Algorithms for node placement in path-disjoint network monitoring

Seminar given at the University of Pittsburgh
Pittsburgh, PA ✳ March 4, 2010

Mauricio G. C. Resende
AT&T Labs Research
Florham Park, New Jersey
mgcr@research.att.com

Joint work with:

L. Breslau (AT&T Research), I. Diakonikolas (Columbia U.),
N. Duffield (AT&T Research), Y. Gu (NEC Research),
M. Hajiaghayi (AT&T Research), D.S. Johnson (AT&T Research),
H, Karloff (AT&T Research), and S. Sen (AT&T Research)

"Host Placement for Path-Disjoint Monitoring"

AT&T Labs Research Technical Report, March 2010.

Summary

- Network monitoring with tomography
- Minimum monitoring set (MMS) problem
- Algorithms for MMS
  - Integer programming
  - Greedy algorithm
  - Genetic algorithm
  - Double hitting set heuristic
- Computational experiments
- Concluding remarks

Network monitoring with tomography

IP Monitoring

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

IP Monitoring

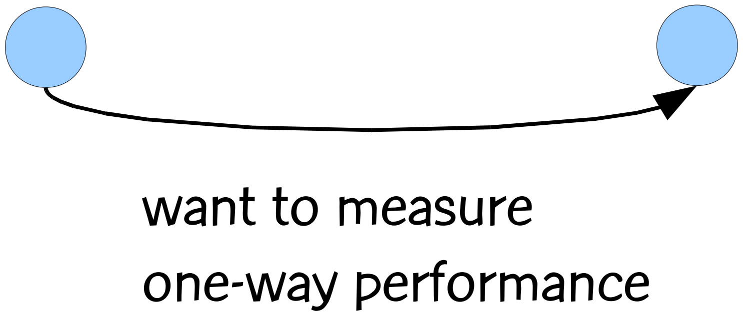
- Internet Service Providers need to monitor the performance of customer traffic within their networks.
- More specifically, ISPs want to measure:
  - Unidirectional reachability
  - Packet loss rate
  - Packet delay along the edge-to-edge paths followed by customer traffic

IP Monitoring

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

IP Monitoring

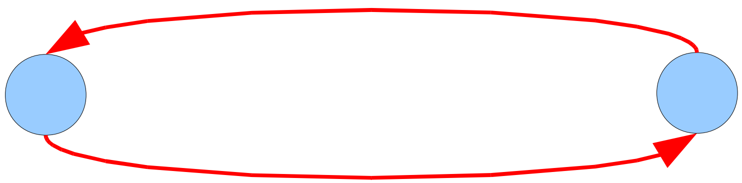
- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;



IP Monitoring

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

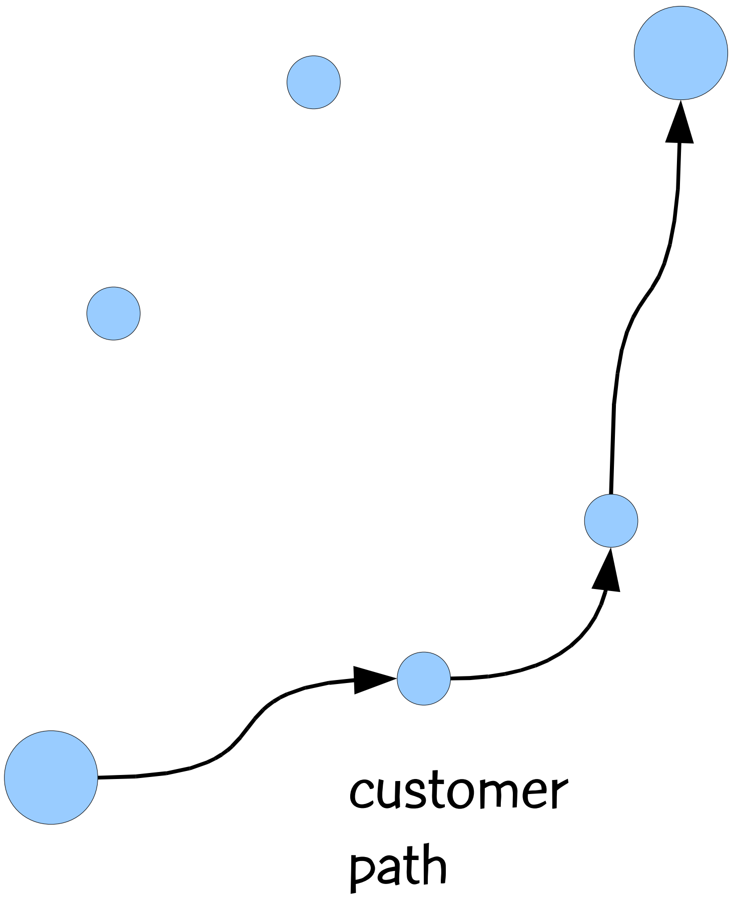
measure round-trip: hard to infer one-way performance



want to measure one-way performance

IP Monitoring

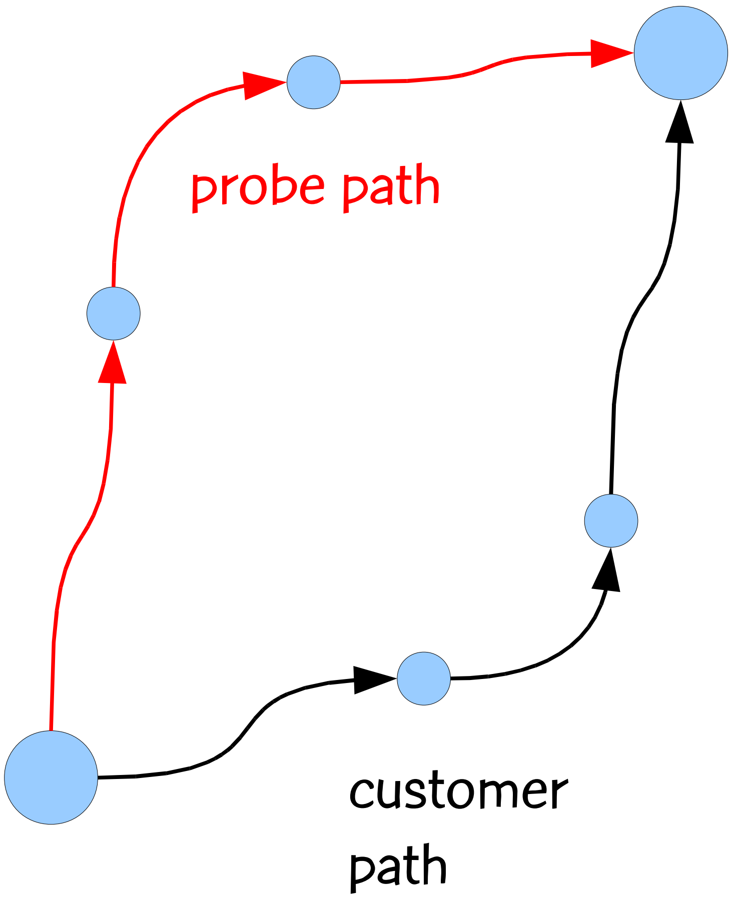
- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;
  - Their probes may not follow the customer paths, either because they transit different links, or experience different router treatment.



Node placement for monitoring

IP Monitoring

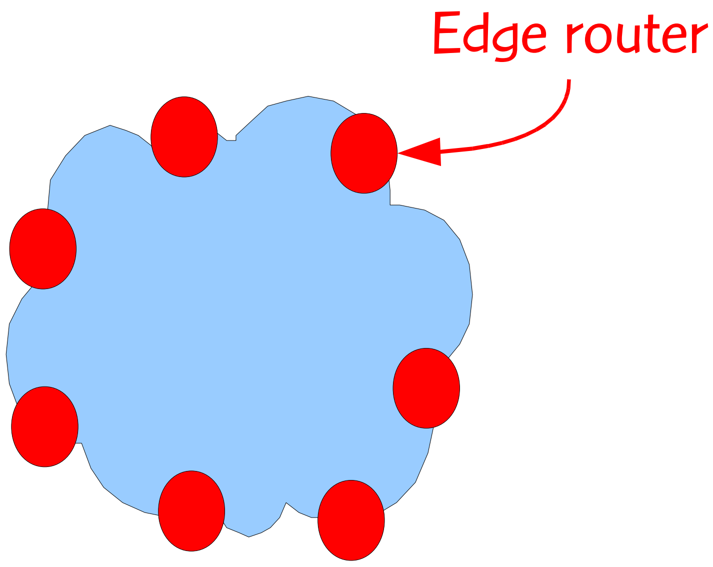
- Tools such as traceroute or ping suffer from one or both of the following limitations:
  - They measure roundtrip performance;
  - Their probes may not follow the customer paths, either because they transit different links, or experience different router treatment.



Node placement for monitoring

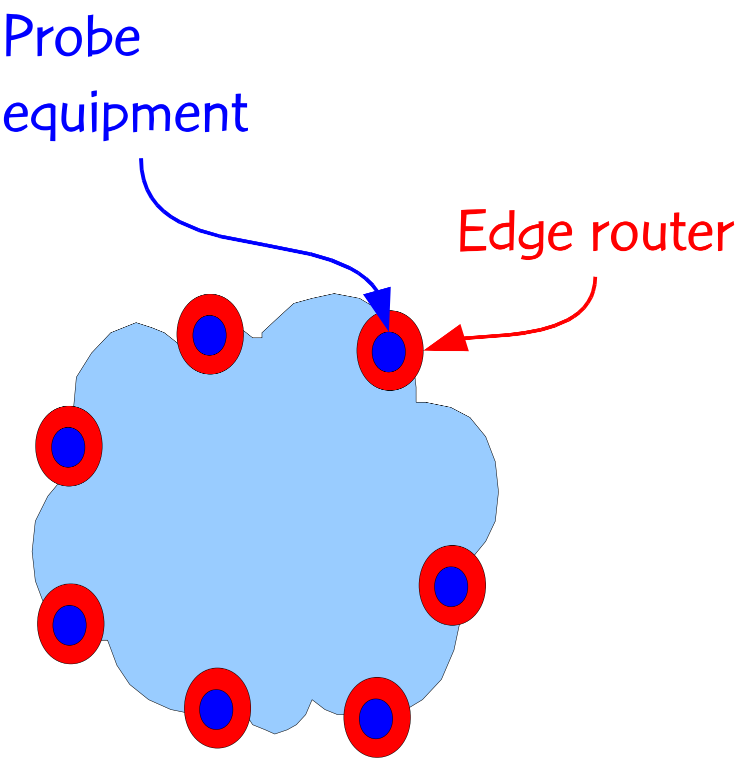
IP Monitoring

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



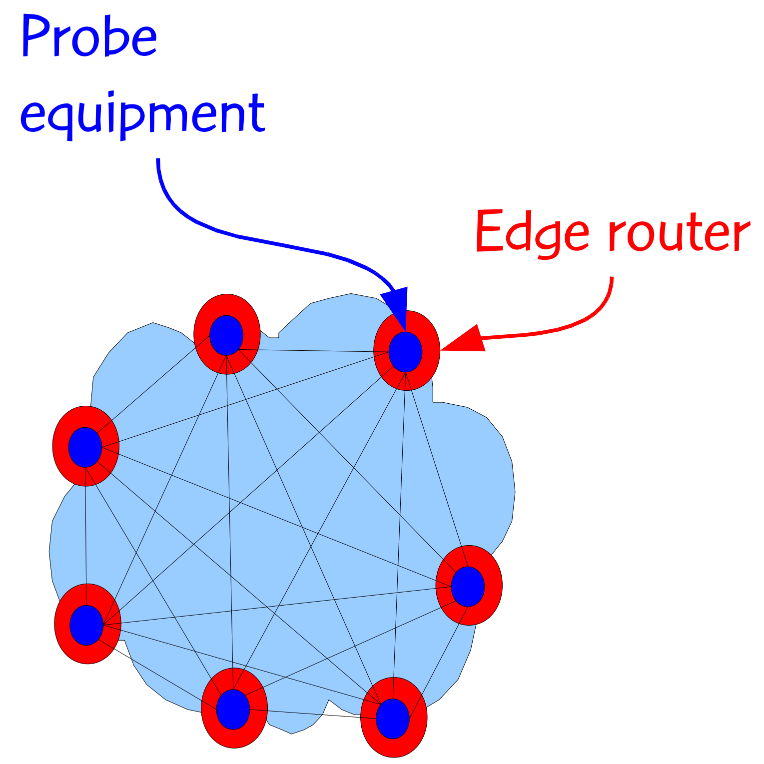
IP Monitoring

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



IP Monitoring

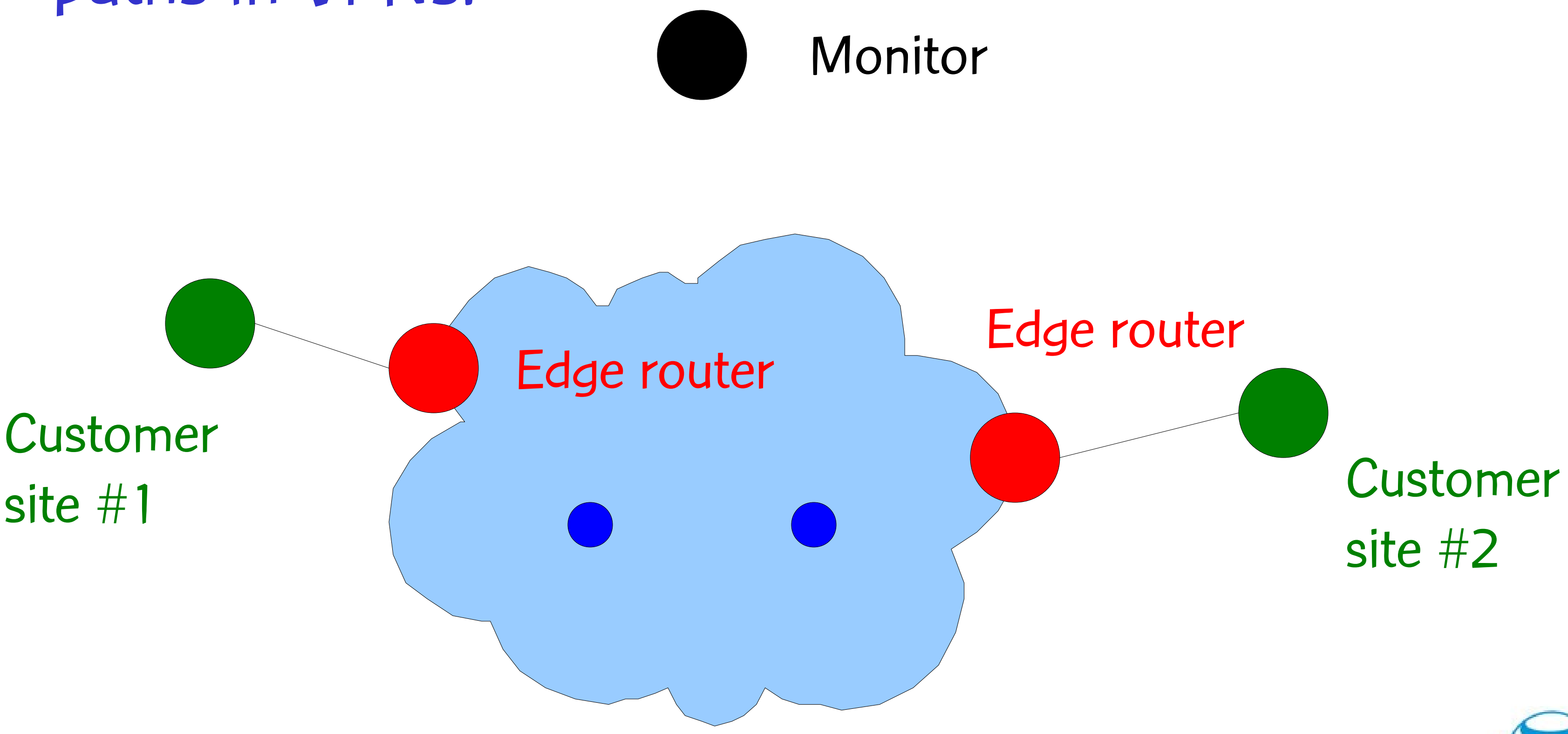
- In principle, edge routers could be equipped to launch and receive probes that follow customer traffic:
  - Could impact network performance
  - Very costly to deploy networkwide
    - Expensive equipment
    - Expensive to manage



Node placement for monitoring

IP Monitoring

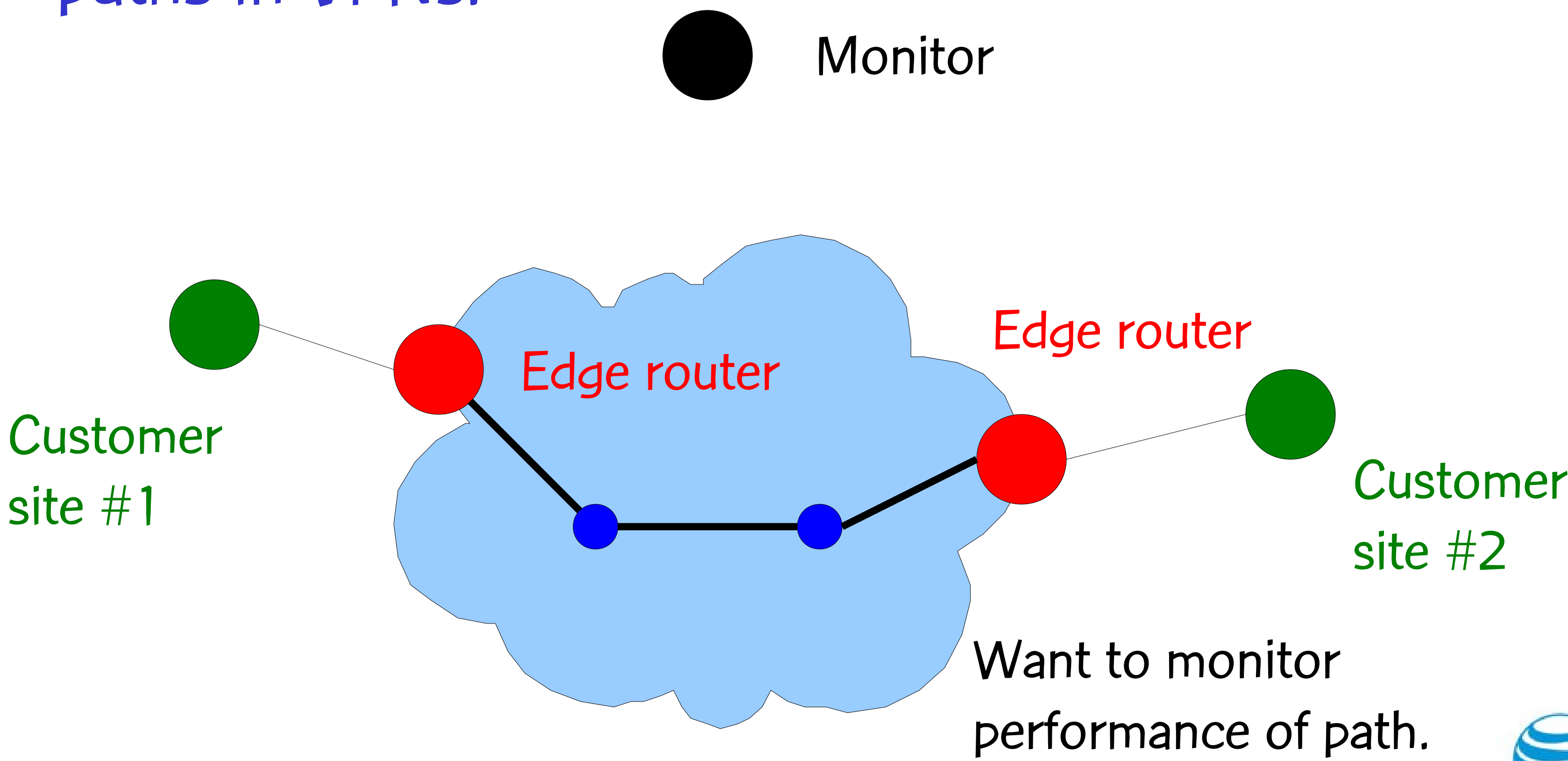
- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.



Node placement for monitoring

IP Monitoring

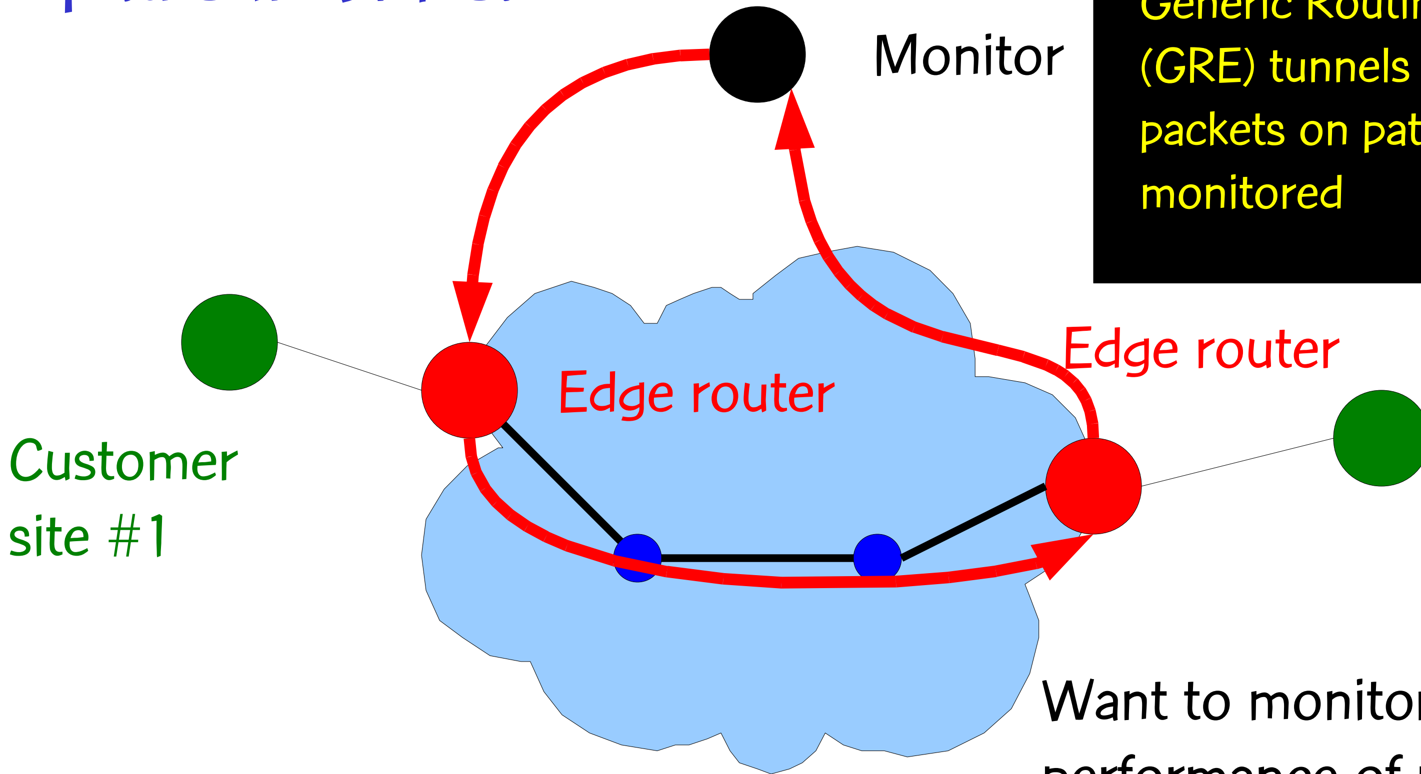
- Breslau et al. (2006) proposed a lightweight approach to measurement of customer traffic paths in VPNs.



IP Monitoring

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

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

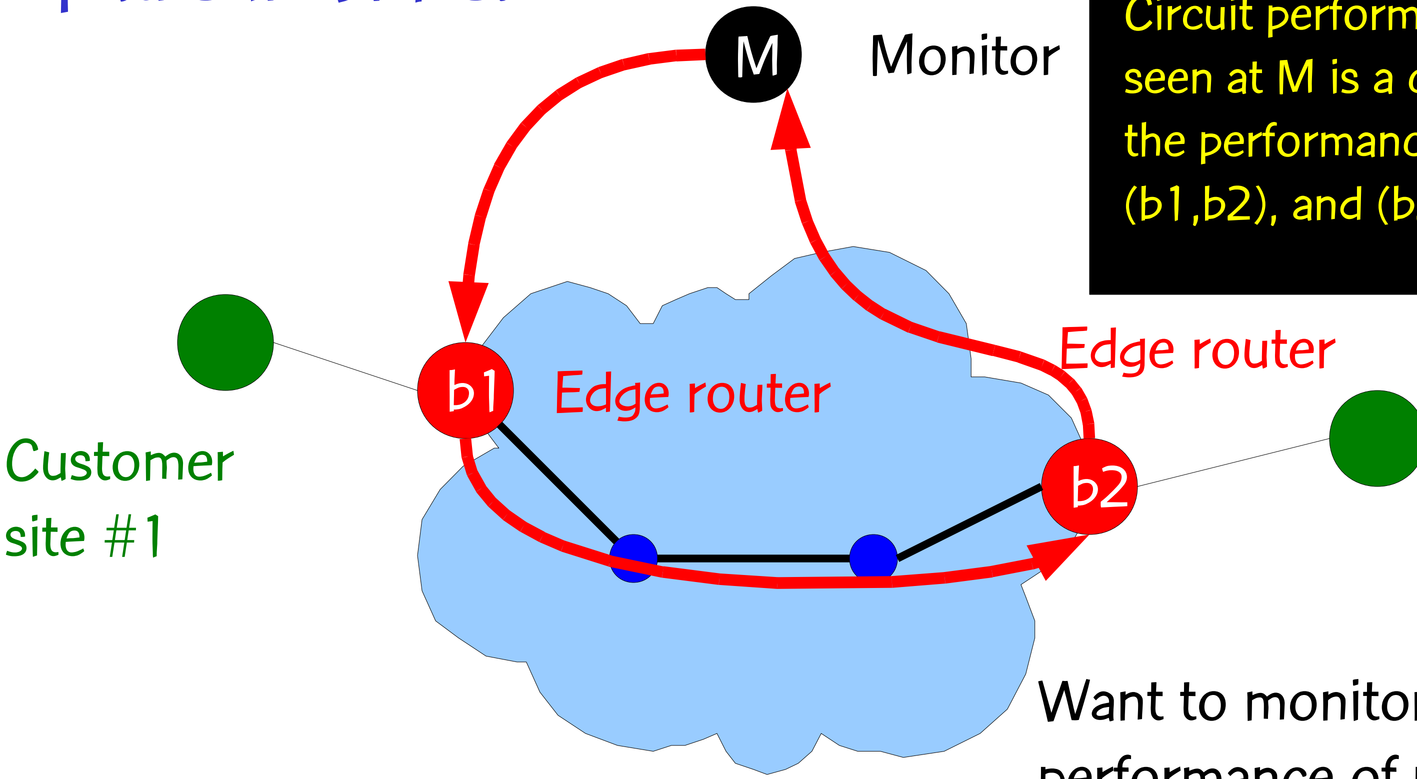


Want to monitor performance of path.

IP Monitoring

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

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

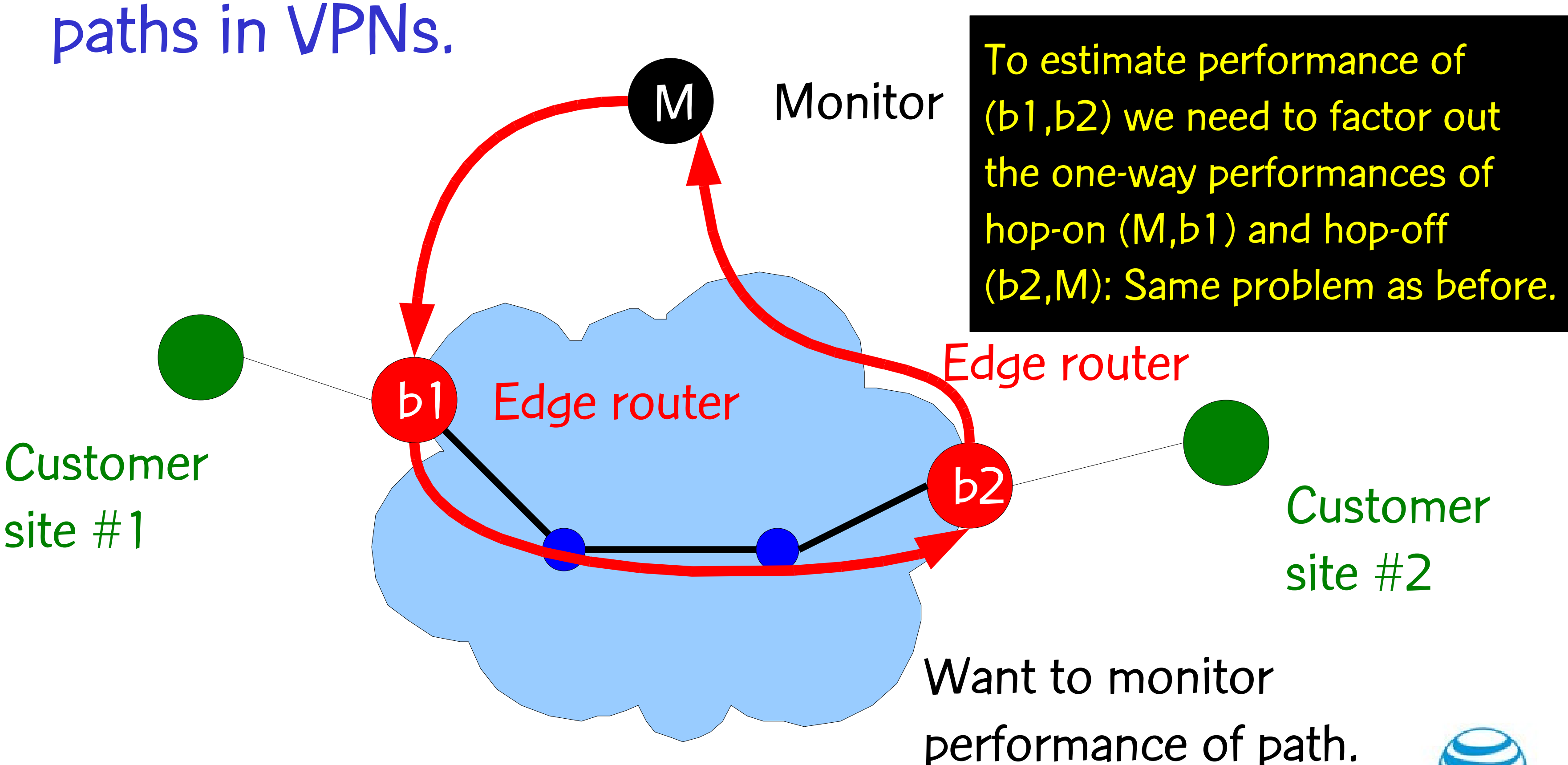


Want to monitor performance of path.

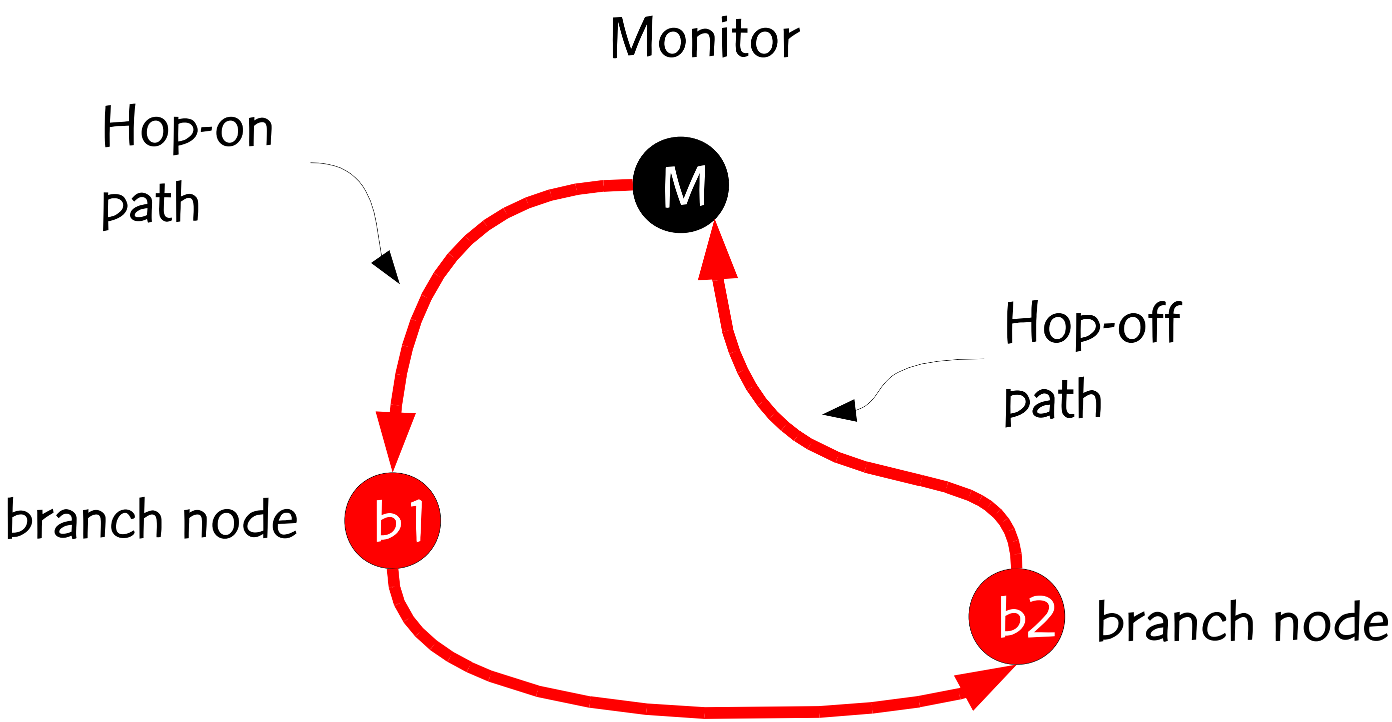
at&t
Your world. Delivered.

