




# Optimization with random-key genetic algorithms



Mauricio G. C. Resende

Algorithms & Optimization Research Dept.  
AT&T Labs Research

[mgcr@att.com](mailto:mgcr@att.com)

# Content

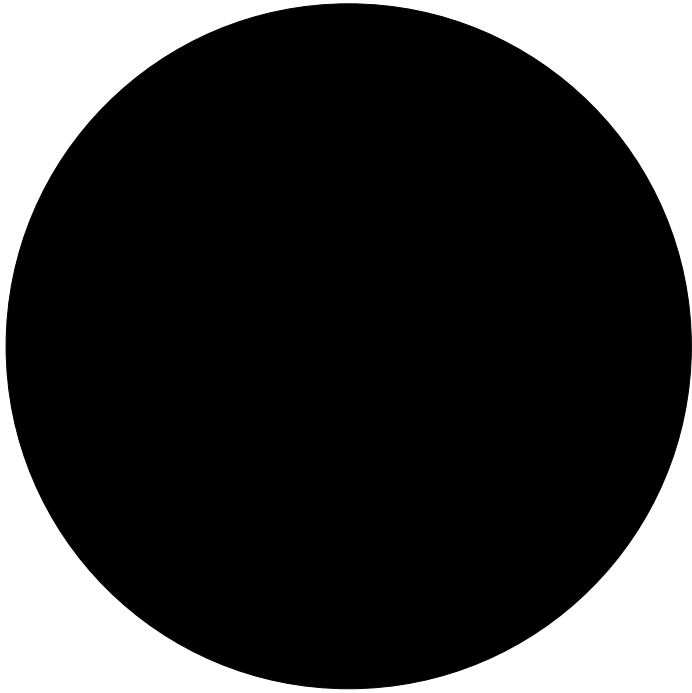
- Genetic algorithms (GAs)
- Random keys and GAs
- Example of optimization with random-key GAs
  - Networking
  - Optimization in other domains

# Genetic algorithms

# Genetic algorithms

Holland (1975)

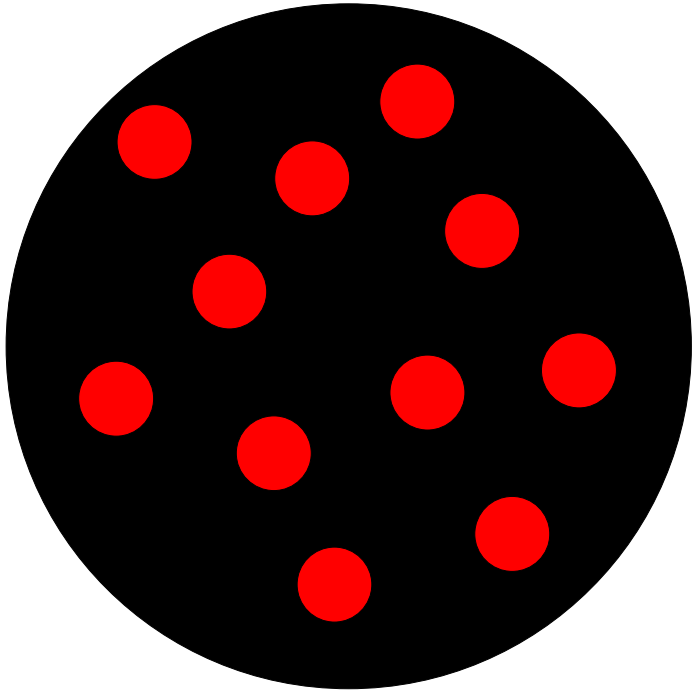
Adaptive methods that are used to solve search and optimization problems.



Individual: solution



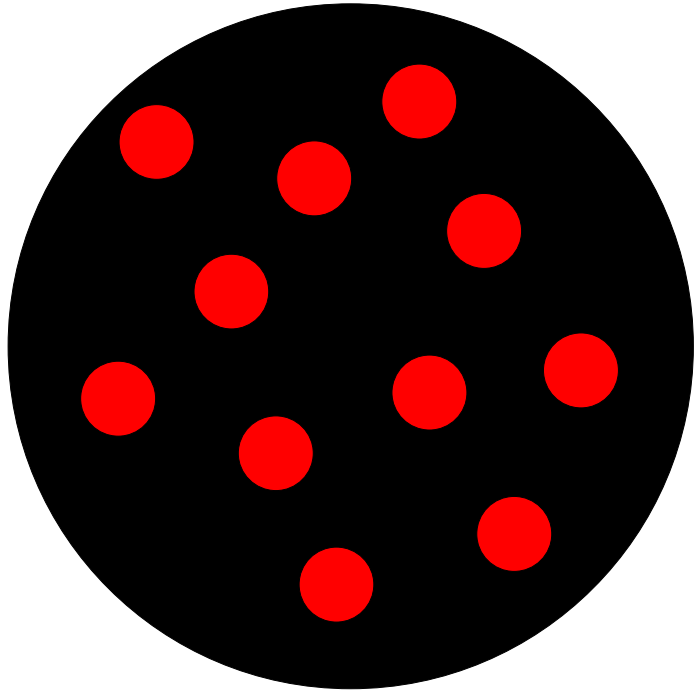
# Genetic algorithms



Individual: solution

Population: set of fixed number of individuals

# Genetic algorithms



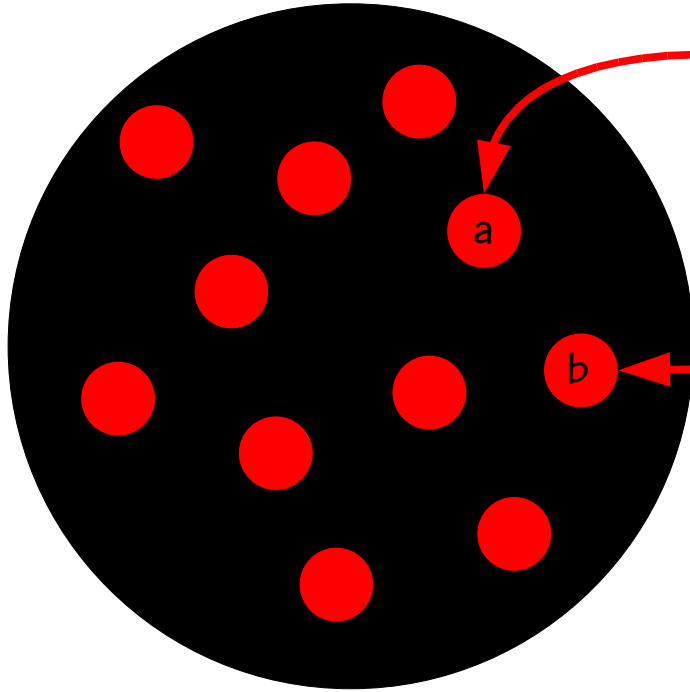
Genetic algorithms evolve population applying the principle of survival of the fittest.

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

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

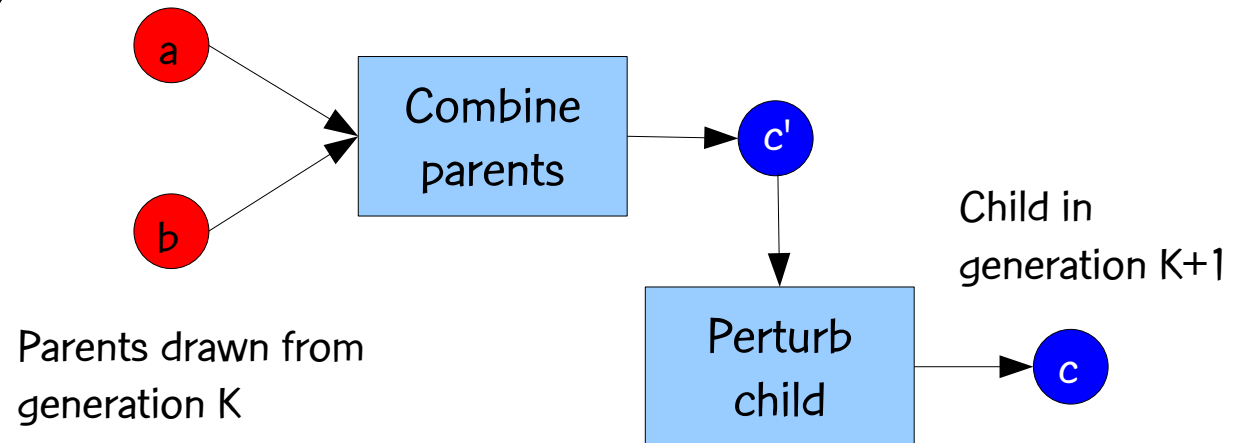
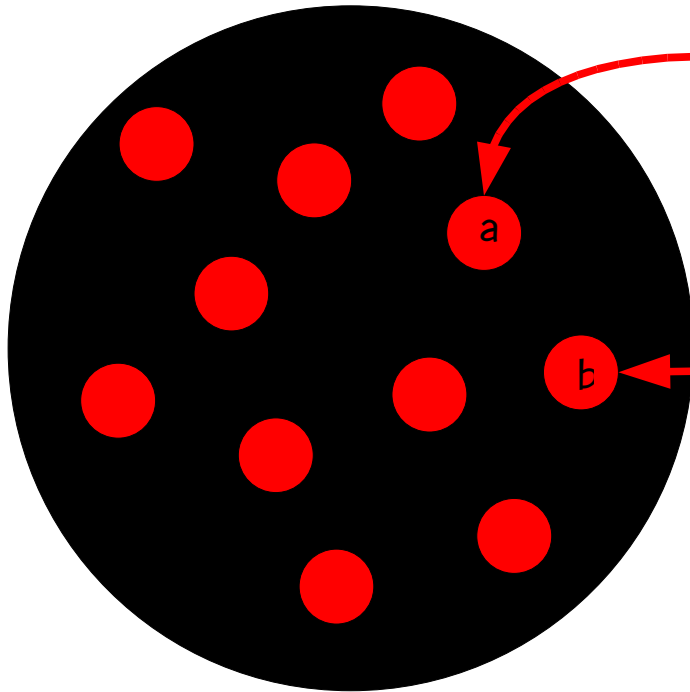
# Genetic algorithms

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



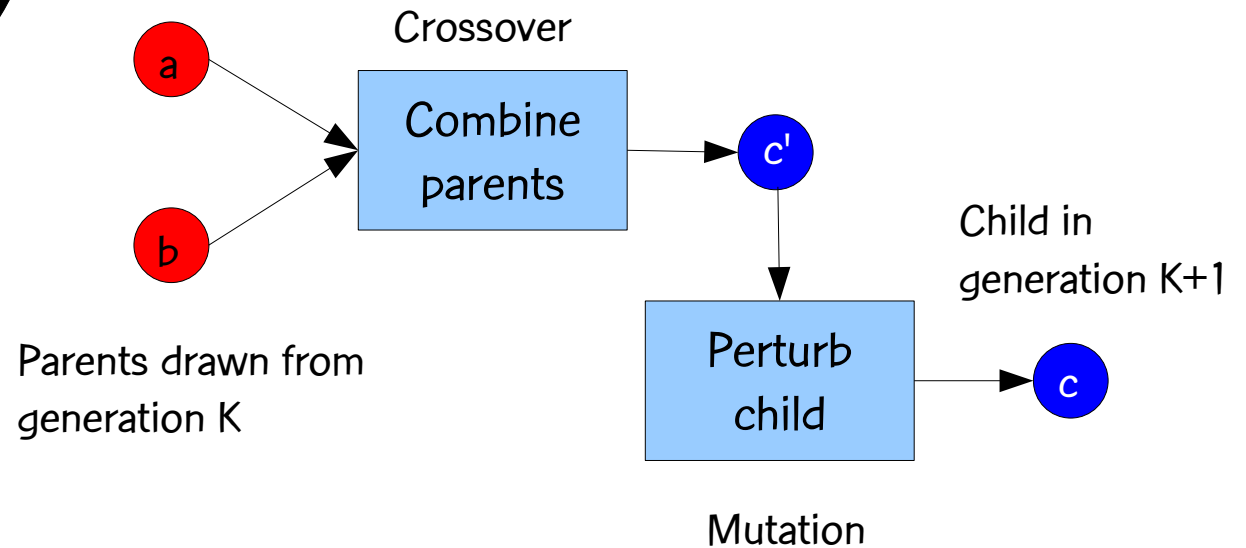
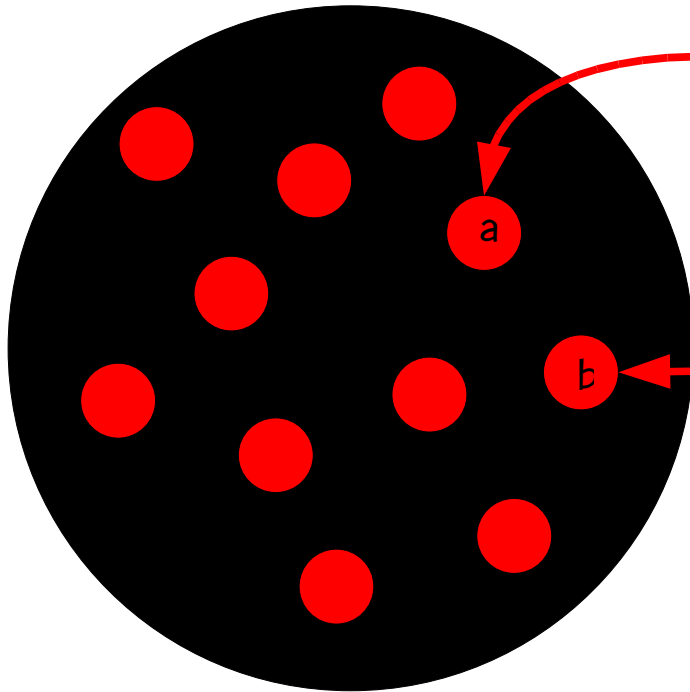
# Genetic algorithms

