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

Plenary talk given at Optimization 2011
Lisbon, Portugal ✤ July 26, 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
  - Design of survivable IP networks with composite links
  - Redundant server location for content distribution
  - Regenerator location
  - Routing & wavelength assignment in optical networks
- Concluding remarks

Reference

M.G.C.R., "Biased random-key genetic algorithms with applications in telecommunications," TOP, published online 23 March 2011.

Tech report version:

http://www2.research.att.com/~mgcr/doc/brkga-telecom.pdf

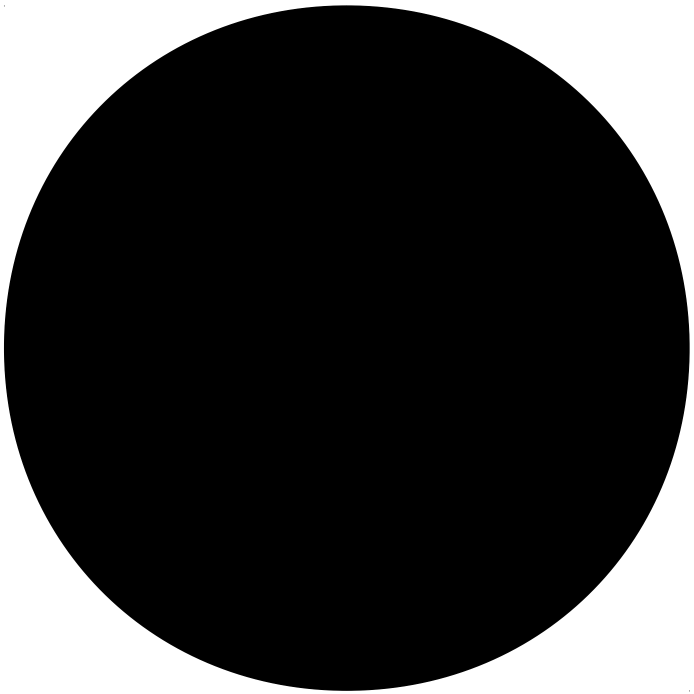
at&t

Your world. Delivered.

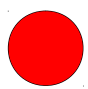
Biased random-key genetic algorithms

Genetic algorithms

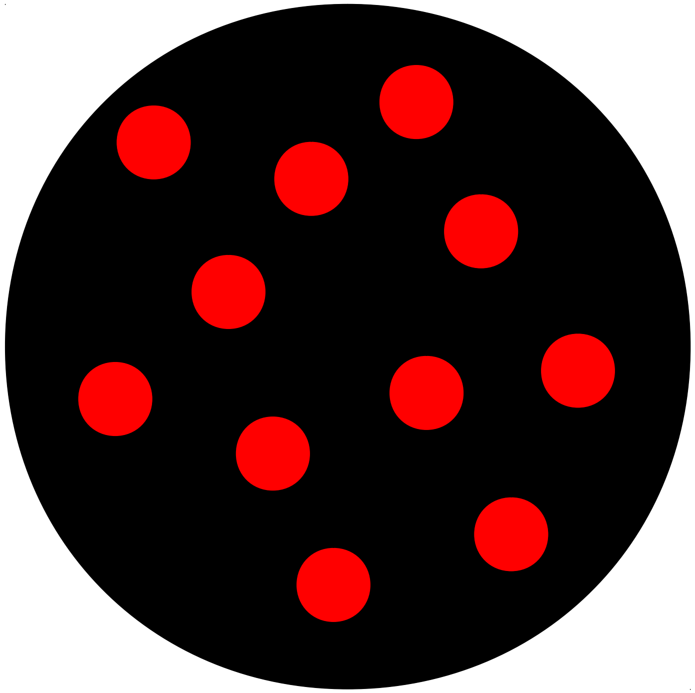
Holland (1975)



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

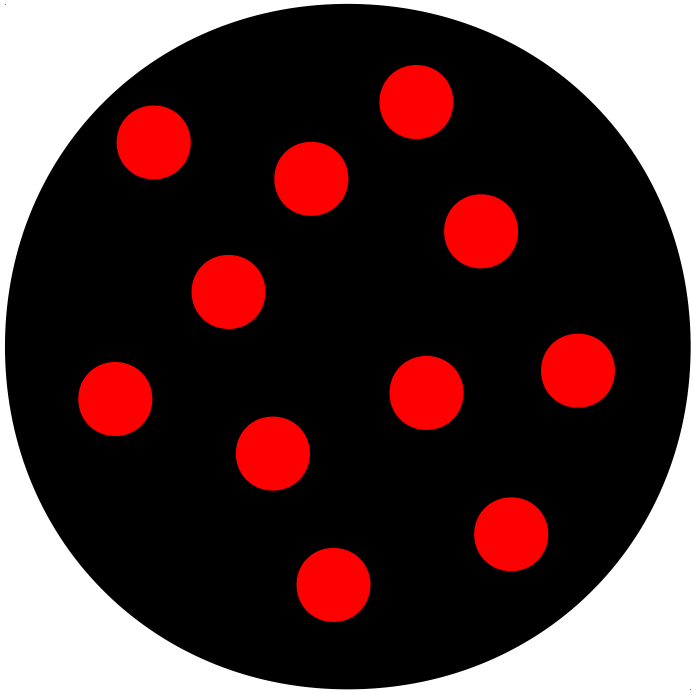
Individual: solution 

Genetic algorithms



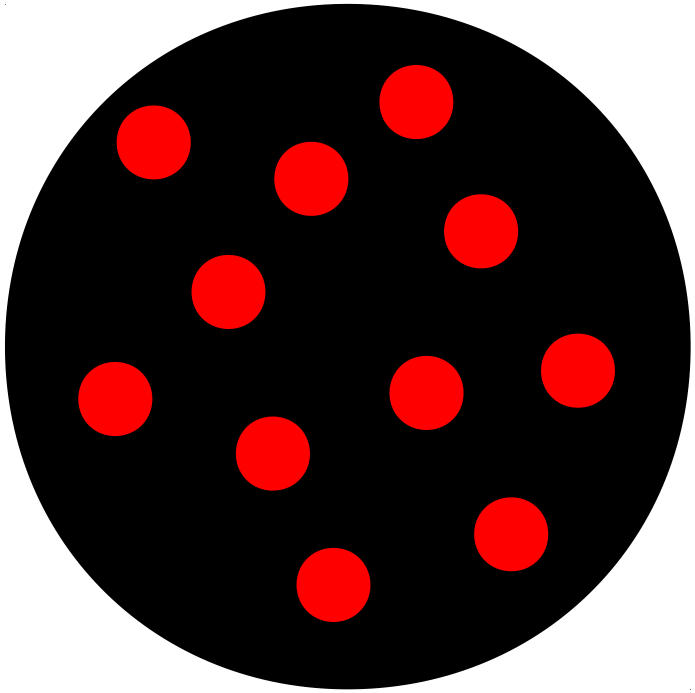
Individual: solution
Population: set of fixed number of individuals

Genetic algorithms



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

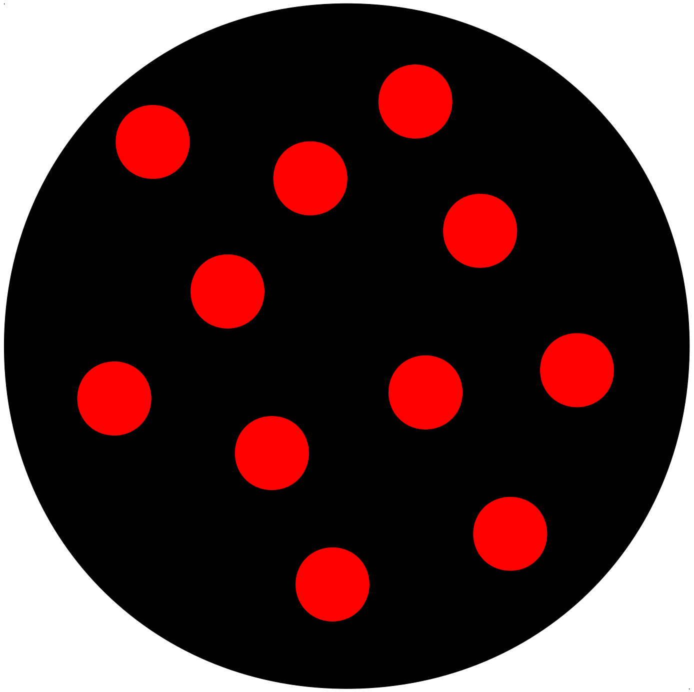
Genetic algorithms



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

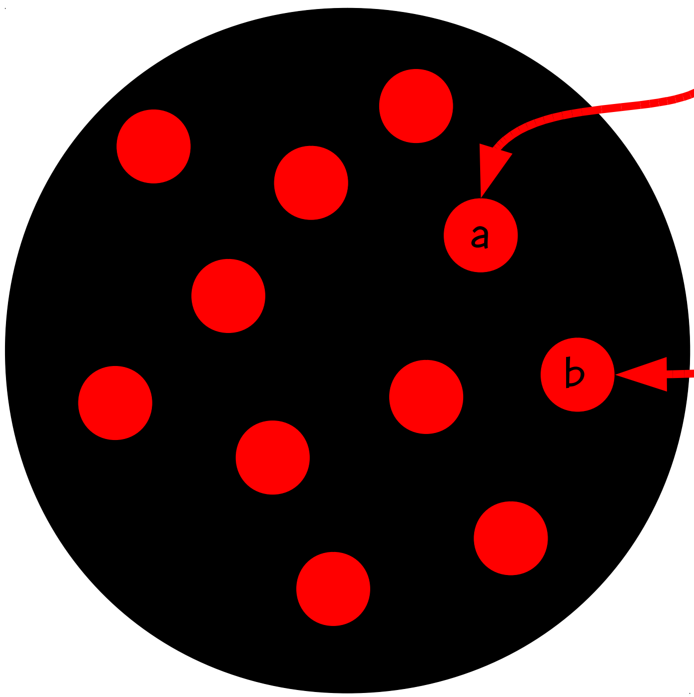


Genetic algorithms evolve population applying the principle of survival of the fittest.

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

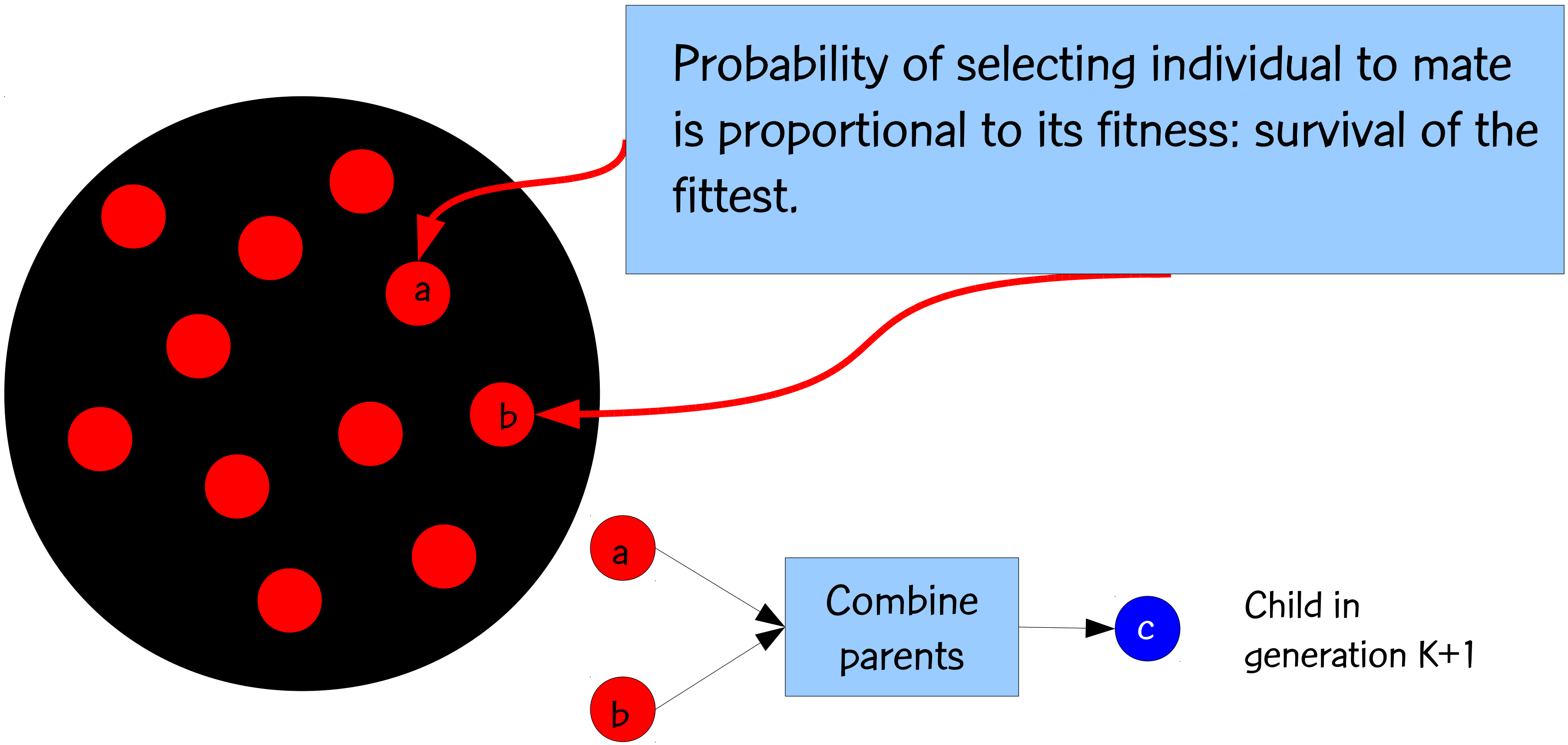
Individuals from one generation are combined to produce offspring that make up next generation.

Genetic algorithms



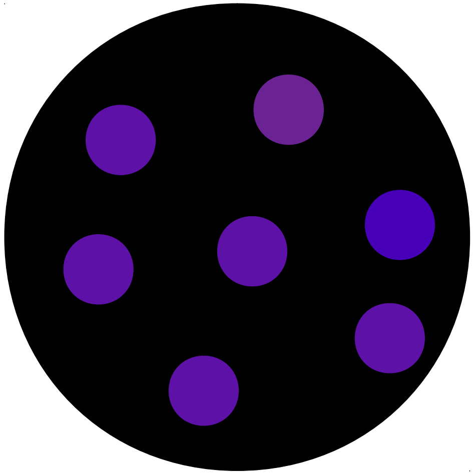
Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.

Genetic algorithms

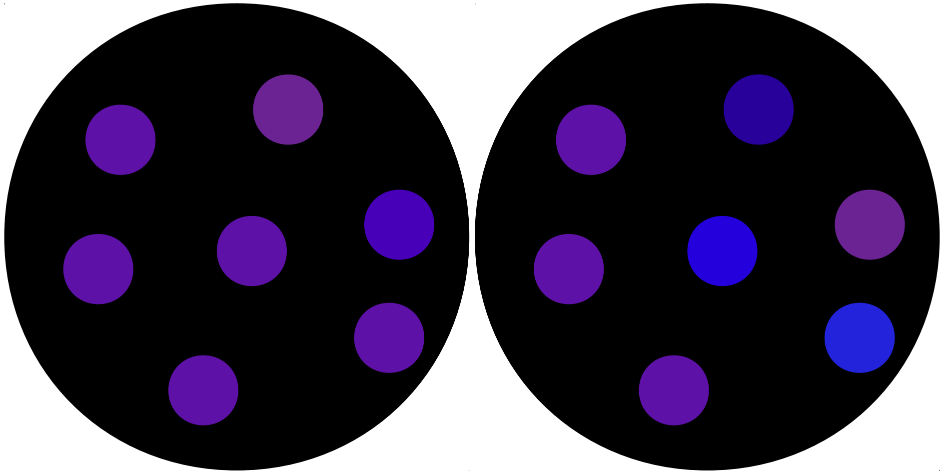


Parents drawn from generation K

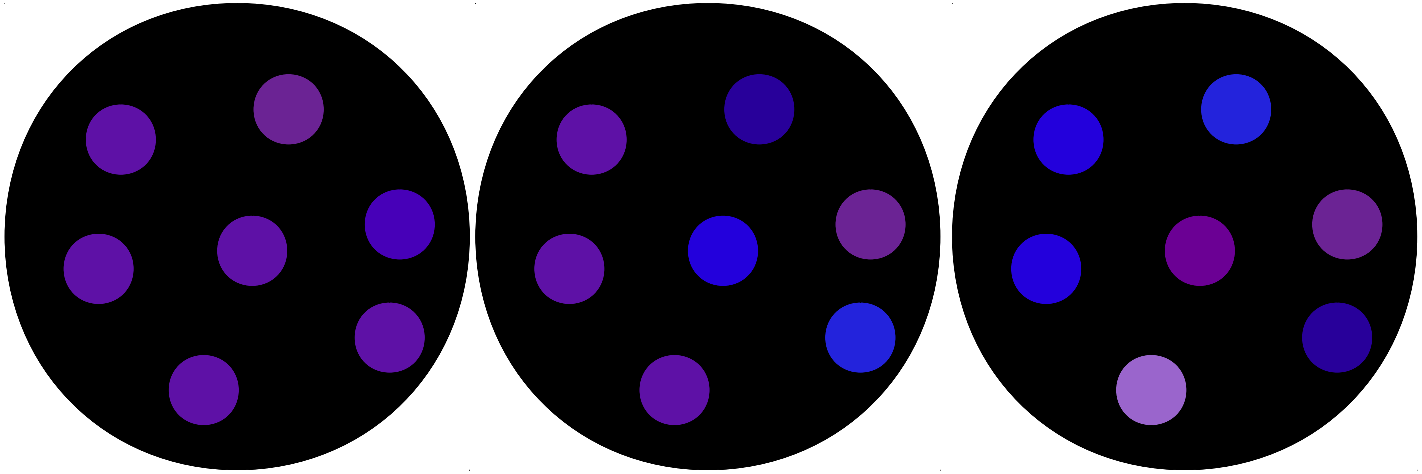
Evolution of solutions



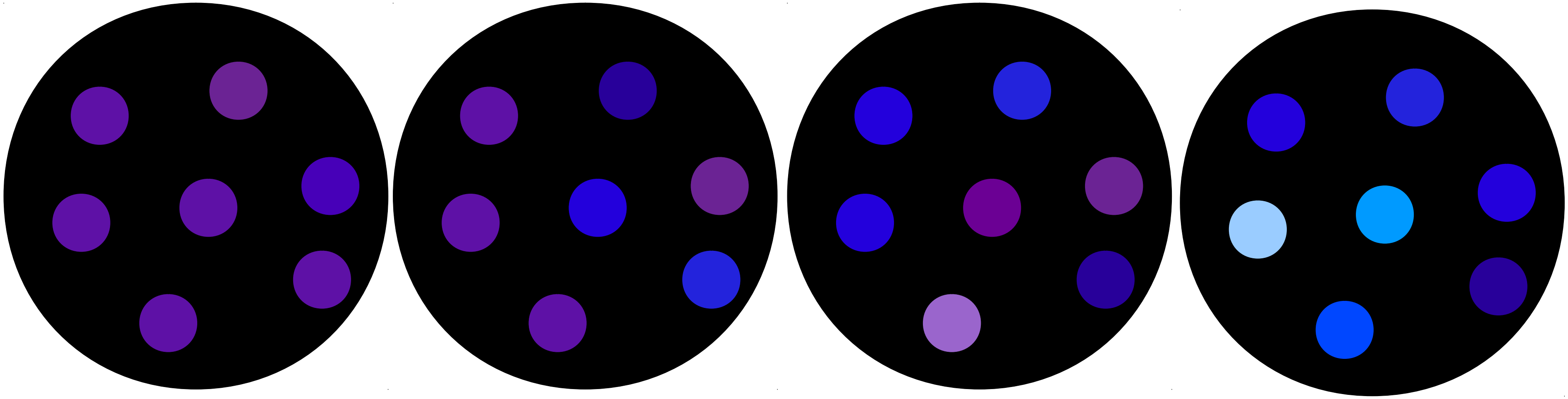
Evolution of solutions



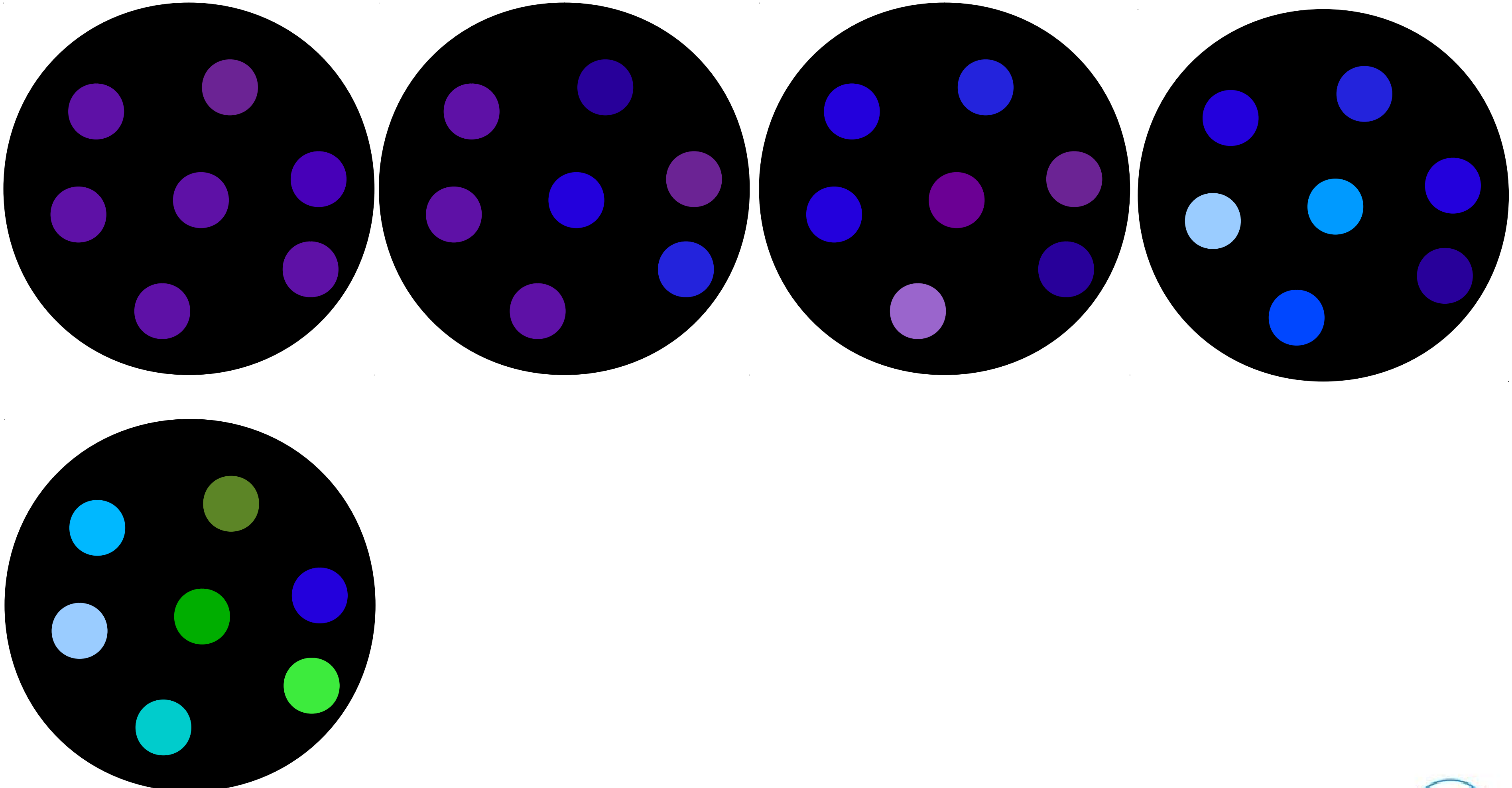
Evolution of solutions



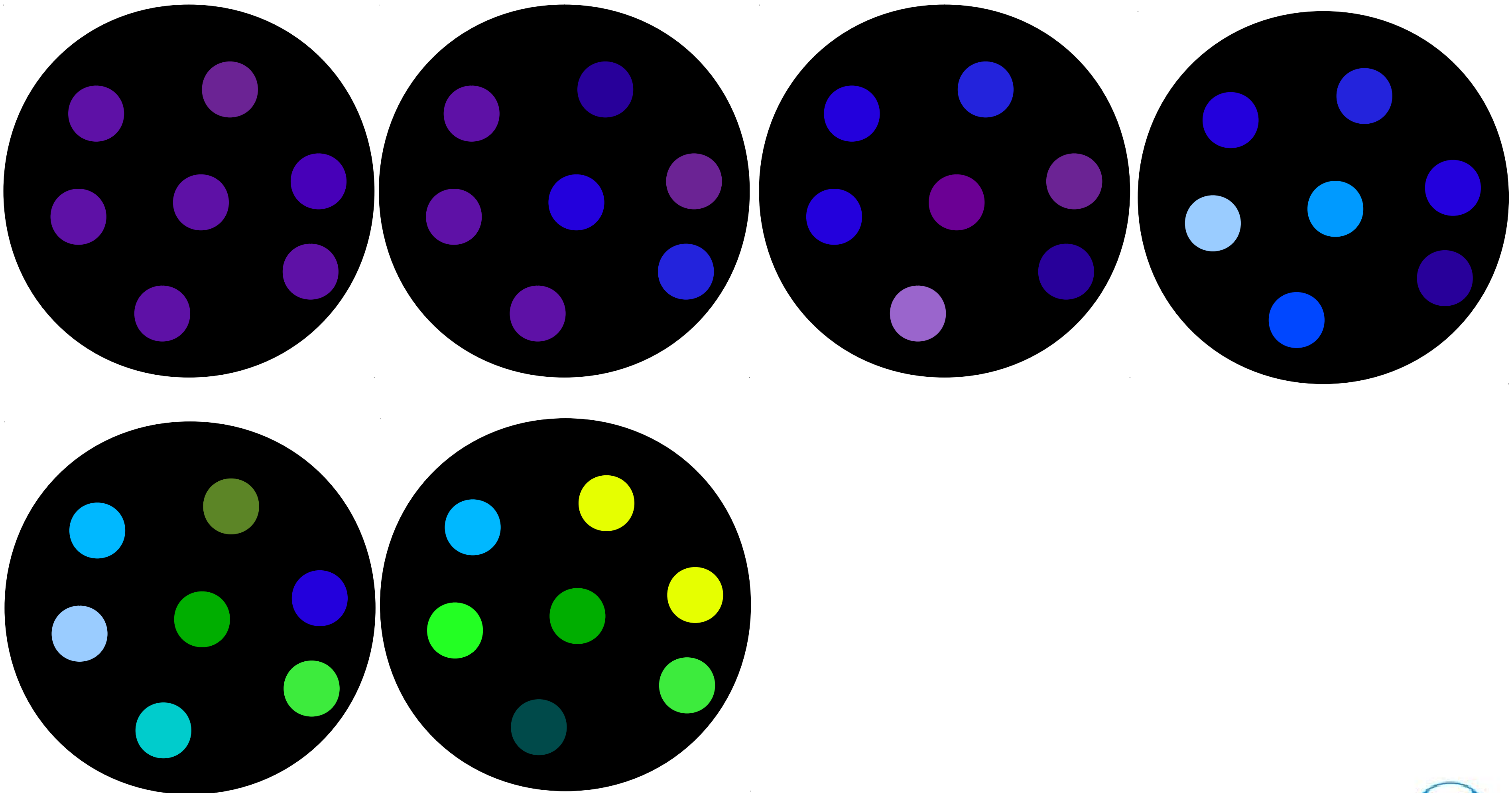
Evolution of solutions



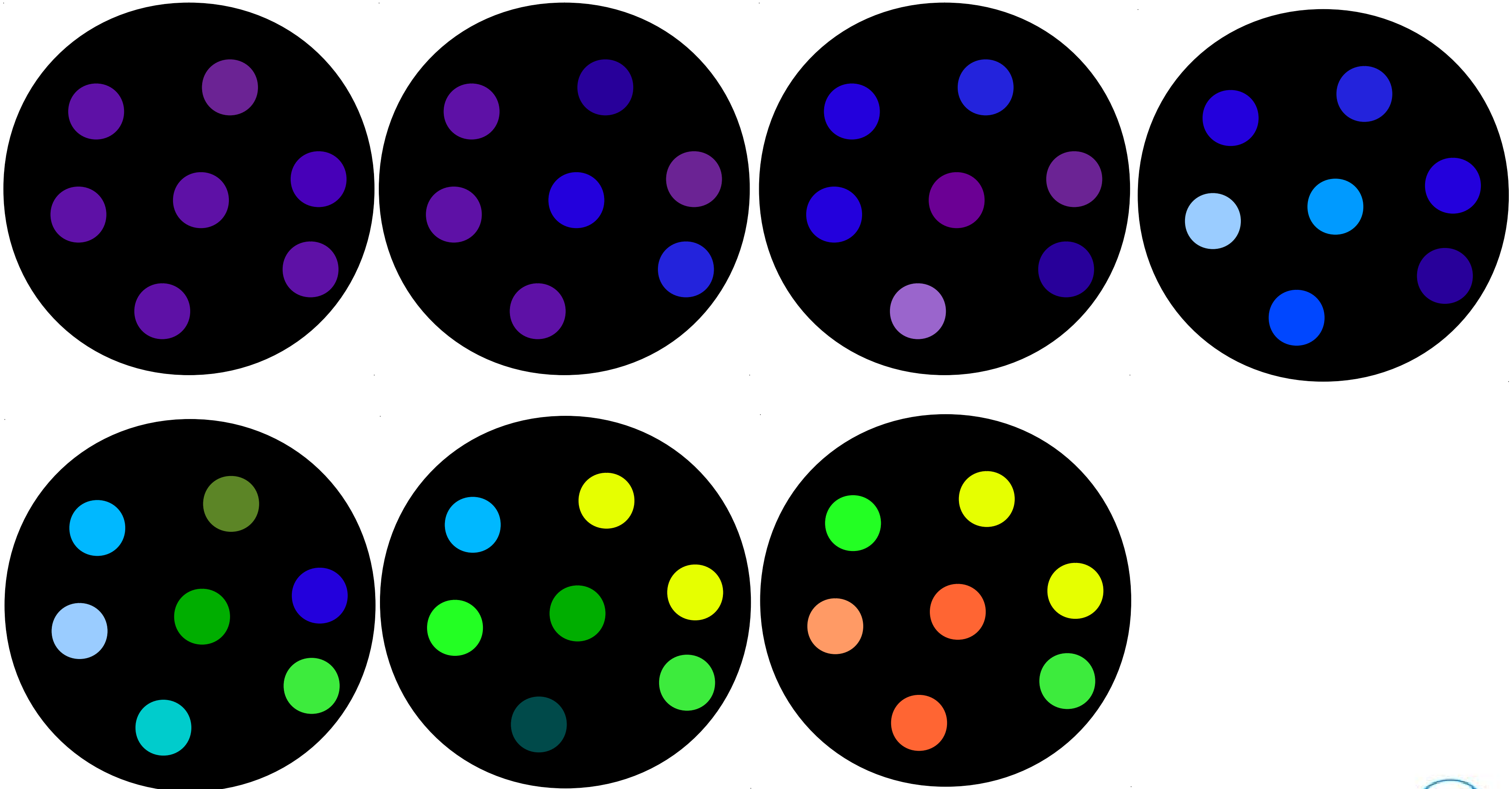
Evolution of solutions



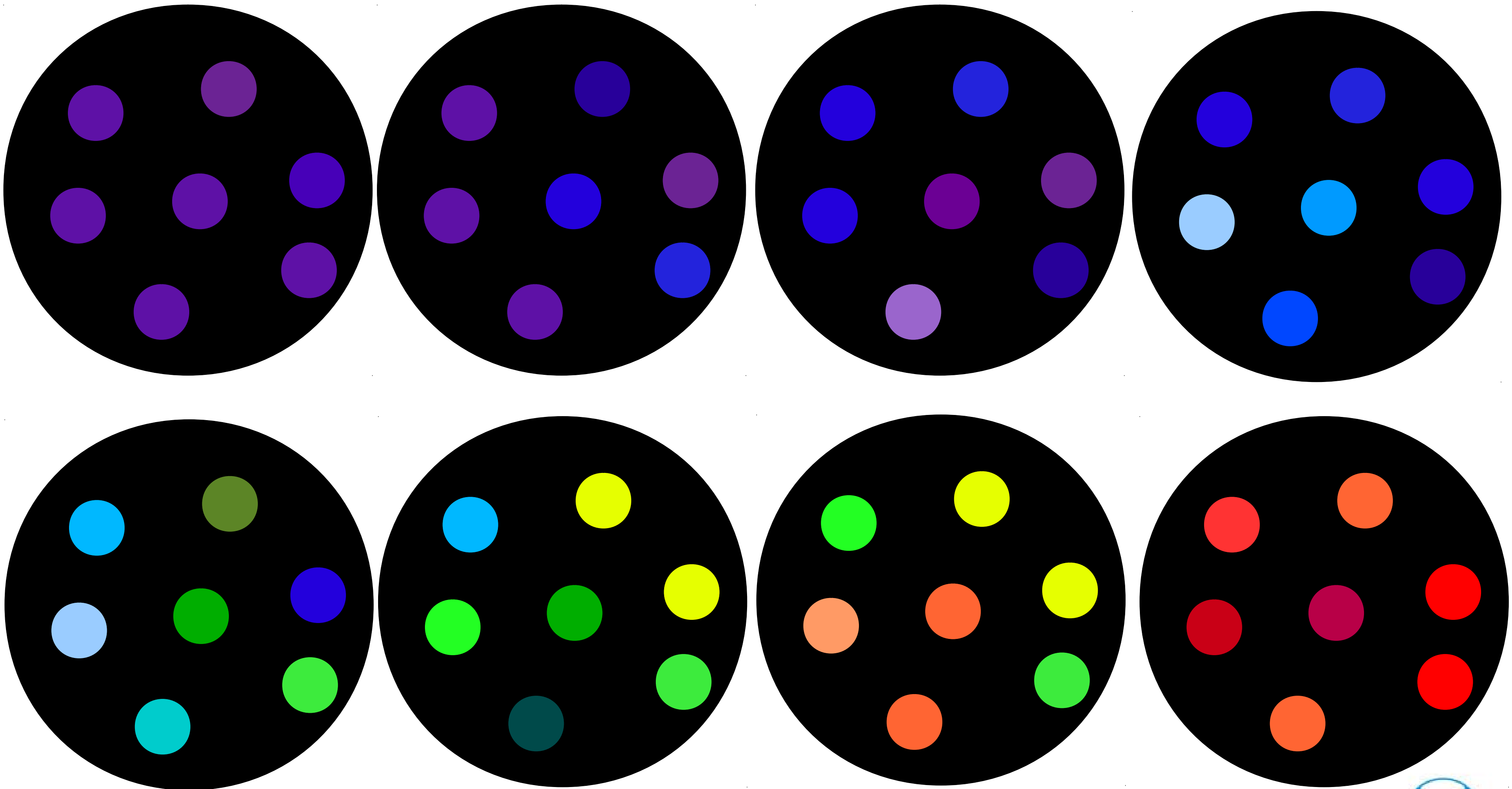
Evolution of solutions



Evolution of solutions

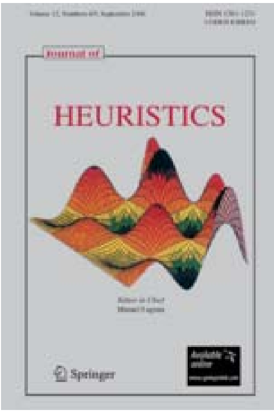


Evolution of solutions



Genetic algorithms with random keys

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

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].

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

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

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

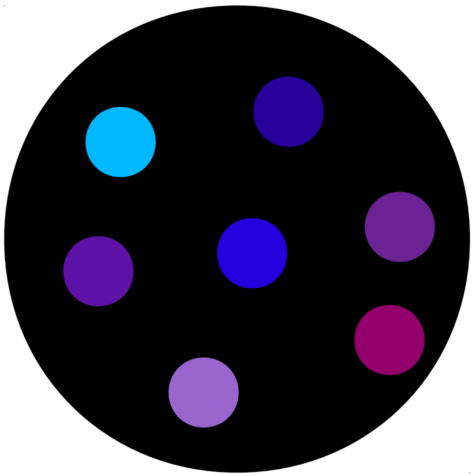
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele to the child.

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.

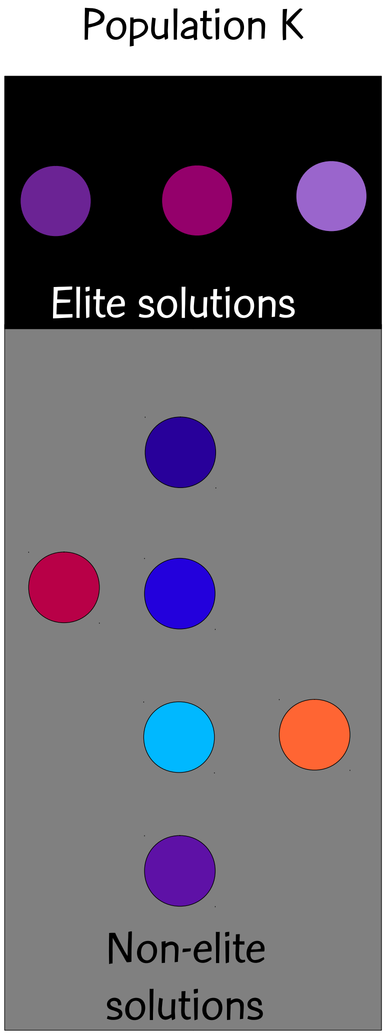
GAs and random keys

Initial population is made up of P random-key vectors, each with N keys, each having a value generated uniformly at random in the interval (0,1].