IP Monitoring

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

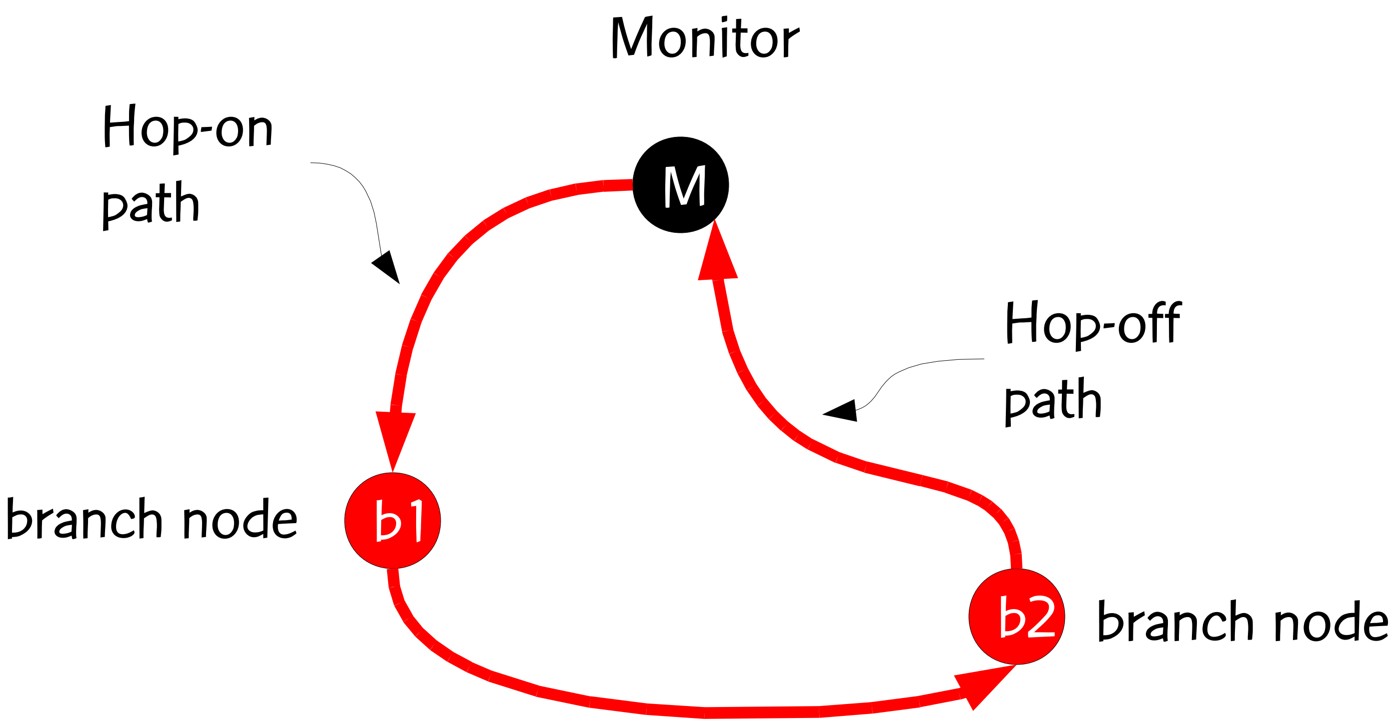


IP Monitoring



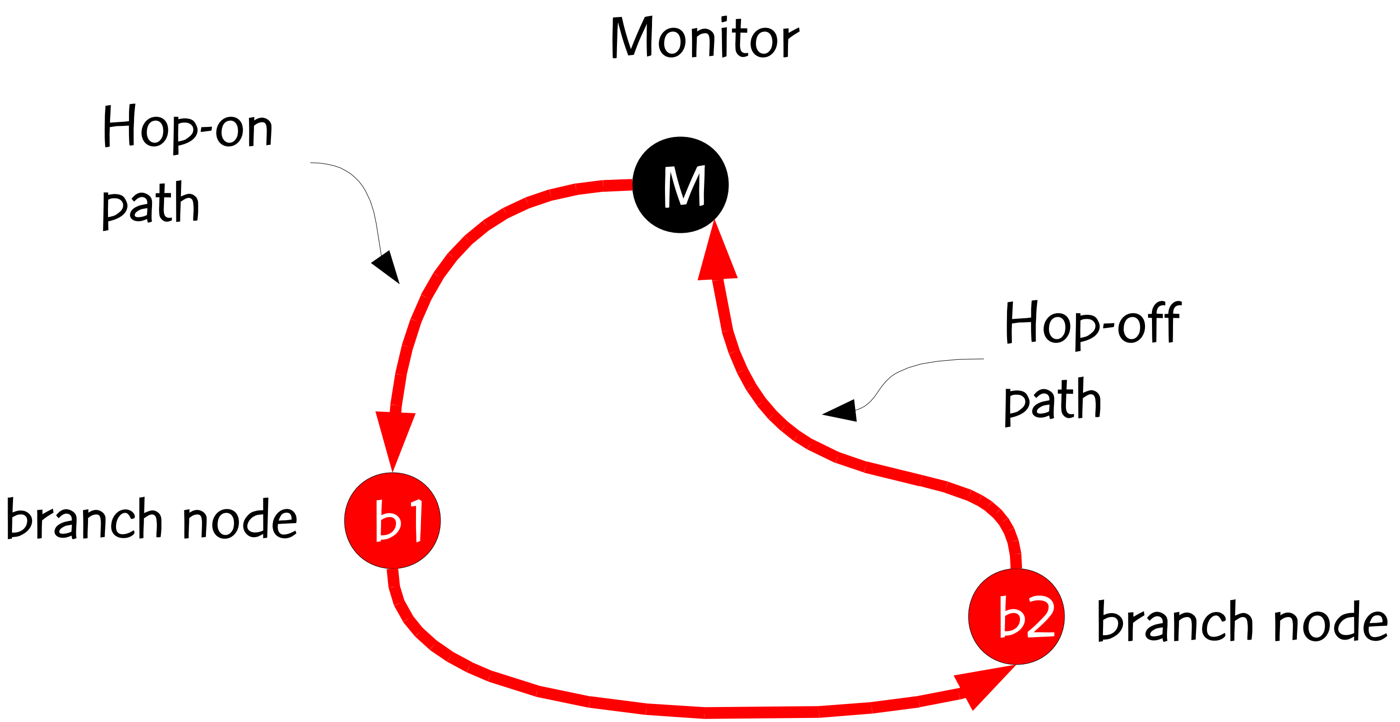
B = "branch nodes" $\subseteq$ V. We want to measure performance (e.g. loss rate) on some directed paths between vertices in B

IP Monitoring



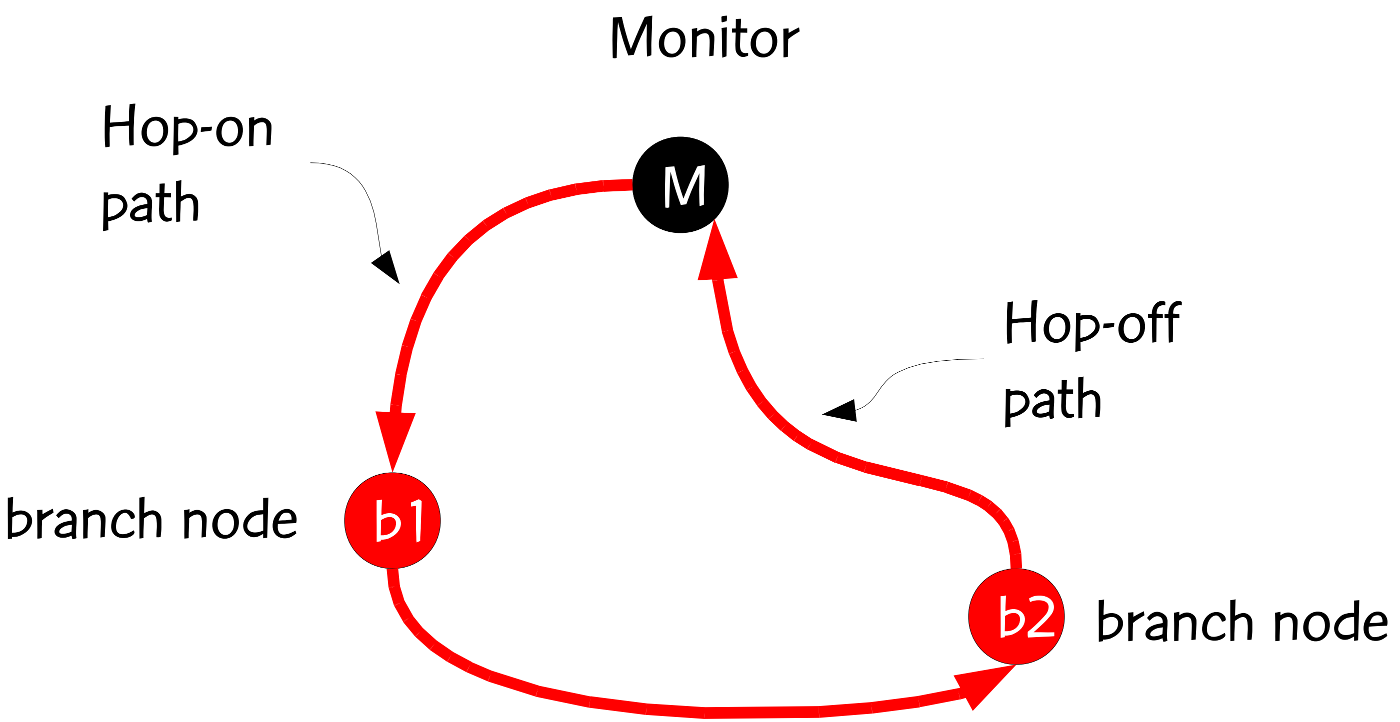
IDEA: Establish a monitoring node M. For some pairs b1, b2 ∈ B, send packet M to b1 to b2 to M.

IP Monitoring



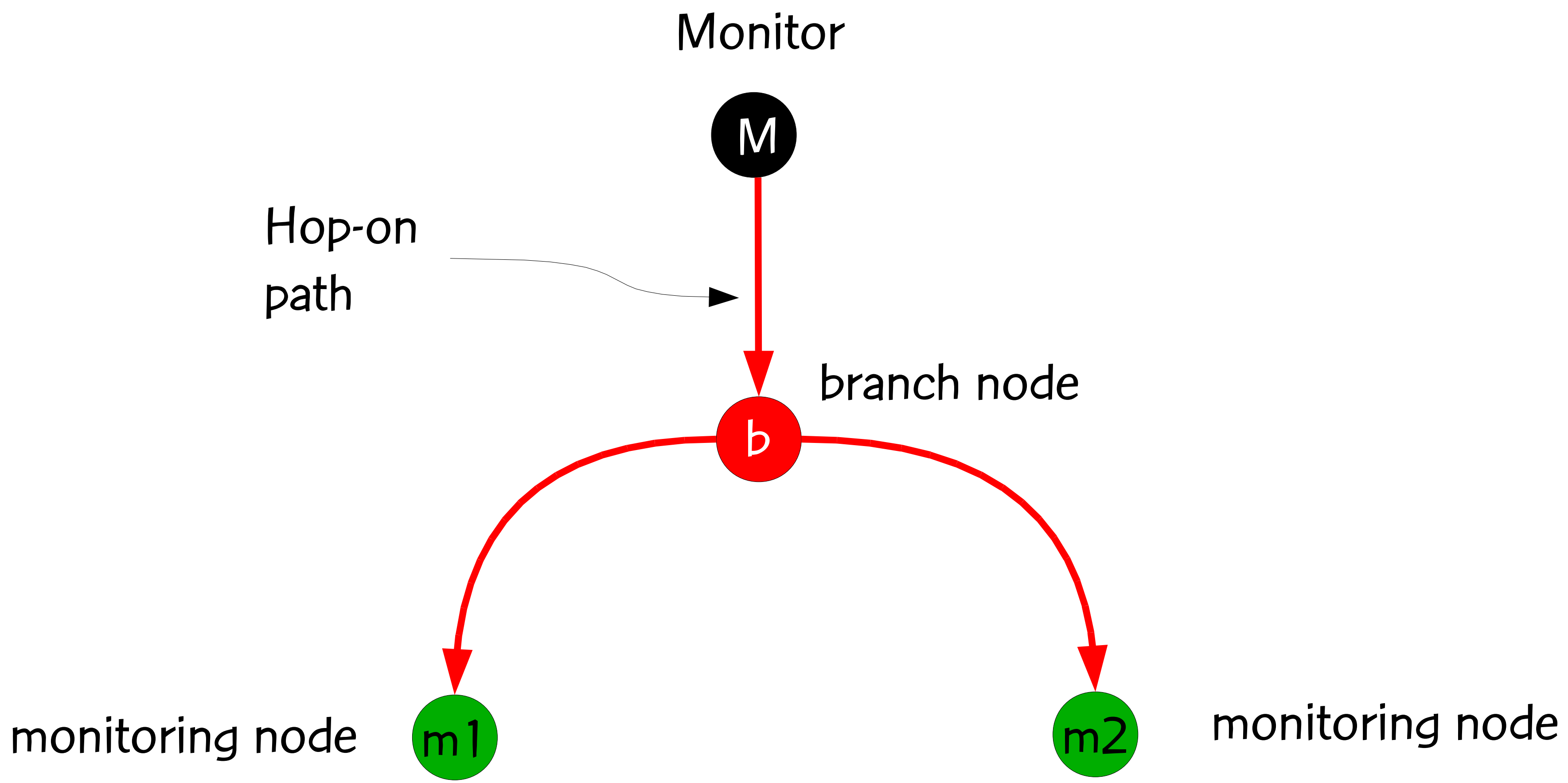
We can measure the "overall" loss rate. Must factor out the hop-on and hop-off. How?

IP Monitoring



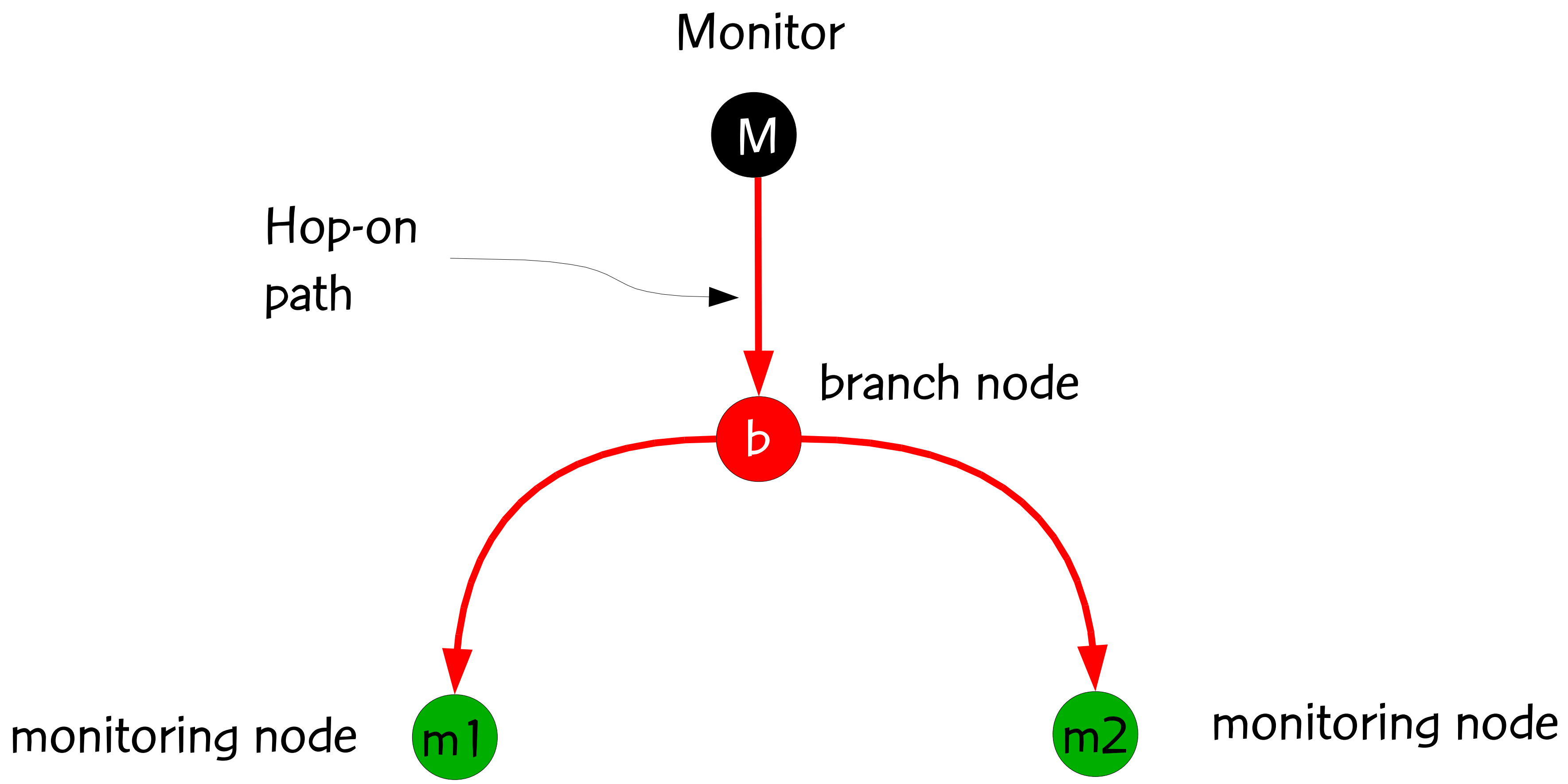
Want "disjoint" paths for independence. Must estimate loss rates for hop-on path and hop-off path to factor them out.

Estimating hop-on path loss



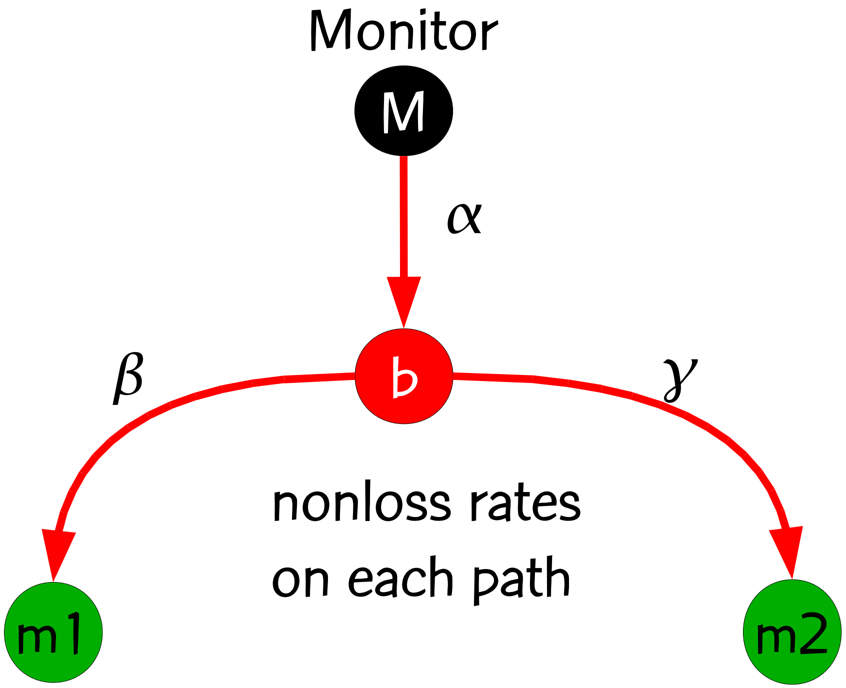
Find two "monitoring" nodes m1 and m2 and send packets from M to b and from b to m1 and m2.

Estimating hop-on path loss



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

Estimating hop-on path loss



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

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

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

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

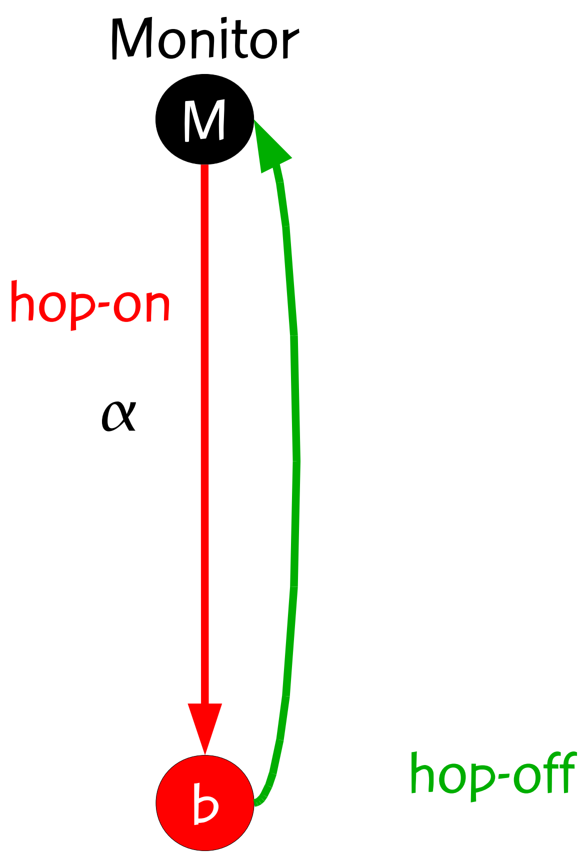
$p11 + p10 = \alpha\ \beta$

$p11 + p01 = \alpha\ \gamma$

Therefore:

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

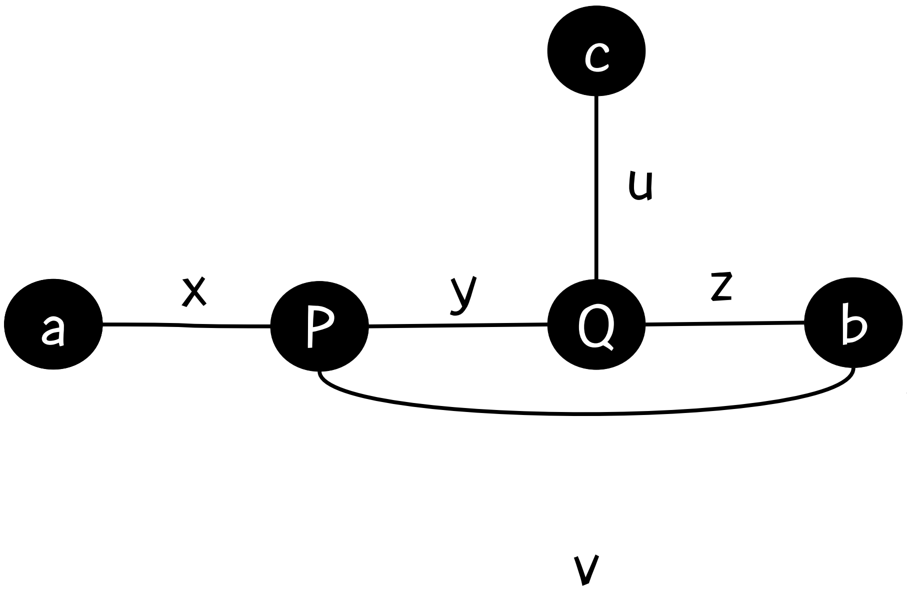
Estimating hop-off path loss



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

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

Simple lemma



LEMMA:

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

PROOF (by contradiction):

Suppose shortest paths are

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

clearly $v \geq y + z$

hence $z \leq v - y$

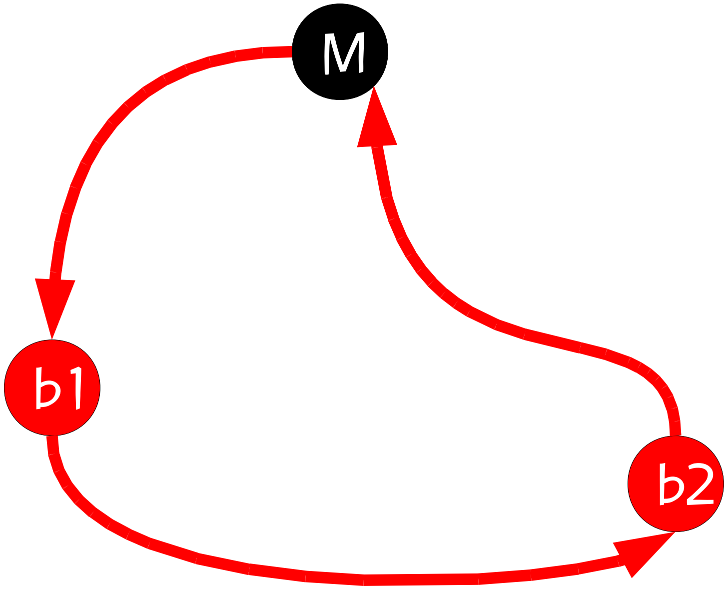
and $z < v + y$ because $y > 0$.

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

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

Consequence of simple lemma

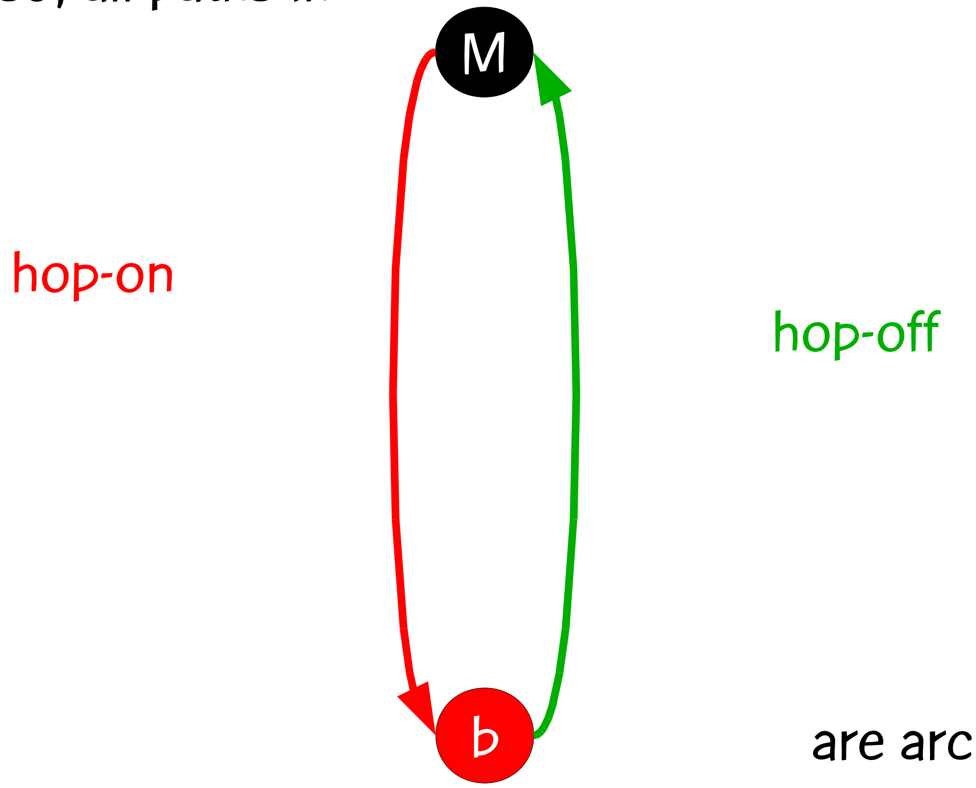
In practice, all or almost all arc weights are symmetric. If so, all paths in



are arc disjoint.

Consequence of simple lemma

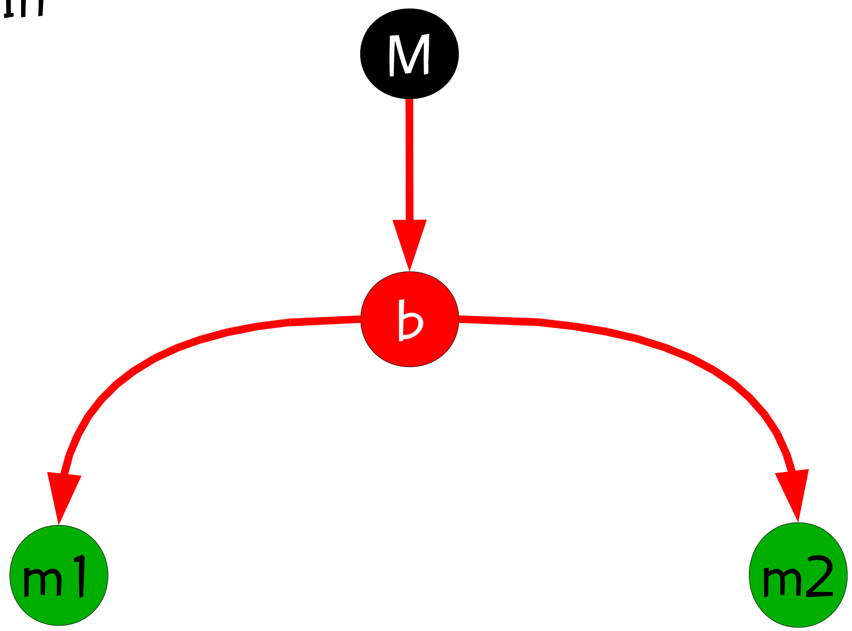
In practice, all or almost all arc weights are symmetric. If so, all paths in



are arc disjoint.

Consequence of simple lemma

In

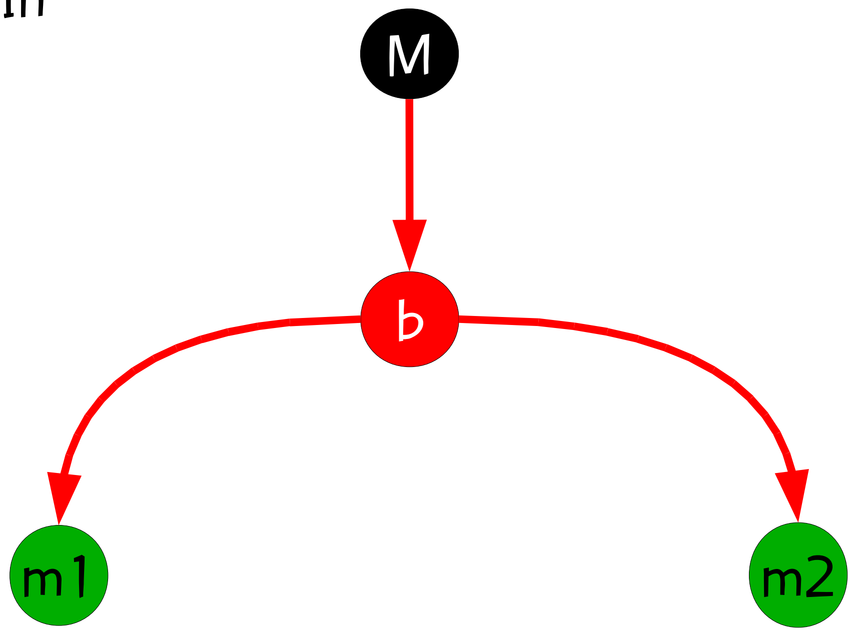


The M $\longrightarrow$ b and b $\longrightarrow$ m1 paths are arc disjoint, as are the M $\longrightarrow$ b and b $\longrightarrow$ m2 paths.