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

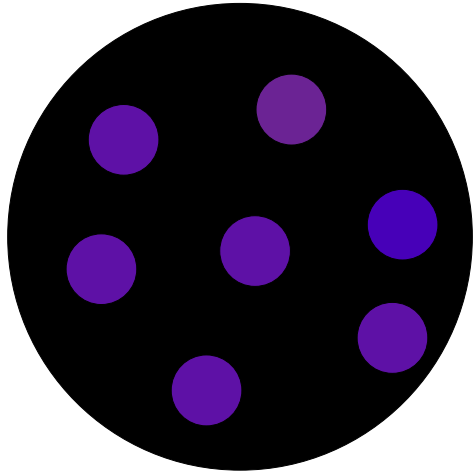


# Genetic algorithms

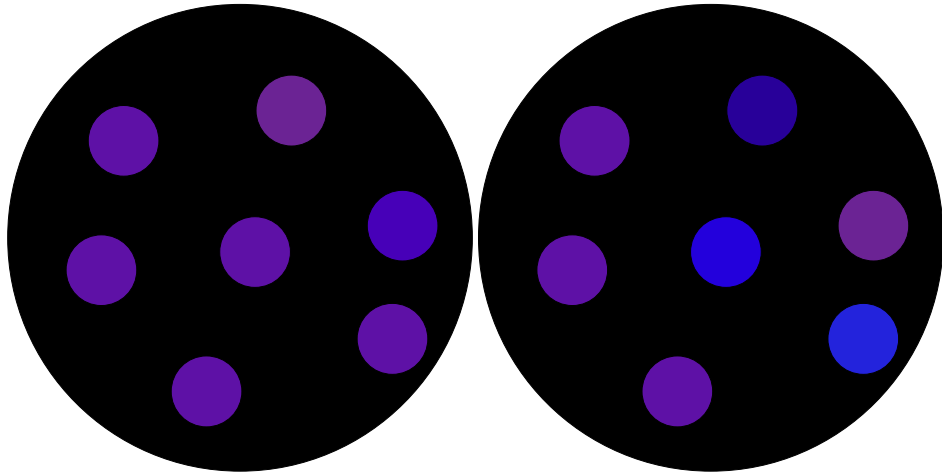
Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.