GAs and random keys

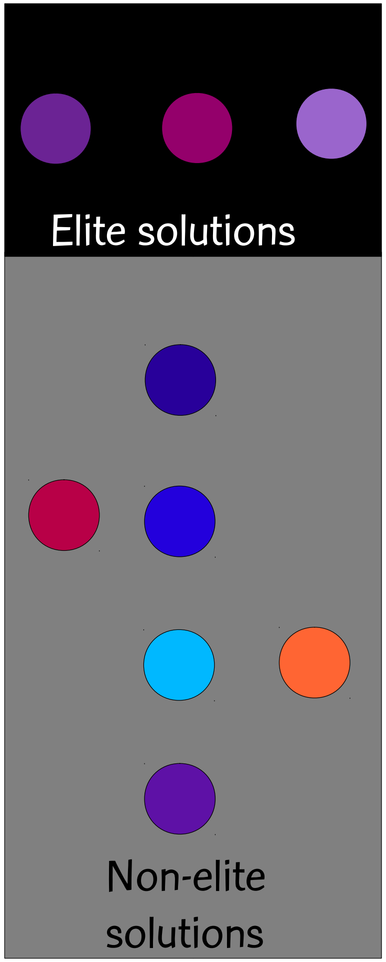
At the K-th generation, compute the cost of each solution ...



GAs and random keys

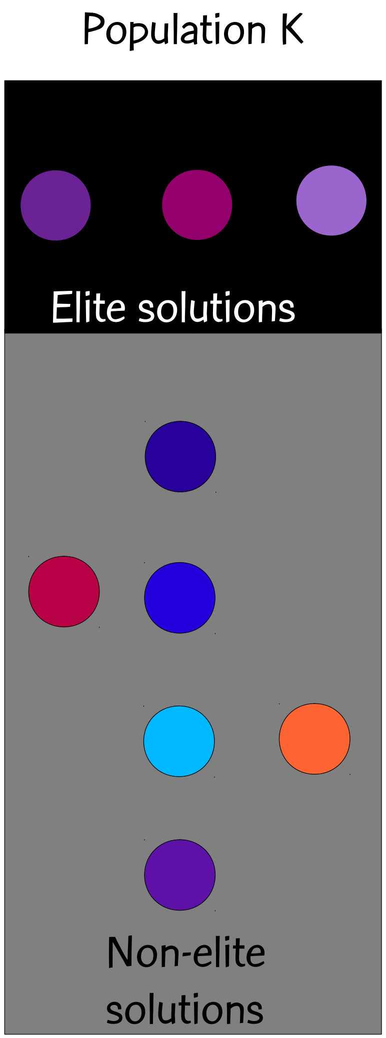
At the K-th generation, compute the cost of each solution and partition the solutions into two sets:

Population K



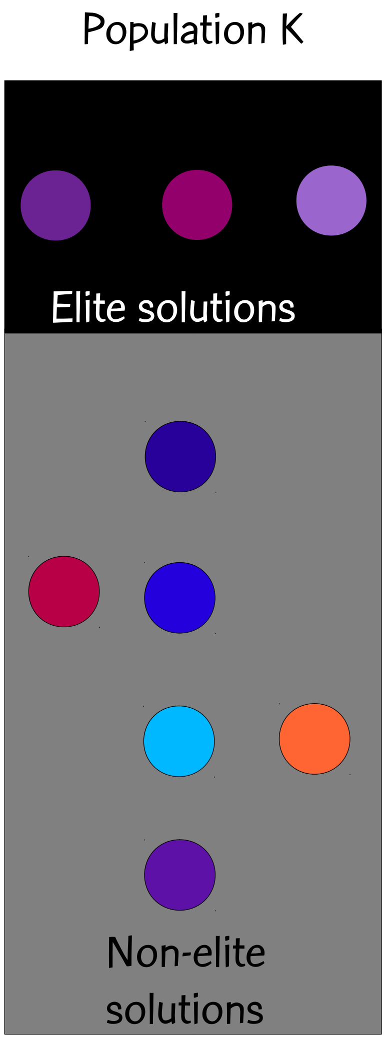
GAs and random keys

At the K-th generation, compute the cost of each solution and partition the solutions into two sets: elite solutions and non-elite solutions.



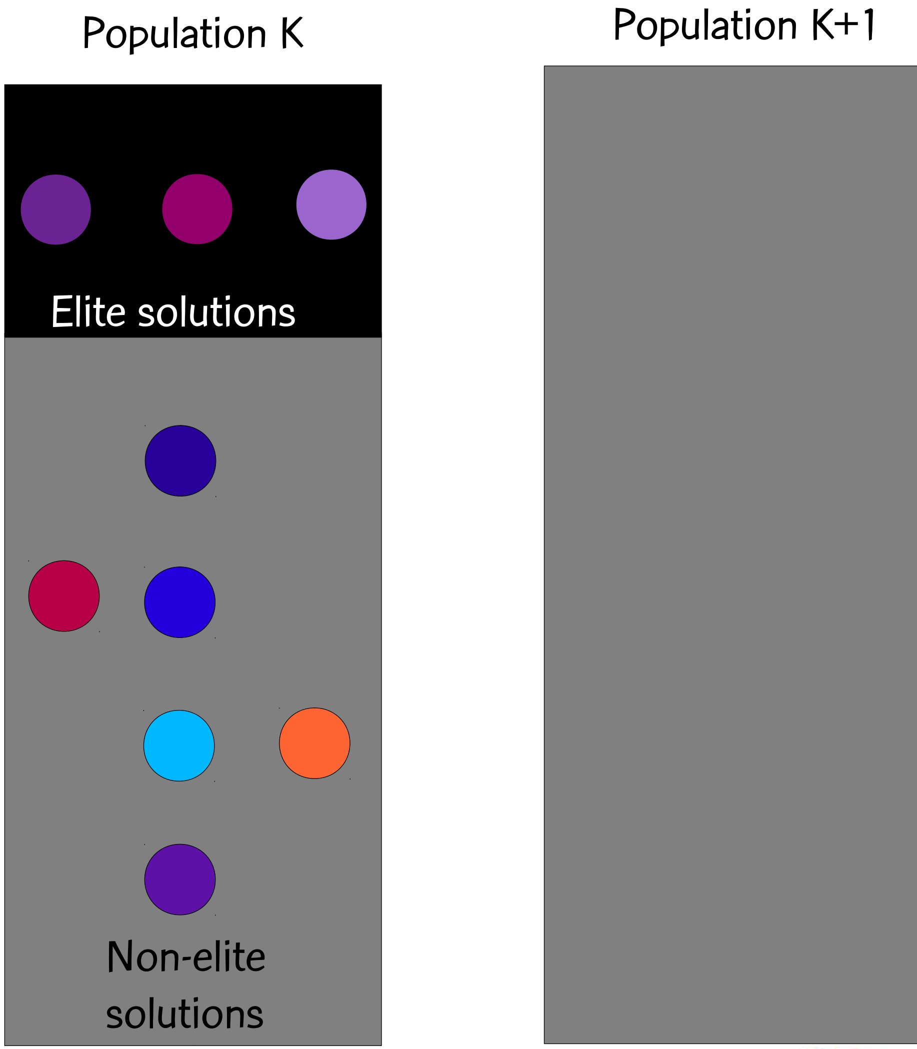
GAs and random keys

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



GAs and random keys

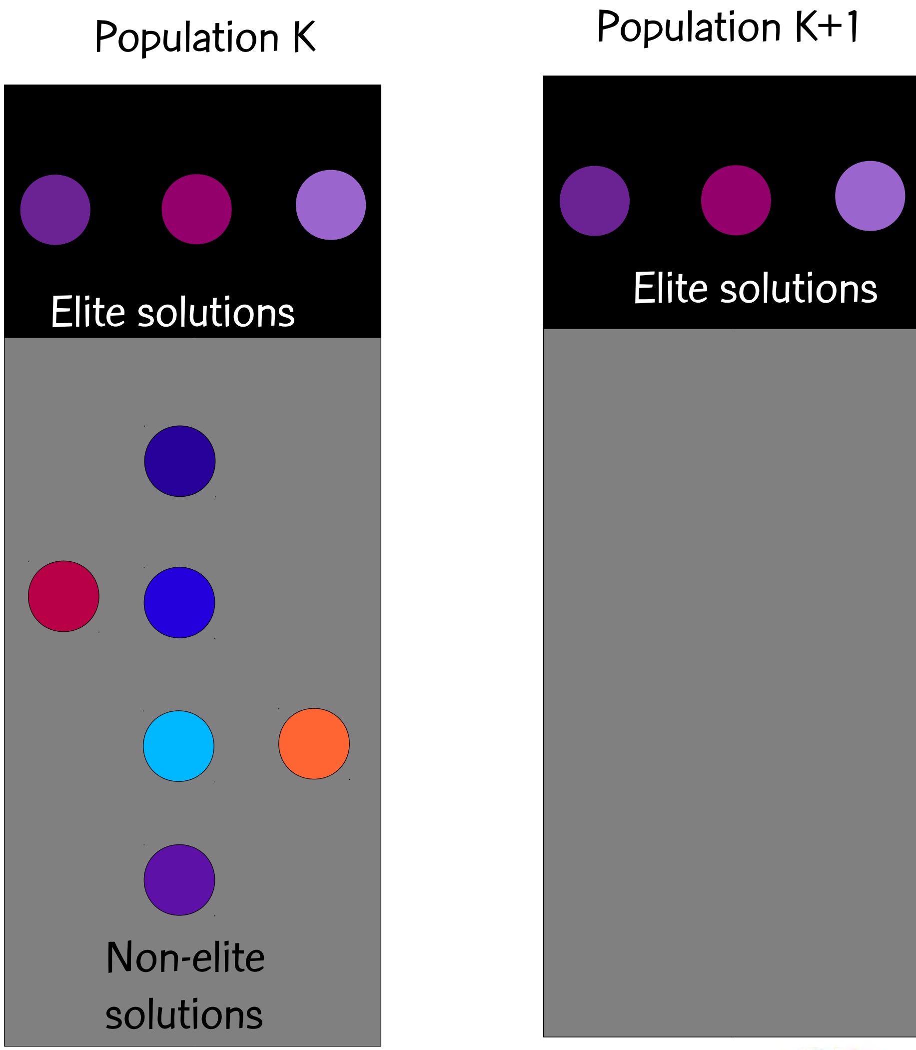
Evolutionary dynamics



GAs and random keys

Evolutionary dynamics

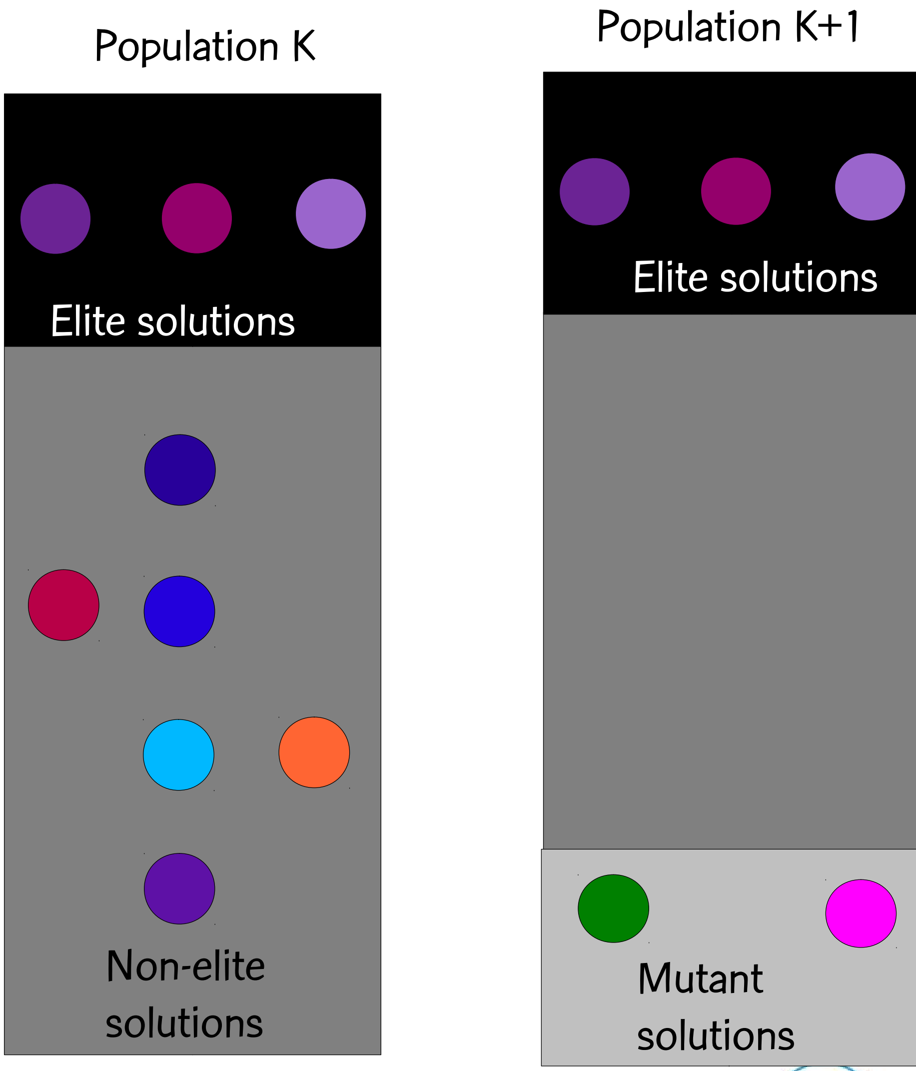
– Copy elite solutions from population K to population K+1



GAs and random keys

Evolutionary dynamics

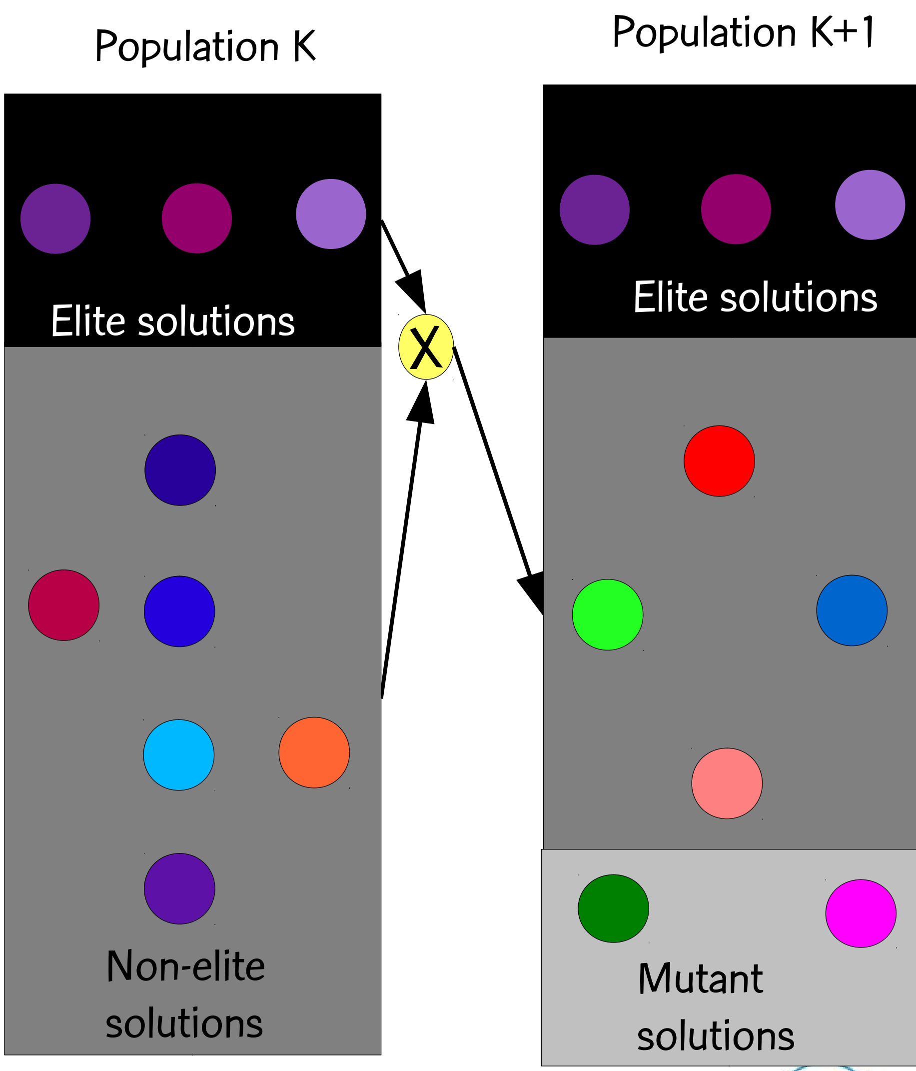
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1



Biased random key GA

Evolutionary dynamics

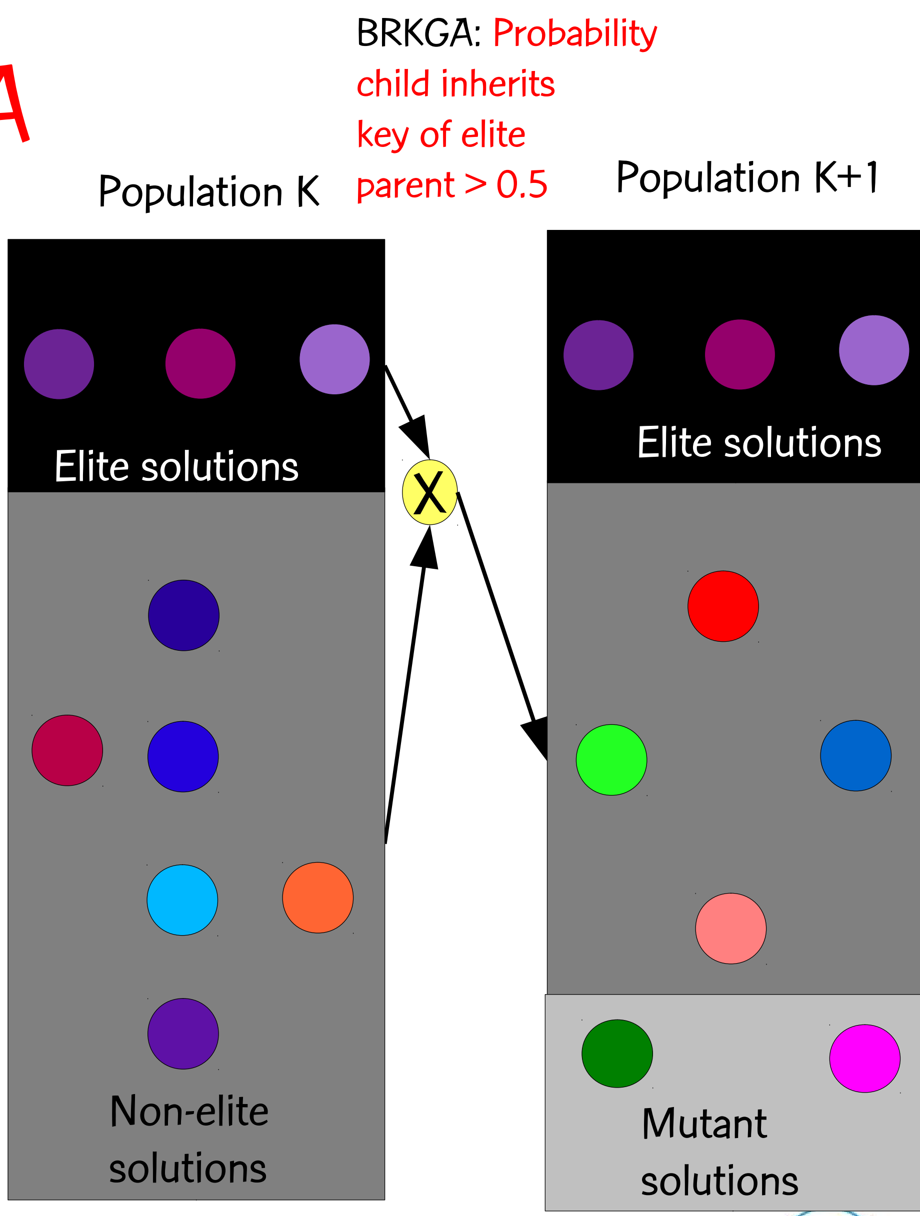
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1
- While K+1-th population < P
  - RANDOM-KEY GA: Use any two solutions in population K to produce child in population K+1. Mates are chosen at random.



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
  - RANDOM-KEY GA: Use any two solutions in population K to produce child in population K+1. Mates are chosen at random.
  - BIASED RANDOM-KEY GA: Mate elite solution with non-elite of population K to produce child in population K+1. Mates are chosen at random.

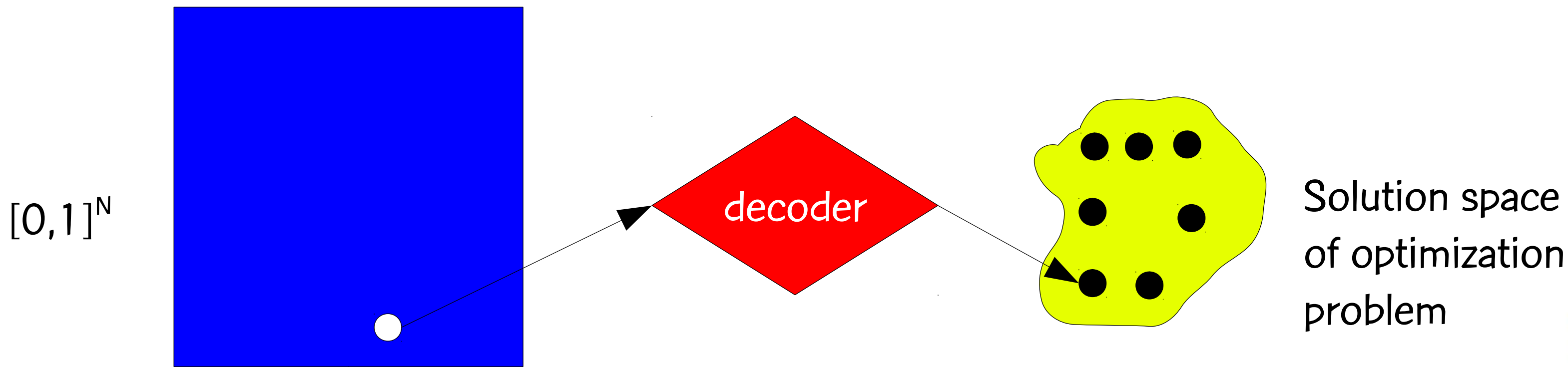


Observations

- 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

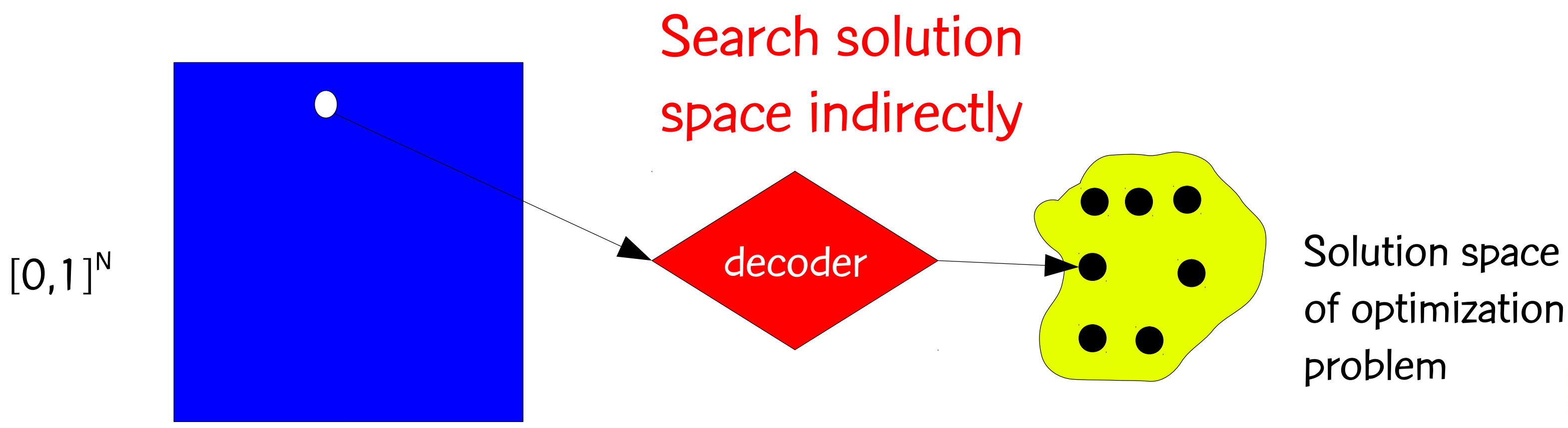
Decoders

- 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.



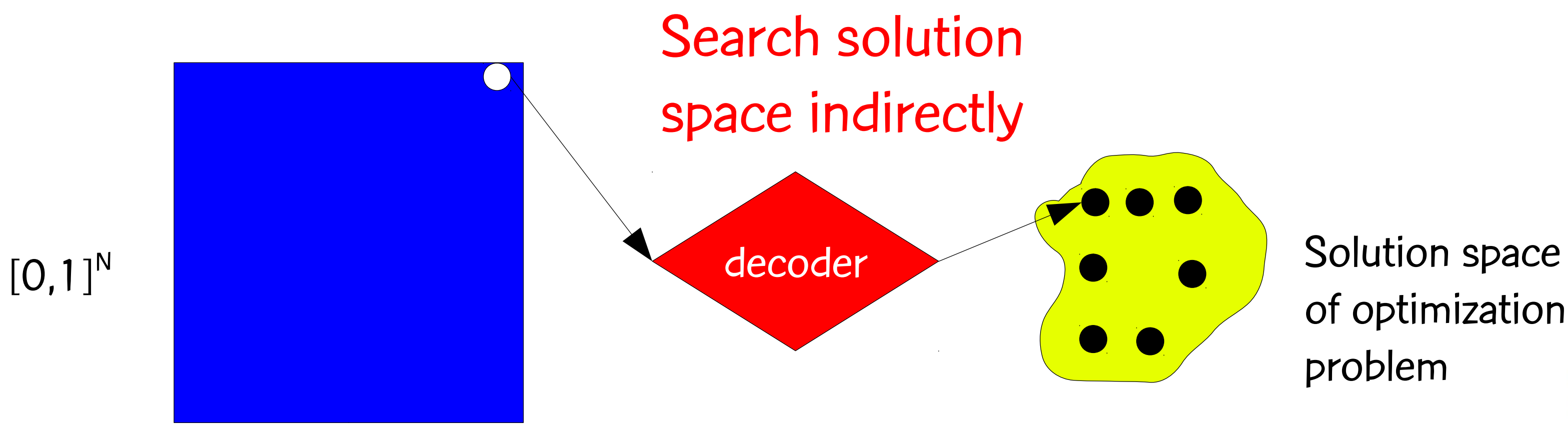
Decoders

- 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.



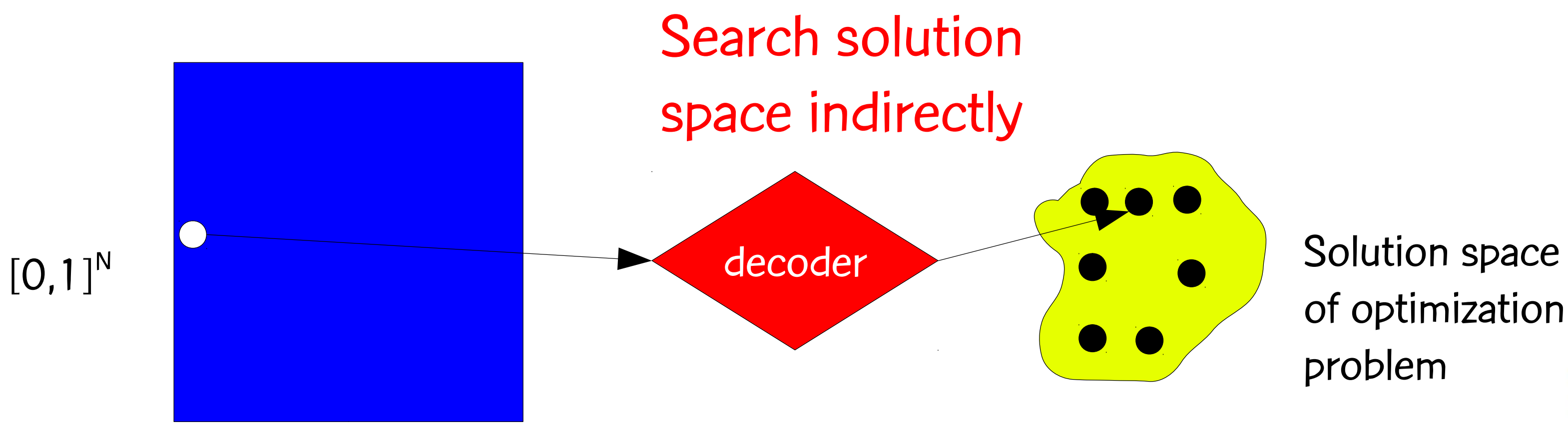
Decoders

- *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.

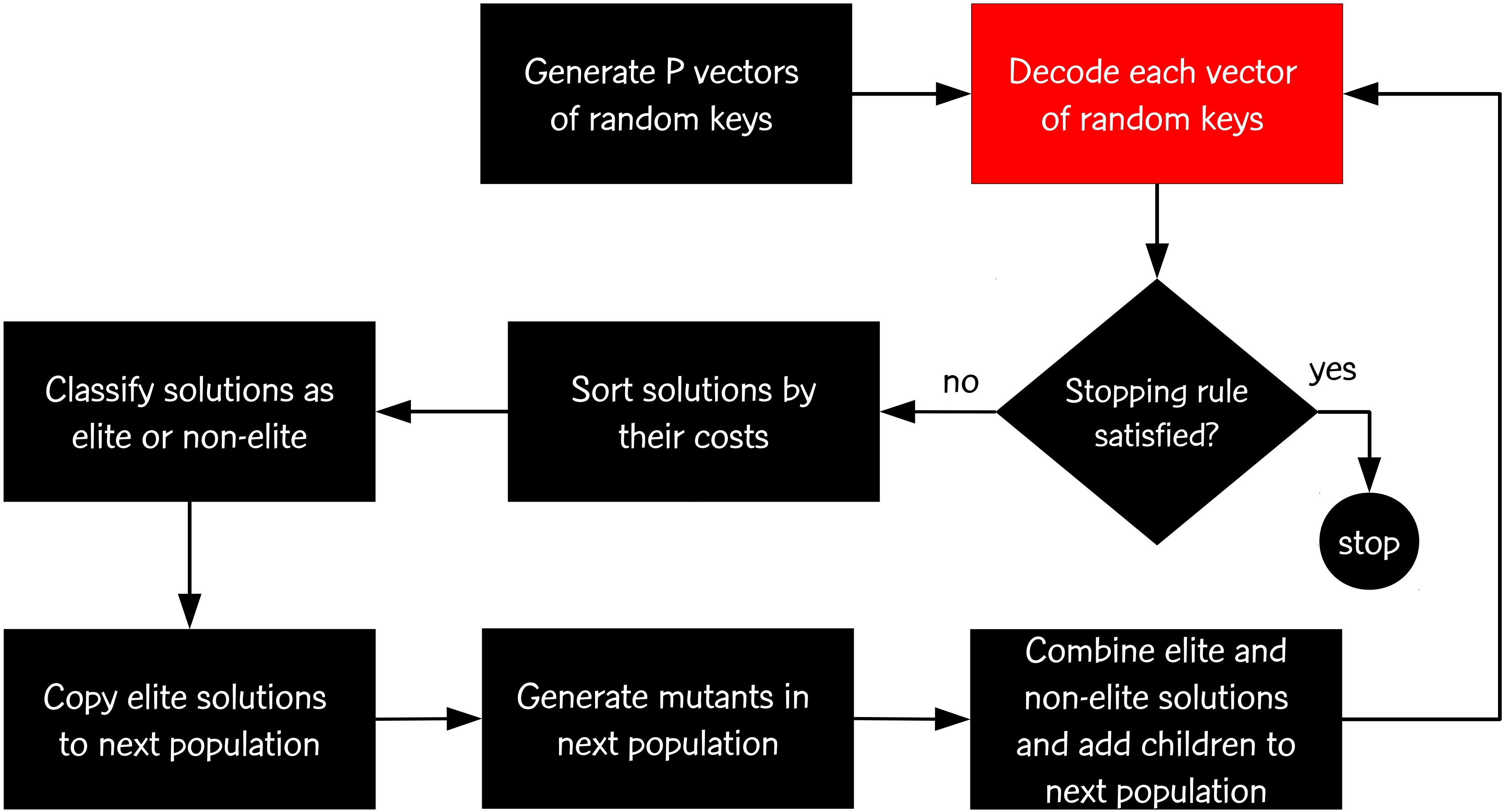


Decoders

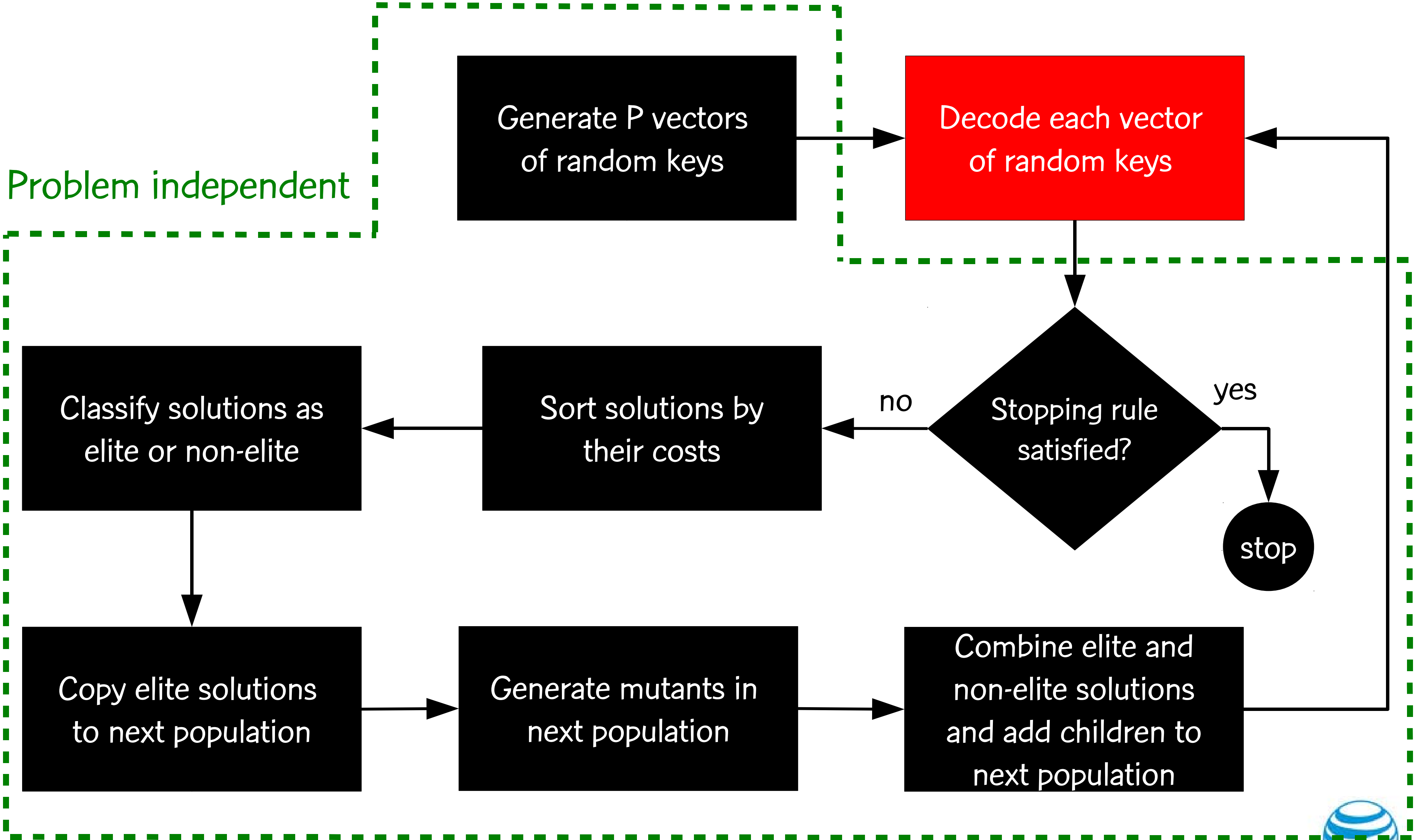
- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly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



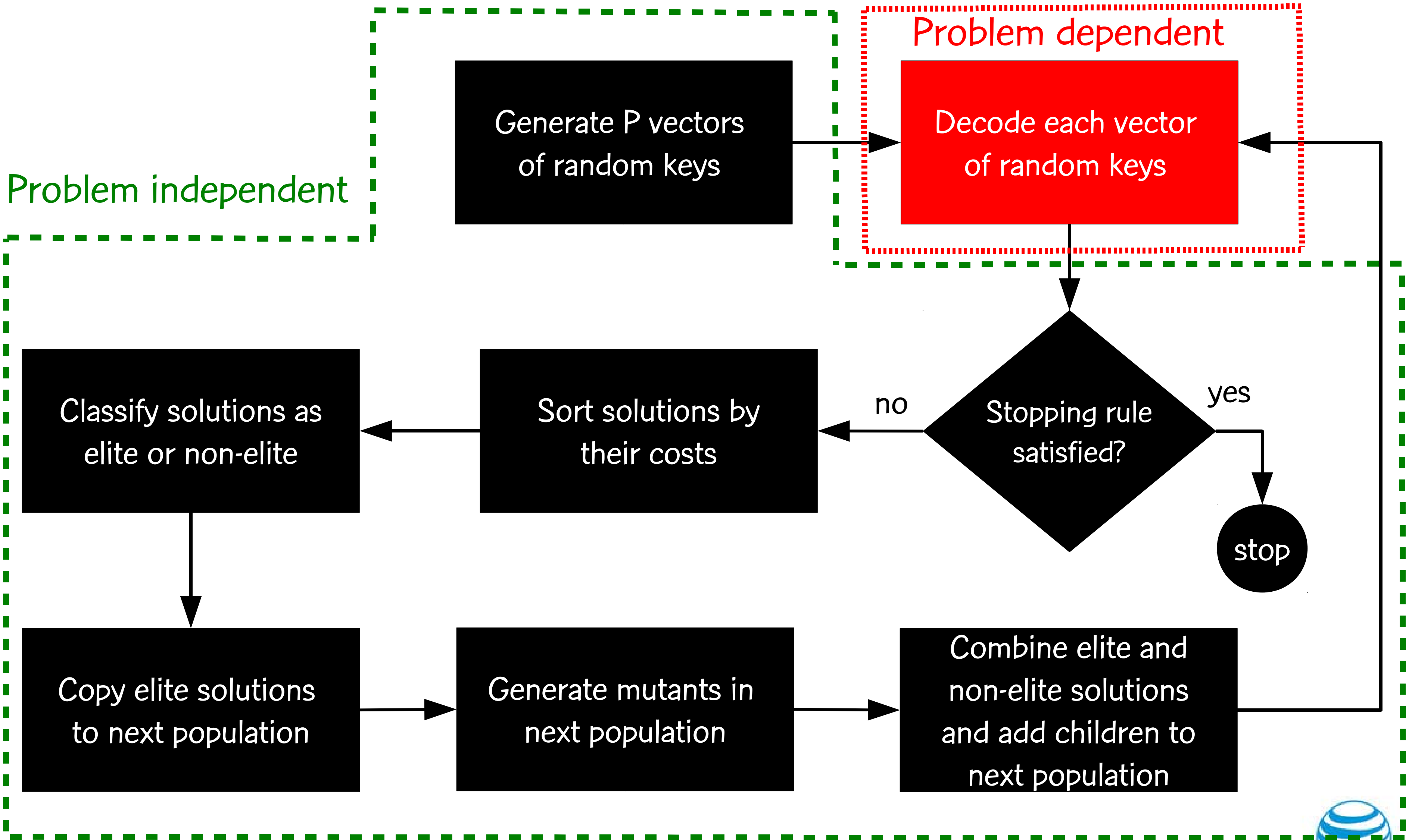
Framework for biased random-key genetic algorithms



