5$ and no preassigned S-1 is already in node i
- For all S-c with demand
  - Compute min-cost of path to each assigned S-1 node
  - If no feasible path exists, save S-c for processing later
  - Else, assign S-c to node associated with min cost path

Decoding vector of random keys: S-c to S-1

- For all nodes i such that X[i] < 0.5
  - Compute min-cost path to all unassigned S-c's
- Repeat until all S-c's are assigned:
  - Greedy algorithm: Place new S-1 in node that can accommodate maximum number of yet unassigned S-c's
  - Assign those S-c's to that S-1
- Remove S-1s that do not receive S-c demand

Decoding vector of random keys: S-1 to S-2 pair

- Place preassigned S-2s in their nodes
- Place new S-2 in node i iff random key $X[n+i] \geq 0.5$ and no preassigned S-2 is already in node i
- Pair up S-2s

Decoding vector of random keys: S-1 to S-2 pair

- For each S-1 compute cost to connect to each S-2 pair using node disjoint paths
- If possible, assign S-1 to least cost S-2 pair;
  Otherwise, save S-1 for processing later
- Apply greedy algorithm to deploy new S-2 pairs (rank pairs by number of yet unassigned S-1's that can be assigned to pair)
- Remove S-2's that do not receive S-1 traffic

Decoding vector of random keys: interconnect S-2s

- Let q be the number of S-2s deployed
- Create ring with S-2s
- For every permutation $\pi = \{\pi_1, \pi_2, \ldots, \pi_q\}$
  - Compute tour $v[\pi_1], v[\pi_2], \ldots, v[\pi_q]$ where links between $v[\pi_i]$ and $v[\pi_{i+1}]$ are feasible min-cost node disjoint paths
- Deploy links corresponding to min-cost permutation

Decoding vector of random keys: interconnecting S-2s and the VPLS-PE

- Let q be the number of S-2s deployed
- For each S-2, compute cost to connect with VPLS-PE (assume link has to accommodate all traffic in network)
- Connect least-cost S-2:VPLS-PE pair (first path)

Decoding vector of random keys: interconnecting S-2s and the VPLS-PE

- For each remaining S-2, compute cost to connect with VPLS-PE using path that is node disjoint with first path
- Connect least-cost S-2:VPLS-PE pair

Decoding vector of random keys: interconnecting S-2s and the VPLS-PE with express lanes

- Repeat until all unassigned S-2 nodes have been tested
  - For each unassigned S-2, compute cost to connect with VPLS-PE using path that is node disjoint with previous paths
  - Connect least-cost S-2:VPLS-PE pair if total cost is reduced (express lane)

Computing least-cost routes

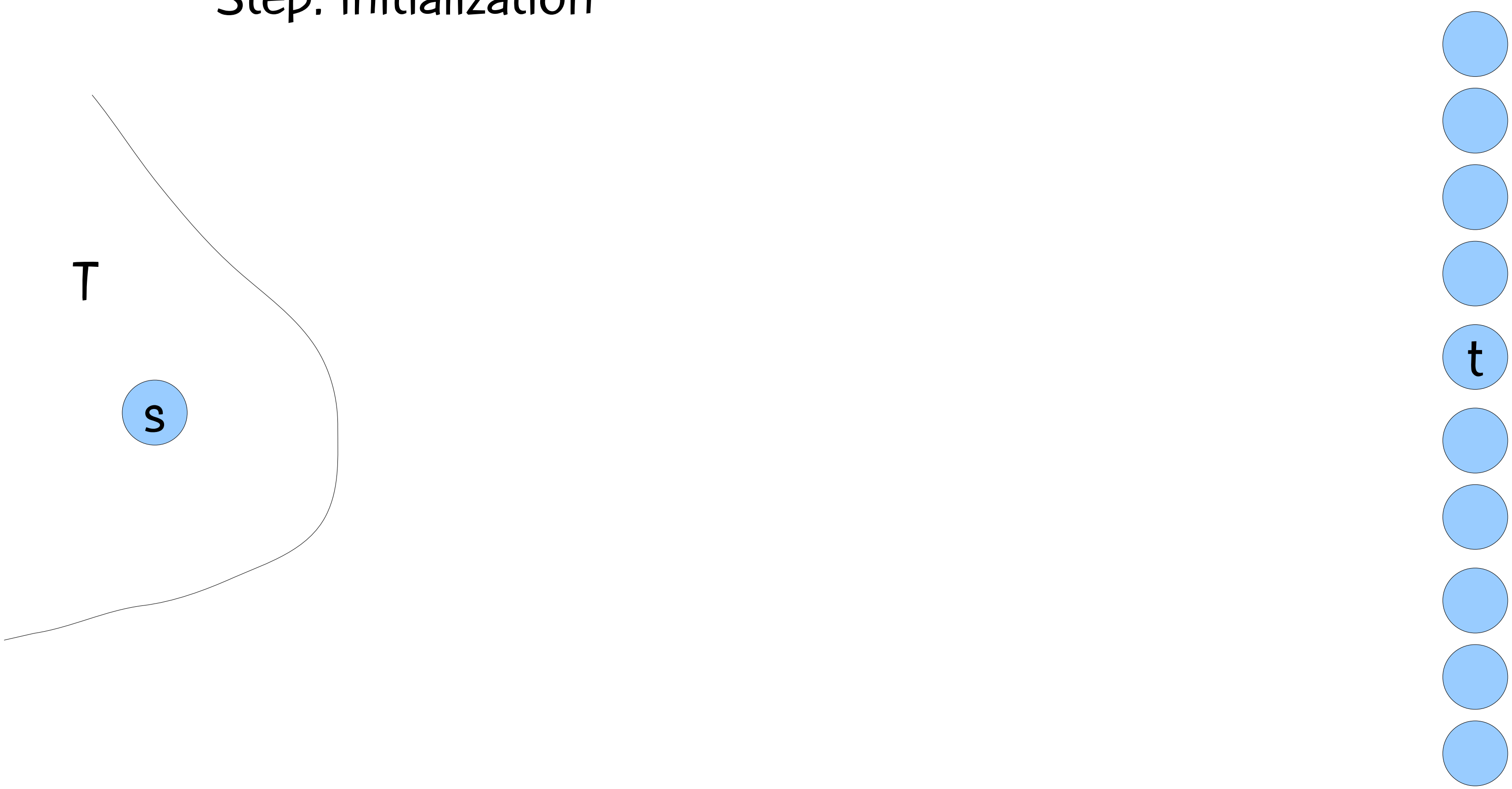
Key heuristic: s–t path-finding

- Dijkstra-based min-cost ("shortest") s–t path:
  - Input: graph G=(V,E), source node s, target node t
  - Complication: two-layer graph (FIBER/ROADM)
- Let's recall Dijkstra's algorithm...

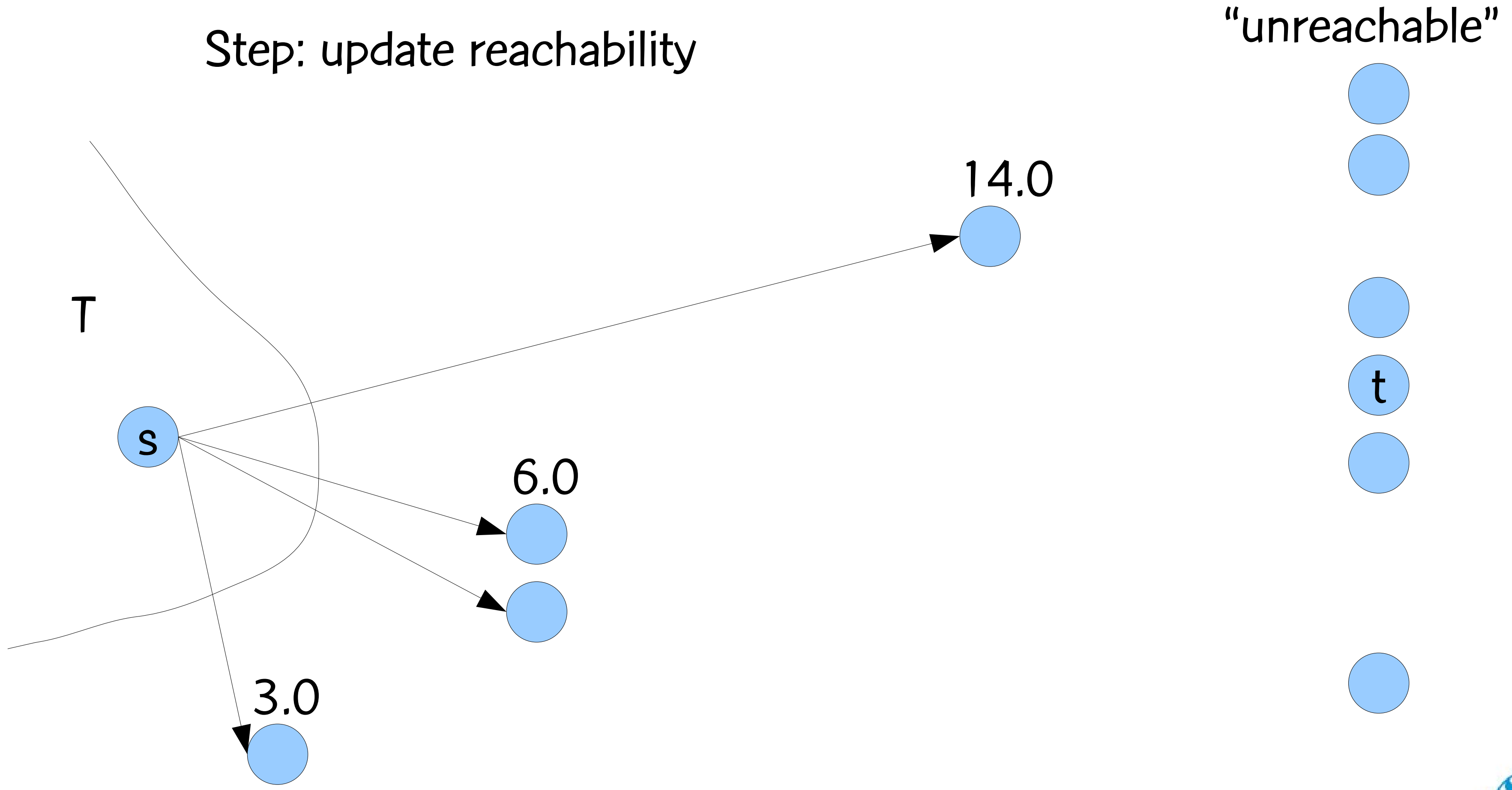
Dijkstra's single-source shortest path algorithm

Step: initialization

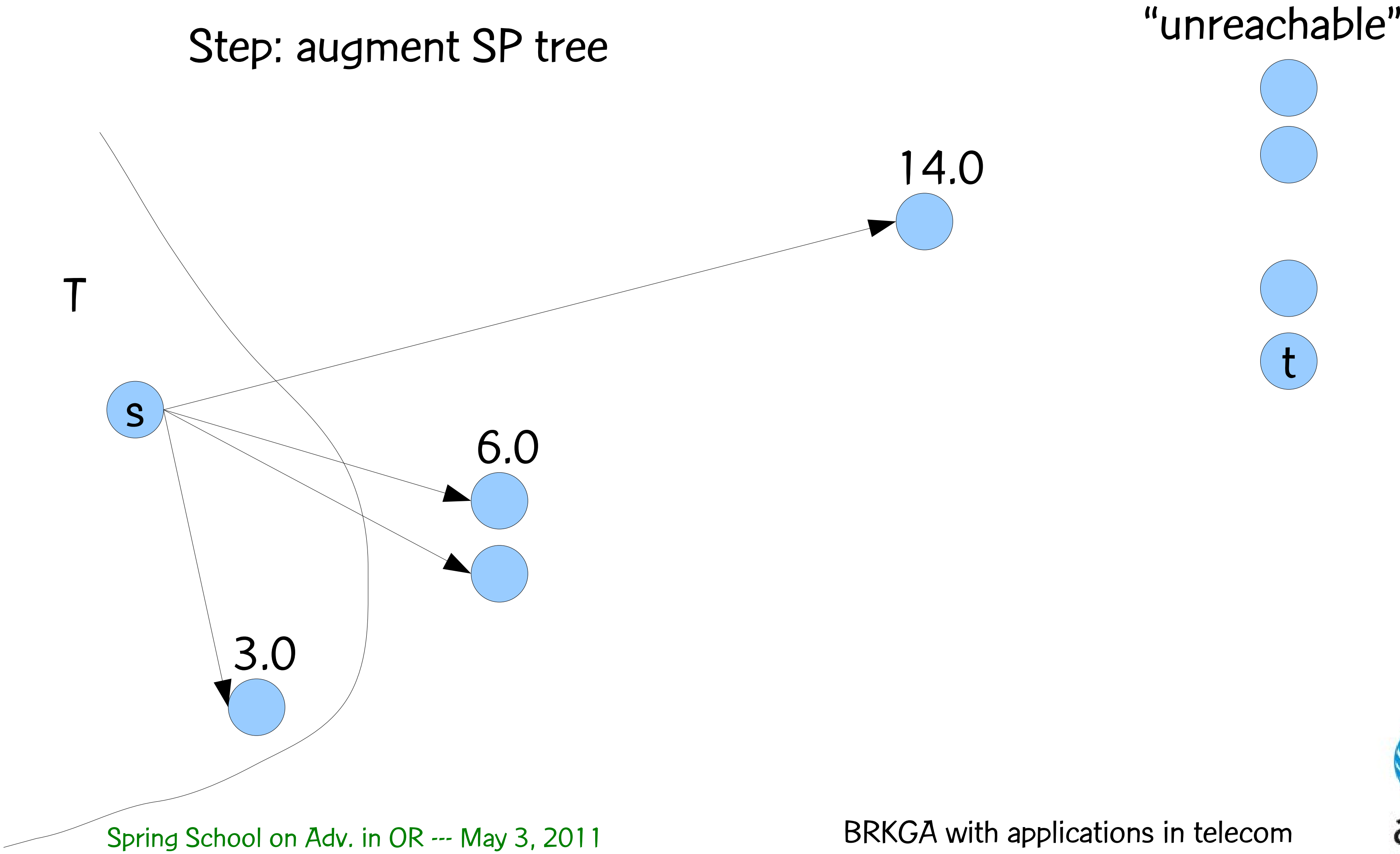
"unreachable"



Dijkstra's single-source shortest path algorithm

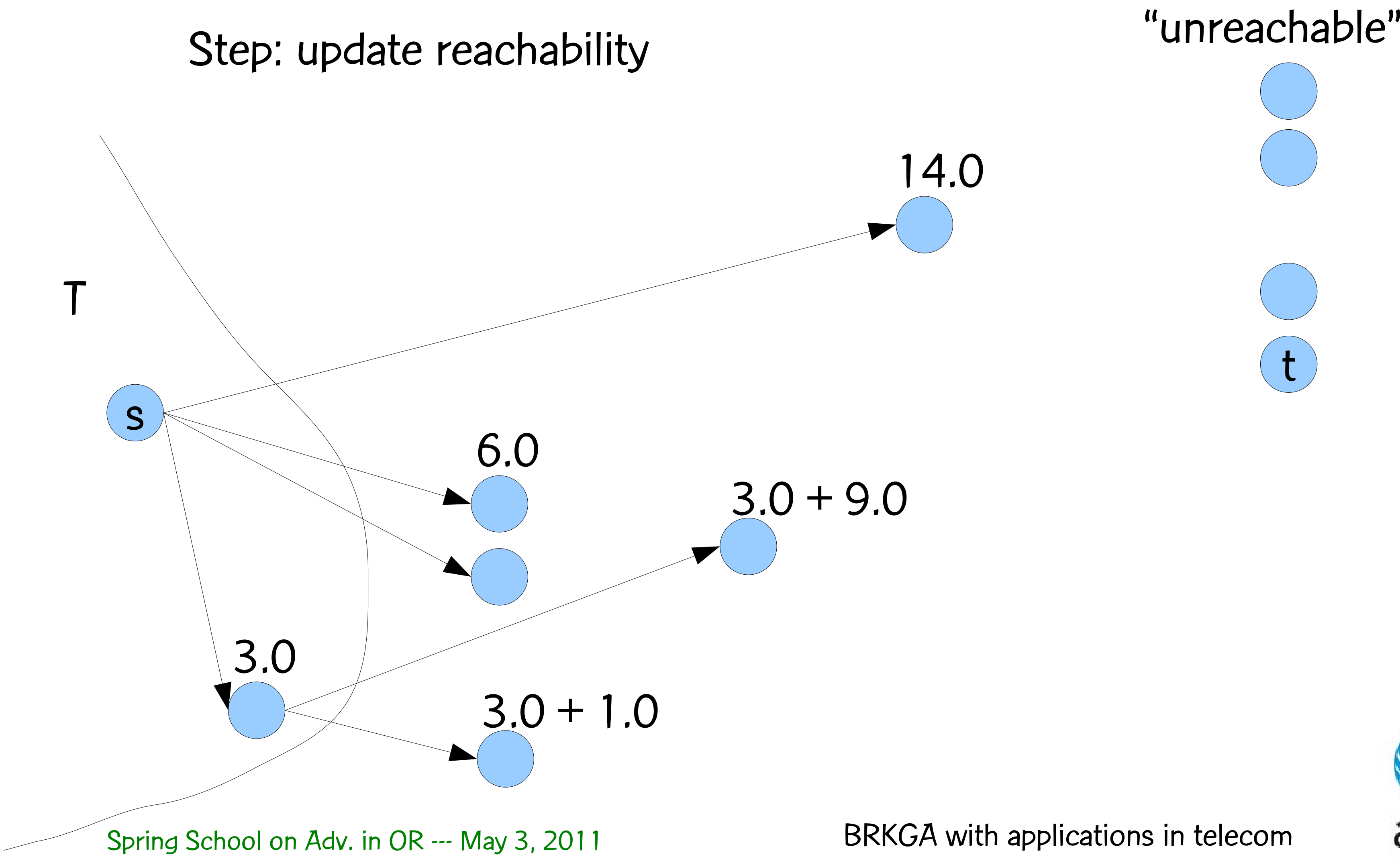


Dijkstra's single-source shortest path algorithm



Dijkstra's single-source shortest path algorithm

Step: update reachability



Key heuristic: s–t path-finding

- Each node i has reachability and a label
- Set label(s) = FIBER
- Change "update reachability" step:

Key heuristic: s–t path-finding

- If edge (u,v) analyzed is FIBER:
  - If label(u) = FIBER, we're continuing on FIBER
    - Extension cost is c_f * length(u,v)

      (fiber utilization)
  - If label(u) = ROADM, we're dropping out of ROADM
    - Extension cost is c_f * length(u,v) + c_t + c_i

      (fiber utilization + transponder + interface)
  - If extension cost is worthwhile, update reachability and set label(v) = FIBER

Key heuristic: s–t path-finding

- If edge (u,v) analyzed is ROADM:
  - If label(u) = ROADM, we're continuing on ROADM
    - Extension cost is c_p (passthrough cost only)
  - If label(u) = FIBER, we're hoping into the ROADM
    - Extension cost is $c_t + c_i + c_c$

      (transponder + interface + common costs)
  - If extension cost is worthwhile, update reachability and set label(v) = ROADM

Key heuristic: s–t path-finding

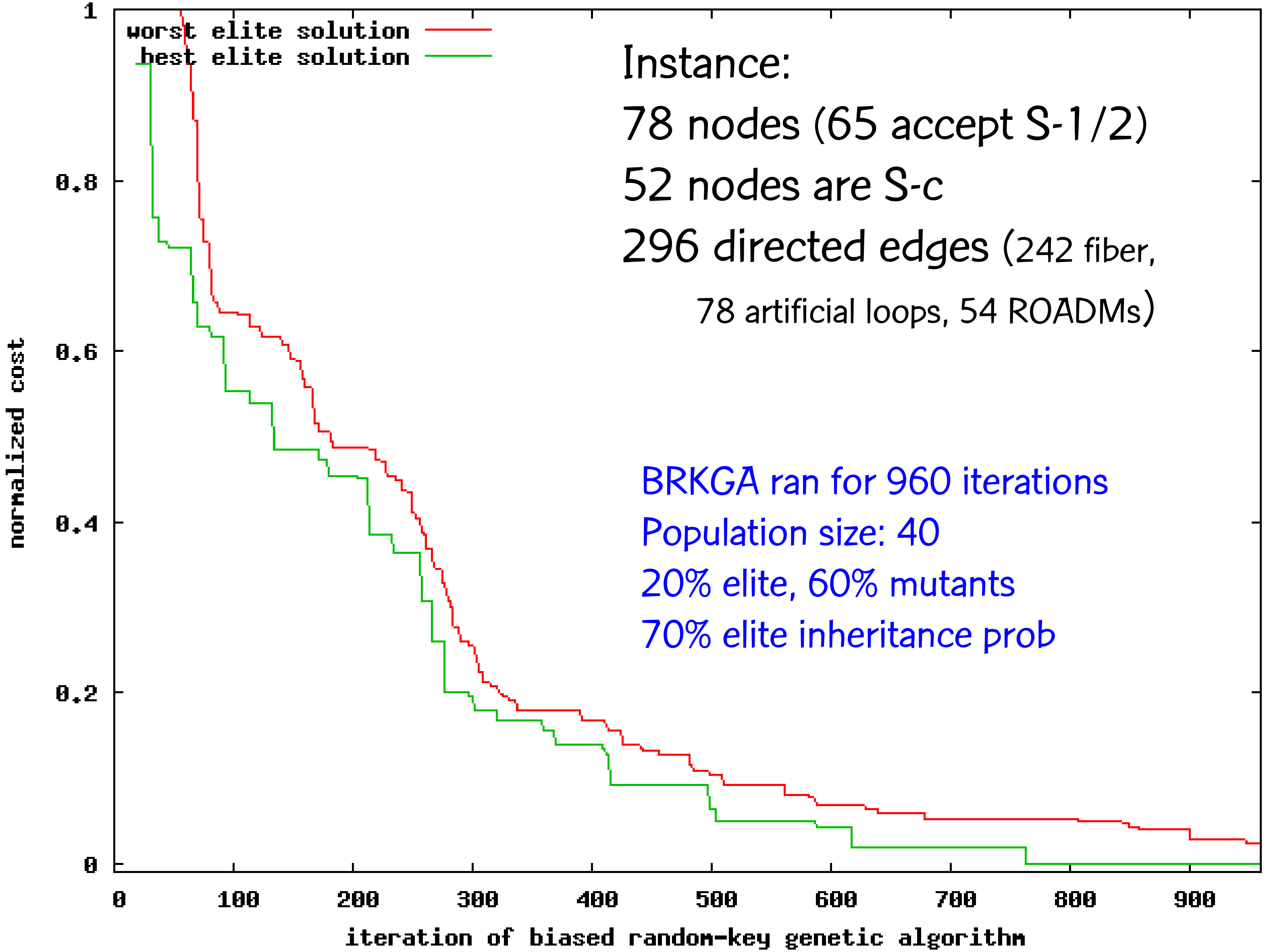
- Observations:
  - It's a heuristic...
  - But: we have observed that it works nicely
    - Avoids ROADM whenever possible
    - When into ROADM, tends to keep going on ROADM
  - Running time: $O(|E| + |V| \log |V|)$ per shortest path if implemented with Fibonacci heaps.
  - Easy to avoid nodes and their incident edges: just remove them from the heap when initializing
    - Application: connect S-1 to {S-2-A, S-2-B}

Implementation

Implementation and next steps

- Implementation is ongoing
- C++, OpenMP, highly modularized
  - BRKGA framework (Toso & Resende, 2010)
  - Decoder tailored for this problem
    - Instance input
    - Decoder heuristics
    - Solution output (including GraphViz output)

Example



Instance:

78 nodes (65 accept S-1/2)