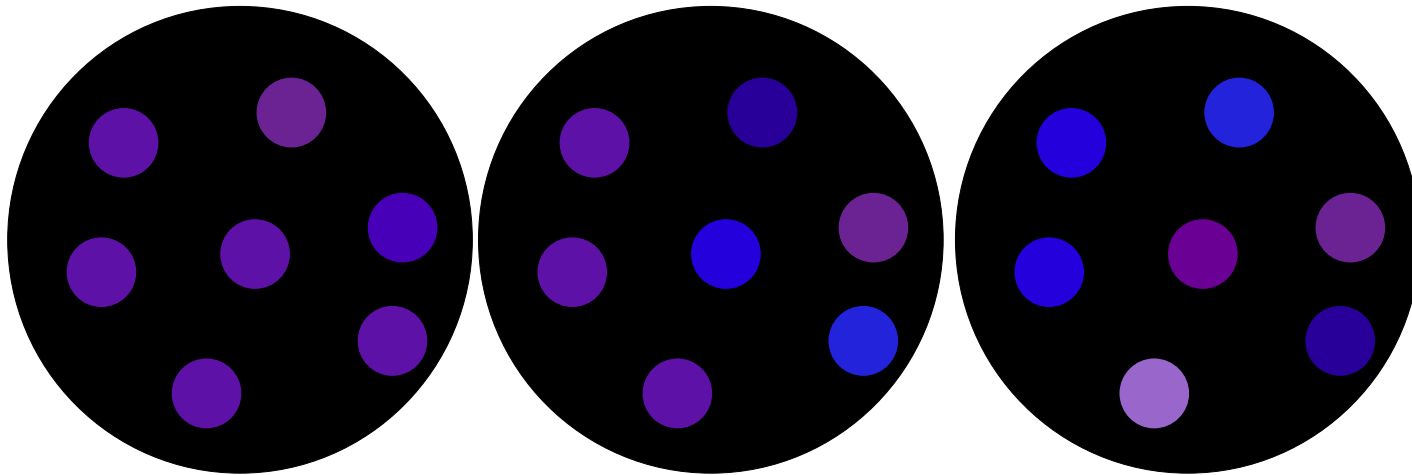
# Evolution of solutions



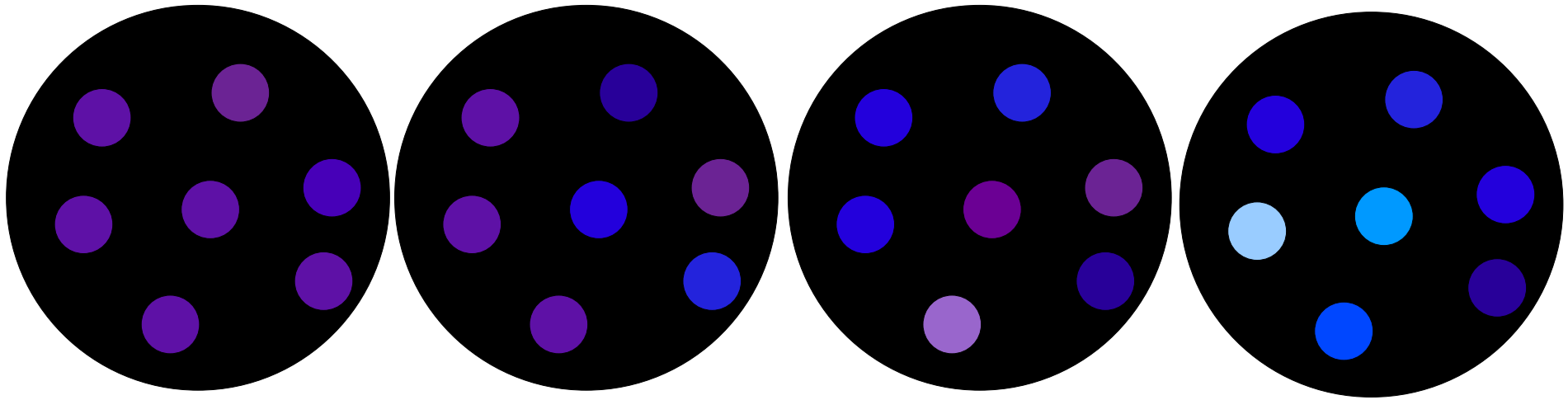
# Evolution of solutions



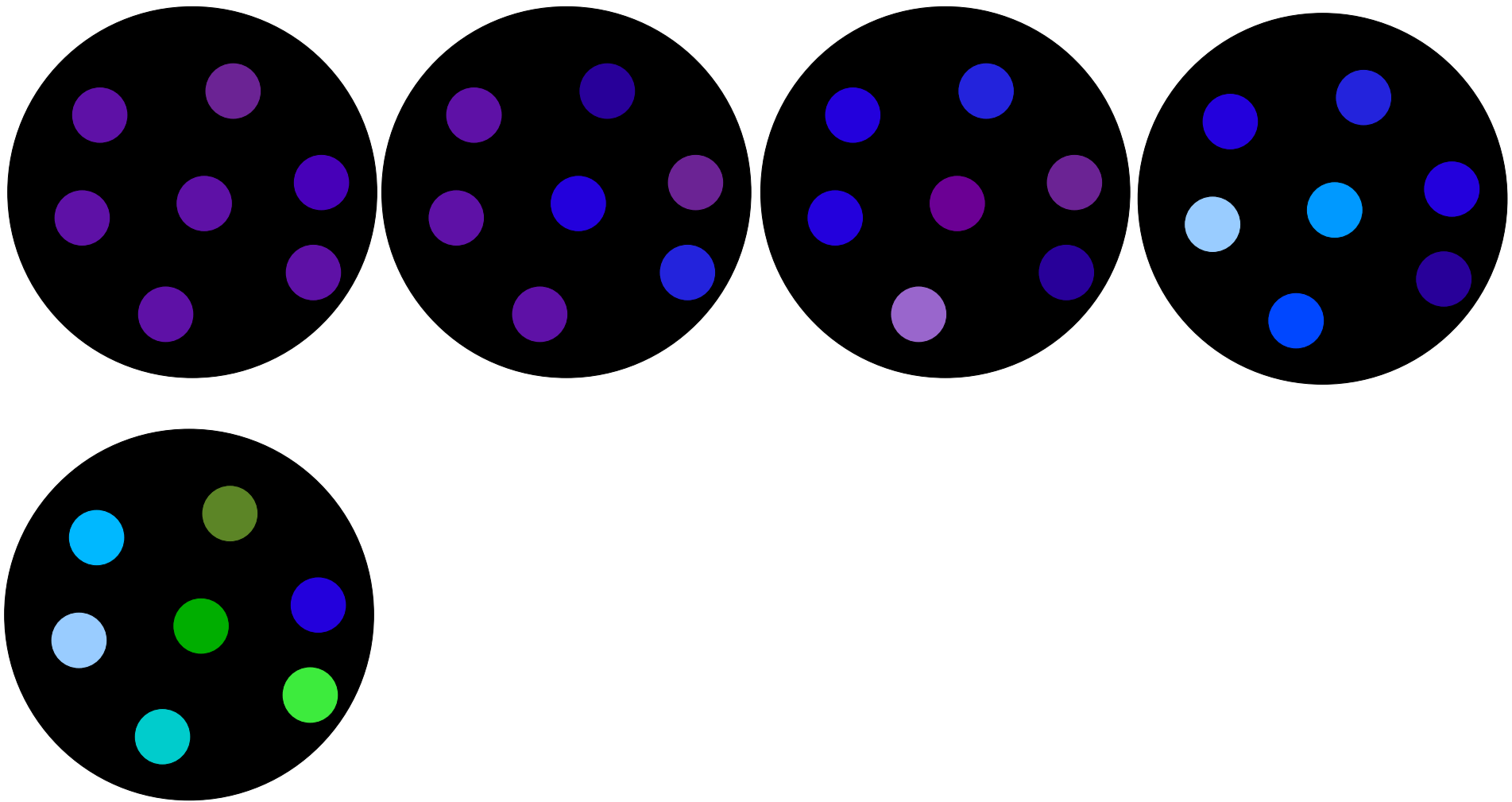
# Evolution of solutions



# Evolution of solutions



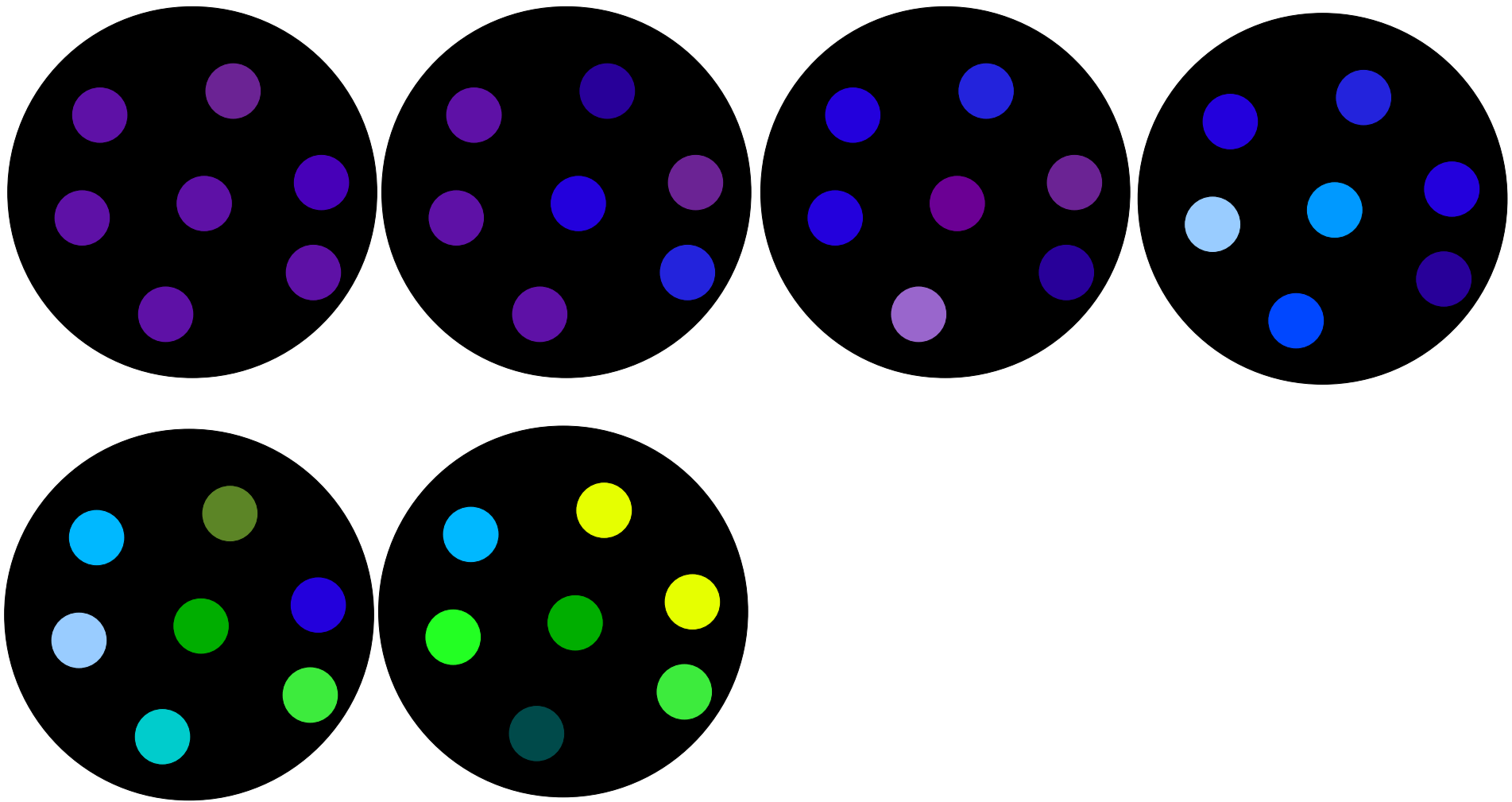
# Evolution of solutions



Florham Park, December 14, 2007

Optimization by random-key  
genetic algorithms

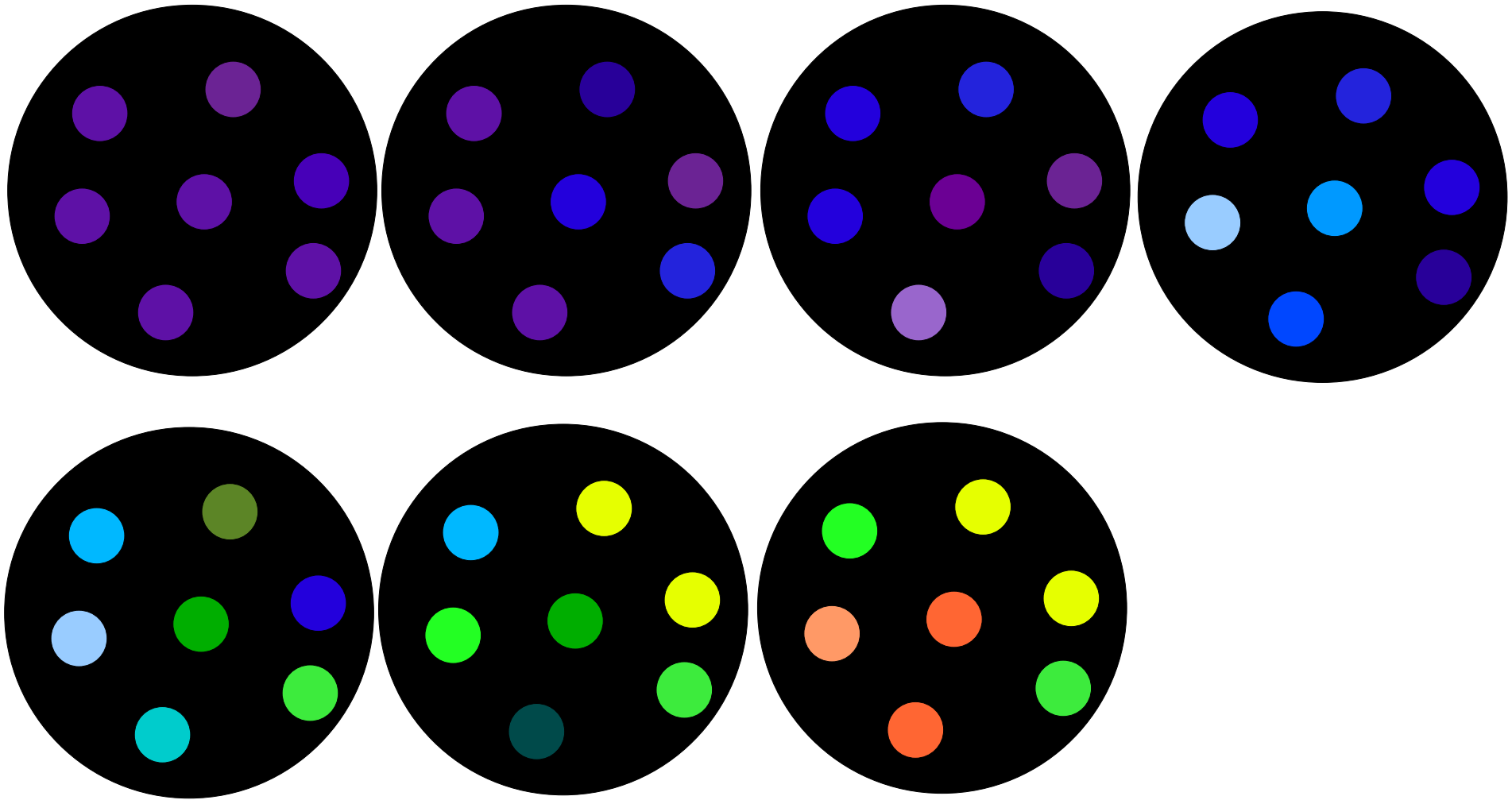
# Evolution of solutions



Florham Park, December 14, 2007

Optimization by random-key  
genetic algorithms

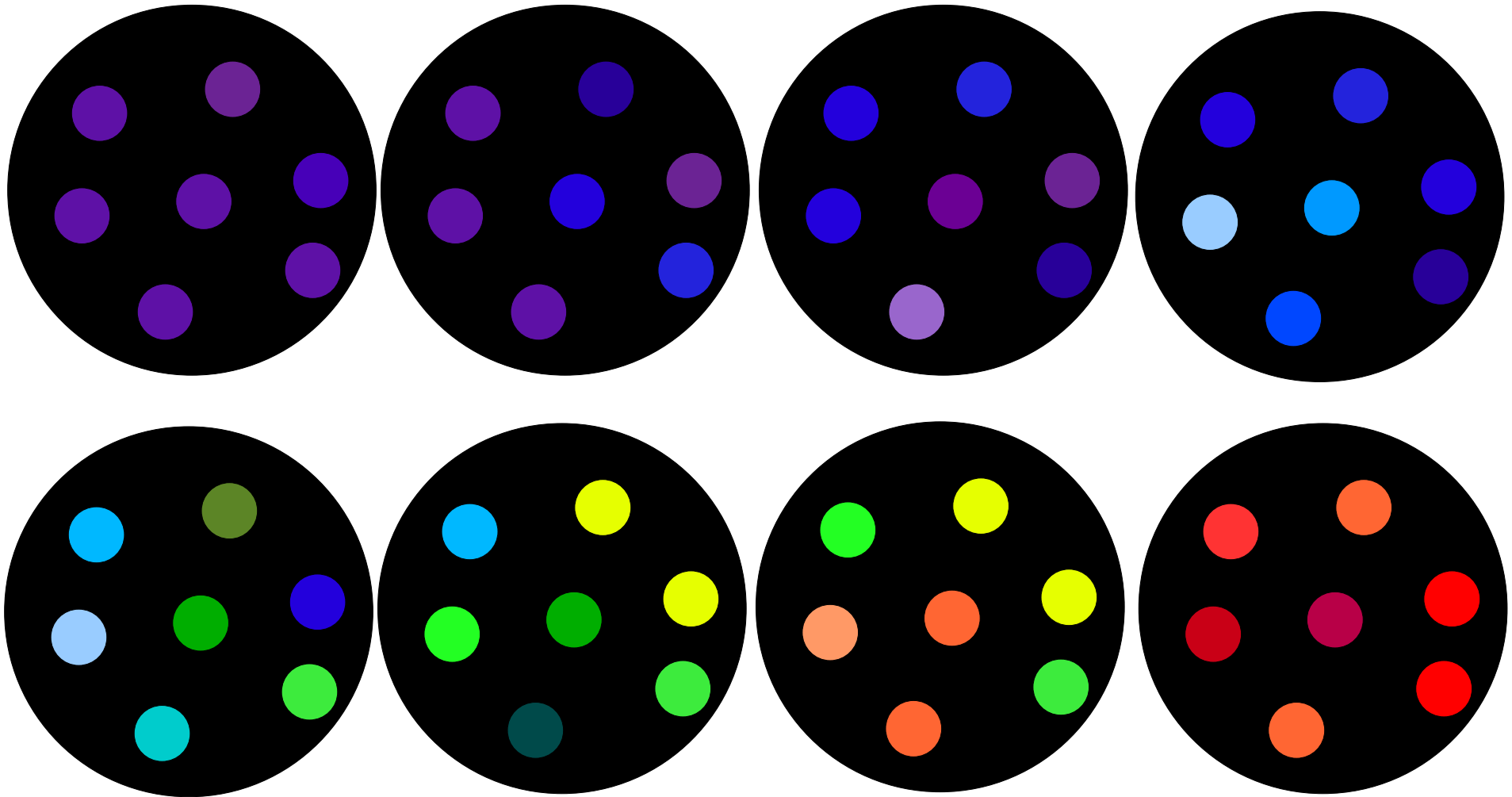
# Evolution of solutions



Florham Park, December 14, 2007

Optimization by random-key  
genetic algorithms

# Evolution of solutions



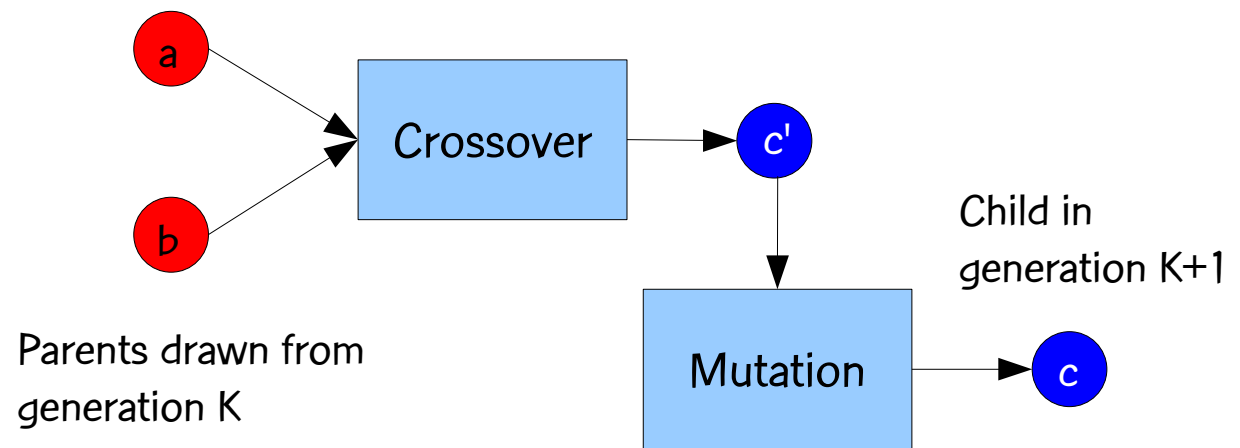
Florham Park, December 14, 2007

Optimization by random-key  
genetic algorithms

# GA lingo

- Population: set of individuals (solutions)
- Chromosome: string (encodes a solution)
- Gene: feature, character, detector (chromosomes are strings of genes)
- Allele: feature value
- Crossover: combination (mating) of two “parent” solutions to produce a “child” solution
- Mutation: perturbation of “child” solution

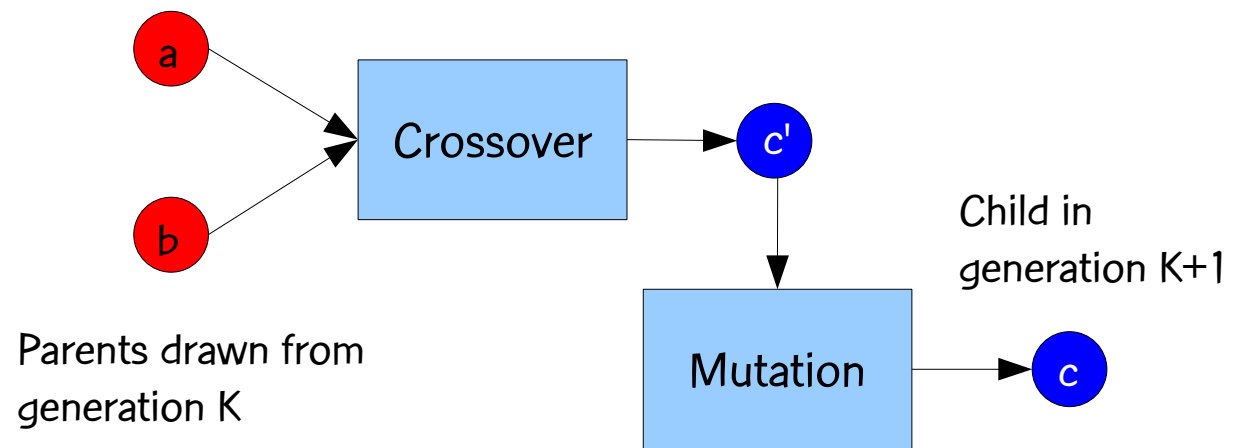
# A drawback of genetic algorithms



Given (feasible) parents “a” and “b” in generation K, **problem-dependent crossover and mutation operators** are needed to guarantee that child “c” in generation K+1 is also feasible.

Therefore, there is a **need for specialized representations as well as crossover and mutation operators** for each problem variation.

# A drawback of genetic algorithms



Given (feasible) parents “a” and “b” in generation K, **problem-dependent crossover and mutation operators** are needed to guarantee that child “c” in generation K+1 is also feasible.

Therefore, there is a **need for specialized representations as well as crossover and mutation operators** for each problem variation.