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 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: 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 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: 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 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: 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 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: 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 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: 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

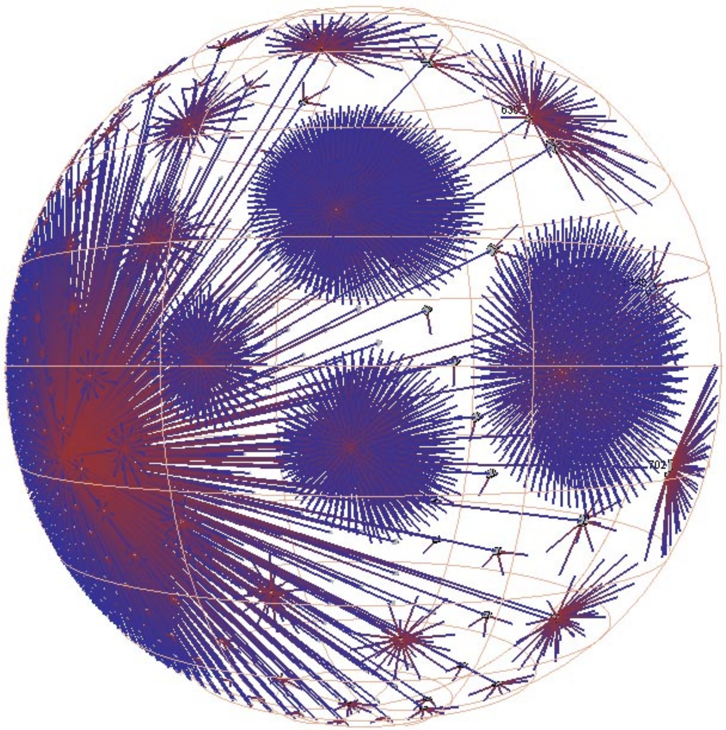
Applications in telecommunications

Applications in telecommunications

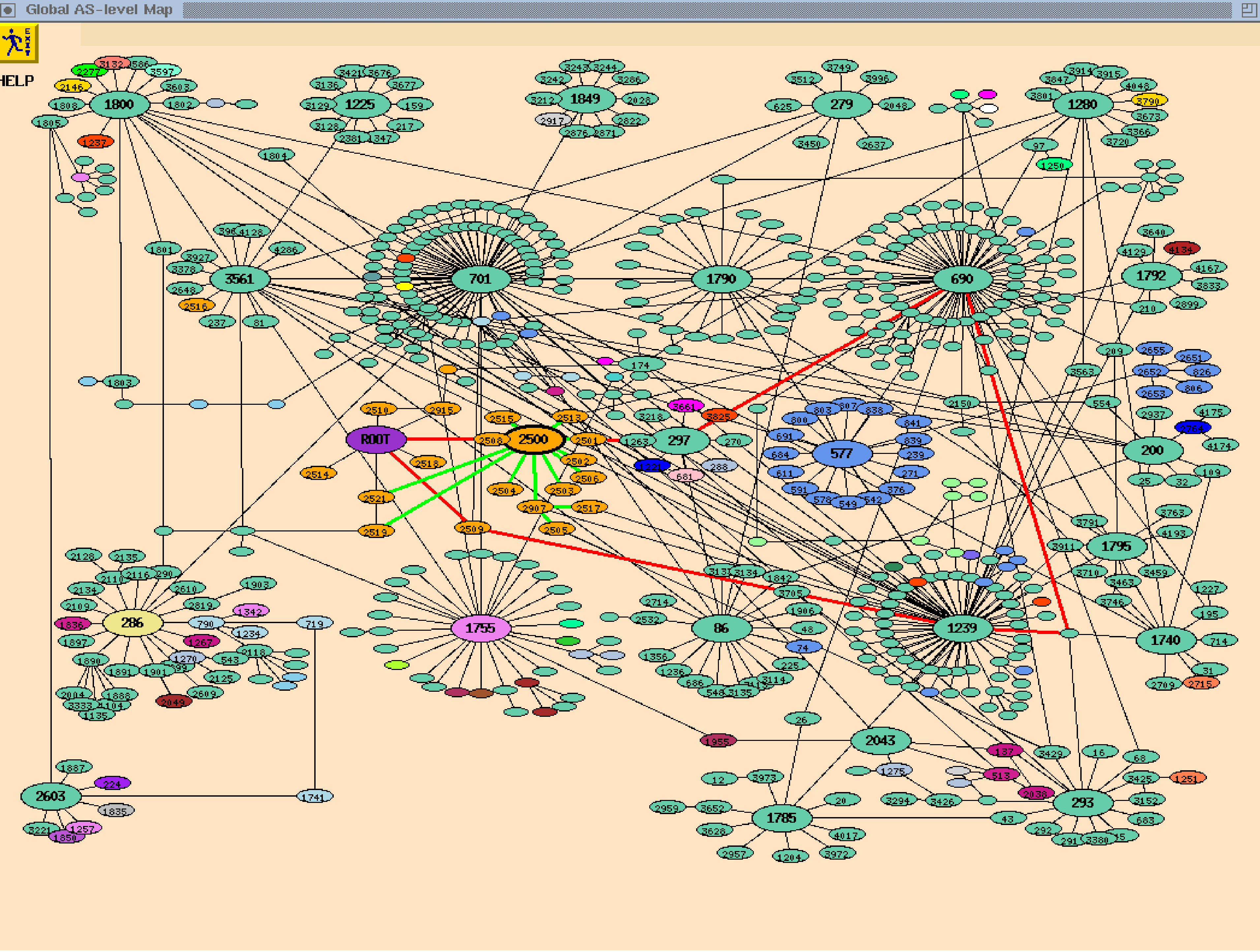
- Routing in IP networks
- Design of survivable IP networks
- Redundant server location for content distribution
- Regenerator location
- Routing and wavelength assignment in optical networks

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
1800
1225
1849
279
1280
3561
701
1790
690
1792
297
577
200
1795
286
1755
86
1239
1740
2043
2603
293
1785

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:

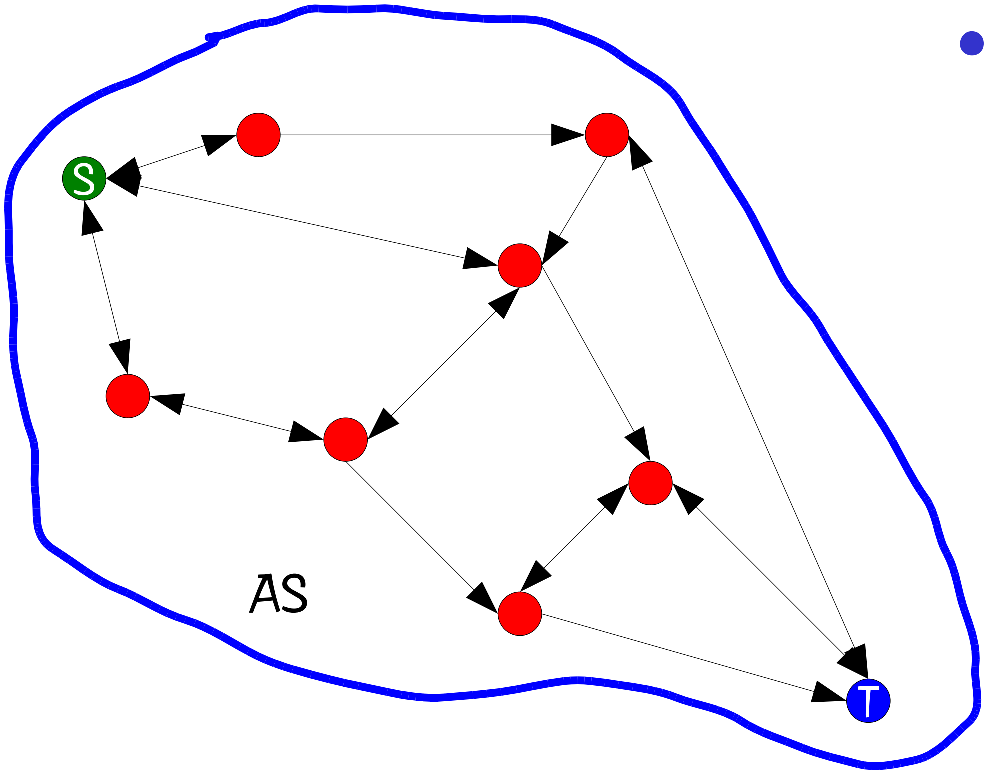
Routing

- 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

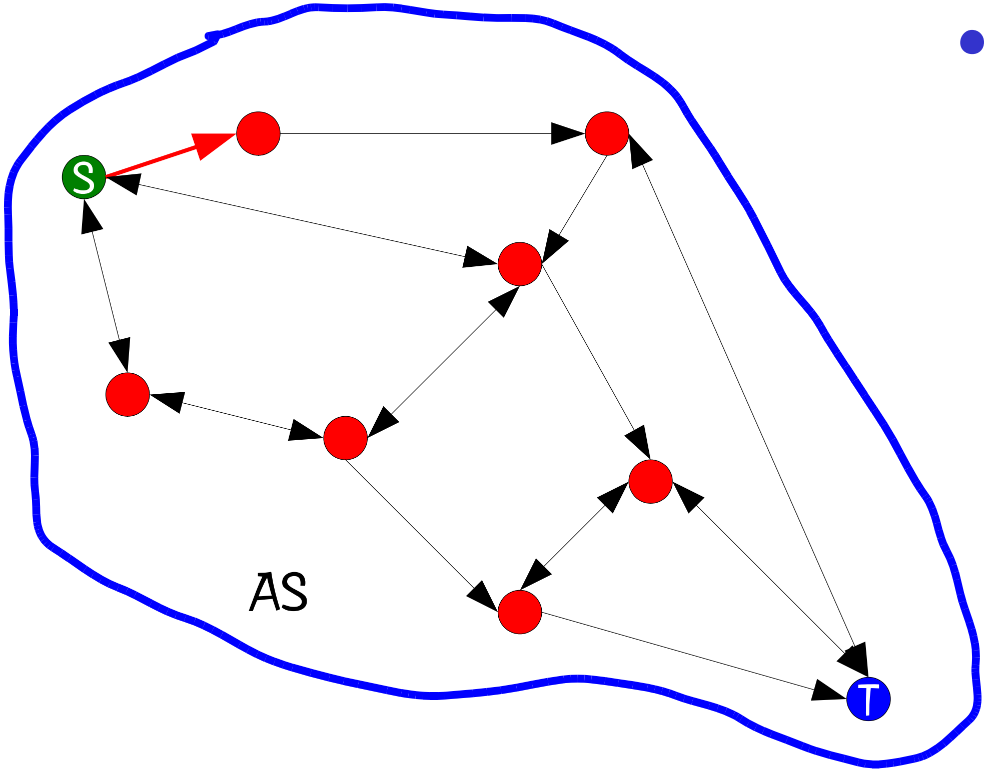
- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS: IGP routing
  - different ASes: BGP routing

IGP Routing



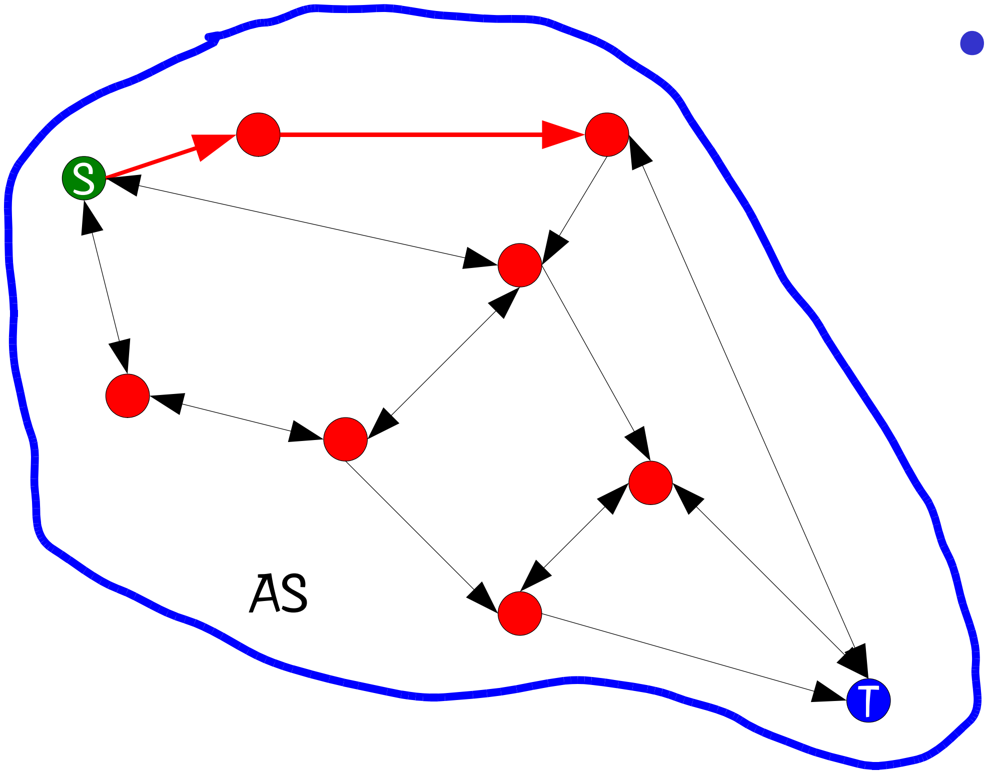
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



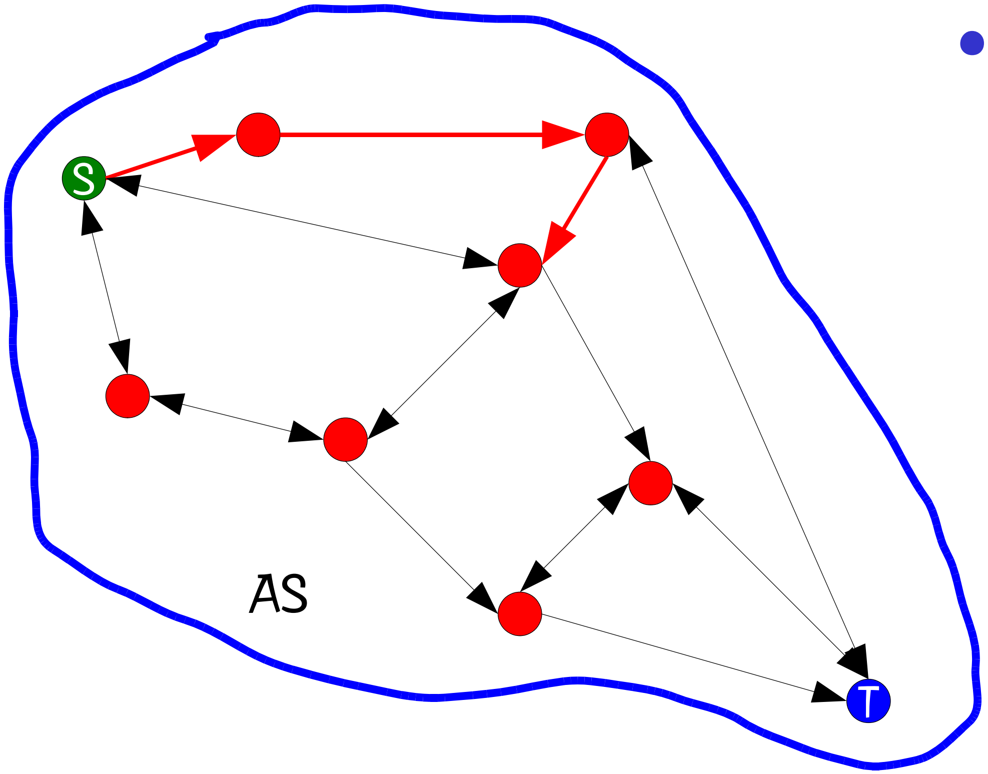
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



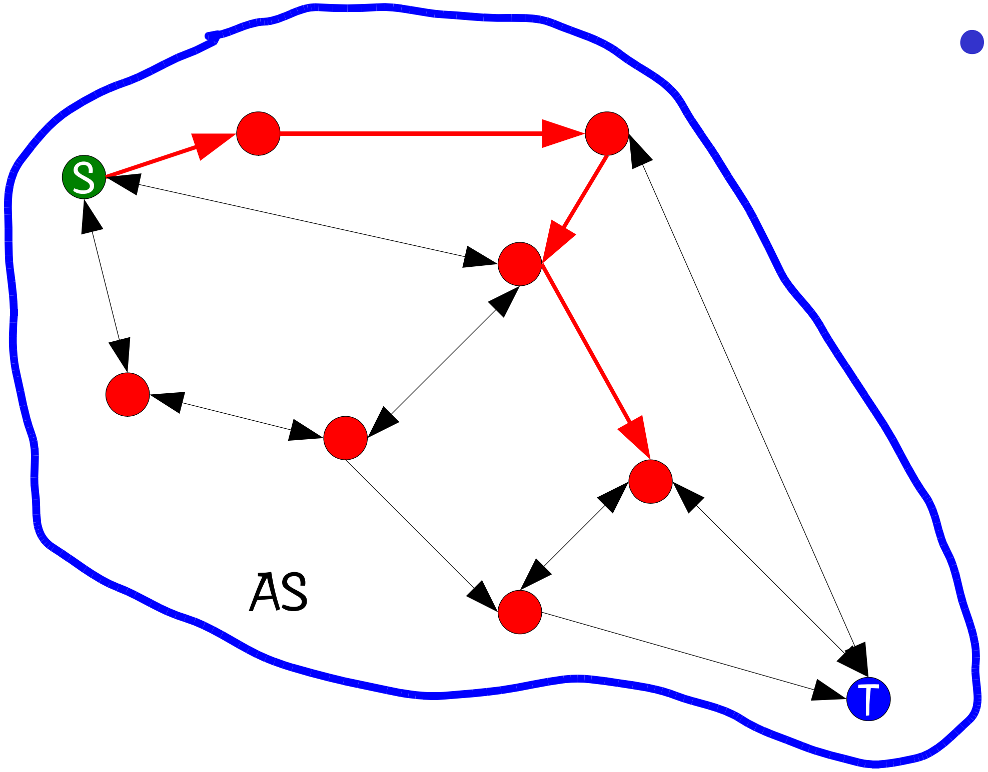
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



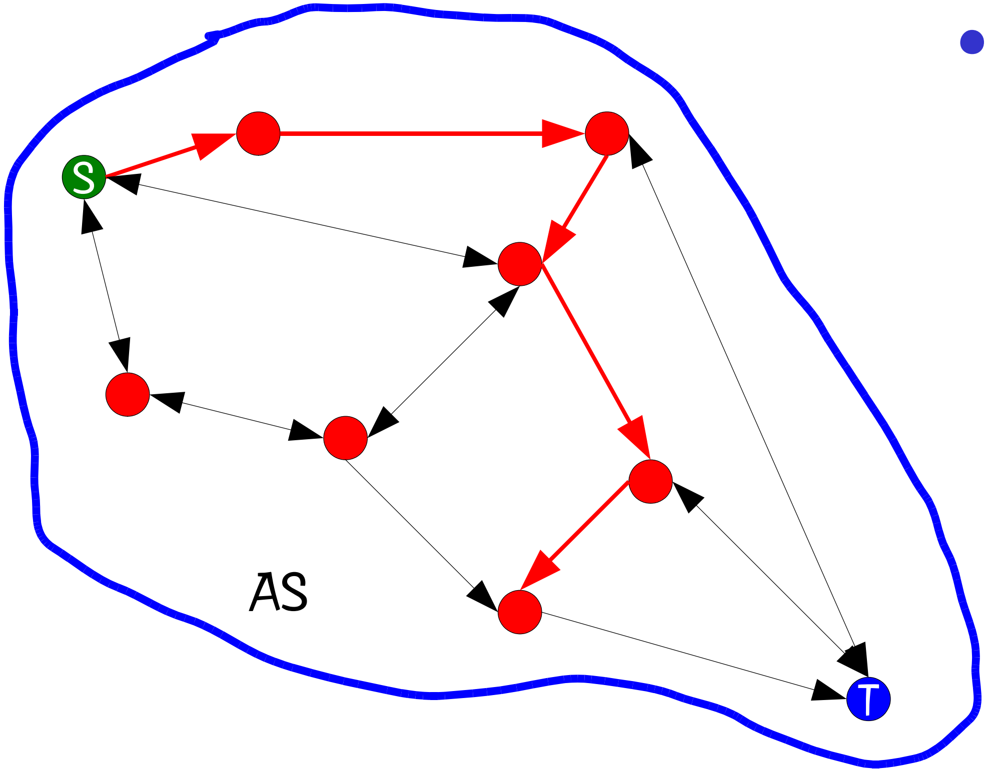
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



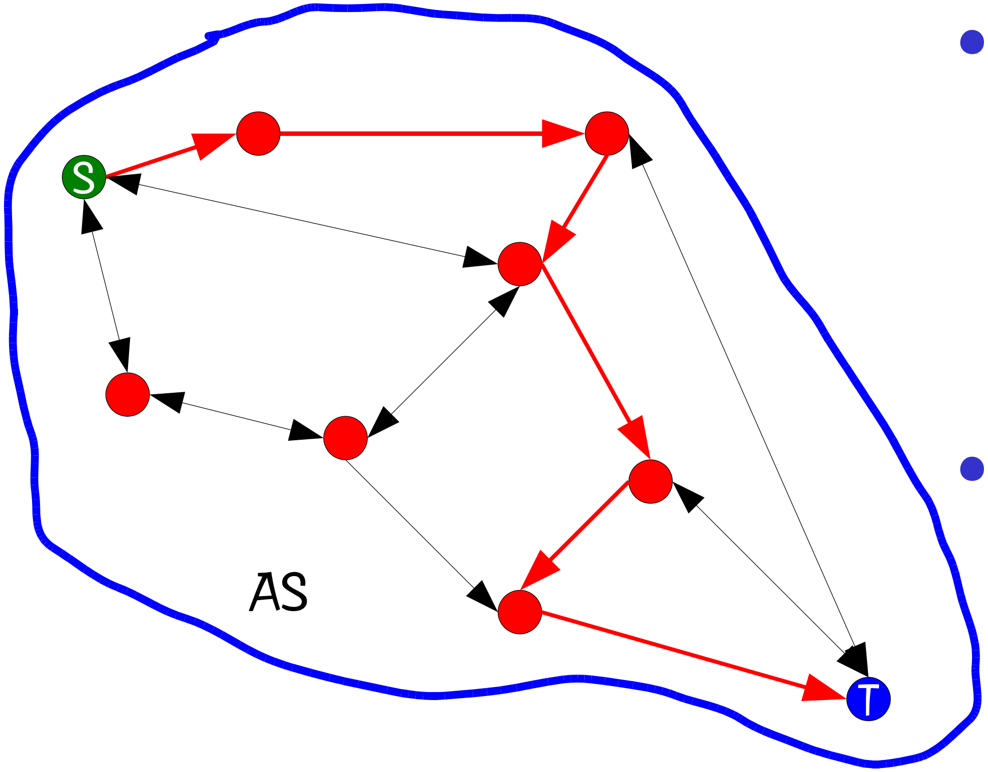
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



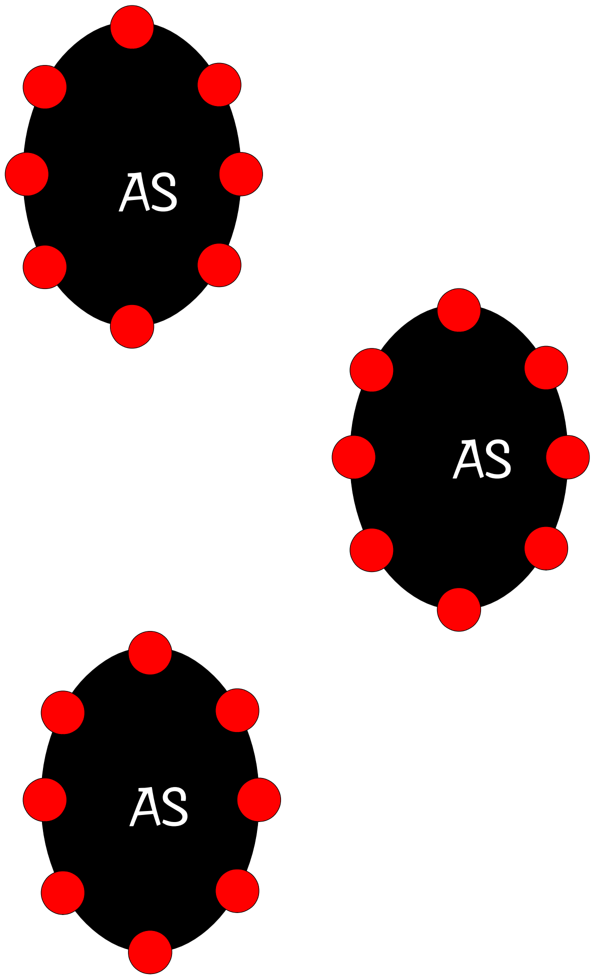
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



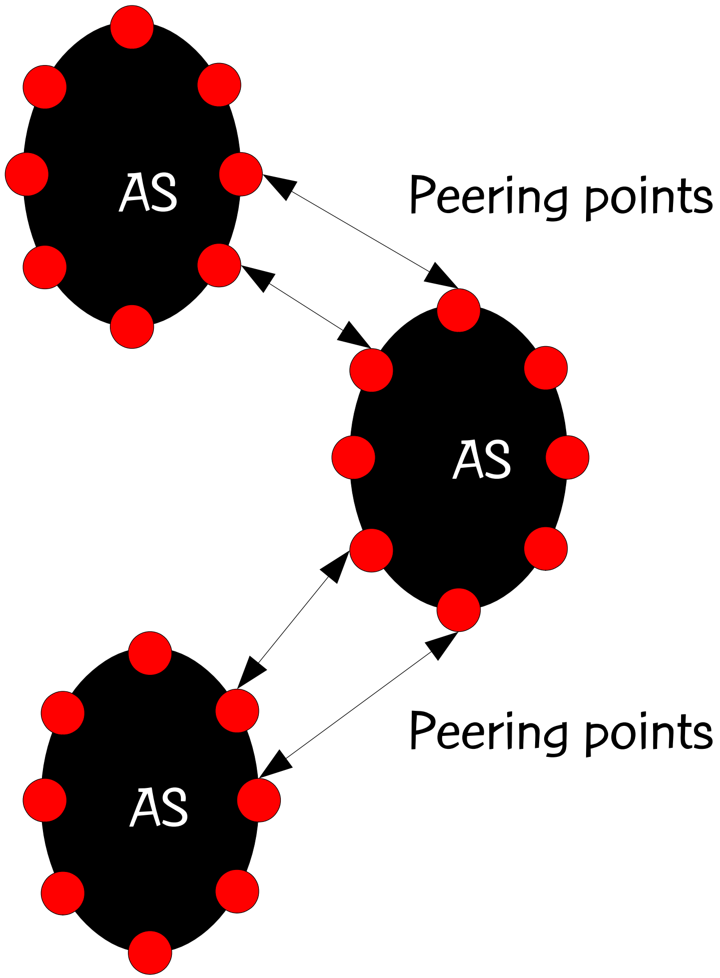
- IGP (interior gateway protocol) routing is concerned with routing within an AS.
- Routing decisions are made by AS operator.

BGP Routing



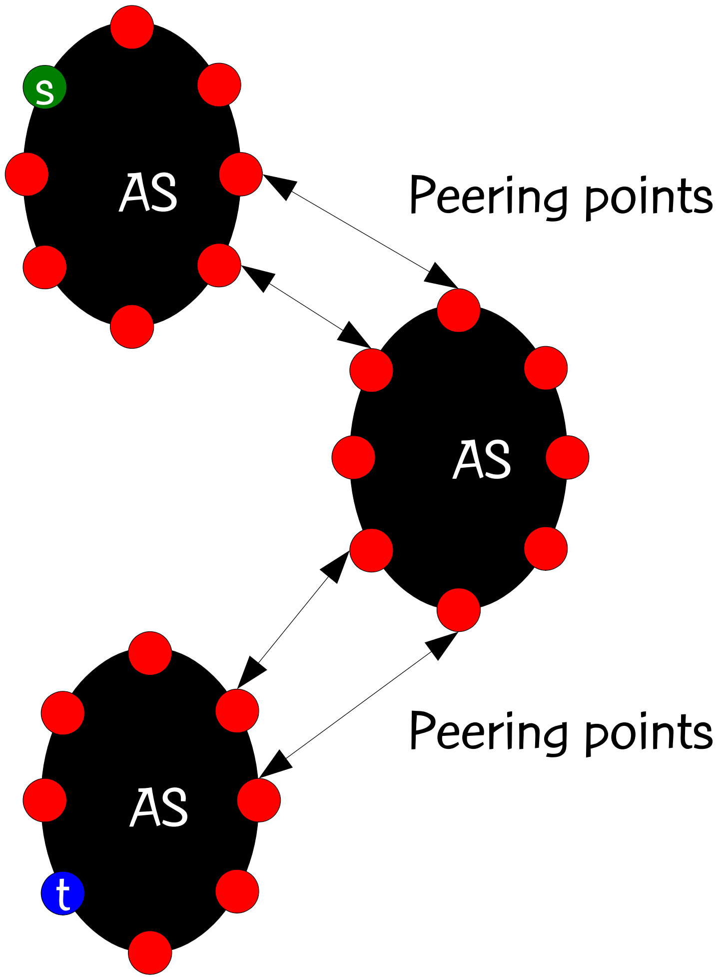
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



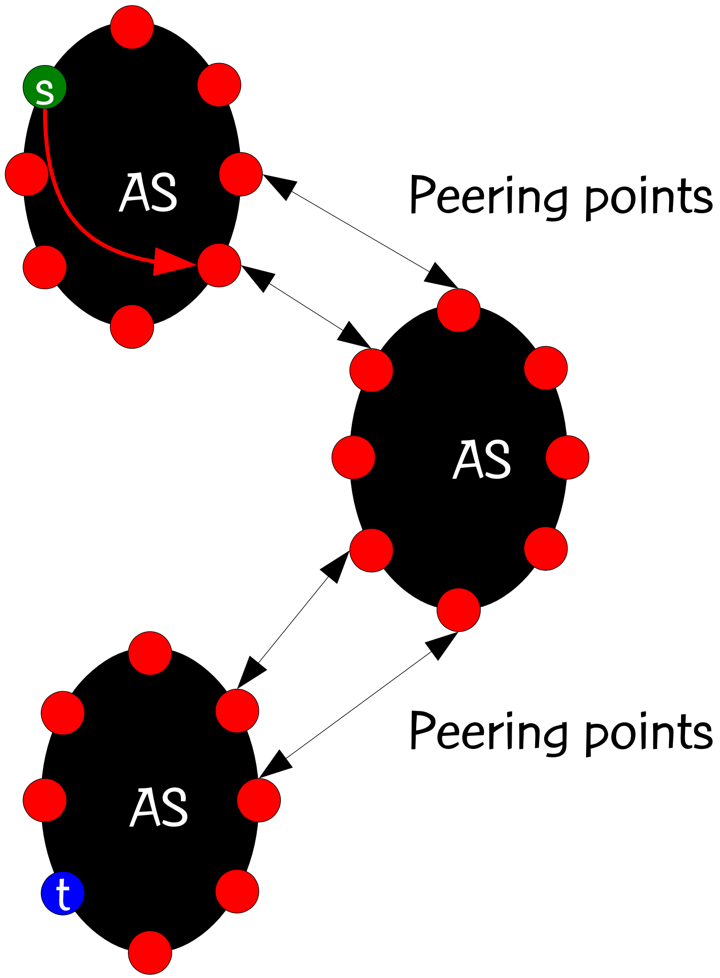
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



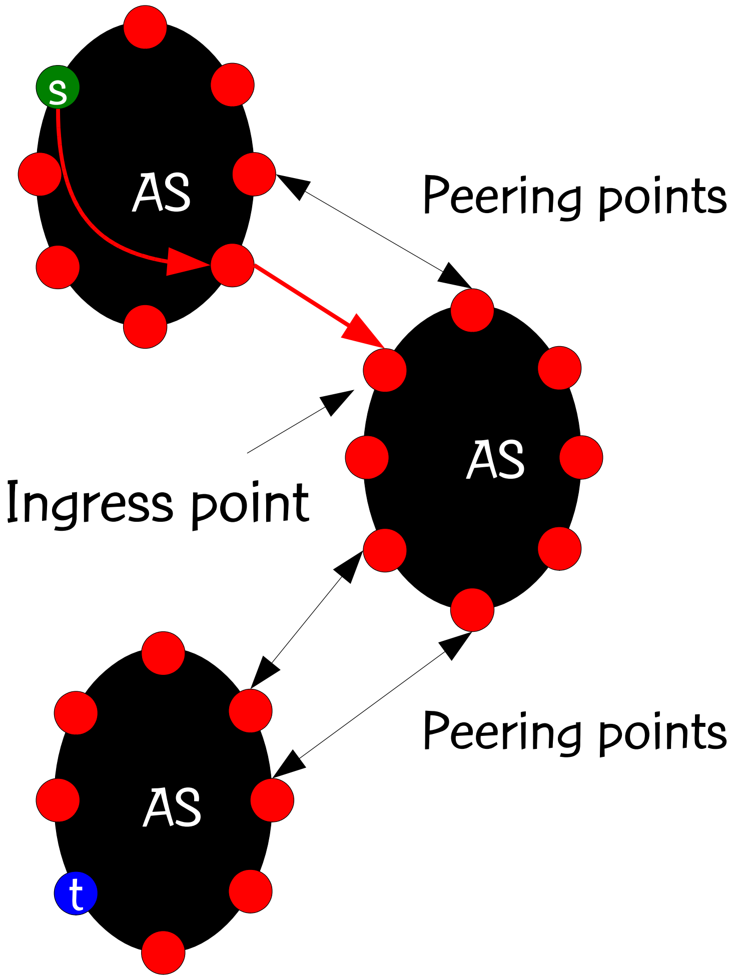
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



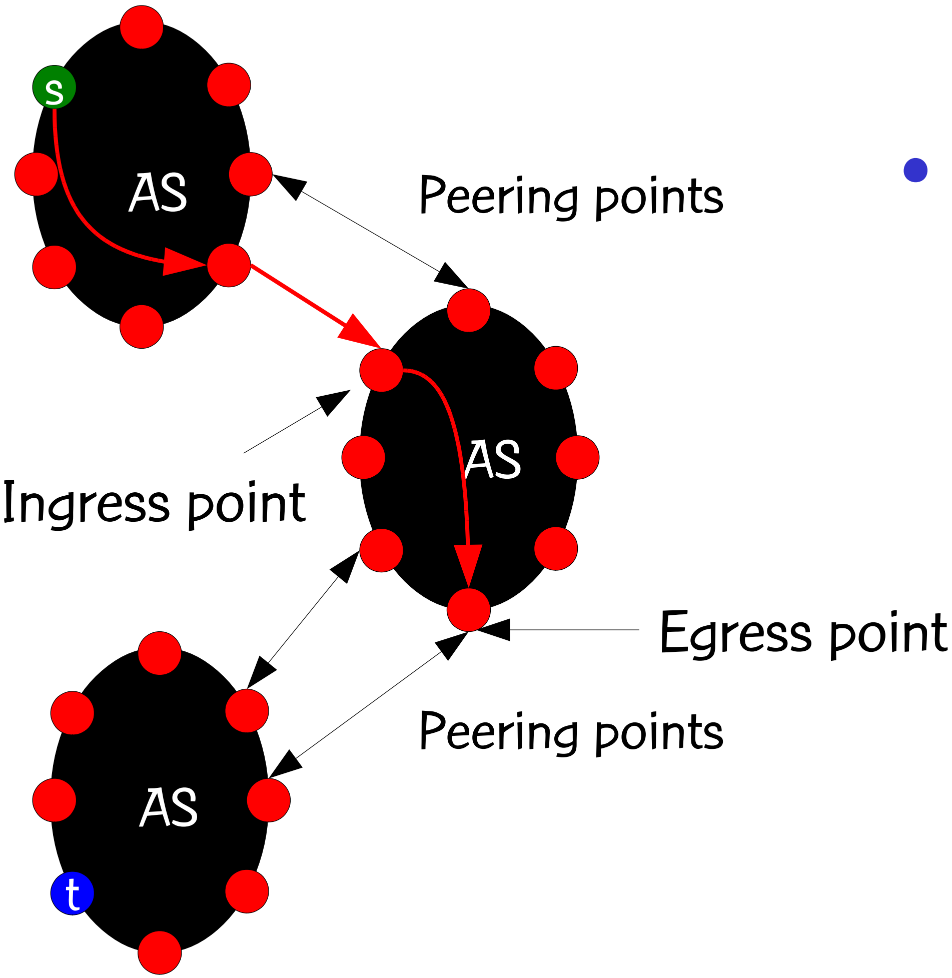
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