52 nodes are S-c

296 directed edges (242 fiber,

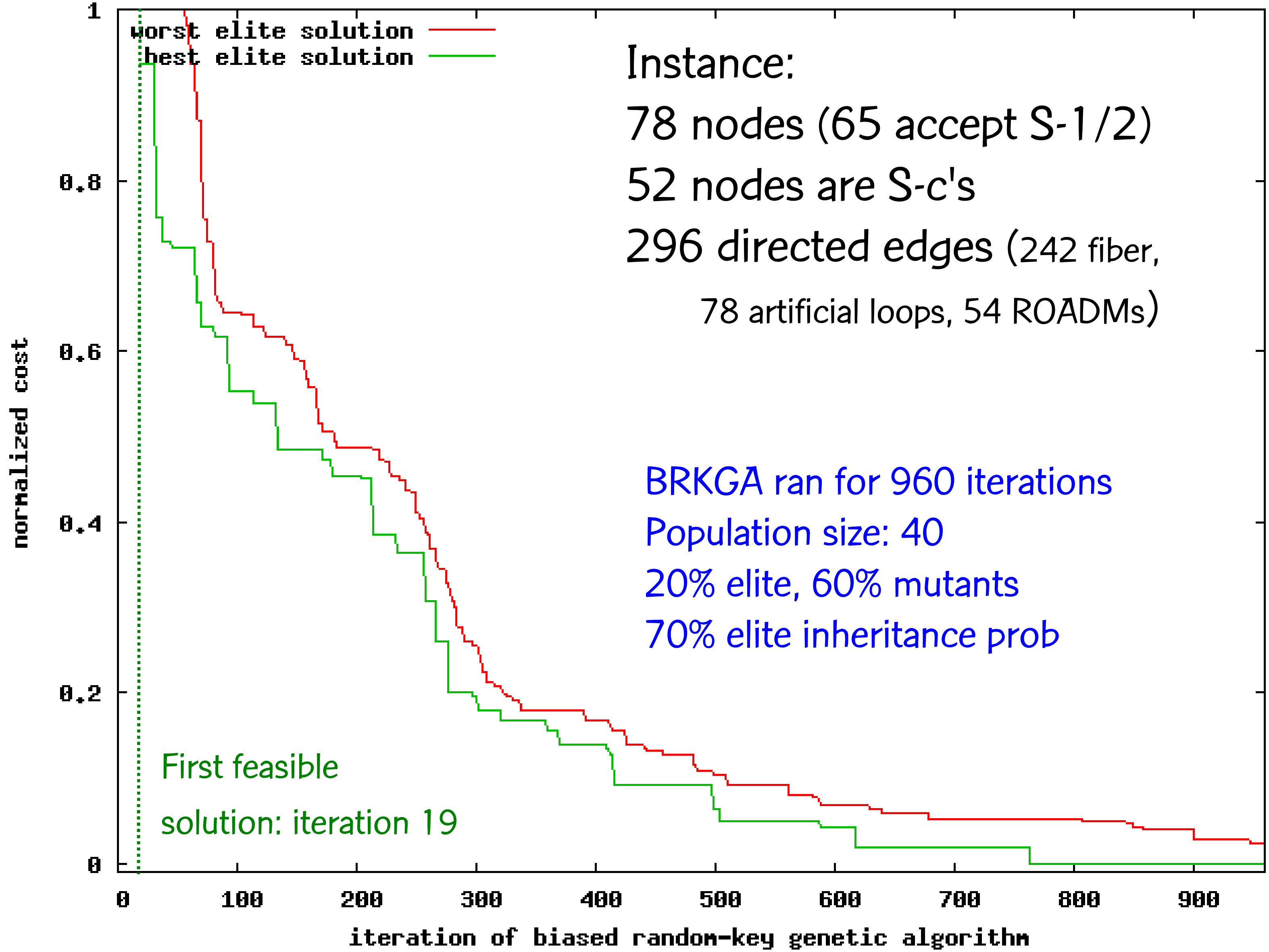
78 artificial loops, 54 ROADMs)

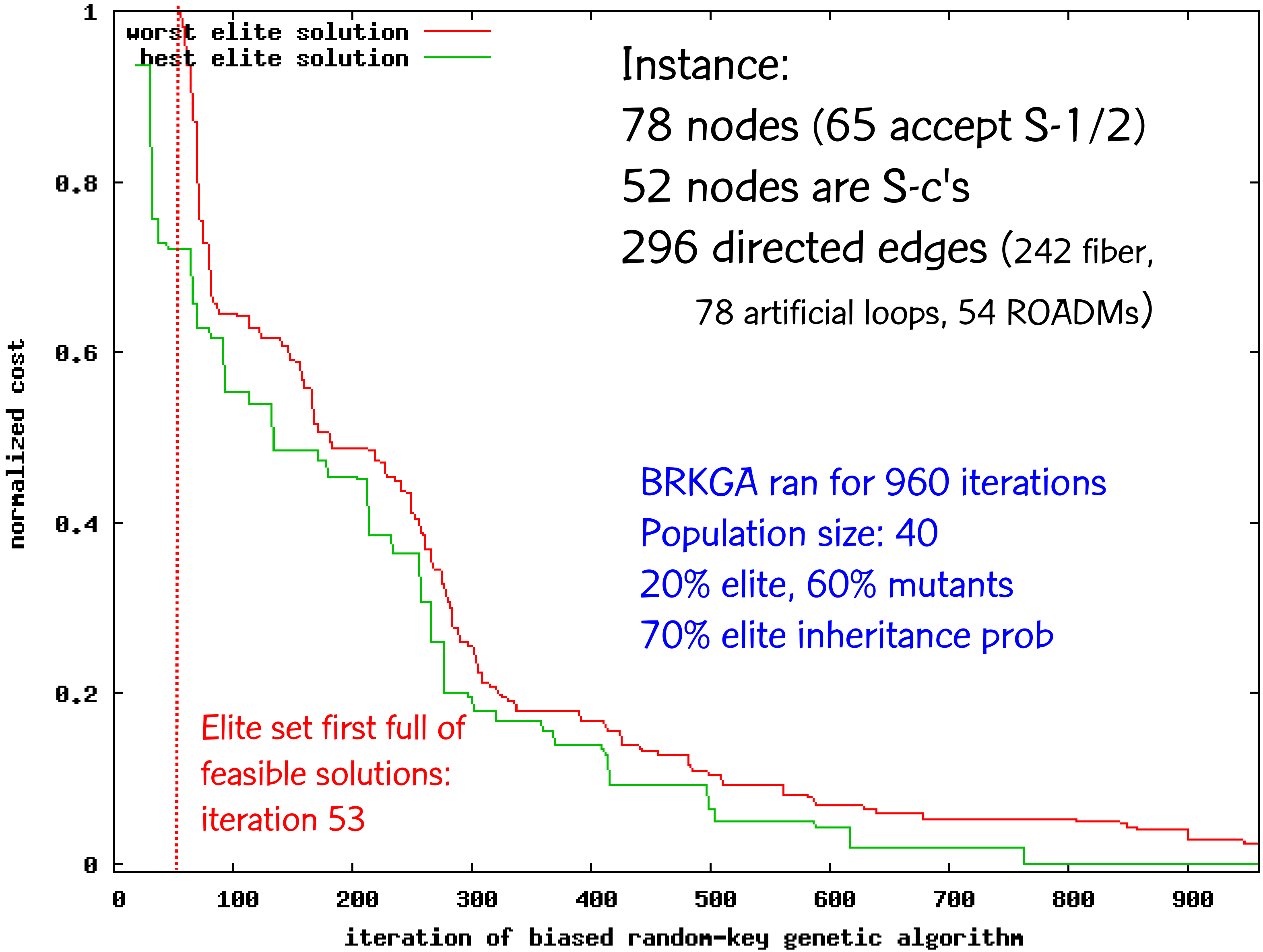
BRKGA ran for 960 iterations

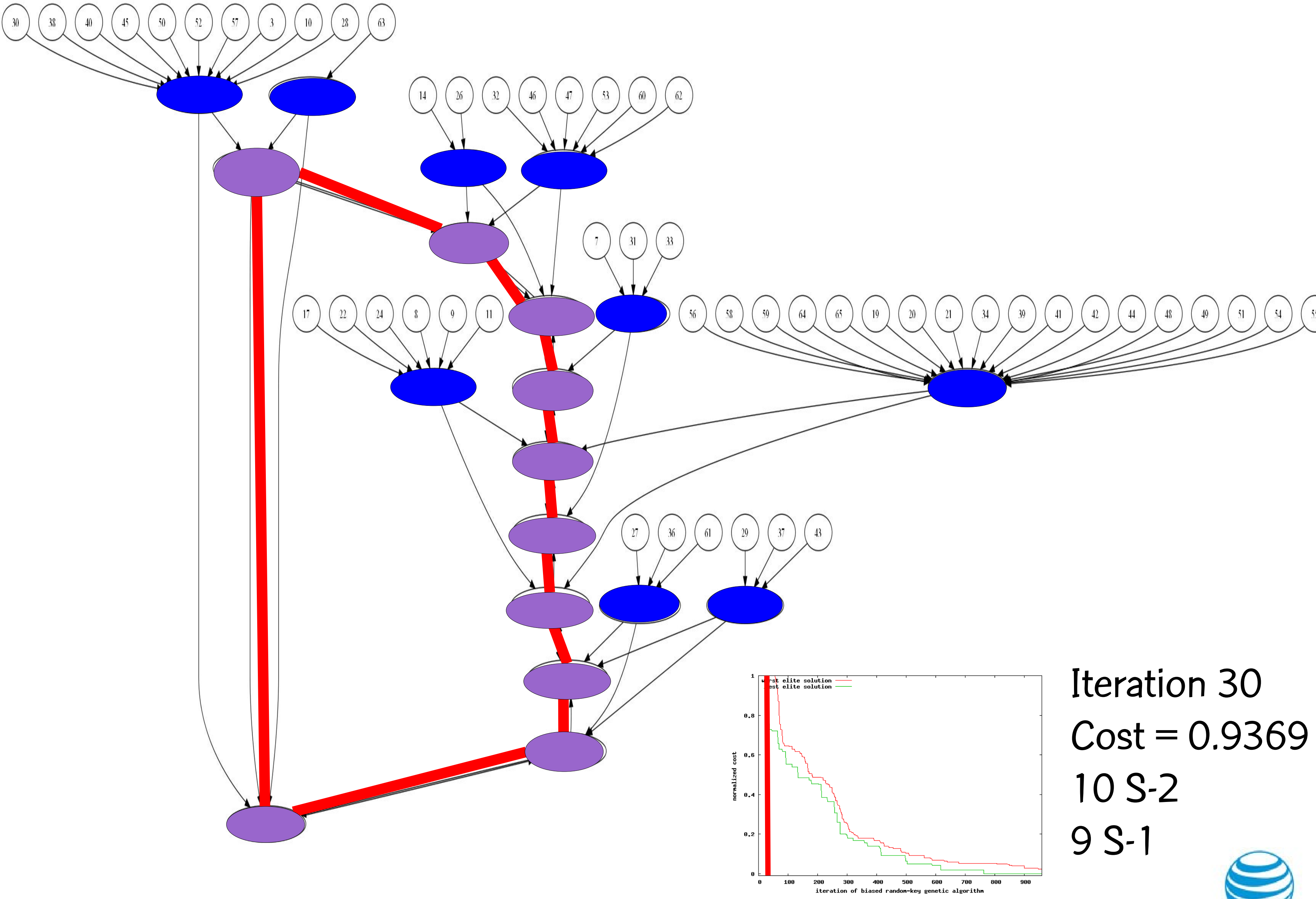
Population size: 40

20% elite, 60% mutants

70% elite inheritance prob


worst elite solution
best elite solution
Instance:
78 nodes (65 accept S-1/2)
52 nodes are S-c's
296 directed edges (242 fiber,
78 artificial loops, 54 ROADMs)
BRKGA ran for 960 iterations
Population size: 40
20% elite, 60% mutants
70% elite inheritance prob
First feasible
solution: iteration 19
normalized cost
iteration of biased random-key genetic algorithm
1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500
600
700
800
900


worst elite solution
best elite solution
Instance:
78 nodes (65 accept S-1/2)
52 nodes are S-c's
296 directed edges (242 fiber,
78 artificial loops, 54 ROADMs)
BRKGA ran for 960 iterations
Population size: 40
20% elite, 60% mutants
70% elite inheritance prob
Elite set first full of
feasible solutions:
iteration 53
normalized cost
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700
800
900
iteration of biased random-key genetic algorithm