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

Consequence of simple lemma

In



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

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

Not disjoint in general.

Minimum monitoring set problem

Monitor placement

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

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

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

Monitor placement

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

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

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

Why every shortest path?

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

Monitor placement

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

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

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

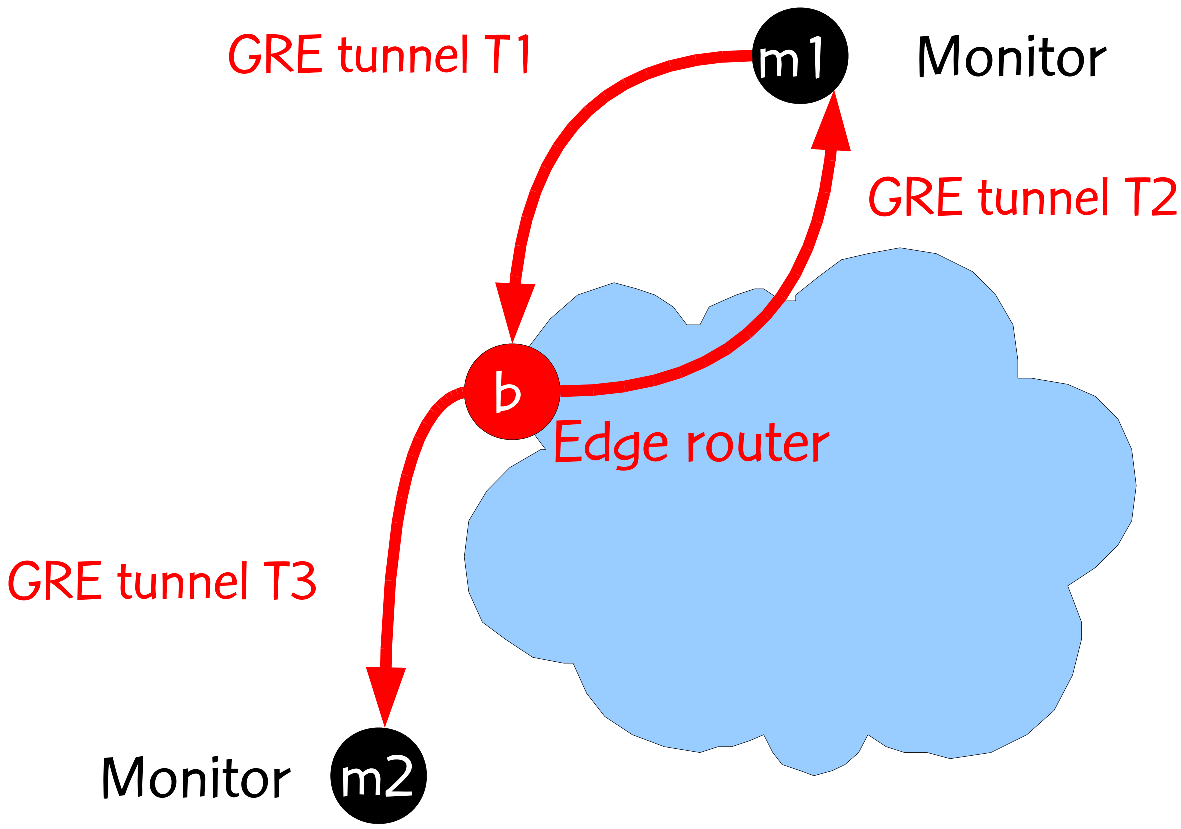
Why every shortest path?

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

Obs: weights need not be symmetric.

IP Monitoring

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

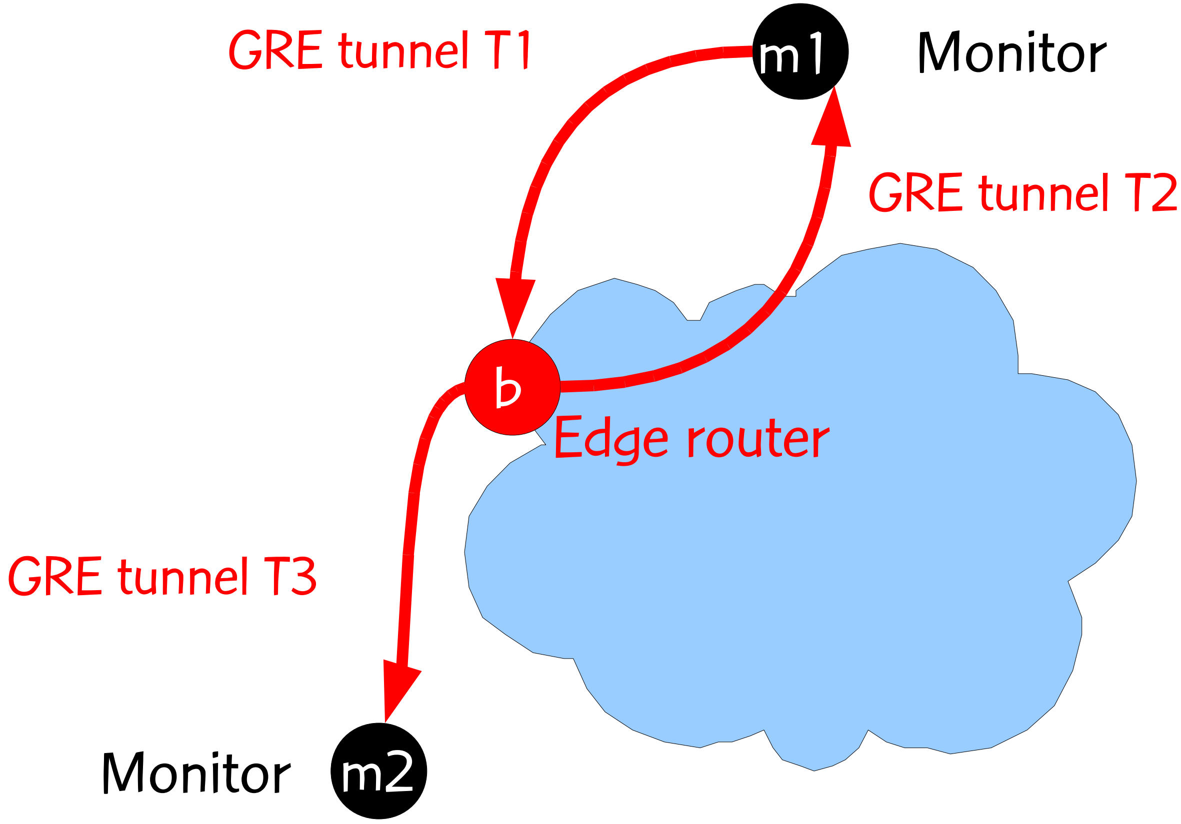


Two monitors and three GRE tunnels make up the multicast overlay topology.

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

IP Monitoring

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



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

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

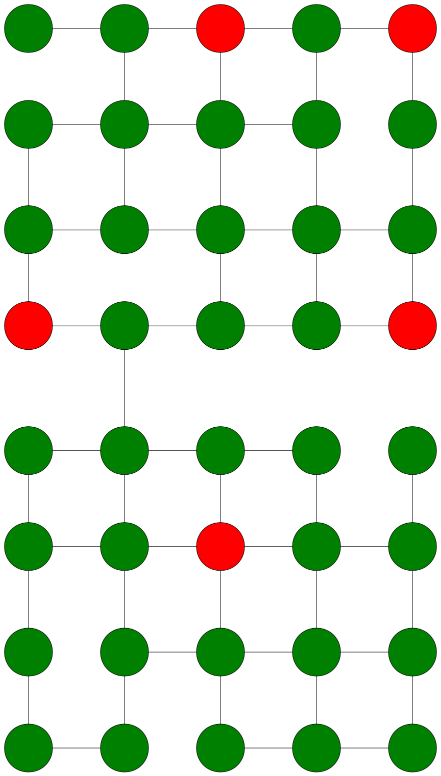
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



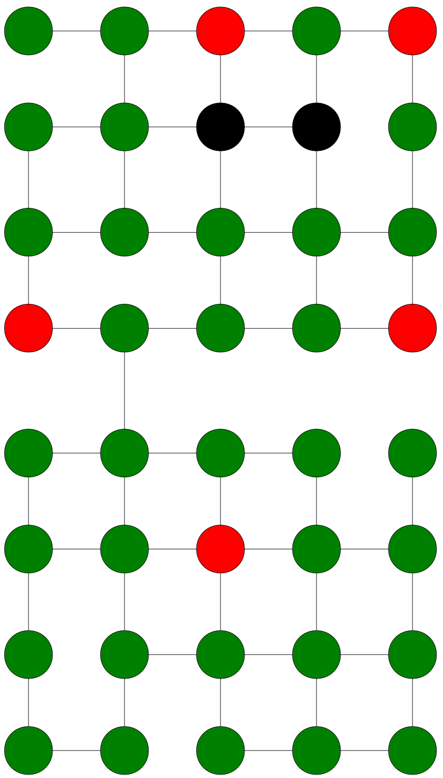
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



Node placement for monitoring

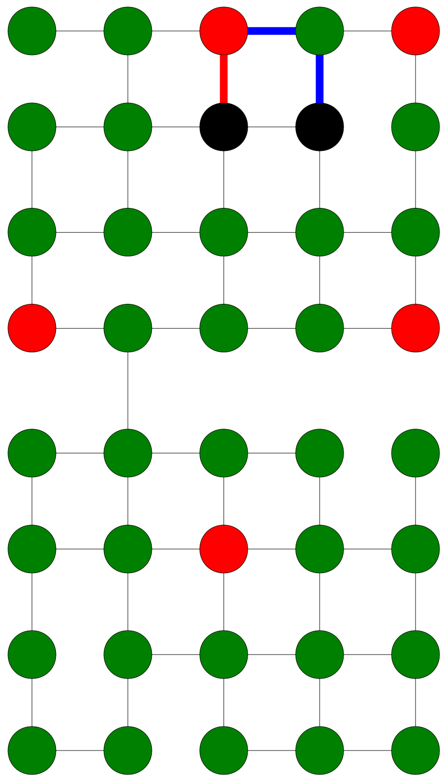
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



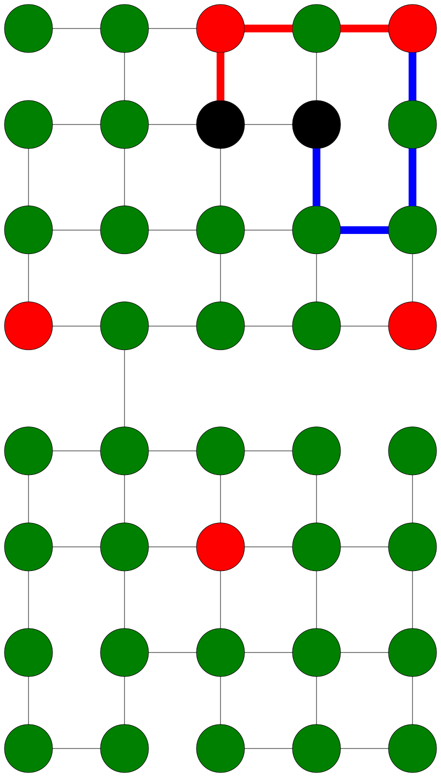
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



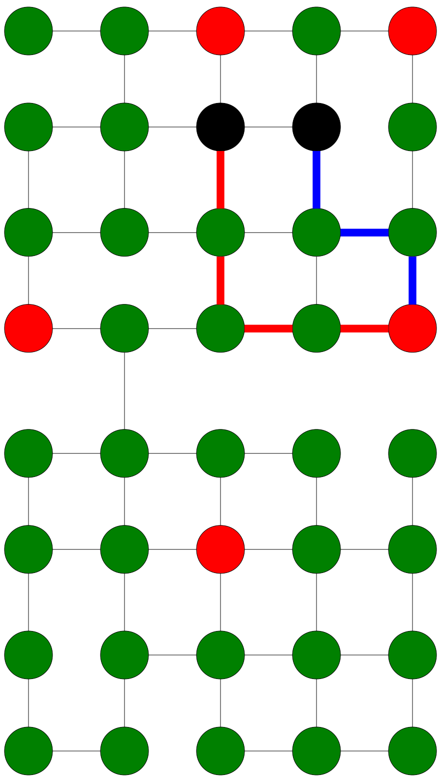
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



Node placement for monitoring

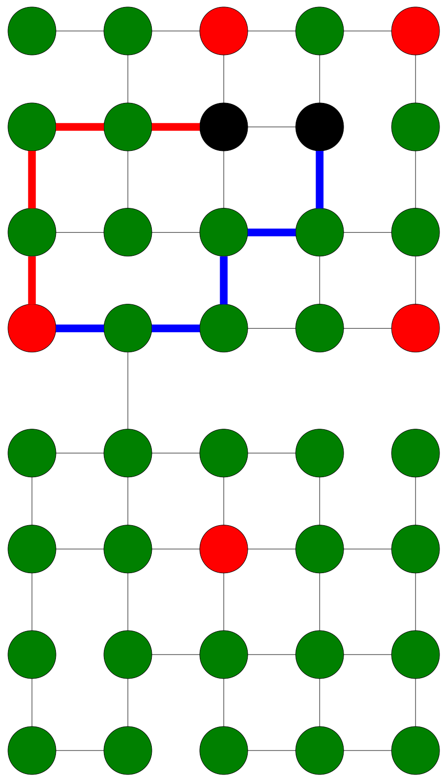
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



Node placement for monitoring

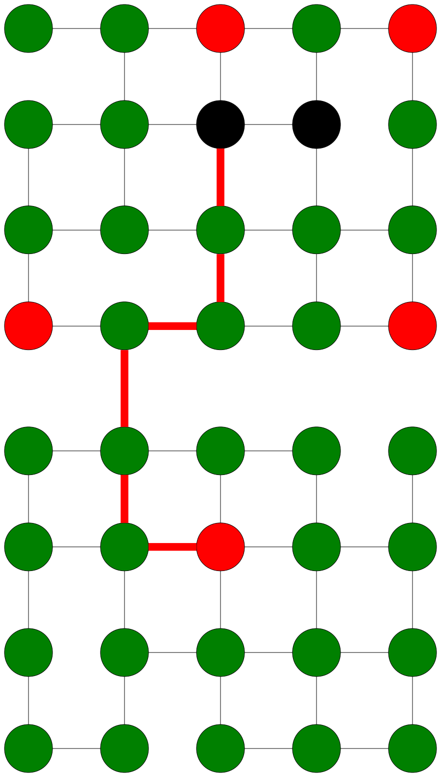
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



Node placement for monitoring

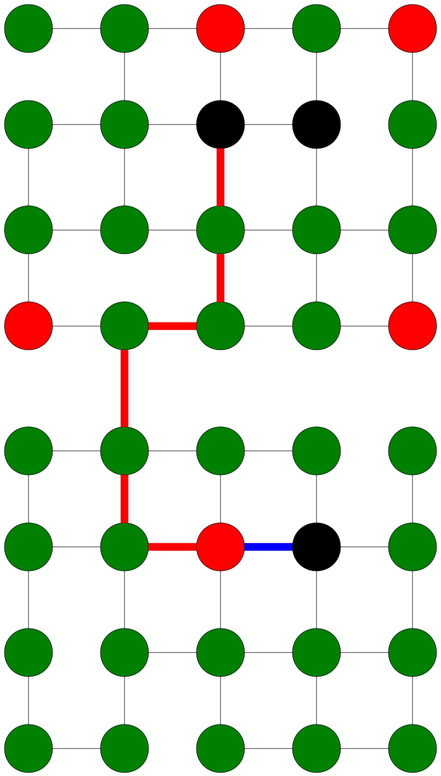
Minimum monitoring set problem

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

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

One objective is to minimize N.

paths are given and are fixed



Set covering with pairs

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

Minimum monitoring set problem

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

Another application: Redundant content distribution

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

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

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

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

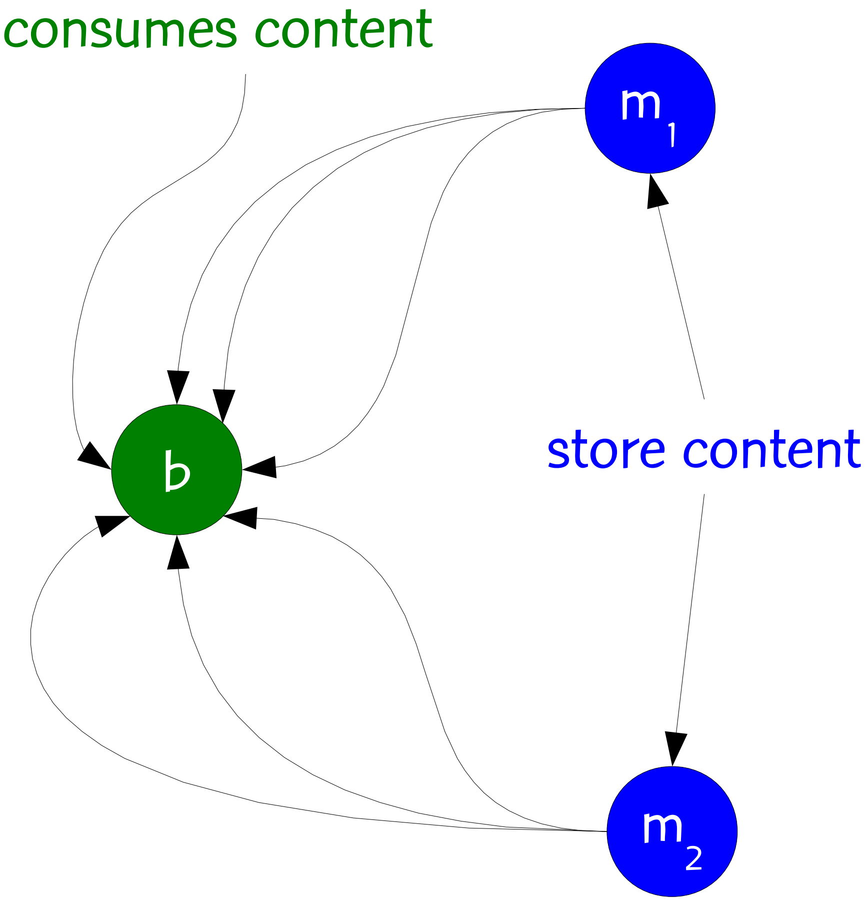
Another application: Redundant content distribution

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

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

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

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



at&t
Your world. Delivered.

Algorithms for minimum monitoring set problem

Algorithms for MMS problem

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

Algorithms for MMS problem

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

Integer programming

Integer programming model

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

$x_v = 1$ iff node v is chosen

Integer programming model

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

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

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

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

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

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

Integer programming model

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

$x_v = 1$ iff node v is chosen

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

$y_{u,v} \leq x_u$

$y_{u,v} \leq x_v$

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

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

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

Integer programming model

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

$x_v = 1$ iff node v is chosen

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

$y_{u,v} \leq x_u$

$y_{u,v} \leq x_v$

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

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

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

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

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

Integer programming model

$\min \sum x_v$

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

$x_v = 1$ iff node v is chosen

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

$y_{u,v} \leq x_u$

$y_{u,v} \leq x_v$

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

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

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

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

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

Greedy algorithm

Greedy algorithm for MMS problem

Greedy algorithm for MMS problem

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

Greedy algorithm for MMS problem

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

Greedy algorithm for MMS problem

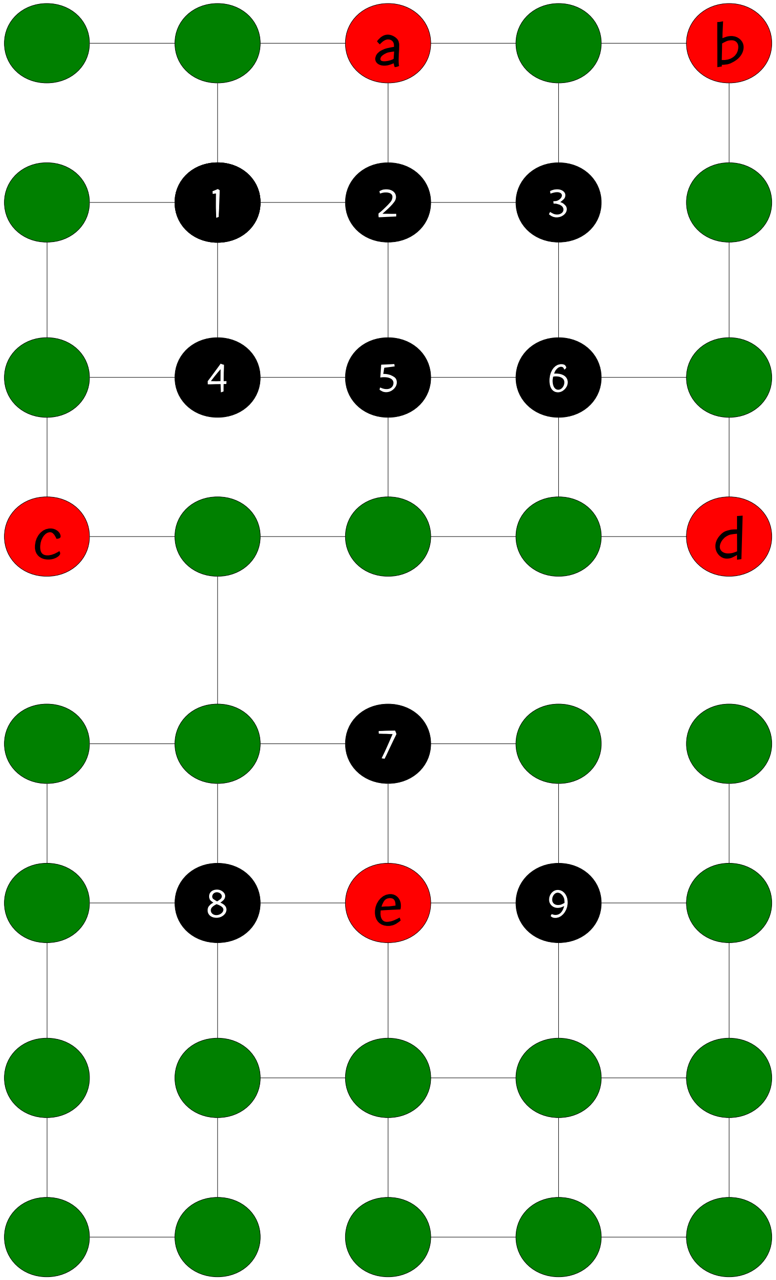
- initialize partial cover $S = \{ \}$
- while S is not a cover do:
  - find $m \in M \setminus S$ such that $S \cup \{m\}$ covers a maximum number of additional branch nodes (break ties by vertex index) and set $S = S \cup \{m\}$

Greedy algorithm for MMS problem

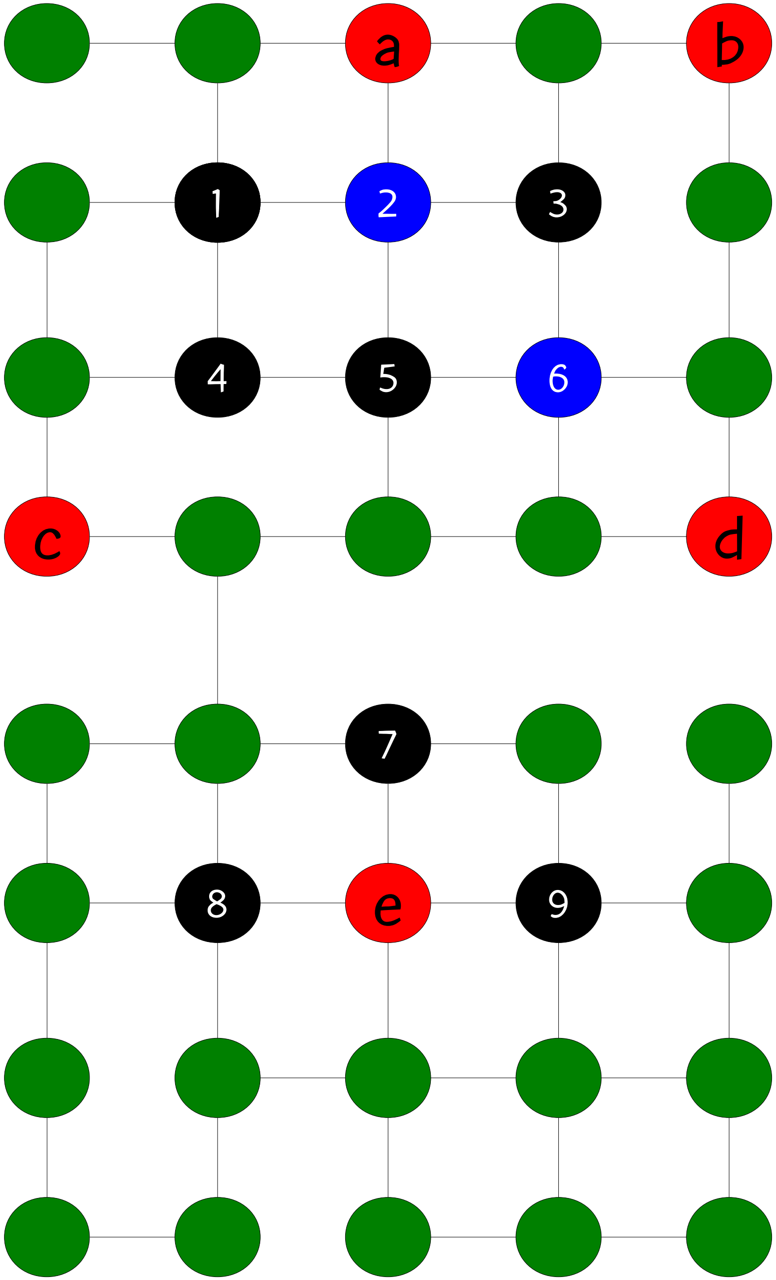
- initialize partial cover $S = \{\ \}$
- while S is not a cover do:
  - find $m \in M \setminus S$ such that $S \cup \{m\}$ covers a maximum number of additional branch nodes (break ties by vertex index) and set $S = S \cup \{m\}$
  - if no $m \in M \setminus S$ yields an increase in coverage, then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

Greedy algorithm for MMS problem

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

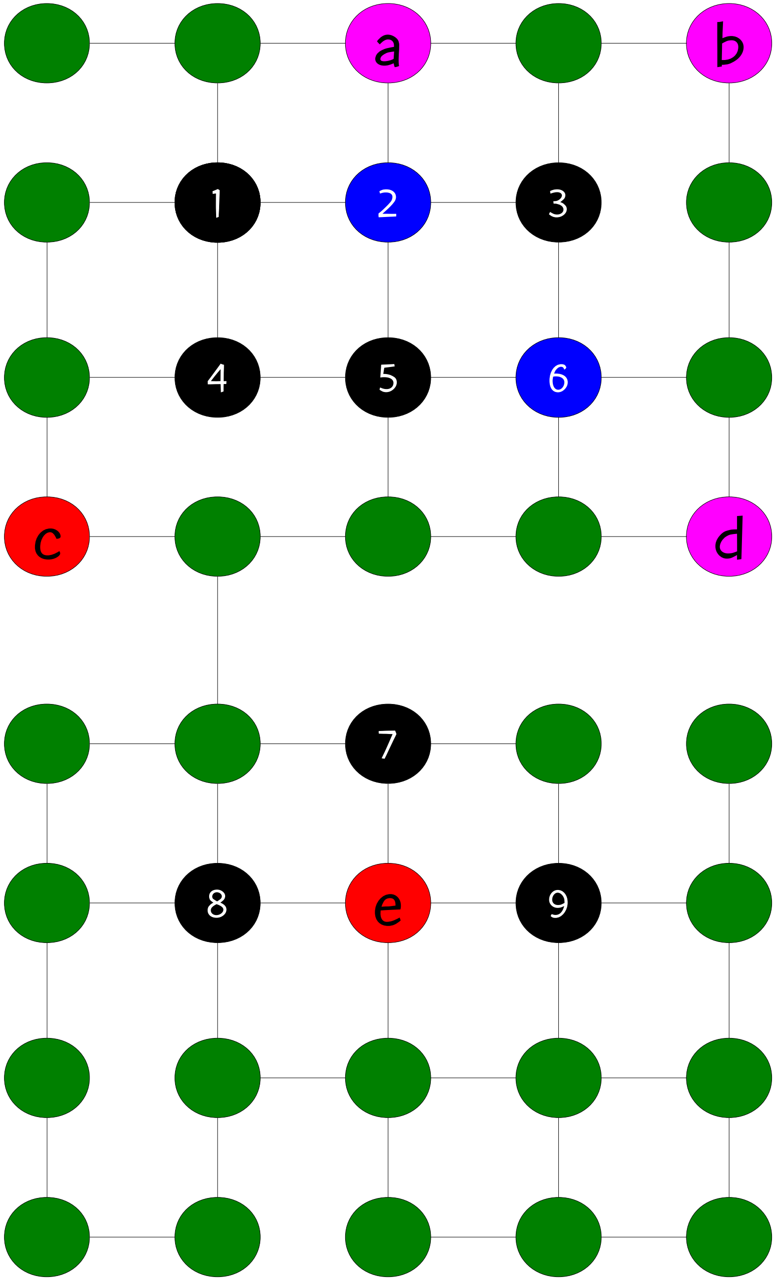


Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e



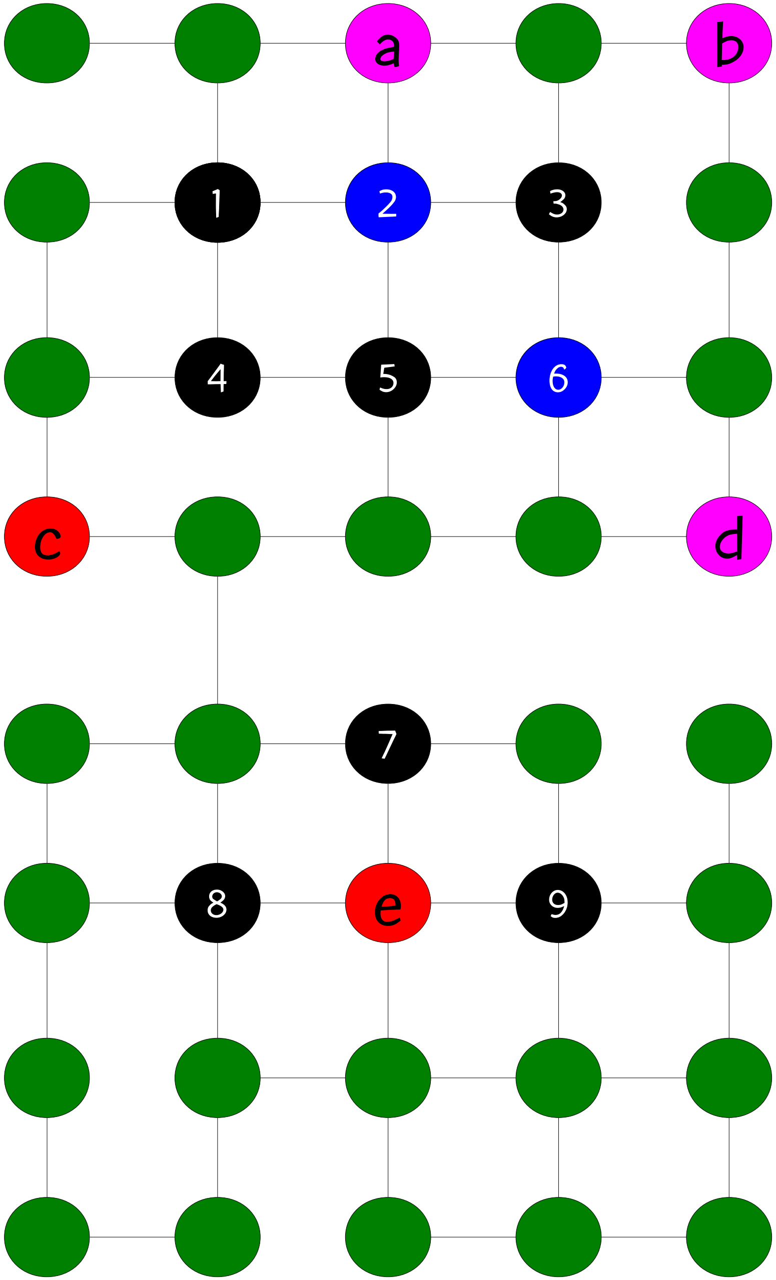
Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Greedy choice: {2, 6}