**A genetic algorithm with problem independent crossover and mutation operators could be more appealing.**

# 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 that gene to 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 that gene to child.

$$\begin{aligned} a &= ( 0.25, 0.19, 0.67, 0.05, 0.89 ) \\ b &= ( 0.63, 0.90, 0.76, 0.93, 0.08 ) \\ c &= ( 0.25 \quad \quad \quad ) \end{aligned}$$

# 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 that gene to child.

$$\begin{aligned} a &= ( 0.25, 0.19, 0.67, 0.05, 0.89 ) \\ b &= ( 0.63, 0.90, 0.76, 0.93, 0.08 ) \\ c &= ( 0.25, 0.90 \quad \quad \quad ) \end{aligned}$$

# 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 that gene to child.

$$\begin{aligned} a &= ( 0.25, 0.19, 0.67, 0.05, 0.89 ) \\ b &= ( 0.63, 0.90, 0.76, 0.93, 0.08 ) \\ c &= ( 0.25, 0.90, 0.76 \quad \quad \quad ) \end{aligned}$$

# 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 that gene to child.

$$\begin{aligned} a &= ( 0.25, 0.19, 0.67, 0.05, 0.89 ) \\ b &= ( 0.63, 0.90, 0.76, 0.93, 0.08 ) \\ c &= ( 0.25, 0.90, 0.76, 0.05 \quad ) \end{aligned}$$

# 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 that gene to 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

- 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 that gene to child.

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

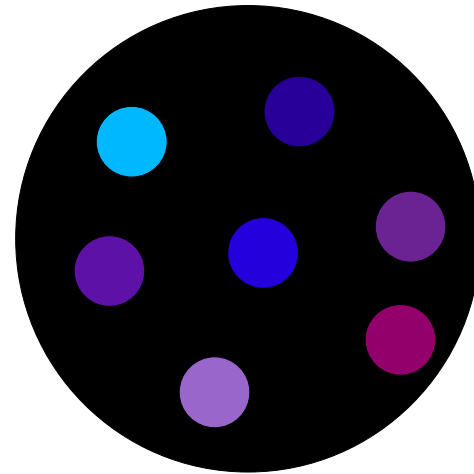
$$b = ( 0.63, 0.90, 0.76, 0.93, 0.08 )$$

$$c = ( 0.25, 0.90, 0.76, 0.05, 0.89 )$$

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

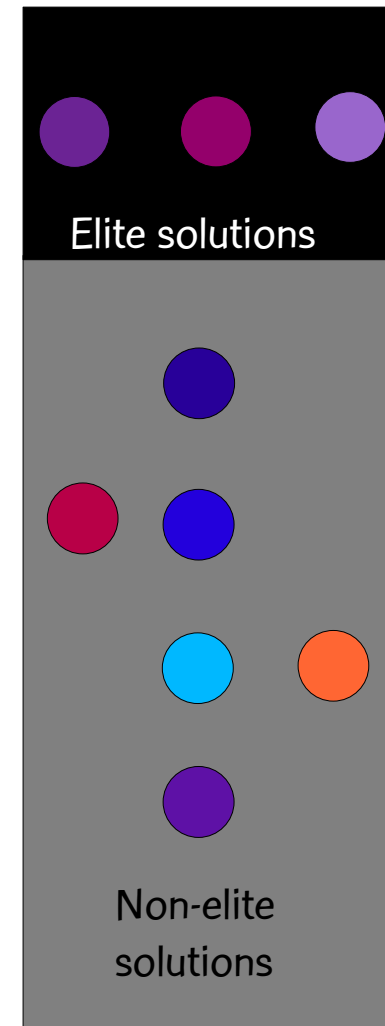
# GAs and random keys

- 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  $[0,1]$ .



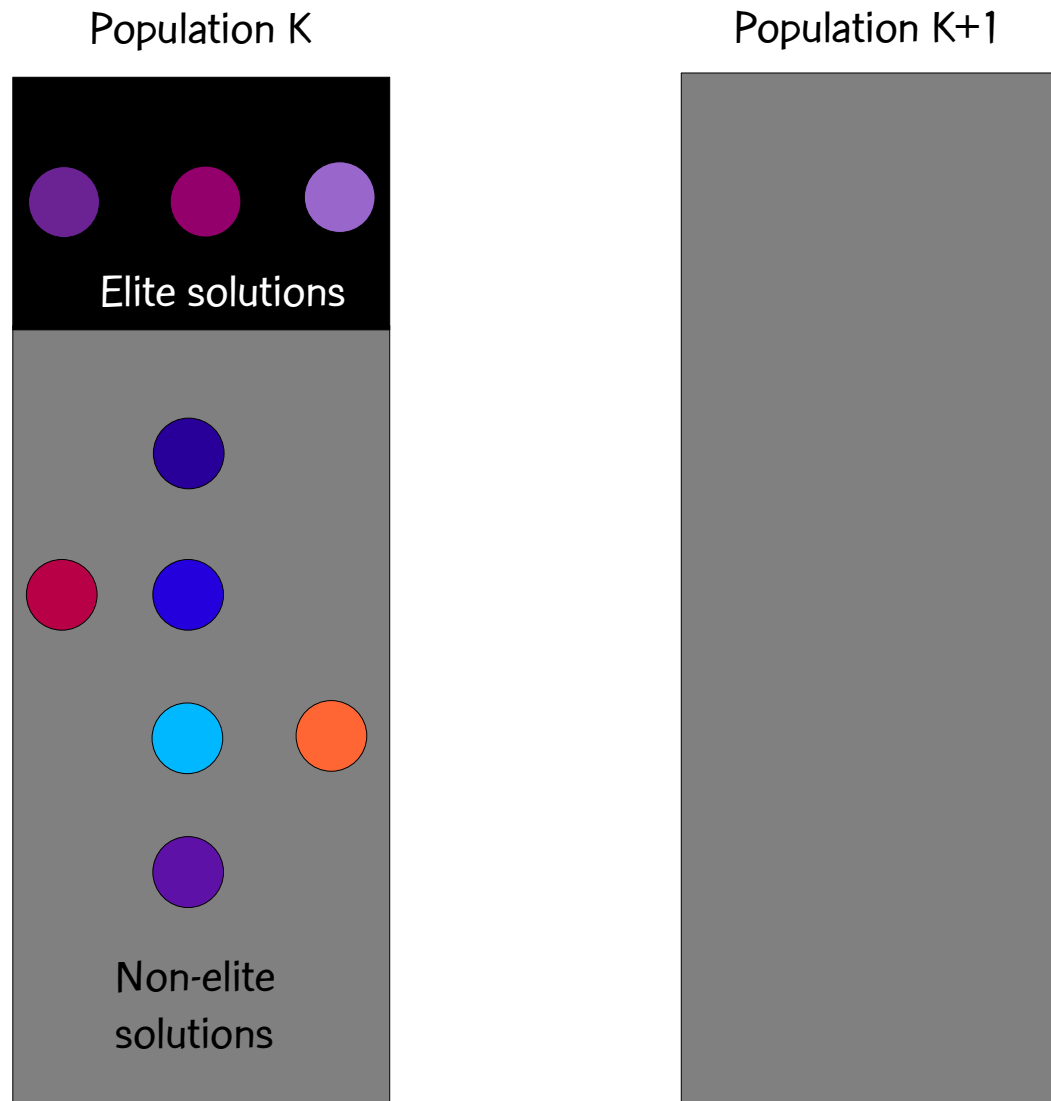
# GAs and random keys

- 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.