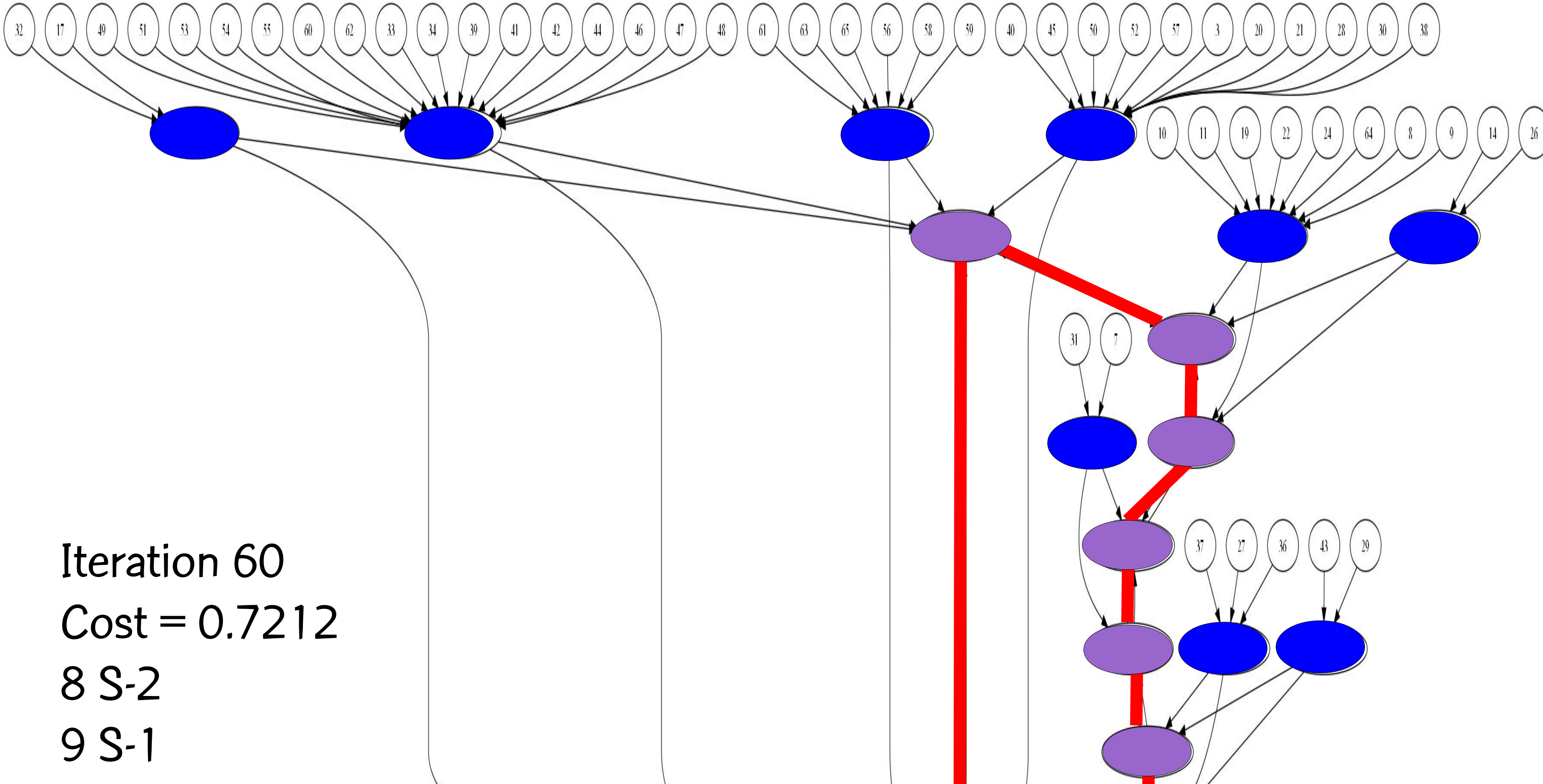


Iteration 30

Cost = 0.9369

10 S-2

9 S-1

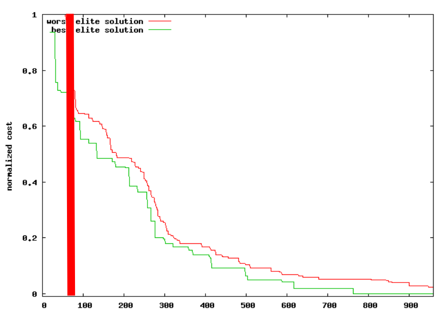


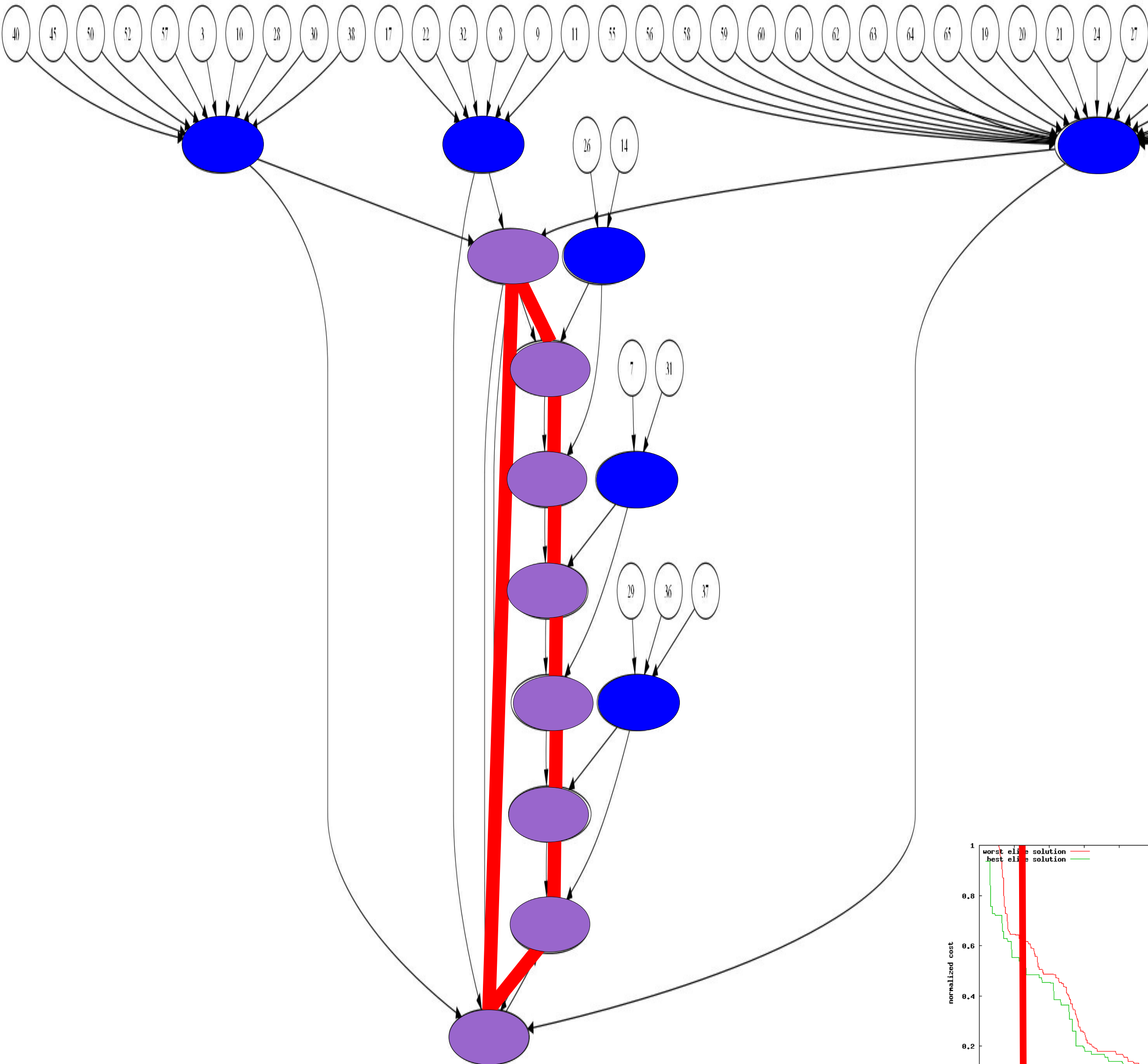
Iteration 60

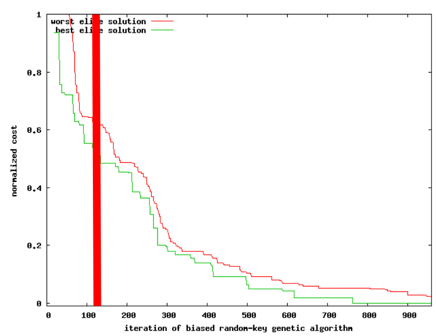
Cost = 0.7212

8 S-2

9 S-1





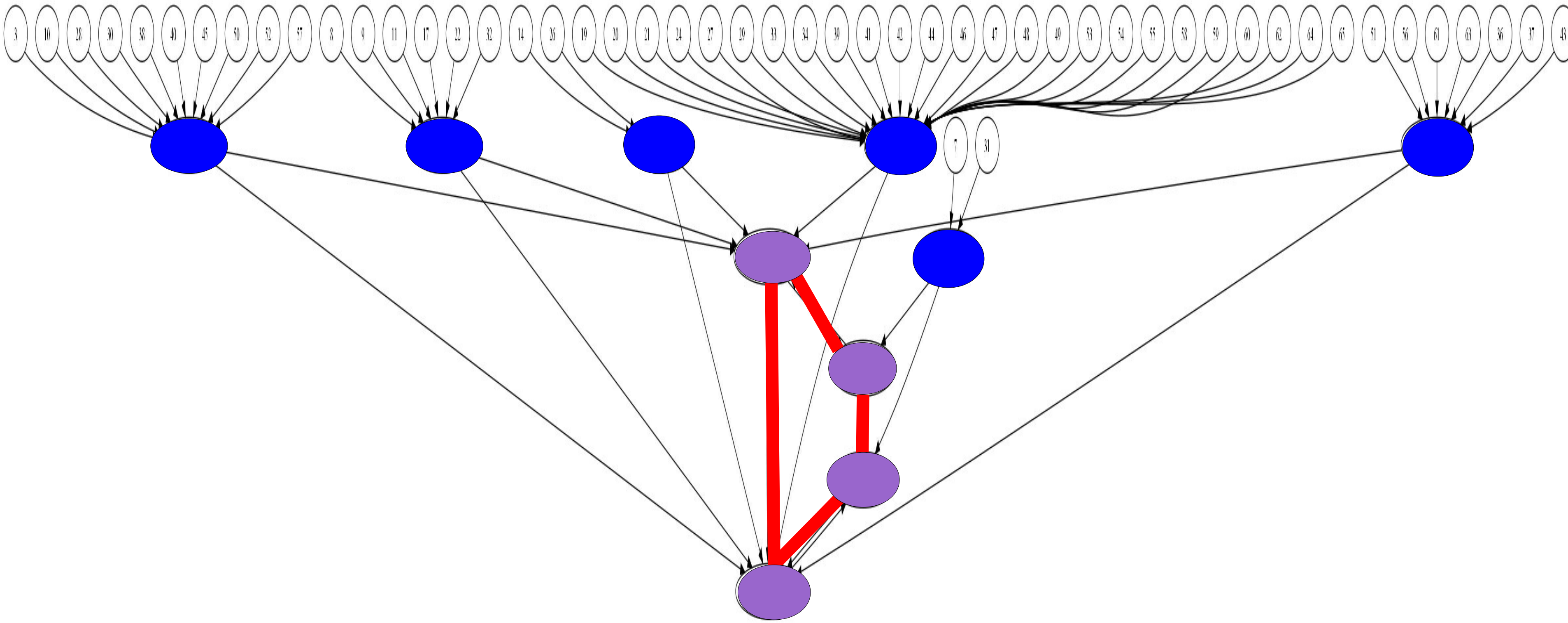


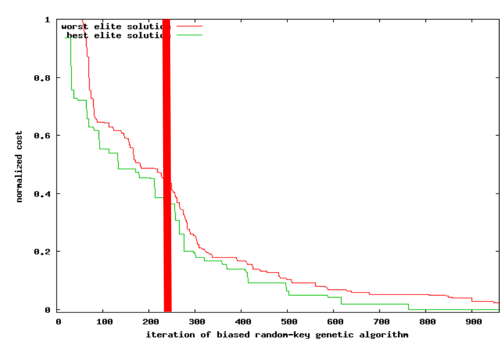
Iteration 120

Cost = 0.5380

8 S-2

6 S-1



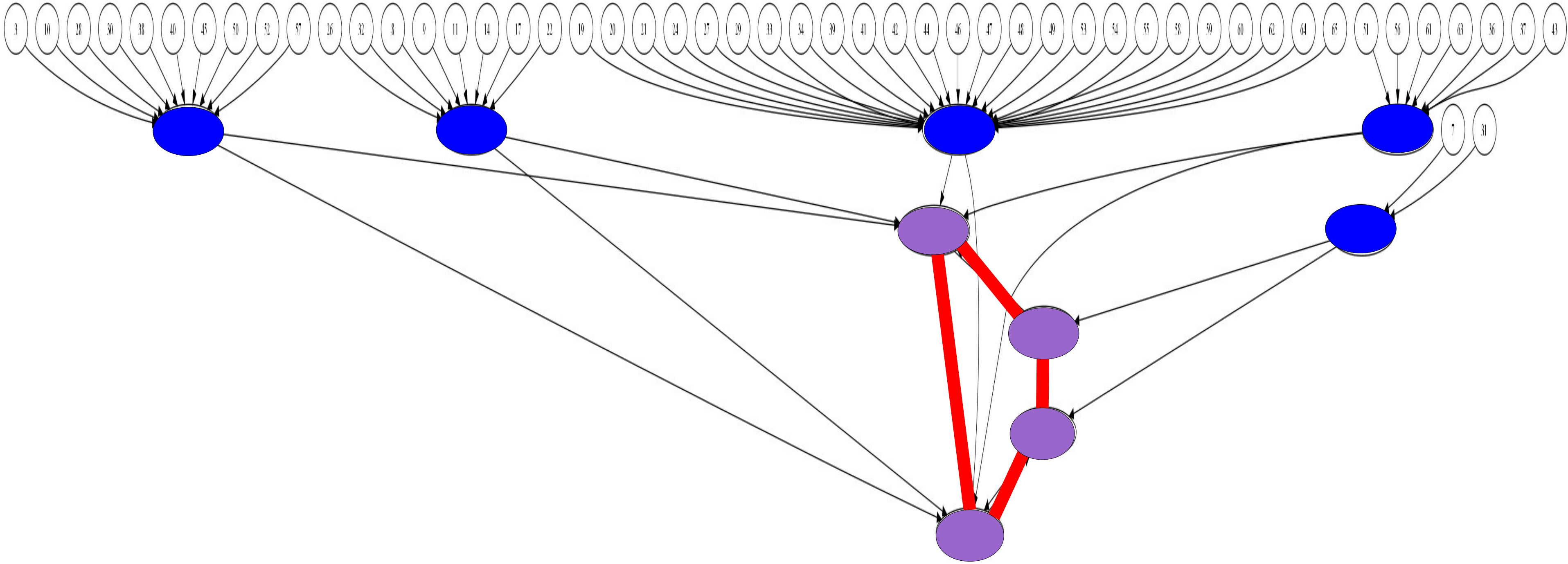


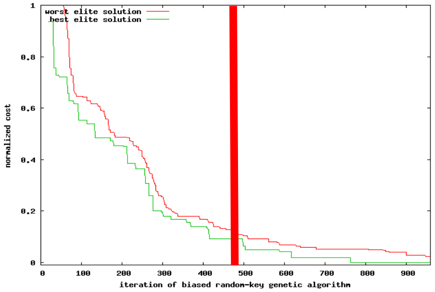
Iteration 240

Cost = 0.3636

4 S-2

6 S-1



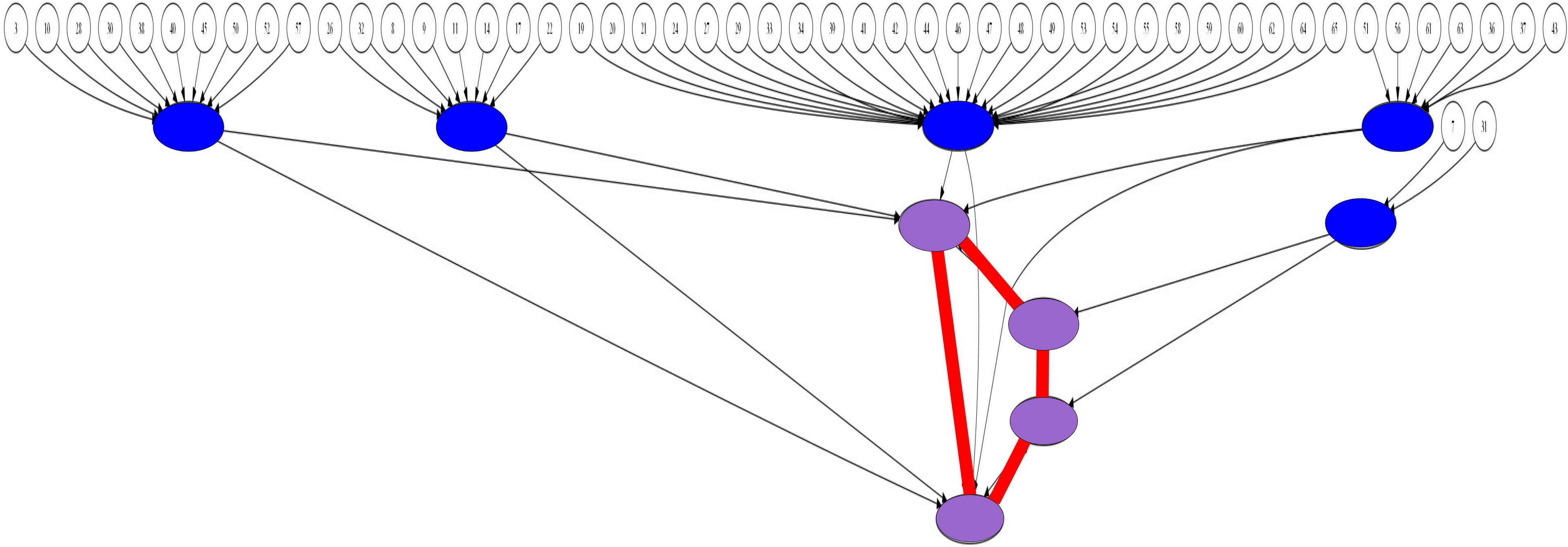


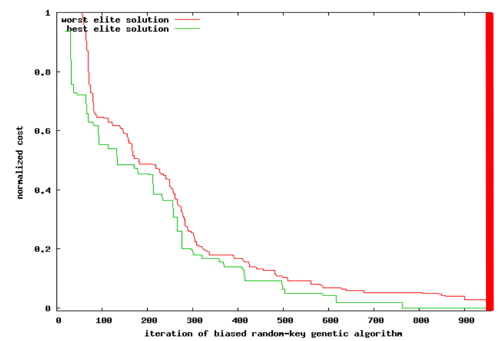
Iteration 480

Cost = 0.0921

4 S-2

5 S-1





Iteration 960

Cost = 0.0000

4 S-2

5 S-1

Concluding remarks

- Introduced metropolitan network design problem
- Proposed to use the BRKGA framework
- A multi-step decoder is proposed and for each step a simple heuristic is described
- Proposed a new variant of Dijkstra's algorithm to compute least-cost routes
- A C++ OpenMP implementation is tested on a small network

Concluding remarks

- Ongoing work
  - Add S-0 switch to design
  - Deploy flexible penalization to deal better with infeasibilities
  - Improve heuristics, in particular, least-cost routing and S-2 ring design
  - Add connectivity to VPLS-PE

Host Placement for Path-Disjoint Monitoring

Reference:

L. Breslau, I. Diakonikolas, N. Duffield, Y. Gu, M. Hajiaghayi, D.S. Johnson, H. Karloff, M.G.C.R., and S. Sen, "Disjoint-path facility location: Theory and practice," Proceedings of the Thirteenth Workshop of Algorithm Engineering and Experiments (ALENEX11), SIAM, San Francisco, pp. 60-74, January 22, 2011

Tech report version:

http://www2.research.att.com/~mgcr/doc/monitoring-alenex.pdf

Network monitoring with tomography

IP Monitoring

- Internet Service Providers need to monitor the performance of customer traffic within their networks.
- More specifically, ISPs want to measure:
  - Unidirectional reachability
  - Packet loss rate
  - Packet delay along the edge-to-edge paths followed by customer traffic

IP Monitoring

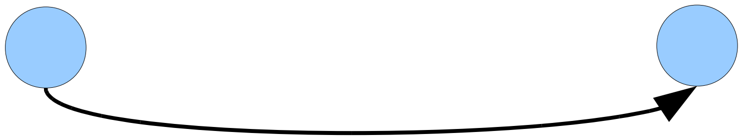
- Internet Service Providers need to monitor the performance of customer traffic within their networks.
- More specifically, ISPs want to measure:
  - Unidirectional reachability
  - Packet loss rate
  - Packet delay along the edge-to-edge paths followed by customer traffic

IP Monitoring

- Traffic entails both the links followed by traffic and the treatment of packets within the routers that move them from link to to link.
- Flow follows fine-grained paths differentiated from others by, e.g.
  - Class of service
  - Application class
  - Virtual private network (VPN) ownership

IP Monitoring

- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;

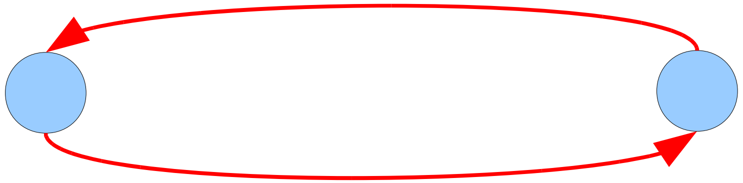


want to measure
one-way performance

IP Monitoring

- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;

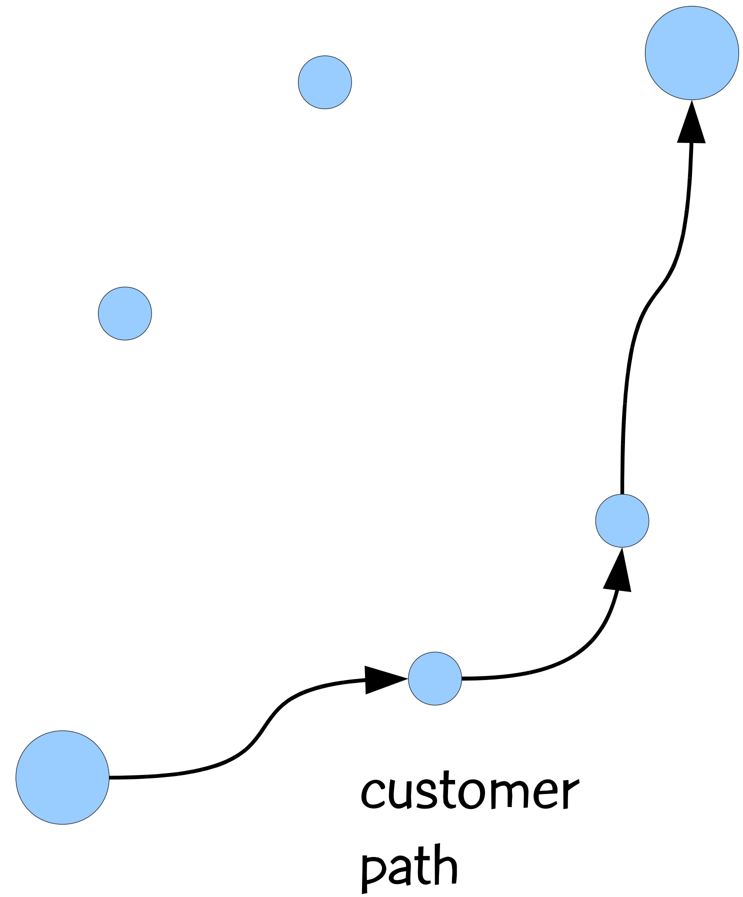
measure round-trip:
hard to infer one-way
performance



want to measure
one-way performance

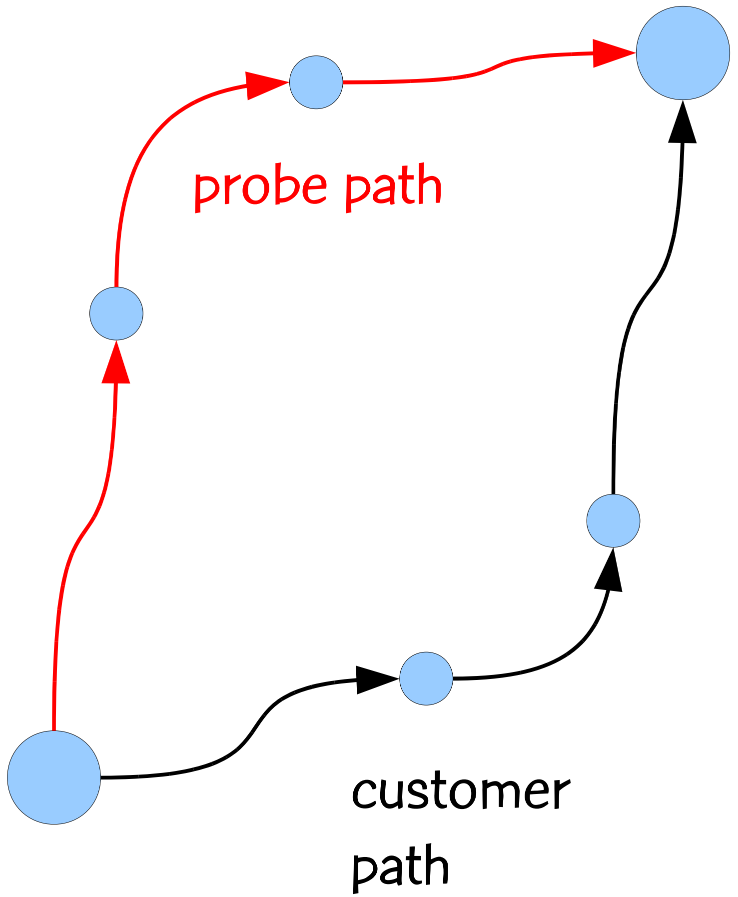
IP Monitoring

- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;
  - Their probes may not follow the customer paths, either because they transit different links, or experience different router treatment.



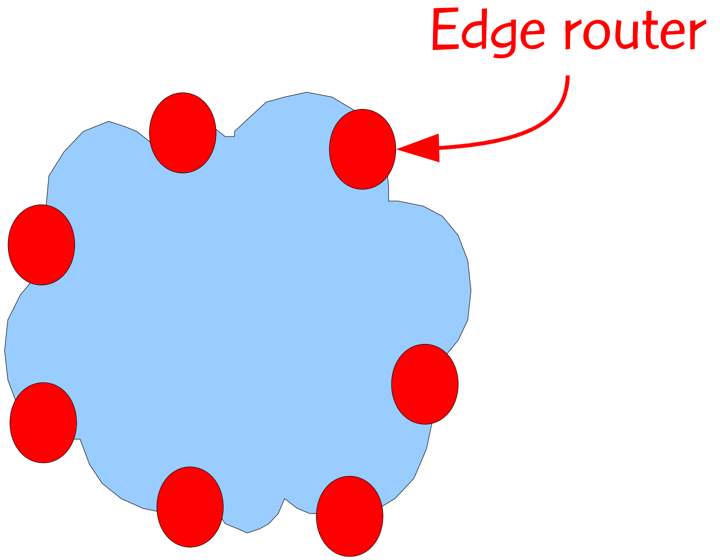
IP Monitoring

- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;
  - Their probes may not follow the customer paths, either because they transit different links, or experience different router treatment.



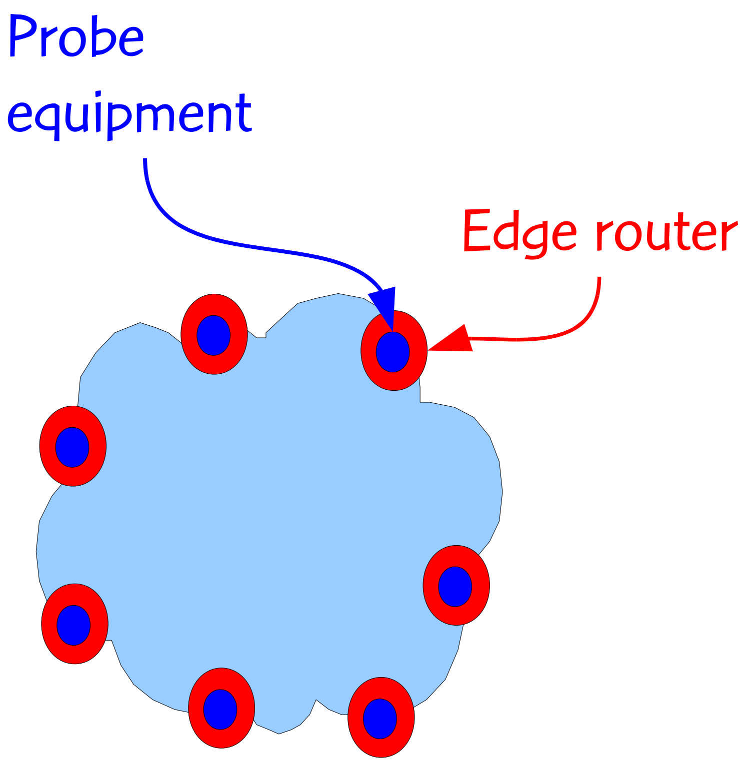
IP Monitoring

- In principle, edge routers could be equipped to launch and receive probes that follow customer traffic:
  - Could impact network performance
  - Very costly to deploy networkwide



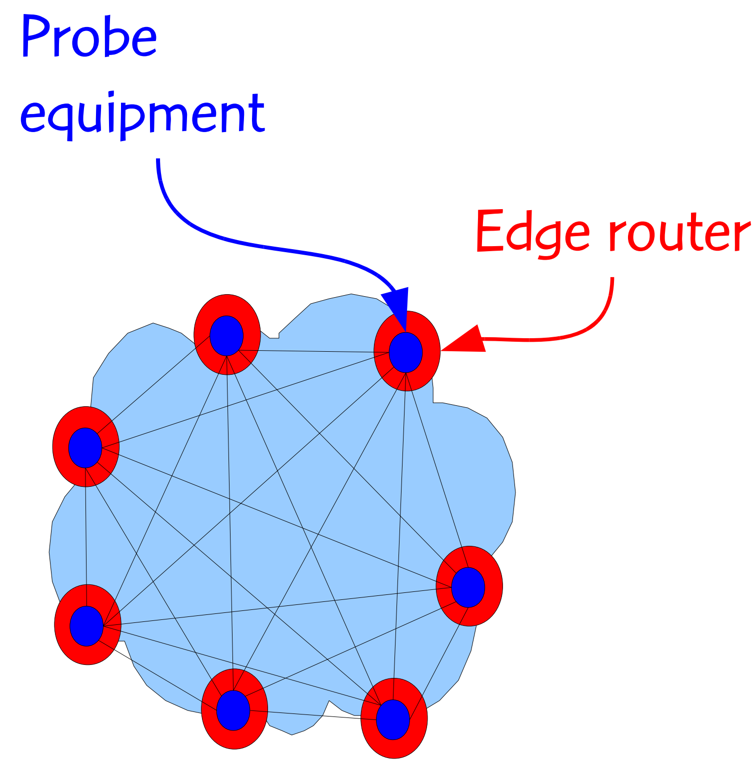
IP Monitoring

- In principle, edge routers could be equipped to launch and receive probes that follow customer traffic:
  - Could impact network performance
  - Very costly to deploy networkwide



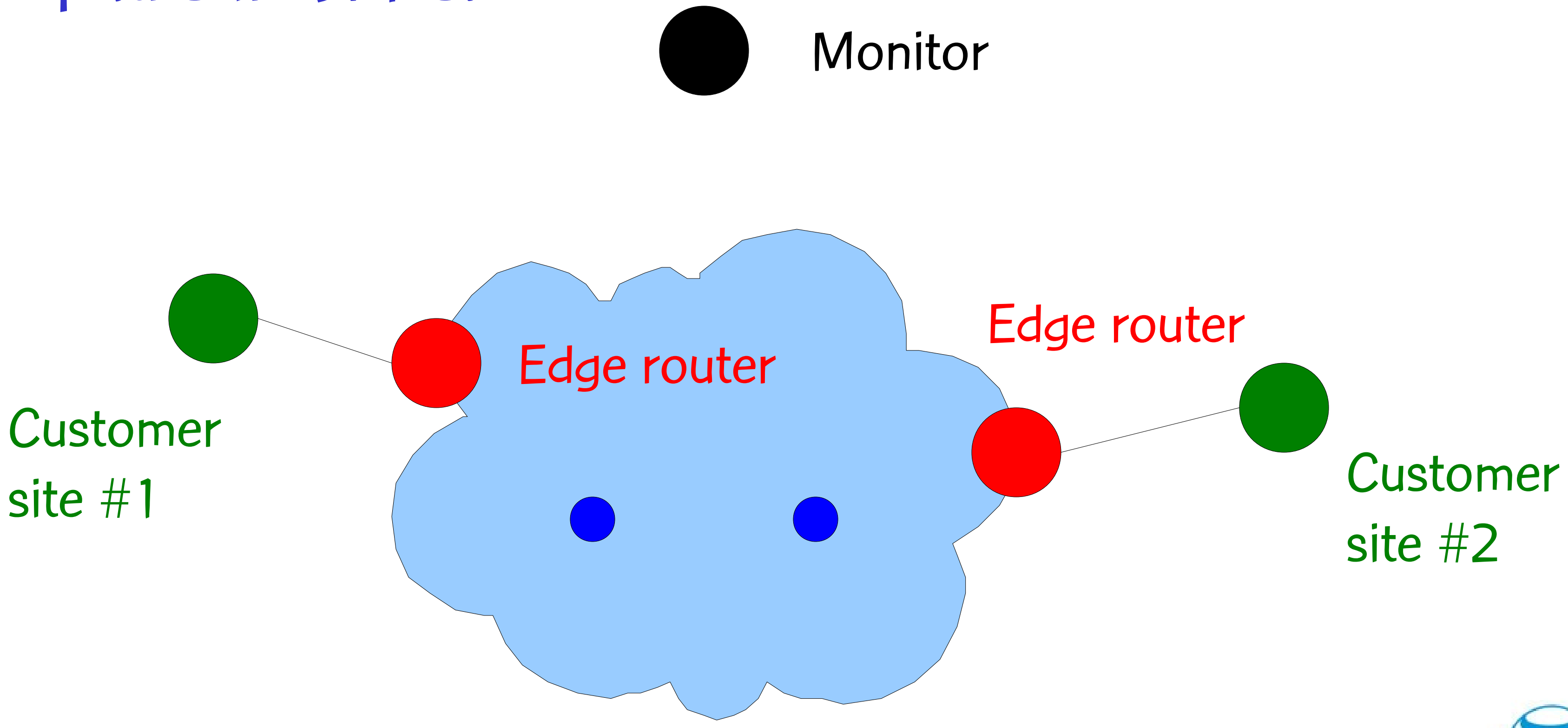
IP Monitoring

- In principle, edge routers could be equipped to launch and receive probes that follow customer traffic:
  - Could impact network performance
  - Very costly to deploy networkwide



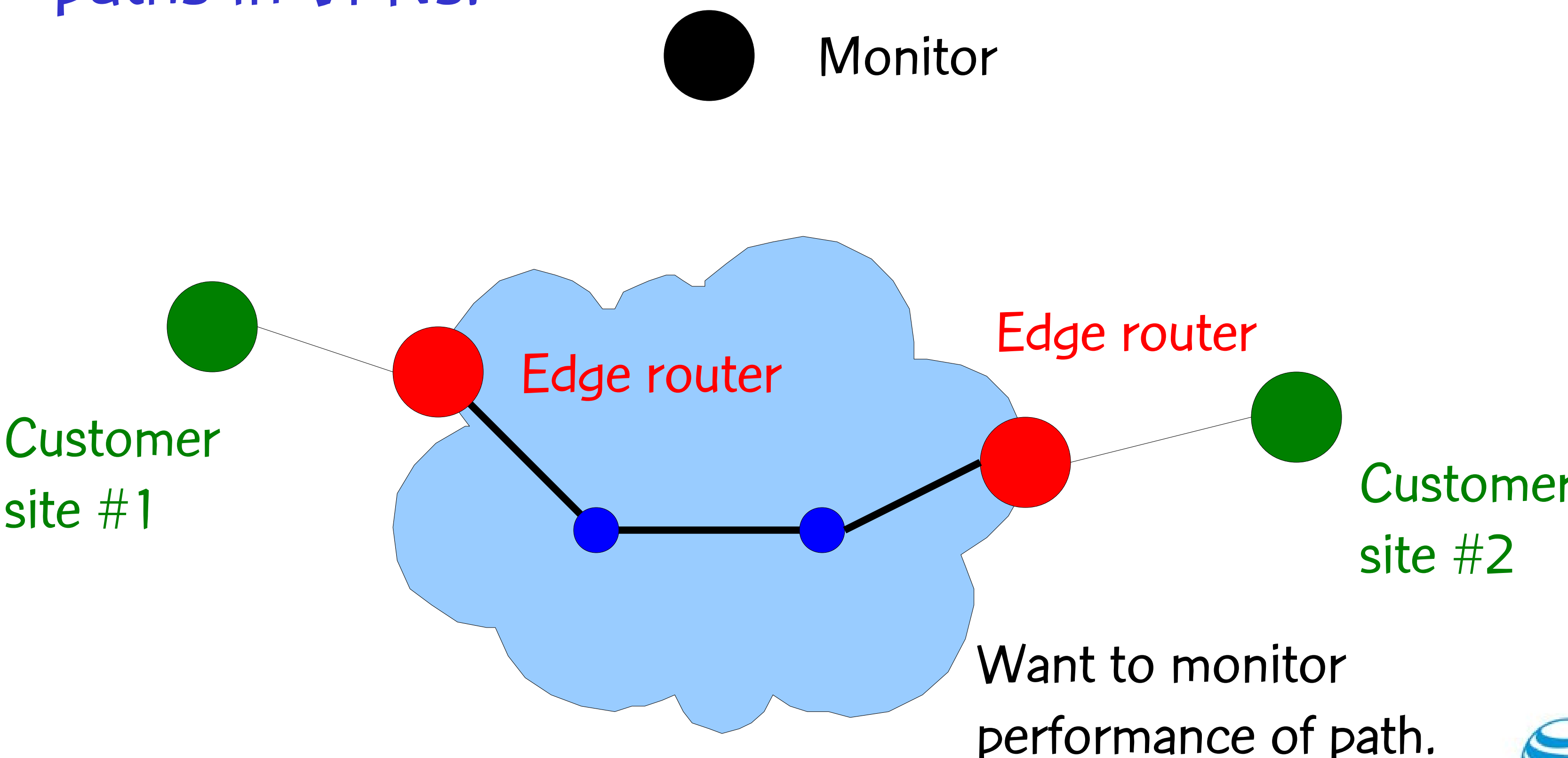
IP Monitoring

- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.