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

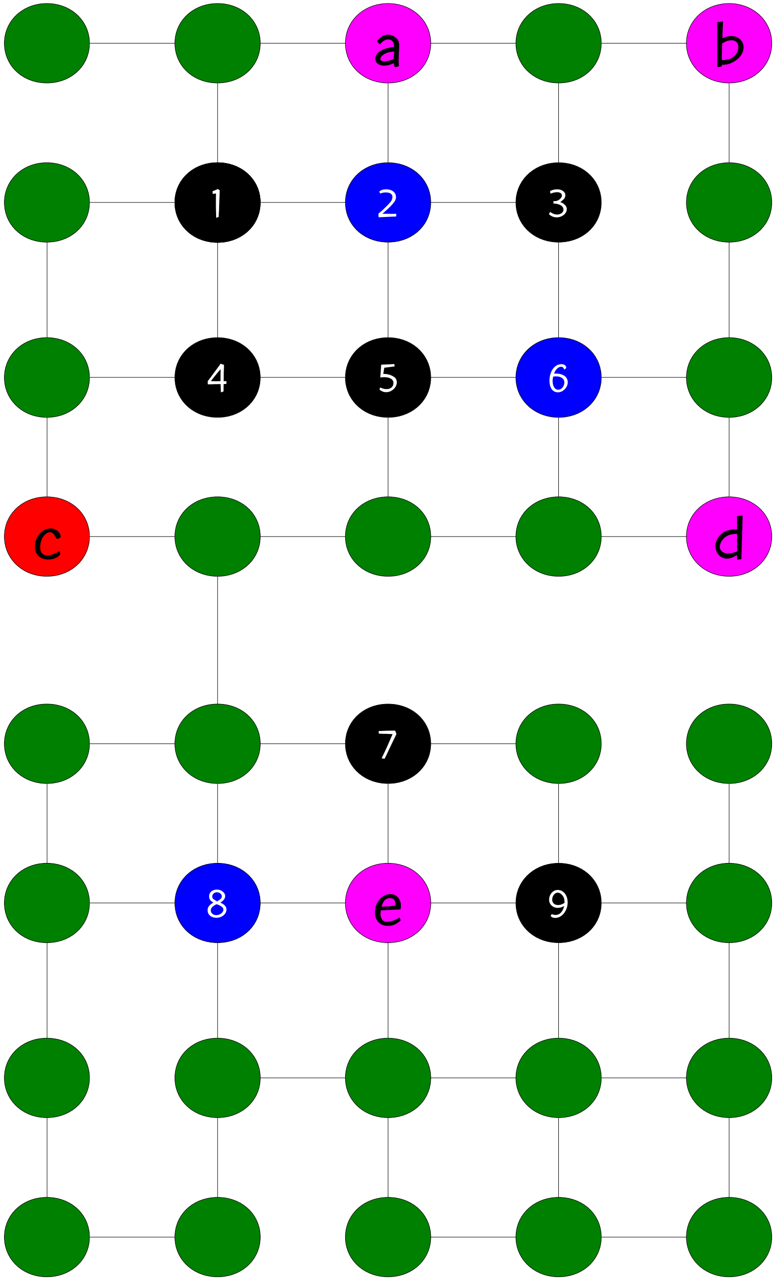
Greedy choice: {2, 6}



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Monitor	Add'l cover
1	none
3	none
4	none
5	none
7	none
8	e
9	none

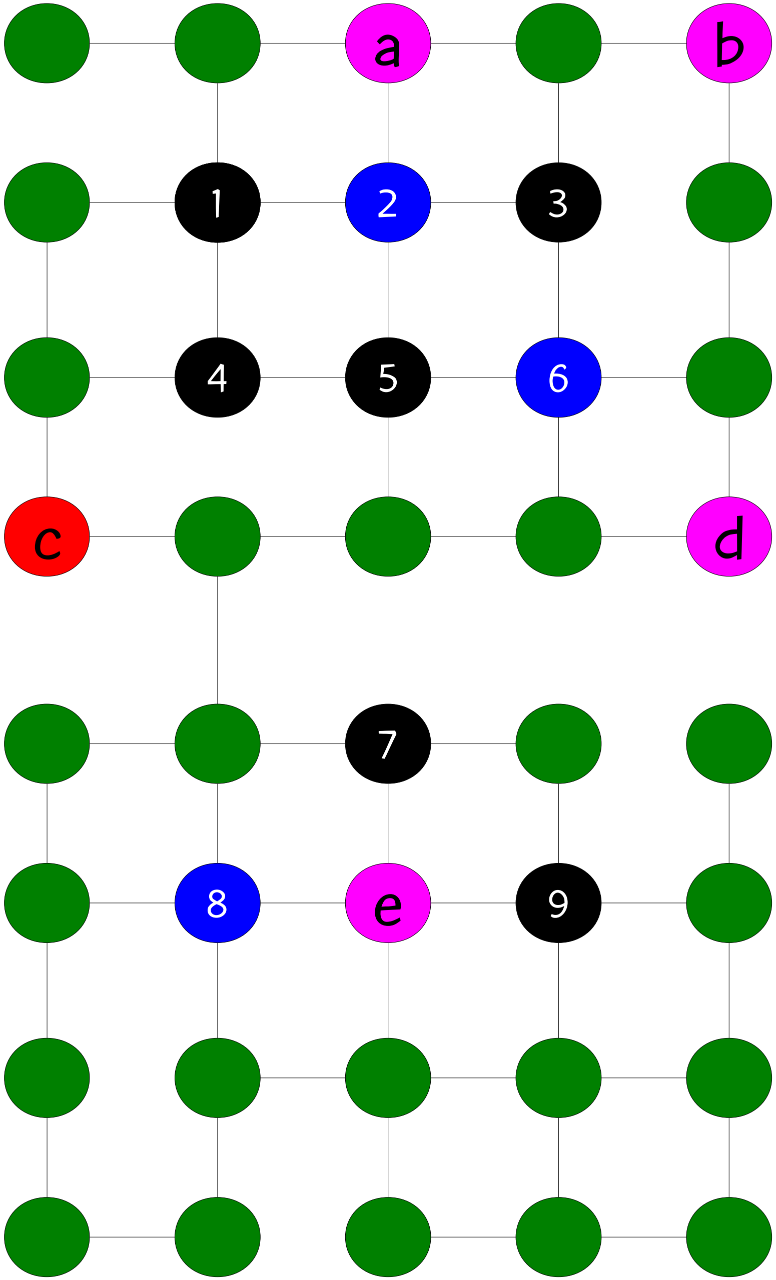
Greedy choice: {8}



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Monitor	Add'l cover
1	none
3	none
4	none
5	none
7	none
8	e
9	none

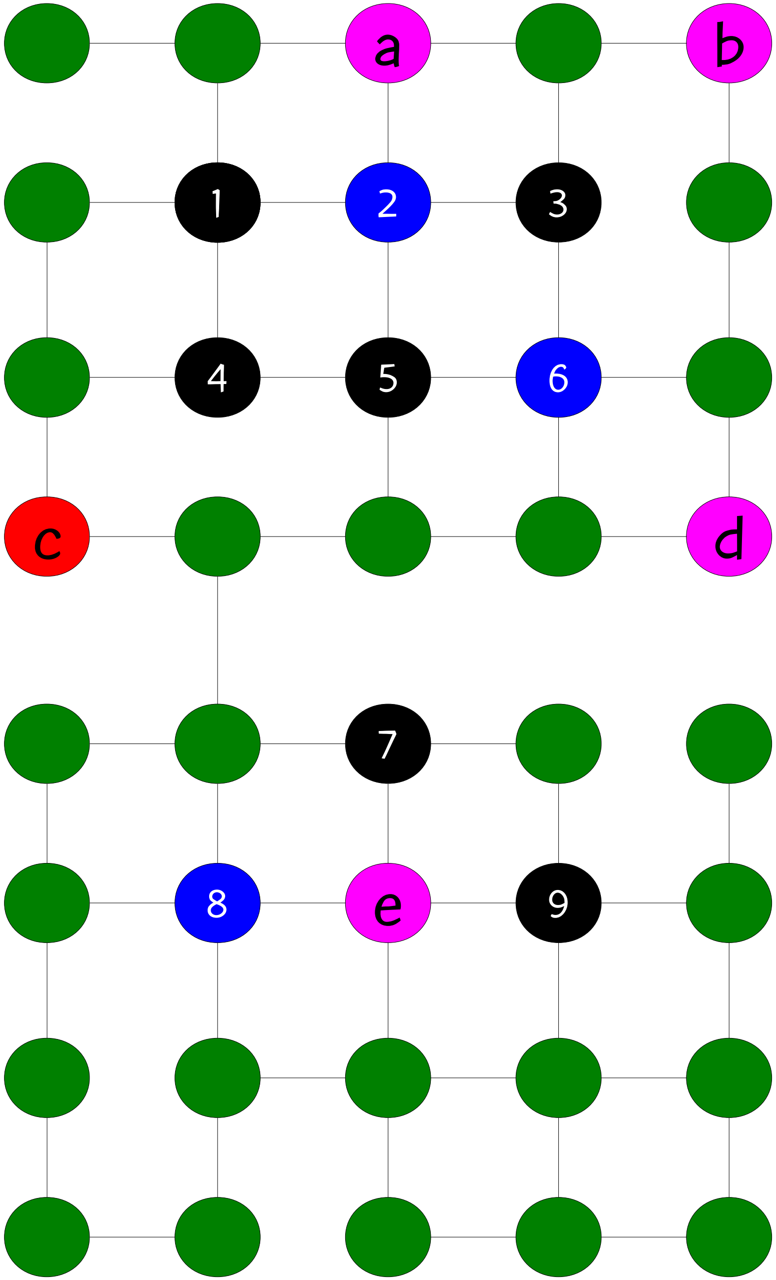
Greedy choice: {8}



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Monitor	Add'l cover
1	none
3	none
4	none
5	c
7	none
9	none

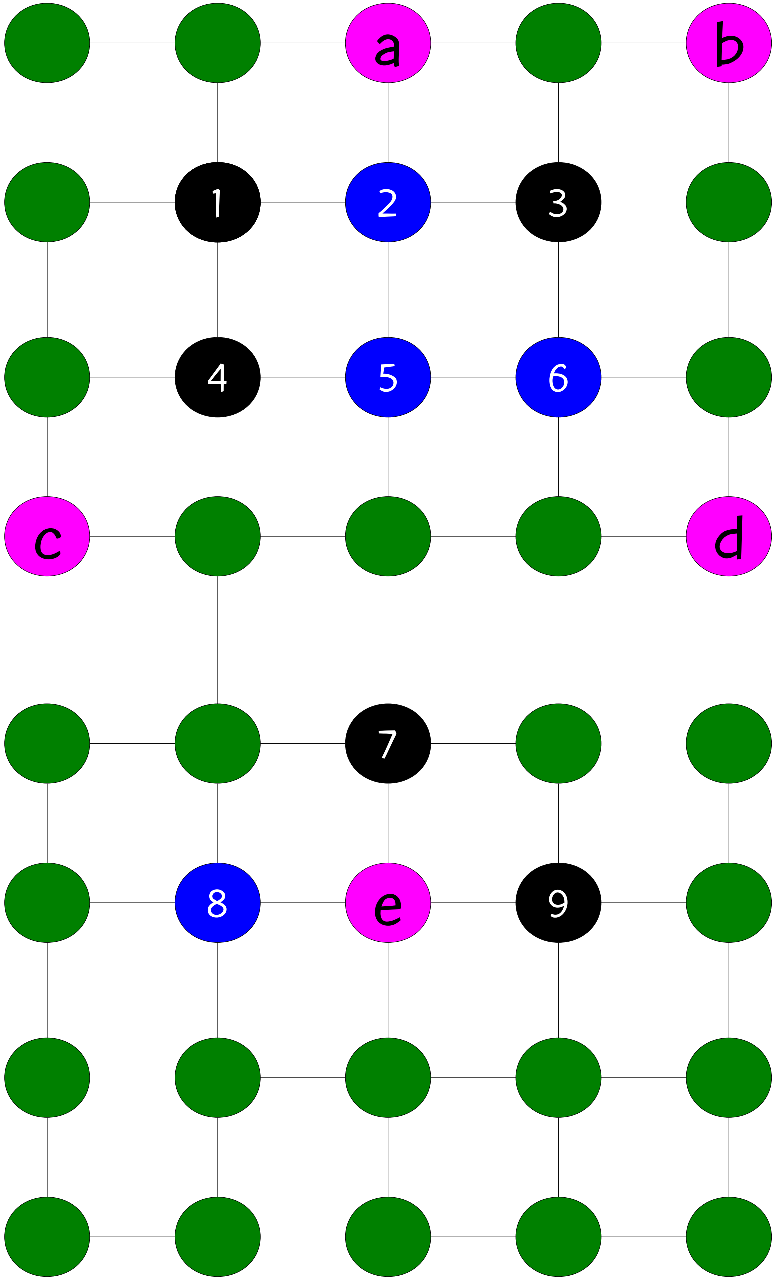
Node placement for monitoring



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Monitor	Add'l cover
1	none
3	none
4	none
5	c
7	none
9	none

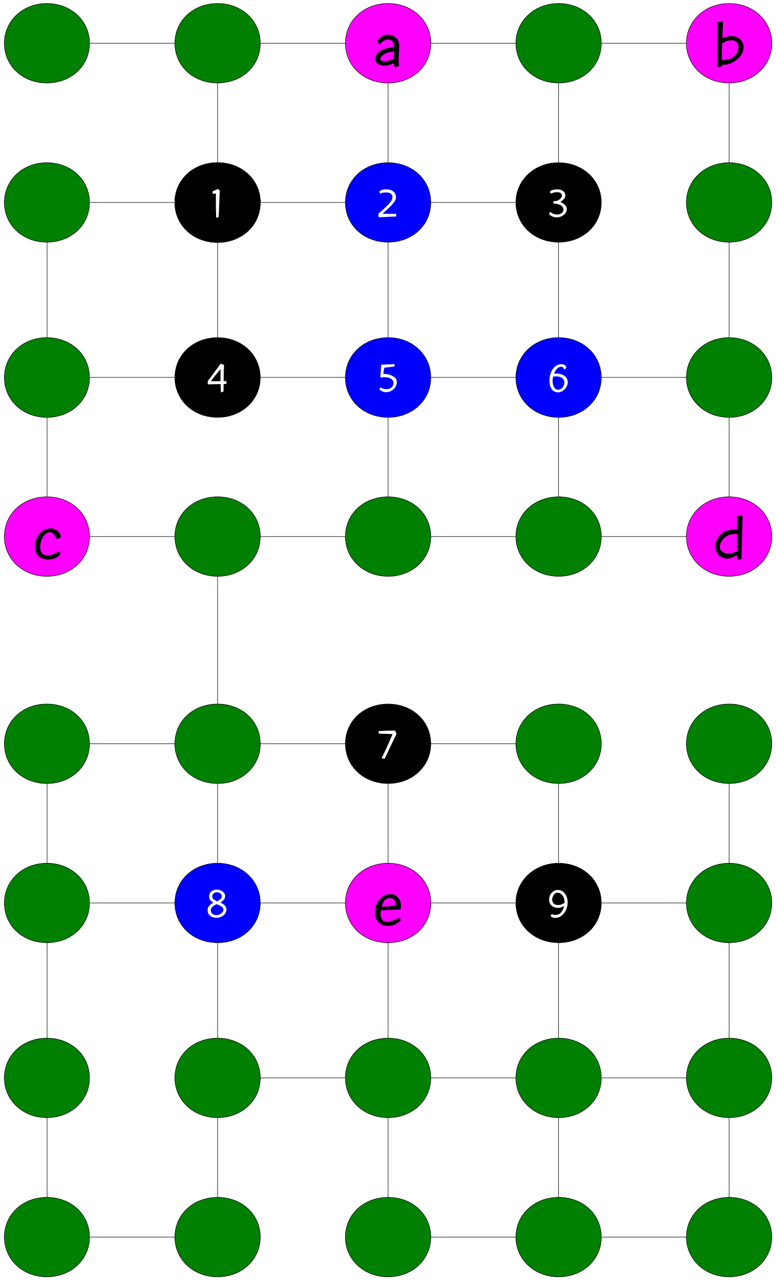
Greedy choice: {5}



Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Monitor	Add'l cover
1	none
3	none
4	none
5	c
7	none
9	none

Greedy choice: {5}



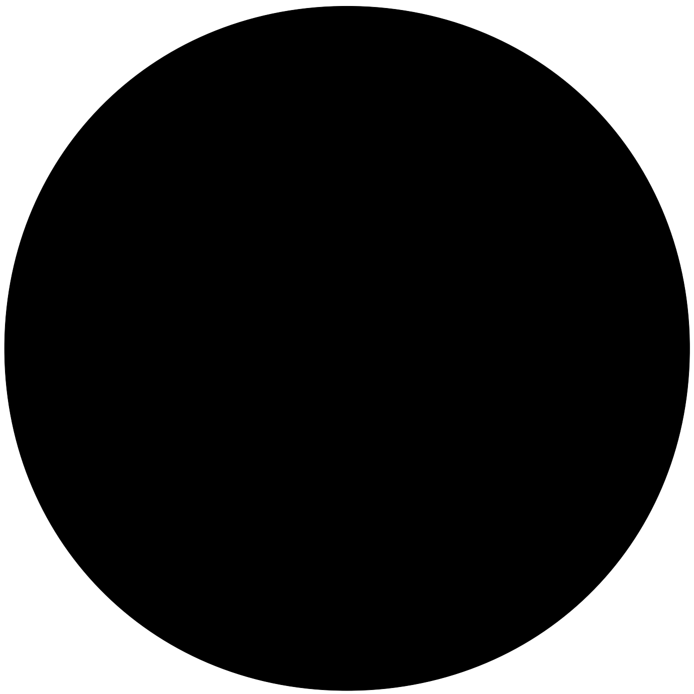
Pairs	Cover
2, 6	a, b, d
3, 6	b, d
1, 4	a, c
4, 7	c, e
5, 8	c, e
6, 8	e
8, 9	e

Solution: {2, 5, 6, 8} of size 4

Optimal solution!

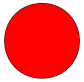
Genetic algorithm

Genetic algorithms Holland (1975)

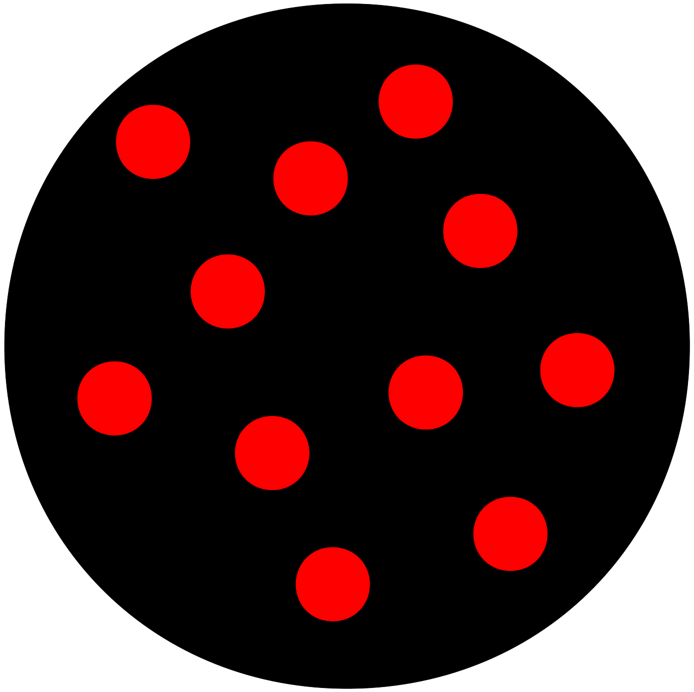


Adaptive methods that are used to solve search and optimization problems.

Individual: solution



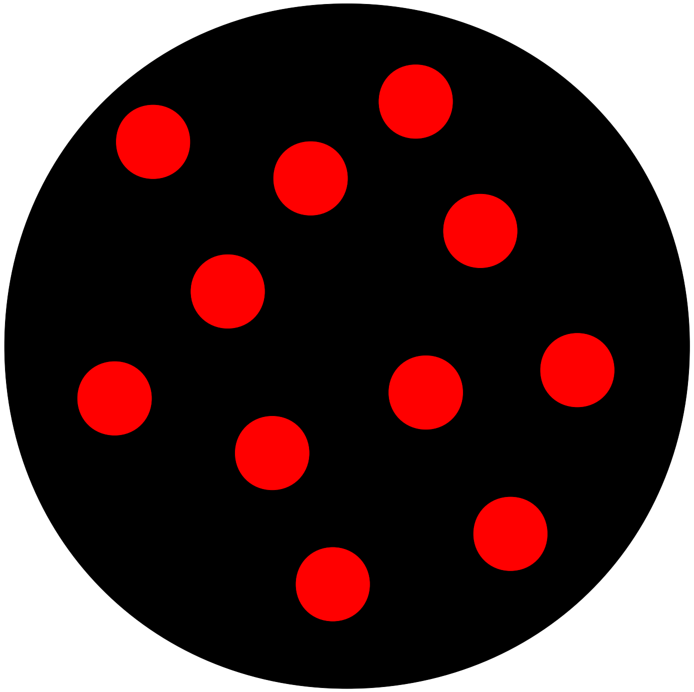
Genetic algorithms



Individual: solution

Population: set of fixed number of individuals

Genetic algorithms

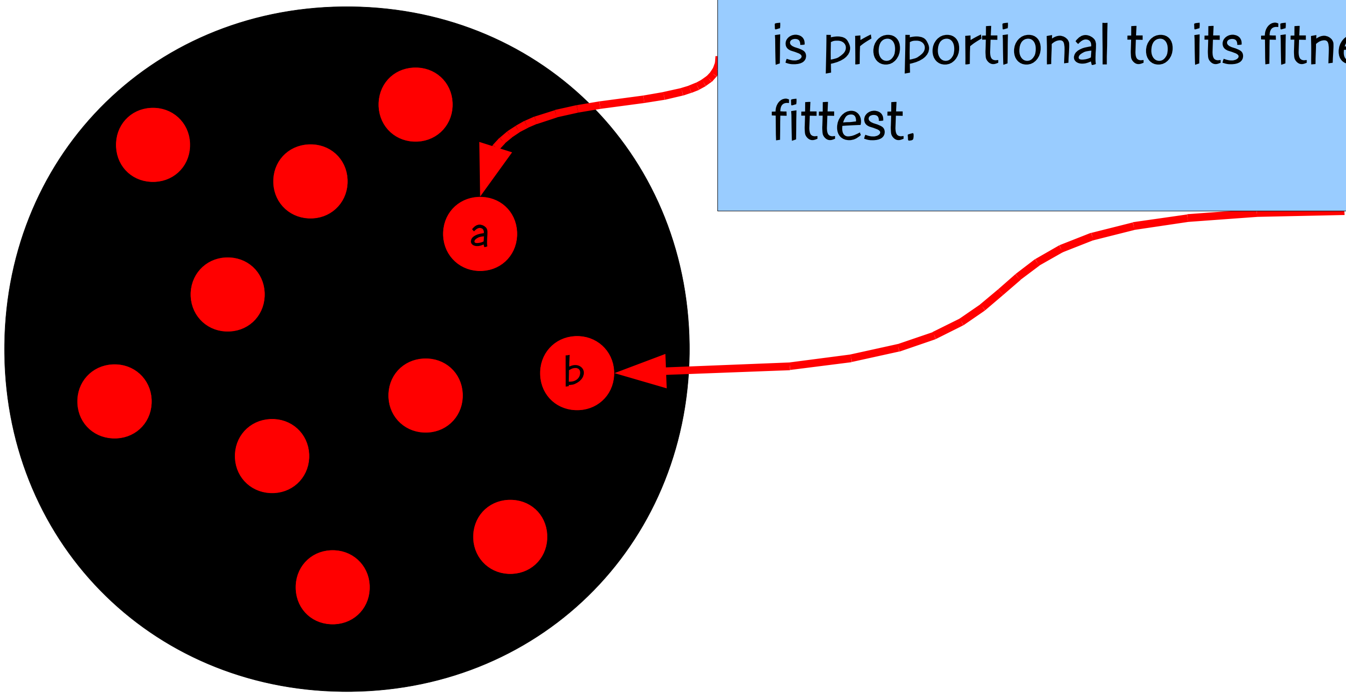


Genetic algorithms evolve population applying the principle of survival of the fittest.

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

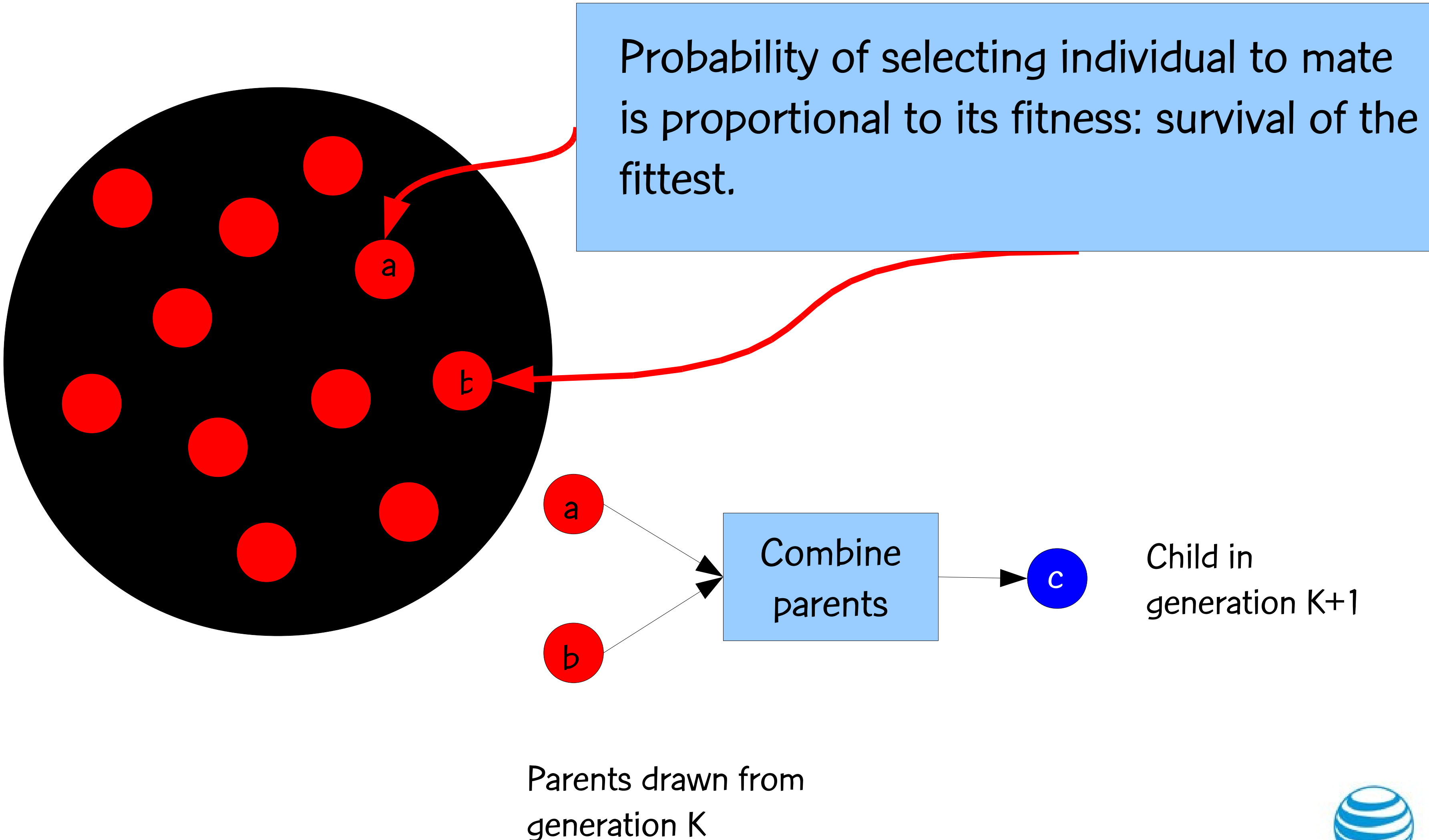
Individuals from one generation are combined to produce offspring that make up next generation.

Genetic algorithms

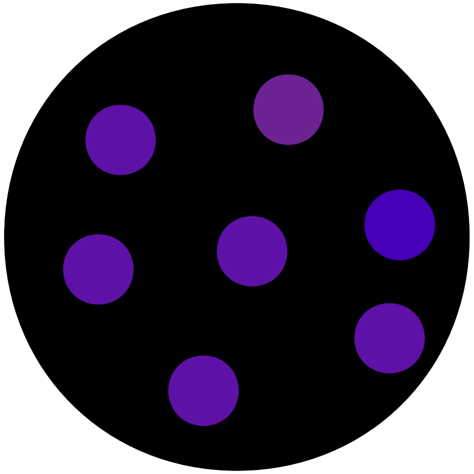


Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.