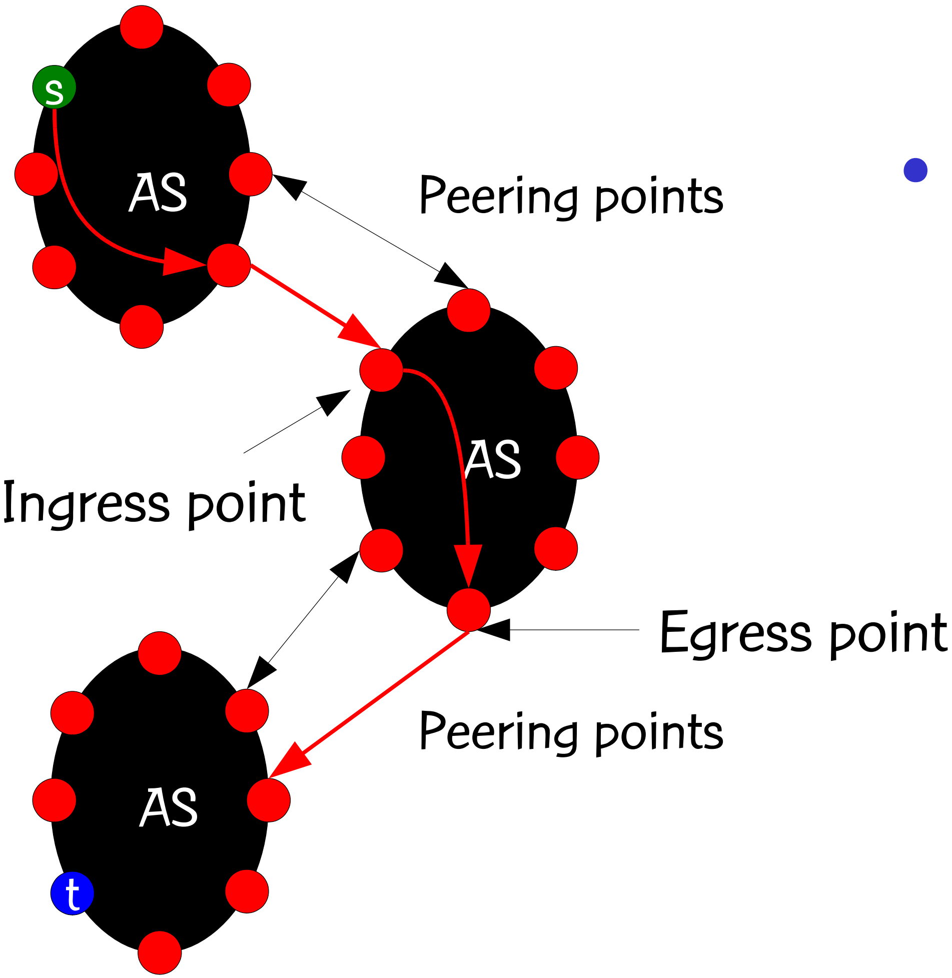
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



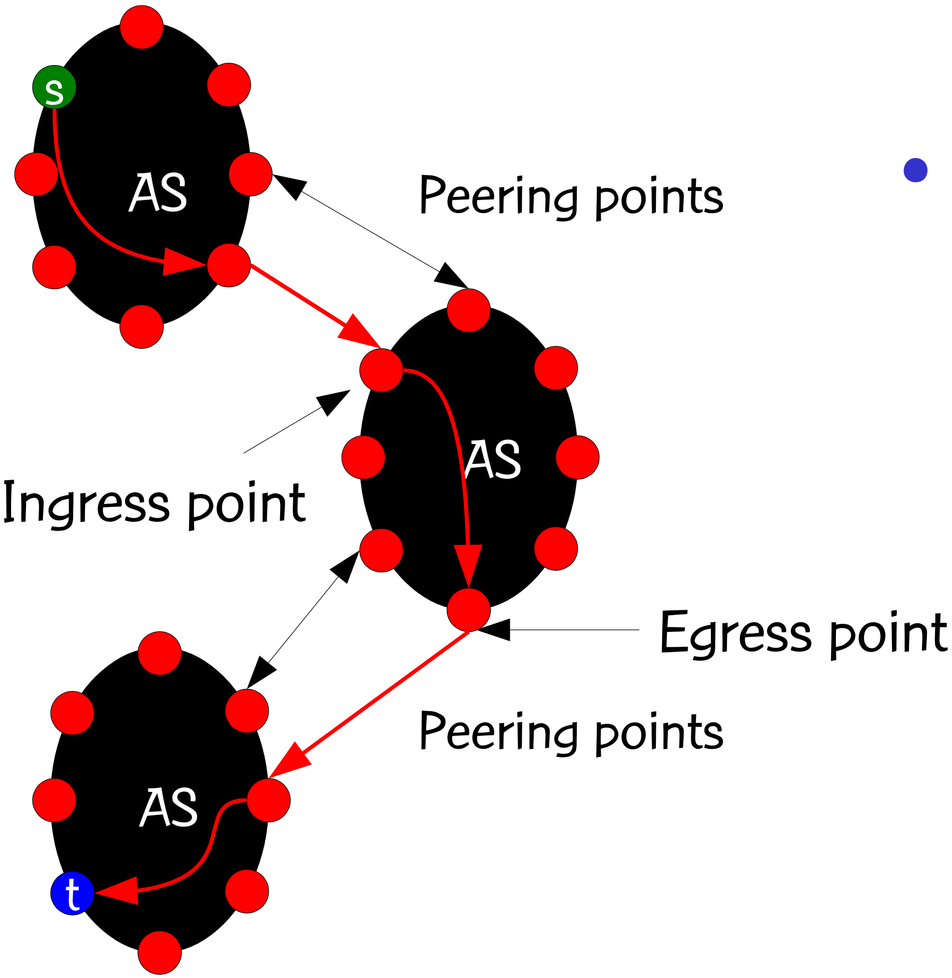
- BGP (border gateway protocol) routing deals with routing between different ASes.

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.

IGP Routing

OSPF routing

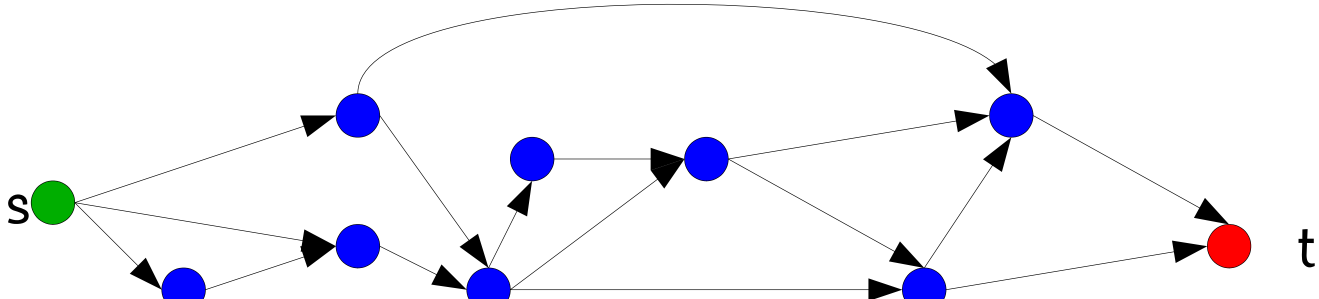
- Given a network $G = (N, A)$, where N is the set of routers and A is the set of links.

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.

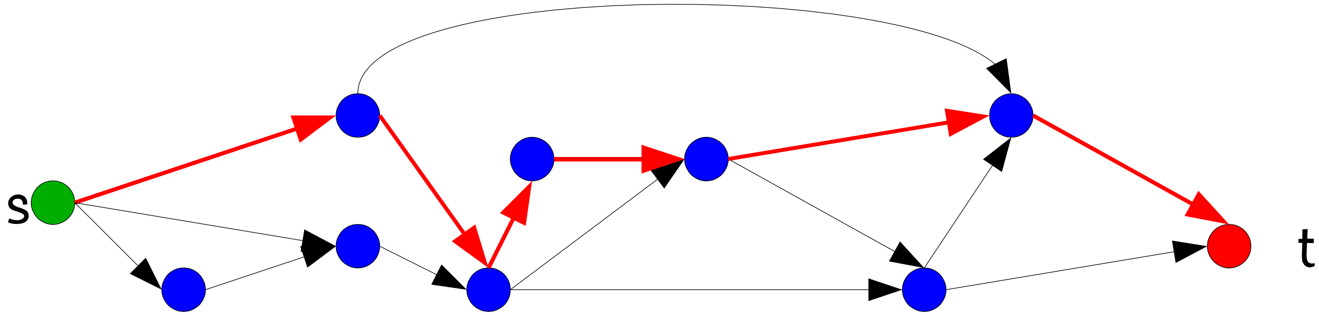
OSPF routing

- 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.



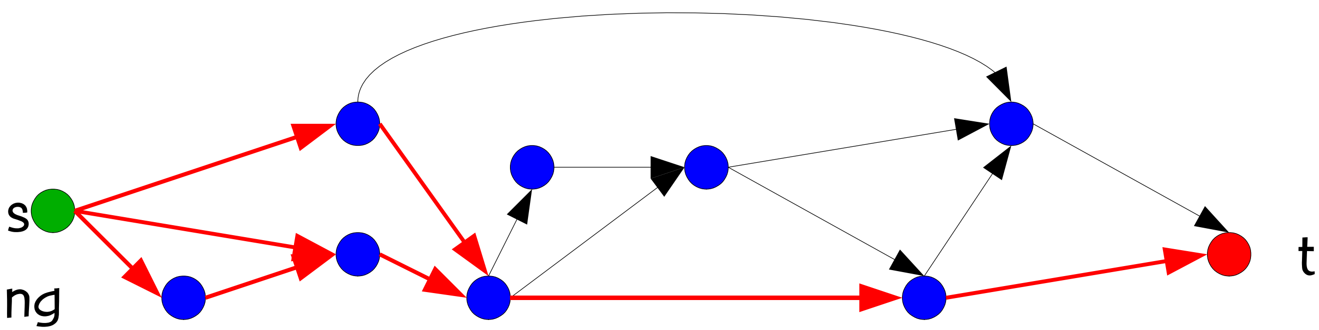
OSPF routing

- 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)

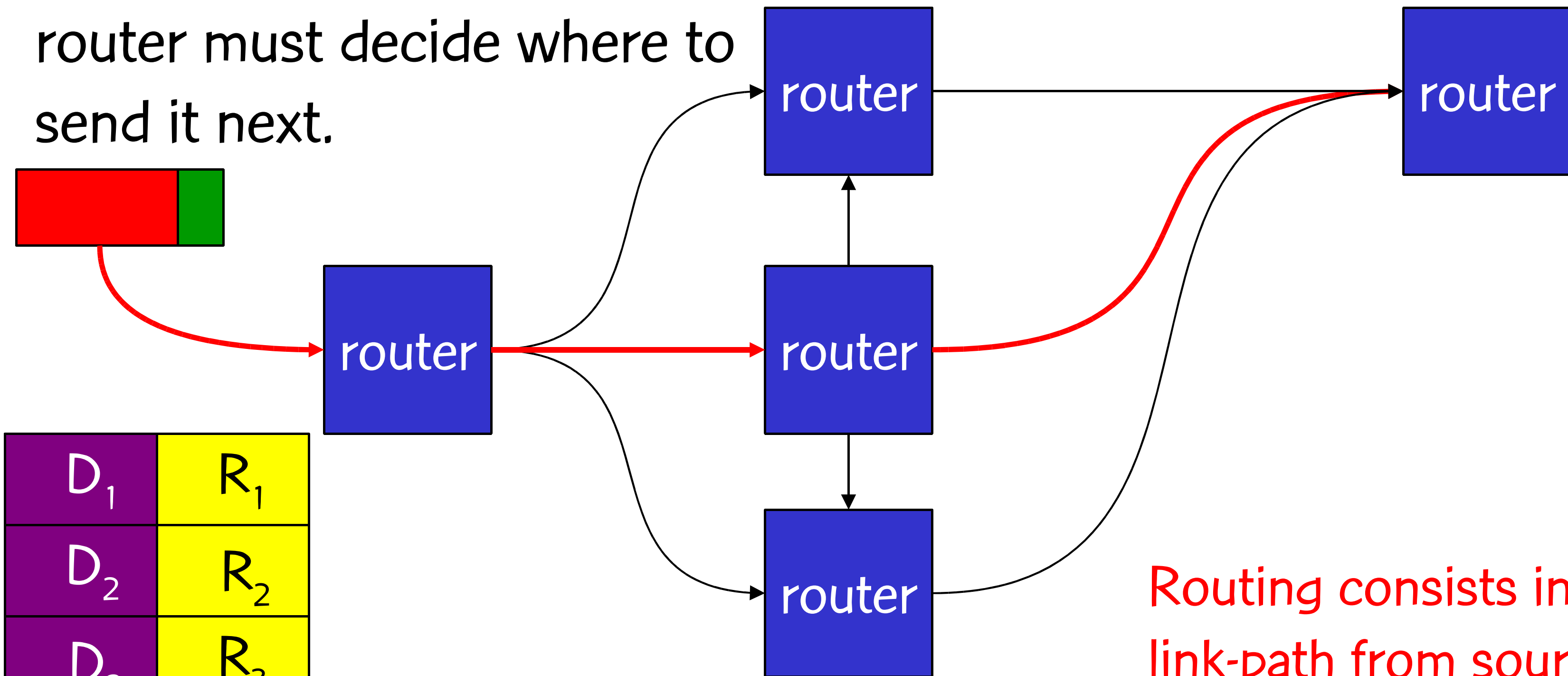
OSPF routing

- 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.



D_1	R_1
D_2	R_2
D_3	R_3
D_4	R_4

Routing table

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

OSPF routing

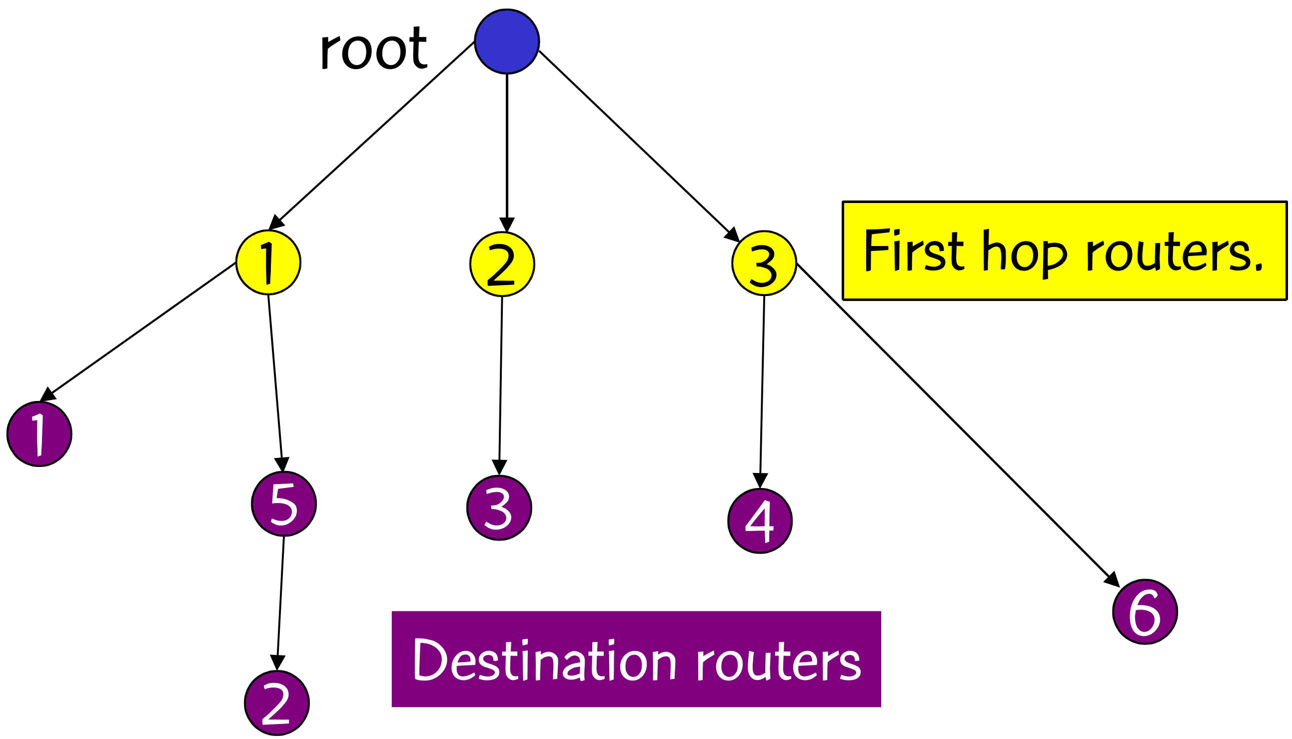
- Assign an integer weight $\in [1, w_{max}]$ to each link in AS. In general, $w_{max} = 65535 = 2^{16} - 1$.
- Each router computes tree of shortest weight paths to all other routers in the AS, with itself as the root, using Dijkstra's algorithm.

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

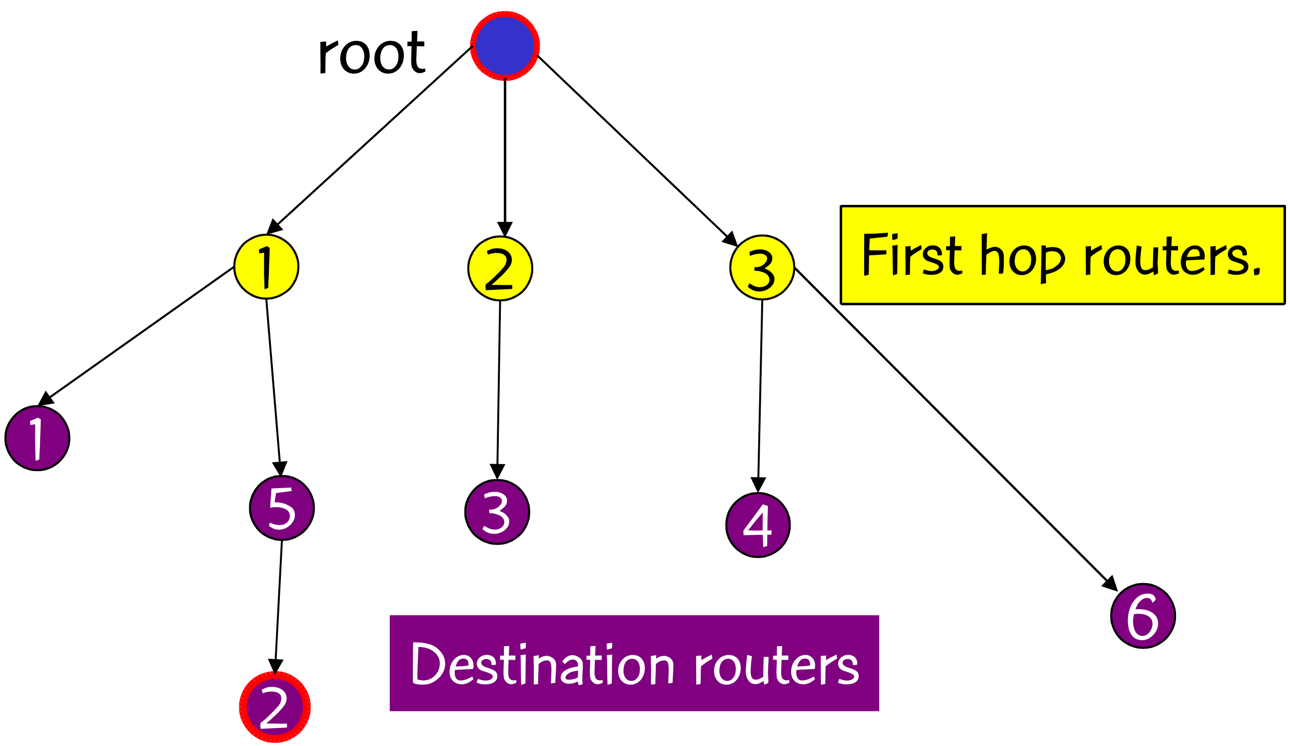


OSPF routing

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

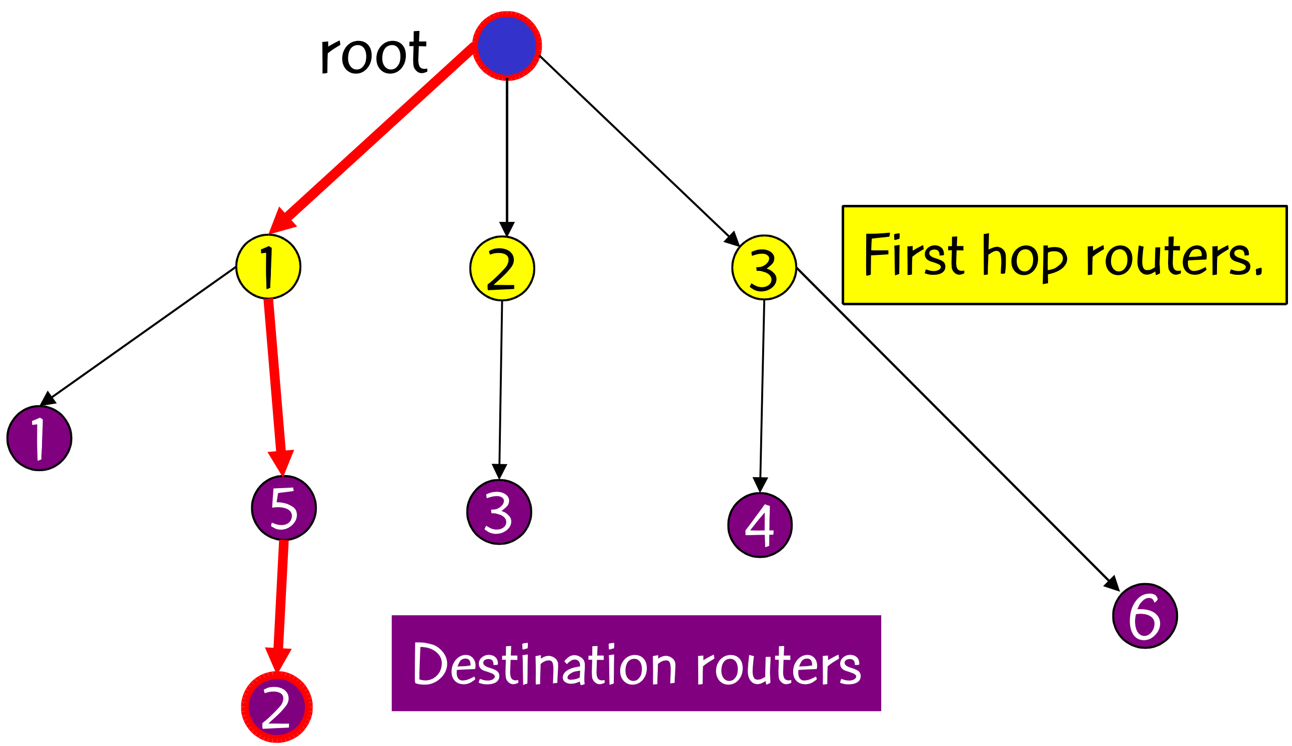


OSPF routing

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

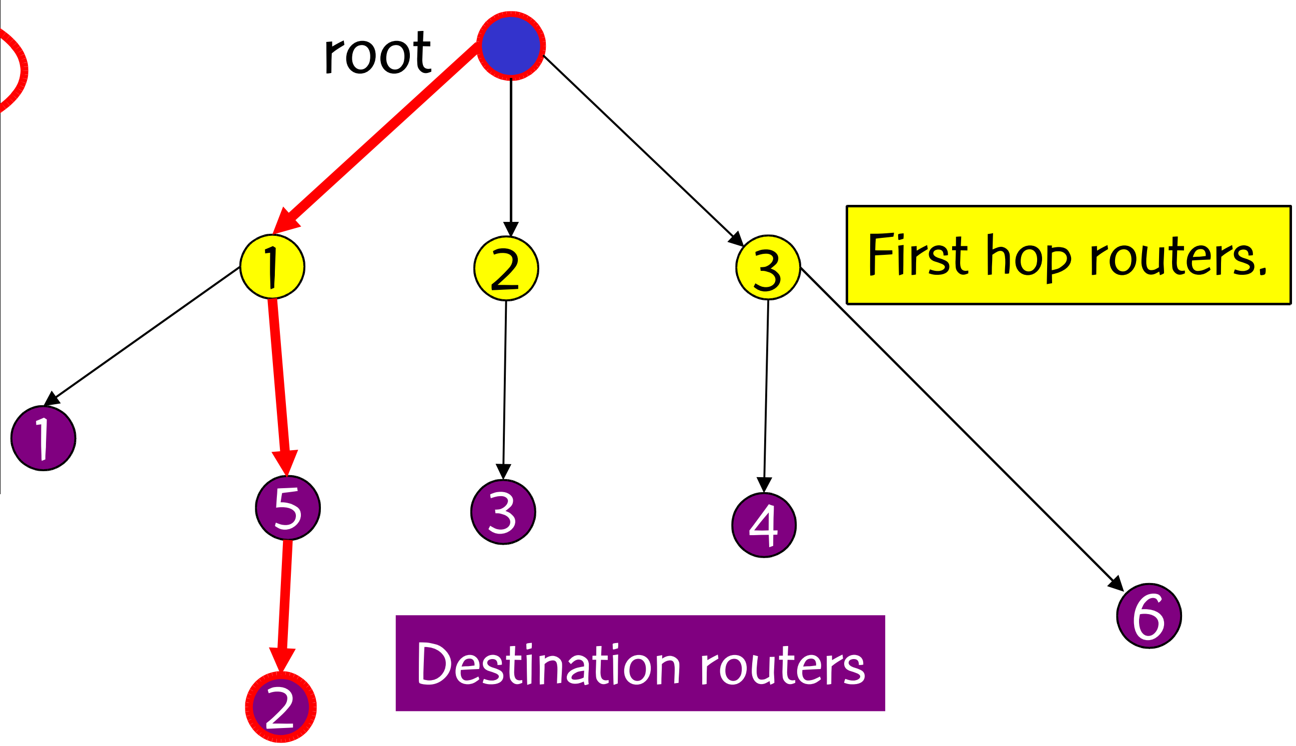


OSPF routing

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

Routing table

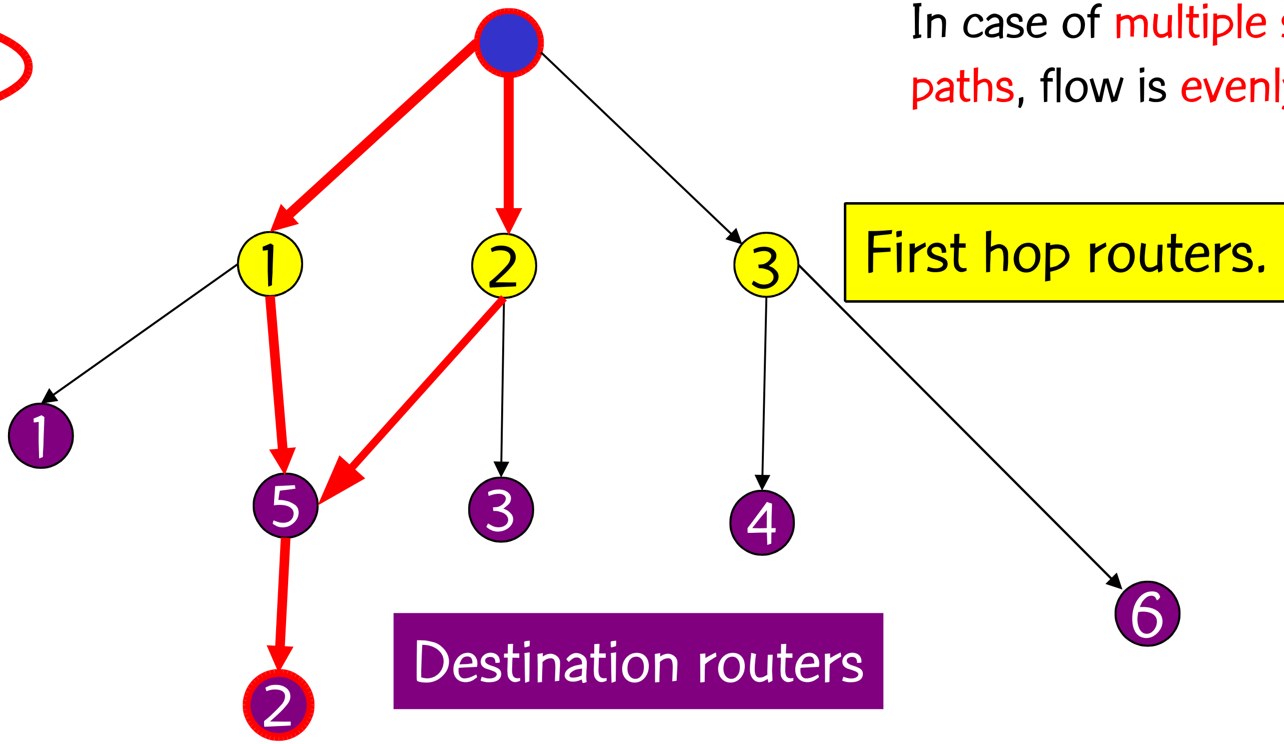
D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



OSPF routing

Routing table

D_1	R_1
D_2	R_1, R_2
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. In case of multiple shortest paths, flow is evenly split.

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:

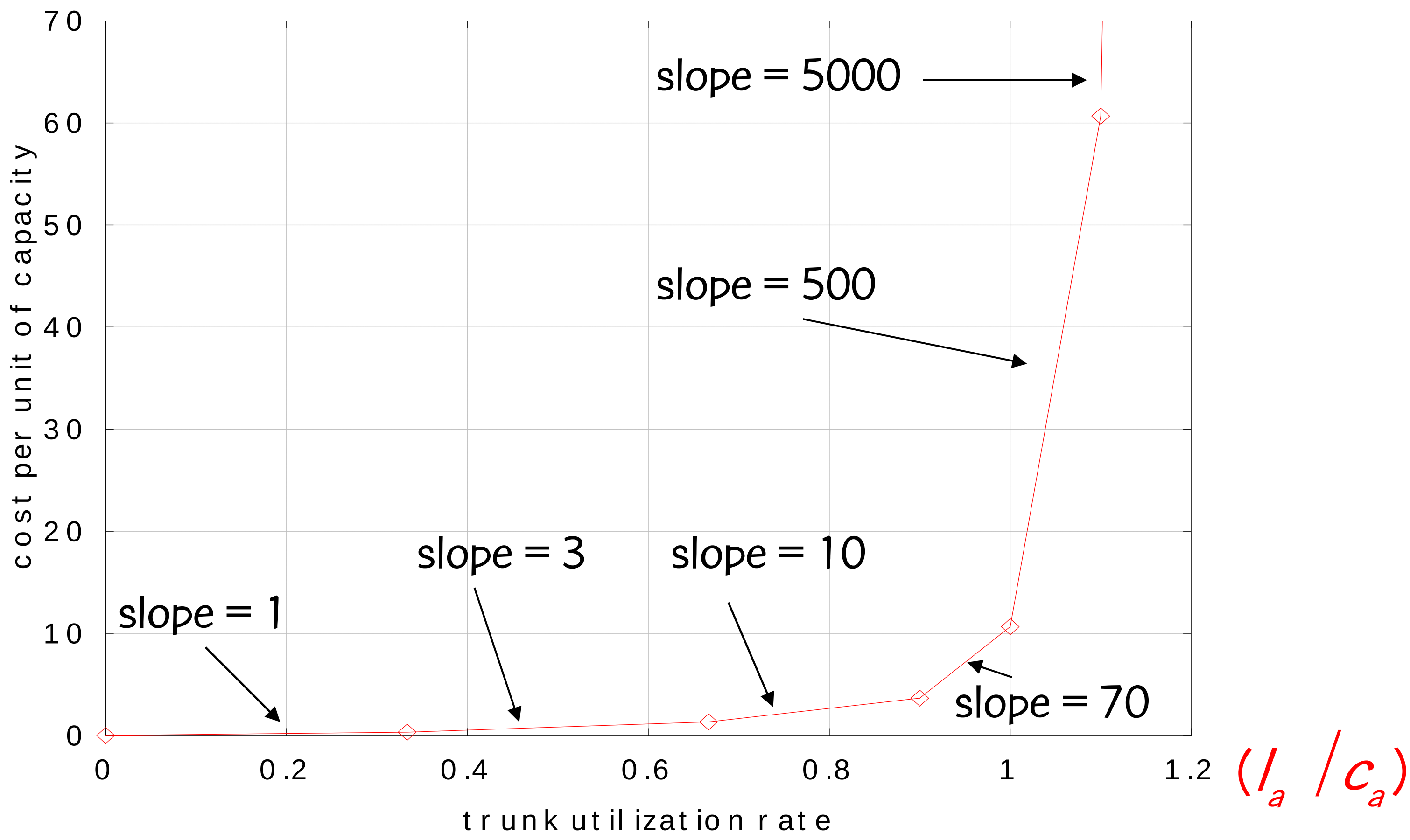
$$\Phi = \Phi_1(l_1) + \Phi_2(l_2) + \ldots + \Phi_{|A|}(l_{|A|})$$

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.

Tech report version:

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

BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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.

BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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.
- Decoding:

BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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.
- Decoding:
  - For $i = 1, ..., N$: set $w(i) = \text{ceil}\left(X(i) \times w_{max} \right)$

BRKGA for OSPF routing in IP networks

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

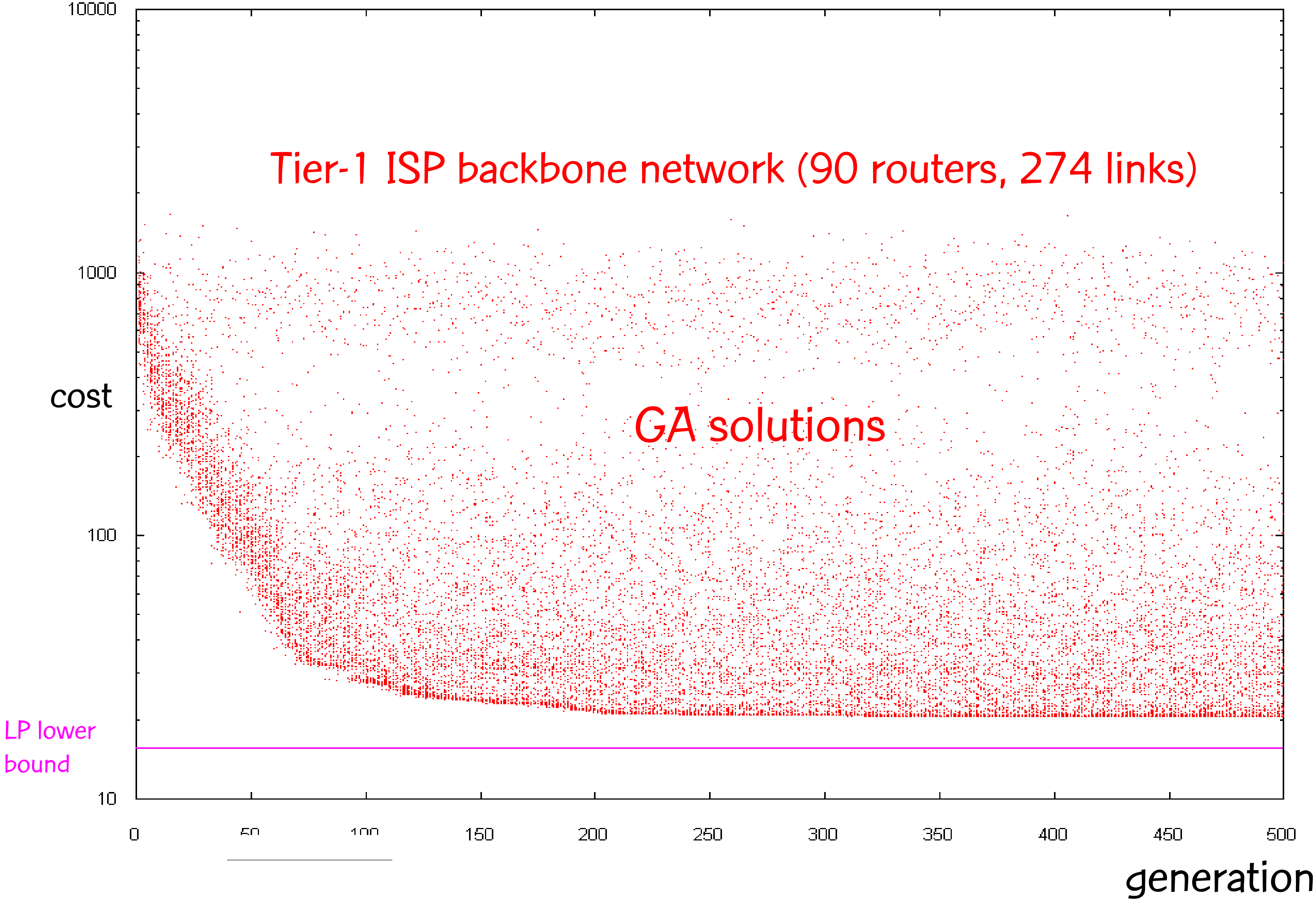
- Encoding:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoding:
  - For $i = 1, ..., N$: set $w(i) = \text{ceil}\ (X(i) \times w_{max})$
  - Compute shortest paths and route traffic according to OSPF.

BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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.
- Decoding:
  - For $i = 1, \ldots, N$: 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.

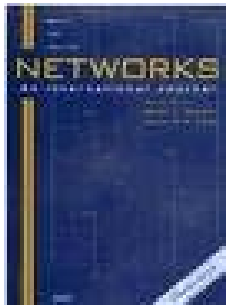
at&t
Your world. Delivered.



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

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

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, pp. 36–56, 2005.

Tech report version:

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

Improved BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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.