IP Monitoring

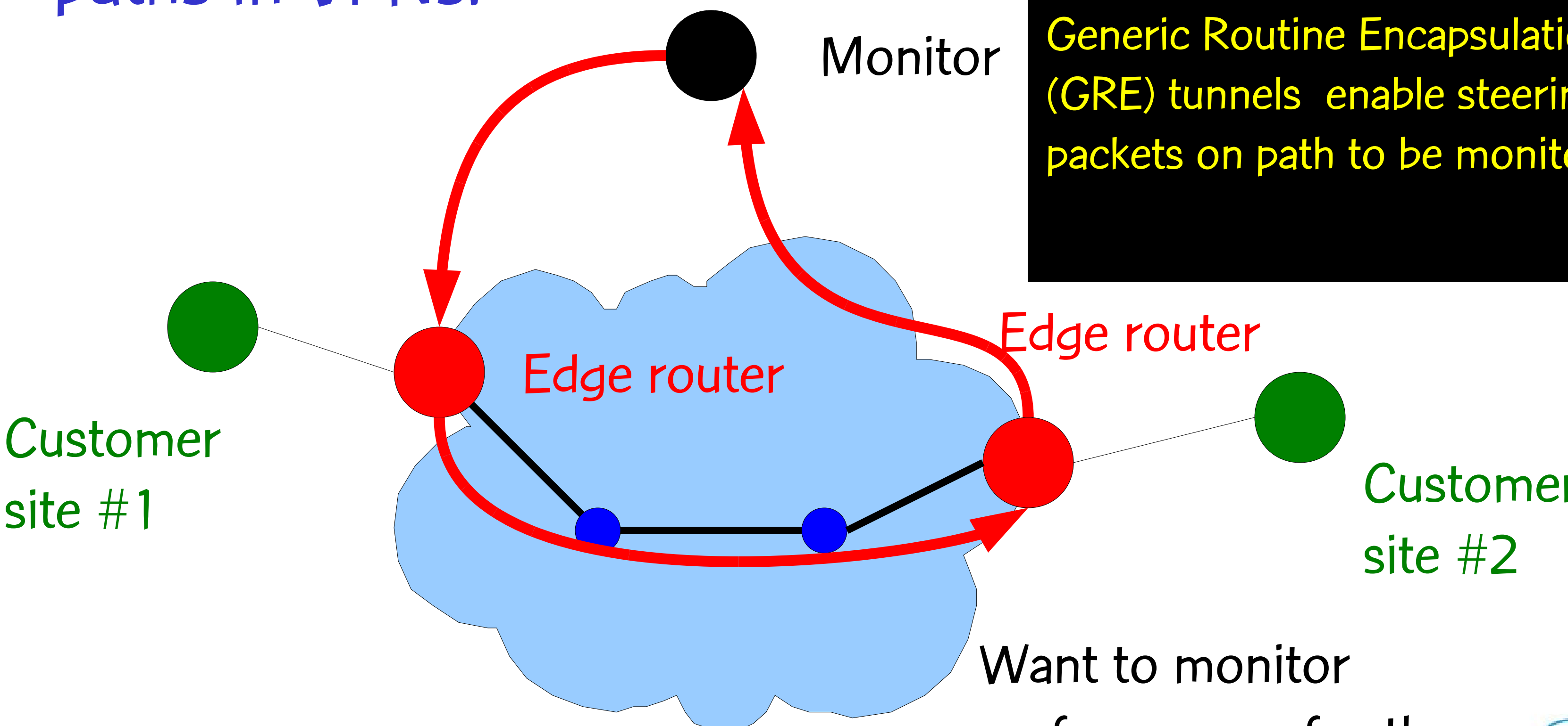
- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.



IP Monitoring

- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.

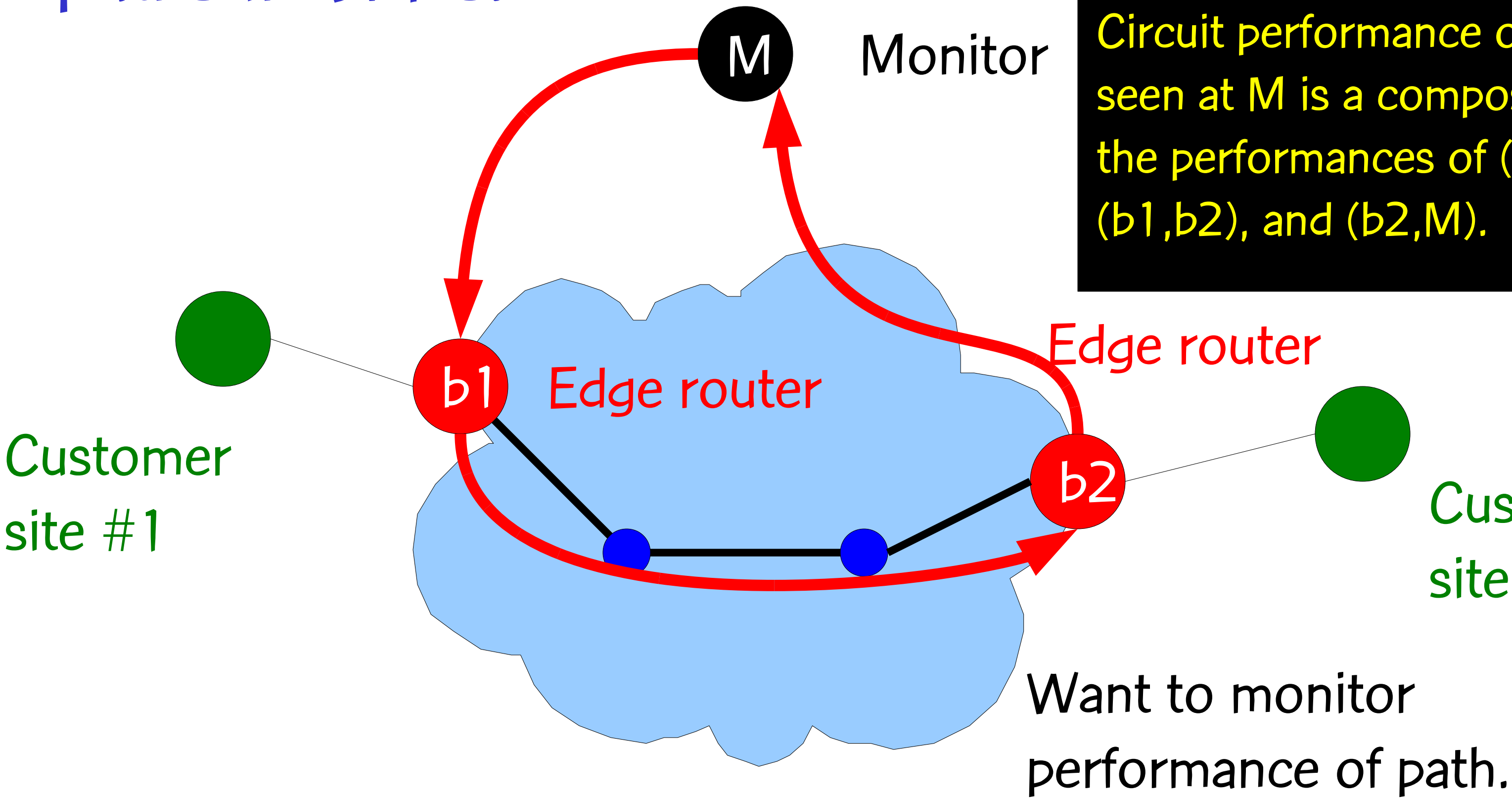
Generic Routine Encapsulation (GRE) tunnels enable steering packets on path to be monitored



Want to monitor performance of path.

IP Monitoring

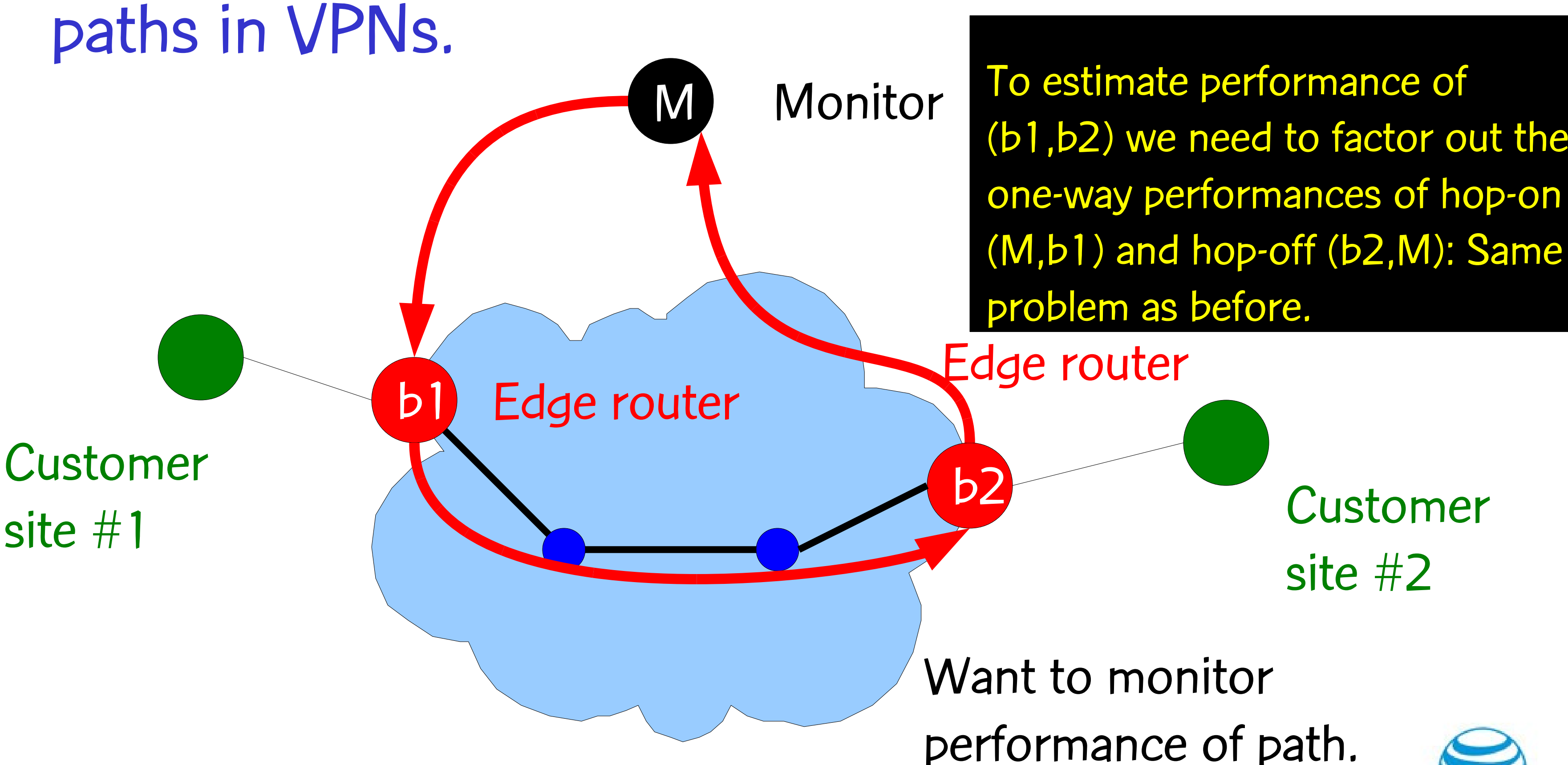
- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.



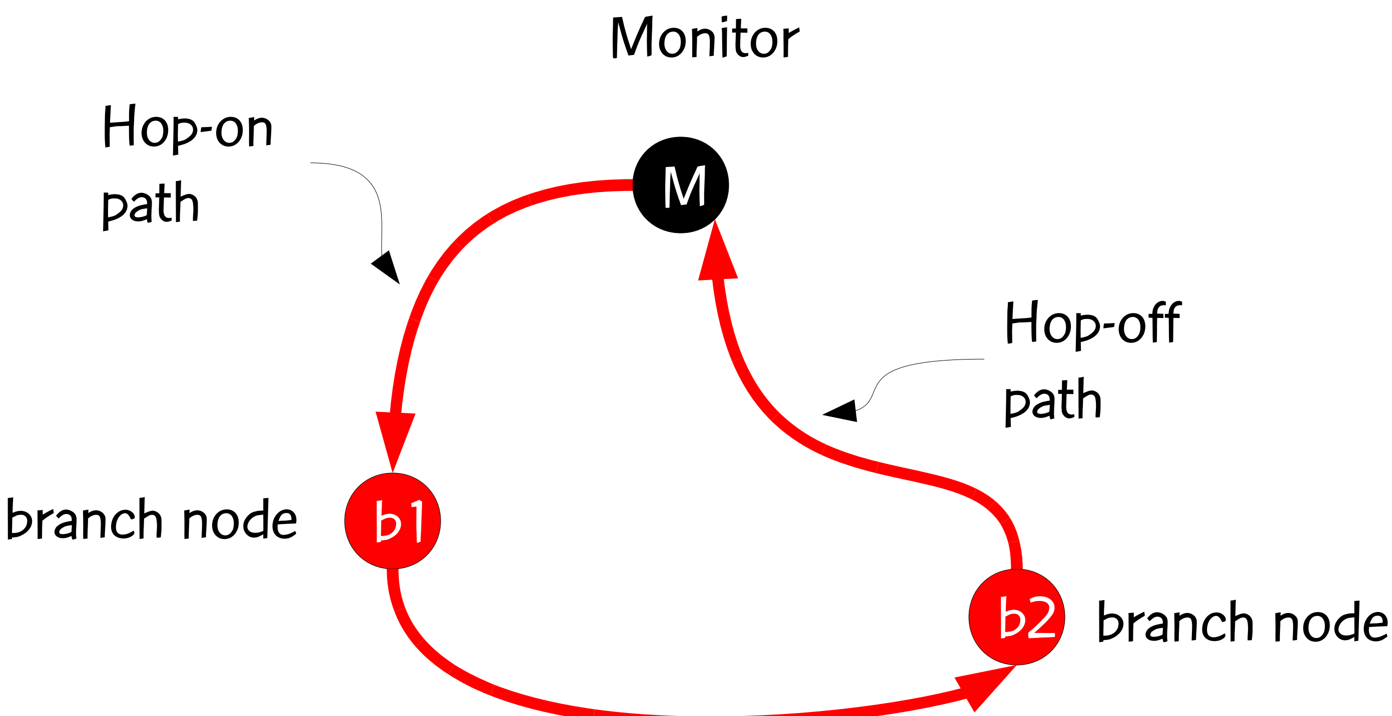
Circuit performance of (b1,b2) seen at M is a composition of the performances of (M,b1), (b1,b2), and (b2,M).

IP Monitoring

- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.

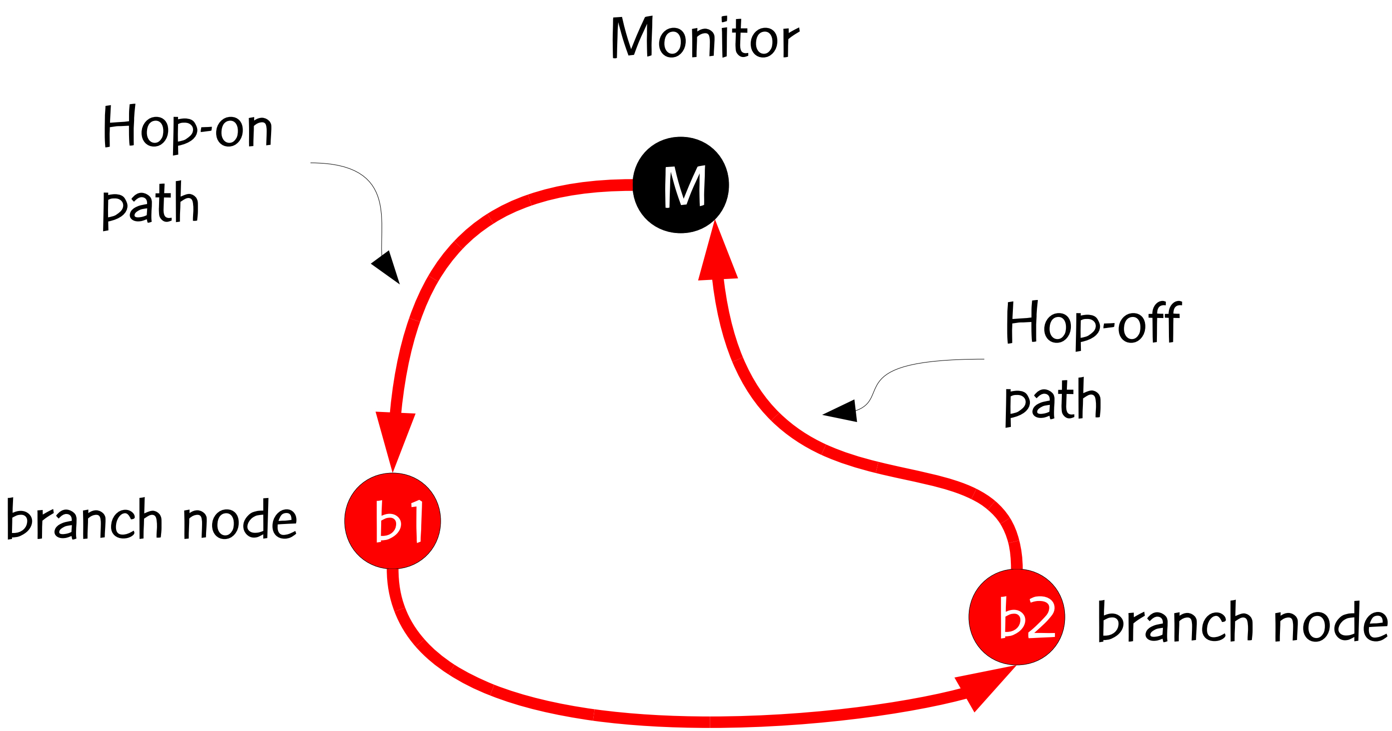


IP Monitoring



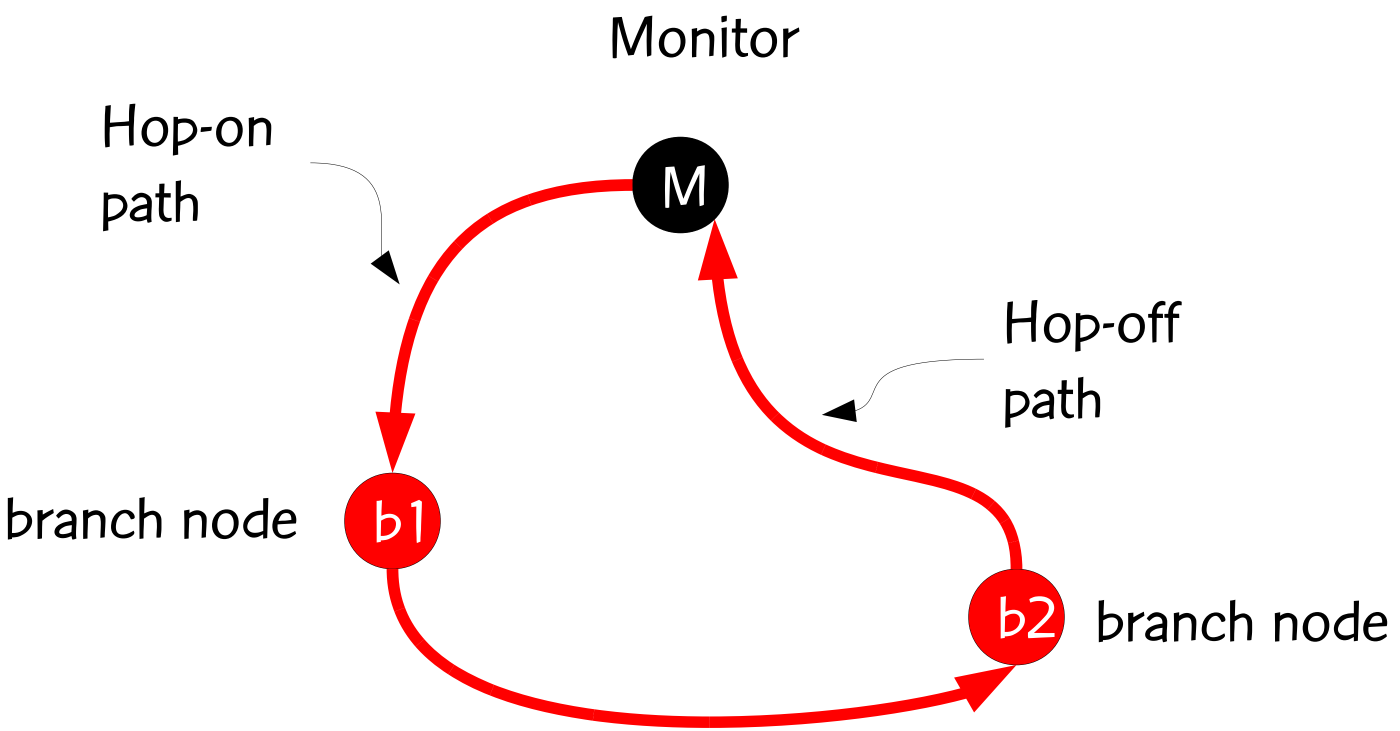
B = “branch nodes” $\subseteq$ V. We want to measure performance (e.g. loss rate) on some directed paths between vertices in B

IP Monitoring



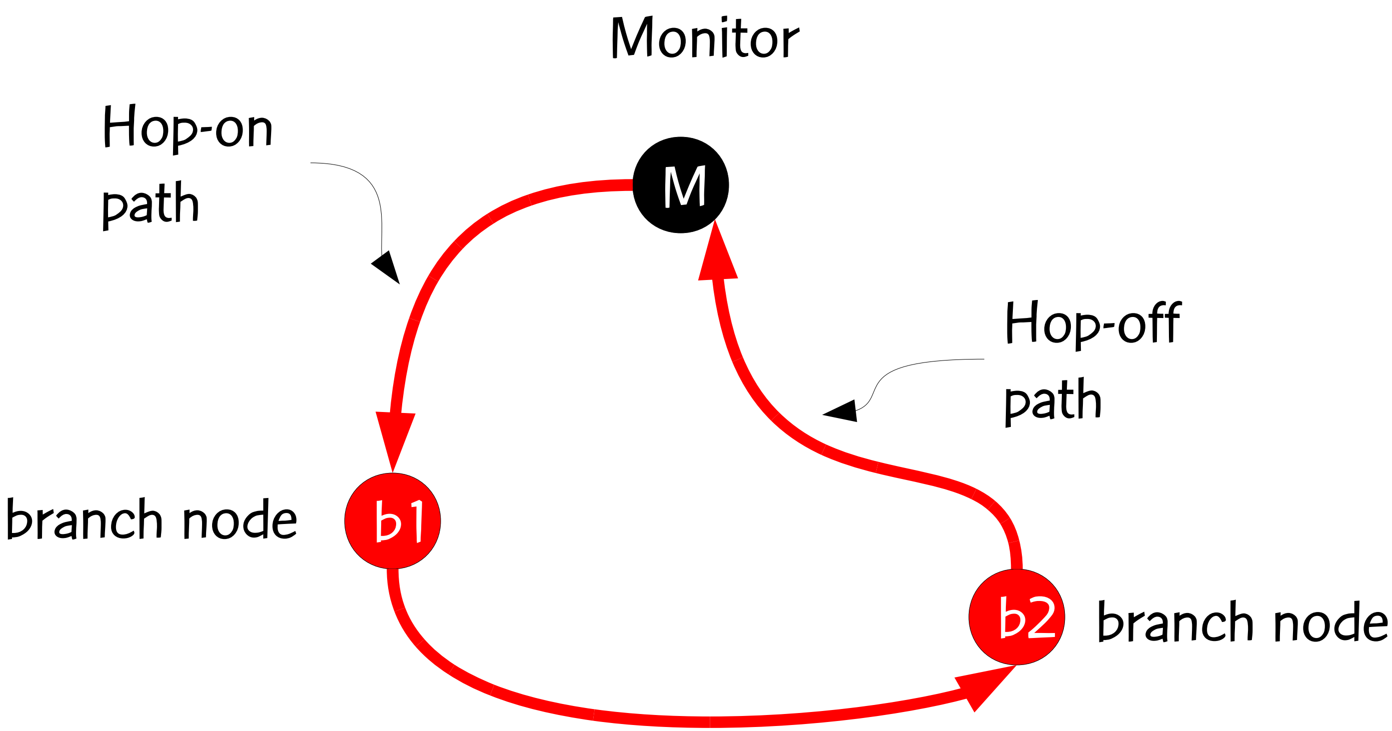
IDEA: Establish a monitoring node M. For some pairs b1, b2 $\in$ B, send packet M to b1 to b2 to M.