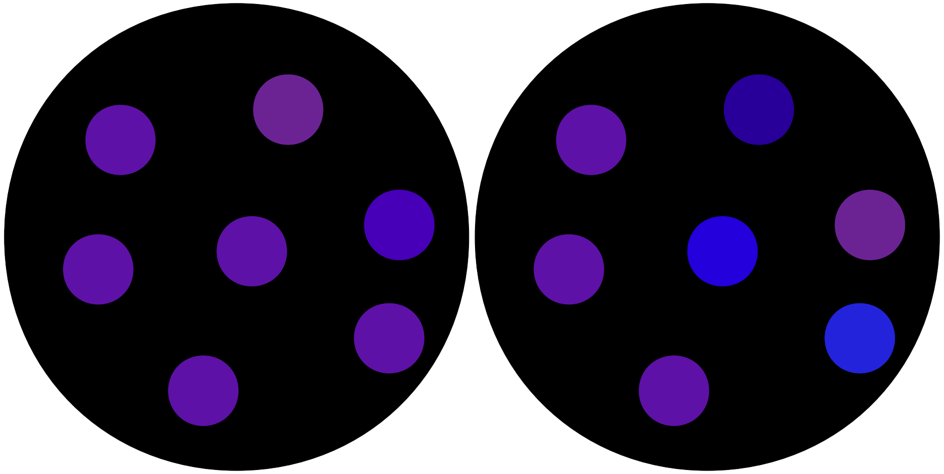
Genetic algorithms



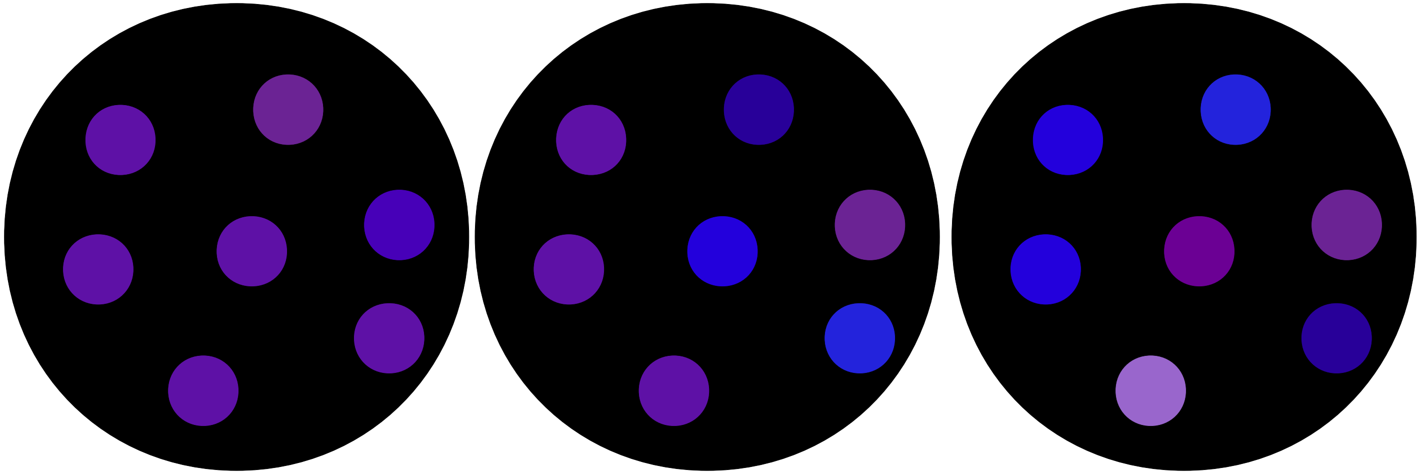
Evolution of solutions



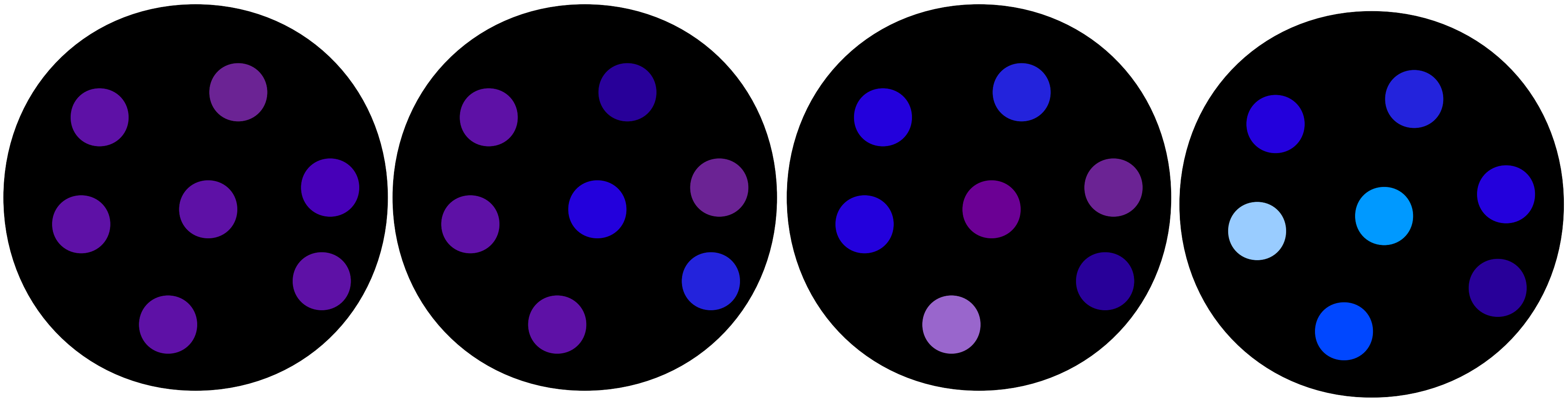
Evolution of solutions



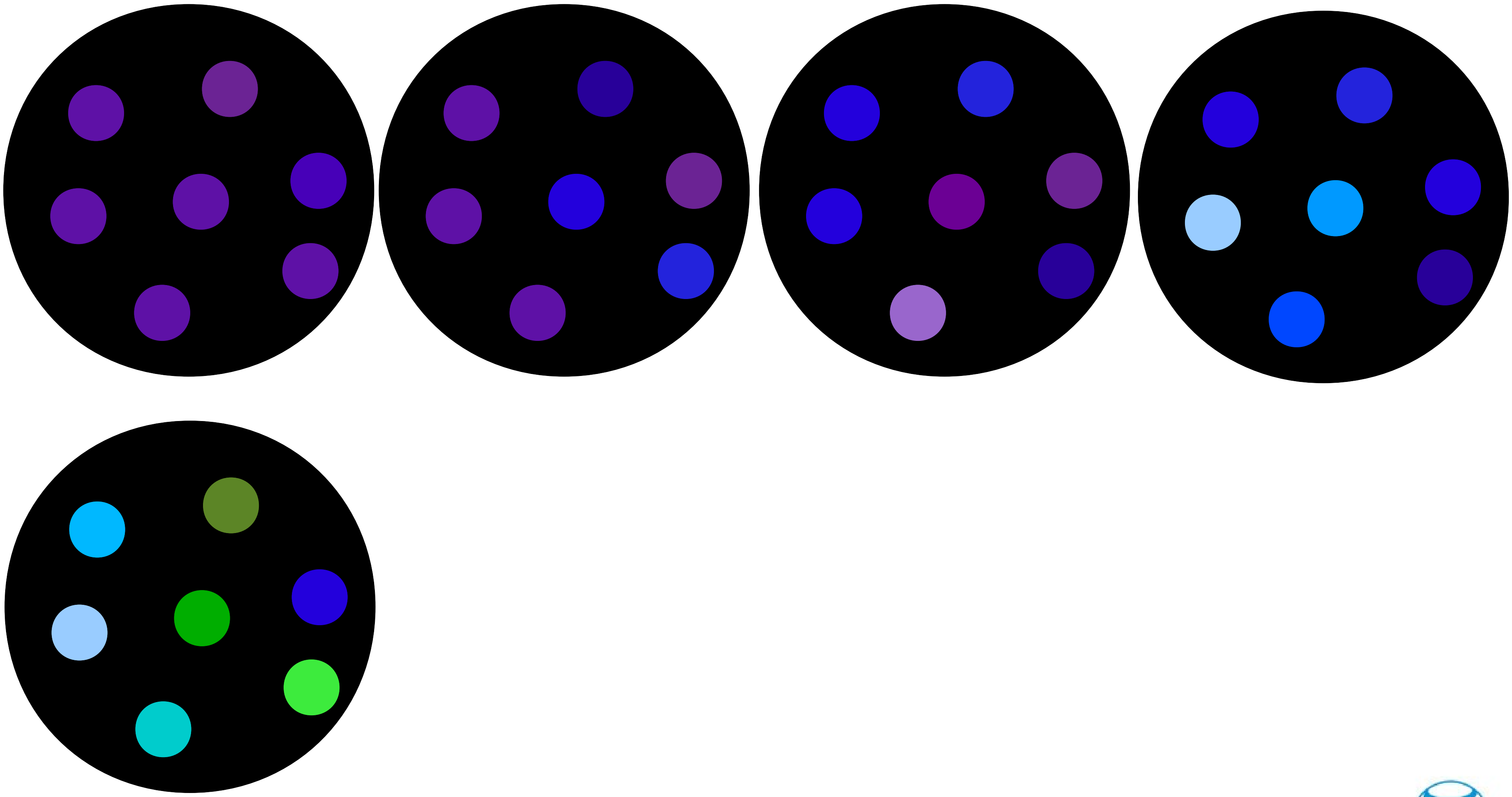
Evolution of solutions



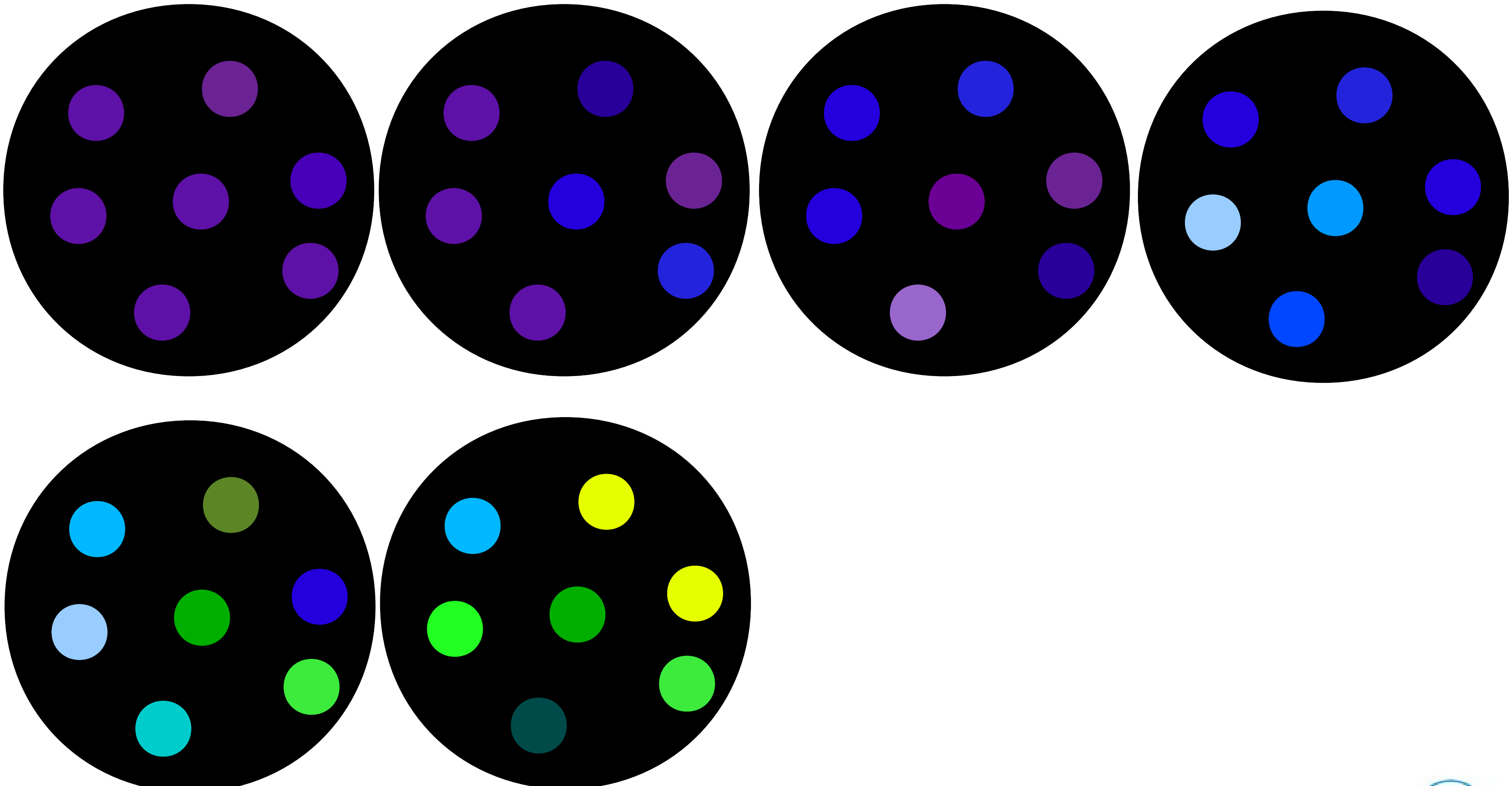
Evolution of solutions



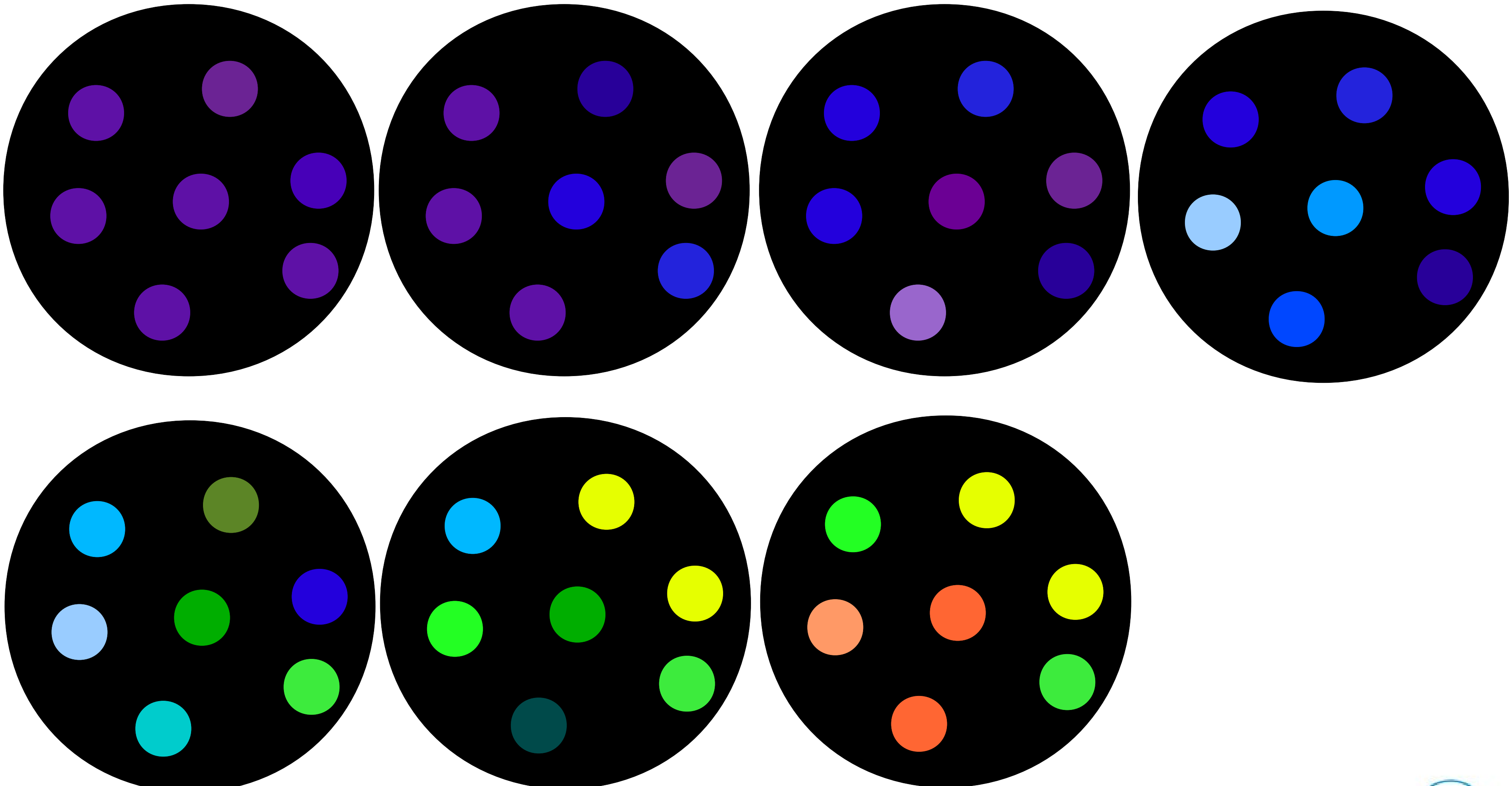
Evolution of solutions



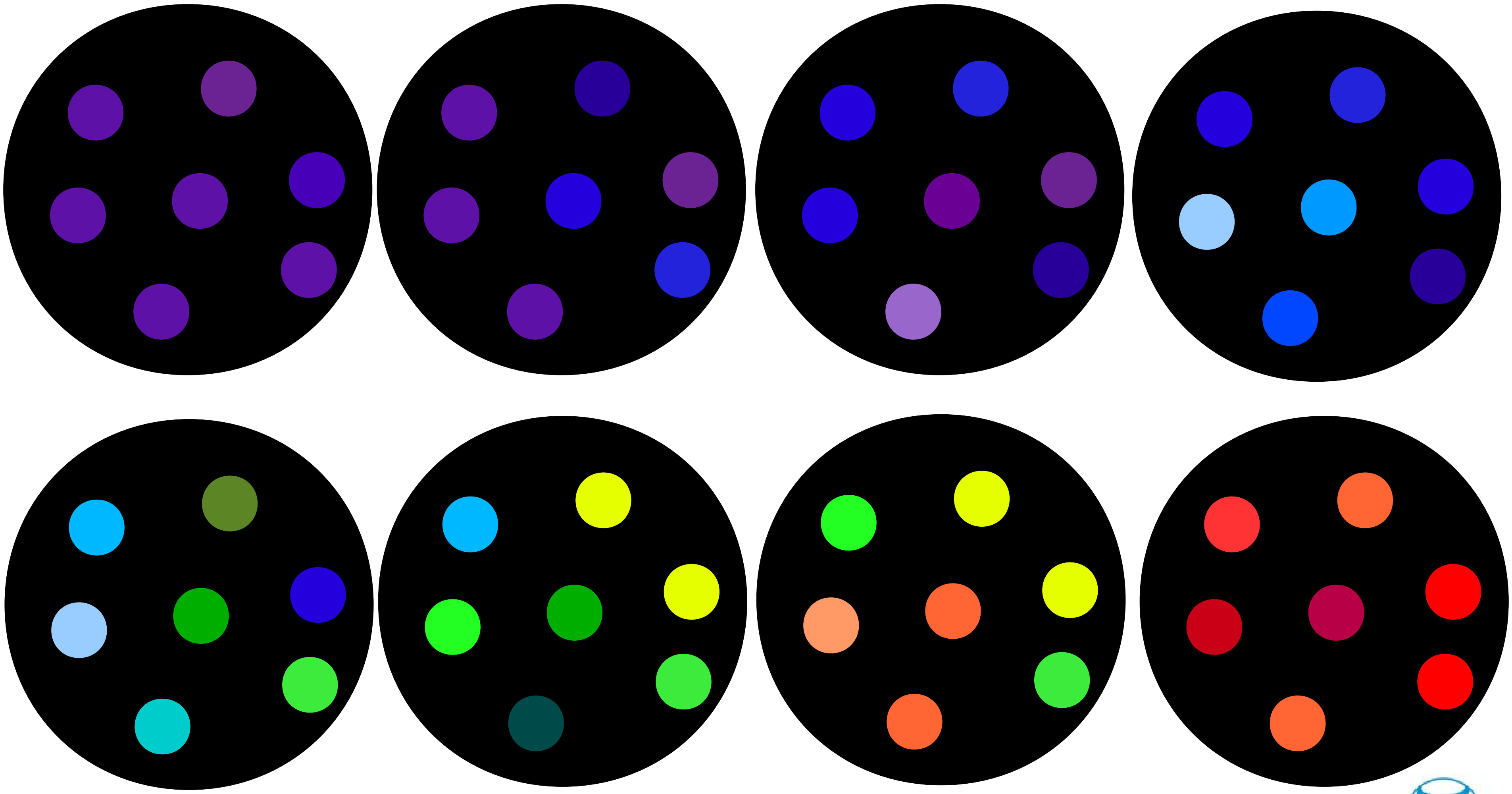
Evolution of solutions



Evolution of solutions



Evolution of solutions



Genetic algorithms with random keys

GAs and random keys

- Introduced by Bean (1994) for sequencing problems.

GAs and random keys

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

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

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

GAs and random keys

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

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

- Introduced by Bean (1994) for sequencing problems.
- 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed 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

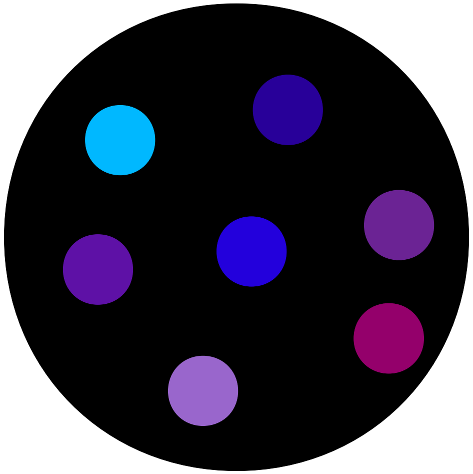
- Introduced by Bean (1994) for sequencing problems.
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passed the allele to the child.

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.

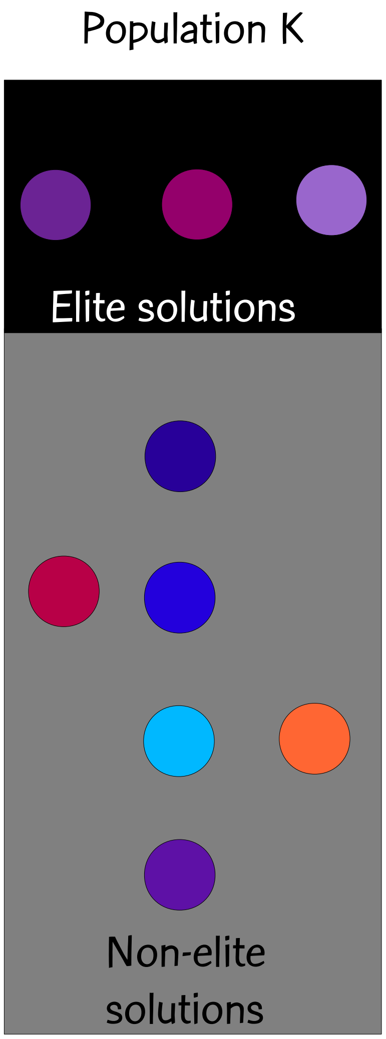
GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- Initial population is made up of P chromosomes, each with N genes, each having a value (allele) generated uniformly at random in the interval [0,1].



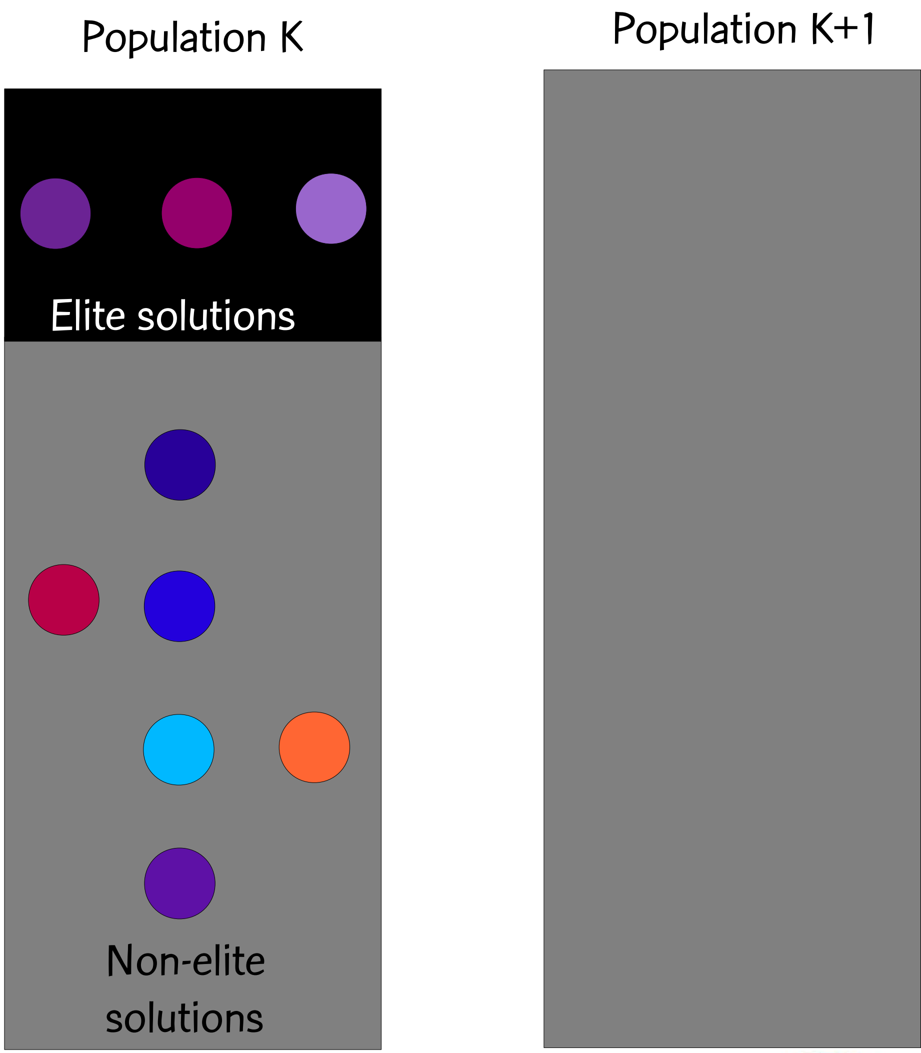
GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- At the K-th generation, compute the cost of each solution and partition the solutions into two sets: elite solutions, non-elite solutions. Elite set should be smaller of the two sets and contain best solutions.



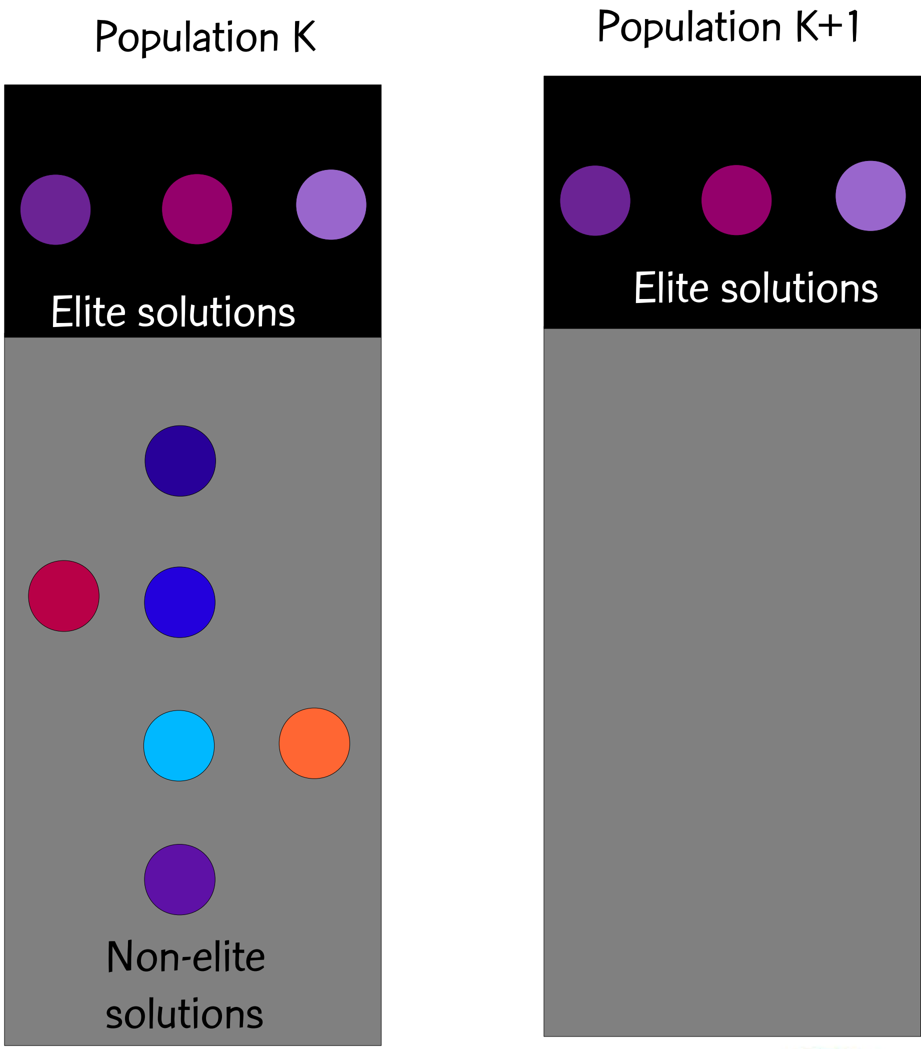
GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- Evolutionary dynamics



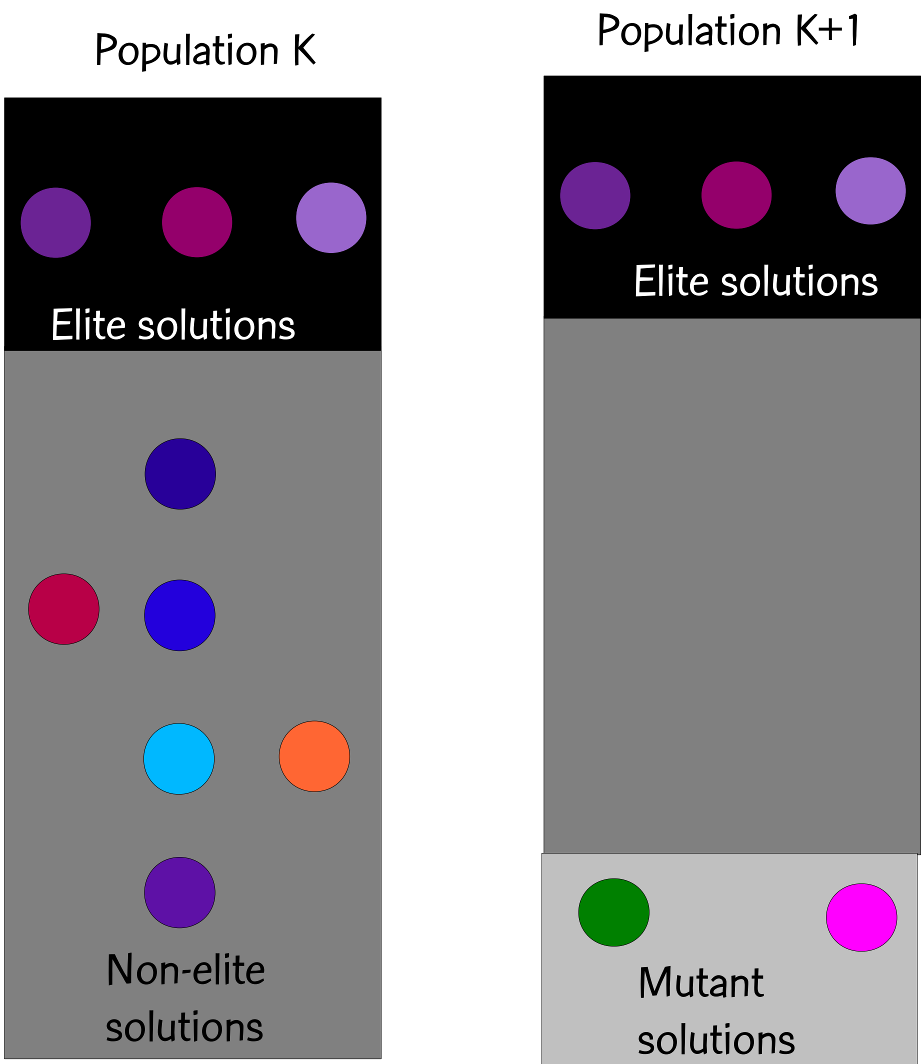
GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- Evolutionary dynamics
  - Copy elite solutions from population K to population K+1



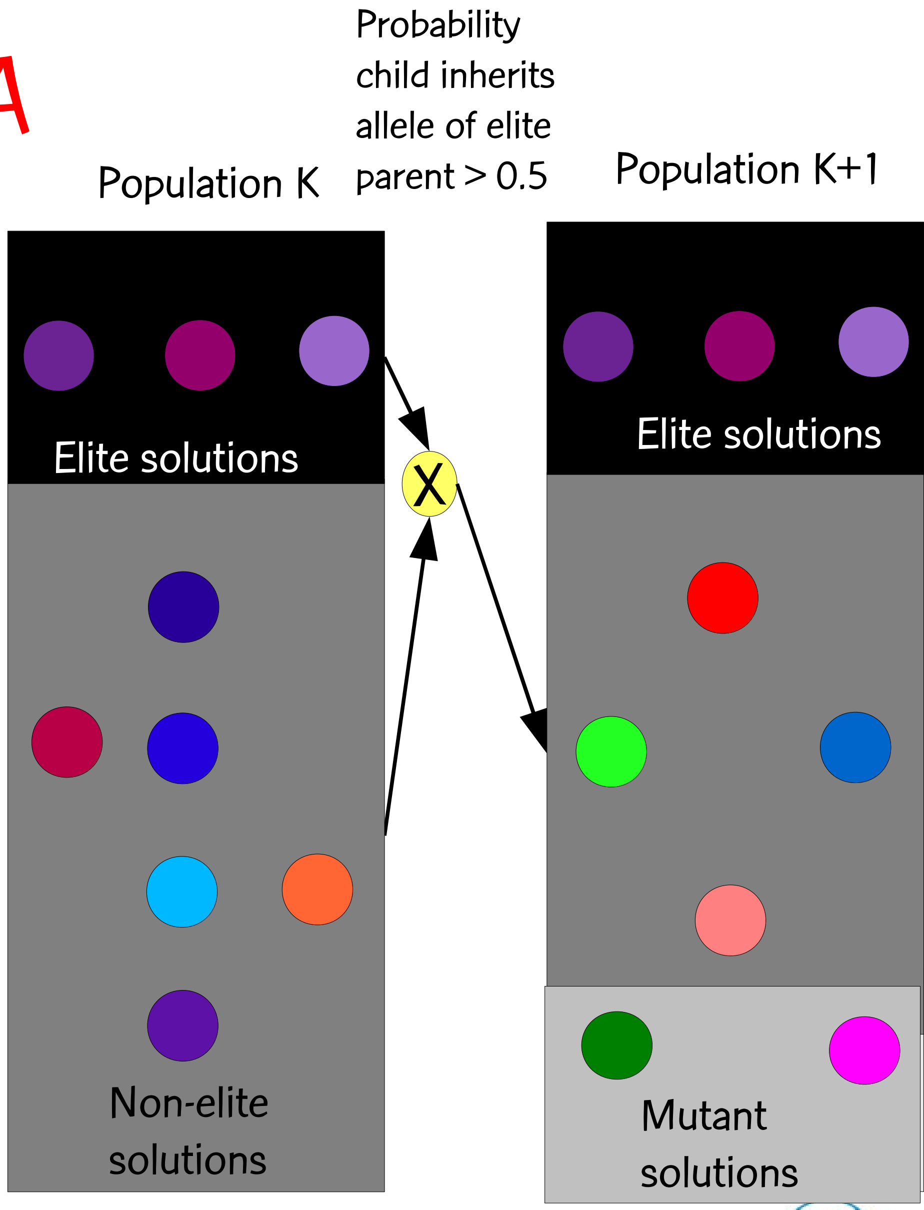
GAs and random keys

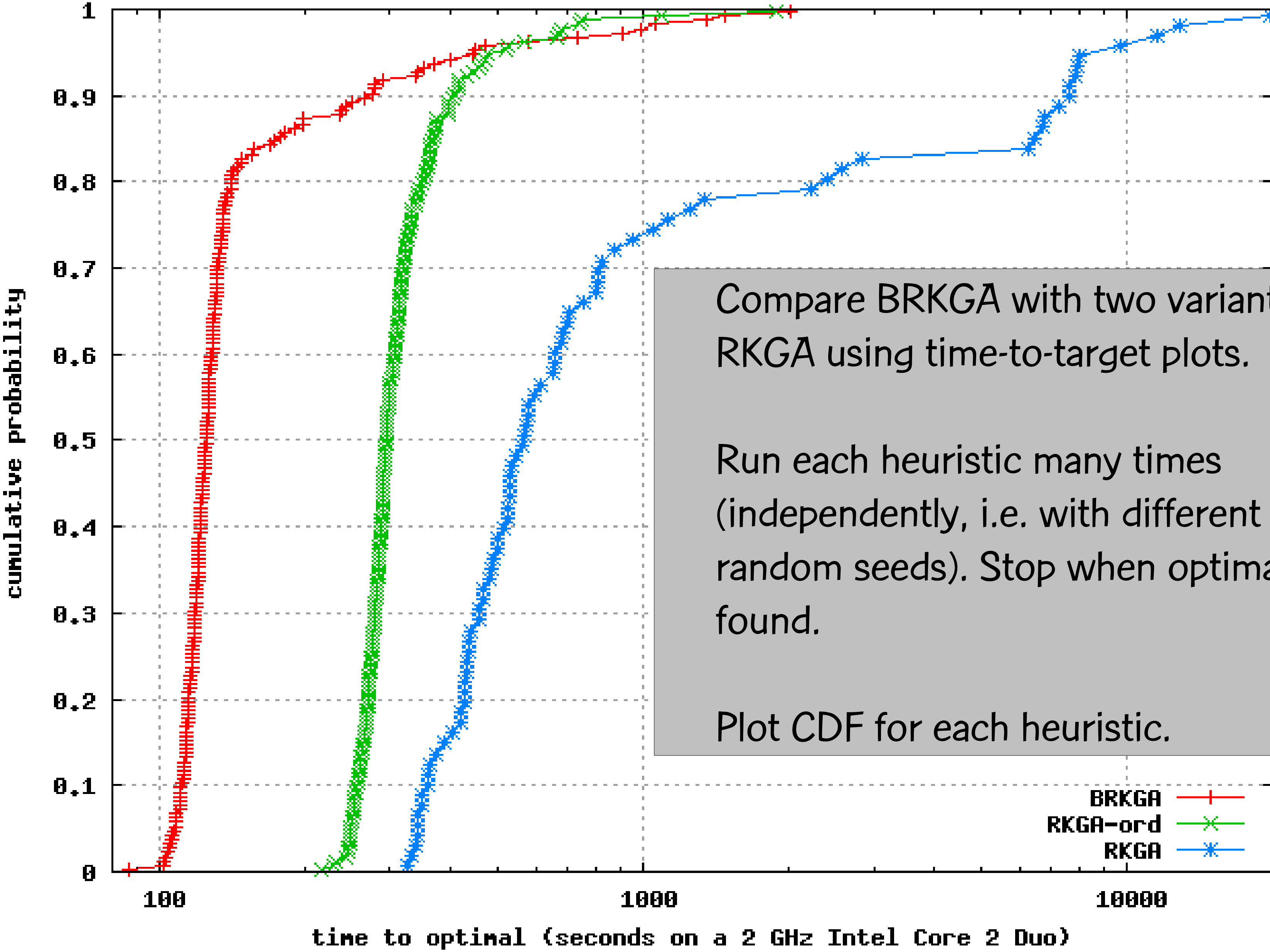
- Introduced by Bean (1994) for sequencing problems.
- Evolutionary dynamics
  - Copy elite solutions from population K to population K+1
  - Add R random solutions (mutants) to population K+1