Improved BRKGA for OSPF routing in IP networks

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

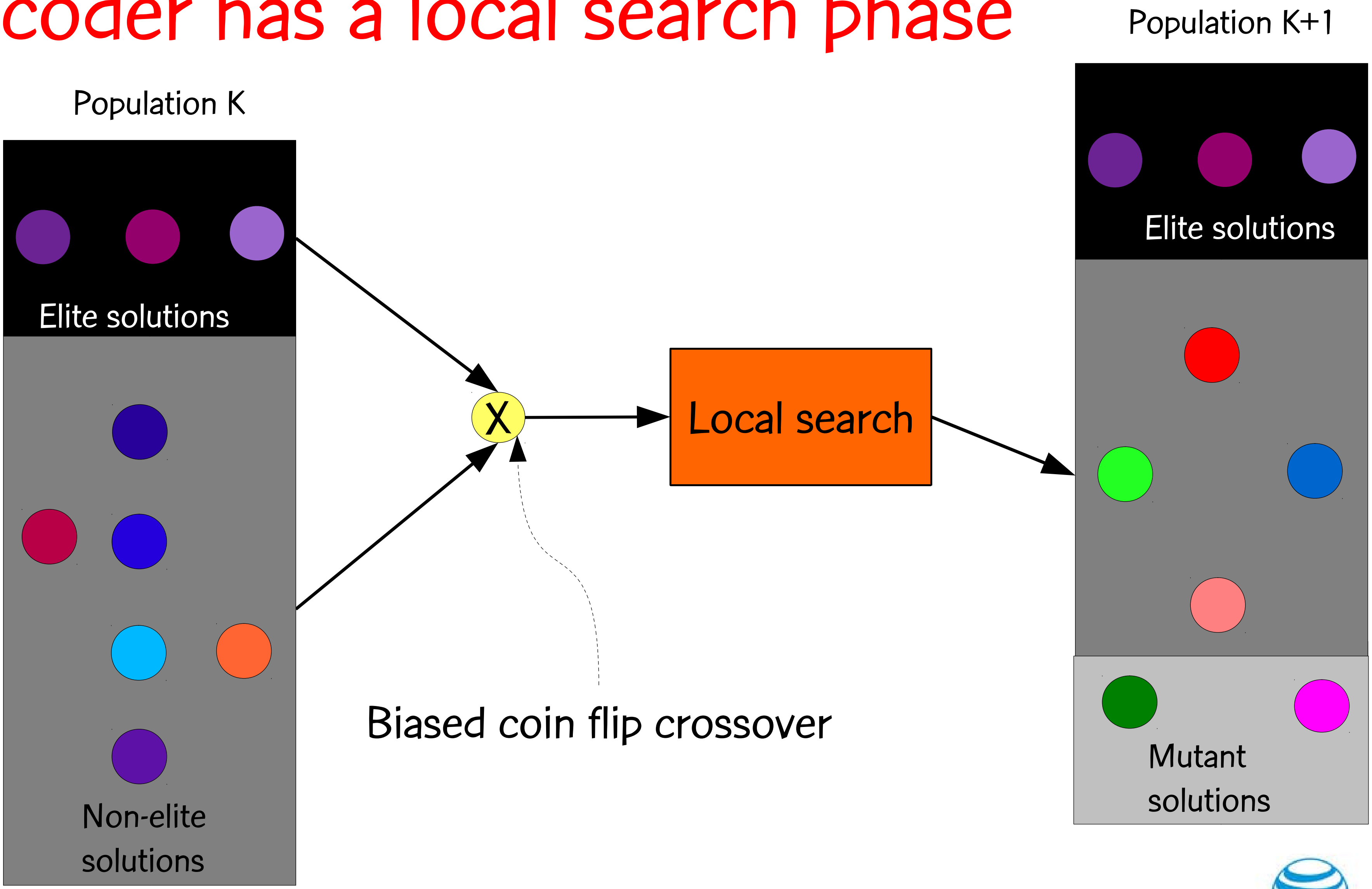
- Encoding:
  - A vector X of N random keys, where N is the number of links. The i-th random key corresponds to the i-th link weight.
- Decoder:
  - For $i = 1, \ldots, N$: set $w(i) = \text{ceil}\,(\,X(i) \times w_{max}\,)$
  - Compute shortest paths and route traffic according to OSPF.
  - Compute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

Improved BRKGA for OSPF routing in IP networks

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

- Encoding:
  - 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 $i = 1, \ldots, N$: 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.

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:

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

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

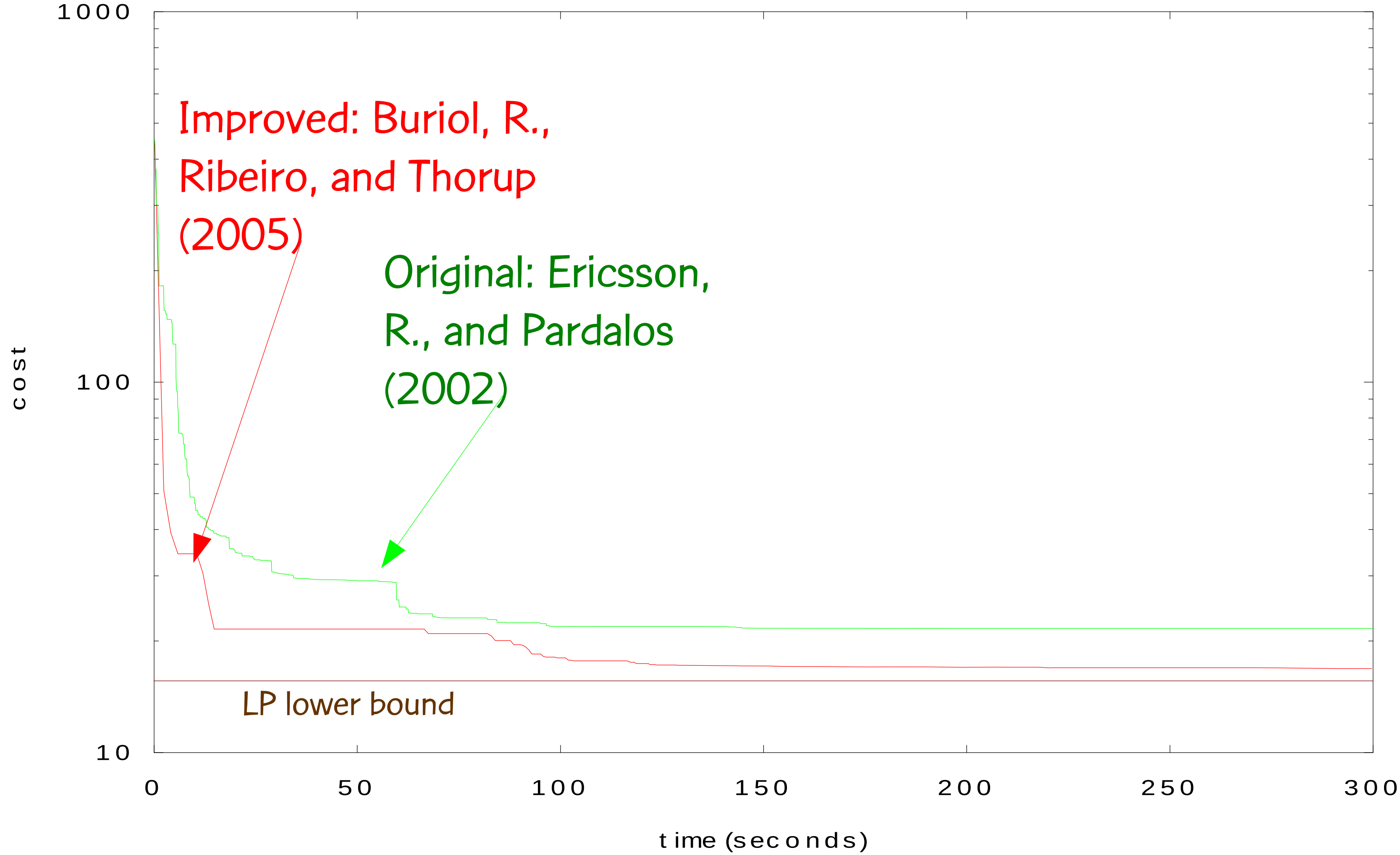
Fast local search

- 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

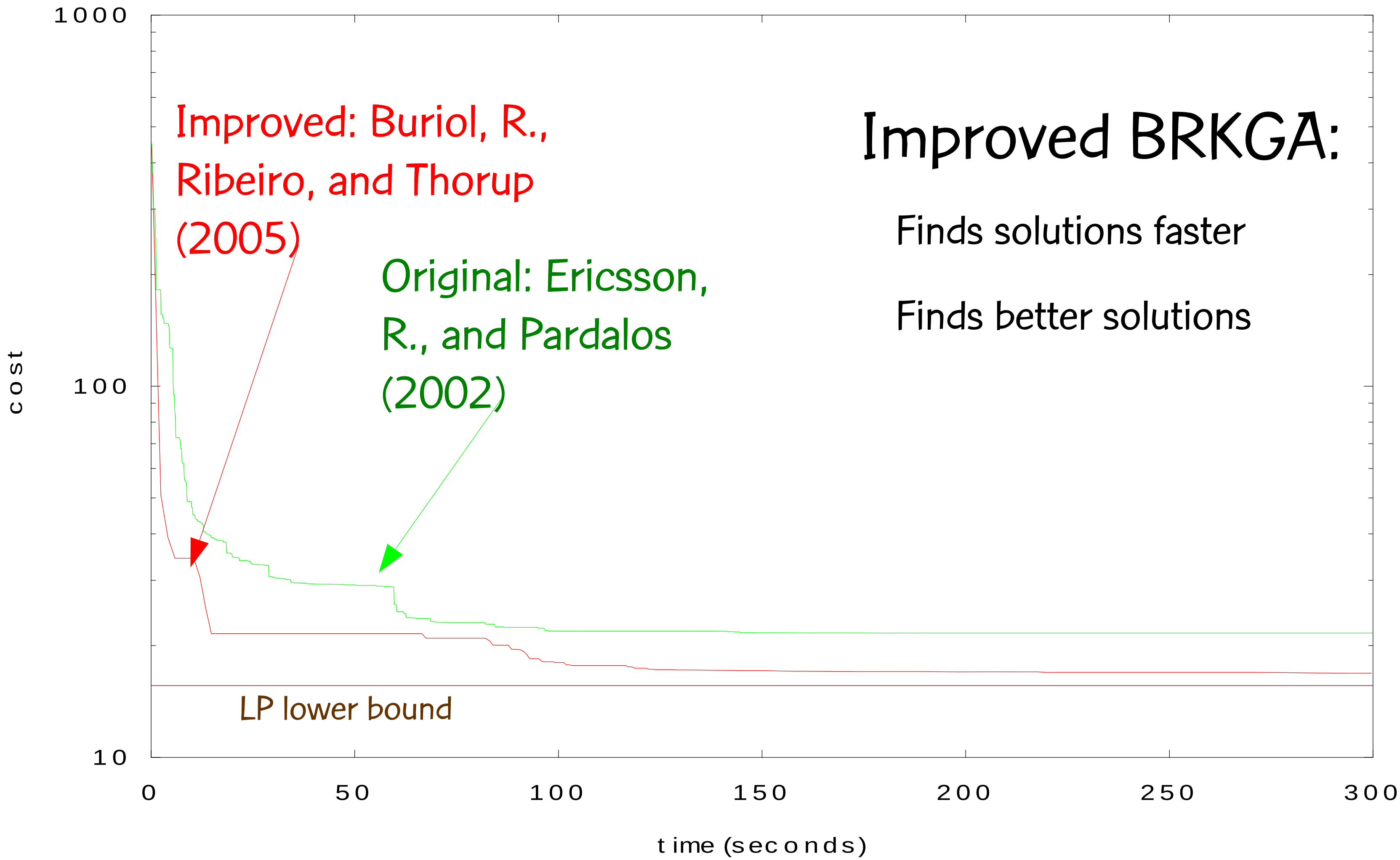
    $$[w_a, w_a + \lceil (w_{max} - w_a)/4 \rceil]$$

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

Effect of decoder with fast local search

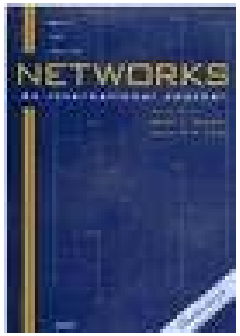


Effect of decoder with fast local search



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://www2.research.att.com/~mgcr/doc/gamult.pdf

Survivable IP network design

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

- Given

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,

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,

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

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.

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

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 $a \in A$,

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 $a \in A$,

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.

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

- Encoding:
  - 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.

Survivable IP network design

- Encoding:
  - 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:

Survivable IP network design

- Encoding:
  - 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 $w(i) = \text{ceil}\ (X(i) \times w_{max})$

Survivable IP network design

- Encoding:
  - 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 $i = 1, \ldots, N$: 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.

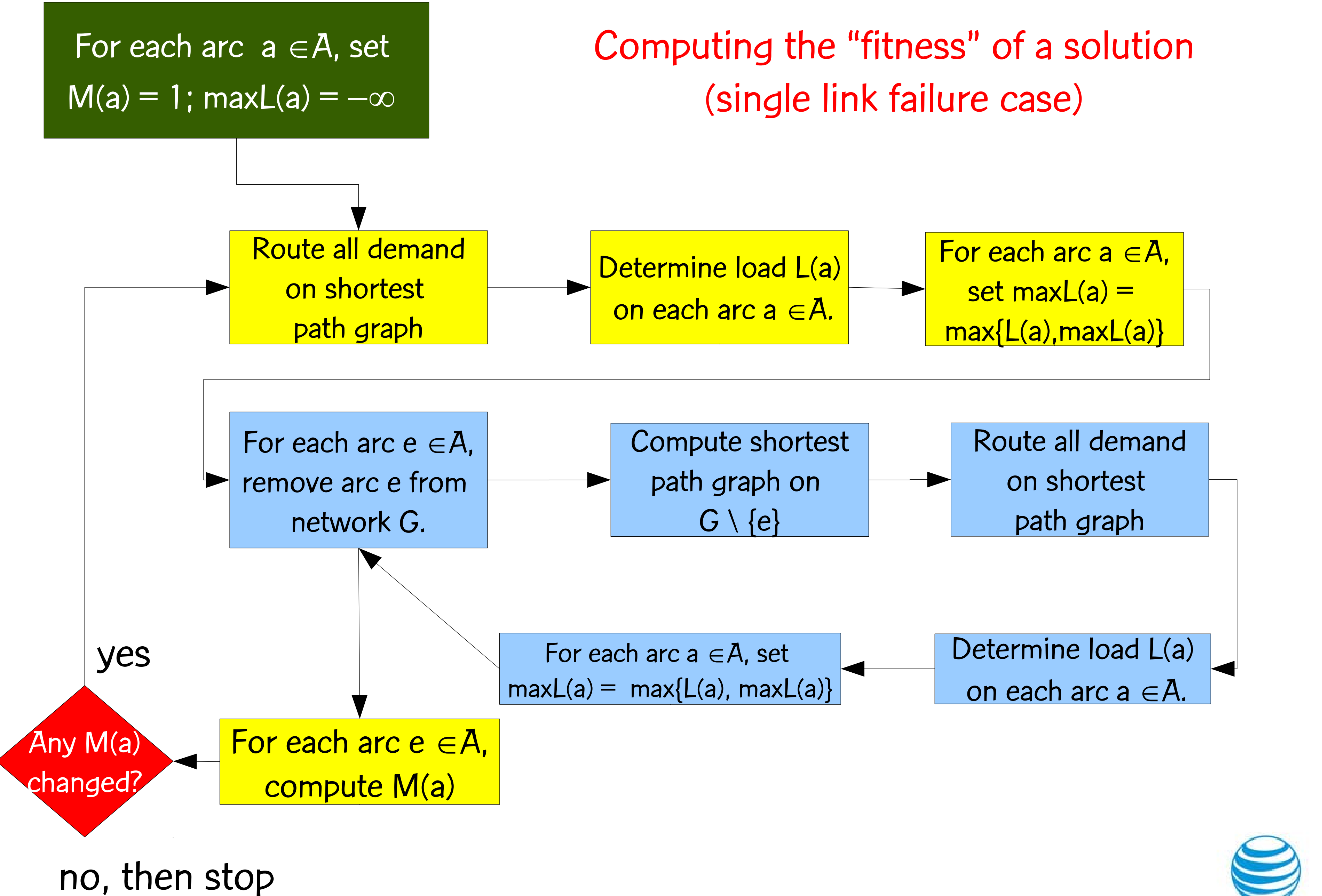
Survivable IP network design

- Encoding:
  - 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.

Survivable IP network design

- Encoding:
  - 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 $i = 1, \ldots, N$: 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 $\text{cost} = \sum_{a \in A} M(a) \times K(a)$

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



Composite-link design

- In Buriol, R., 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

D.V. Andrade, L.S. Buriol, M.G.C.R., and M. Thorup, "Survivable composite-link IP network design with OSPF routing," The Eighth INFORMS Telecommunications Conference, Dallas, Texas, April 2006.

Tech report:

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

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, ..., 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, ..., 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:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$: 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

- Encoding:
  - 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.

Survivable composite link IP network design

- Encoding:
  - 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:

Survivable composite link IP network design

- Encoding:
  - 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 $w(i) = \text{ceil}\ (X(i) \times w_{max})$

Survivable composite link IP network design

- Encoding:
  - 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 $i = 1, \ldots, N$: 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.

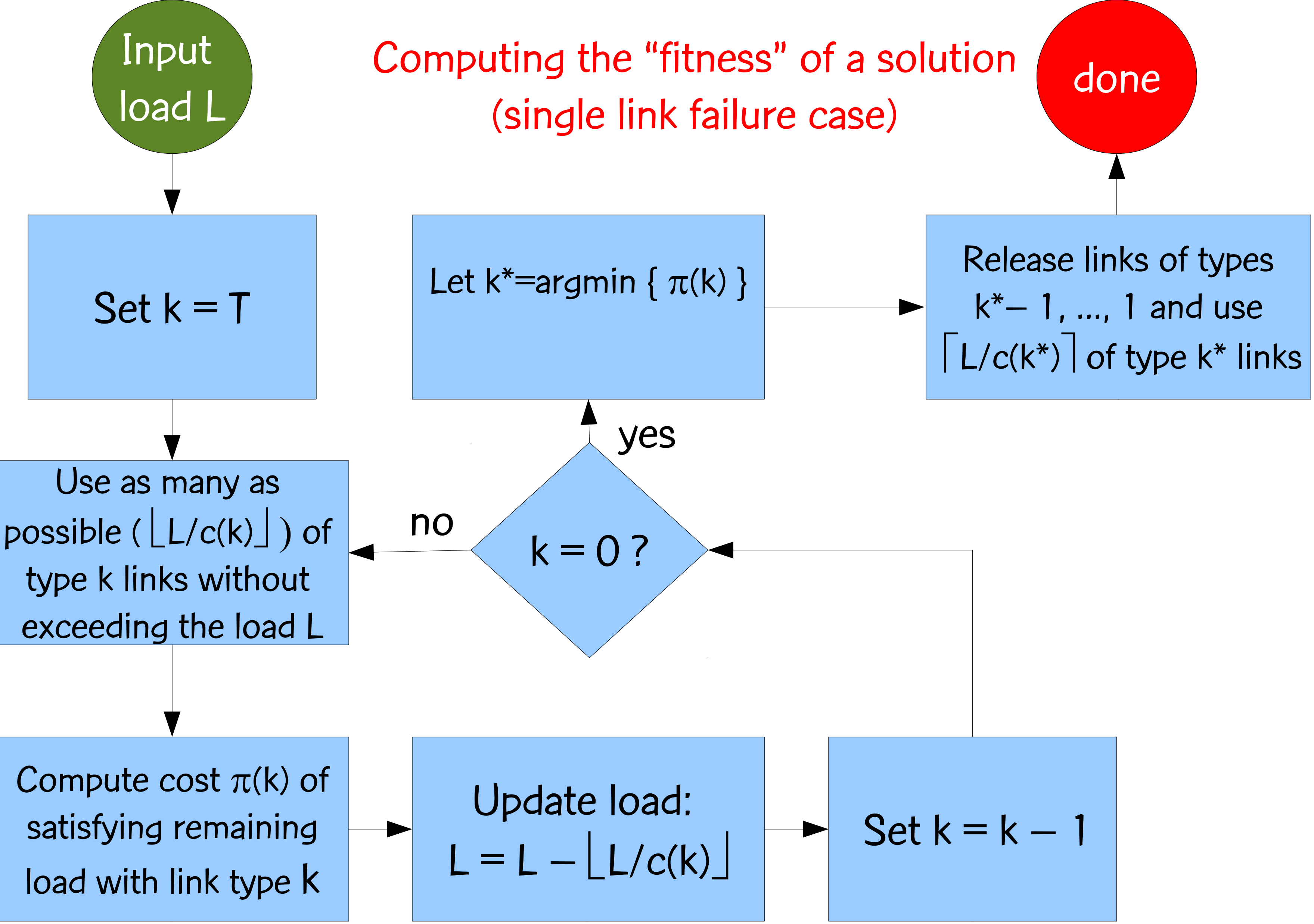
Survivable composite link IP network design

- Encoding:
  - 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 $i = 1, \ldots, N$: 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.

Survivable composite link IP network design

- Encoding:
  - 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 $i = 1, \ldots, N$: 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 $\text{cost} = \sum_{i \in A} \sum_{t \text{ used in arc } i} M(t,i) \times p(t)$

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



Redundant content distribution

Reference:

ALENEX11
Workshop on
Algorithm Engineering & Experiments
January 22, 2011
Holiday Inn San Francisco Golden Gateway
San Francisco, California USA

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 on 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

Redundant content distribution (RCD)

- Suppose a number of users located at nodes in a network demand content.

Redundant content distribution

- Suppose a number of users located at nodes in a network demand content.
- Copies of content are stored throughout the network in data warehouses.

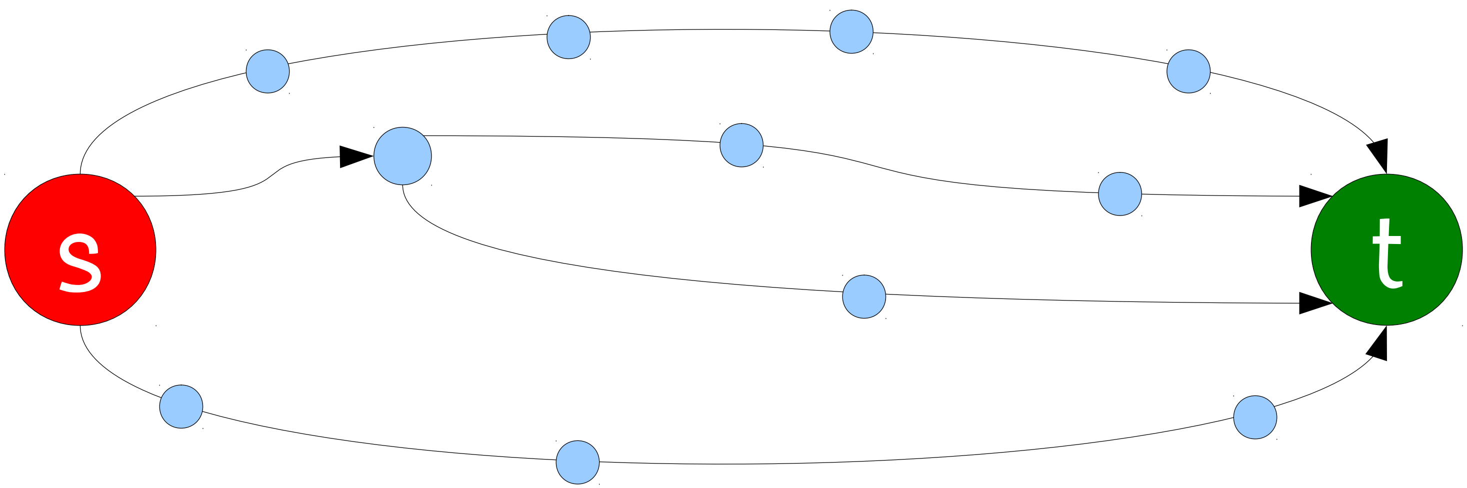
Redundant content distribution

- Suppose a number of users located at nodes in a network demand content.
- Copies of content are stored throughout the network in data warehouses.
- Content is sent from data warehouse to user on routes determined by OSPF.

Redundant content distribution

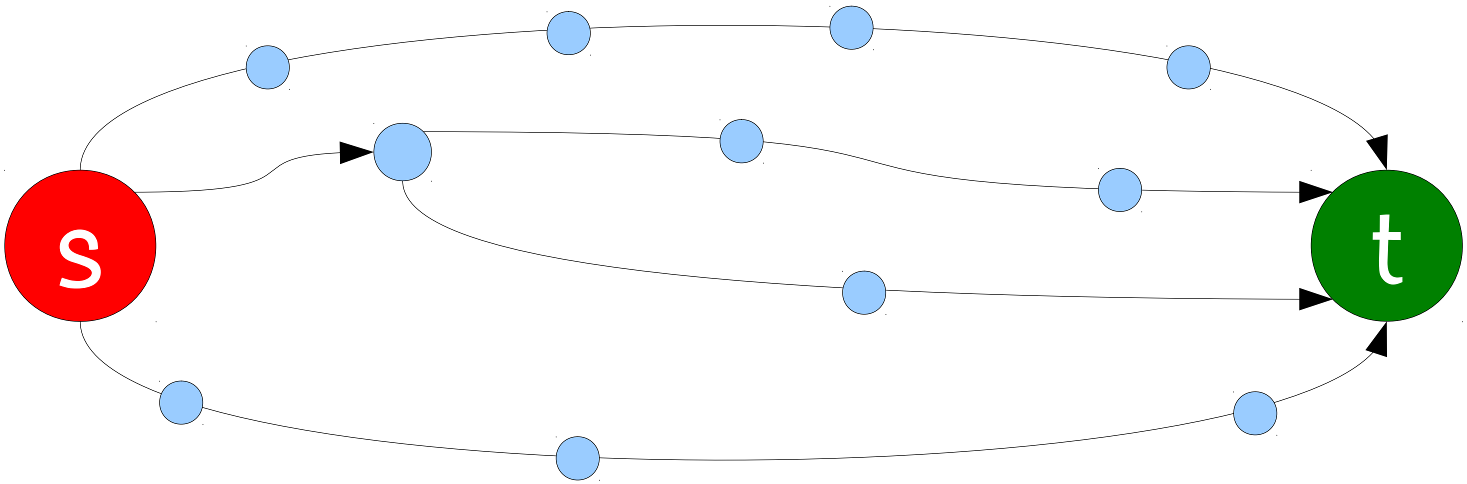
- Suppose a number of users located at nodes in a network demand content.
- Copies of content are stored throughout the network in data warehouses.
- Content is sent from data warehouse to user on routes determined by OSPF.
- Problem: Locate minimum number of warehouses in network such all users get their content even in presence of edge failures.

Redundant content distribution



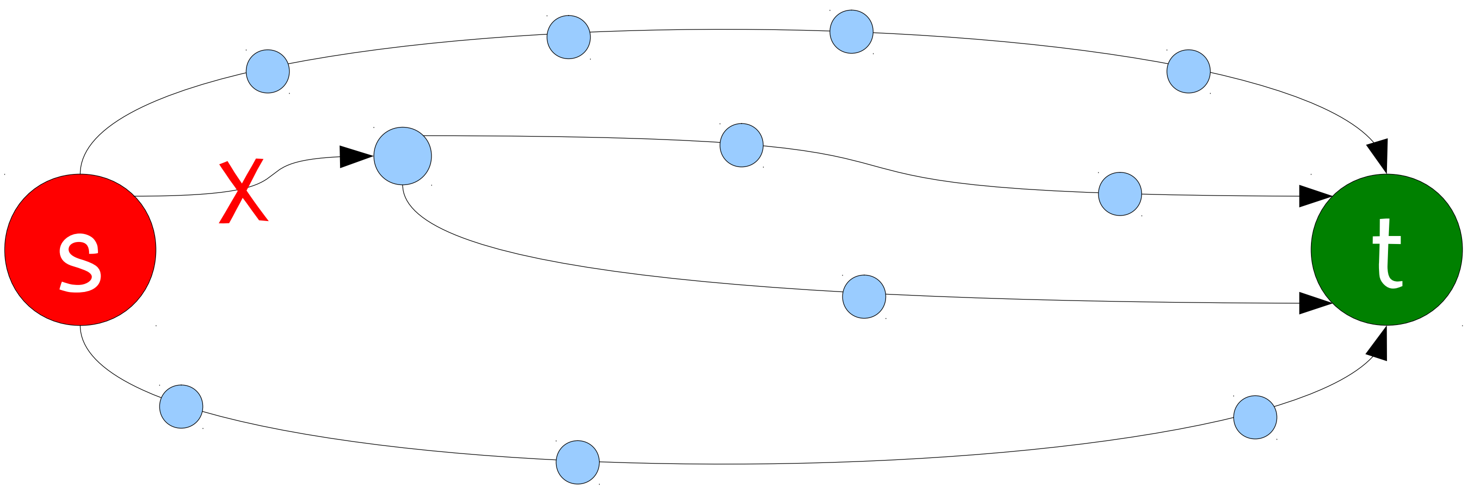
Traffic from node s to node t flows on paths defined by OSPF.

Redundant content distribution

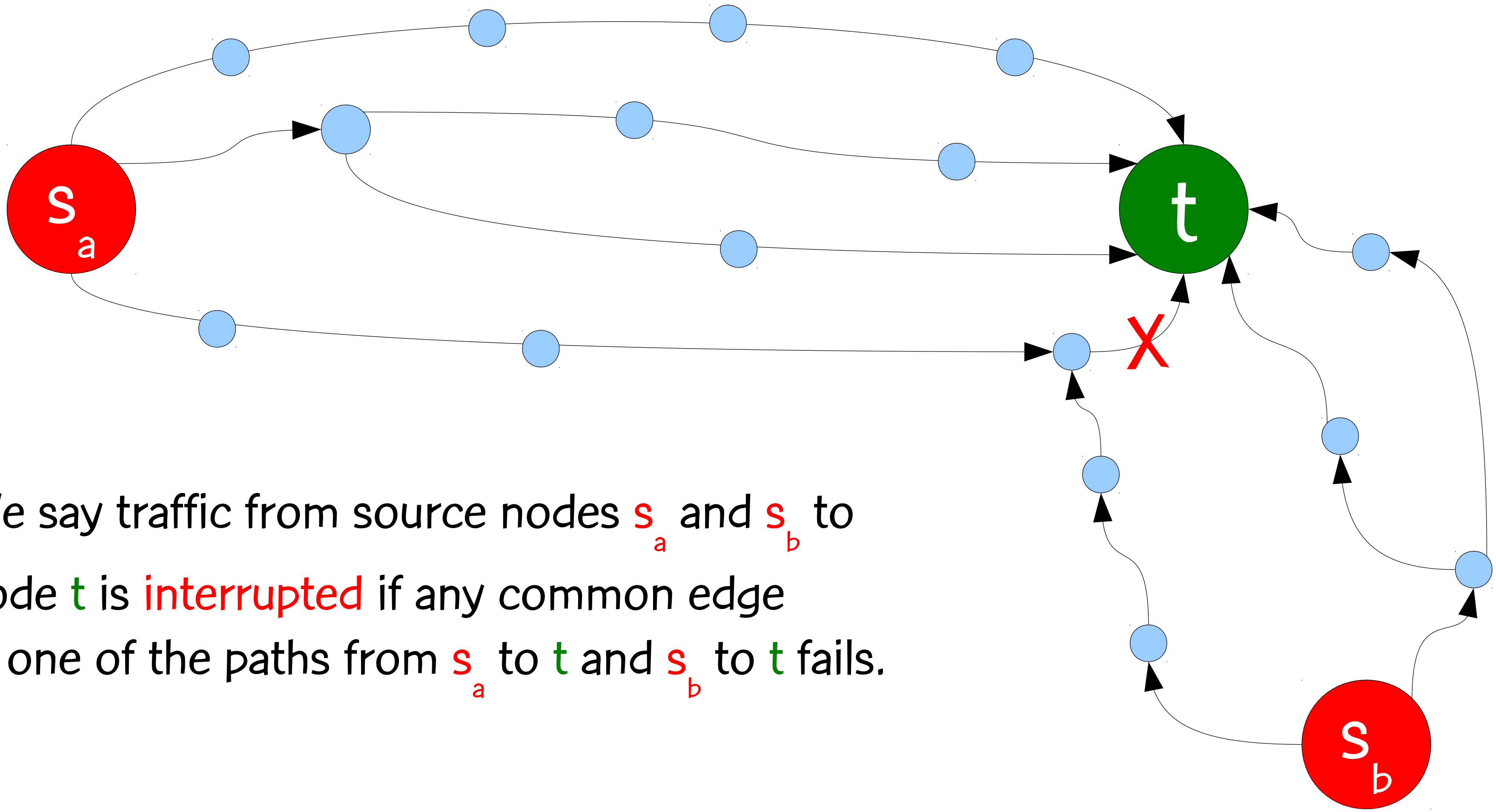


We don't know on which path a particular packet will flow.

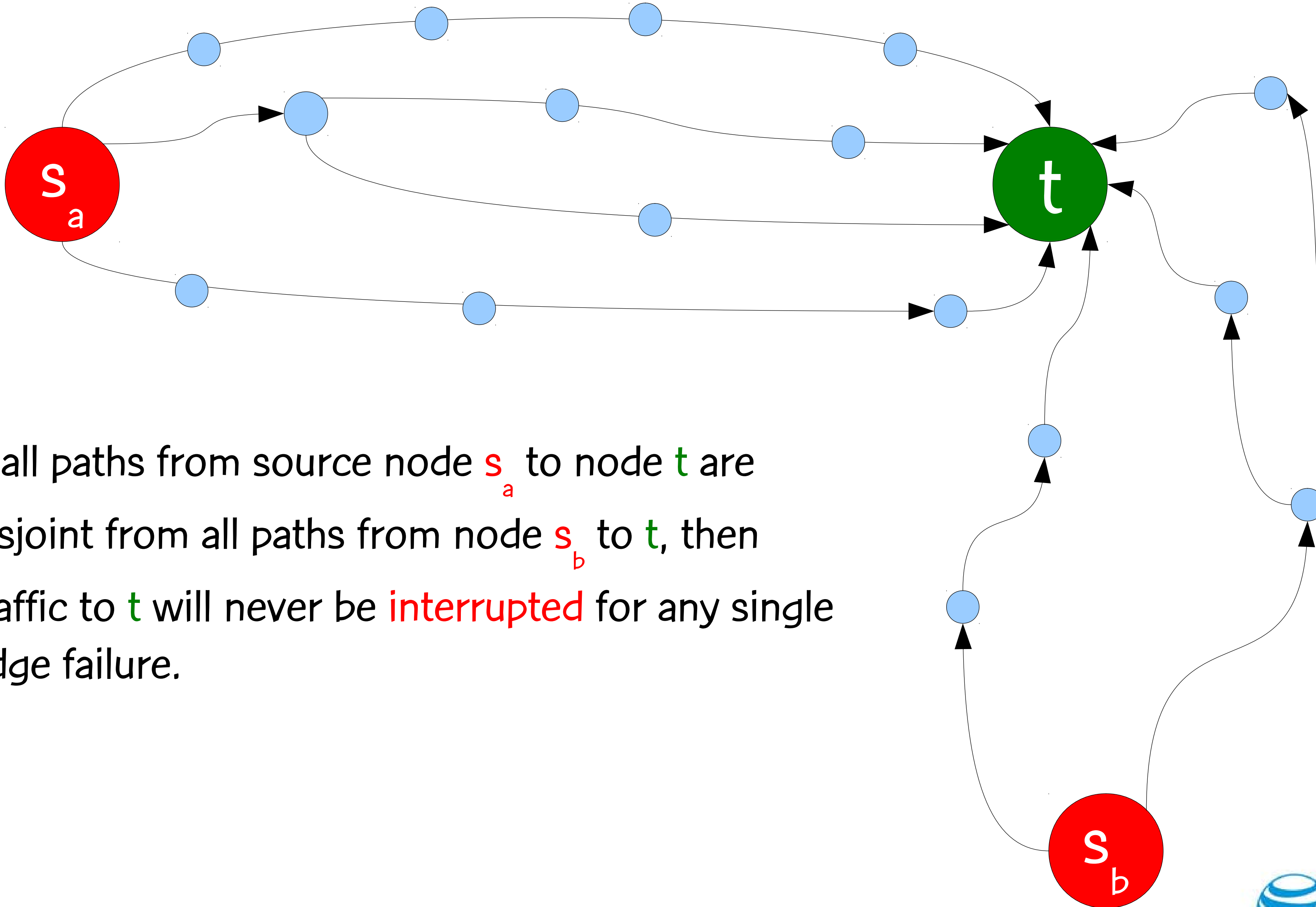
Redundant content distribution



We say traffic from node s to node t is interrupted if any edge in one of the paths from s to t fails.



We say traffic from source nodes s_a and s_b to node t is interrupted if any common edge in one of the paths from s_a to t and s_b to t fails.



If all paths from source node s_a to node t are disjoint from all paths from node s_b to t, then traffic to t will never be interrupted for any single edge failure.