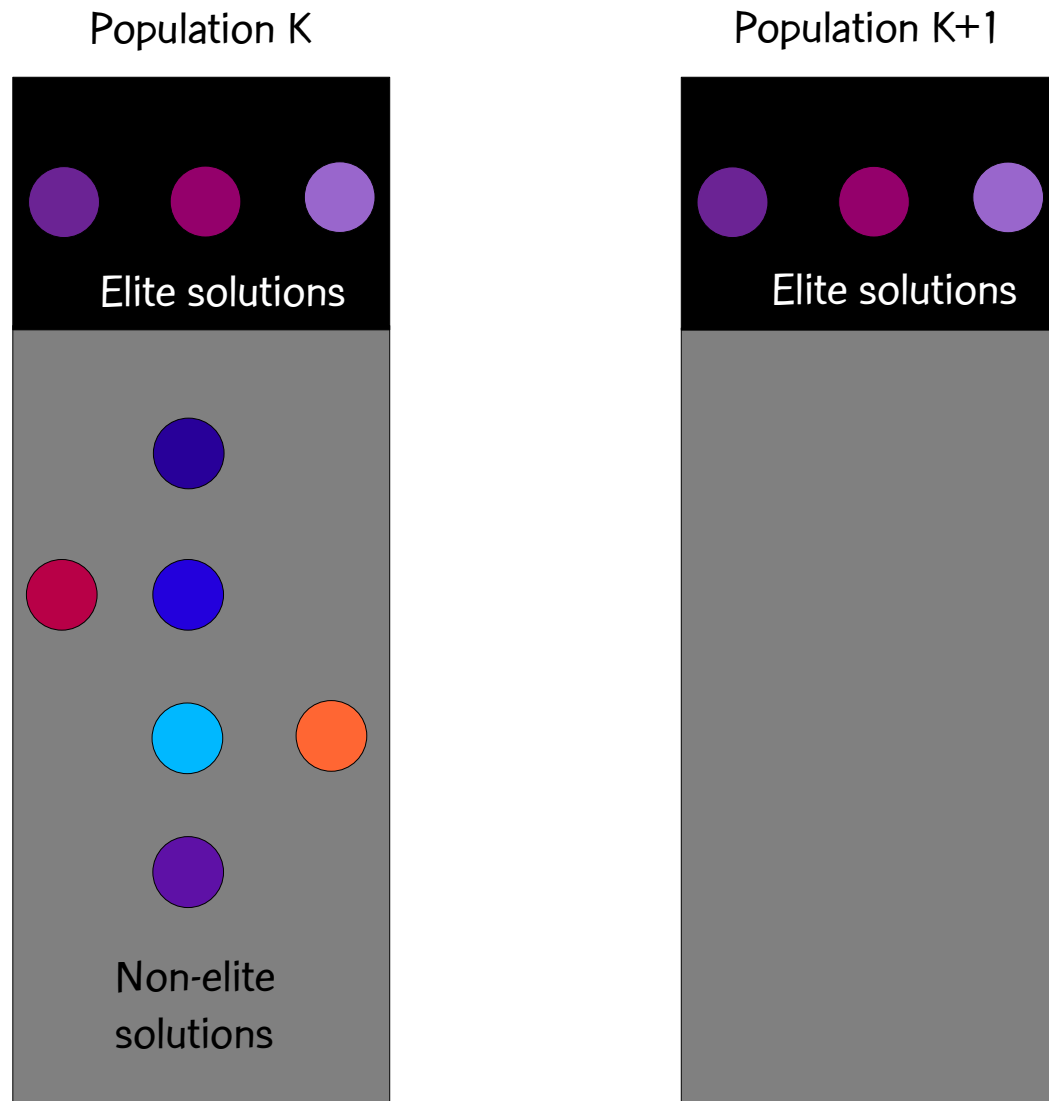
# GAs and random keys

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



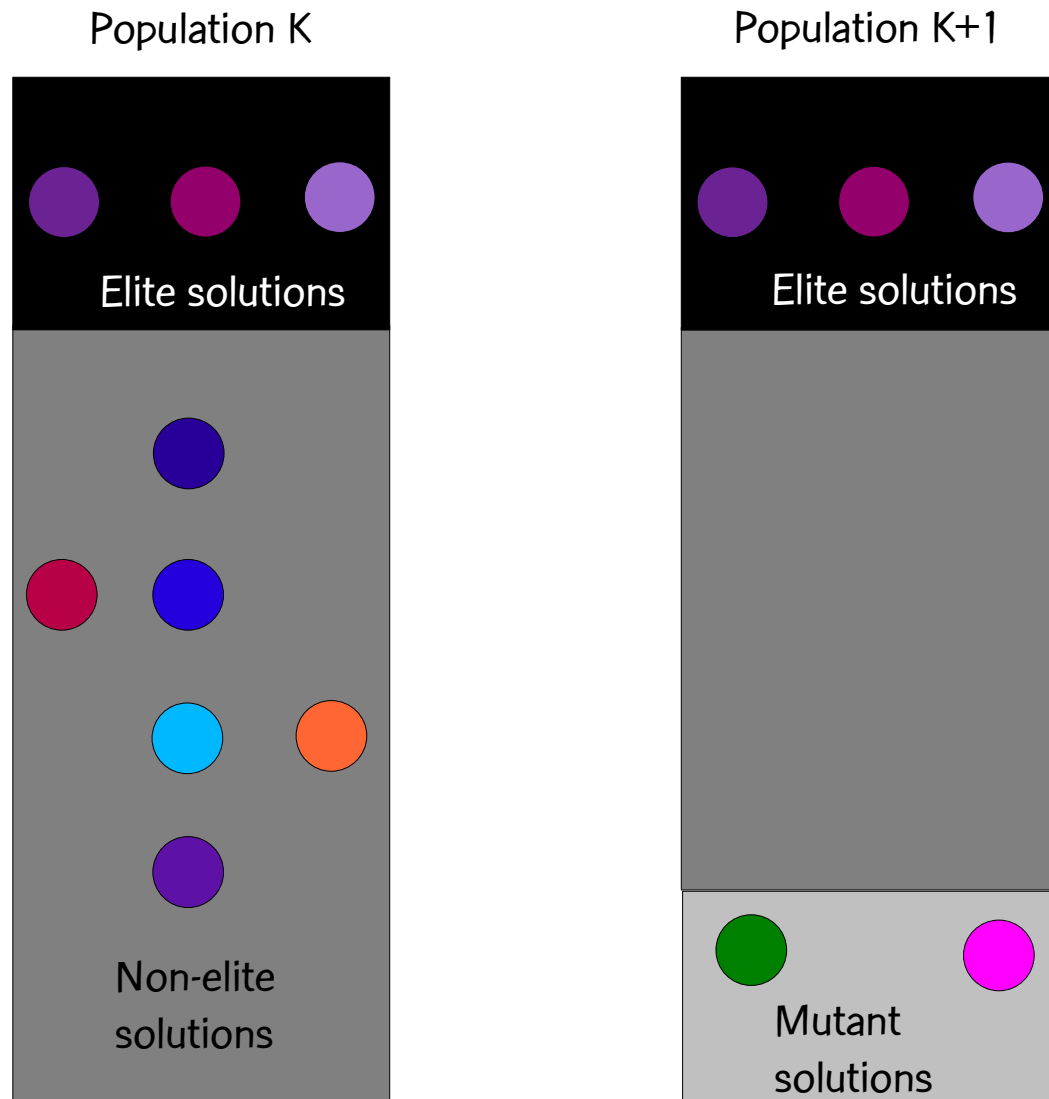
# GAs and random keys

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



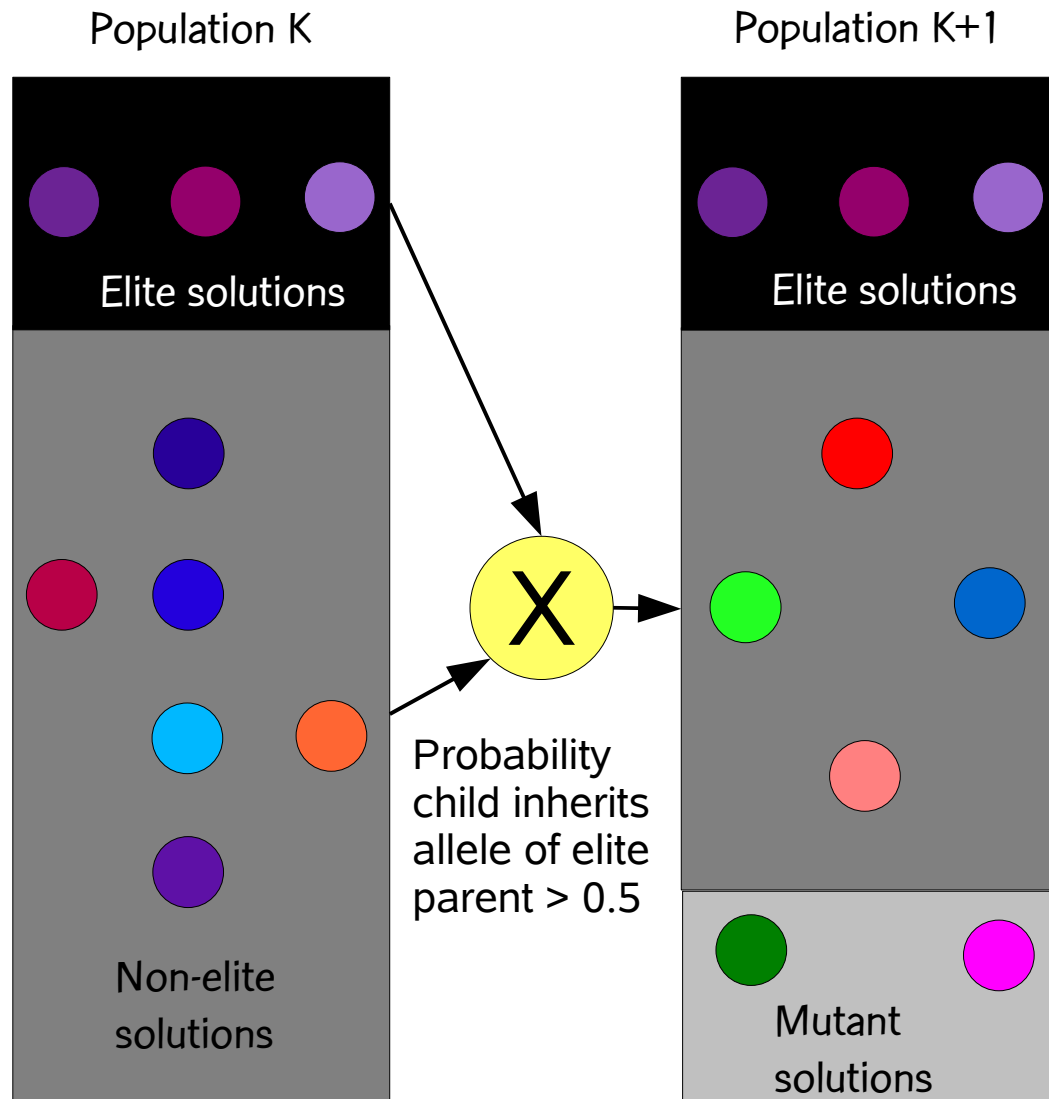
# GAs and random keys

- 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



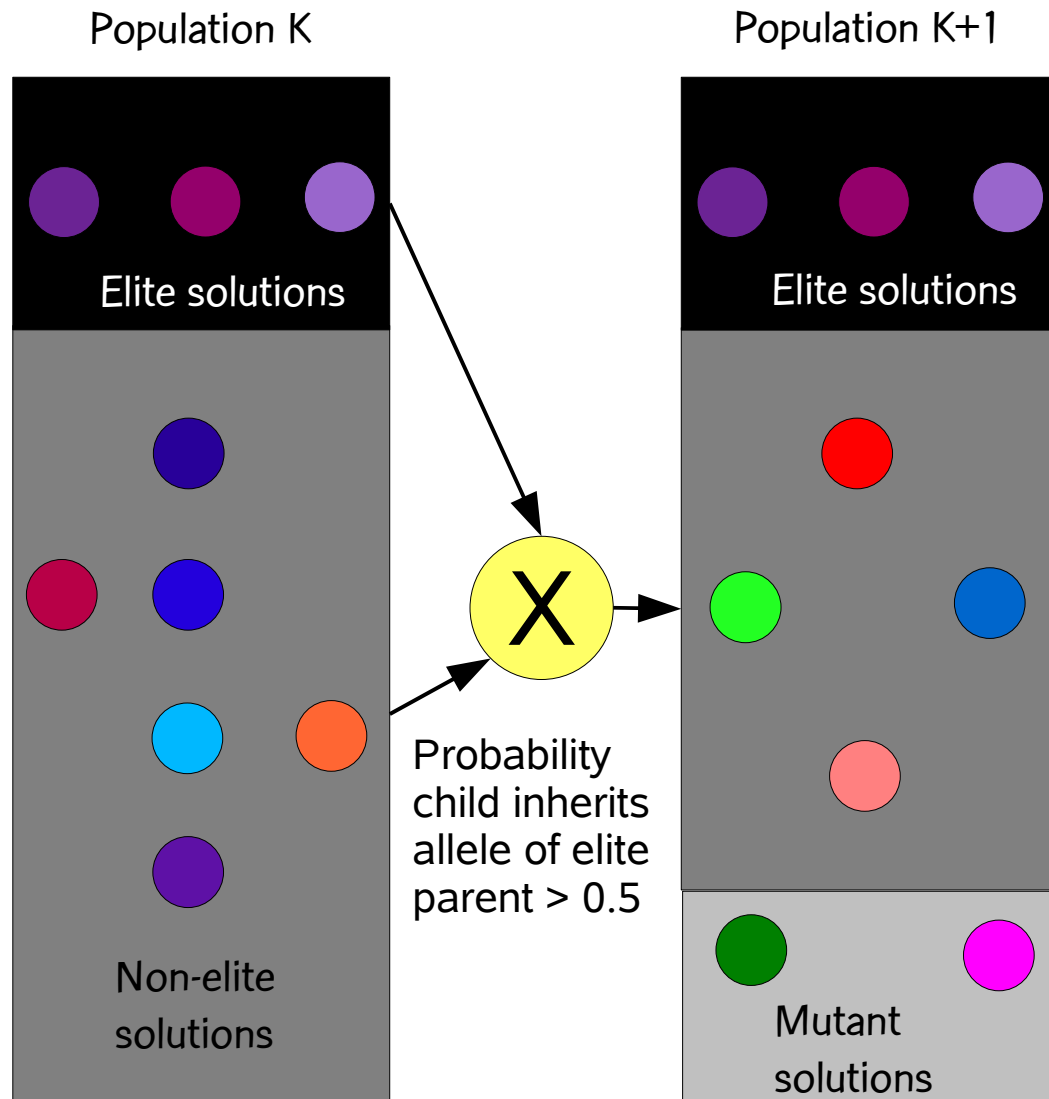
# GAs and random keys

- 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
  - While K+1 -th population  $< P$ 
    - Mate elite solution with non elite to produce child in population K+1. Mates are chosen at random.



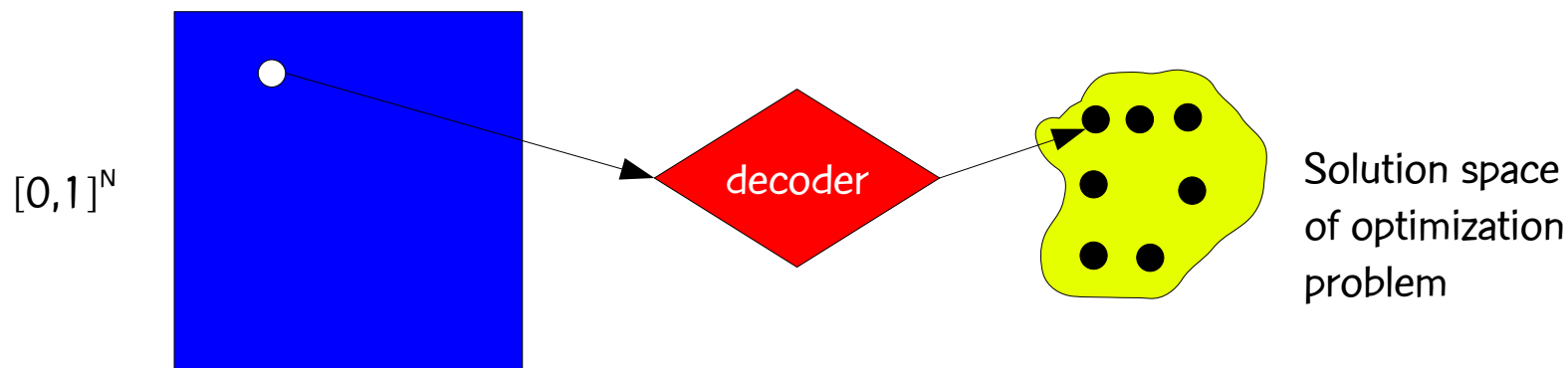
# GAs and random keys

- Introduced by Bean (1994) for sequencing problems.
- In practice, good choices are:
  - Elite solutions are top 10 to 20% of population
  - Mutants are 5 to 15% of population
  - Probability child inherits allele of elite parent is 60 to 80%



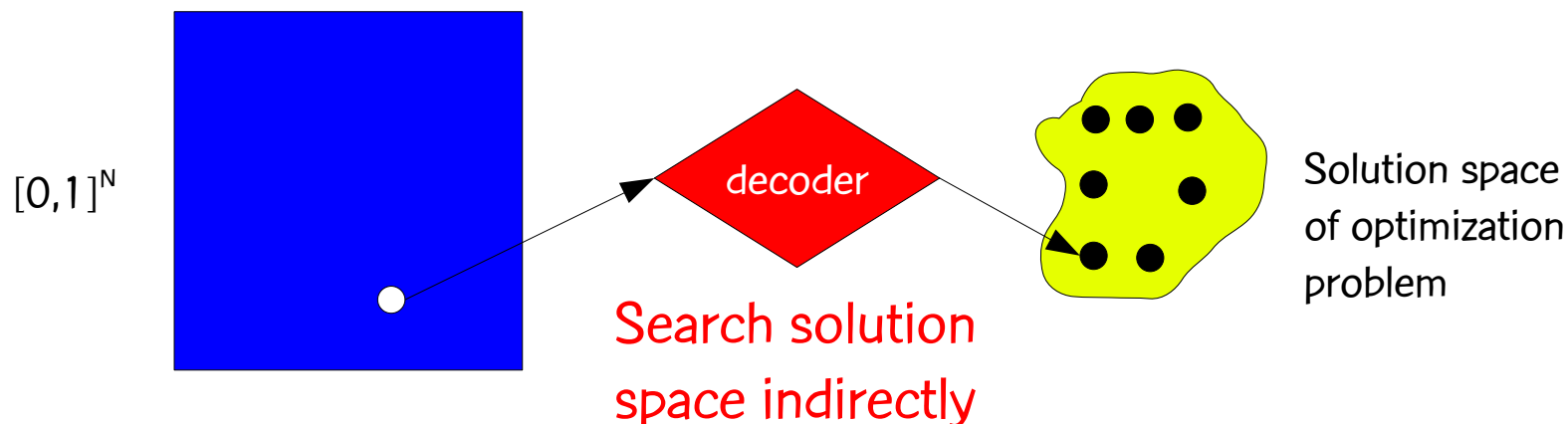
# Decoders

- 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.