Biased random key GA

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



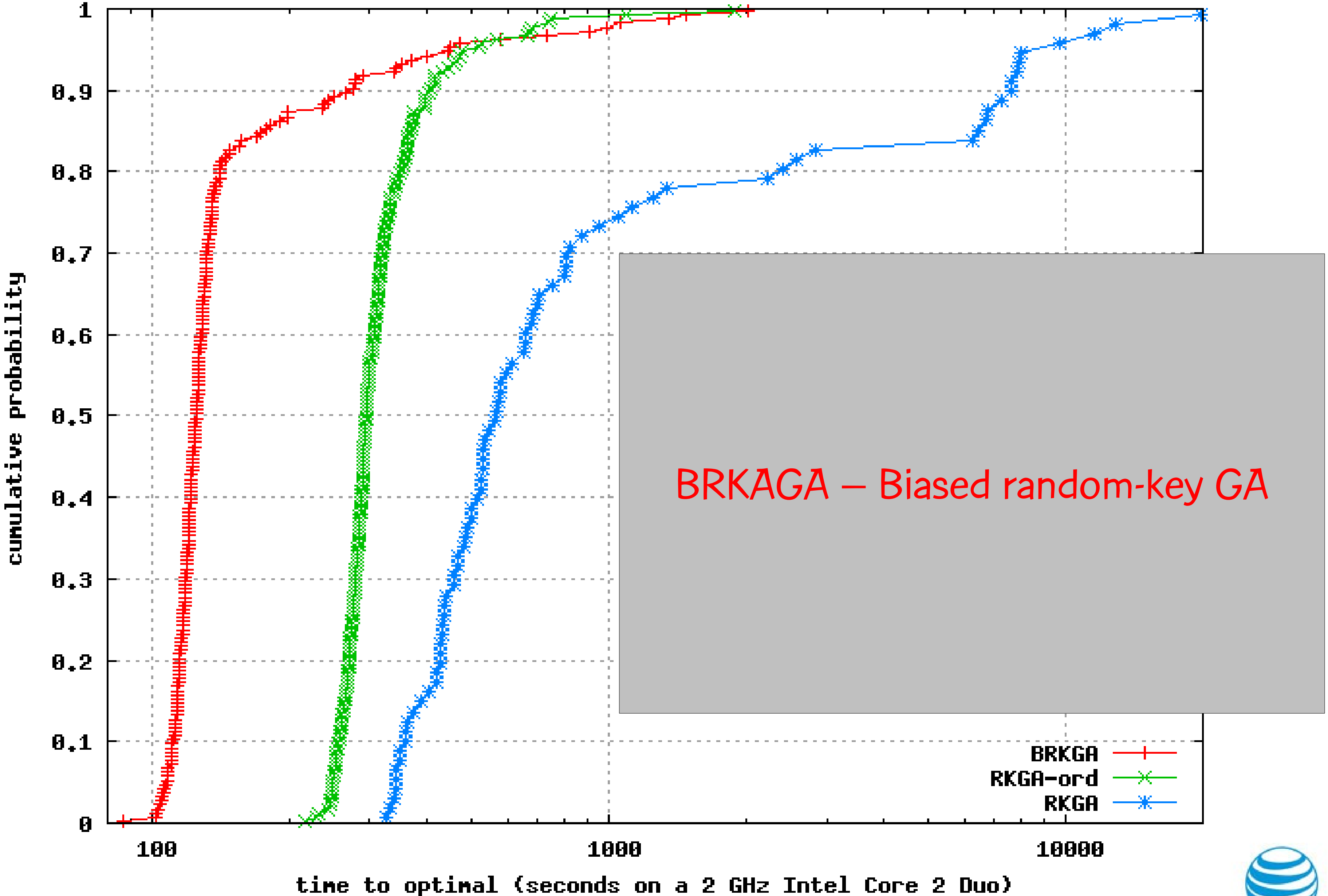


Compare BRKGA with two variants of RKGA using time-to-target plots.

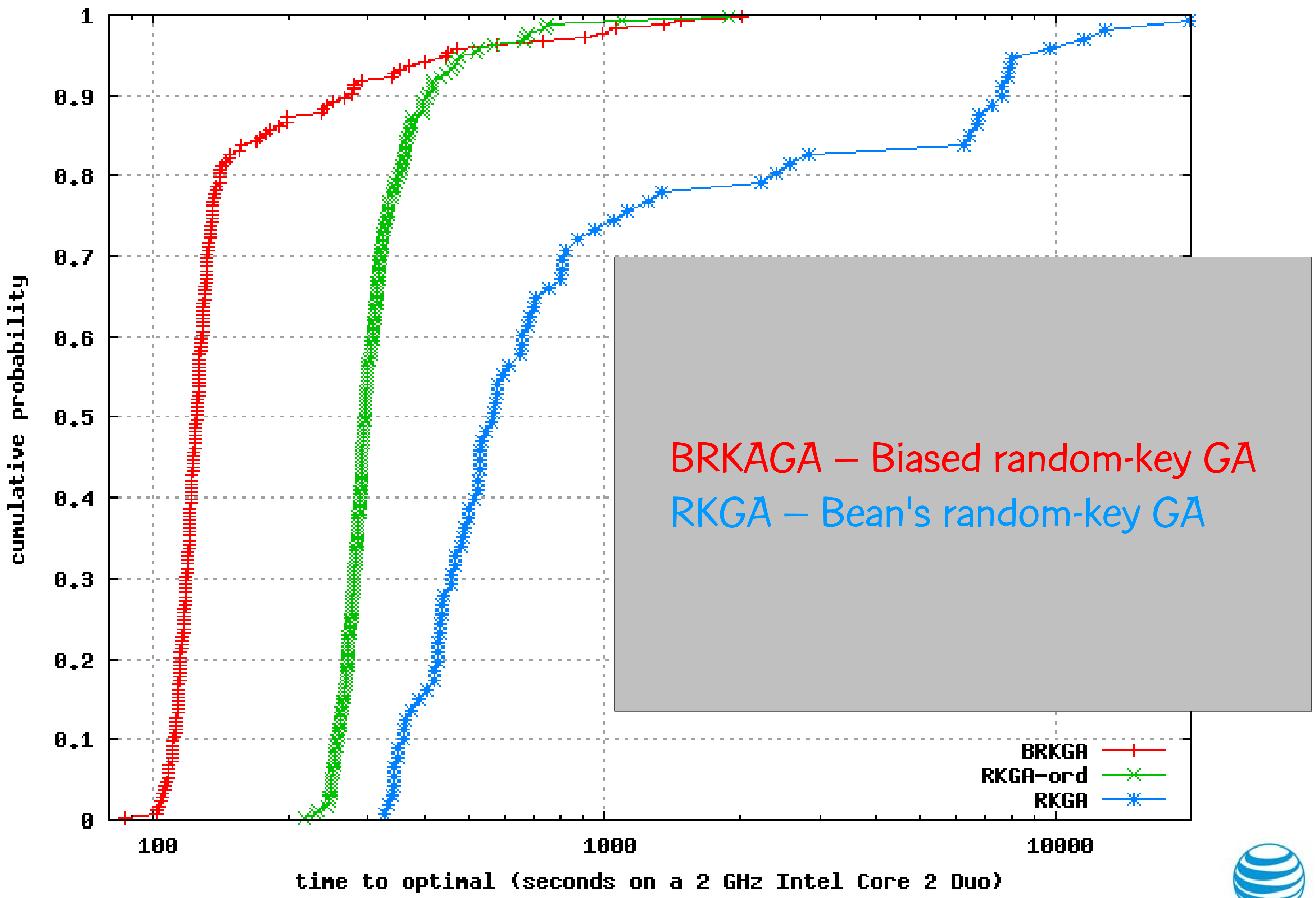
Run each heuristic many times (independently, i.e. with different random seeds). Stop when optimal is found.

Plot CDF for each heuristic.

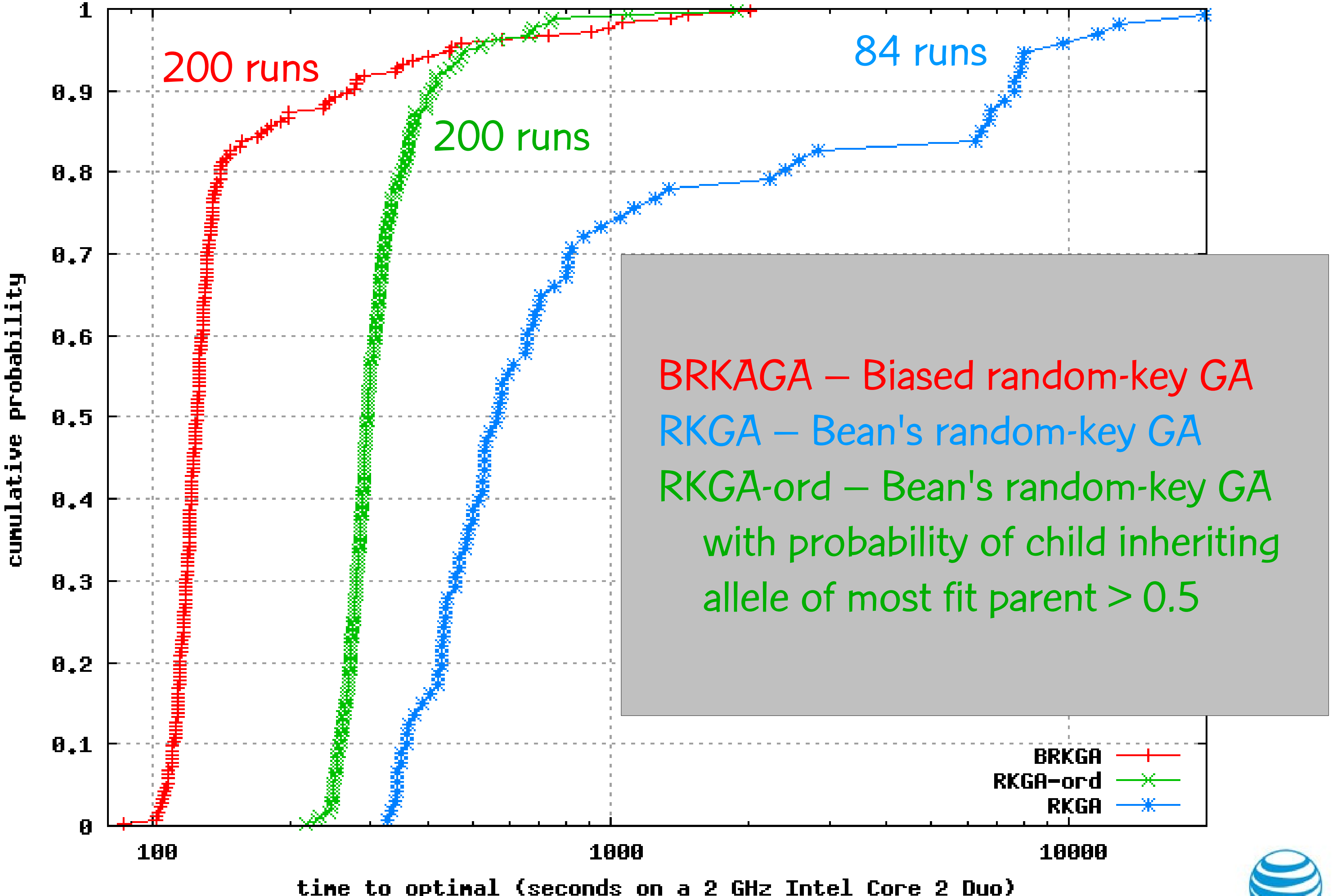
at&t
Your world. Delivered.



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

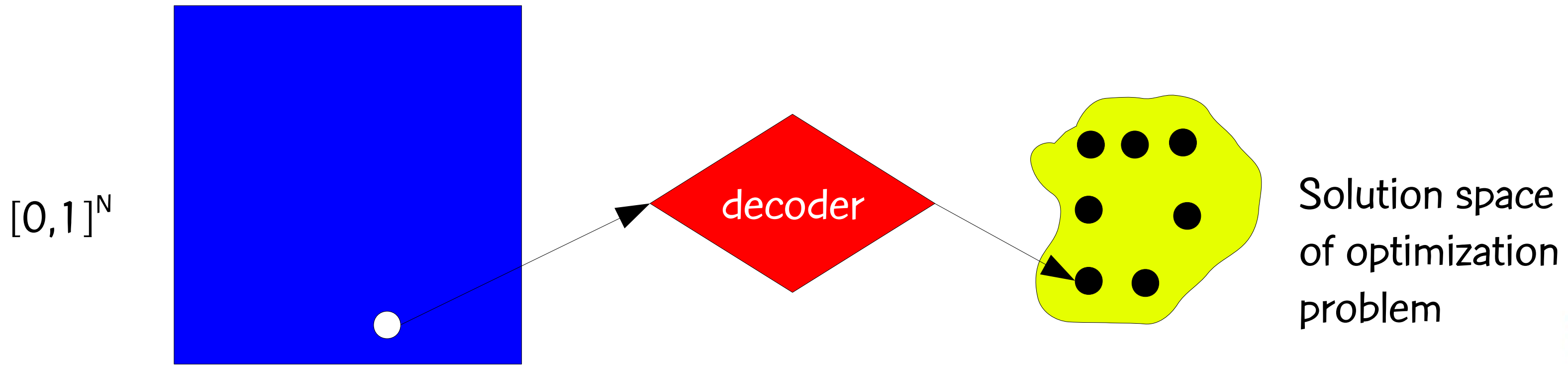

BRKAGA – Biased random-key GA
RKGA – Bean's random-key GA
cumulative probability
time to optimal (seconds on a 2 GHz Intel Core 2 Duo)
BRKGA
RKGA-ord
RKGA
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
100
1000
10000

220 node example



Decoders

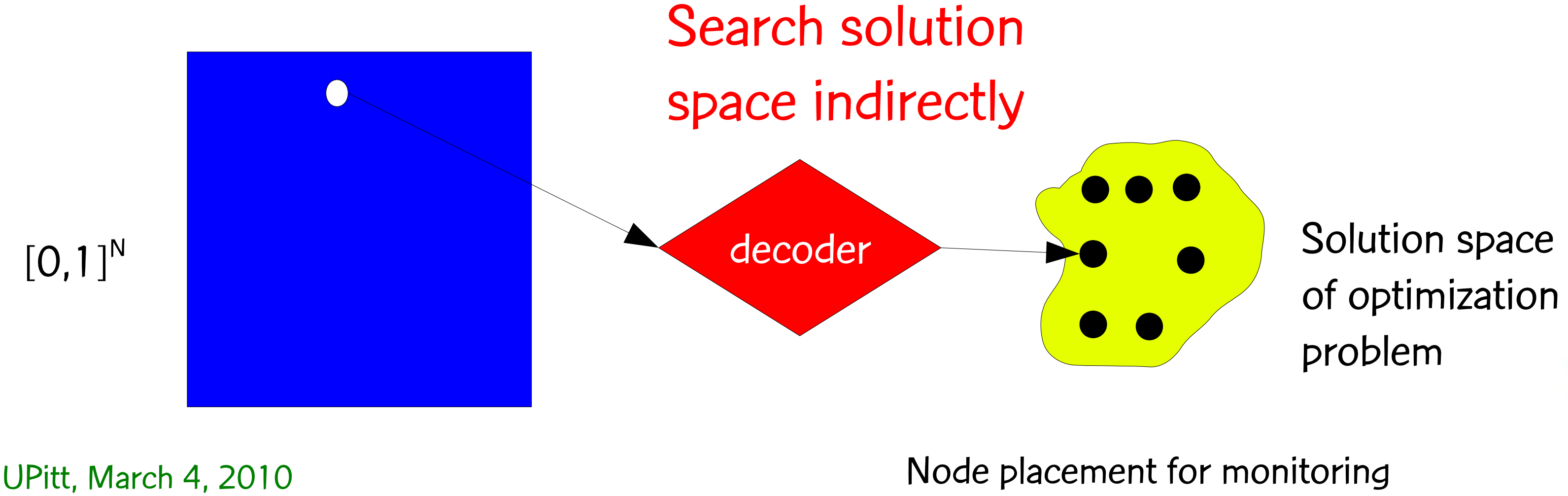
- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly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



Node placement for monitoring

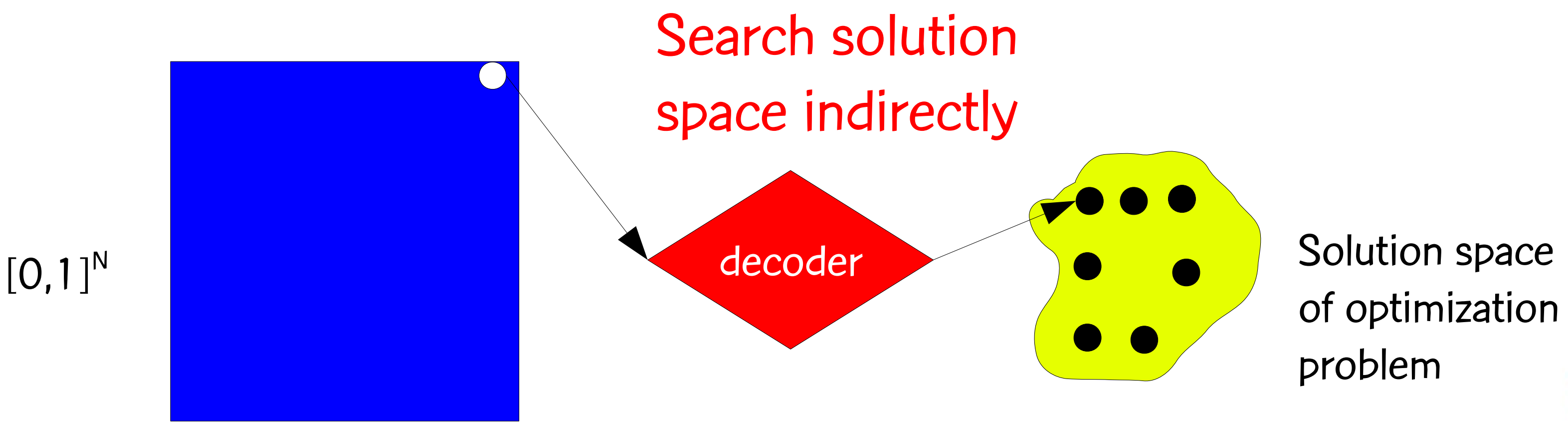
Decoders

- 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

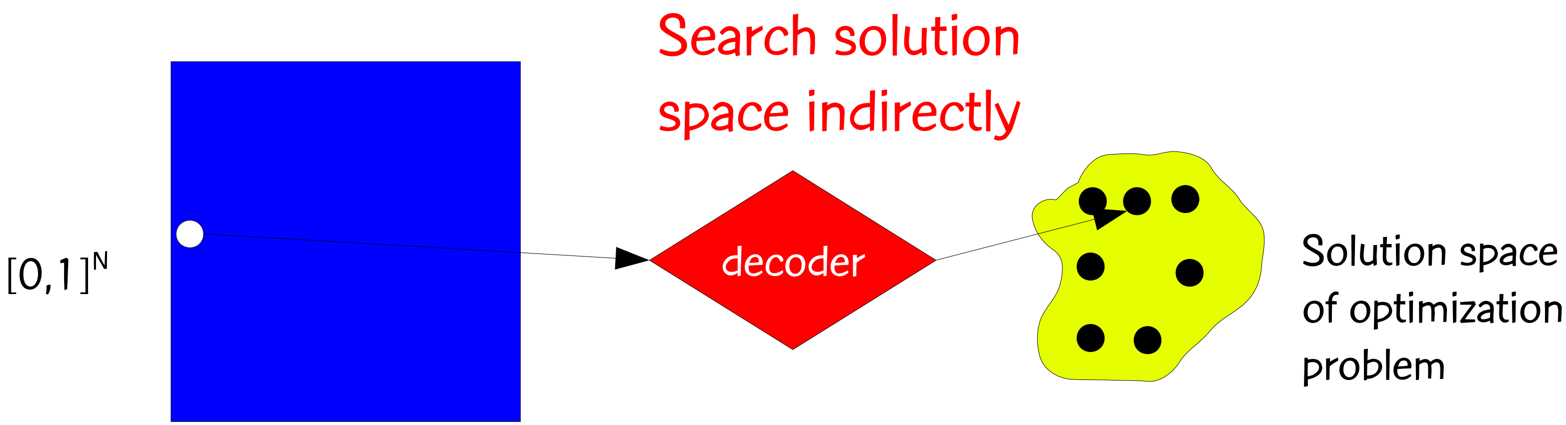
- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly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



Node placement for monitoring

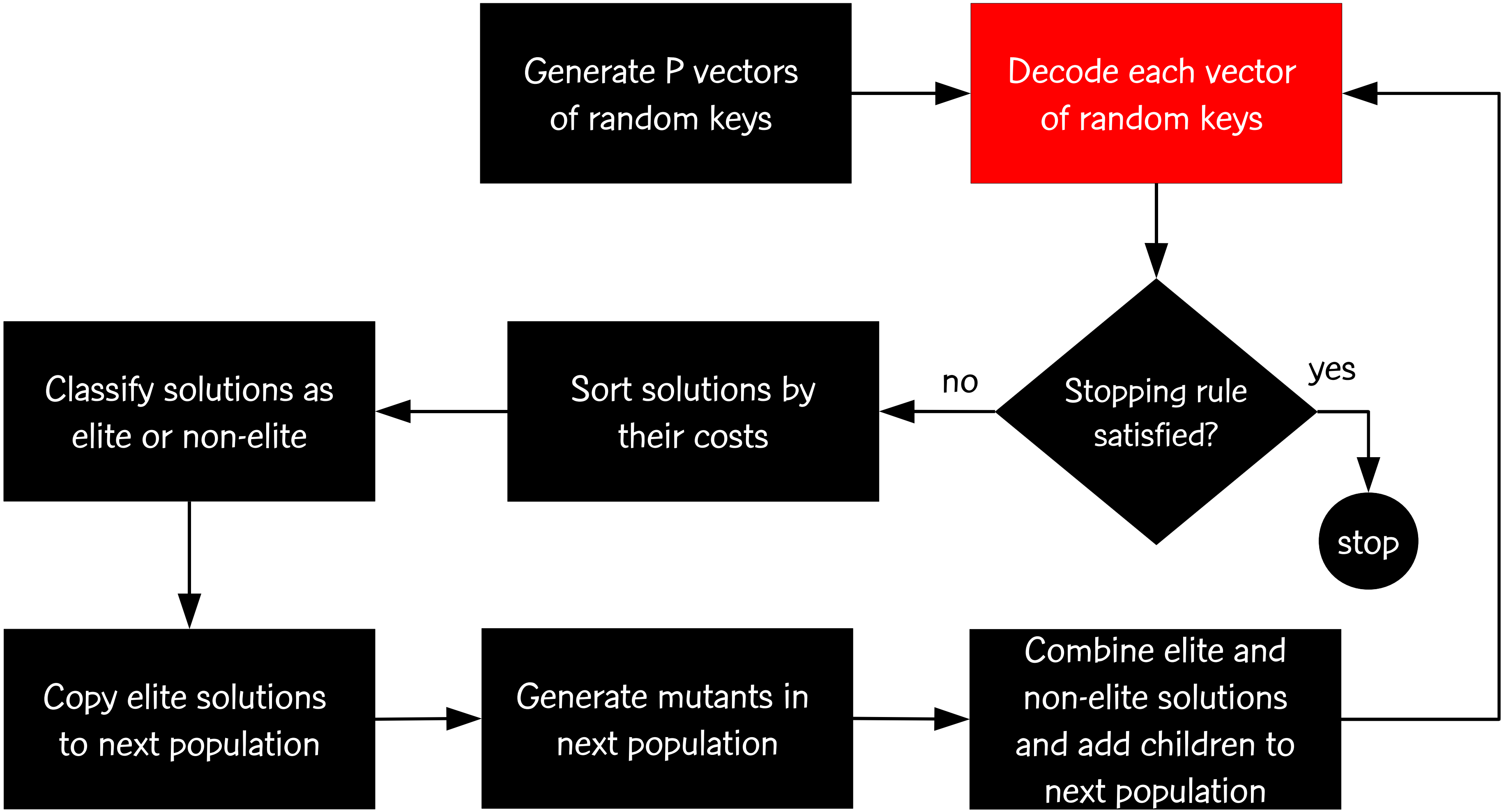
Decoders

- 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.

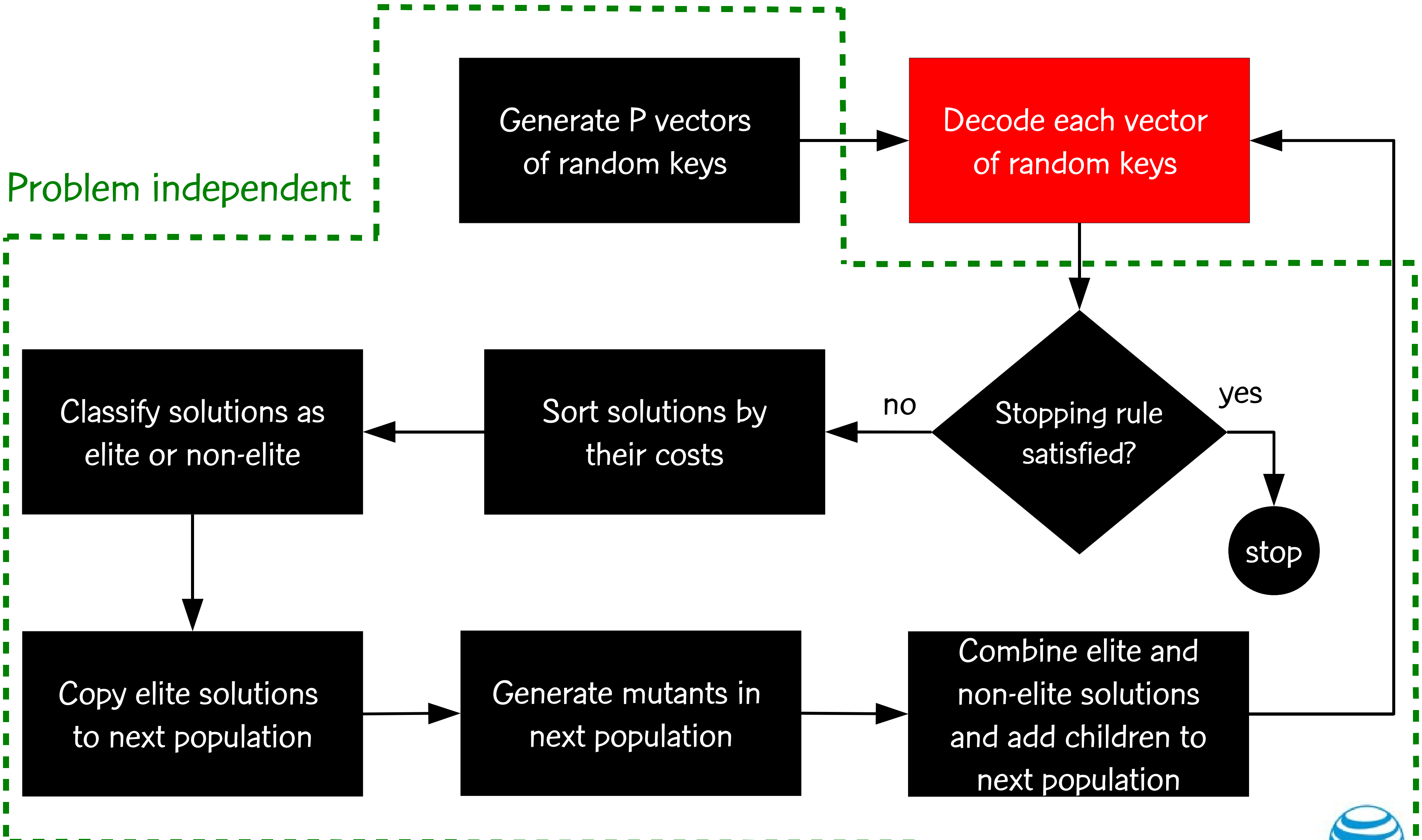


Node placement for monitoring

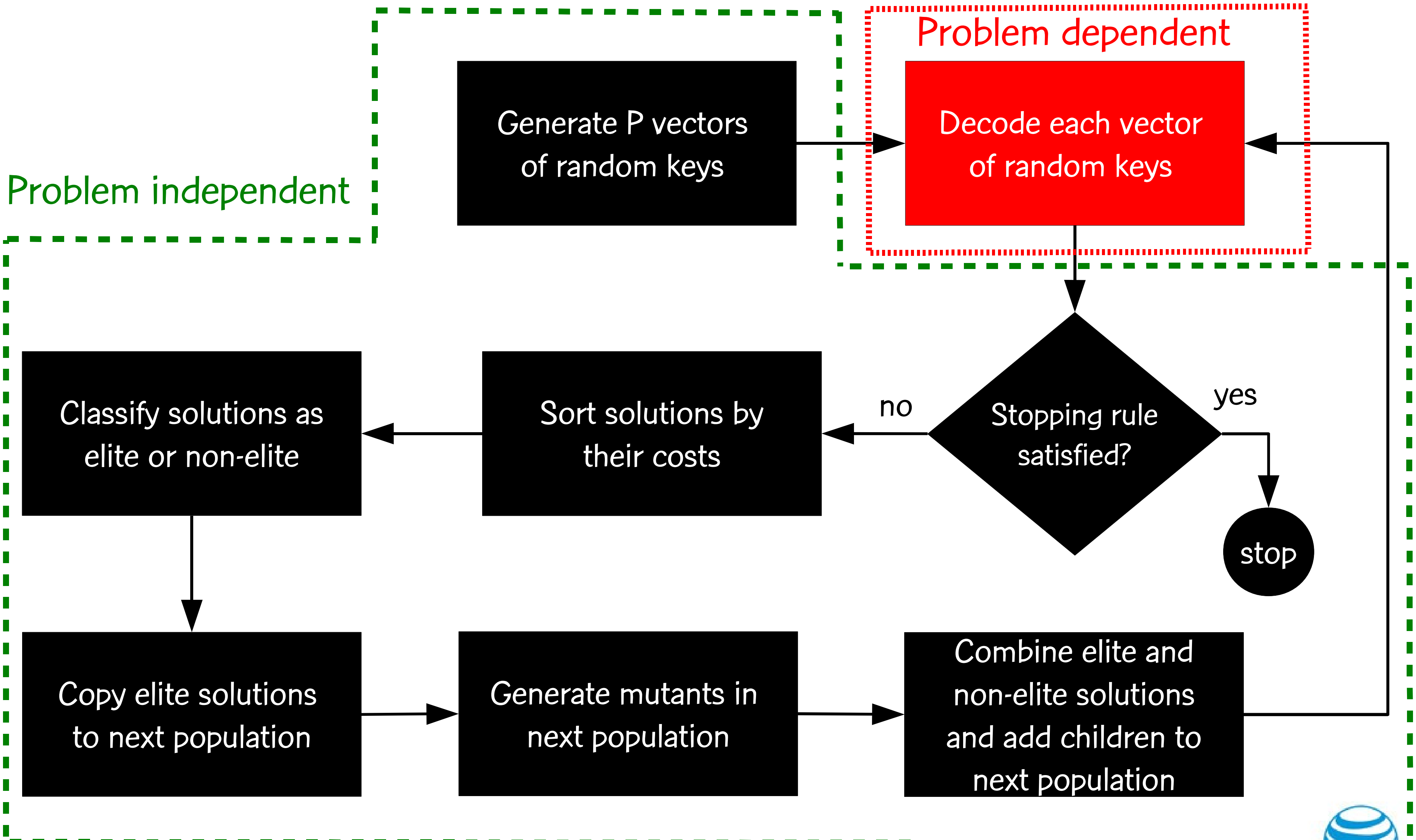
Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