IP Monitoring



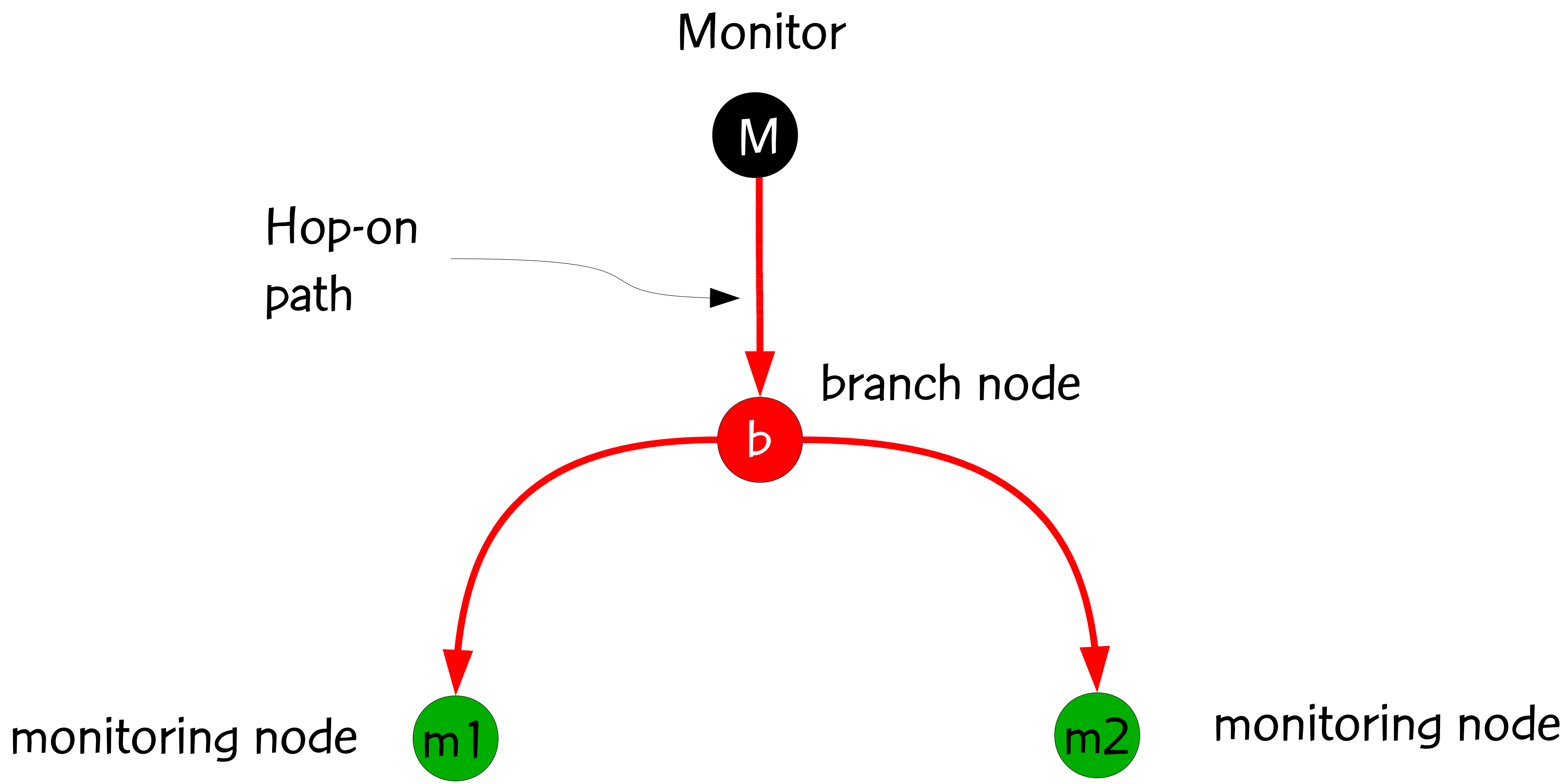
We can measure the "overall" loss rate. Must factor out the hop-on and hop-off. How?

IP Monitoring



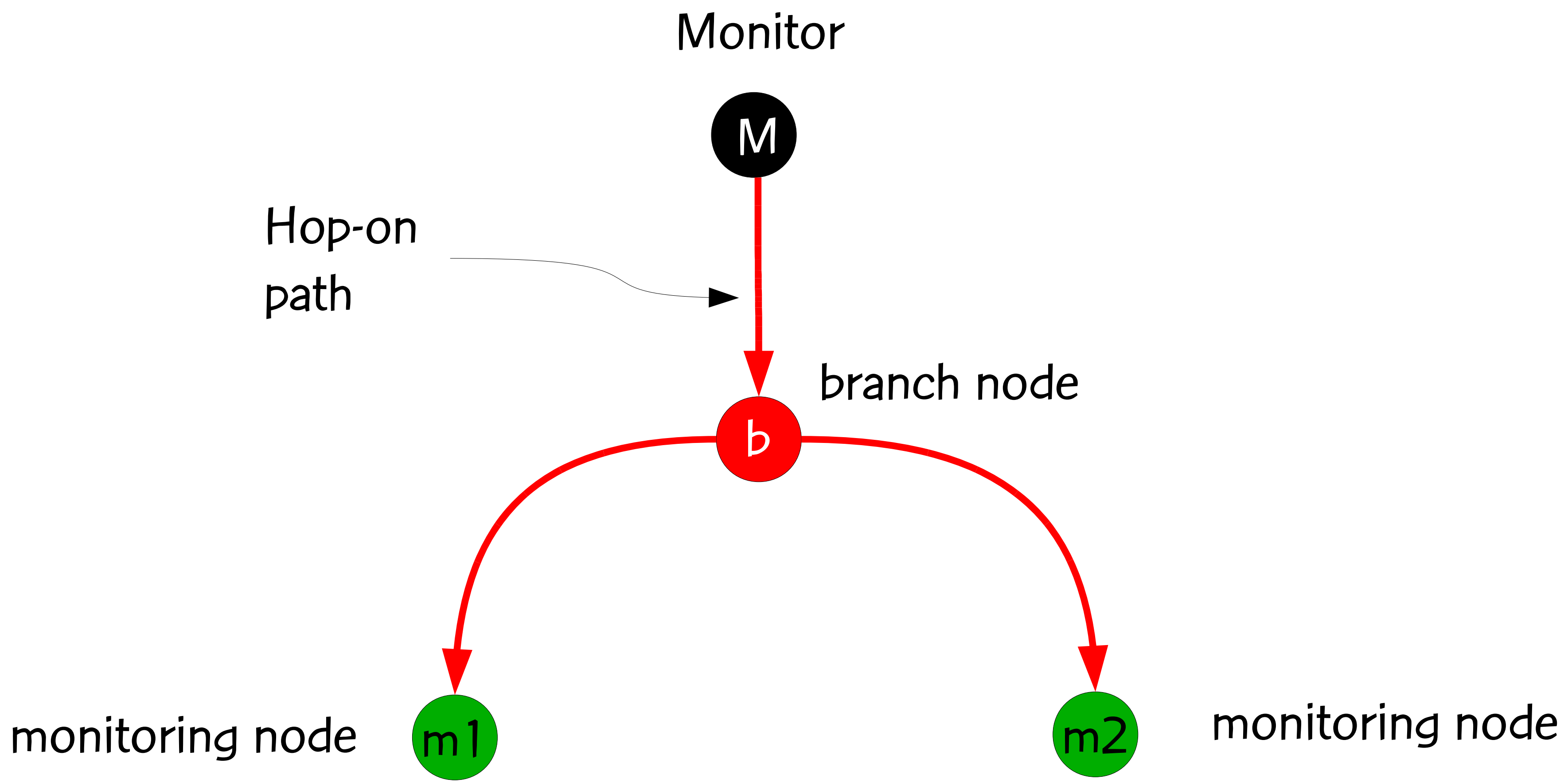
Want "disjoint" paths for independence. Must estimate loss rates for hop-on path and hop-off path to factor them out.

Estimating hop-on path loss



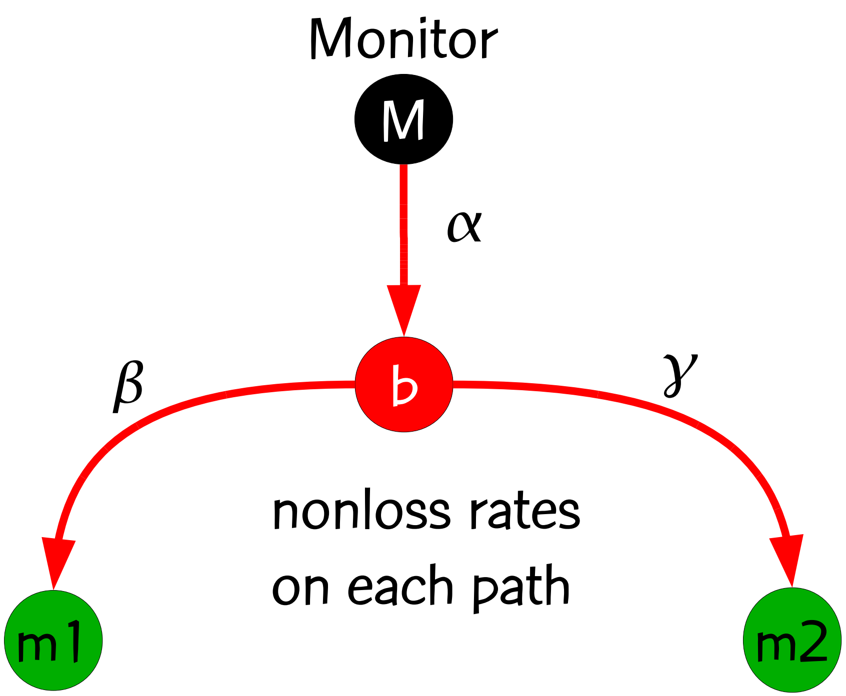
Find two "monitoring" nodes m1 and m2 and send packets from M to b and from b to m1 and m2.

Estimating hop-on path loss



What fraction of packets arrive at: 1) both m1 and m2? (p11);
2) m1, but not m2? (p10); 3) m2, but not m1? (p01)

Estimating hop-on path loss



If the three paths are arc-disjoint, estimate nonloss rate α on hop-on path $M \rightarrow b$ as follows:

$p11 = \alpha\ \beta\ \gamma$

$p10 = \alpha\ \beta\ (1-\gamma)$

$p01 = \alpha\ (1-\beta)\ \gamma$

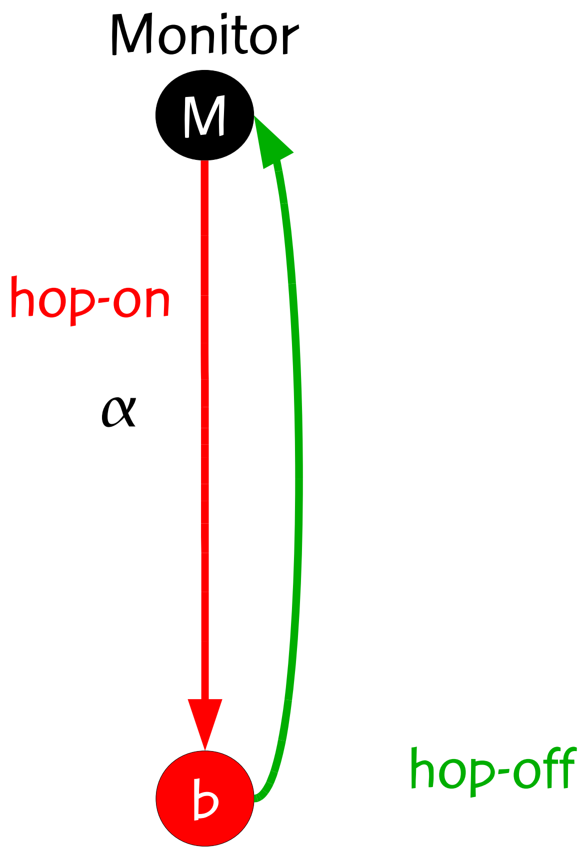
$p11 + p10 = \alpha\ \beta$

$p11 + p01 = \alpha\ \gamma$

Therefore:

$\alpha = (p11+p10)(p11+p01)\ /\ p11$

Estimating hop-off path loss



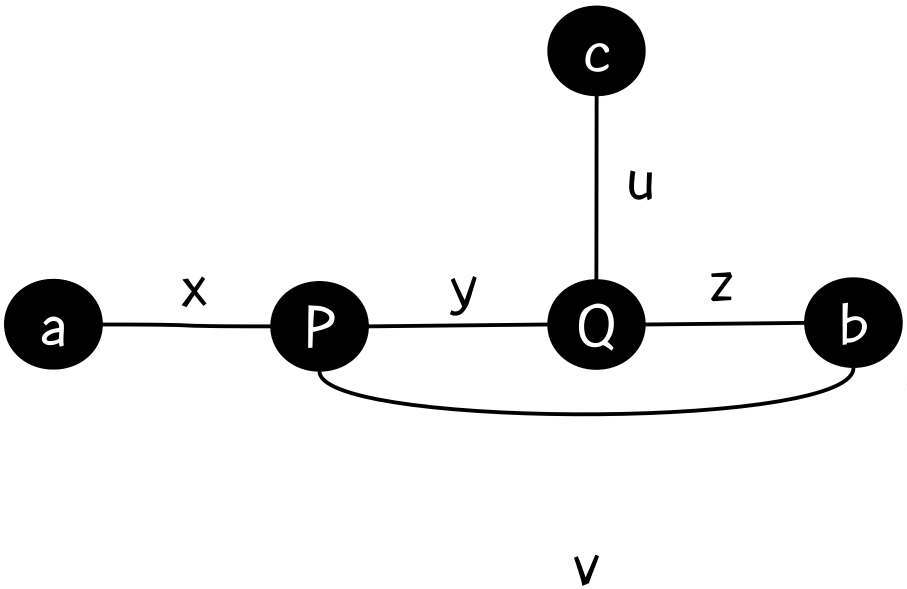
To estimate loss rate on hop-off path $b \longrightarrow M$, send packet $M \longrightarrow b \longrightarrow M$. Since we have already loss rate estimate α for hop-on path $M \longrightarrow b$, we can estimate loss rate for $b \longrightarrow M$ from roundtrip loss rate,

if path $M \longrightarrow b$ is arc-disjoint from path $b \longrightarrow M$.

Simple lemma

LEMMA:

If weight (u,v) = weight (v,u) > 0 for all u,v $\in$ V, then for all nodes a, b, c, shortest a $\rightarrow$ b and b $\rightarrow$ c paths are (directed) arc disjoint.



PROOF (by contradiction):

Suppose shortest paths are

a $\rightarrow$ P $\rightarrow$ Q $\rightarrow$ b and b $\rightarrow$ P $\rightarrow$ Q $\rightarrow$ c

clearly $v \geq y + z$

hence $z \leq v - y$

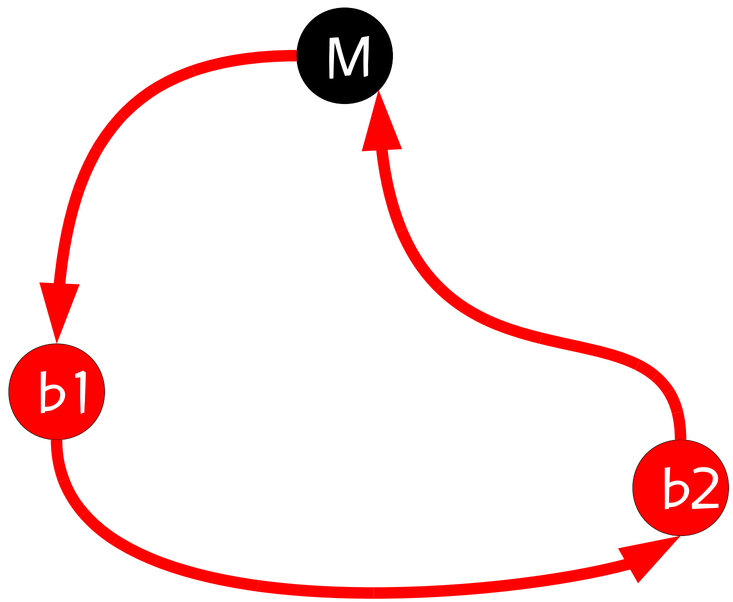
and $z < v + y$ because $y > 0$.

So b $\rightarrow$ Q $\rightarrow$ c is shorter than

b $\rightarrow$ P $\rightarrow$ Q $\rightarrow$ c !!!

Consequence of simple lemma

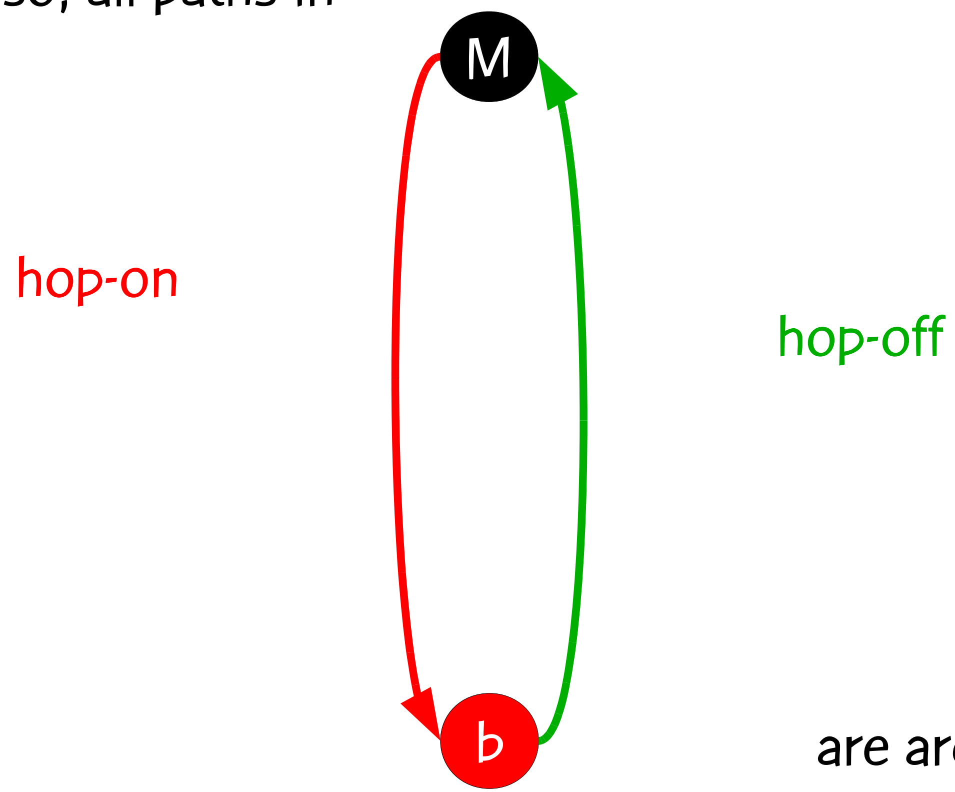
In practice, all or almost all arc weights are symmetric. If so, all paths in



are arc disjoint.

Consequence of simple lemma

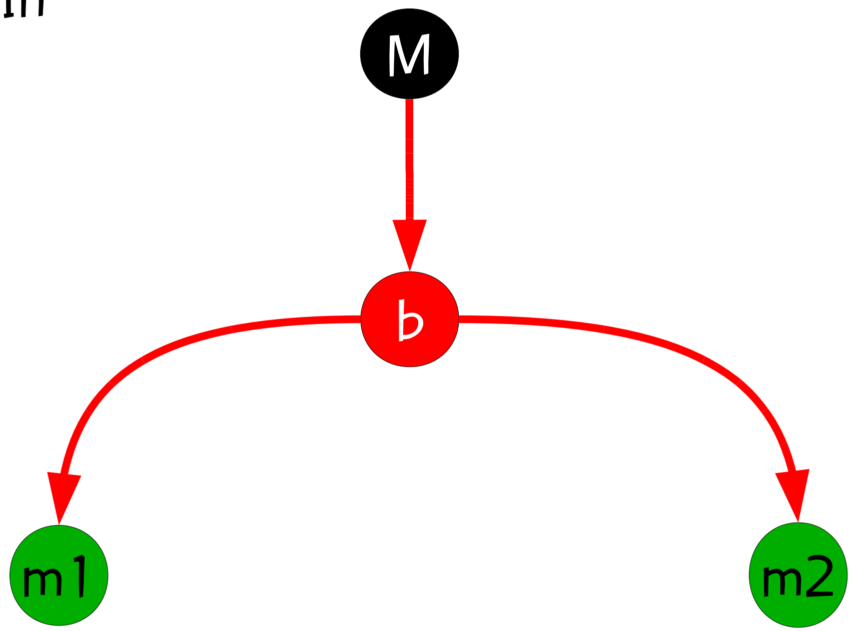
In practice, all or almost all arc weights are symmetric. If so, all paths in



are arc disjoint.