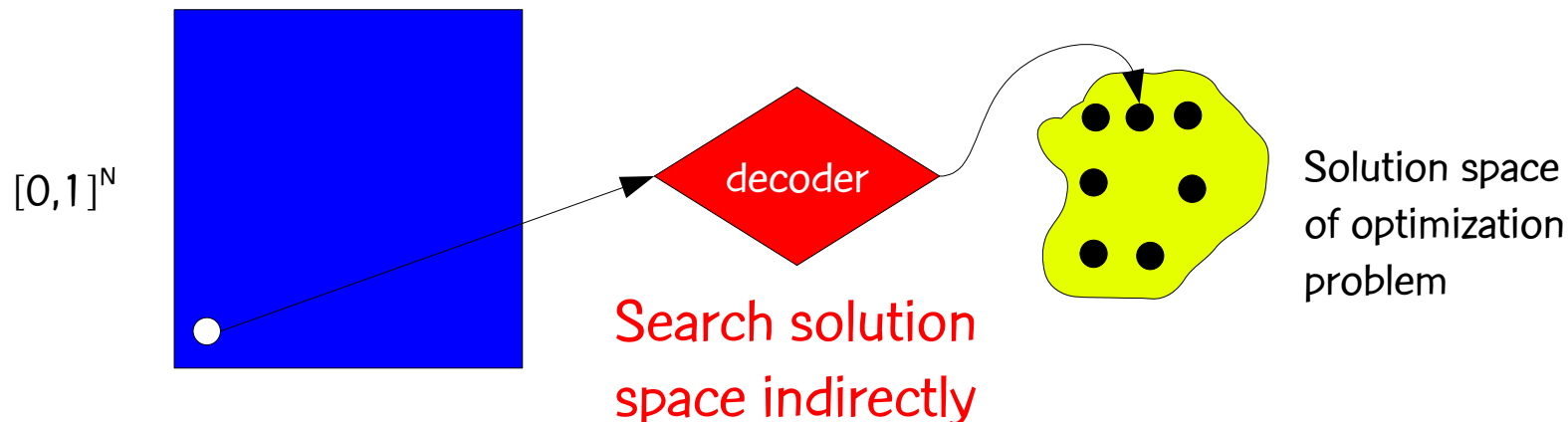
# Decoders

- 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.



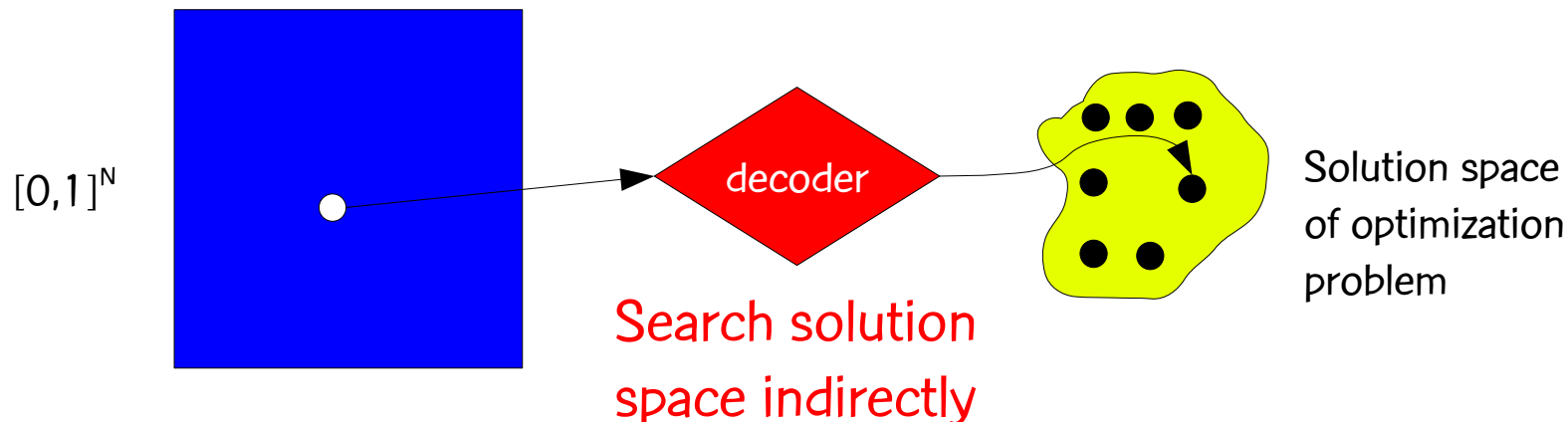
# Decoders

- 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.

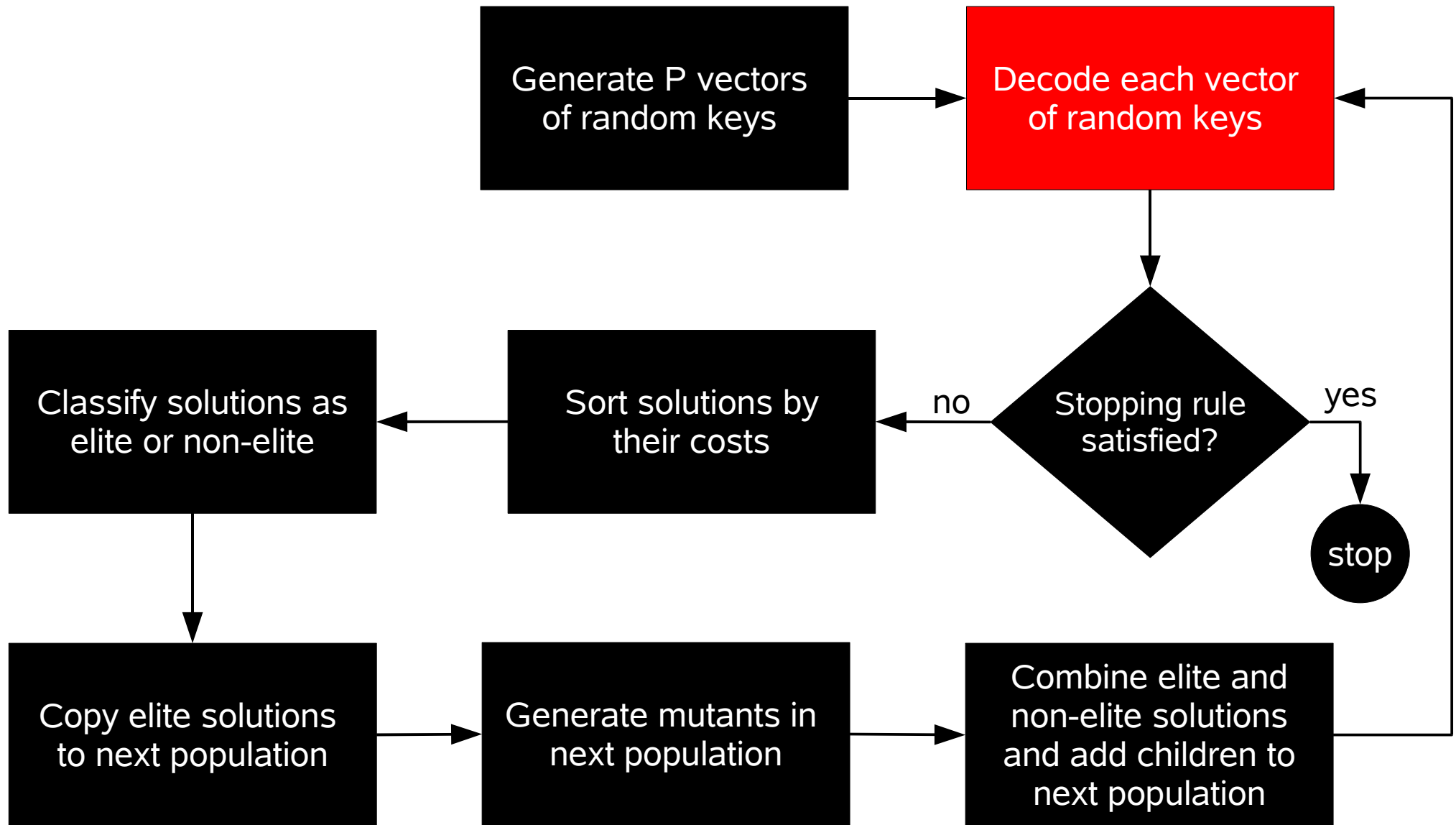


# Decoders

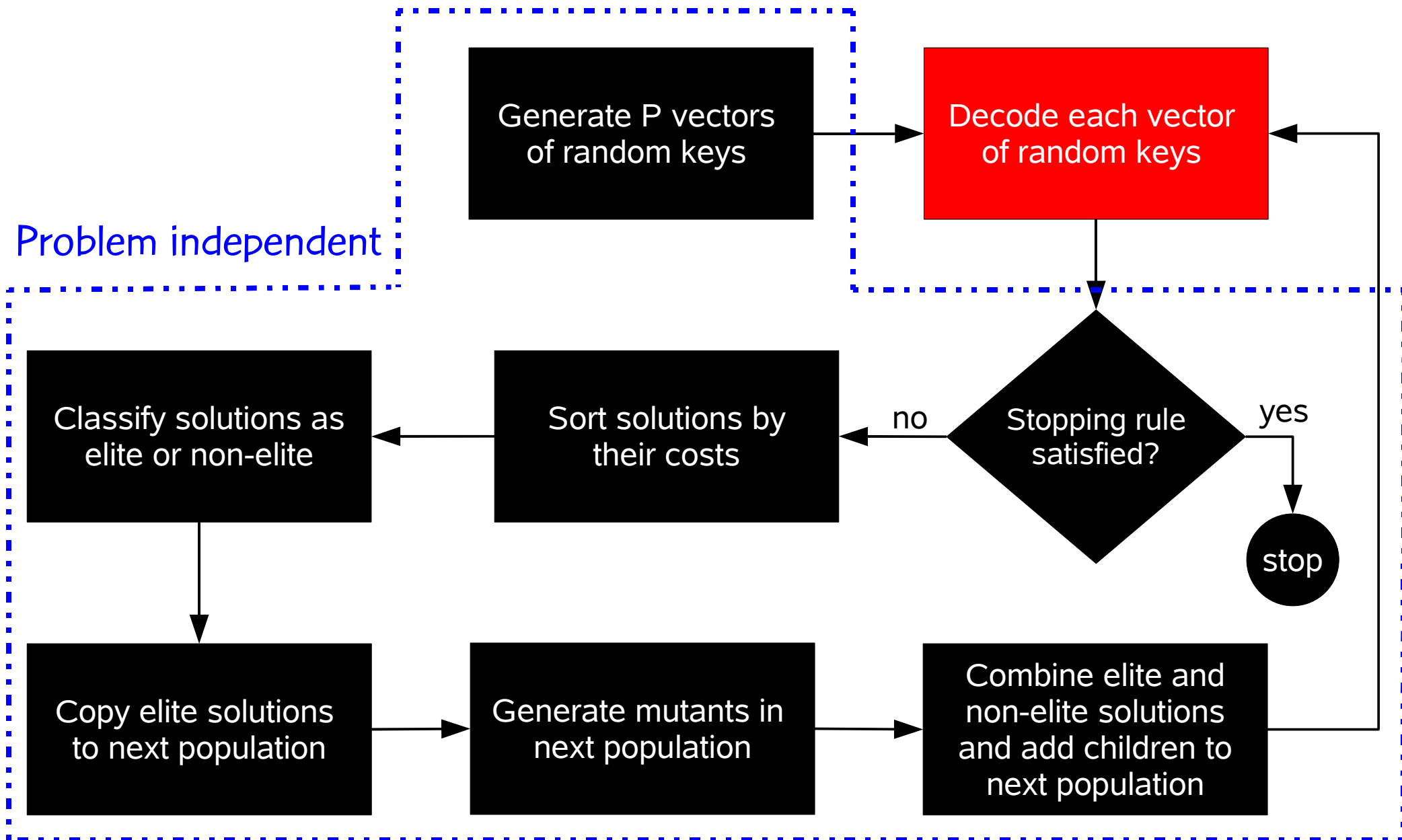
- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly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