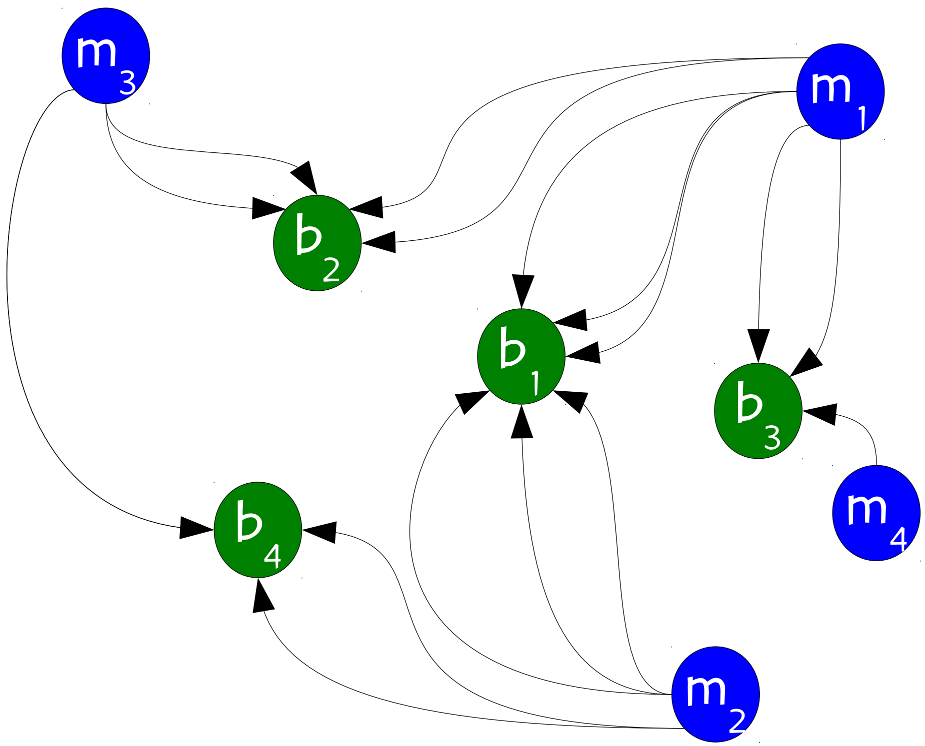
Redundant content distribution

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

We want the smallest 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 $m_2 \rightarrow b$



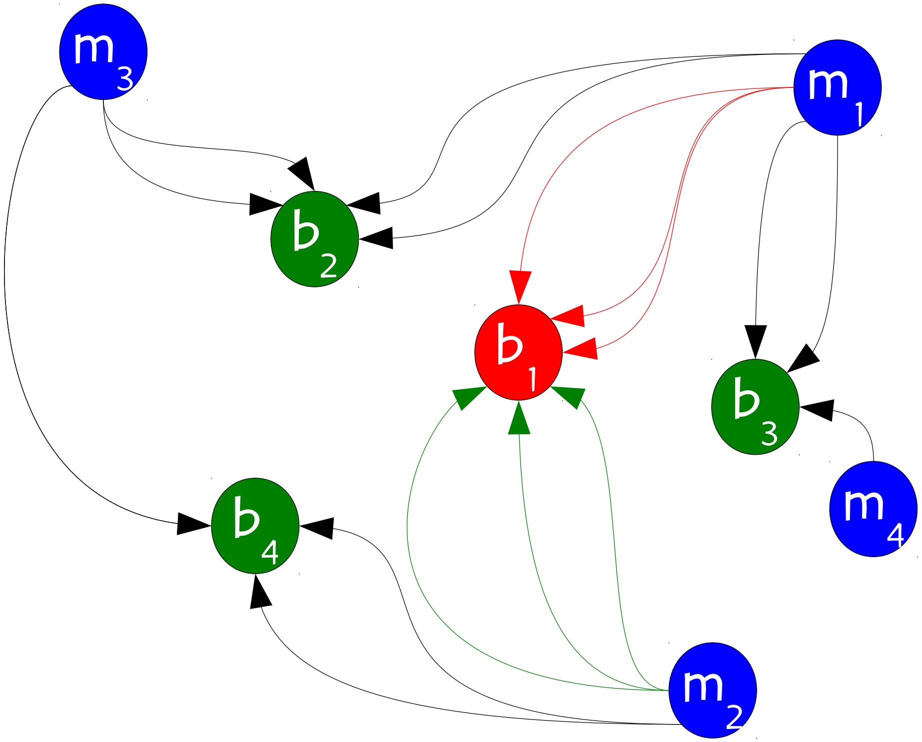
Redundant content distribution

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

We want the smallest 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 $m_2 \rightarrow b$



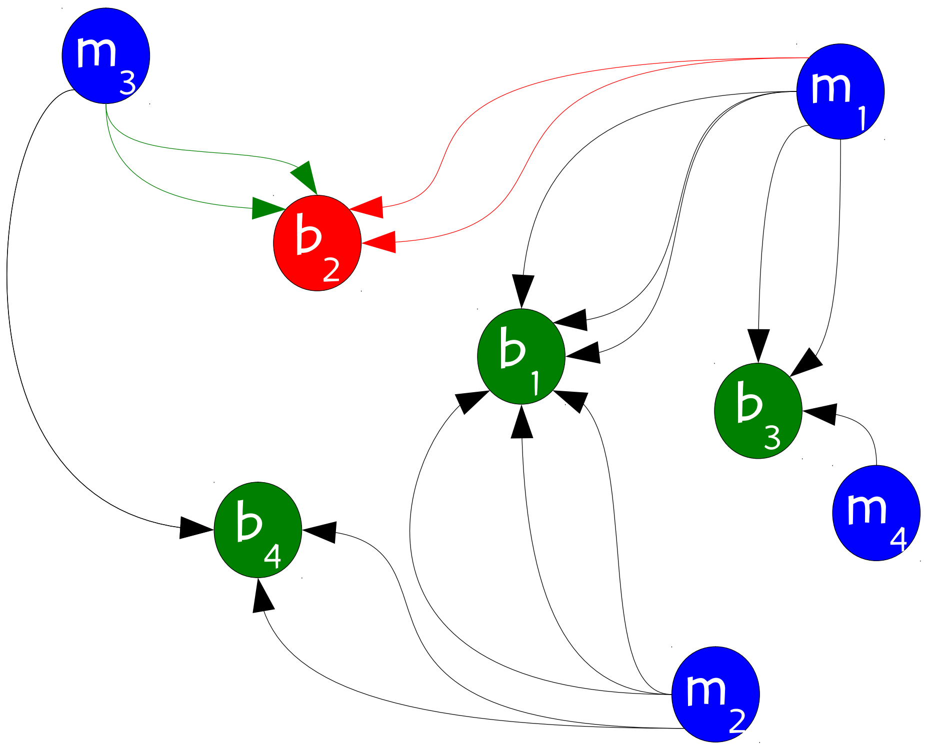
Redundant content distribution

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

We want the smallest 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 $m_2 \rightarrow b$



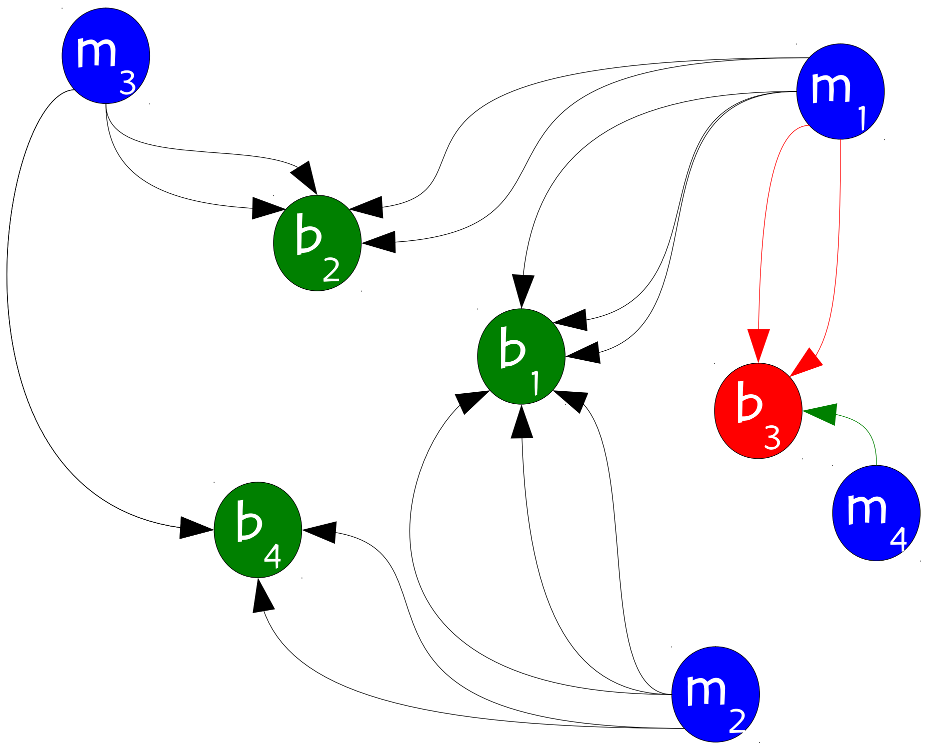
Redundant content distribution

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

We want the smallest 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 $m_2 \to b$



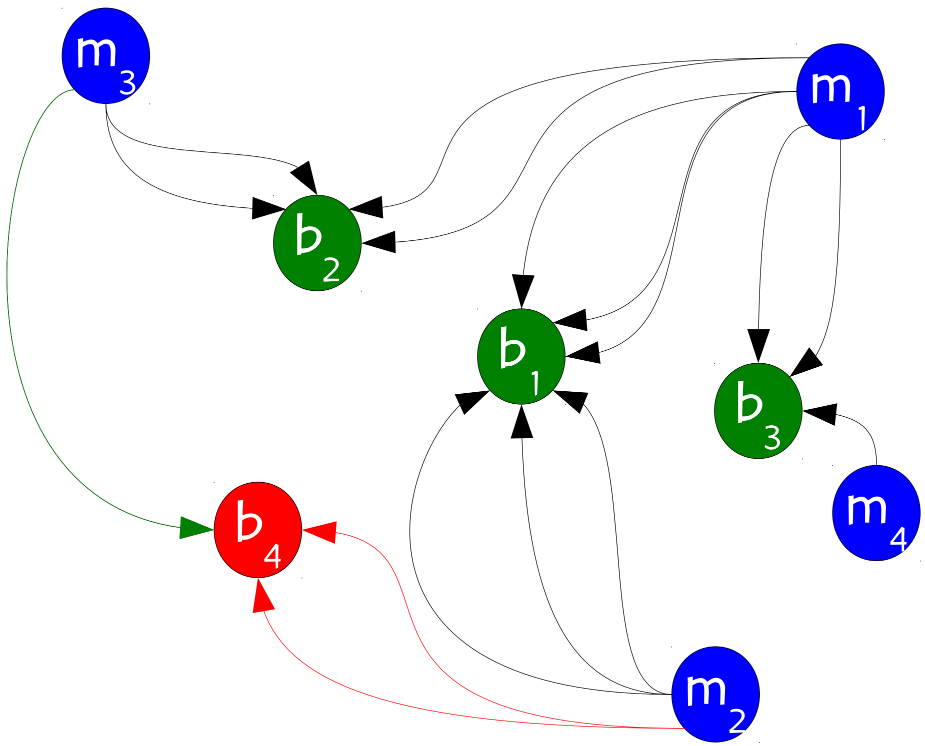
Redundant content distribution

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

We want the smallest 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 $m_2 \rightarrow b$



Redundant content distribution

- Given:
  - A directed network $G = (V, E)$;
  - A set of nodes $B \subseteq E$ where content-demanding users are located;
  - A set of nodes $M \subseteq E$ where content warehouses can be located;
  - The set of all OSPF paths from m to b, for $m \in M$ and $b \in B$.

Redundant content distribution

- Compute:
  - The set of triples $\{ m_1, m_2, b \}^i$, $i = 1, 2, ..., T$, such that all paths from m_1 to b and from m_2 to b are disjoint, where $m_1, m_2 \in M$ and $b \in B$.
  - Note that if $B \cap M \neq \emptyset$, then some triples will be of the type $\{ b, b, b \}$, where $b \in B \cap M$, i.e. a data warehouse that is co-located with a user can provide content to the user by itself.

Redundant content distribution

- Solve the covering by pairs problem:
  - Find a smallest-cardinality set $M^* \subseteq M$ such that for all $b \in B$, there exists a triple $\{ m_1, m_2, b \}$ in the set of triples such that $m_1, m_2 \in M^*$.

Greedy algorithm for covering by pairs

- initialize partial cover $M^* = \{ \}$

Greedy algorithm for covering by pairs

- initialize partial cover $M^* = \{ \}$
- while M^* is not a cover do:

Greedy algorithm for covering by pairs

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

Greedy algorithm for covering by pairs

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

Greedy algorithm for covering by pairs

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

BRKGA for redundant content distribution

BRKGA for the RCD problem

- Encoding:
  – A vector X of N keys randomly generated in the real interval $(0,1]$, where $N = |M|$ is the number of potential data warehouse nodes. The i-th random key corresponds to the i-th potential data warehouse node.

BRKGA for the RCD problem

- Encoding:
  - A vector X of N keys randomly generated in the real interval $(0,1]$, where $N = |M|$ is the number of potential data warehouse nodes. The i-th random key corresponds to the i-th potential data warehouse node.
- Decoder:

BRKGA for the RCD problem

- Encoding:
  - A vector X of N keys randomly generated in the real interval $(0,1]$, where $N = |M|$ is the number of potential data warehouse nodes. The i-th random key corresponds to the i-th potential data warehouse node.
- Decoder:
  - For $i = 1, ..., N$: if $X(i) > ½$, add i-th data warehouse node to solution

BRKGA for the RCD problem

- Encoding:
  - A vector X of N keys randomly generated in the real interval $(0,1]$, where $N = |M|$ is the number of potential data warehouse nodes. The i-th random key corresponds to the i-th potential data warehouse node.
- Decoder:
  - For $i = 1, ..., N$: if $X(i) > ½$, add i-th data warehouse node to solution
  - If solution is feasible, i.e. all users are covered: STOP

BRKGA for the RCD problem

- Encoding:
  - A vector X of N keys randomly generated in the real interval $(0,1]$, where $N = |M|$ is the number of potential data warehouse nodes. The i-th random key corresponds to the i-th potential data warehouse node.
- Decoder:
  - For $i = 1, ..., N$: if $X(i) > ½$, add i-th data warehouse node to solution
  - If solution is feasible, i.e. all users are covered: STOP
  - Else, apply greedy algorithm to cover uncovered user nodes.

BRKGA for the RCD 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

Another application: Host placement for end-to-end monitoring

- Internet service provider (ISP) delivers virtual private network (VPN) service to customers.

Another application: Host placement for end-to-end monitoring

- Internet service provider (ISP) delivers virtual private network (VPN) service to customers.
- The ISP agrees to send traffic between locations specified by the customer and promises to provide certain level of service on the connections.

Another application: Host placement for end-to-end monitoring

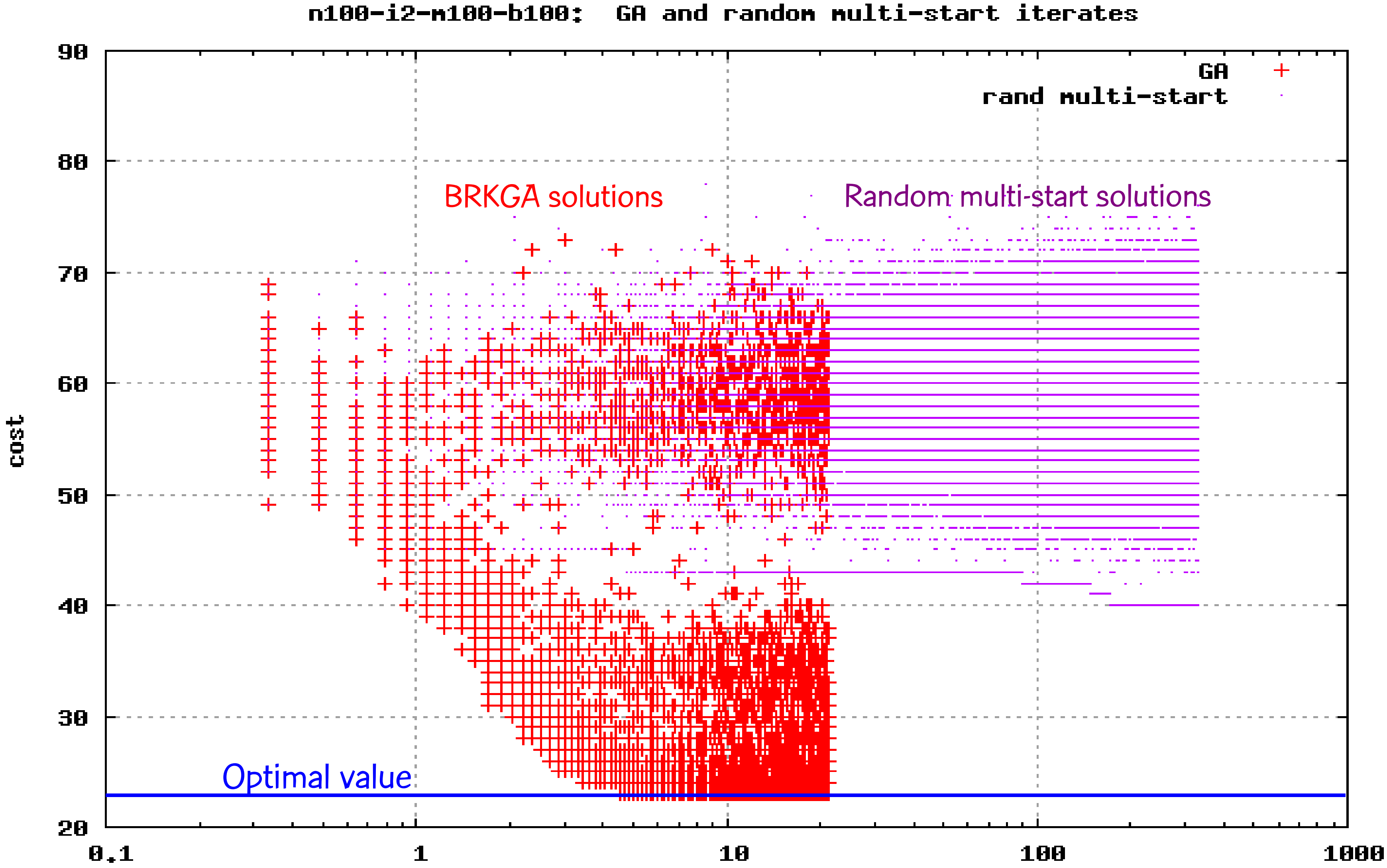
- Internet service provider (ISP) delivers virtual private network (VPN) service to customers.
- The ISP agrees to send traffic between locations specified by the customer and promises to provide certain level of service on the connections.
- A key service quality metric is packet loss rate.

Another application: Host placement for end-to-end monitoring

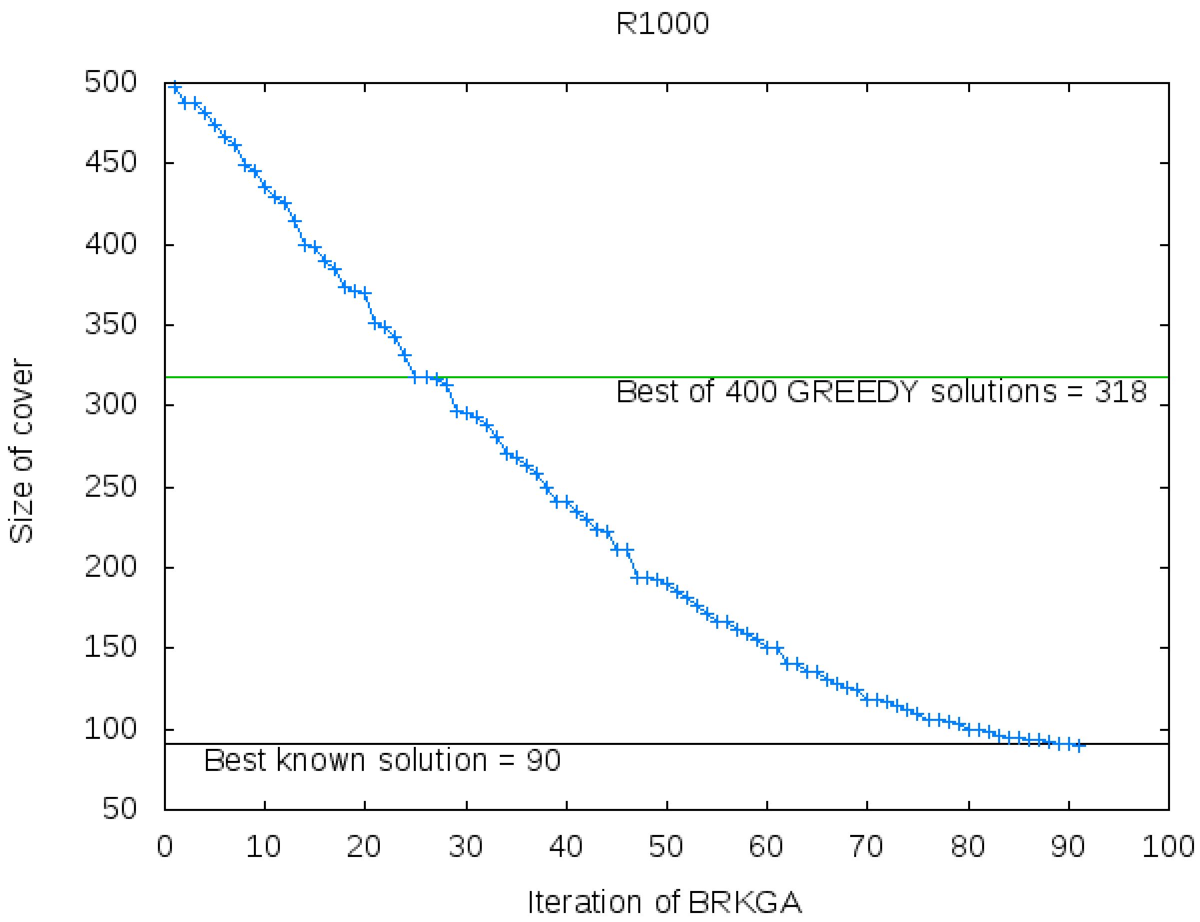
- Internet service provider (ISP) delivers virtual private network (VPN) service to customers.
- The ISP agrees to send traffic between locations specified by the customer and promises to provide certain level of service on the connections.
- A key service quality metric is packet loss rate.
- We want to minimize the number of monitoring equipment placed in the network to measure packet loss rate: This is a type of covering by pairs problem.

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

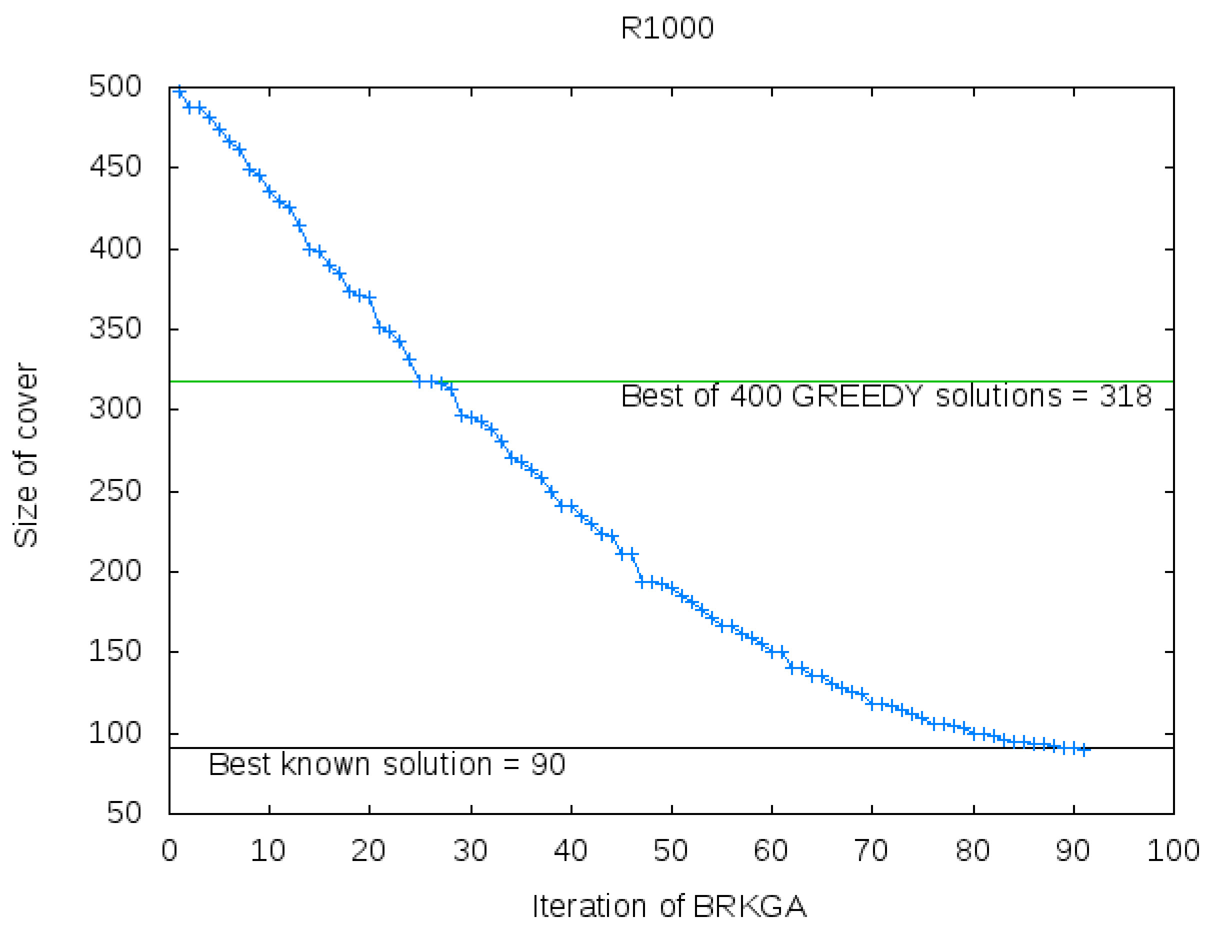
solution



Time (ibm t41 secs)



Real-world instance derived from a proprietary Tier-1 Internet Service Provider (ISP) backbone network using OSPF for routing.



Size of network: about 1000 nodes, where almost all can store content and about 90% have content-demanding users. Over 45 million triples.

Regenerator location problem

Reference

A. Duarte, R. Martí, M.G.C.R., and R.M.A. Silva, "Randomized heuristics for the regenerator location problem," AT&T Labs Research Technical Report, Florham Park, NJ, July 13, 2010.

Tech report version:

http://www.research.att.com/~mgcr/doc/gpr-regenloc.pdf

Signal regeneration

- Telecommunication systems use optical signals to transmit information

Signal regeneration

- Telecommunication systems use optical signals to transmit information
- Strength of signal deteriorates and loses power as it gets farther from source

Signal regeneration

- Telecommunication systems use optical signals to transmit information
- Strength of signal deteriorates and loses power as it gets farther from source
- Signal must be regenerated periodically to reach destination: Regenerators

Signal regeneration

- Telecommunication systems use optical signals to transmit information
- Strength of signal deteriorates and loses power as it gets farther from source
- Signal must be regenerated periodically to reach destination: Regenerators
- Regenerators are expensive: minimize the number of regenerators in the network

Regenerator location problem (RLP)

- Given:
  - Graph $G=(V,E)$, where V are vertices, E are edges, where edge (i,j) has a real-valued length $d(i,j) > 0$
  - D is the maximum length that a signal can travel before it must regenerated

Regenerator location problem (RLP)

- Find:
  - Paths that connect all pairs of nodes in $V \times V$
  - Nodes where it is necessary to locate single regenerators
- Minimize number of deployed regenerators

Regenerator location problem (RLP)

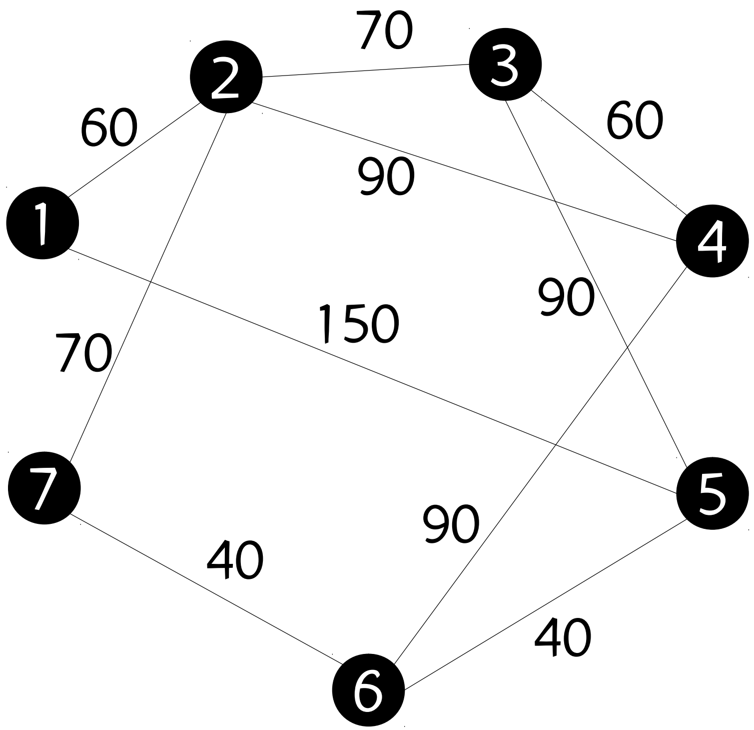
- Path between $\{s,t\} \in E$
  - $\{ (s,v[1]), (v[1],v[2]), \ldots,(v[k],t) \}$ is formed by one or more path segments
- Path segment is sequence of consecutive edges
  - $\{ (v[i],v[i+1]), (v[i+1],v[i+2]), \ldots,(v[q-1],v[q]) \}$ in the path satisfying the condition

$$d(v[i],v[i+1]) + d(v[i+1],v[i+2]) + \cdots + (v[q-1],v[q]) \leq D$$

Regenerator location problem (RLP)

- If total length of path is no more than D, then path consists of a single path segment
- Otherwise, it consists of one or more segments
  - Regenerators will be located in the internal nodes of the path

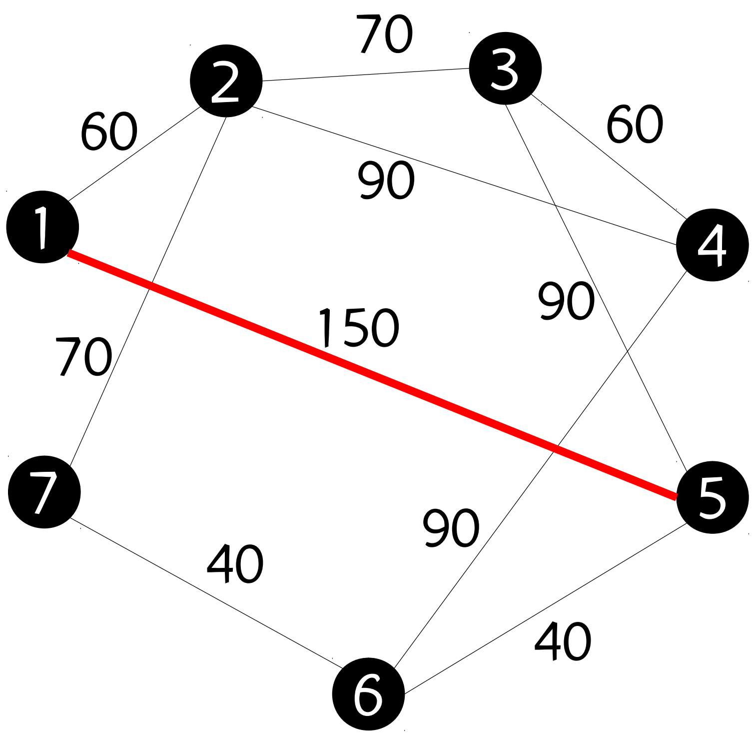
Regenerator location problem (RLP)



D = 100

7-node graph with D = 100

Regenerator location problem (RLP)

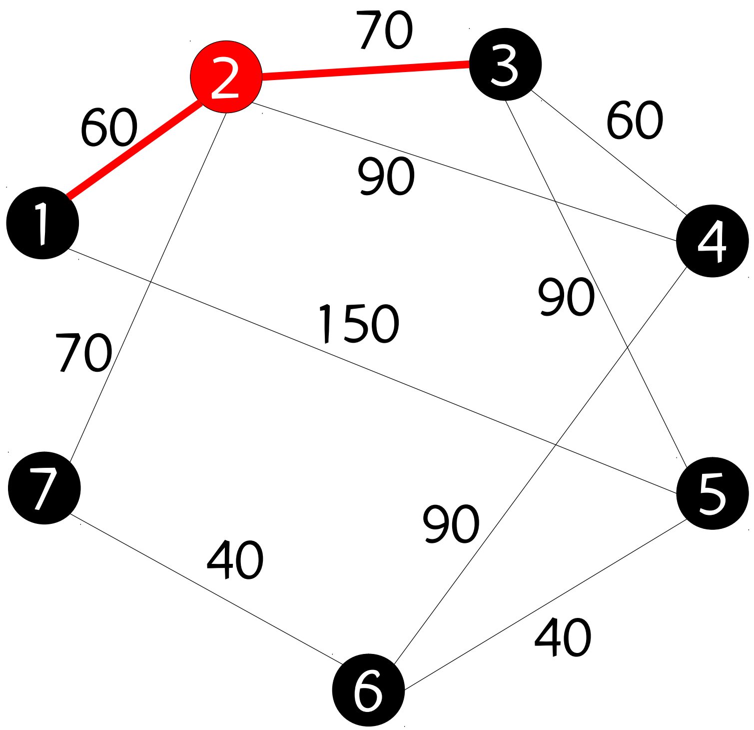


D = 100

(1) Note that:

- D(1,5) = 150 > 100 = D
- Edge (1,5) cannot be part of any path

Regenerator location problem (RLP)



D = 100

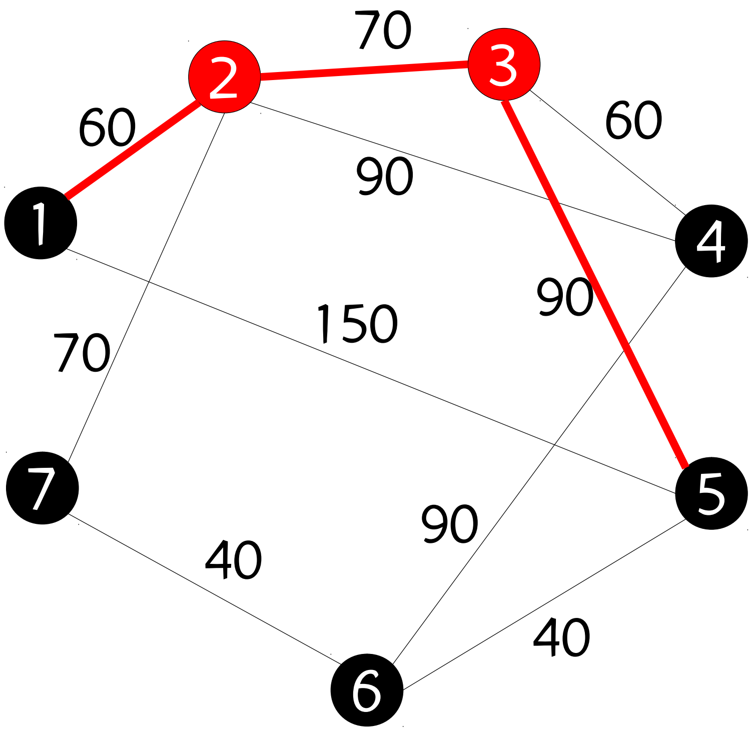
(2) Note that:

- Shortest path from 1 to 3 is { (1,2), (2,3) } with total length

  60 + 70 = 130 > 100 = D

- Must be decomposed into two path segments { (1,2) } and { (2,3) } with a regenerator in node 2

Regenerator location problem (RLP)

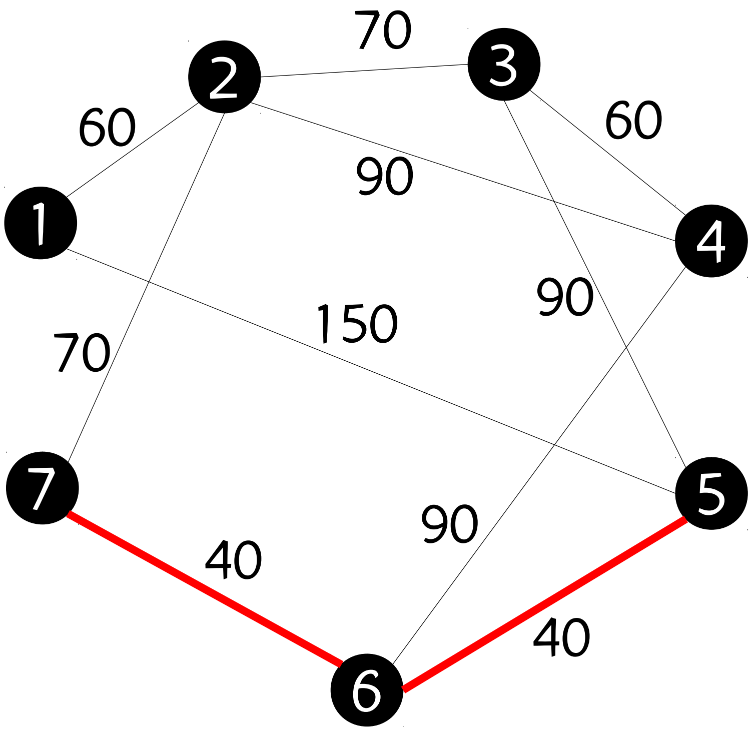


D = 100

(3) Note that:

- Shortest feasible path from 1 to 5 is { (1,2), (2,3), (3,5) } with total length 60 + 70 + 90 = 220 > 100 = D
- Must be decomposed into three path segments { (1,2) }, { (2,3) }, and { (3,5) } with regenerators in nodes 2 and 3

Regenerator location problem (RLP)