Consequence of simple lemma

In



The M $\rightarrow$ b and b $\rightarrow$ m1 paths are arc disjoint, as are the M $\rightarrow$ b and b $\rightarrow$ m2 paths.

How about b$\rightarrow$ m1 and b $\rightarrow$ m2 path?

Not disjoint in general.

Minimum monitoring set problem

Monitor placement

GOAL: Choose a small subset **S** of given set **M** of potential monitoring nodes such that

for every $b \in$ **B**, there exist $m1, m2 \in$ **S** ($m1 \neq m2$) such that

every shortest $b \longrightarrow m1$ path is vertex-disjoint from every shortest $b \longrightarrow m2$ path

Monitor placement

GOAL: Choose a small subset **S** of given set **M** of potential monitoring nodes such that

for every $b \in$ **B**, there exist $m1, m2 \in$ **S** ($m1 \neq m2$) such that

every shortest $b \rightarrow m1$ path is vertex-disjoint from every shortest $b \rightarrow m2$ path

Why every shortest path?

Because OSPF routing protocol will choose a shortest path, but we do not know which one.

Monitor placement

GOAL: Choose a small subset **S** of given set **M** of potential monitoring nodes such that

for every $b \in$ **B**, there exist $m1, m2 \in$ **S** ($m1 \neq m2$) such that

every shortest $b \rightarrow m1$ path is vertex-disjoint from every shortest $b \rightarrow m2$ path

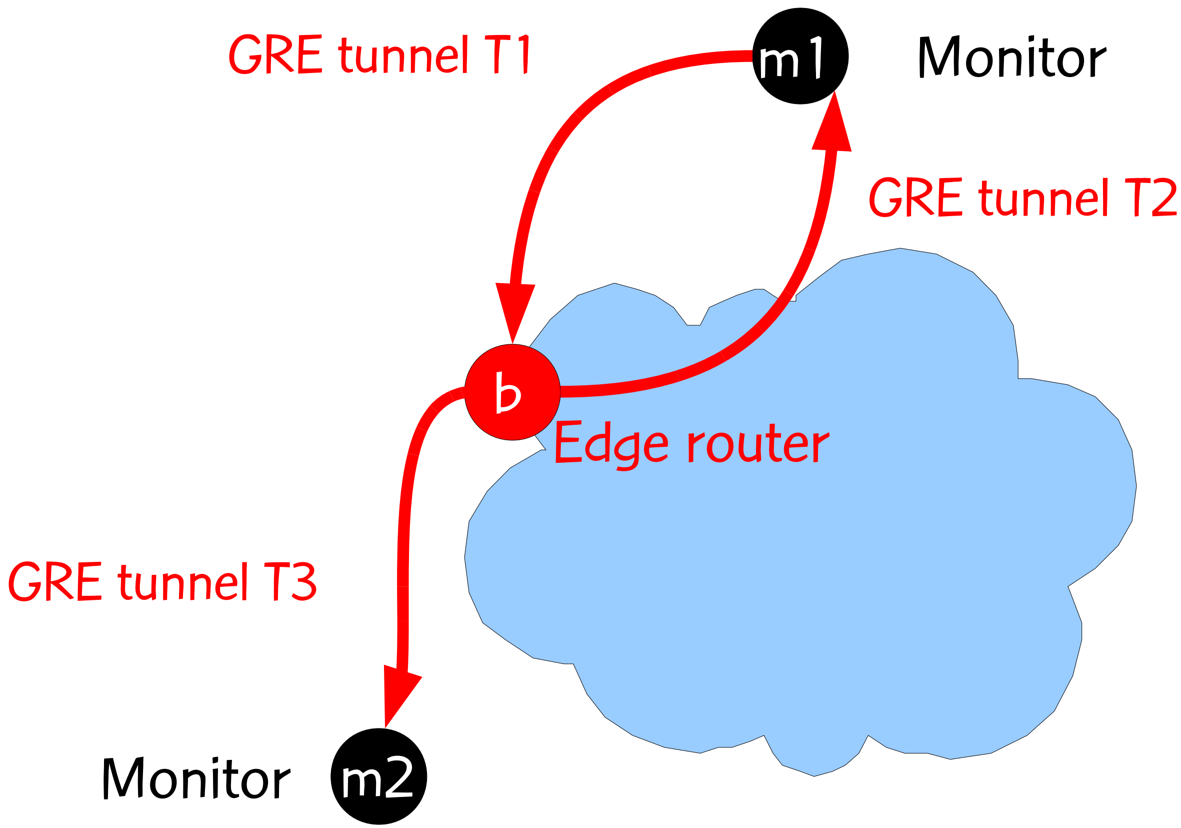
Why every shortest path?

Because OSPF routing protocol will choose a shortest path, but we do not know which one.

Obs: weights need not be symmetric.

IP Monitoring

Gu et al. (2008) propose a technique based on network tomography to infer unidirectional performance on the hop-on and hop-off paths.

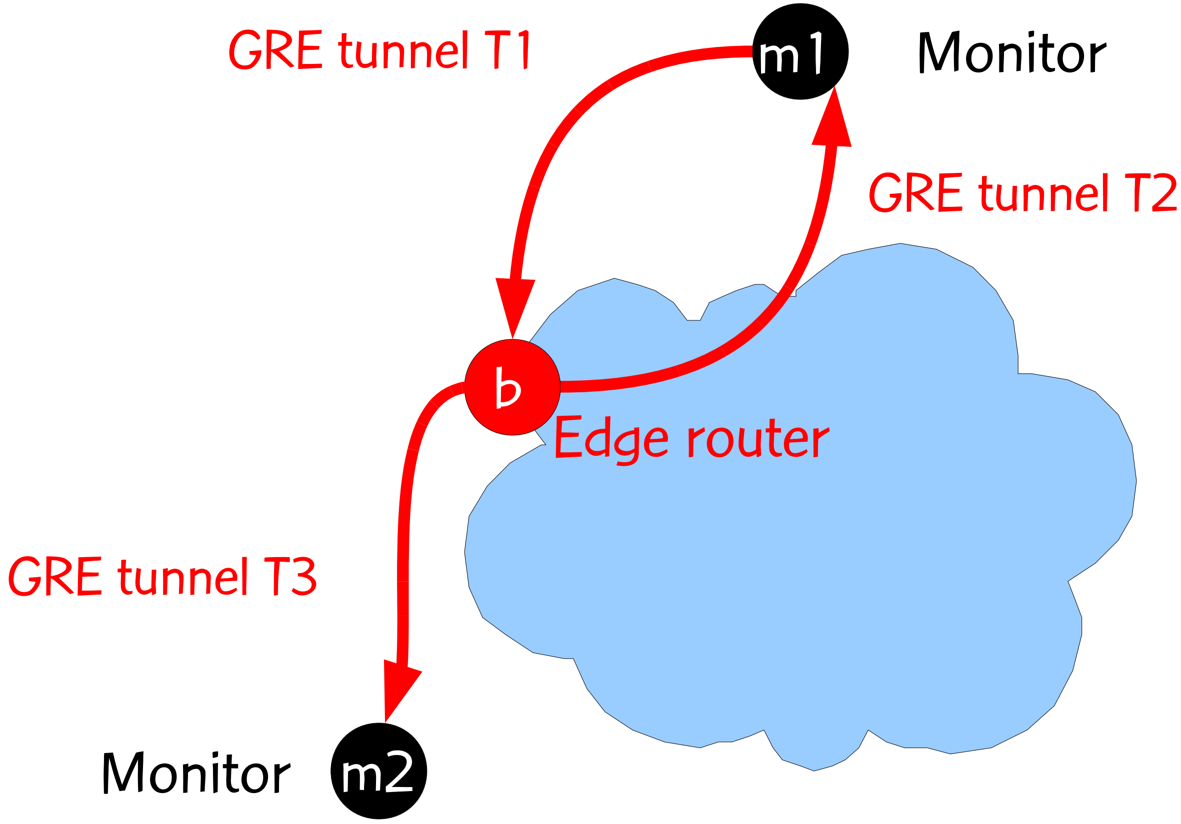


Two monitors and three GRE tunnels make up the multicast overlay topology.

Probe is dispatched from m1 to b via T1, multicast routing at b send copies back to m1 via T2 and to m2 via T3.

IP Monitoring

Gu et al. (2008) propose a technique based on network tomography to infer unidirectional performance on the hop-on and hop-off paths.



It is worth noting that native multicast support is by now a standard router capability.

After a relatively slow start, multicast services are now readily available in provider backbones.

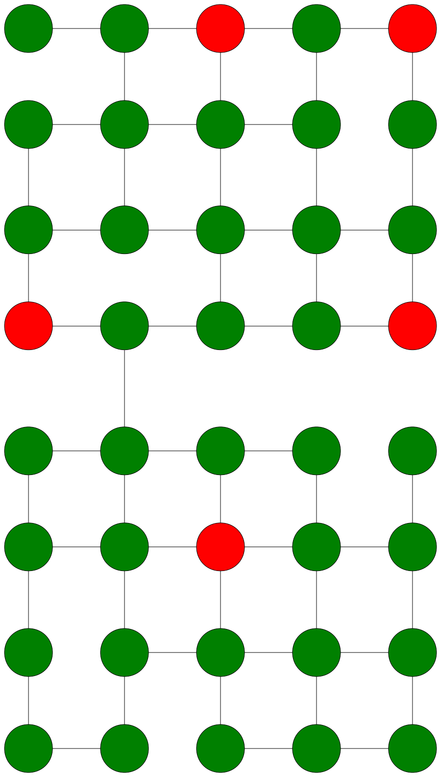
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



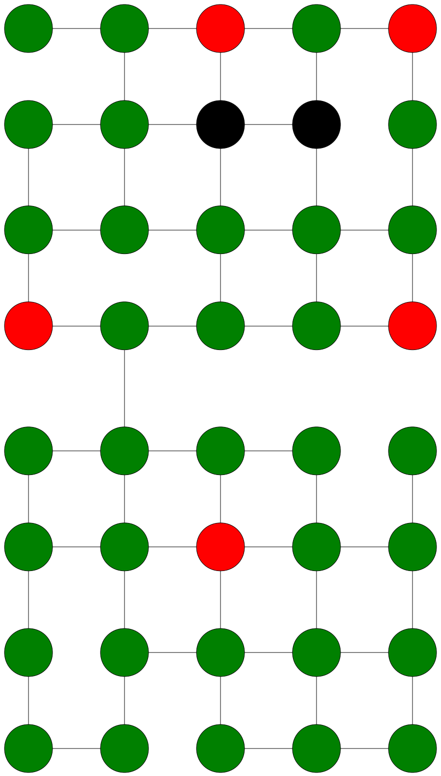
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



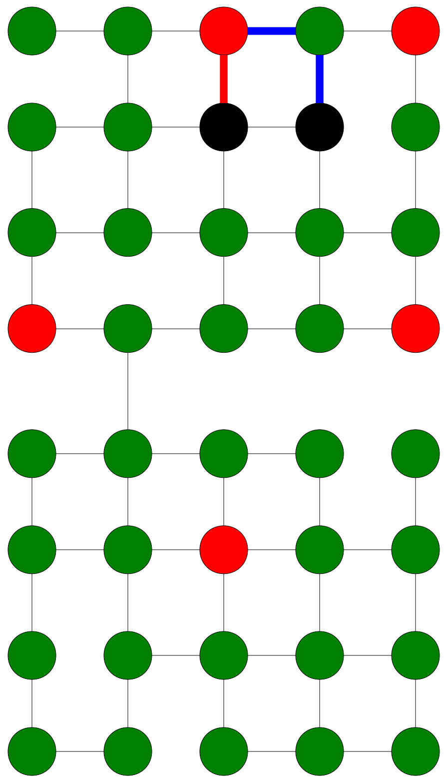
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



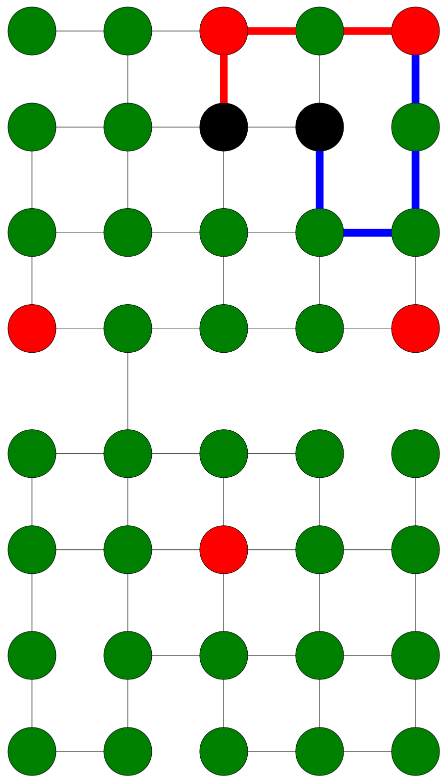
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



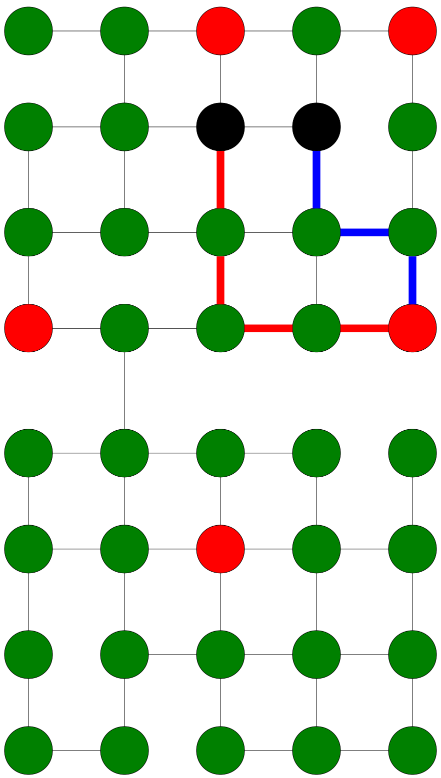
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



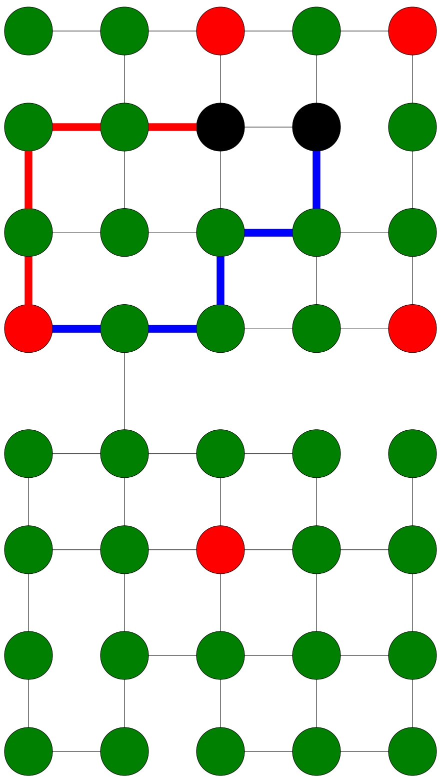
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed

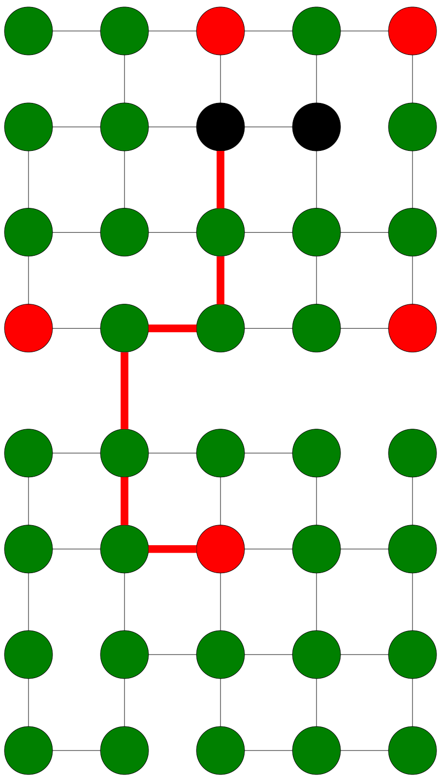


Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

- Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.
- One objective is to minimize N.

paths are given and are fixed



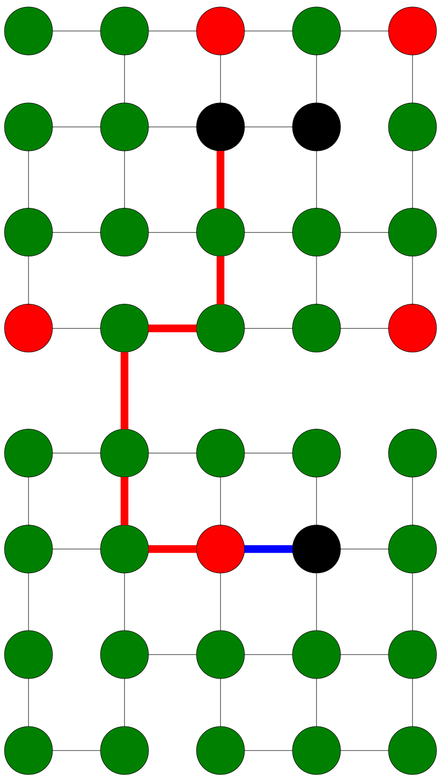
Minimum monitoring set problem

We wish to perform the tomographic inference of hop-on and hop-off performance for each provider edge router:

Deploy a set of N measurement hosts $\{M_1, M_2, \ldots, M_N\}$ such that for each provider edge router b, there are two measurement hosts M_i and M_j such that the physical paths (b, M_i) and (b, M_j) are disjoint.

One objective is to minimize N.

paths are given and are fixed



Set covering with pairs

- Set covering with pairs (SCP) was introduced by Hassin & Segev (2005):
  - GIVEN a ground set X of elements and a set Y of cover items, and for each $x \in X$ a set P_x of pairs of items in Y that cover x. A subset $Y' \subseteq Y$ covers X if for each $x \in X$ one of the pairs in P_x is contained in Y', FIND a minimum-size covering subset.
- SCP is NP-hard and, unless $P = NP$, is hard to approximate.

Minimum monitoring set problem

- The MMS problem is a special case of SCP. We prove that:
  - Let R(w,u) be the set of all routes from w to u
  - MMS is at least as hard to approximate as SCP, even if:
    - Each set R(w,u) is the set of all shortest paths from w to u;
    - Each set R(w,u) contains only one item, and that is a shortest path from w to u
- However, if we allow arbitrary disjoint paths, then using dynamic programming, the problem can be solved in O(|V|+|E|) time.

Another application: Redundant content distribution

Suppose nodes b_1, b_2, ... want some content (e.g. video).

We want a small set **S** of servers such that:

for every b_i there exist m_1, $m_2 \in$ **S** both of which can provide content to b_i

and all paths $m_1 \rightarrow b$ are disjoint with all paths $m_2 \rightarrow b$

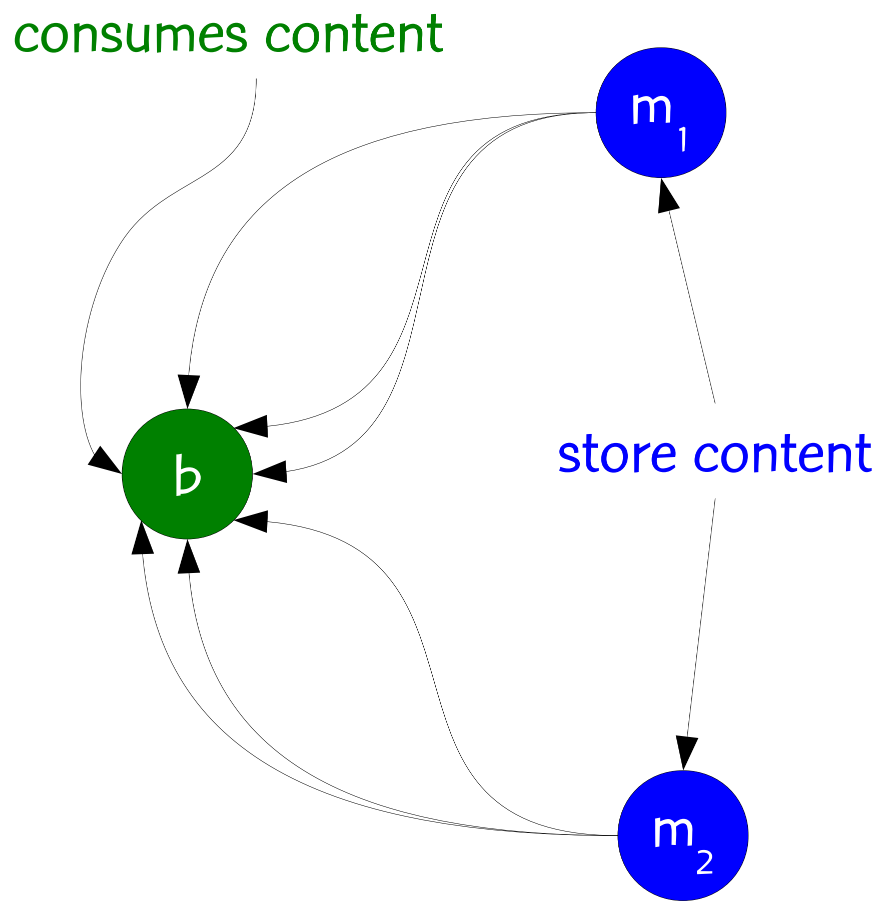
Another application: Redundant content distribution

Suppose nodes b_1, b_2, ... want some content (e.g. video).