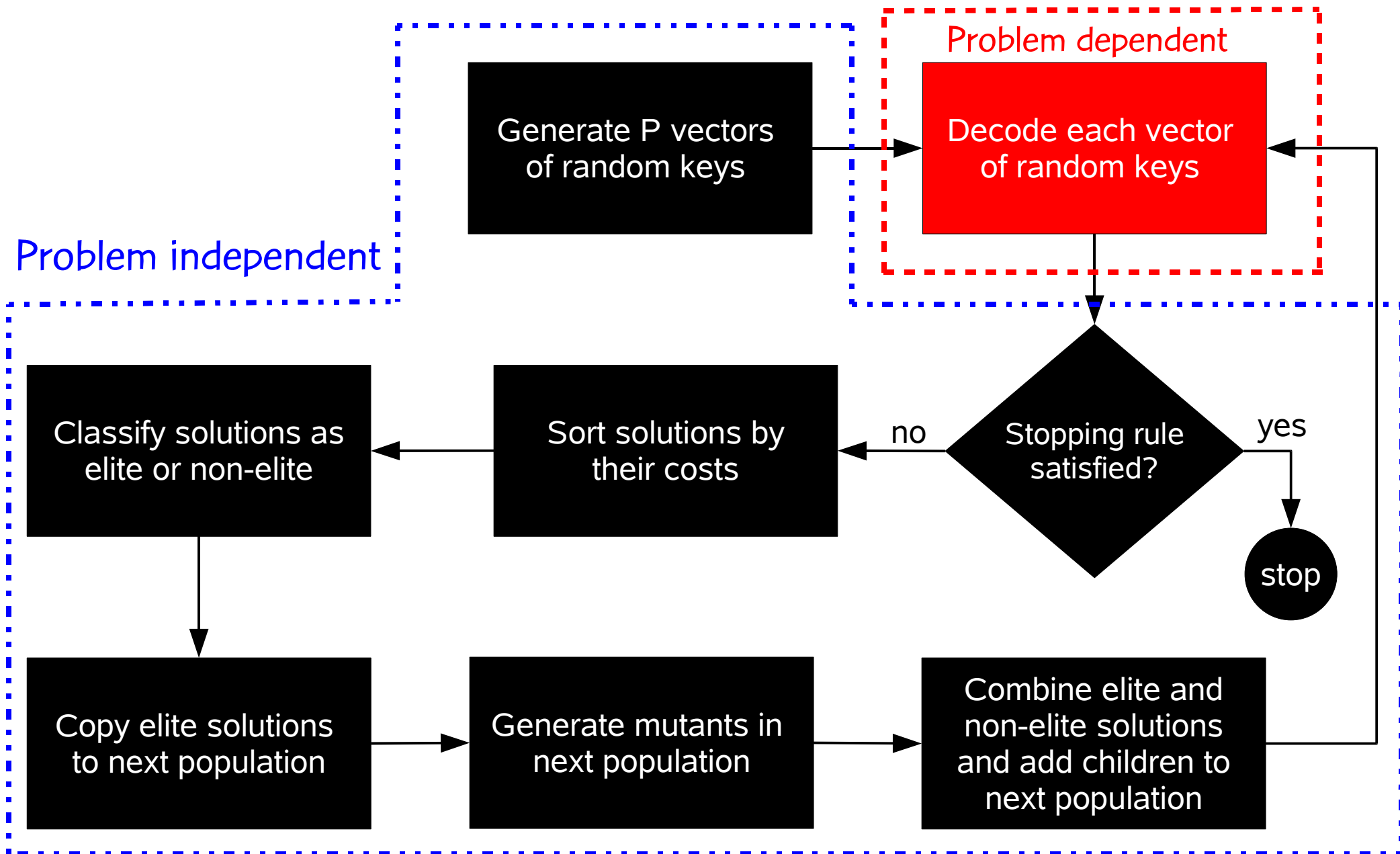
# Framework for random-key genetic algorithms



# Framework for random-key genetic algorithms



# Framework for random-key genetic algorithms



# Examples of optimization with random-key genetic algorithms

# Examples of optimization with random-key genetic algorithms

- Networking applications

- Covering by pairs: applied to optimal node placement for path disjoint network monitoring
- IP network routing: OSPF, DEFT
- Routing and wavelength assignment
- Survivable IP network design

- Operations research applications

- 2-dim packing
- Project, multi-project, and job-shop scheduling
- Manufacturing cell formation
- Congestion pricing in transportation networks

# Covering by pairs

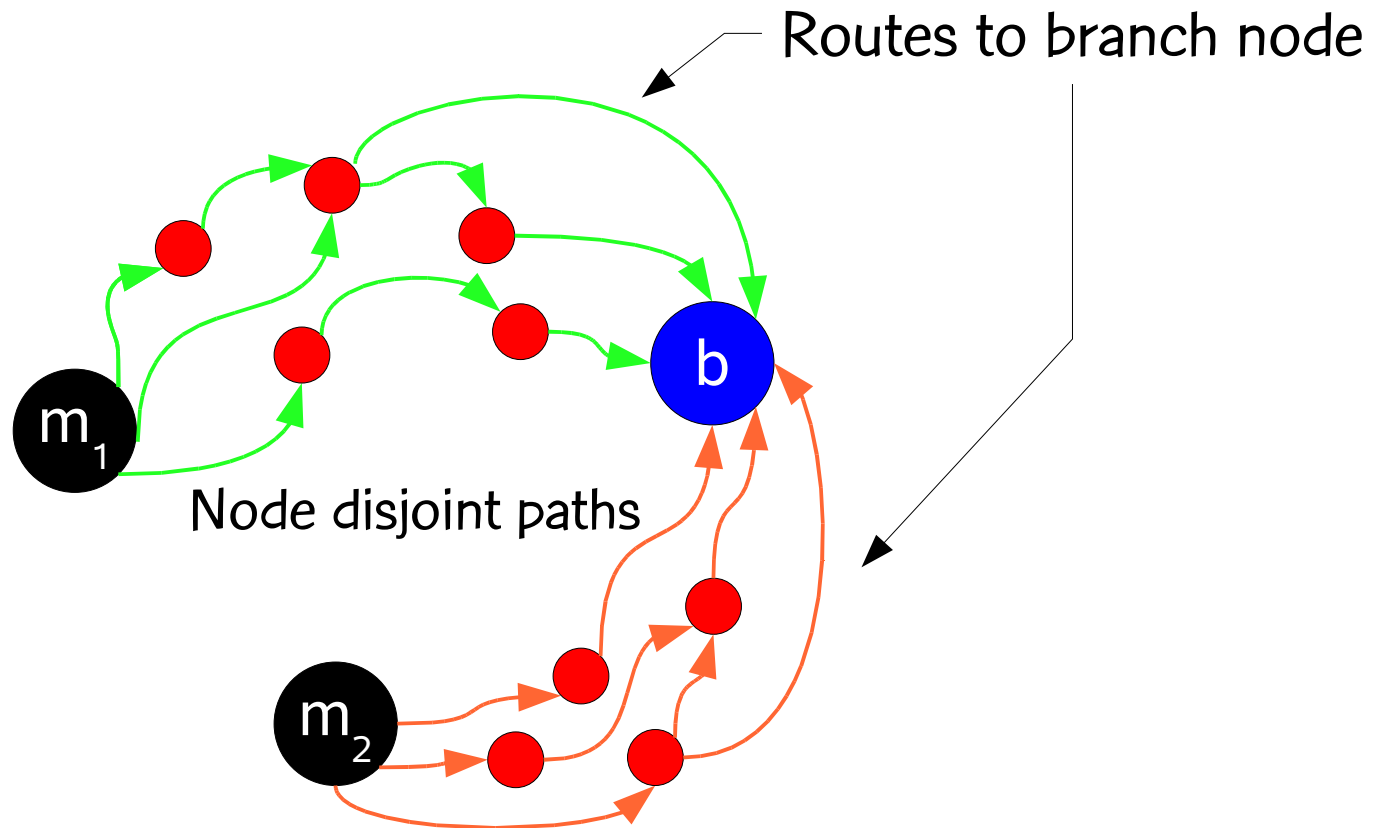
# Covering by pairs

- Application to problem in path disjoint network monitoring
- Breslau, Diakonikolas, Duffield, Gu, Hajiaghayi, Johnson, Karloff, R., Sen, and Towsley (2007)

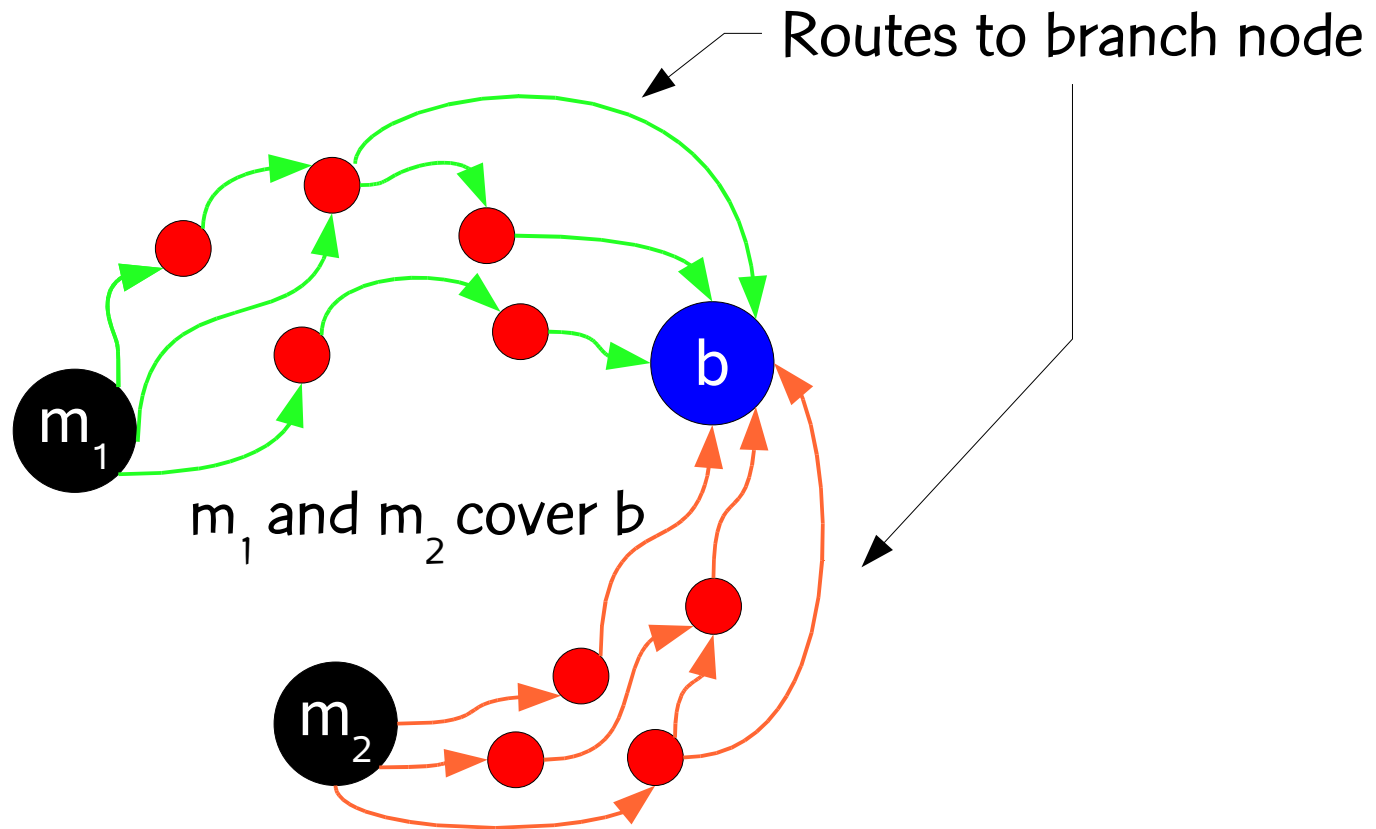
# Covering by pairs

- Given a directed network  $G = (N, V)$ , one or more paths between all pairs of nodes in  $N$ , a subset  $M$  of nodes called measurement nodes, a subset  $B$  of nodes called branch nodes, we wish to select the smallest number of measurement nodes such that for each branch node  $b \in B$ , either:
  - There are at least two measurement nodes  $m_1$  and  $m_2 \in M$  such that all routes from  $m_1$  to  $b$  are node disjoint with all routes from  $m_2$  to  $b$ , or
  - Node  $b$  is one of the selected measurement nodes ( $b$  covers itself)