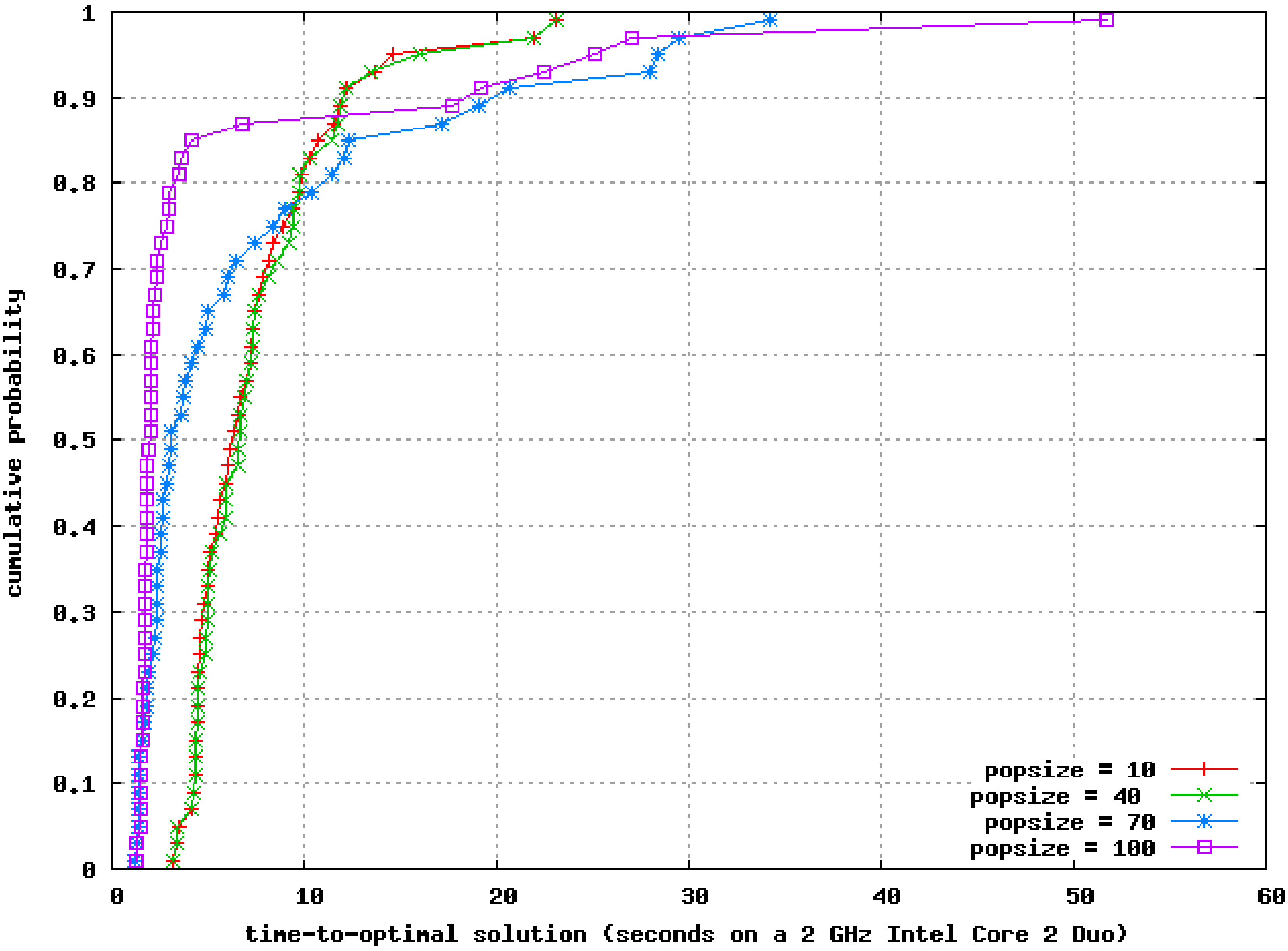
BRKGA for the MMS problem

at&t
Your world. Delivered.

BRKGA for the MMS problem

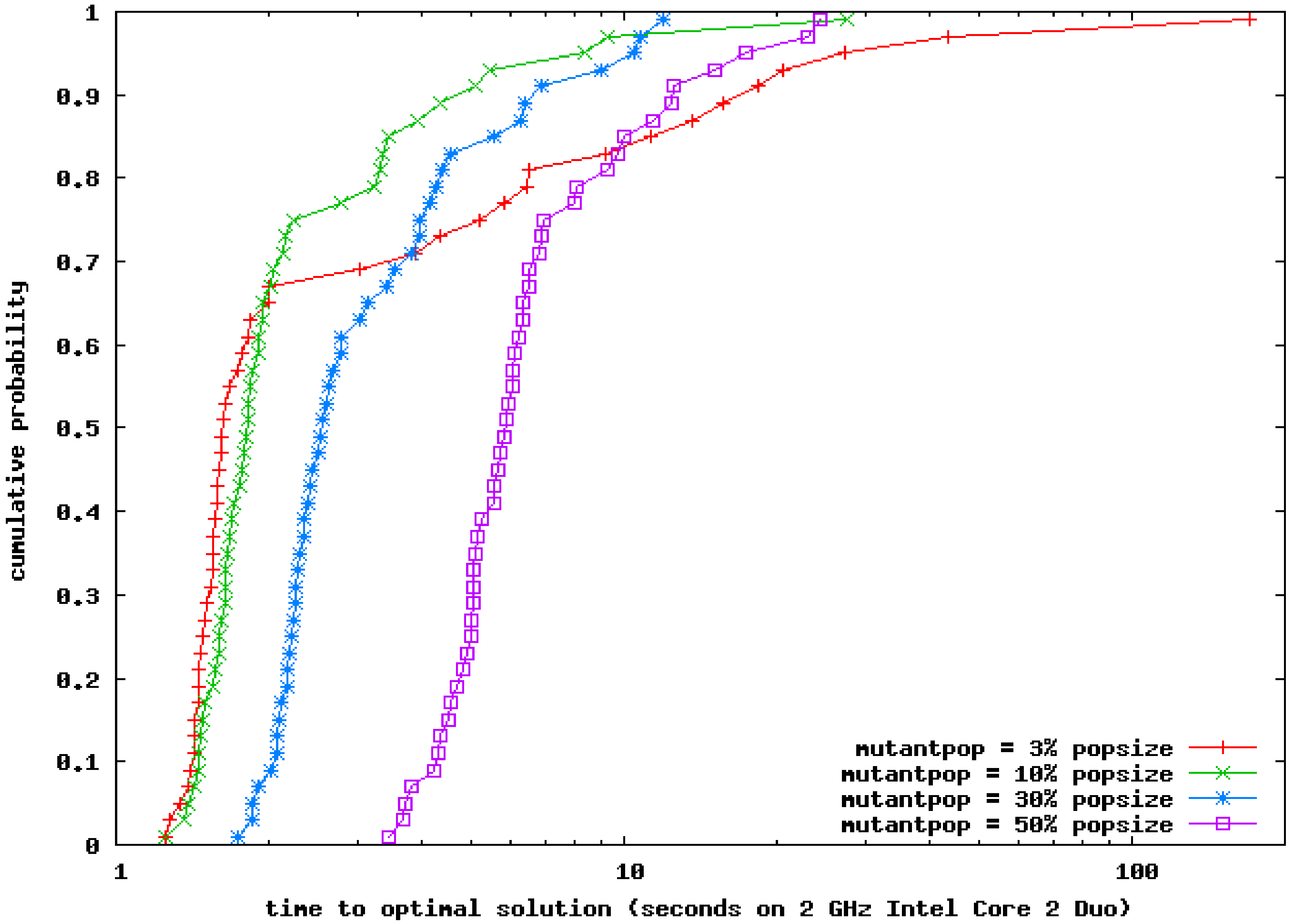
- Chromosome:
  - A vector X of N random random keys (random numbers in the interval [0,1]), where N is the number of potential monitoring nodes. The i-th random key corresponds to the i-th monitoring node.
- Decoder:
  - For i = 1,N: if $X(i) \geq 0.5$, add i-th monitoring node to solution
  - If solution is feasible, i.e. all customer nodes are covered: STOP
  - Else, apply greedy algorithm to cover uncovered branch nodes.

Effect of size of population: 100 node example

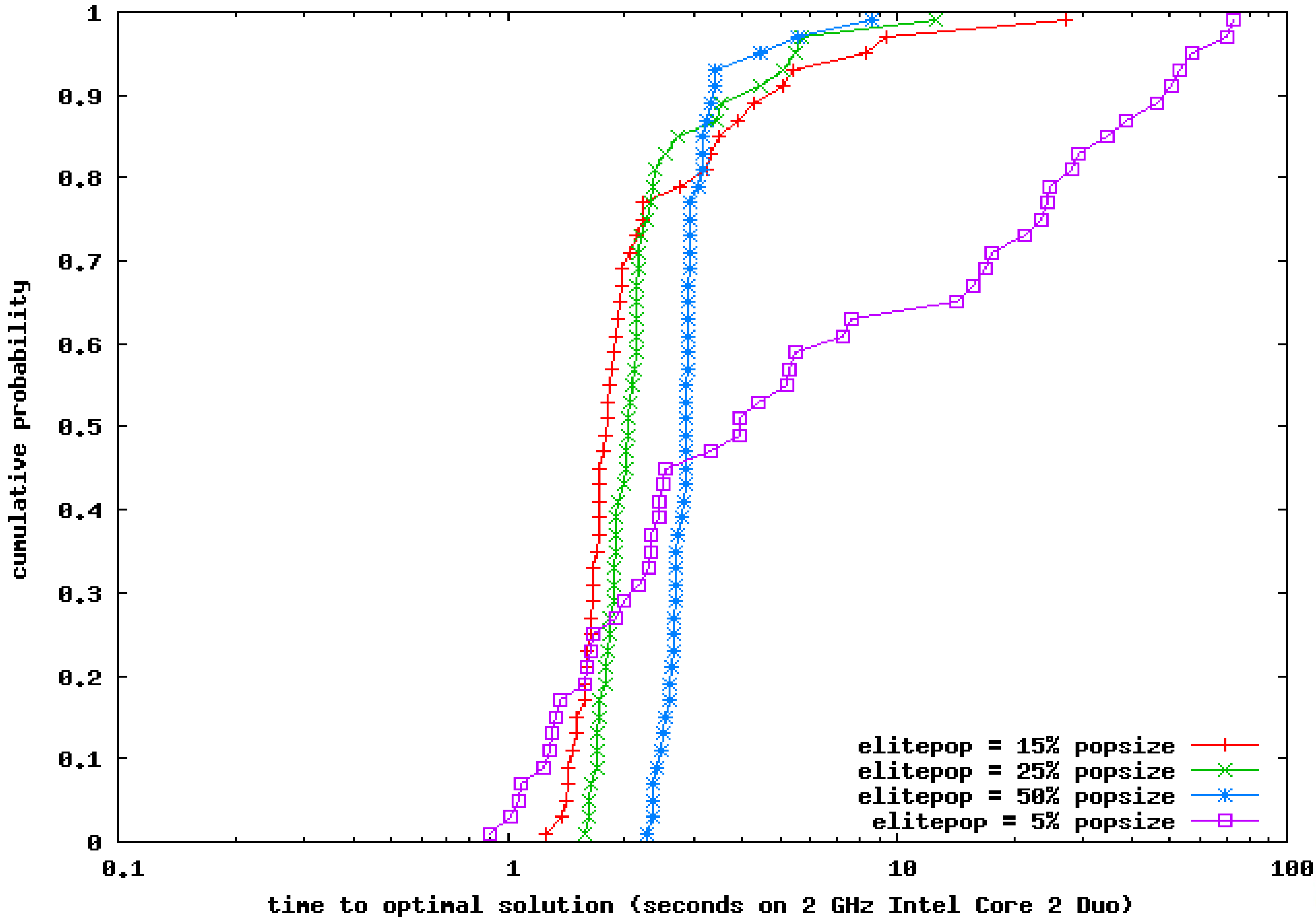


at&t Your world. Delivered.

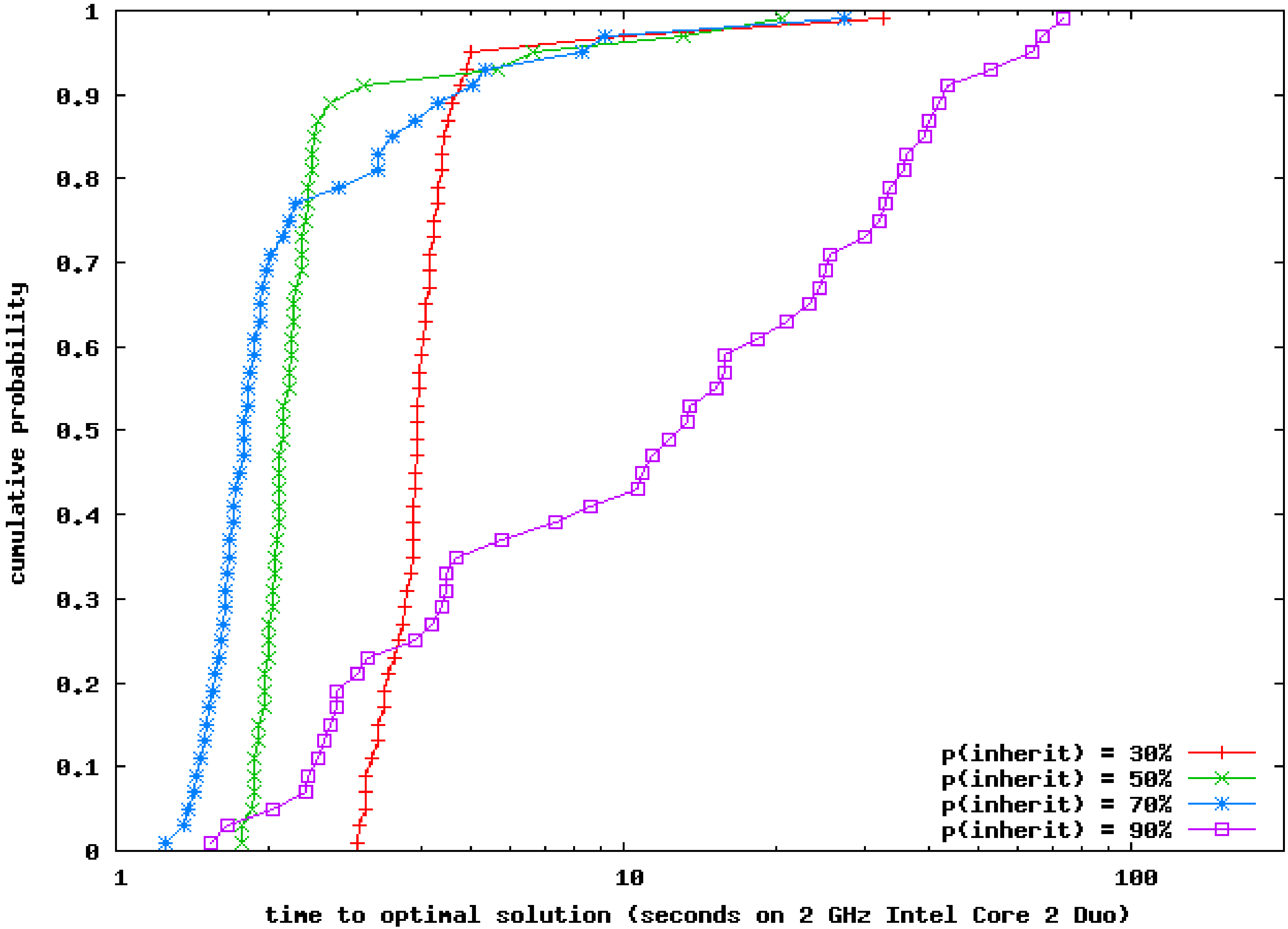
Effect of size of mutant set: 100 node example



Effect of size of elite set: 100 node example



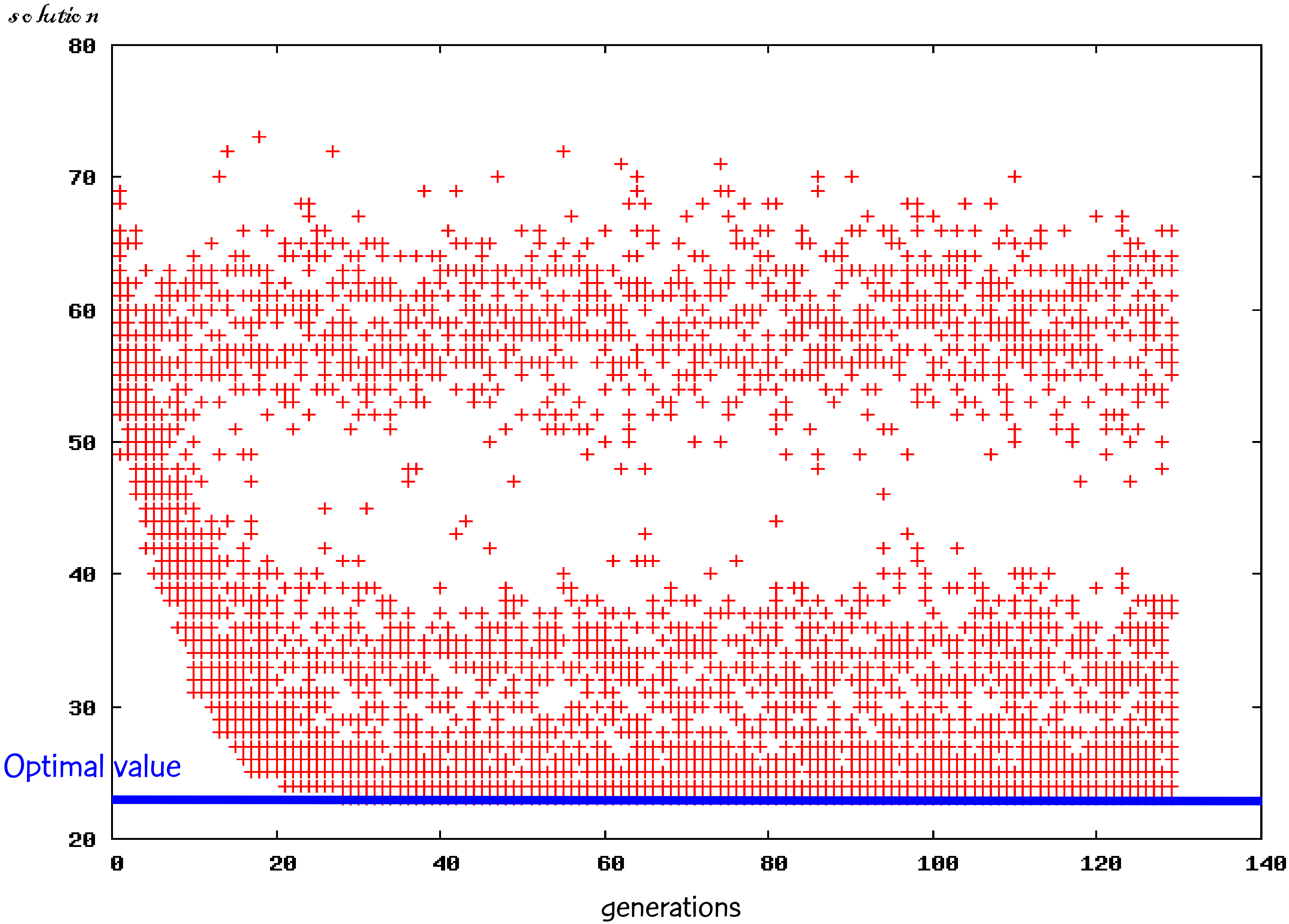
Effect of inheritance probability: 100 node example



BRKGA for the MMS problem

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

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

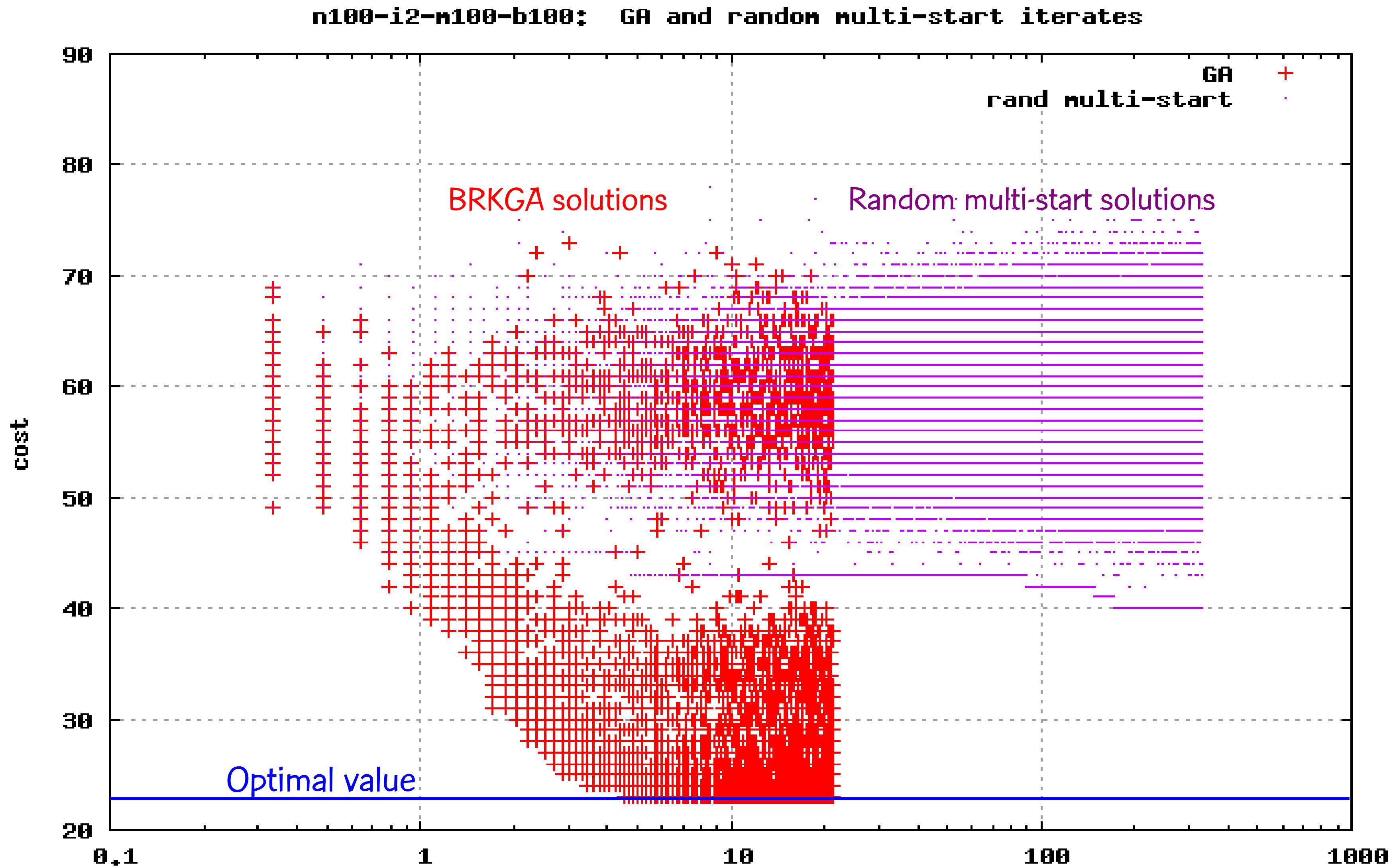


UPitt, March 4, 2010

Node placement for monitoring

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

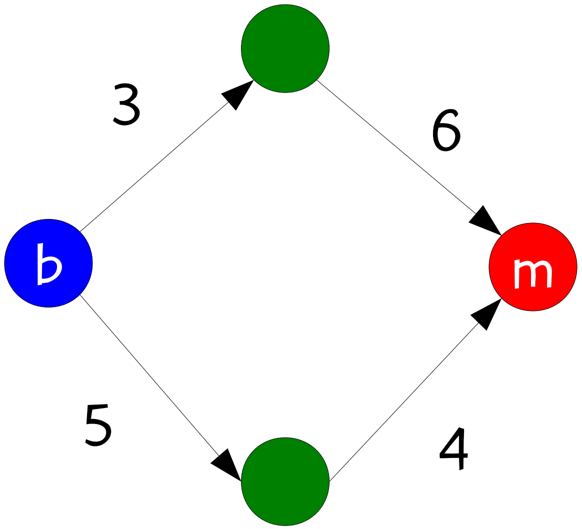
solution



Time (ibm t41 secs)

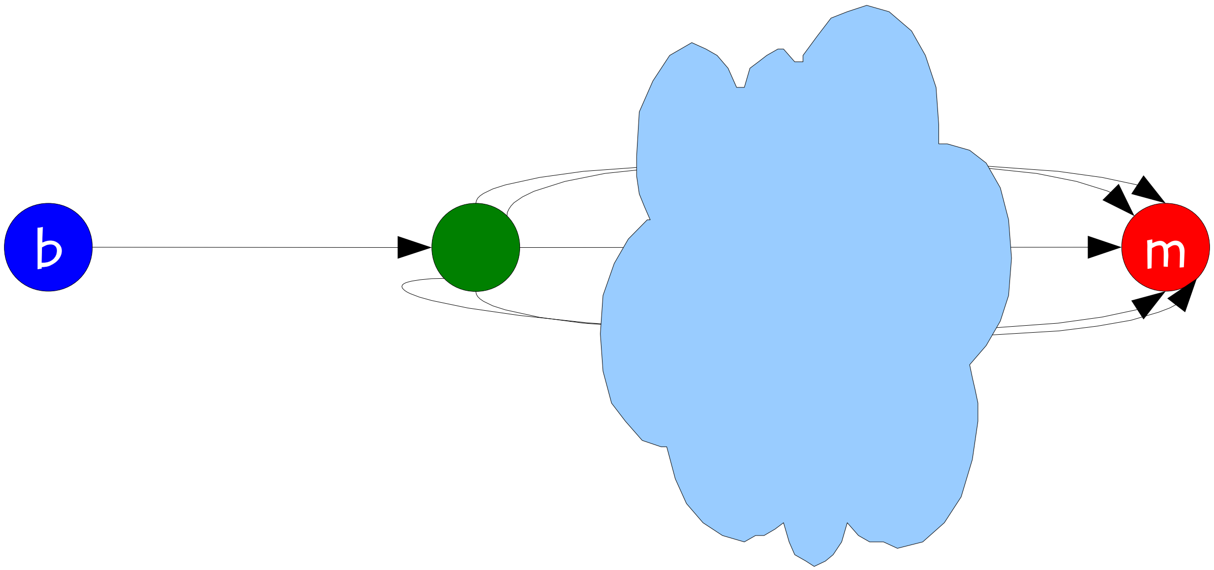
Double hitting set heuristic

Double hitting-set heuristic



- The above situation should NOT happen too much (especially if the weights are widely distributed)
- IDEA: Design an algorithm for the case of not much splitting and hope for the best

Double hitting-set heuristic



- DEFINITION: We say m is good for b if all shortest b ⟶ m paths depart b via the same arc.
- DEFINITION: For some fixed parameter t ($1 \leq t \leq |M|$), we say b is t-good if at least t m's are good for b.
- We LIKE t-good nodes, DESPISE t-bad ones.
- IDEA: Set aside t-bad nodes and deal with them later.