We want a small set **S** of servers such that:

for every b_i there exist m_1, $m_2 \in$ **S** both of which can provide content to b_i

and all paths $m_1 \to b$ are disjoint with all paths $m_2 \to b$



Algorithms for minimum monitoring set problem

Algorithms for MMS problem

- Exact integer programming model
- Dynamic programming for arbitrary paths variant
- Greedy heuristic
- Genetic algorithm (heuristic)
- Double hitting set heuristic (DHS)
- Lower bound derived from DHS

Algorithms for MMS problem

- Exact integer programming model
- Dynamic programming for arbitrary paths variant
- Greedy heuristic
- Genetic algorithm (heuristic)
- Double hitting set heuristic (DHS)
- Lower bound derived from DHS

Integer programming

Integer programming model

for every potential monitoring node $v \in M$, let binary variable

$$x_v = 1 \text{ iff node } v \text{ is chosen}$$

Integer programming model

for every potential monitoring node $v \in M$, let binary variable

$$x_v = 1 \text{ iff node } v \text{ is chosen}$$

for each pair $\{u,v\}$ of potential monitoring nodes ($u < v$) define continuous variable $y_{u,v}$ such that

$$y_{u,v} \leq x_u$$

$$y_{u,v} \leq x_v$$

$y_{u,v} > 0$ then $x_u = x_v = 1$

Integer programming model

for every potential monitoring node $v \in M$, let binary variable

$x_v = 1$ iff node v is chosen

for each pair {u,v} of potential monitoring nodes ($u < v$) define continuous variable $y_{u,v}$ such that

$y_{u,v} \leq x_u$

$y_{u,v} \leq x_v$

$y_{u,v} > 0$ then $x_u = x_v = 1$

for each branch node b that is not a potential monitoring node:

$\Sigma\, y_{u,v} \geq 1$ (summed over all pairs {u,v} that cover b ($u < v$))

Integer programming model

for every potential monitoring node $v \in M$, let binary variable

$$x_v = 1 \text{ iff node } v \text{ is chosen}$$

for each pair $\{u,v\}$ of potential monitoring nodes $(u < v)$ define continuous variable $y_{u,v}$ such that

$$y_{u,v} \leq x_u$$

$$y_{u,v} \leq x_v$$

$y_{u,v} > 0$ then $x_u = x_v = 1$

for each branch node b that is not a potential monitoring node:

$\Sigma\, y_{u,v} \geq 1$ (summed over all pairs $\{u,v\}$ that cover b $(u < v)$)

for each branch node $b \in B \cap M$

$x_b + \Sigma\, y_{u,v} \geq 1$ (summed over all pairs $\{u,v\}$ that cover b $(u < v)$)

Integer programming model

$\min \sum x_v$

for every potential monitoring node $v \in M$, let binary variable

$x_v = 1$ iff node v is chosen

for each pair {u,v} of potential monitoring nodes (u < v) define continuous variable $y_{u,v}$ such that

$y_{u,v} \leq x_u$

$y_{u,v} \leq x_v$

$y_{u,v} > 0$ then $x_u = x_v = 1$

for each branch node b that is not a potential monitoring node:

$\sum y_{u,v} \geq 1$ (summed over all pairs {u,v} that cover b (u < v))

for each branch node $b \in B \cap M$

$x_b + \sum y_{u,v} \geq 1$ (summed over all pairs {u,v} that cover b (u < v))

Greedy algorithm

Greedy algorithm for MMS problem

Greedy algorithm for MMS problem

- initialize partial cover $S = \{ \}$

Greedy algorithm for MMS problem

- initialize partial cover S = { }
- while S is not a cover do:

Greedy algorithm for MMS problem

- initialize partial cover $S = \{\ \}$
- while S is not a cover do:
  - find $m \in M \setminus S$ such that $S \cup \{m\}$ covers a maximum number of additional branch nodes (break ties by vertex index) and set $S = S \cup \{m\}$

Greedy algorithm for MMS problem

- initialize partial cover $S = \{\ \}$
- while S is not a cover do:
  - find $m \in M \setminus S$ such that $S \cup \{m\}$ covers a maximum number of additional branch nodes (break ties by vertex index) and set $S = S \cup \{m\}$
  - if no $m \in M \setminus S$ yields an increase in coverage, then choose a pair $\{m_1,m_2\} \in M \setminus S$ that yields a maximum increase in coverage and set $S = S \cup \{m_1\} \cup \{m_2\}$

Greedy algorithm for MMS problem

- initialize partial cover $S = \{\ \}$
- while S is not a cover do:
  - find $m \in M \setminus S$ such that $S \cup \{m\}$ covers a maximum number of additional branch nodes (break ties by vertex index) and set $S = S \cup \{m\}$
  - if no $m \in M \setminus S$ yields an increase in coverage, then choose a pair $\{m_1,m_2\} \in M \setminus S$ that yields a maximum increase in coverage and set $S = S \cup \{m_1\} \cup \{m_2\}$
  - if no pair exists, then the problem is infeasible

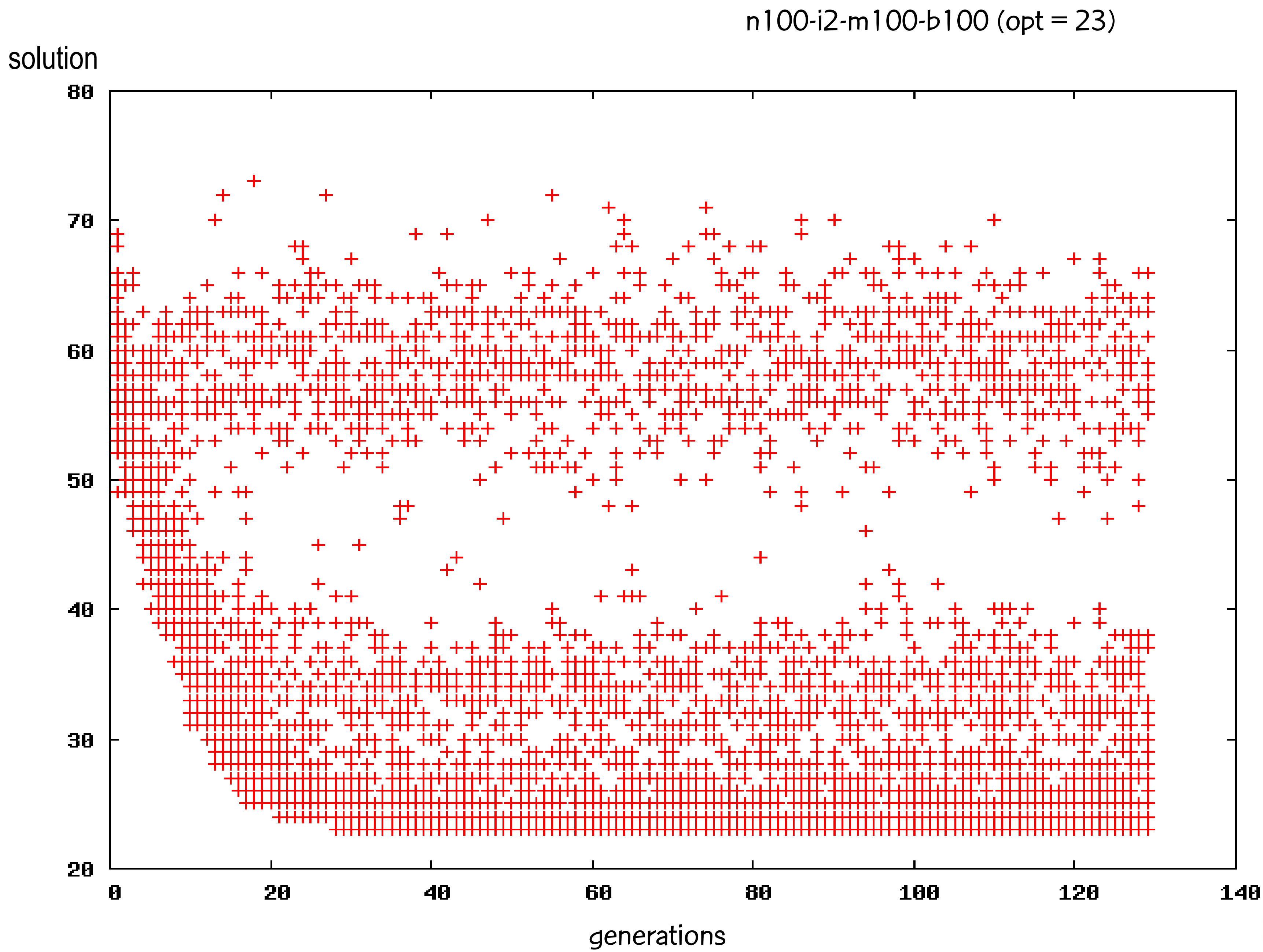
BRKGA for the MMS problem

BRKGA for the MMS problem

- Chromosome:
  - A vector X of N random 0-1values (random keys), where N is the number of potential monitoring nodes. The i-th random key corresponds to the i-th monitoring node.
- Decoder:
  - For i = 1,N: if X(i) = 1, add i-th monitoring node to solution
  - If solution is feasible, i.e. all customer nodes are covered: STOP
  - Else, apply greedy algorithm to cover uncovered branch nodes.

BRKGA for the MMS problem

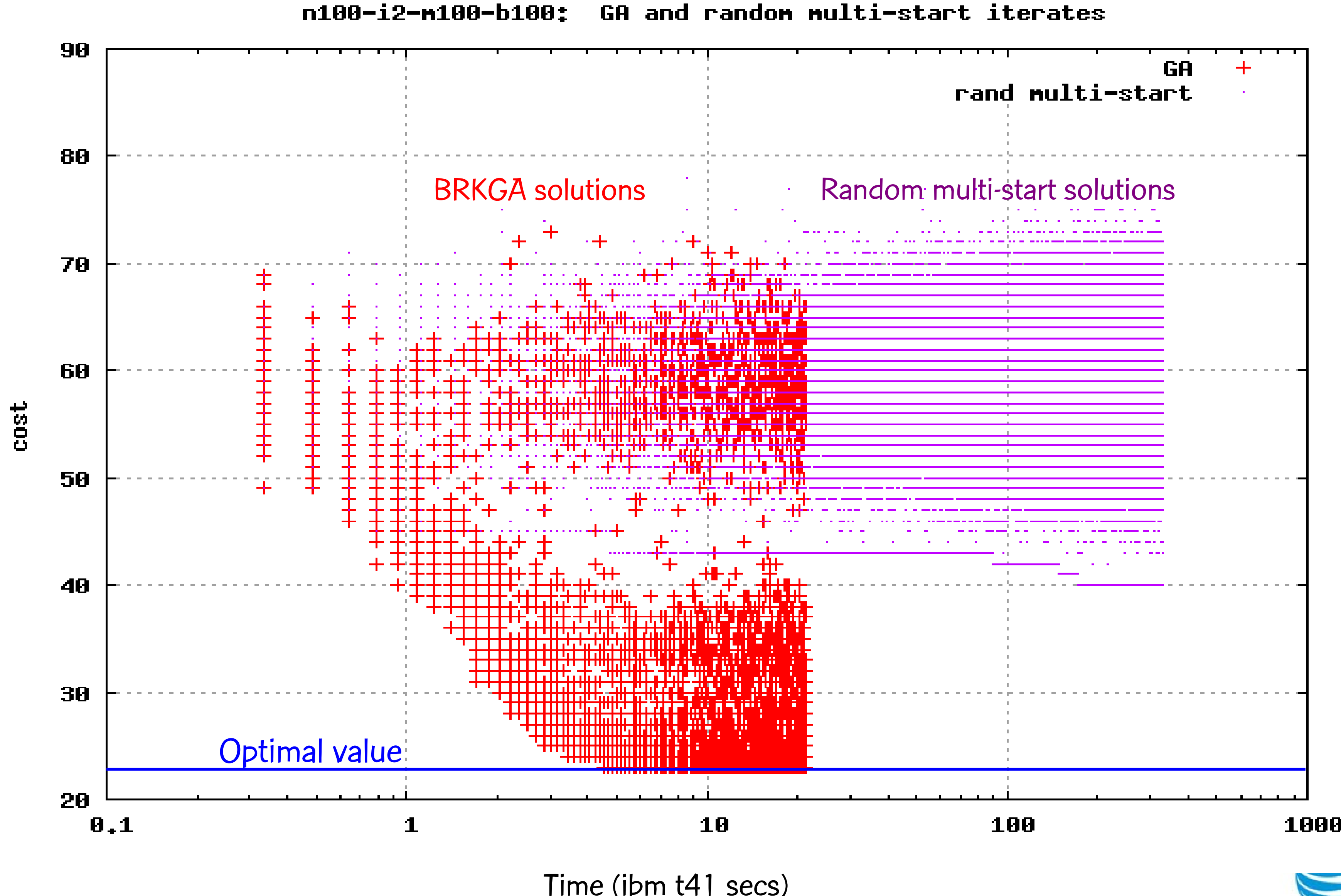
- Size of population: N (number of monitoring nodes)
- Size of elite set: 15% of N
- Size of mutant set: 10% of N
- Biased coin probability: 70%
- Stop after N generations without improvement of best found solution



n100-i2-m100-b100 (opt = 23)
solution
80
70
60
50
40
30
20
0
20
40
60
80
100
120
140
generations

solution

n100-i2-m100-b100 (opt = 23)



Lower bound

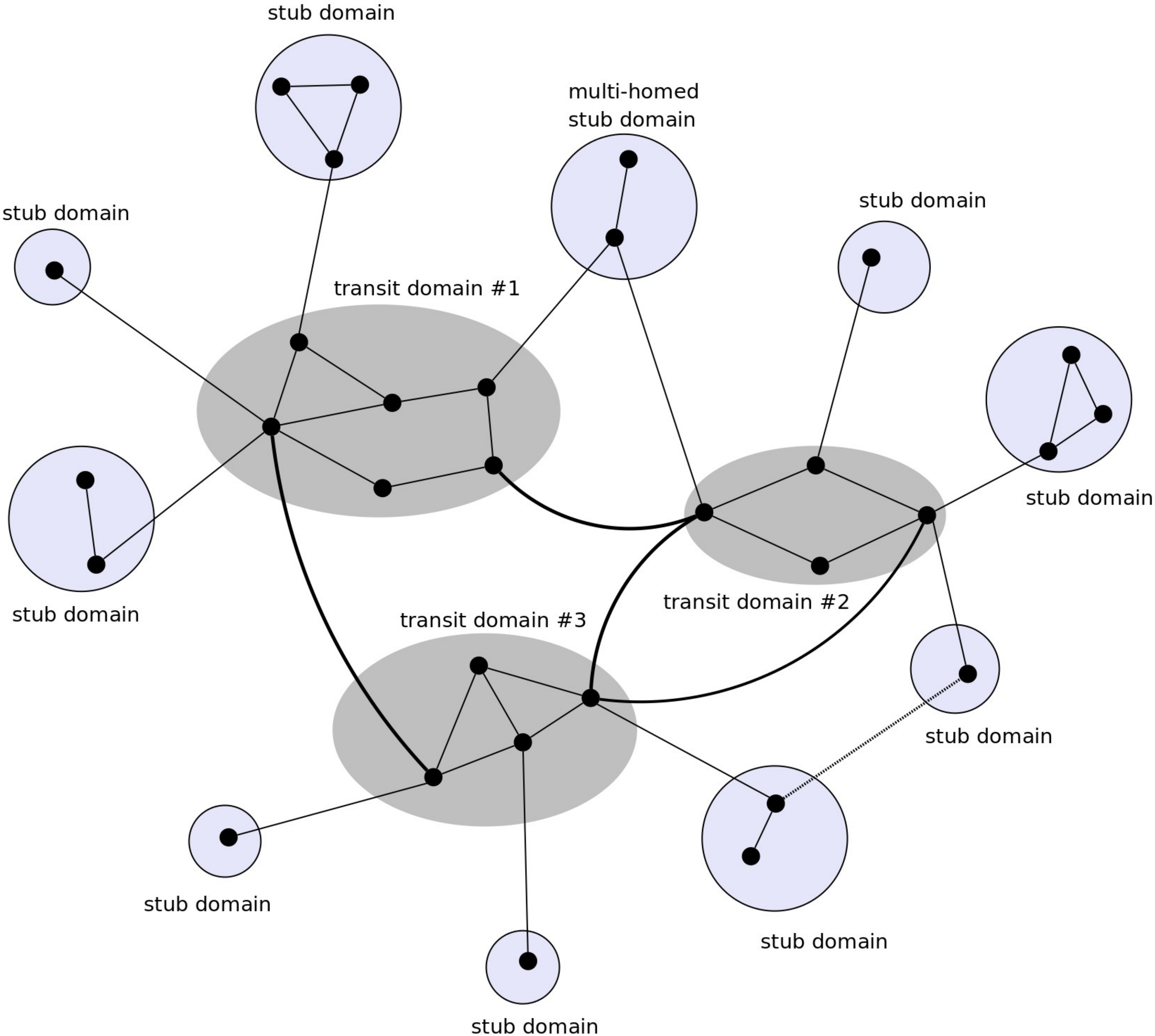
Lower bound on OPT

- OPT for monitor placement $\geq$ OPT for the 2^{nd} hitting set problem
- We can solve the 2^{nd} hitting set instance optimally using CPLEX
- On our test instances, bounds are quite tight

Experimental results

Experimental results

- 560 synthetic instances, with 25, 50, 100, 190, 220, 250, 300, and 558 nodes and varying sizes of potential monitoring nodes and branch nodes.
  - Largest 2-connected component in any of the synthetic instances contained 34% of the nodes and the largest instance had only 10% of the nodes.
- 65 real-world instances derived from five large scale Tier 1 ISP backbone networks and using real OSPF weights. These networks ranged in size from a little more than 100 routers to nearly 1000 routers.
  - Largest 2-connected component had at least 84% of the nodes.


stub domain
multi-homed
stub domain
stub domain
stub domain
transit domain #1
stub domain
stub domain
transit domain #2
transit domain #3
stub domain
stub domain
stub domain
stub domain
stub domain

Experimental results

- Integer program (CPLEX) could only solve instances with up to 100 nodes. This is in contrast to "classical" set covering where much larger instance are solved easily.
- On the other hand, the 2nd hitting set problem could be easily solved to optimality using CPLEX. Lower bounds were produced for all test instances.
- DHS and GREEDY are both much faster than GA. On some of the largest instances (about 1000 routers) DHS and GREEDY took one hour while GA took 10 days. GA can be sped up with trivial parallel implementation.

Synthetic networks

- CPLEX solved 324 of 560 instances to OPT
- Heuristics found optimal solutions for some of those instances:
  - Greedy algorithm: 59/324 = 18.2%
  - Double hitting set algorithm: 65/324 = 20.0%
  - Genetic algorithm: 318/324 = 98.1%

Synthetic networks

- CPLEX computed lower bounds for all 560 instances
- Heuristics matched the lower bound for some of those instances:
  - Greedy algorithm: 236/560 = 42.1%
  - Double hitting set algorithm: 363/560 = 64.8 %
  - Genetic algorithm: 394/560 = 70.4%

Synthetic networks: comparing heuristic solutions

- Double hitting set (DHS) vs Greedy
  - DHS better than Greedy: 456/560 = 81.4%
  - DHS equal to Greedy: 90/560 = 16.1%
  - Greedy better than DHS: 14/560 = 2.5%
- Genetic algorithm (GA) vs DHS
  - GA better than DHS: 68/560 = 12.1%
  - GA equal to DHS: 482/560 = 86.1%
  - DHS better than GA: 10/560 = 1.8%
- GA vs Greedy
  - GA better than Greedy: 487/560 = 87.0%
  - GA equal to Greedy: 73/560 = 13.0%
  - Greedy better than GA: 0/560 = 0%

Synthetic networks

- CPLEX found optimal solutions for instances with fewer than 100 routers
- Only 20-30% of branch nodes need to be monitoring nodes.
- Greedy algorithm did not perform well.

Real networks

- CPLEX could not solve any instance to optimality.
- Lower bounds were computed for all 65 instances.
- Heuristics matched lower bounds for some of the instances:
  - Greedy: 27/65 = 41.5%
  - GA: 48/65 = 73.8%
  - DHS: 54/65 = 83.%

Real networks: comparing heuristic solutions

- Double hitting set (DHS) vs Greedy
  - DHS better than Greedy: 9/65 = 13.9%
  - DHS equal to Greedy: 54/65 = 83.1%
  - Greedy better than DHS: 2/65 = 3.1%
- Genetic algorithm (GA) vs DHS
  - GA better than DHS: 6/65 = 9.2%
  - GA equal to DHS: 54/65 = 83.1%
  - DHS better than GA: 5/65 = 7.7%
- GA vs Greedy
  - GA better than Greedy: 12/65 = 18.5%
  - GA equal to Greedy: 48/65 = 73.8%
  - Greedy better than GA: 5/65 = 7.7%

Real networks

- Too large for CPLEX
- Only 15-20% of branch nodes need to be monitoring nodes.
- Greedy algorithm did perform well. It found a solution equal to LB in 27 of the 65 instances. Matched HH on 54 instances and GA on 48.

Concluding remarks

- We constructed a number of network test instances to capture the topology and routing of large internetworks;
- We demonstrated algorithms that provide a feasible combination of accuracy and execution times;
- We showed that solutions derived from our methods provide a useful saving in the number of measurement nodes compared with the naive approach of using each branch point as a measurement node: Networks having a large number of branch nodes need only 10-30% of branch points to be measurement nodes.

Routing and wavelength assignment in optical networks

Routing and wavelength assignment (RWA)

- Objective: Route a set of connections (called lightpaths) and assign a wavelength to each of them.
- Two lightpaths may use the same wavelength, provided they do not share any common link.
- Connections whose paths share a common link in the network are assigned to different wavelengths (wavelength clash constraints).
- If no wavelength converters are available, the same wavelength must be assigned along the entire route (wavelength continuity constraints).

Routing and wavelength assignment (RWA)

- Variants of RWA are characterized by different optimization criteria, traffic patterns, and whether wavelength conversion is available or not.
- We consider the min-RWA offline variant:
  - Connection requirements are known beforehand.
  - No wavelength conversion is possible.
  - Objective is to minimize the number of wavelengths used for routing all connections.
  - Asymmetric traffic matrices and bidirectional links.
  - NP-hard (Erlebach and Jansen, 2001)

Routing and wavelength assignment (RWA)

Connections

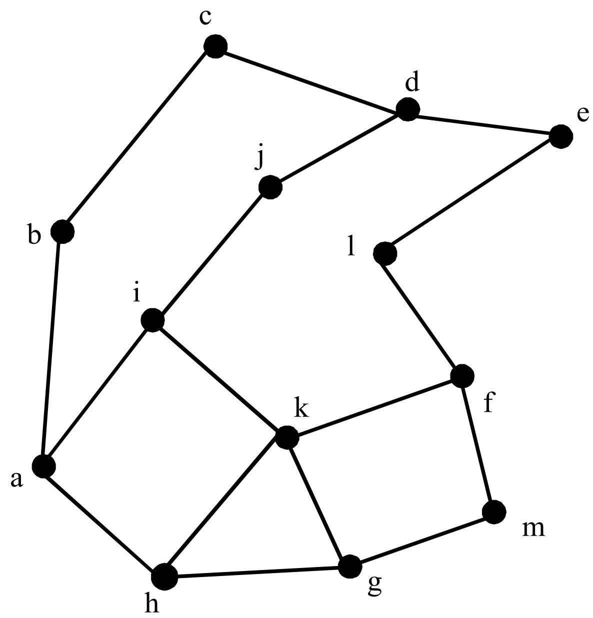
c ↔ m

d ↔ b

e ↔ h

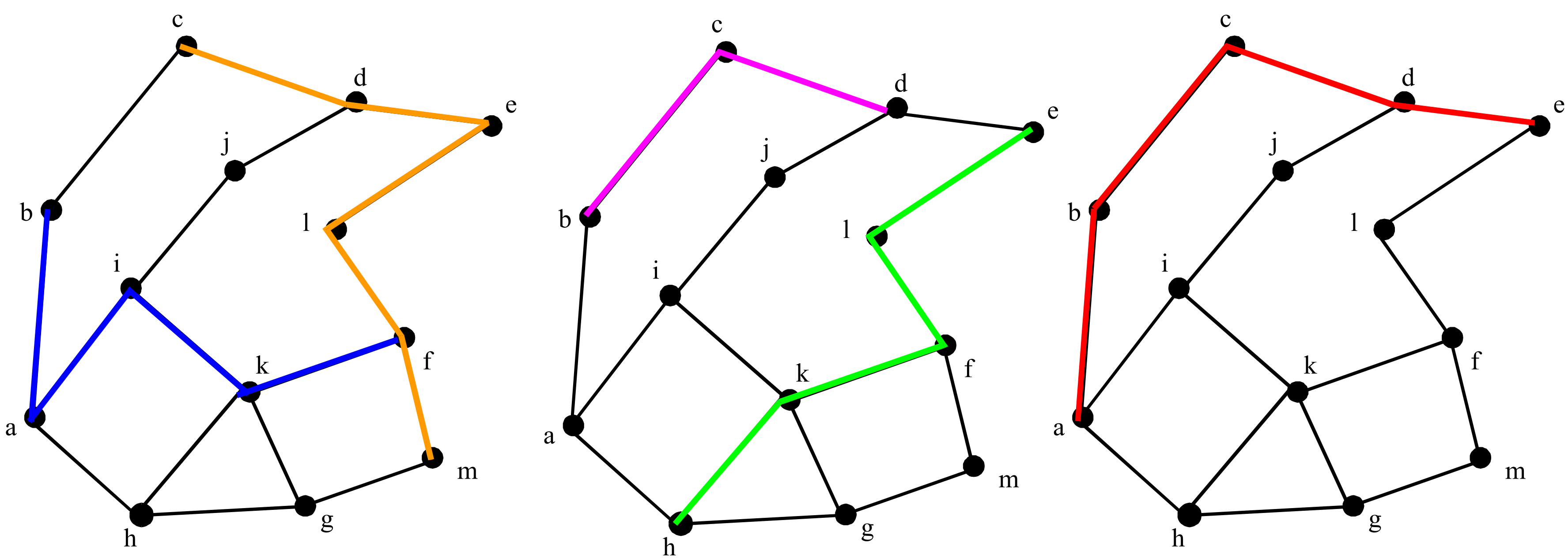
a ↔ e

b ↔ f



Routing and wavelength assignment (RWA)

Conections: (a ↔ e) (b ↔ f) (c ↔ m) (d ↔ b) (e ↔ h)



wavelength 1 wavelength 2 wavelength 3

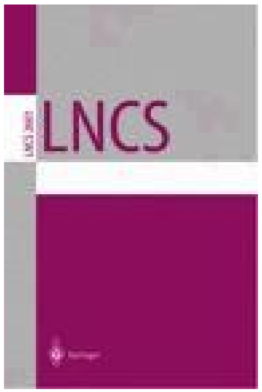
Heuristic of N. Skorin-Kapov (EJOR, 2007)

- Associates the min-RWA with the bin packing problem.
  - Wavelengths are associated with bins.
  - The capacity of a bin is defined as its number of arcs.
  - The size of a connection is defined as the number of arcs in its shortest path.
- Developed RWA heuristics based on the following classical bin packing heuristics:
  - First Fit (FF)
  - Best Fit (BF)
  - First Fit Decreasing (FFD)
  - Best Fit Decreasing (BFD)

Heuristic of N. Skorin-Kapov (EJOR, 2007)

- Associates the min-RWA with the bin packing problem.
  - Wavelengths are associated with bins.
  - The capacity of a bin is defined as its number of arcs.
  - The size of a connection is defined as the number of arcs in its shortest path.
- Developed RWA heuristics based on the following classical bin packing heuristics:
  - First Fit (FF)
  - Best Fit (BF)
  - First Fit Decreasing (FFD)
  - Best Fit Decreasing (BFD): state of the art heuristic for RWA

Efficient implementation of BFD-RWA



T.F. Noronha, M.G.C.R., and C.C. Ribeiro,

"Efficient implementations of heuristics for routing and wavelength assignment," in "Experimental Algorithms,"

7th International Workshop (WEA 2008), C.C. McGeoch (Ed.), LNCS, vol. 5038, pp. 169-180, Springer, 2008.