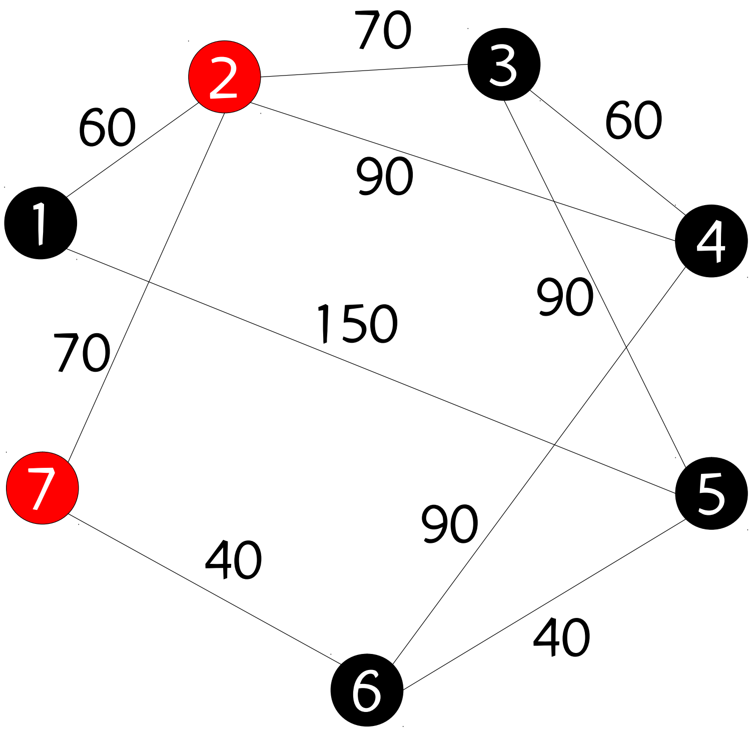


D = 100

(4) Note that:

- Shortest feasible path from 5 to 7 is { (5,6), (6,7) } with total length 40 + 40 = 80 ≤ 100 = D
- No regenerator is needed to connect nodes 5 and 7

Regenerator location problem (RLP)

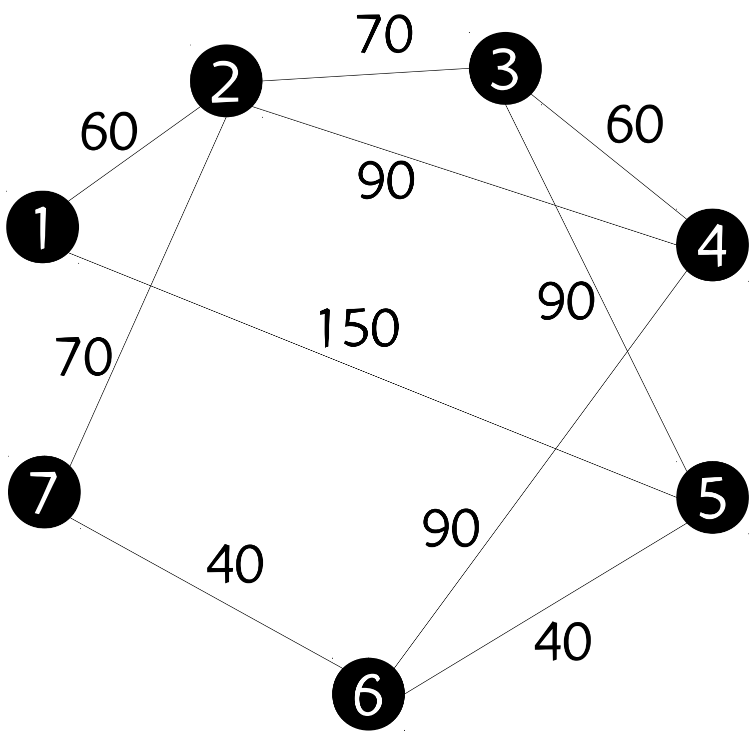


D = 100

(5) Note that:

- Placing regenerator in nodes 2 and 7 allows for communication between all pairs of nodes in the graph

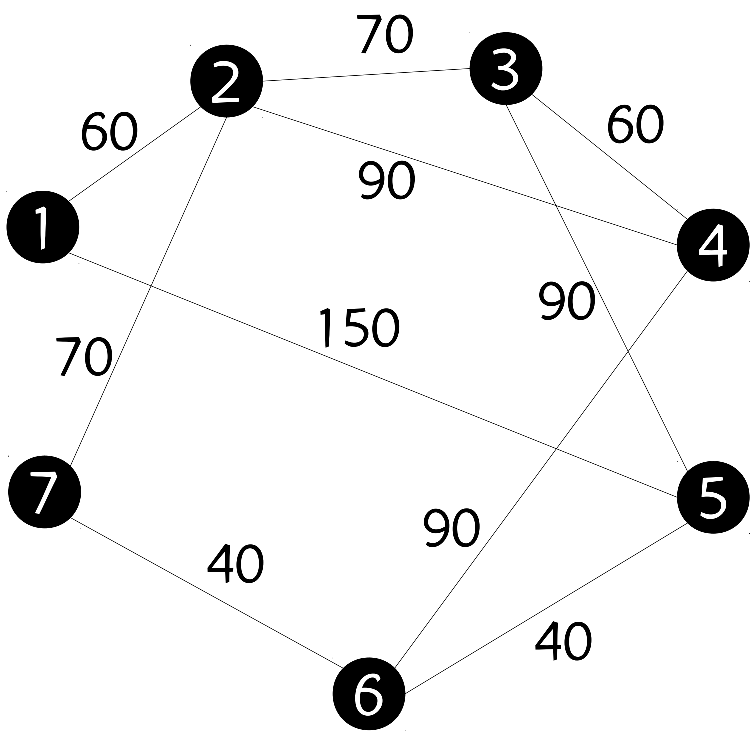
Communication graph (Chen et al., 2010)



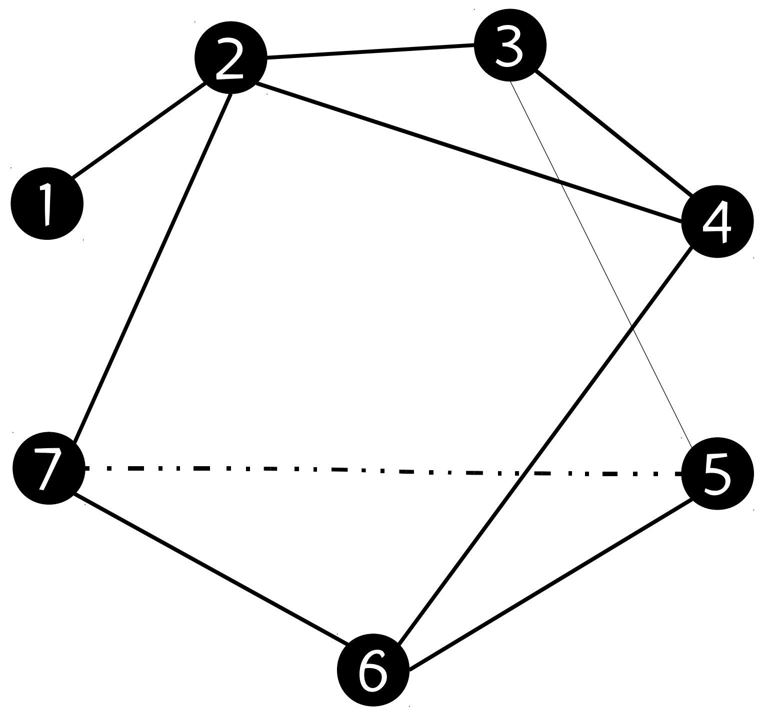
G = (V,E)

- Given weighted graph G
  - Delete all edges having length greater than D
  - For all non-adjacent nodes, add an edge between them of length equal to the corresponding shortest path in G if it is less than D
  - Disregard all length info

Communication graph (Chen et al., 2010)

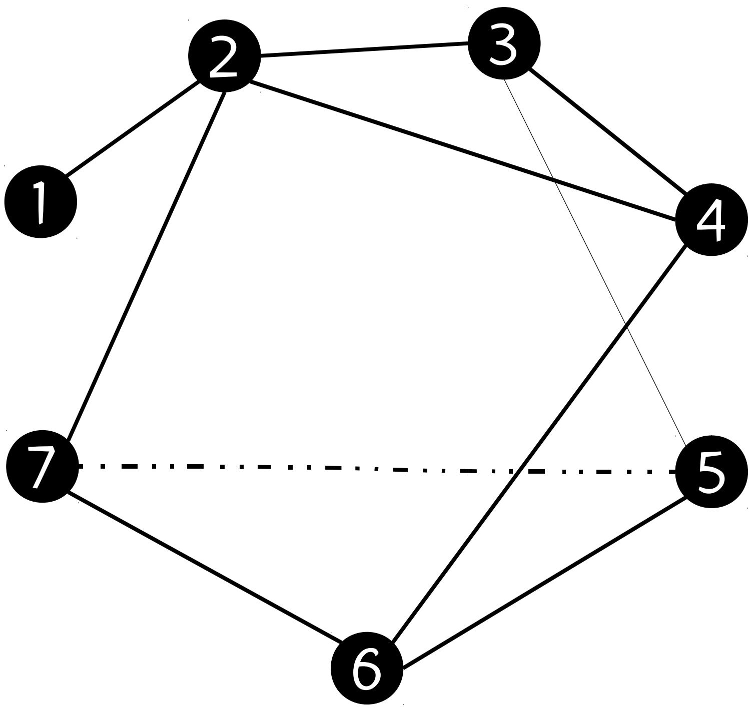


G = (V,E)



M = (V,E')

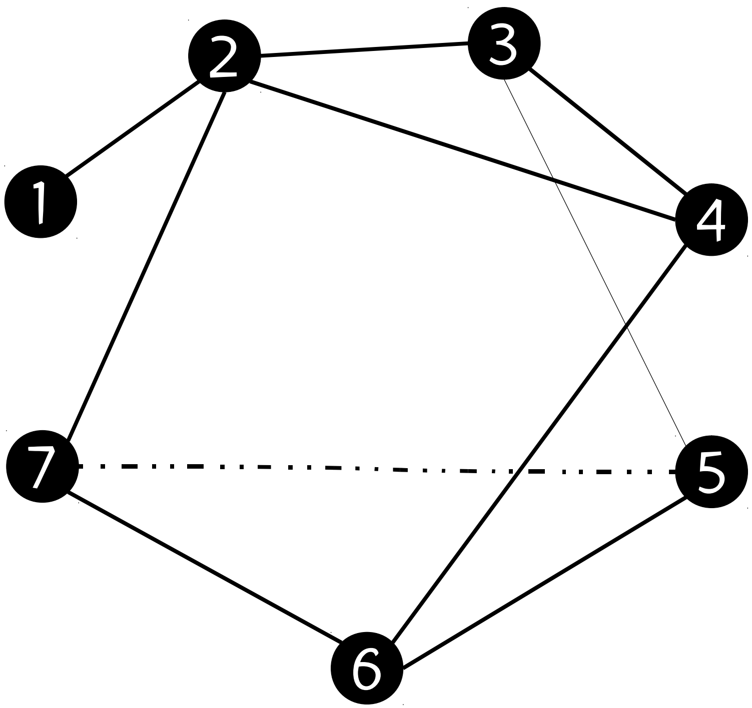
Communication graph (Chen et al., 2010)



M = (V,E')

- If M is complete, then there is no need for regenerators

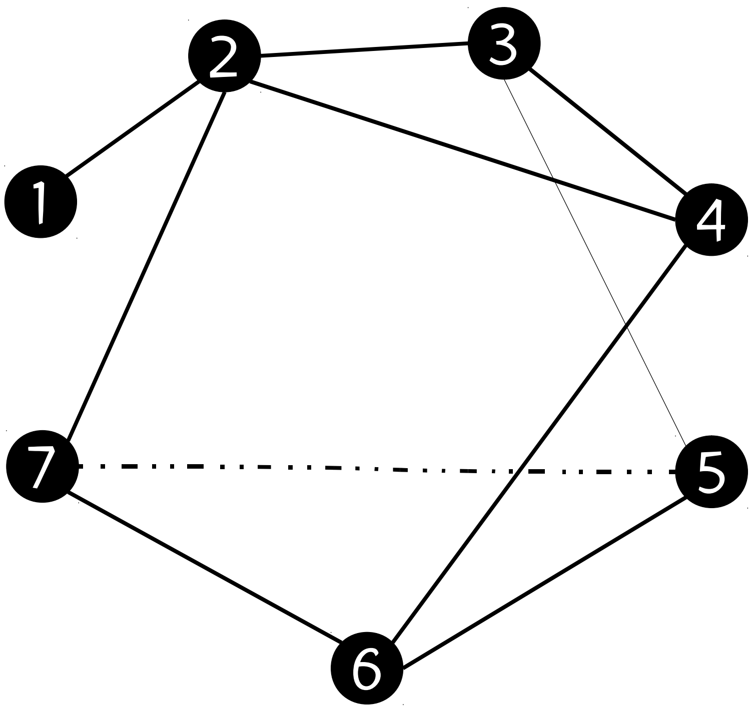
Communication graph (Chen et al., 2010)



M = (V,E')

- If M is complete, then there is no need for regenerators
- If M is not connected, then the problem is infeasible

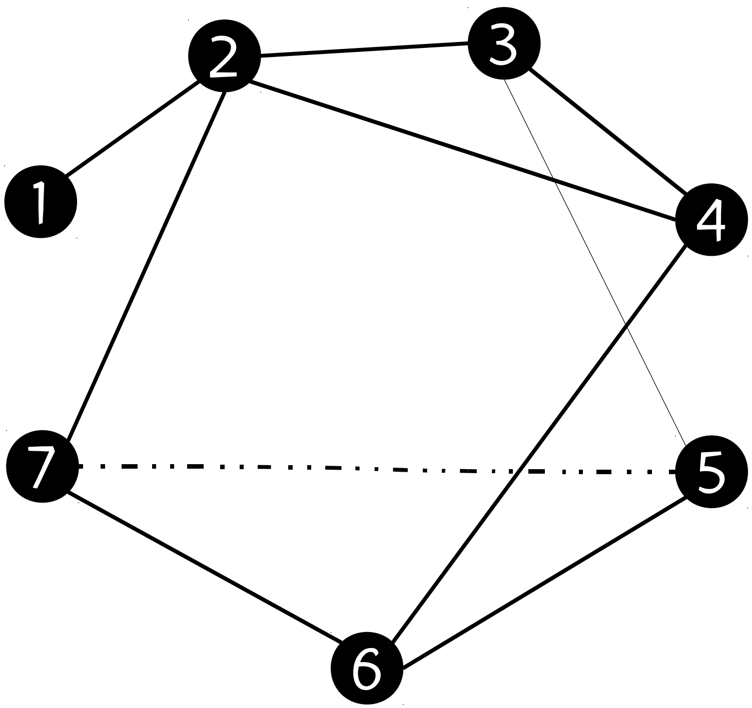
Communication graph (Chen et al., 2010)



M = (V,E')

- If M is complete, then there is no need for regenerators
- If M is not connected, then the problem is infeasible
- Otherwise, one or more regenerators are needed

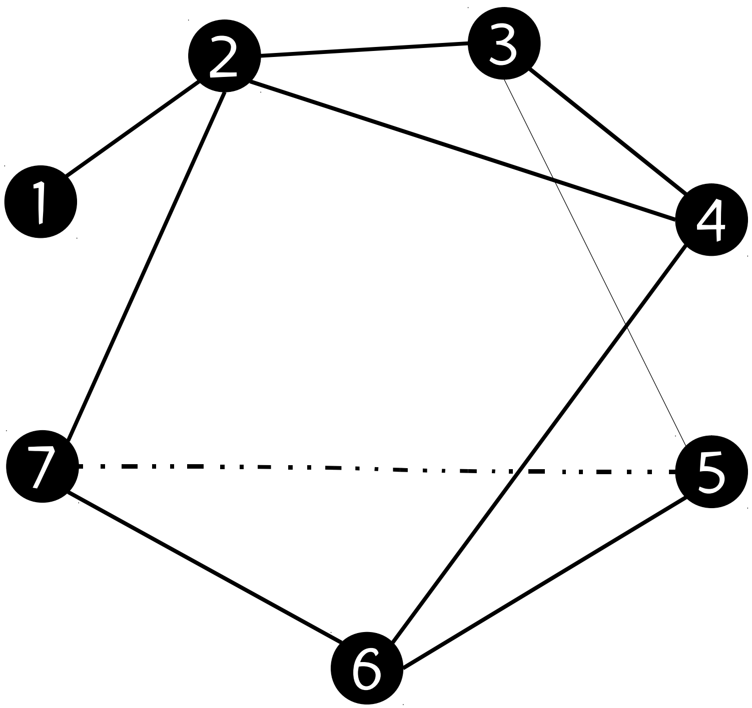
Greedy algorithm (Chen et al., 2010)



M = (V,E')

- Works on communication graph M

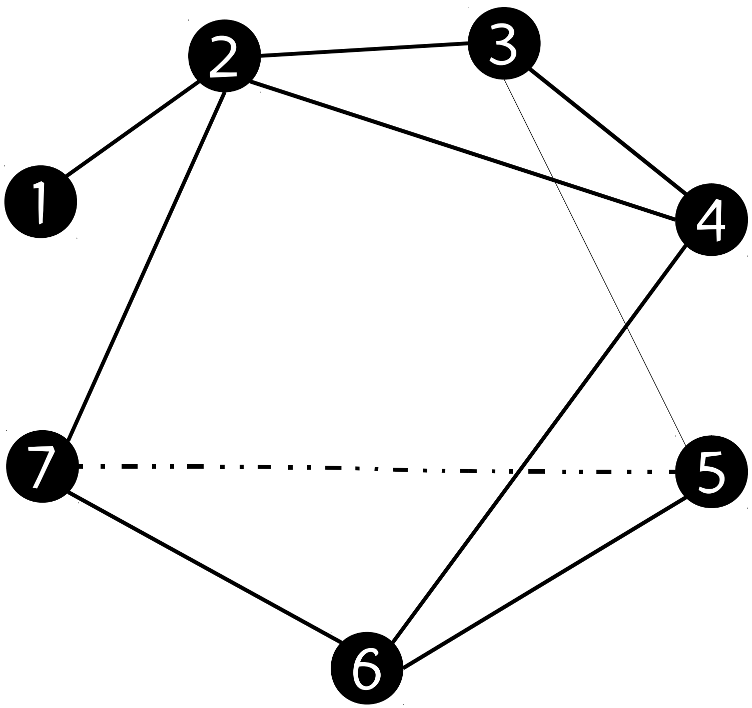
Greedy algorithm (Chen et al., 2010)



M = (V,E')

- Works on communication graph M
- Input: set of nodes not directly connected (NDC) in M and builds a set R of regenerator nodes

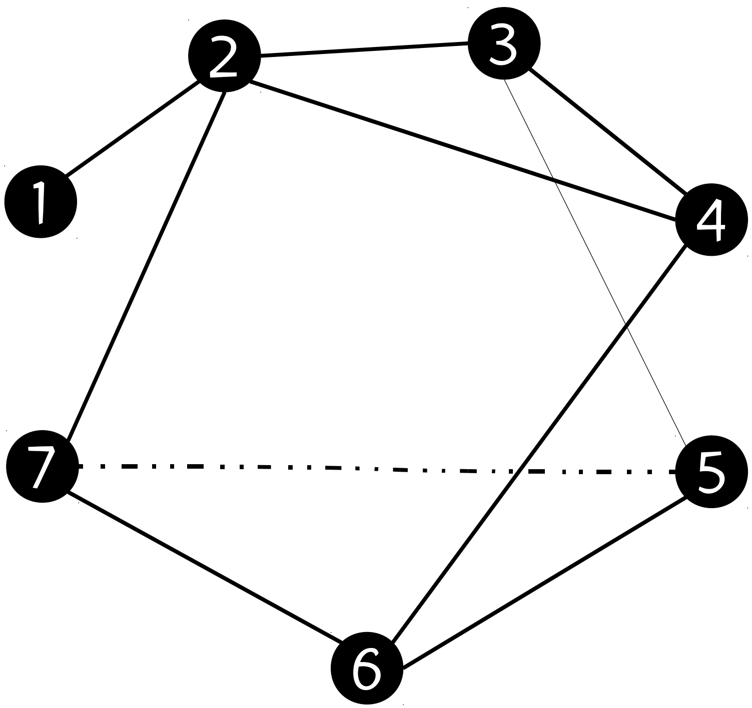
Greedy algorithm (Chen et al., 2010)



M = (V,E')

- Works on communication graph M
- Input: set of nodes not directly connected (NDC) in M and builds a set R of regenerator nodes
- At each step the procedure determines a node u* whose inclusion in R enables the connection of the largest number g(u*) of yet unconnected pairs X(u*) in M

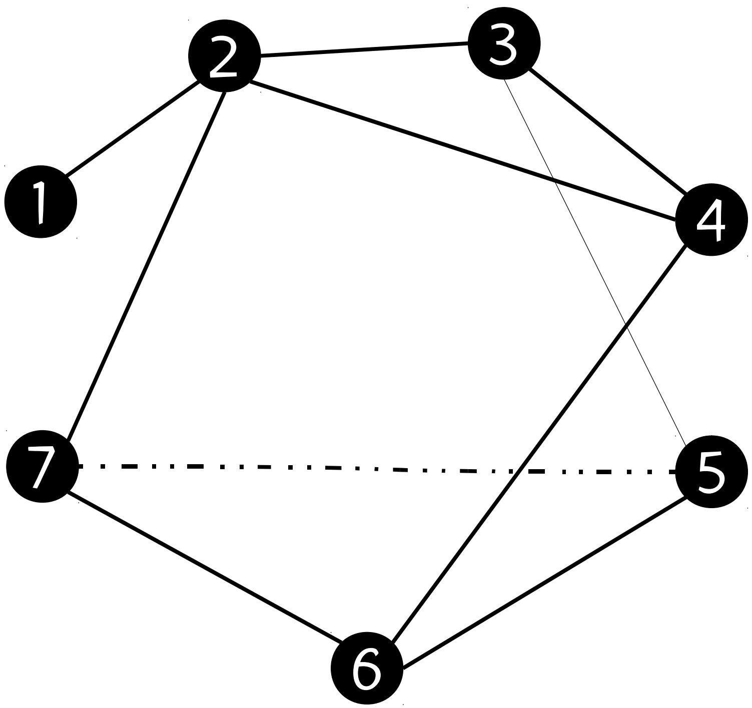
Greedy algorithm (Chen et al., 2010)



M = (V,E')

- Works on communication graph M
- Input: set of nodes not directly connected (NDC) in M and builds a set R of regenerator nodes
- At each step the procedure determines a node u* whose inclusion in R enables the connection of the largest number g(u*) of yet unconnected pairs X(u*) in M
- Node u* is added to R and M is updated

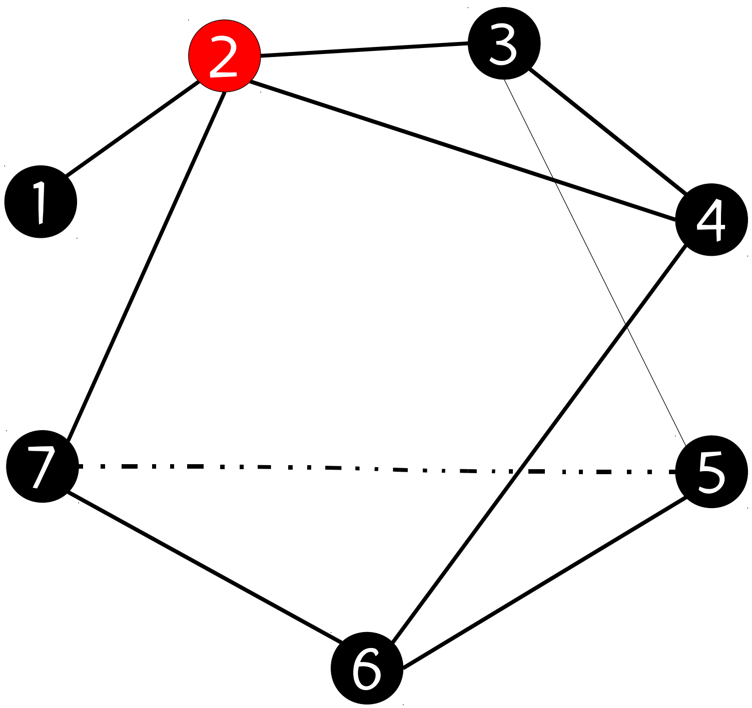
Greedy algorithm (Chen et al., 2010)



M = (V,E')

u	X(u)	g(u)
1	∅	0
2	{ (1,3),(1,4),(1,7),(3,7), (4,7) }	5
3	{ (4,5),(2,5) }	2
4	{ (2,6),(3,6) }	2
5	{ (3,7),(3,6) }	2
6	{ (4,7),(4,5) }	2
7	{ (2,6),(2,5) }	2

Greedy algorithm (Chen et al., 2010)

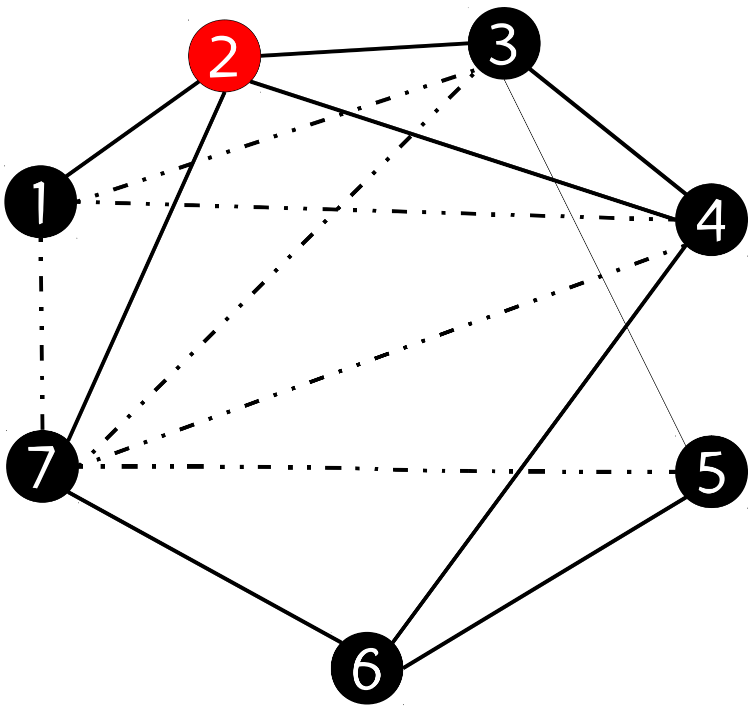


M = (V,E')

Add regenerator to node 2

u	X(u)	g(u)
1	∅	0
2	**{ (1,3),(1,4),(1,7),(3,7), (4.7) }**	**5**
3	{ (4,5),(2,5) }	2
4	{ (2,6),(3,6) }	2
5	{ (3,7),(3,6) }	2
6	{ (4,7),(4,5) }	2
7	{ (2,6),(2,5) }	2

Greedy algorithm (Chen et al., 2010)

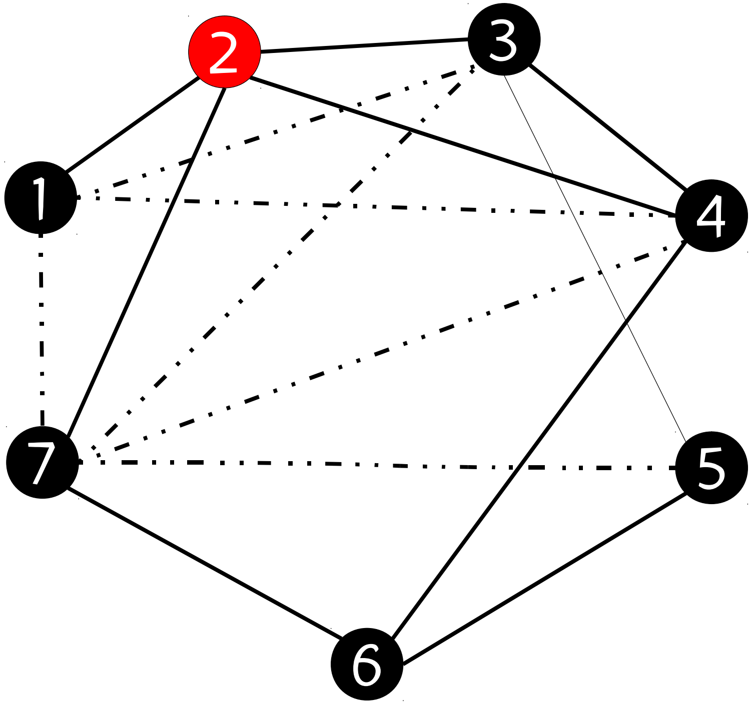


M = (V,E')

u	X(u)	g(u)
1	∅	0
2	**{ (1,3),(1,4),(1,7),(3,7), (4.7) }**	**5**
3	{ (4,5),(2,5) }	2
4	{ (2,6),(3,6) }	2
5	{ (3,7),(3,6) }	2
6	{ (4,7),(4,5) }	2
7	{ (2,6),(2,5) }	2

Update M to account for regenerator in node 2

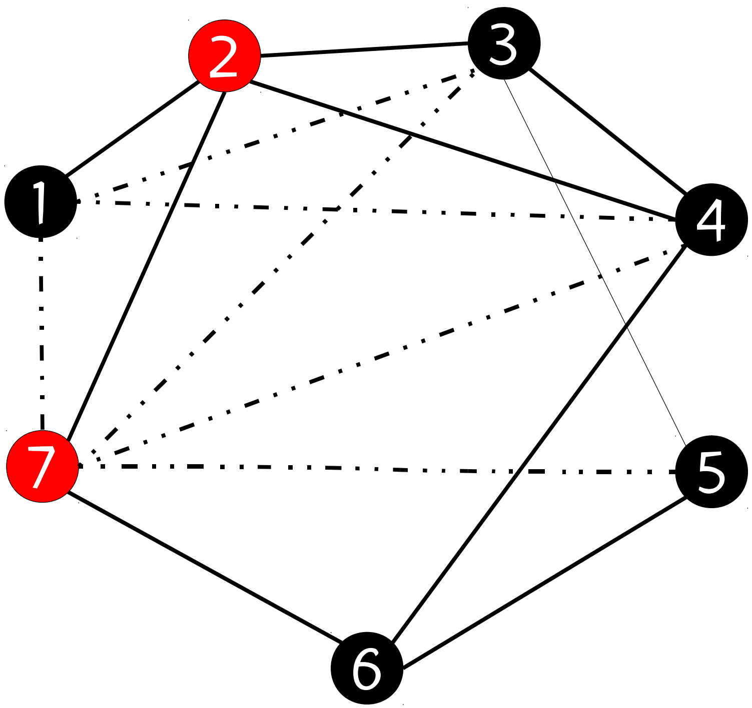
Greedy algorithm (Chen et al., 2010)



M = (V,E')

u	X(u)	g(u)
1	∅	0
-	-	-
3	{ (1,5),(2,5),(4,5) }	3
4	{ (1,6),(2,6),(3,6) }	3
5	{ (3,6) }	1
6	{ (4,5) }	1
7	{ (1,5),(1,6),(2,5),(2,6),(3,6),(4,5) }	6

Greedy algorithm (Chen et al., 2010)

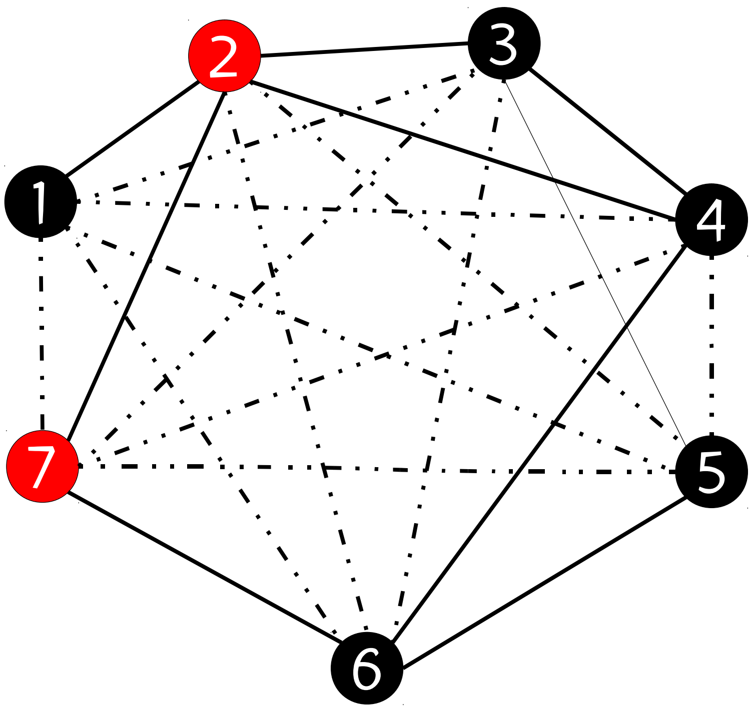


M = (V,E')

u	X(u)	g(u)
1	∅	0
-	-	-
3	{ (1,5),(2,5),(4,5) }	3
4	{ (1,6),(2,6),(3,6) }	3
5	{ (3,6) }	1
6	{ (4,5) }	1
7	**{ (1,5),(1,6),(2,5),(2,6),(3,6),(4,5) }**	**6**

Add regenerator to node 7

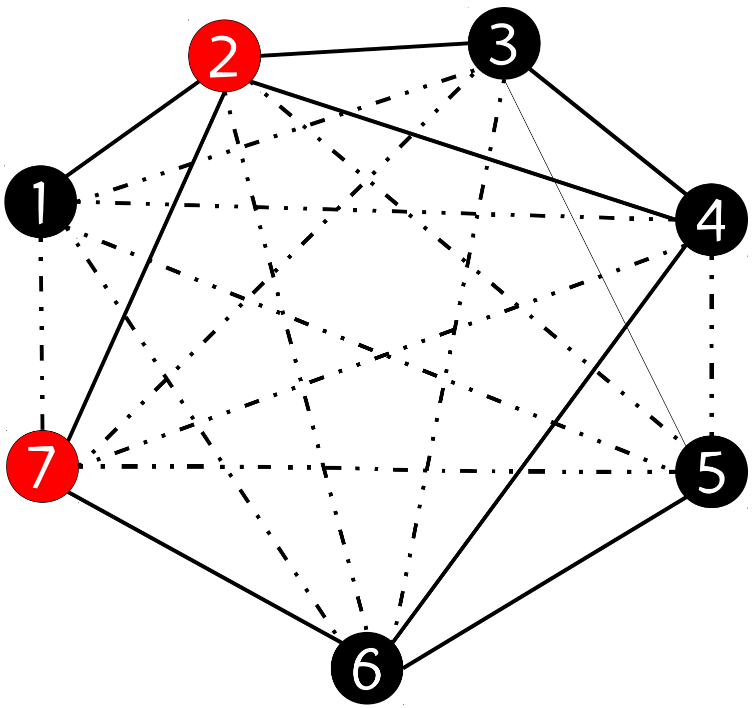
Greedy algorithm (Chen et al., 2010)



M = (V,E')

u	X(u)	g(u)
1	∅	0
-	-	-
3	{ (1,5),(2,5),(4,5) }	3
4	{ (1,6),(2,6),(3,6) }	3
5	{ (3,6) }	1
6	{ (4,5) }	1
7	**{ (1,5),(1,6),(2,5),(2,6),(3,6),(4,5) }**	**6**

Greedy algorithm (Chen et al., 2010)



M = (V,E')

Since M is complete, all pairs can communicate and solution R = {2,7}

BRKGA for the regenerator location problem

Encoding

Solutions are encoded as vectors Y of $n = |V|$ random keys, each in the real interval $[0,1)$

Random key $Y[i]$ corresponds to node $i \in V$

Decoding

Takes as input a communication graph $M = (V,E')$ and a vector of random keys Y

Outputs a set of regenerator nodes $R \subseteq V$

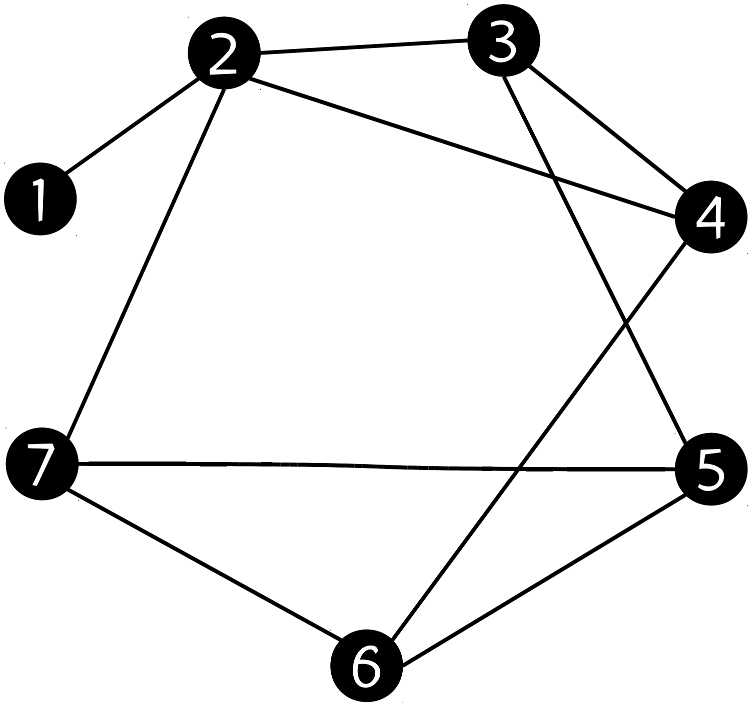
Decoding

Takes as input a communication graph $M = (V,E')$ and a vector of random keys Y

Outputs a set of regenerator nodes $R \subseteq V$

Sorting Y implies an ordering of V

Decoding



M = (V,E')

Scan V in order implied by Y

while come pair in $V \times V$ cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

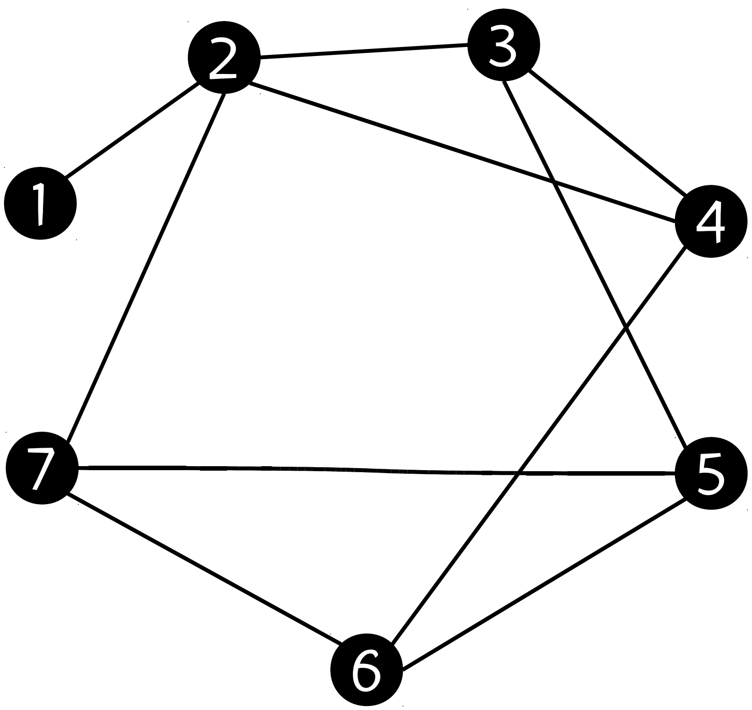
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