Double hitting-set heuristic

- 1a) For every t-good b, let
  X_b = { monitoring nodes m which are good for b }   ($|X_b| \geq t$)
- 1b) Find a small set $X \subseteq M$ such that, for all b, $X \cap X_b \neq \emptyset$
  (hitting set problem)

monitoring nodes

t-good branch nodes

1	1	0	0	1	0	← #1's ≥ t
0	1	1	1	0	1	
1	1	1	0	1	0	
0	0	1	0	1	1	
1	1	0	1	0	1	

t = 3 here

Double hitting-set heuristic

- 1a) For every t-good b, let
  X_b = { monitoring nodes m which are good for b }   ($|X_b| \geq t$)
- 1b) Find a small set $X \subseteq M$ such that, for all b, $X \cap X_b \neq \emptyset$
  (hitting set problem)

monitoring nodes

t-good branch nodes							
	1	1	0	0	1	0	← #1's ≥ t
	0	1	1	1	0	1	
	1	1	1	0	1	0	
	0	0	1	0	1	1	t = 3
	1	1	0	1	0	1	here

Greedy algorithm

REPEAT:
1) choose column with most 1's
2) delete rows with 1 in that column

Double hitting-set heuristic

- 1a) For every t-good b, let X_b = { monitoring nodes m which are good for b }   ($|X_b| \geq t$)
- 1b) Find a small set $X \subseteq M$ such that, for all b, $X \cap X_b \neq \emptyset$

(hitting set problem)

monitoring nodes

t-good branch nodes							
	1	1	0	0	1	0	← #1's ≥ t
	0	1	1	1	0	1	
	1	1	1	0	1	0	
	0	0	1	0	1	1	t = 3 here
	1	1	0	1	0	1	

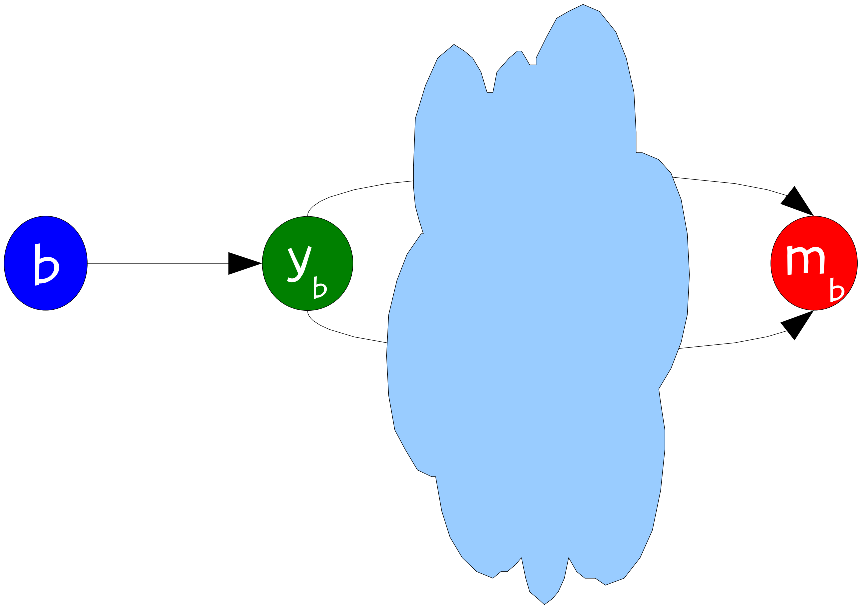
Greedy algorithm

REPEAT:
1) choose column with most 1's
2) delete rows with 1 in that column

Double hitting-set heuristic

- 1a) For every t-good b, let
  X_b = { monitoring nodes m which are good for b }   ($|X_b| \geq t$)
- 1b) Find a small set $X \subseteq M$ such that, for all b, $X \cap X_b \neq \emptyset$

  (hitting set problem)
- 1c) For all t-good b, choose
  $m_b \in X \cap X_b$. Let y_b be such that arc (b, y_b) is on shortest path $b \rightarrow m_b$



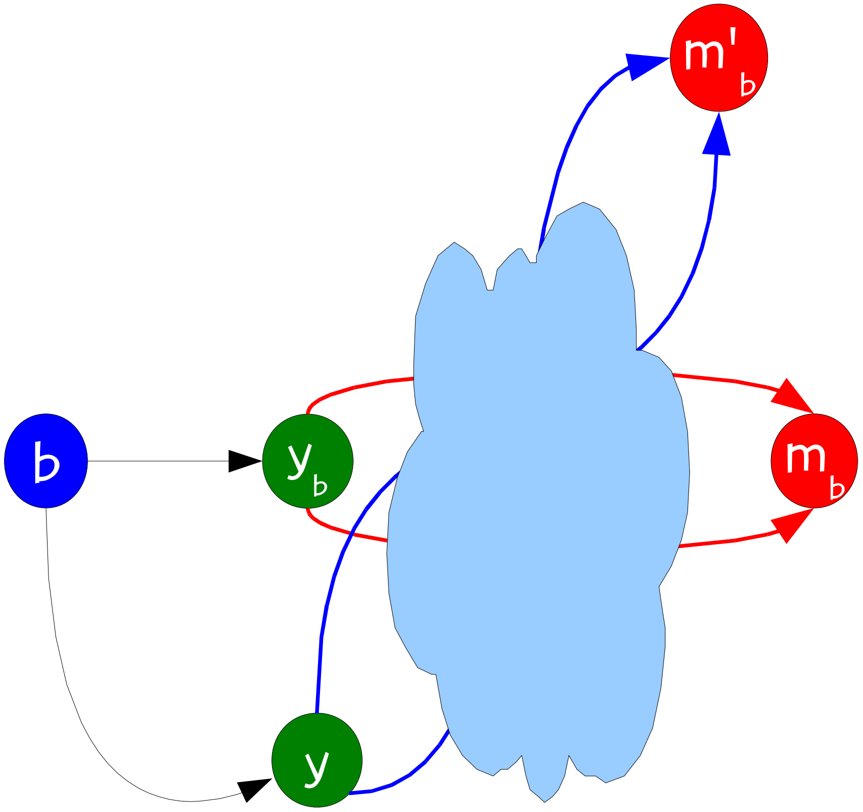
Double hitting-set heuristic

- We now have one monitoring node for b: m_b. We need a second monitoring node m'_b such that { m_b, m'_b } covers b.
- 2a) For every t-good node b, let Y_b = { monitoring nodes m such that all

  b $\rightarrow$ m shortest paths avoid
  arc (b, y_b) }
- Find a small set Y such that, for all b, $Y \cap Y_b \neq \emptyset$ (another hitting-set problem)

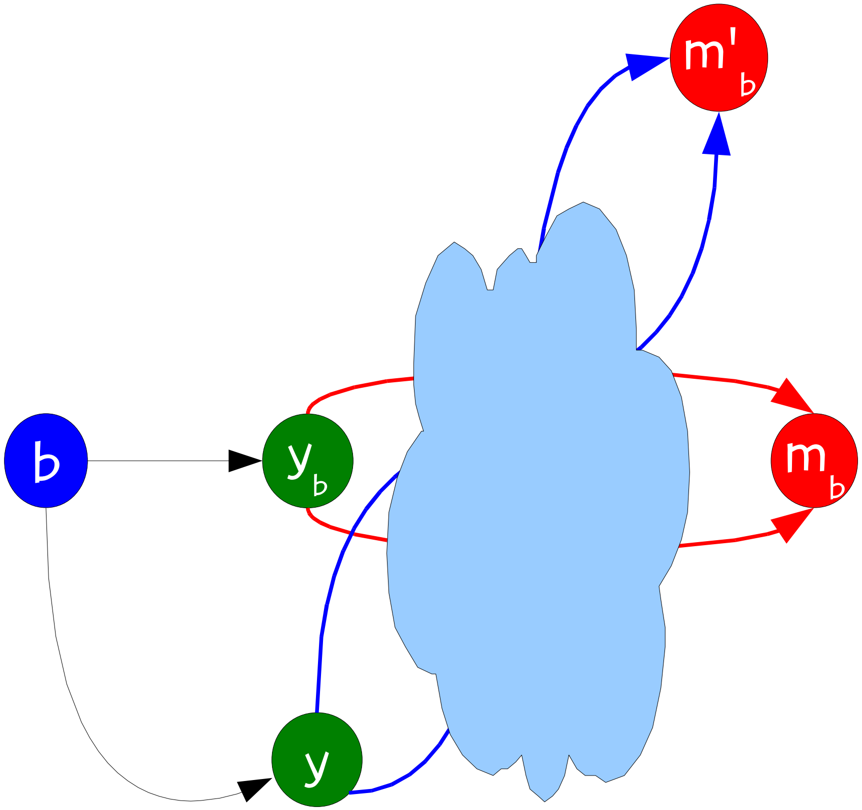
Double hitting-set heuristic

- We now have one monitoring node for b: m_b. We need a second monitoring node m'_b such that { m_b, m'_b } covers b.
- 2a) For every t-good node b, let Y_b = { monitoring nodes m such that all
  b $\longrightarrow$ m shortest paths avoid
  arc (b, y_b) }
- 2b) Find a small set Y such that, for all b, $Y \cap Y_b \neq \emptyset$ (another hitting-set problem)



Double hitting-set heuristic

- We now have one monitoring node for b: m_b. We need a second monitoring node m'_b such that $\{ m_b, m'_b \}$ covers b.
- 2a) For every t-good node b, let $Y_b = \{$ monitoring nodes m such that all
  b $\rightarrow$ m shortest paths avoid
  arc (b, y_b) $\}$
- 2b) Find a small set Y such that, for all b, $Y \cap Y_b \neq \emptyset$ (another hitting-set problem)



CLAIM: paths b $\rightarrow$ m_b and b $\rightarrow$ m'_b are disjoint:
$\{ m_b, m'_b \}$ covers b

Node placement for monitoring

Double hitting-set heuristic

- We now have covered all the t-good nodes. We must now cover the t-bad nodes (hopefully only a handful for them)
- 3a) Cover the t-bad nodes with the greedy algorithm.
- 4a) Remove redundant monitoring nodes ("minimalize")

Double hitting-set heuristic

- We now have covered all the t-good nodes. We must now cover the t-bad nodes (hopefully only a handful for them)
- 3a) Cover the t-bad nodes with the greedy algorithm.
- 4a) Remove redundant monitoring nodes ("minimalize")

What should be the value of parameter t?

If t is large, then first hitting set has small solution

If t is small, then there are more t-good nodes, fewer t-bad nodes

IDEA: try t = floor(|M|/2) and t = 1 and take better solution.

Double hitting-set heuristic

- We now have covered all the t-good nodes. We must now cover the t-bad nodes (hopefully only a handful for them)
- 3a) Cover the t-bad nodes with the greedy algorithm.
- 4a) Remove redundant monitoring nodes ("minimalize")

Performance in theory: If t = floor(|M|/2), the cover of the n t-good nodes has size (1+ log n) times OPT.

What should be the value of parameter t?

If t is large, then first hitting set has small solution

If t is small, then there are more t-good nodes, fewer t-bad nodes

IDEA: try t = floor(|M|/2) and t = 1 and take better solution.

Lower bound

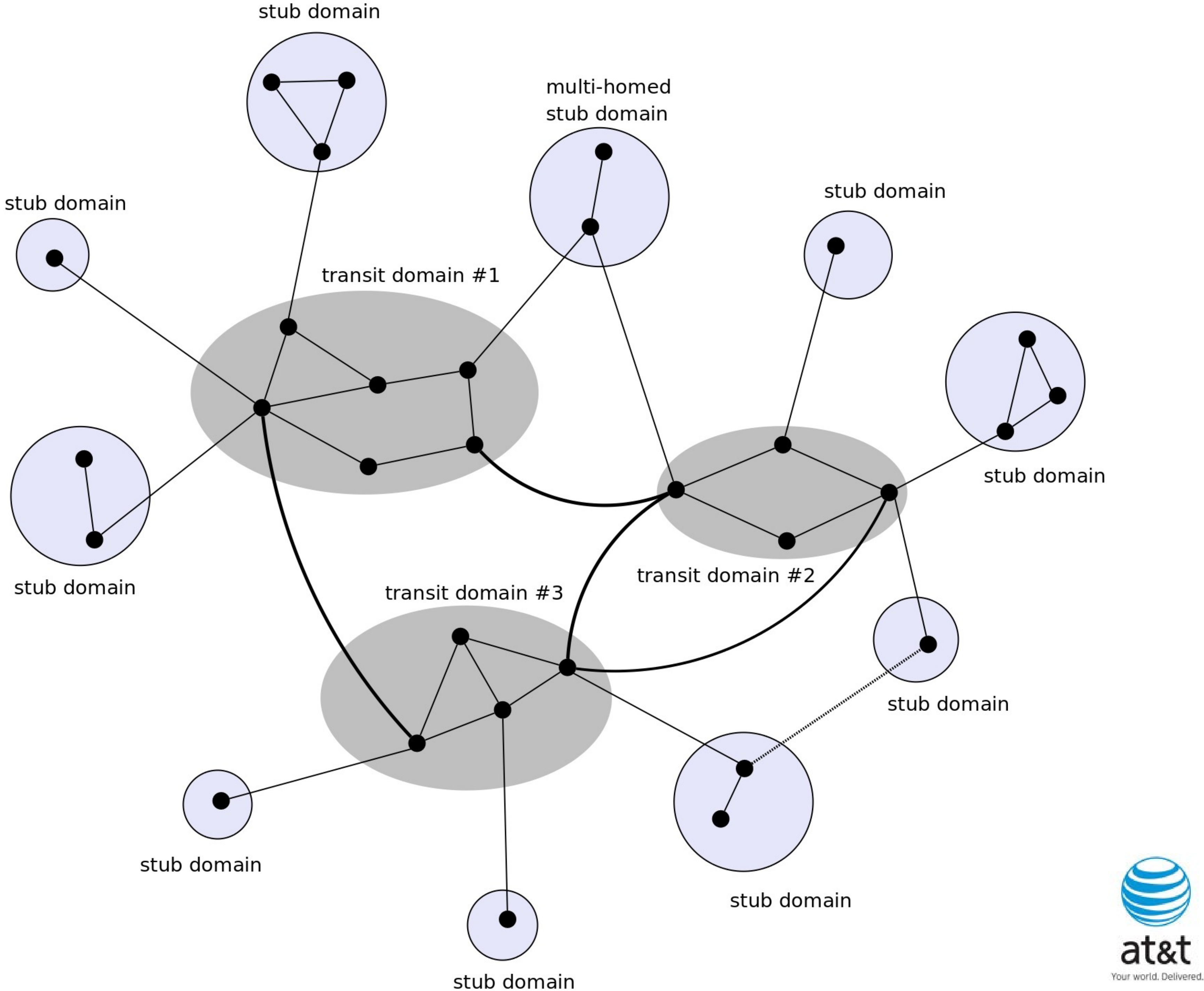
Lower bound on OPT

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

Experimental results

Experimental results

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


stub domain
multi-homed
stub domain
stub domain
stub domain
transit domain #1
stub domain
stub domain
stub domain
transit domain #2
transit domain #3
stub domain
stub domain
stub domain
stub domain
at&t
Your world. Delivered.

Experimental results

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

Synthetic networks

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

Synthetic networks

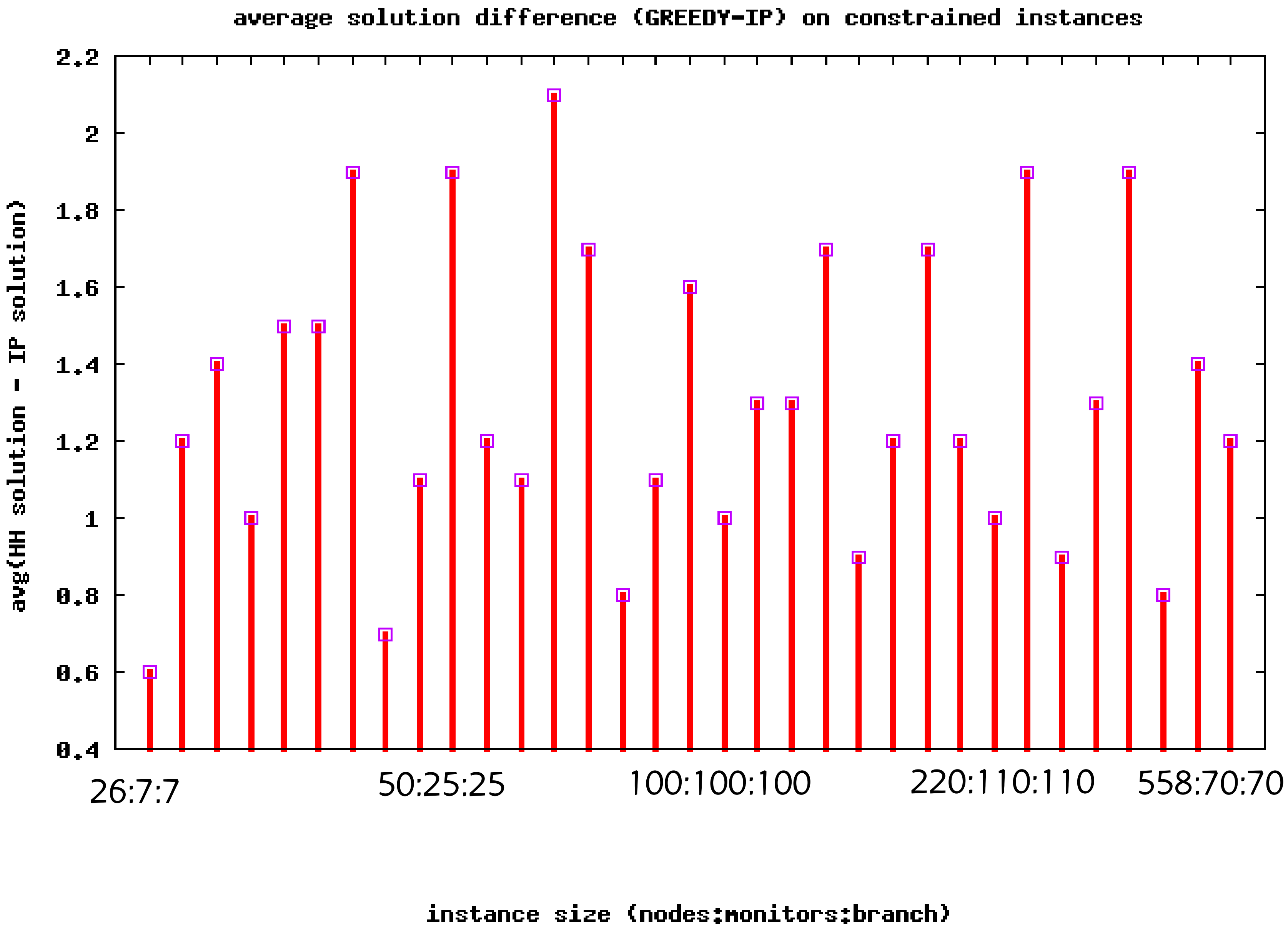
- CPLEX computed lower bounds for all 560 instances
- Heuristics matched the lower bound for some of those instances:
  - Greedy algorithm: 236/560 = 42.1%
  - Double hitting set algorithm: 363/560 = 64.8 %
  - Genetic algorithm: 394/560 = 70.4%

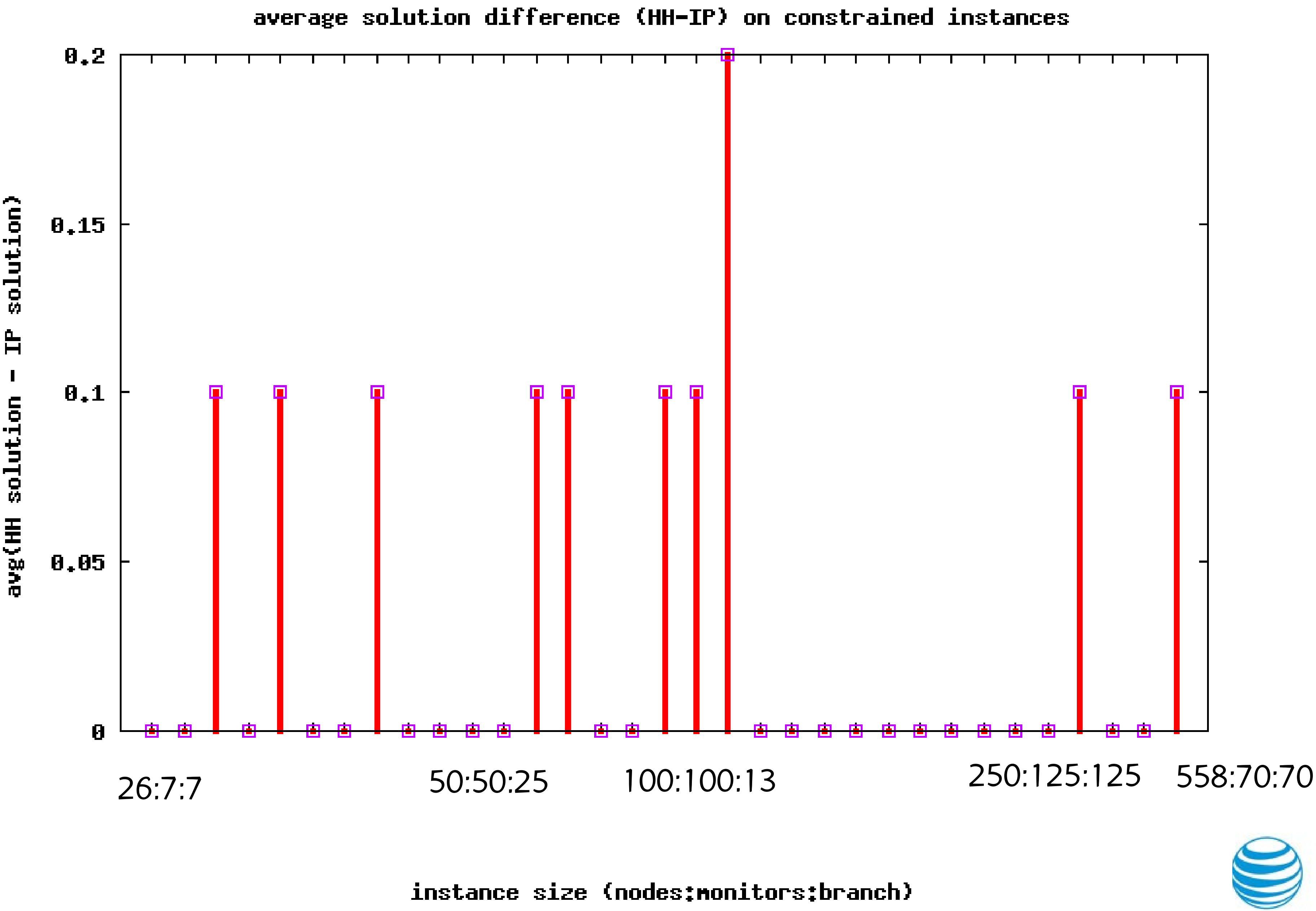
Synthetic networks: comparing heuristic solutions

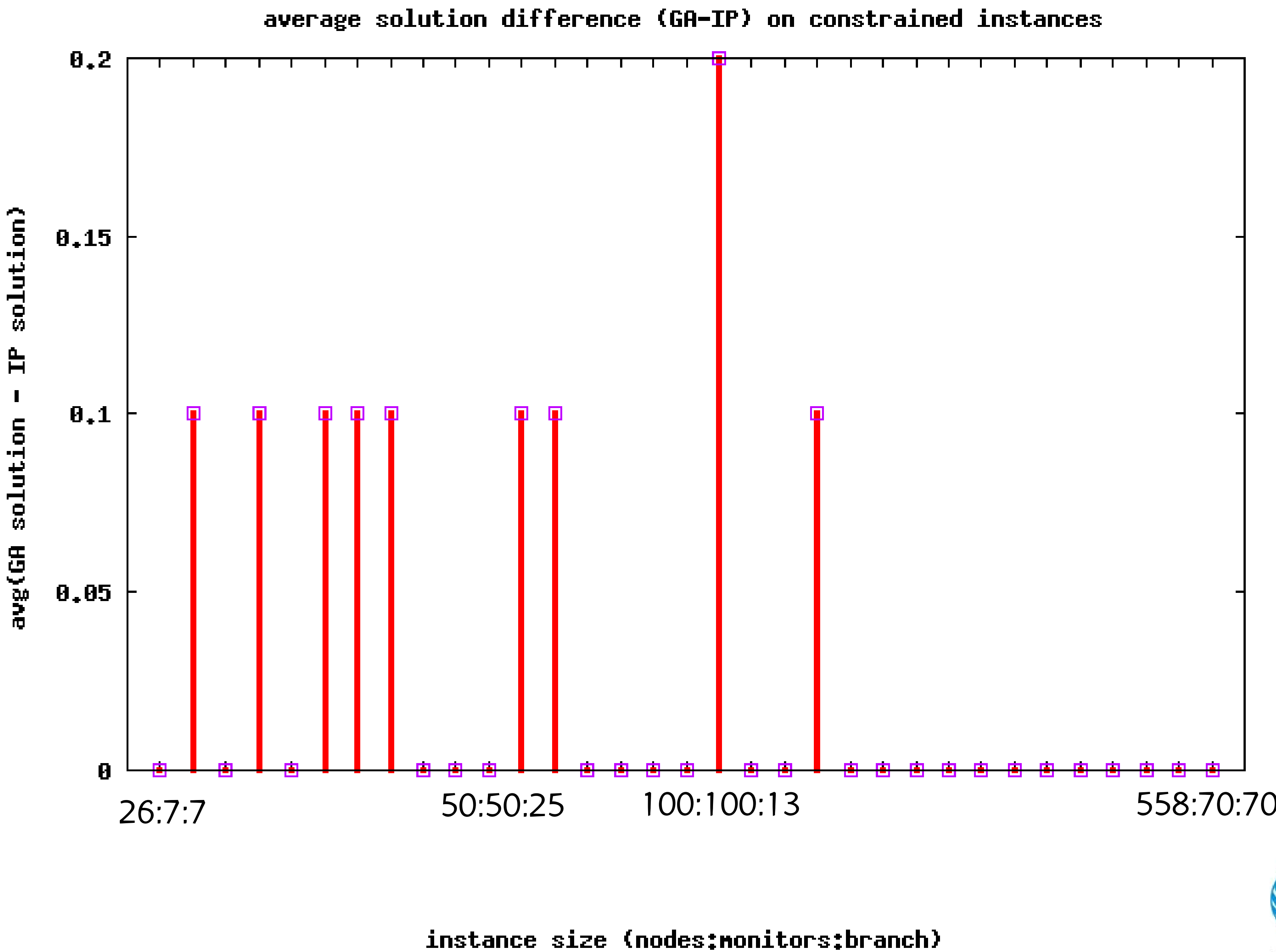
- Double hitting set (DHS) vs Greedy
  - DHS better than Greedy: 456/560 = 81.4%
  - DHS equal to Greedy: 90/560 = 16.1%
  - Greedy better than DHS: 14/560 = 2.5%
- Genetic algorithm (GA) vs DHS
  - GA better than DHS: 68/560 = 12.1%
  - GA equal to DHS: 482/560 = 86.1%
  - DHS better than GA: 10/560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

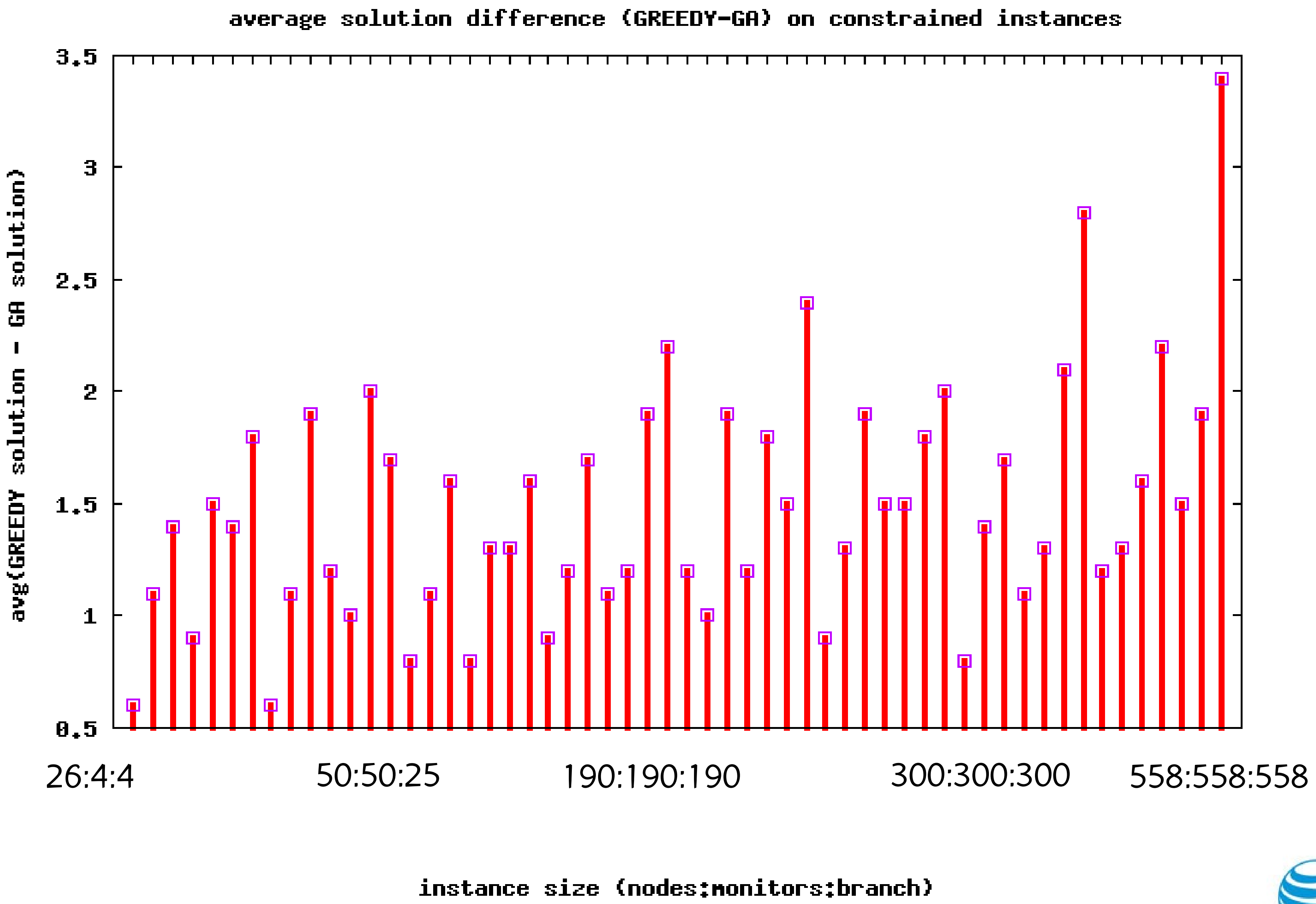
Synthetic networks

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

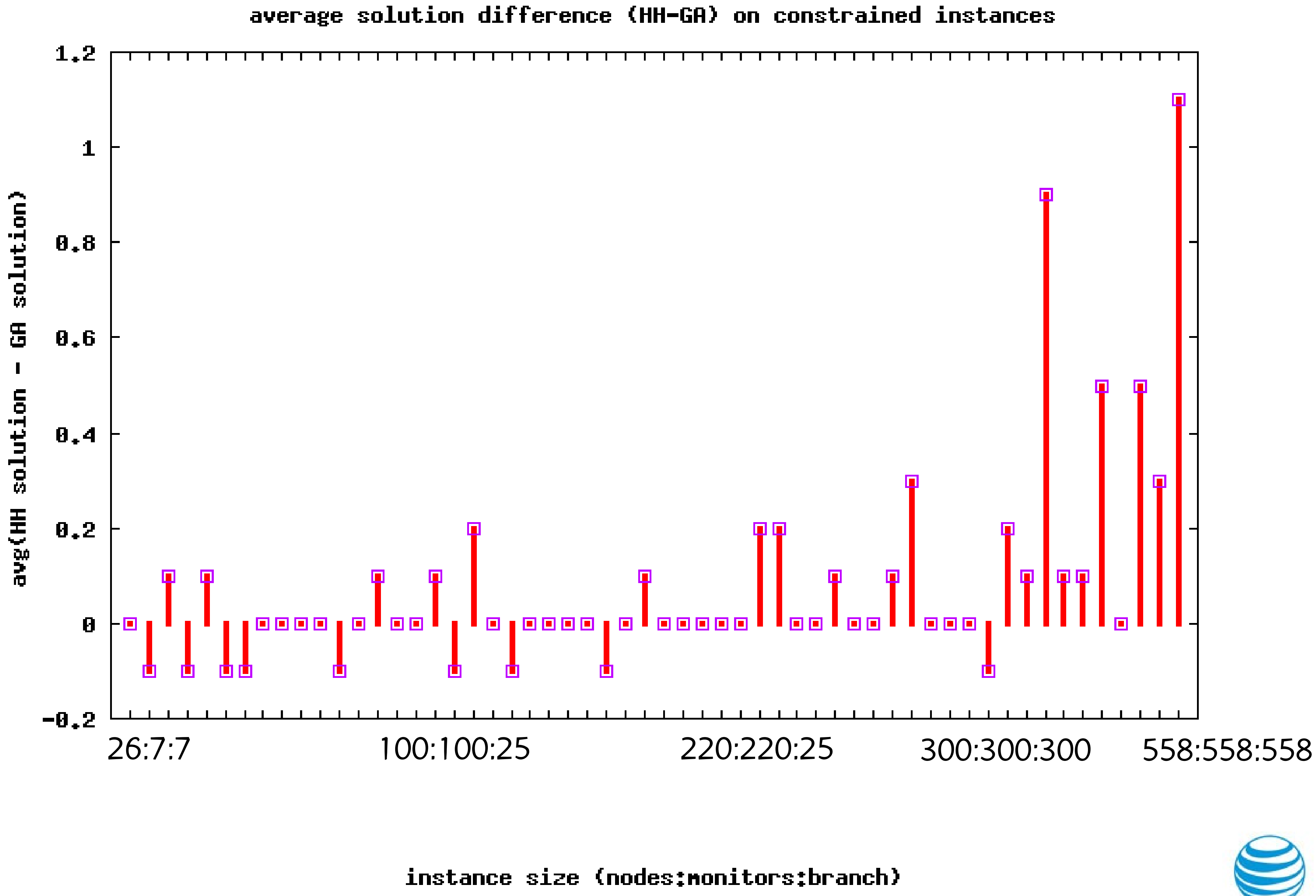

average solution difference (GREEDY-IP) on constrained instances
avg(HH solution - IP solution)
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
26:7:7
50:25:25
100:100:100
220:110:110
558:70:70
instance size (nodes:monitors:branch)


average solution difference (HH-IP) on constrained instances
avg(HH solution - IP solution)
0.2
0.15
0.1
0.05
0
26:7:7
50:50:25
100:100:13
250:125:125
558:70:70
instance size (nodes:monitors:branch)


average solution difference (GA-IP) on constrained instances
avg(GA solution - IP solution)
0.2
0.15
0.1
0.05
0
26:7:7
50:50:25
100:100:13
558:70:70
instance size (nodes:monitors:branch)



average solution difference (GREEDY-GA) on constrained instances
avg(GREEDY solution - GA solution)
3.5
3
2.5
2
1.5
1
0.5
26:4:4
50:50:25
190:190:190
300:300:300
558:558:558
instance size (nodes:monitors:branch)


average solution difference (HH-GA) on constrained instances
avg(HH solution - GA solution)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
26:7:7
100:100:25
220:220:25
300:300:300
558:558:558
instance size (nodes:monitors:branch)

Real networks

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

Real networks: comparing heuristic solutions

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

Real networks

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

Concluding remarks

Concluding remarks

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

The End

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

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