Tech report version:

http://www.research.att.com/~mgcr/doc/impl_rwa_heur.pdf

BFD-RWA

N. Skorin-Kapov (2007); Noronha, R., and Ribeiro (2008)

- Input:
  - A directed graph G representing the network topology.
  - A set T of connection requests.
  - The value *d* of of the maximum number of arcs in each route. It is set to be the maximum of the square root of the number of links in the network and the diameter of G.
- Starts with only one copy of G (called G_1).
- Connections are selected according to non-increasing order of the lengths of their shortest paths in G_i. Ties are broken at random.
- The connection is assigned wavelength *i* , and the arcs along path are deleted from G_i.
- If no existing bin can accommodate the connection with fewer than *d* arcs, a new bin is created.

BRKGA for RWA: GA-RWA

T.F. Noronha, M.G.C.R., and C.C. Ribeiro, "A biased random-key genetic algorithm for routing and wavelength assignment," J. of Global Optimization, published online 24 September 2010.

Tech report version:

http://www.research.att.com/~mgcr/doc/garwa-full.pdf

BRKGA for RWA: GA-RWA

Noronha, R., and Ribeiro (2010)

- Encoding of solution: A vector X of |T| random keys in the range [0,1], where T is the set of connection request node pairs.
- Decoding:
  - 1) Sort the connection in set T in non-increasing order of $c(i) = SP(i) \times 10 + X[i]$, for each connection $i \in T$.
  - 2) Apply BFD-RWA in the order determined in step 1.

BRKGA for RWA: GA-RWA

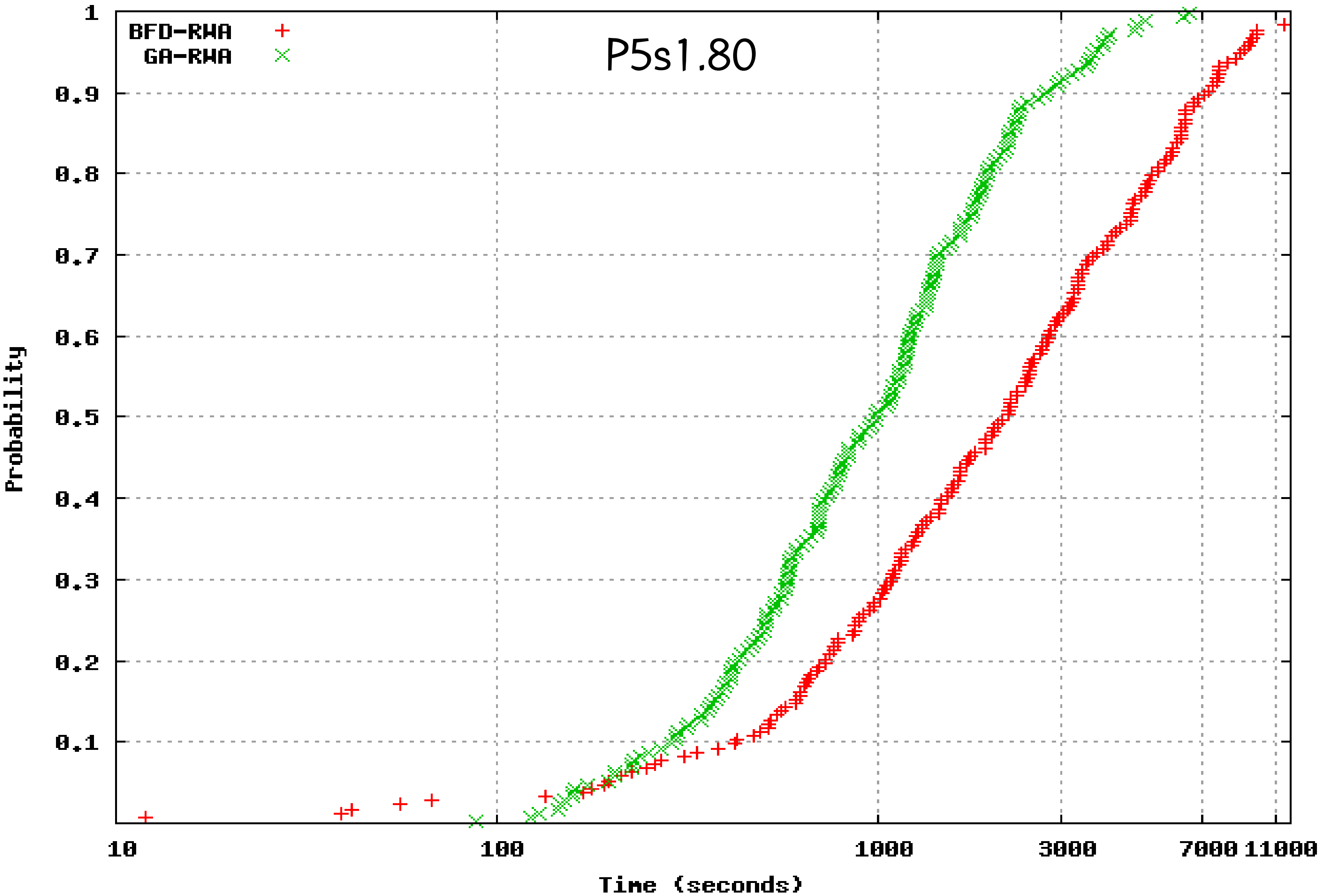
Noronha, R., and Ribeiro (2010)

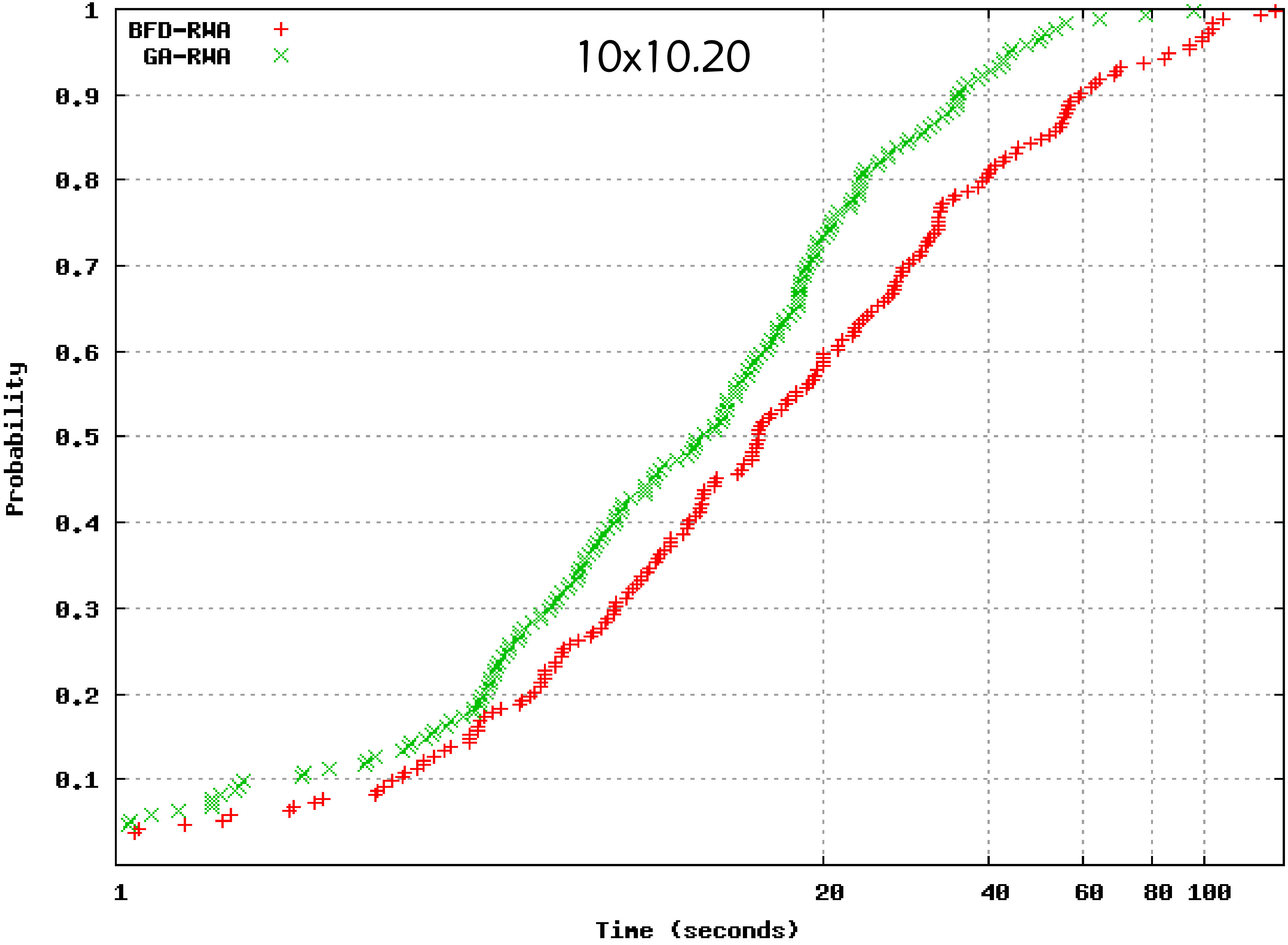
- Encoding of solution: A vector X of |T| random keys in the range [0,1], where T is the set of connection request node pairs.
- Decoding:
  - 1) Sort the connection in set T in non-increasing order of $c(i) = SP(i) \times 10 + X[i]$, for each connection $i \in T$.
  - 2) Apply BFD-RWA in the order determined in step 1.

Since there are many ties connection pairs with The same SP(i) value, in the original algorithm of Skorin-Kapov, ties are broken at random. In the BRKGA, the algorithm "learns" how to break ties.

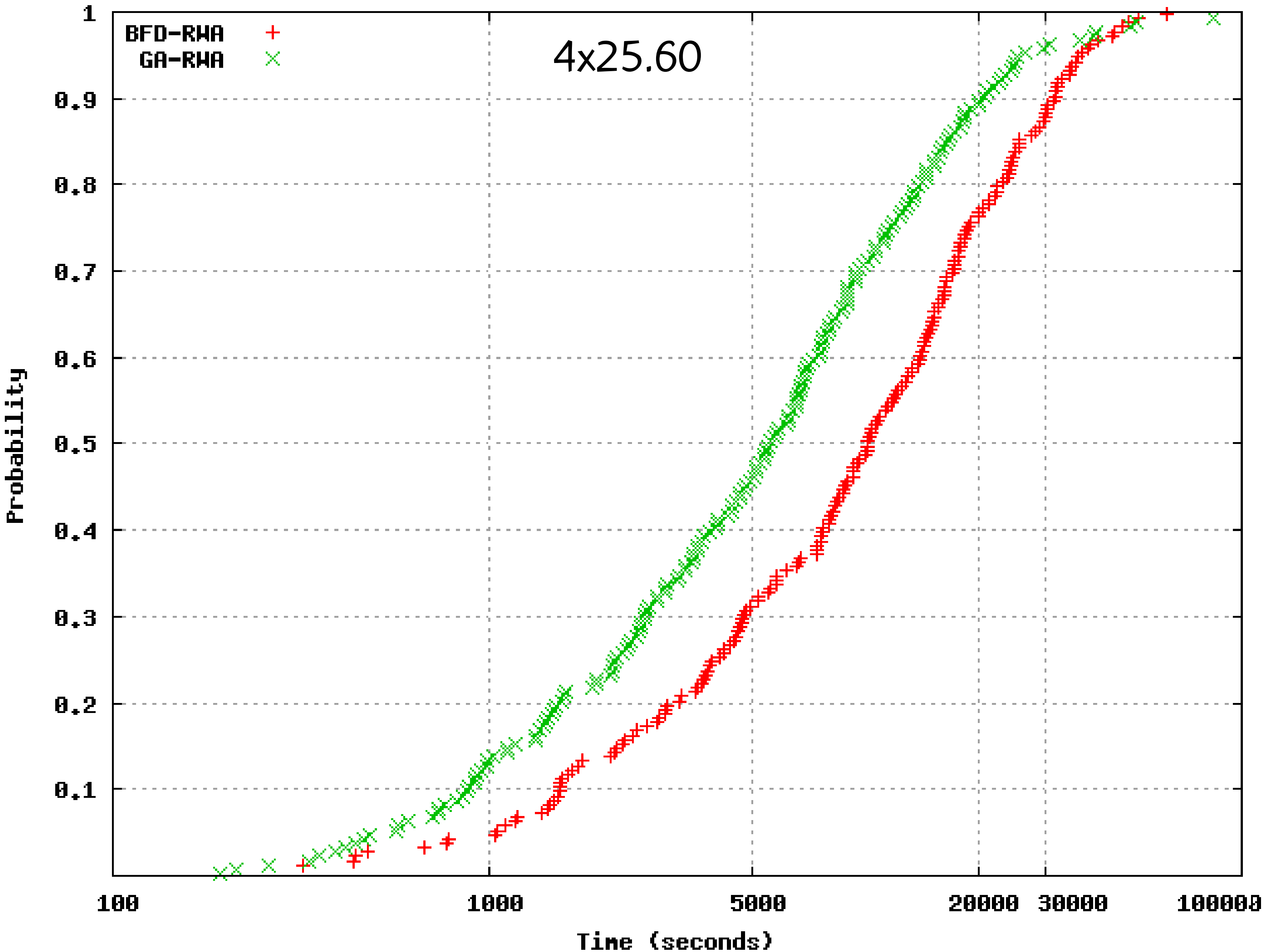
Experiments

- Compare multi-start version of Skorin-Kapov's heuristic (MS-RWA) with GA-RWA.
- Make 200 independent runs of each heuristic on each heuristic on five instances, stopping when target solution was found (target was set to be best solution found by MS-RWA after 10,000 multi-start iterations. Plot CDF for each heuristic.

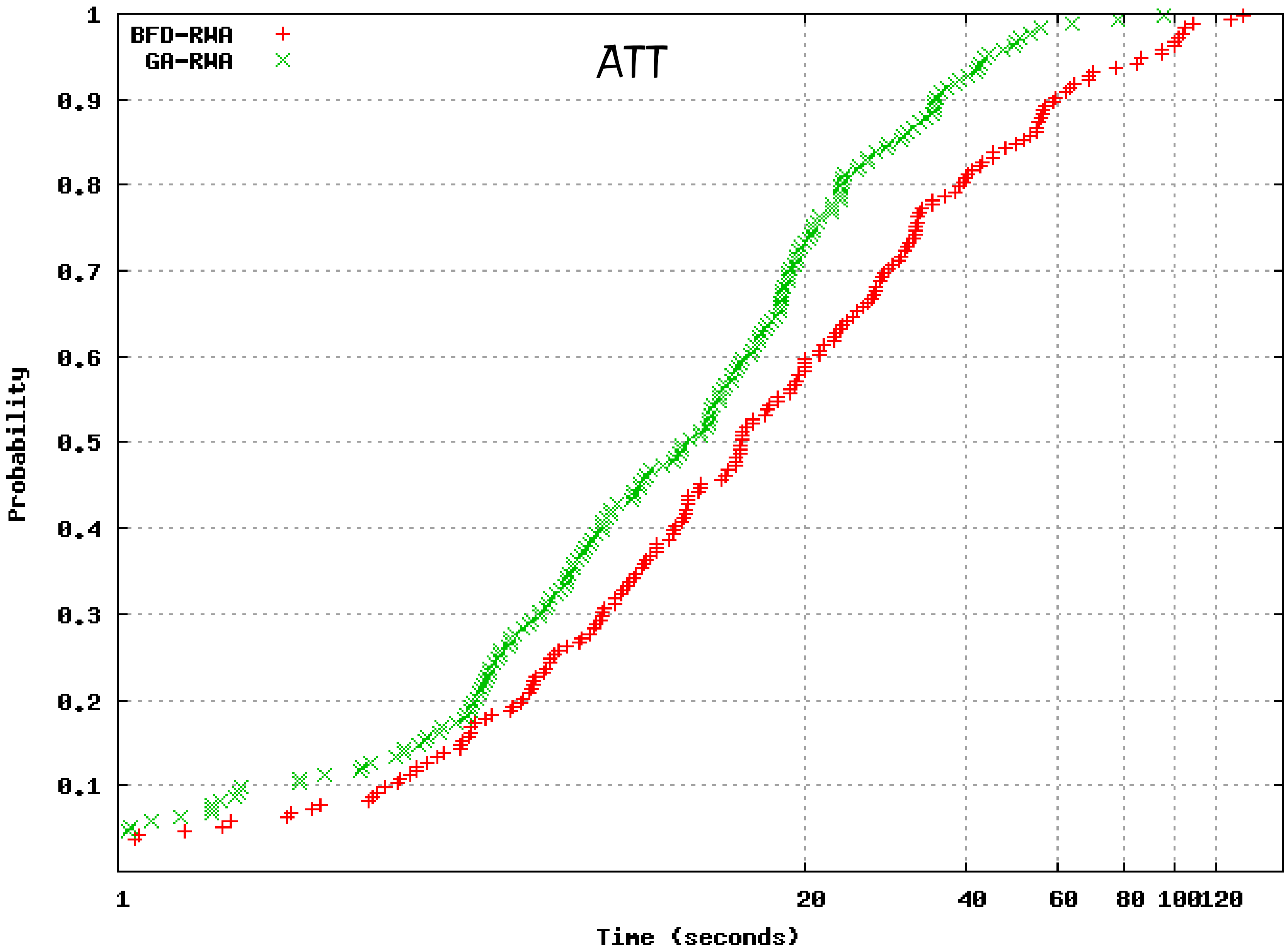

P5s1.80
BFD-RWA
GA-RWA
Probability
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
10
100
1000
3000
7000
11000
Time (seconds)



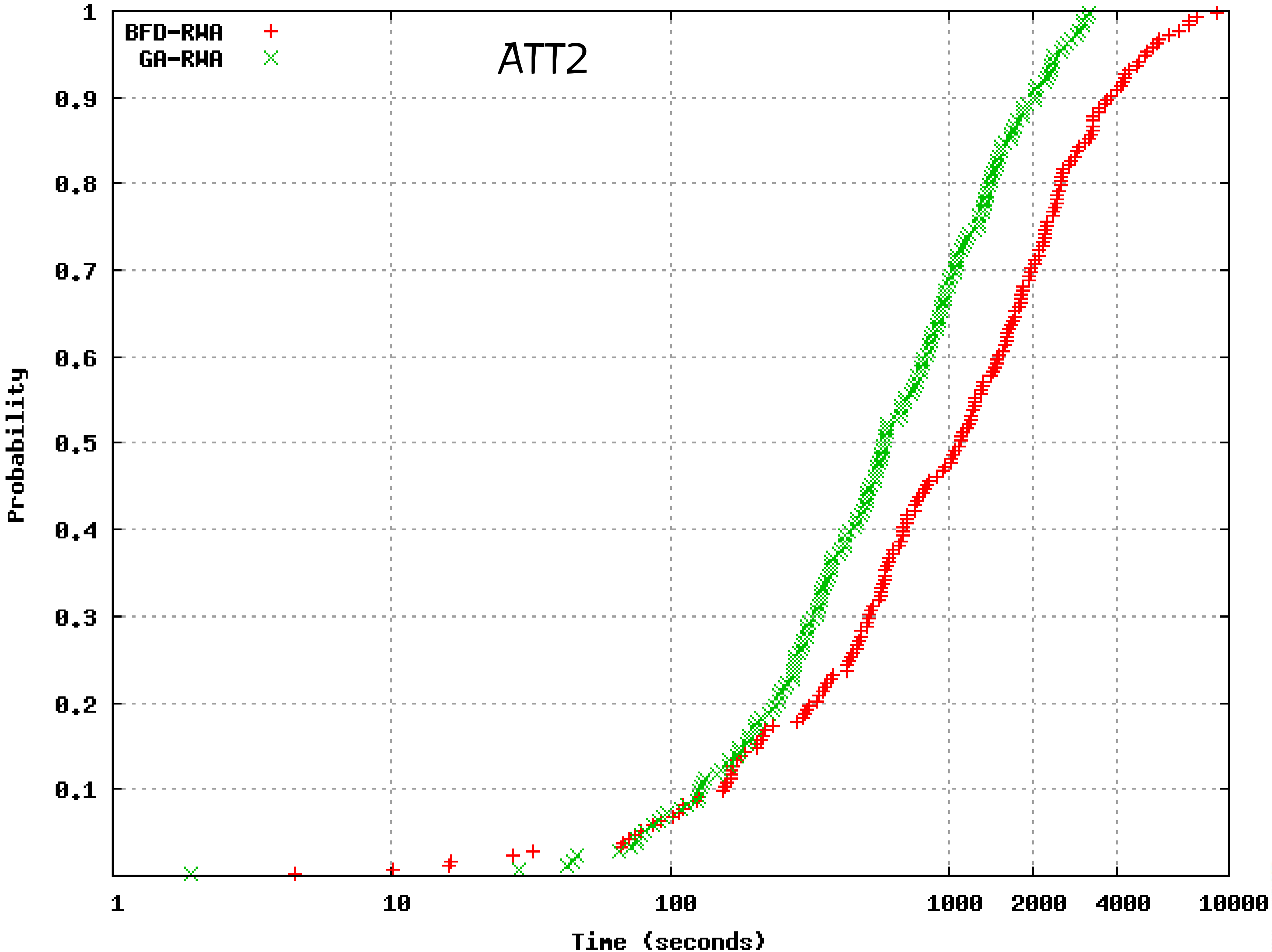
at&t
Your world. Delivered.



4x25.60
BFD-RWA
GA-RWA
Probability
Time (seconds)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
100
1000
5000
20000
30000
100000



at&t
Your world. Delivered.


ATT2
BFD-RWA
GA-RWA
Probability
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
10
100
1000
2000
4000
10000
Time (seconds)
at&t
Your world. Delivered.

Concluding remarks

Concluding remarks

- A small modification of Bean's RKGA results in a BRKGA.
- Though small, this modification, leads to significant performance improvements.
- BRKGA are true metaheuristics: they coordinate simple heuristics and produce better solutions than the simple heuristics alone.
- Problem independent module of a BRKGA needs to be implemented once and can be reused for a wide range of problems. User can focus on problem dependent module.
- BRKGA heuristics are highly parallelizable.

Concluding remarks

- BRKGA have been applied in a wide range of application areas, including scheduling, packing, cutting, tollbooth assignment, ...
- We have had only a small glimpse at BRKGA applications to problems arising in telecommunications.
- The BRKGAs described in this talk are all state-of-the-art heuristics for these applications
- We are currently working on a number of tree-based applications in telecommunications, including degree-constrained spanning tree problem and regenerator location.

The End

These slides and all of the papers cited in this talk can be downloaded from my homepage:

http://www.research.att.com/~mgcr