$X = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)$

Scan V in order implied by Y

while come pair in V × V cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

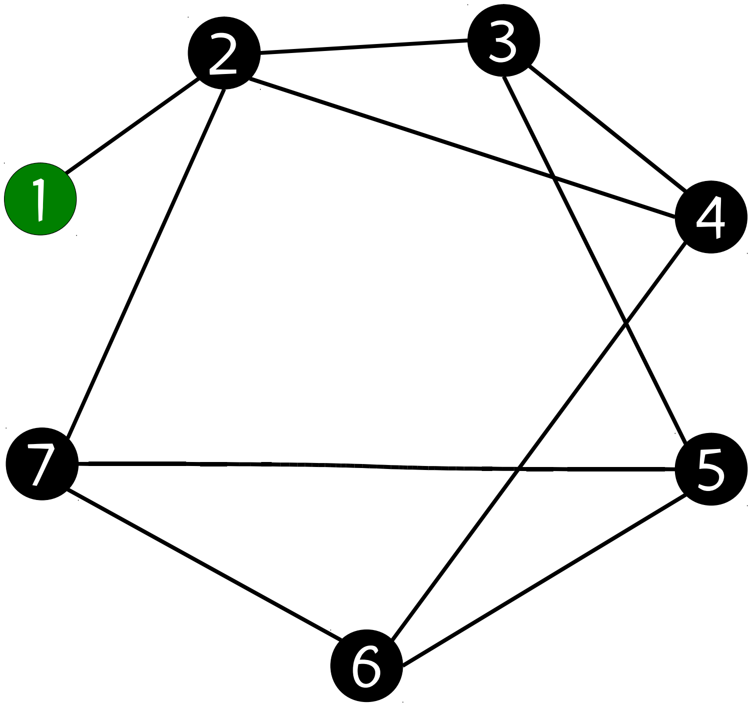
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



$X = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)$

⬆

$i = 1$

Scan V in order implied by Y

while come pair in $V \times V$ cannot communicate ($M = (V, E')$ is not complete):

Add next vertex v in order into R

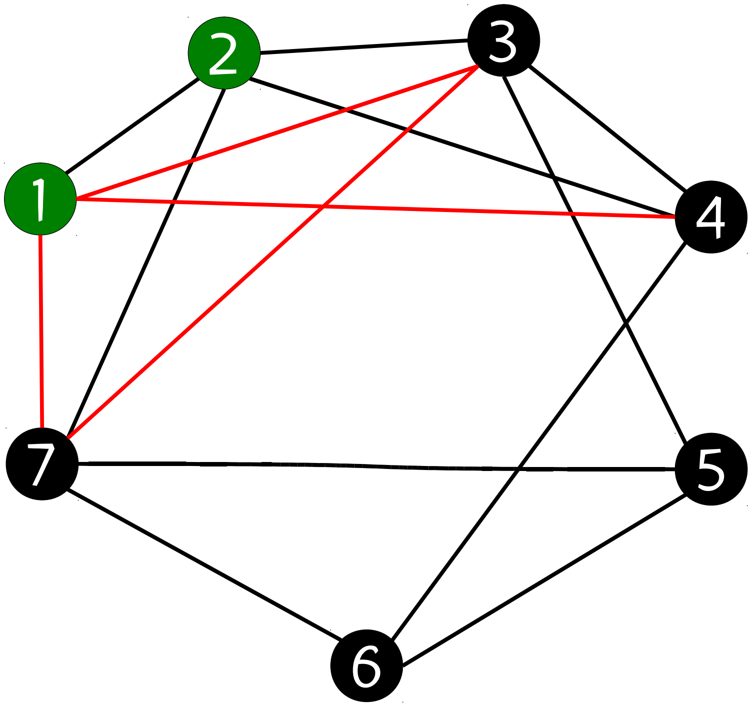
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



X = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)

i = 2

Scan V in order implied by Y

while come pair in V × V cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

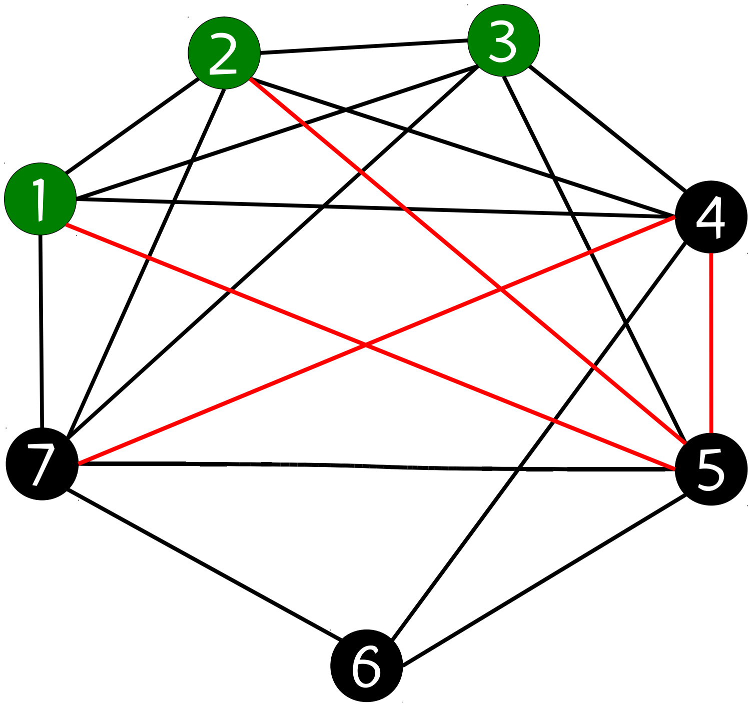
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



Scan V in order implied by Y

while come pair in V × V cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

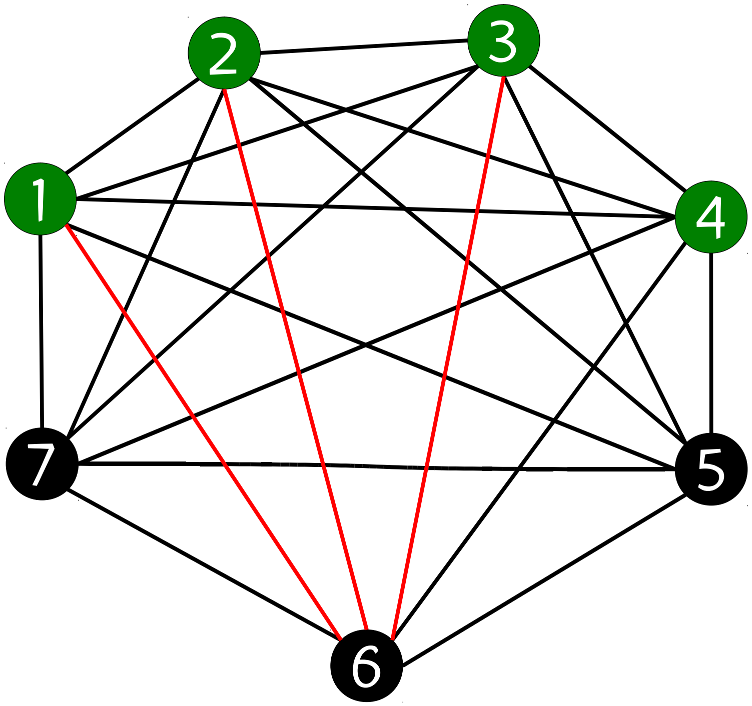
return R

X = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)

⬆

i = 3

Decoding



$X = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)$

$i = 4$

Scan V in order implied by Y

while come pair in $V \times V$ cannot communicate ($M = (V, E')$ is not complete):

Add next vertex v in order into R

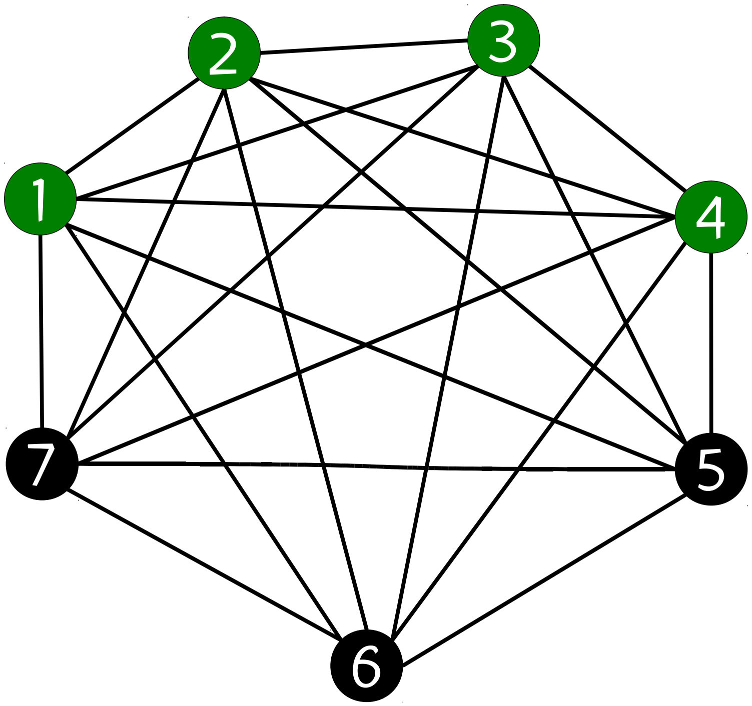
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



M is complete!

R = { 1, 2, 3, 4 }

Scan V in order implied by Y

while come pair in V × V cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

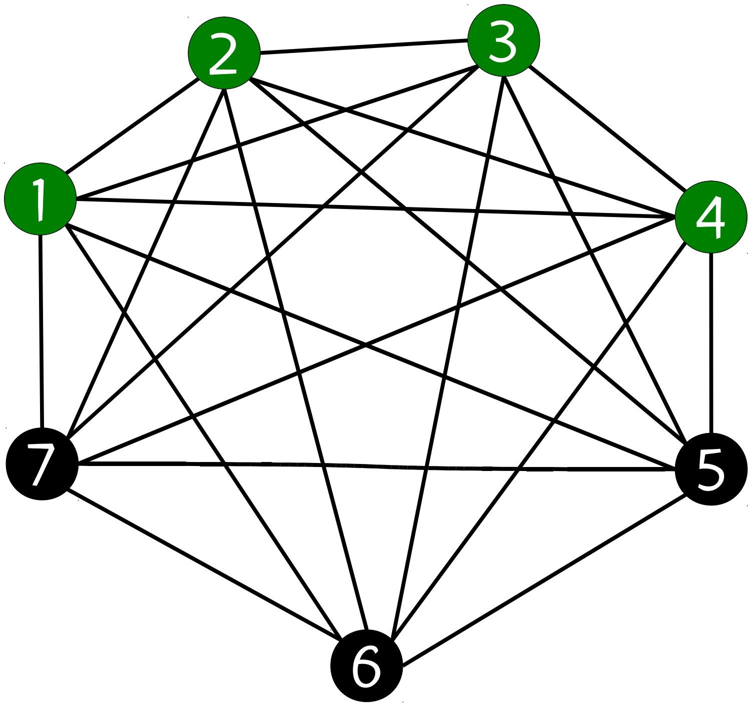
Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

Decoding



M is complete!

R = { 1, 2, 3, 4 }

Scan V in order implied by Y

while come pair in V × V cannot communicate (M = (V, E') is not complete):

Add next vertex v in order into R

Compute set X of pairs that do not communicate that would if v becomes a regenerator

Add X to E'

end while

return R

local search

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.

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.

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 constraint).

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 constraint).
- If no wavelength converters are available, the same wavelength must be assigned along the entire route (wavelength continuity constraint).

Routing and wavelength assignment (RWA)

- Variants of RWA are characterized by different optimization criteria, traffic patterns, and whether wavelength conversion is available or not.

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:

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.

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.

at&t
Your world. Delivered.

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.

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.

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

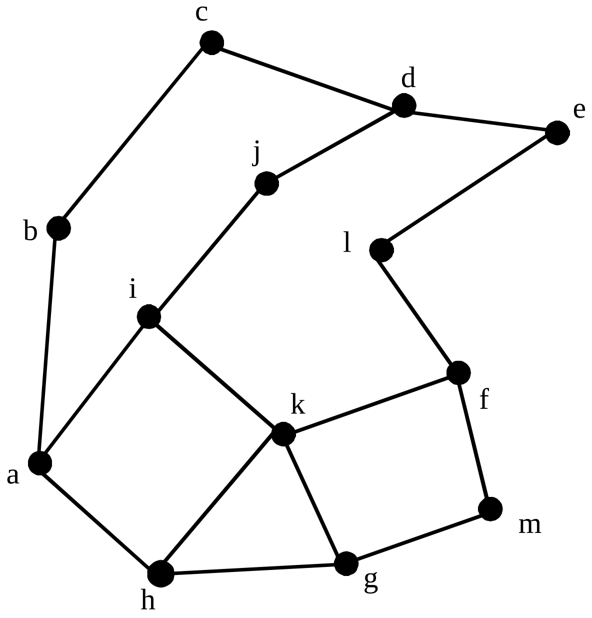
c ↔ m

d ↔ b

e ↔ h

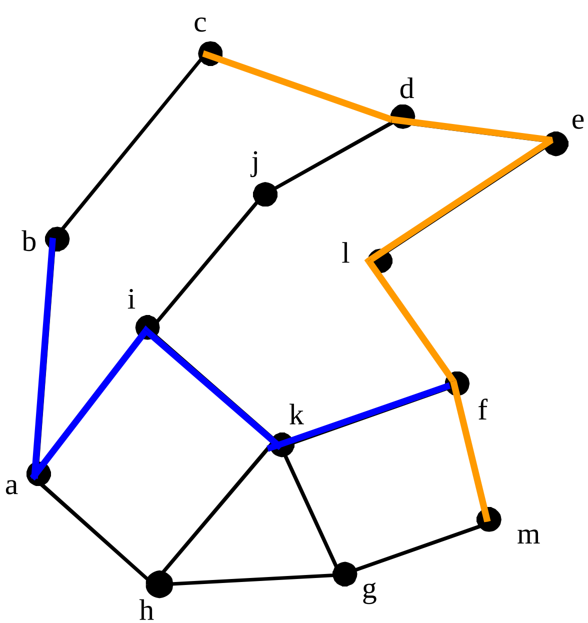
a ↔ e

b ↔ f

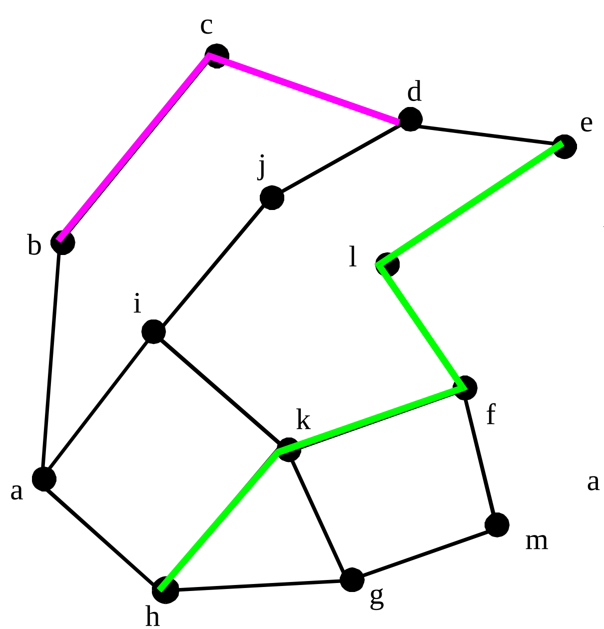


Routing and wavelength assignment (RWA)

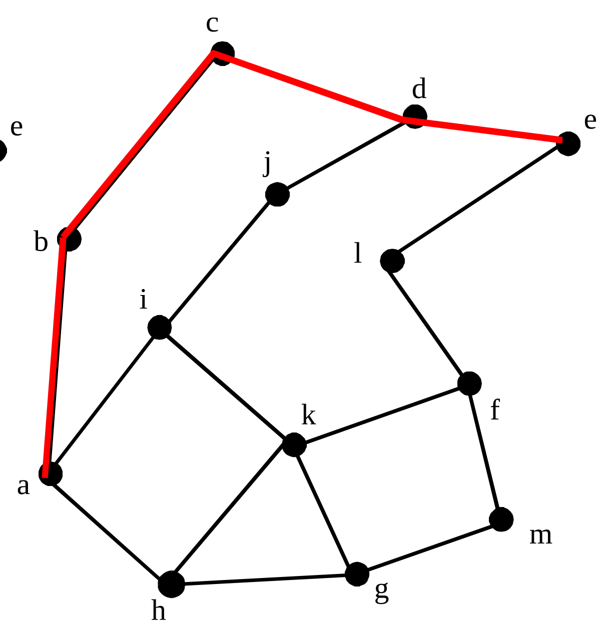
Connections: (a ↔ e) (b ↔ f) (c ↔ m) (d ↔ b) (e ↔ h)



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.

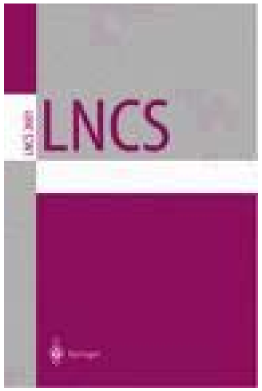
Heuristic of N. Skorin-Kapov (EJOR, 2007)

- 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, vol. 50, pp. 503–518, 2011.

Tech report version:

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

BRKGA for RWA: GA-RWA

Noronha, R., and Ribeiro (2011)

- 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.

BRKGA for RWA: GA-RWA

Noronha, R., and Ribeiro (2011)

- 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.

at&t

Your world. Delivered.

BRKGA for RWA: GA-RWA

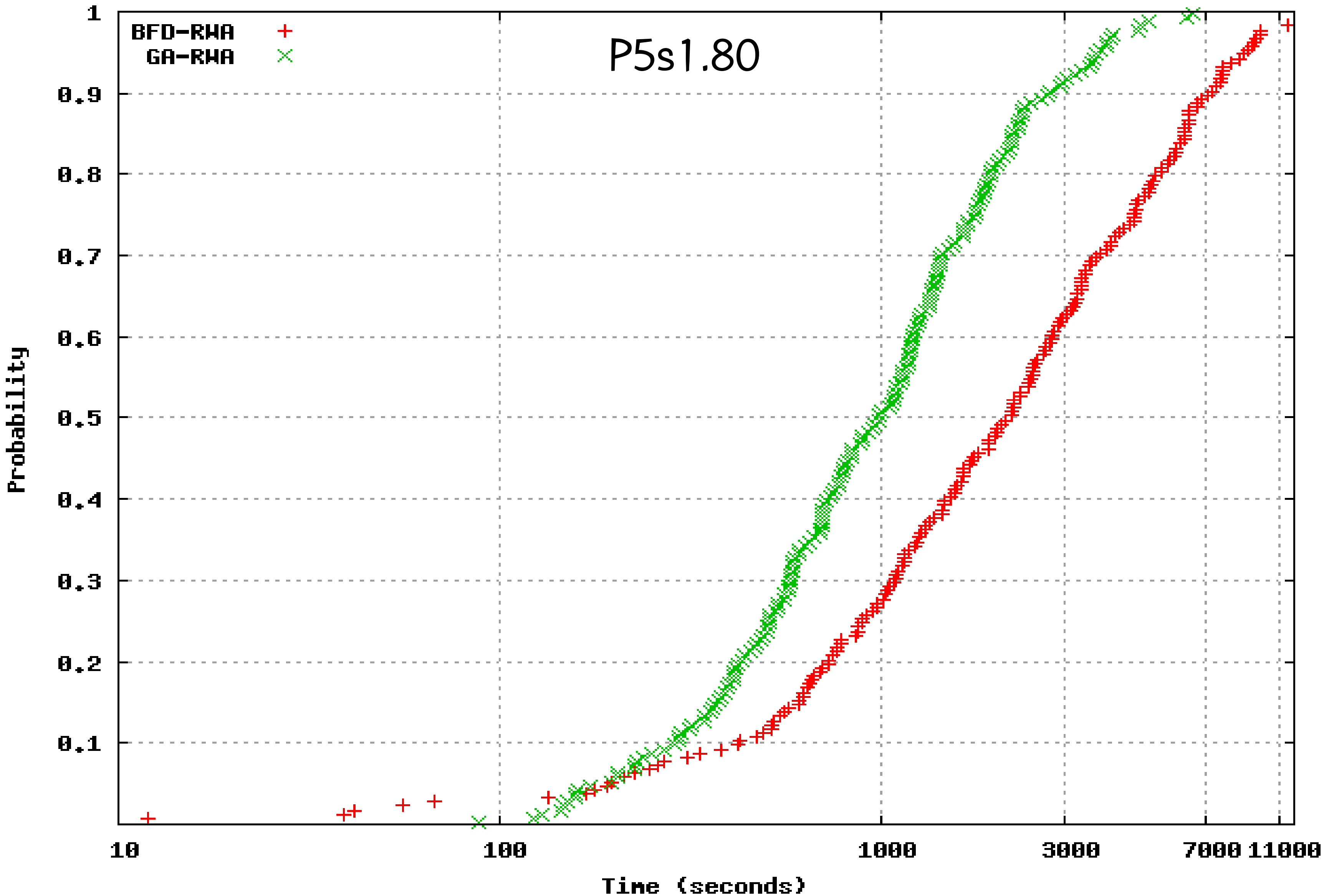
Noronha, R., and Ribeiro (2011)

- 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.

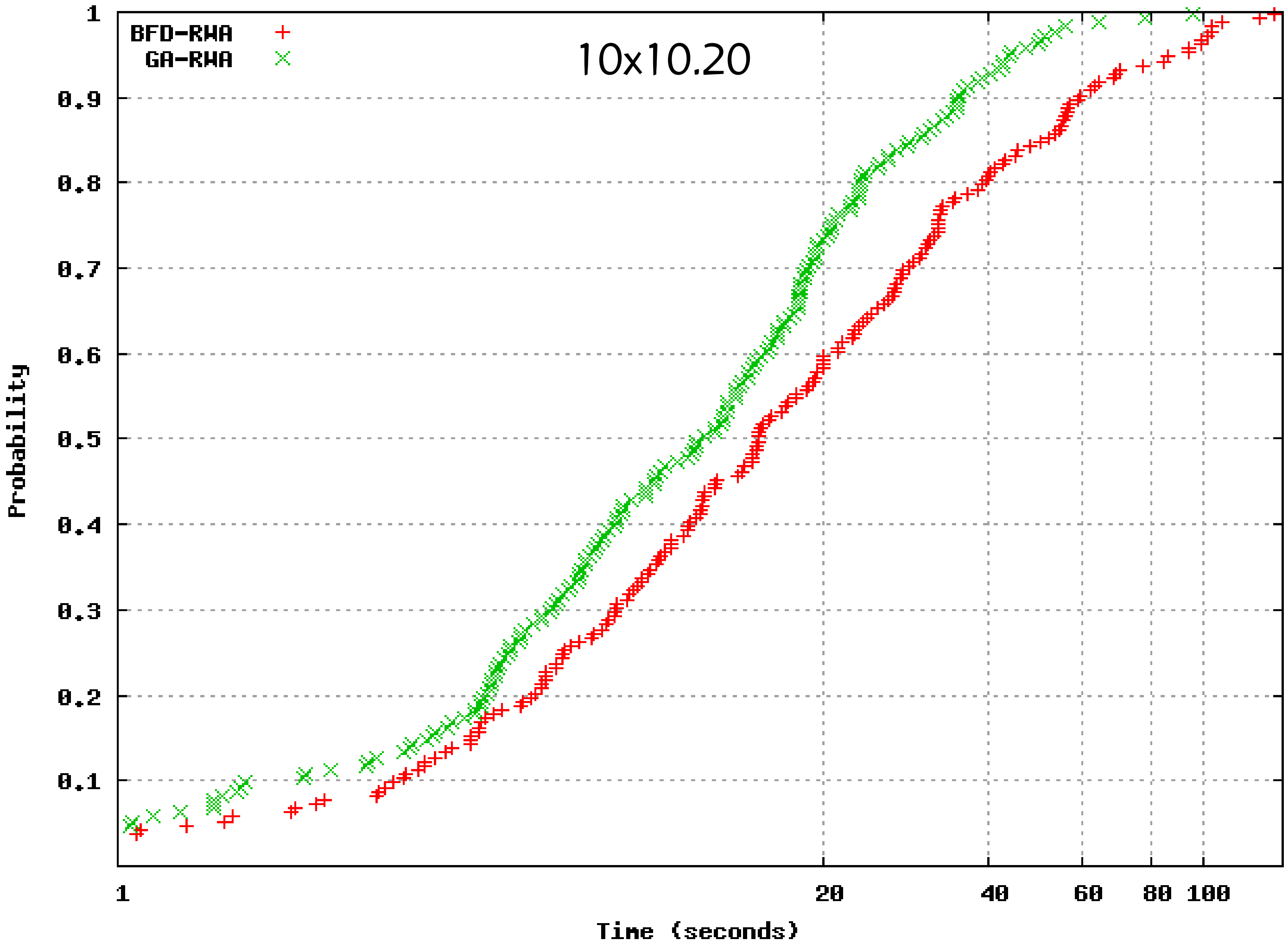
Since there are many ties connection pairs with The same SP(i) value, in the original algorithm of Skorin-Kapov, ties are broken at random. In the BRKGA, the algorithm "learns" how to break ties.

Experiments

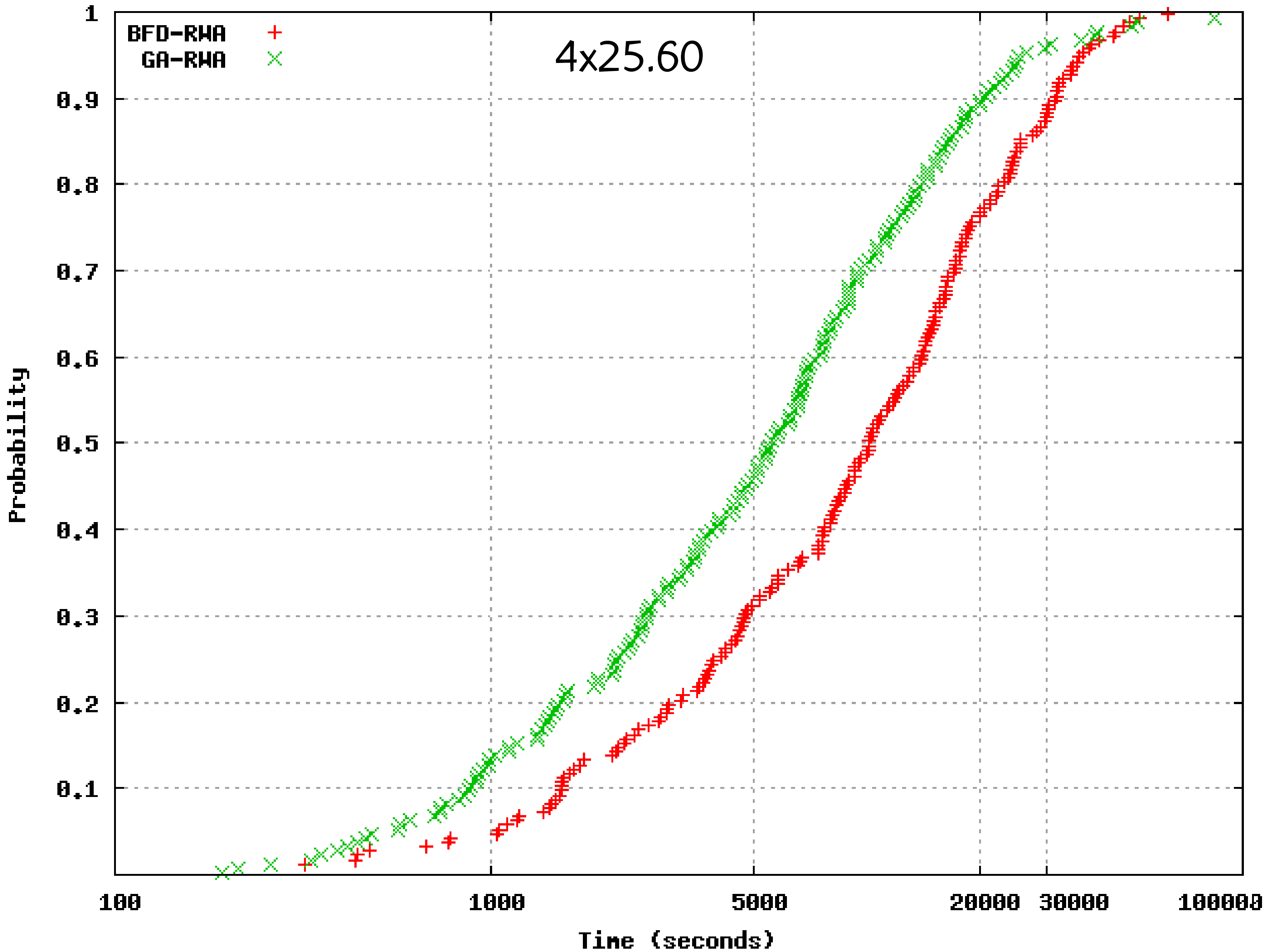
- Compare multi-start version of Skorin-Kapov's heuristic (MS-RWA) with GA-RWA.
- Make 200 independent runs of each heuristic of each heuristic on five instances, stopping when target solution was found (target was set to be best solution found by MS-RWA after 10,000 multi-start iterations.
- Plot CDF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



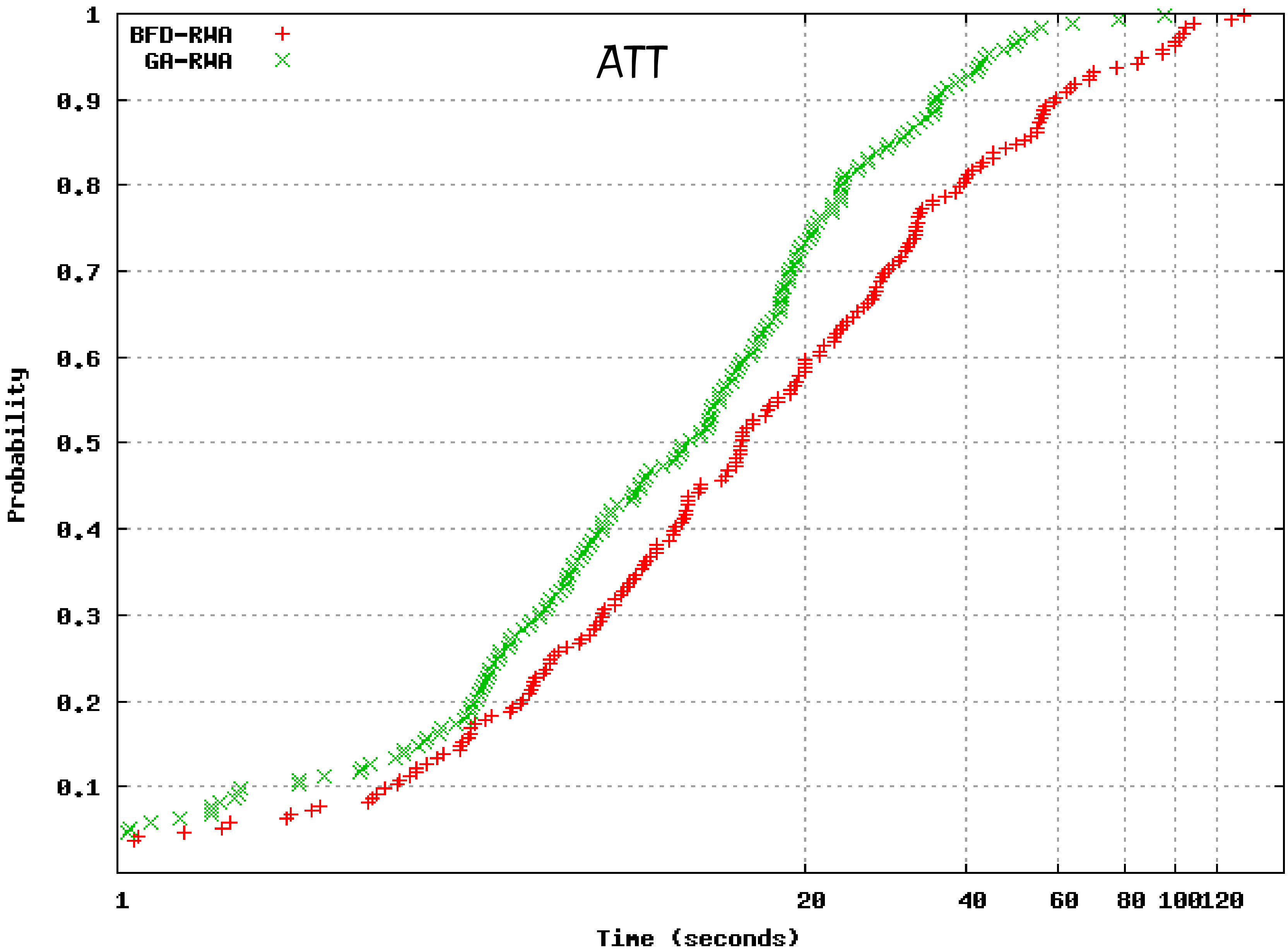
at&t
Your world. Delivered.

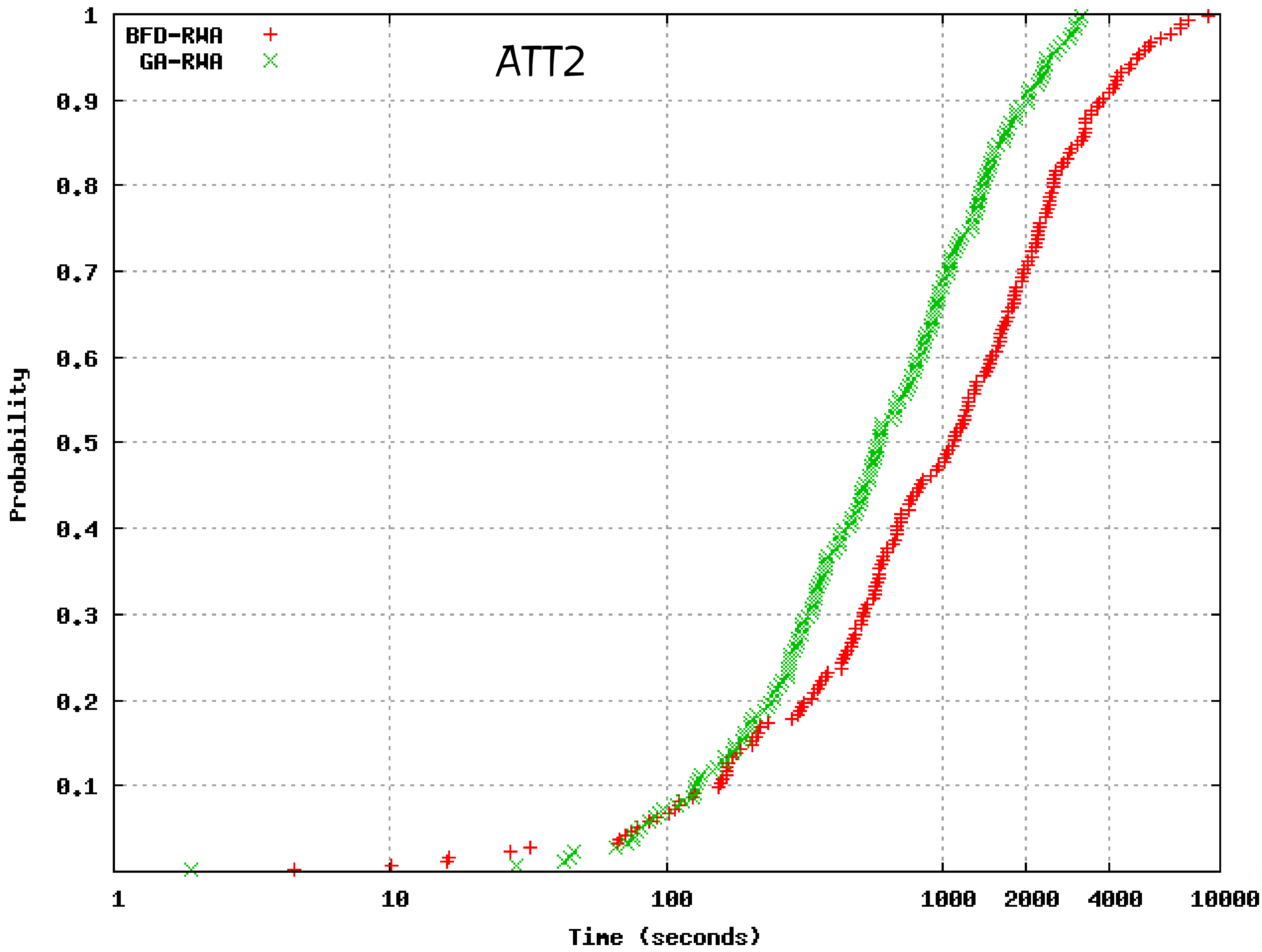


Your world. Delivered.



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


ATT
BFD-RWA
GA-RWA
Probability
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
20
40
60
80
100
120
Time (seconds)



at&t
Your world. Delivered.

Concluding remarks

Concluding remarks

- A small modification of Bean's RKGA results in a BRKGA.

Concluding remarks

- A small modification of Bean's RKGA results in a BRKGA.
- Though small, this modification, leads to significant performance improvements.

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.

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.

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. Calls to decoder are independent.

Concluding remarks

- BRKGA have been applied in a wide range of application areas, including scheduling, packing, cutting, tollbooth assignment, ...

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.

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

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 other applications in telecommunications, including the degree-constrained and the capacitated spanning tree problems and a metropolitan network design problem.

Thanks!

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

http://www2.research.att.com/~mgcr