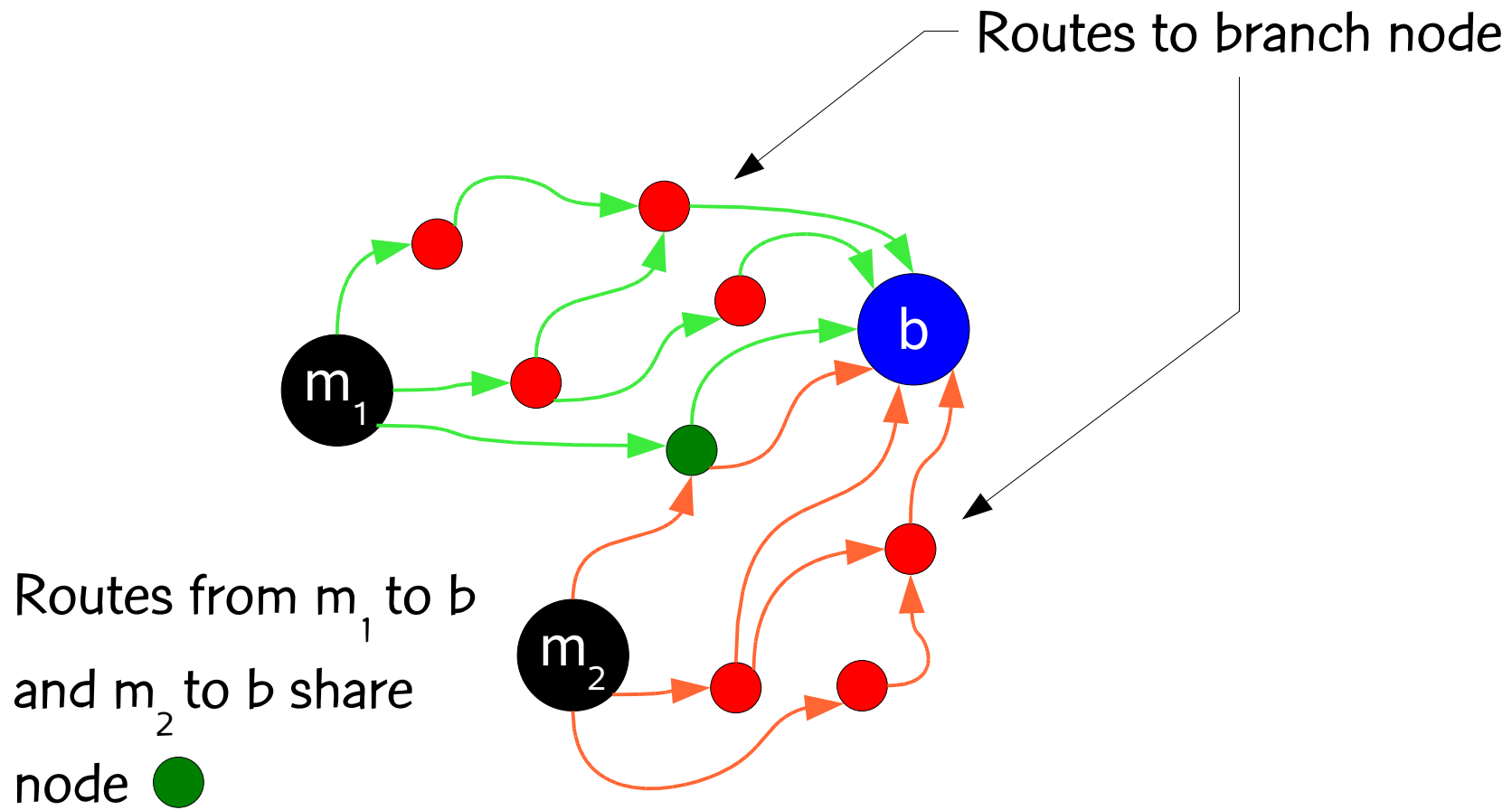
# Covering by pairs



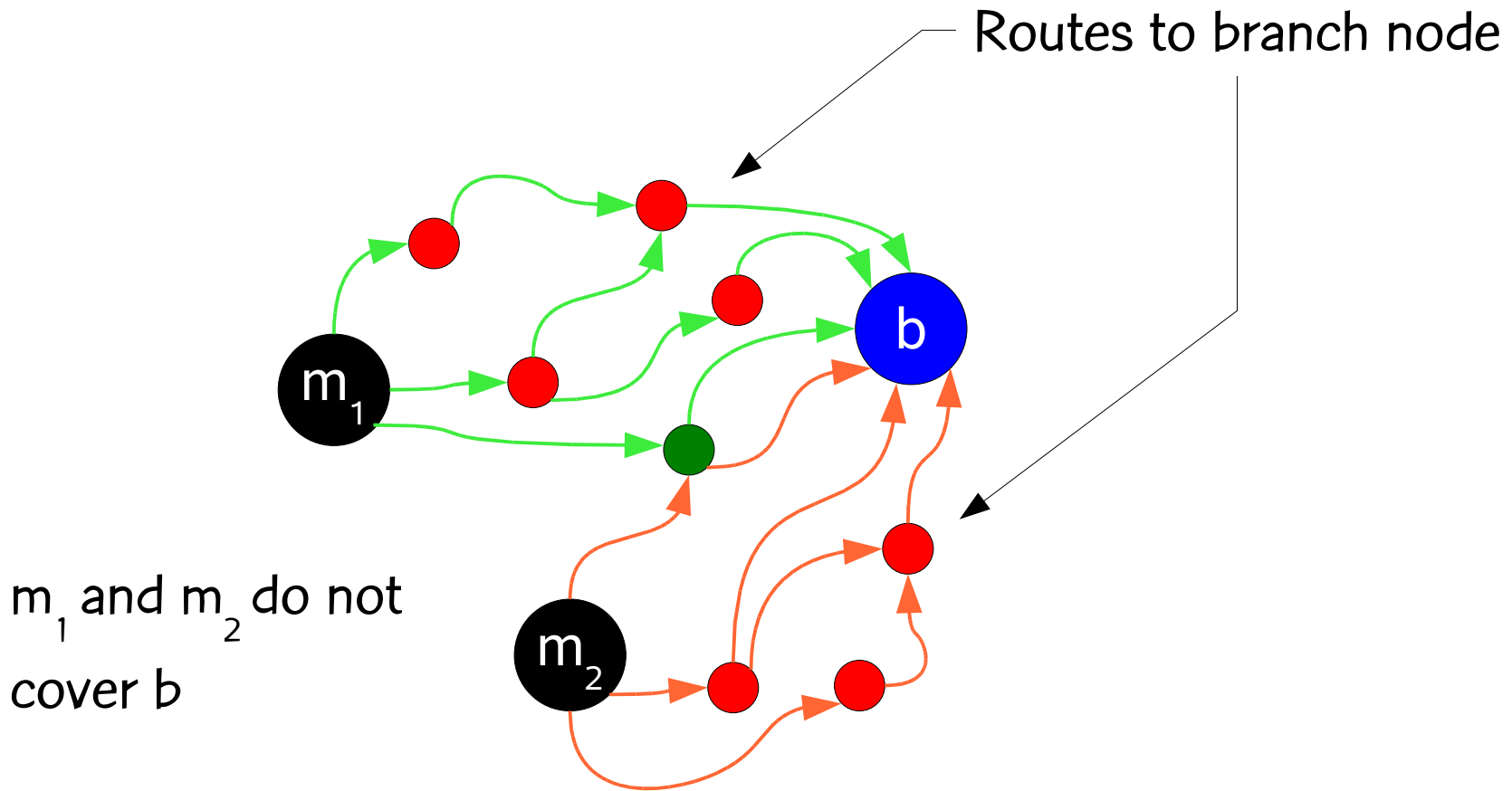
# Covering by pairs



# Covering by pairs



# Covering by pairs



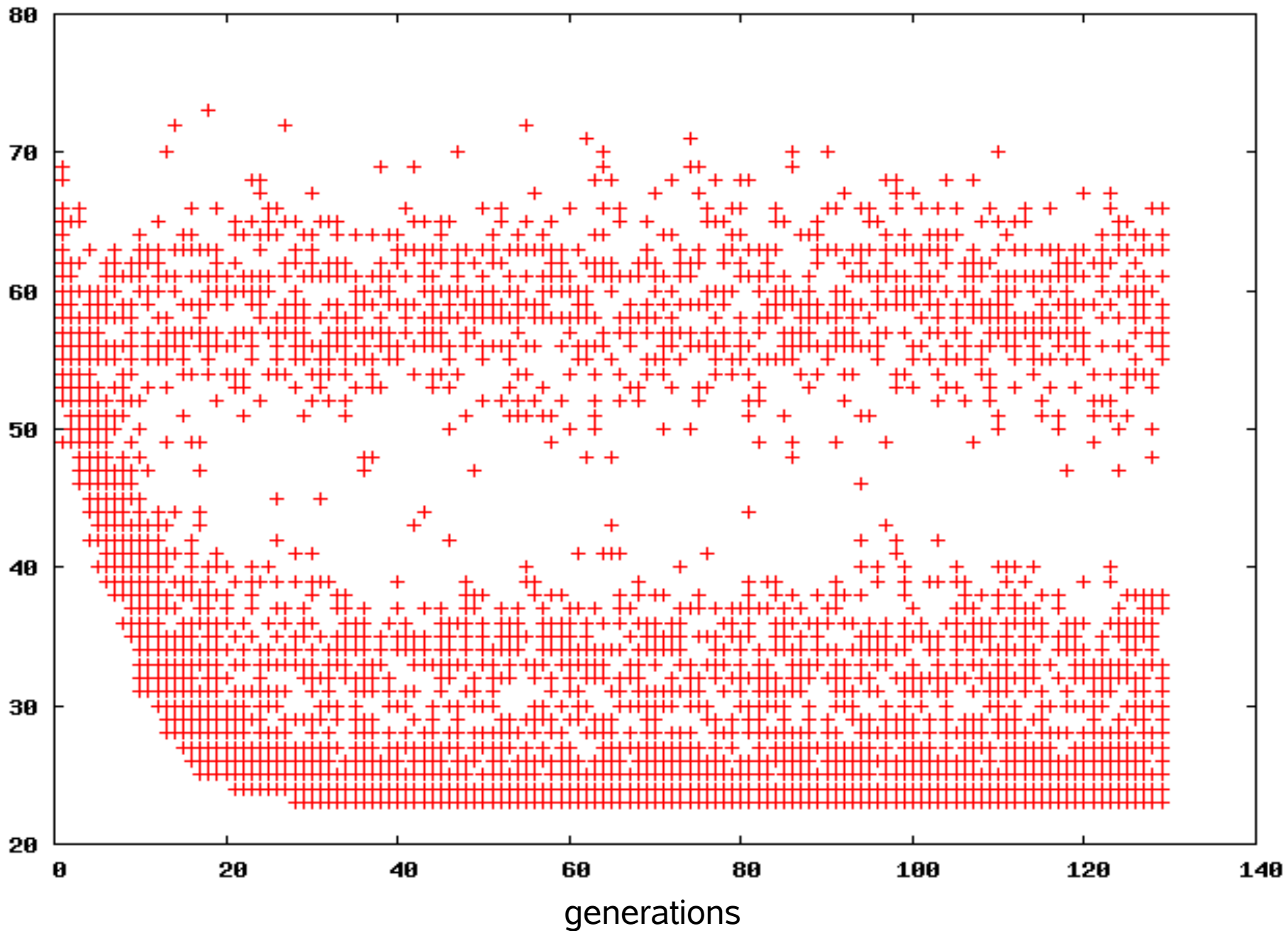
# Covering by pairs

- **Chromosome:**
  - A vector  $X$  of  $N$  random keys, where  $N$  is the number of measurement nodes. The  $i$ -th random key corresponds to the  $i$ -th measurement node.
- **Decoder:**
  - For  $i = 1, N$ : if  $X(i) > 0.5$  add  $i$ -th measurement node to solution
  - If solution is feasible, i.e. all branch nodes are covered: STOP
  - Else, apply greedy algorithm to cover uncovered branch nodes.

# Covering by pairs

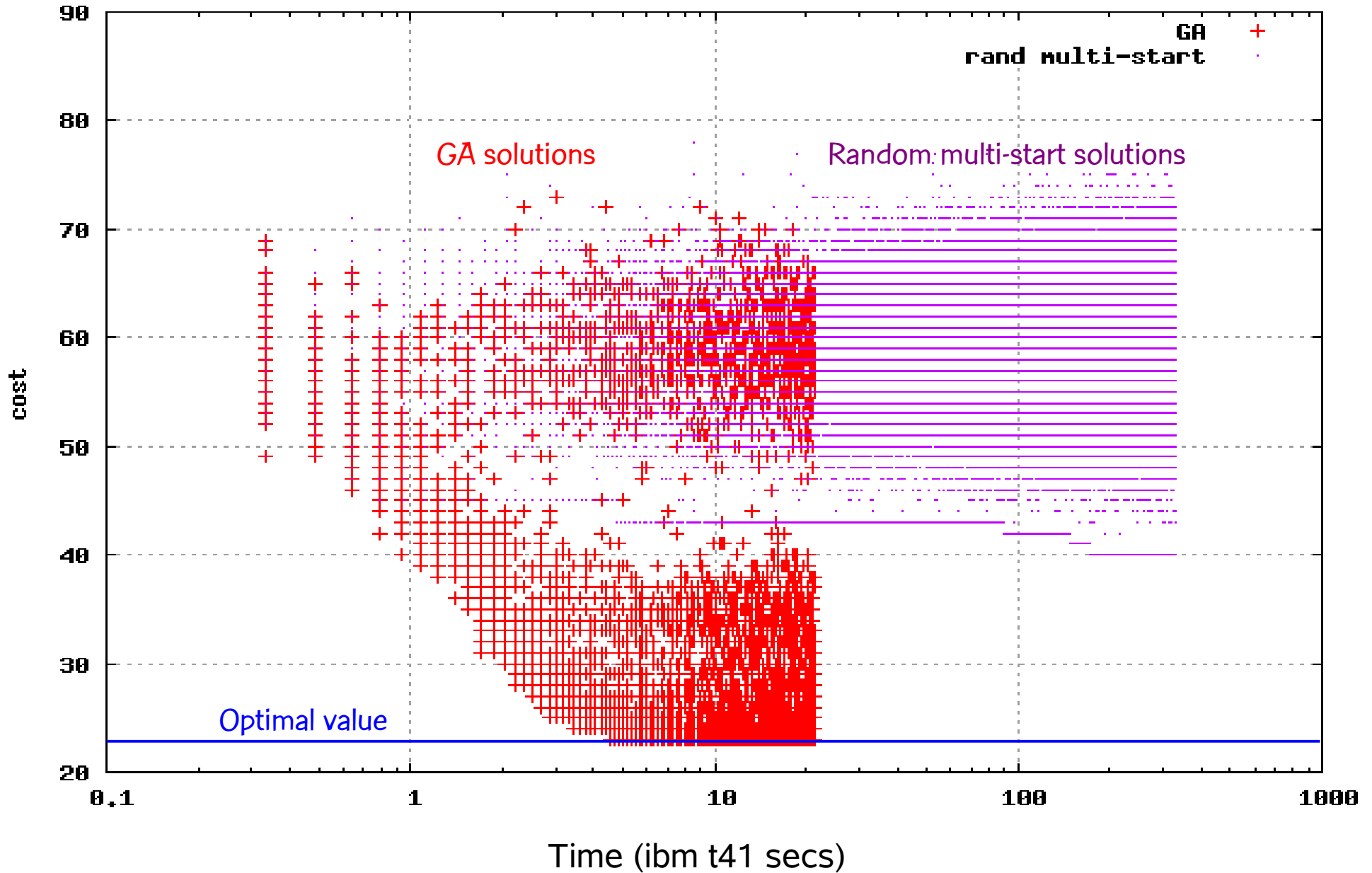
- **Size of population:**  $N$  (number of measurement nodes)
- **Size of elite set:** 15% of  $N$
- **Size of mutant set:** 10% of  $N$
- **Biased coin probability:** 70%
- **Stop** after  $N$  generations without improvement of best found solution

solution

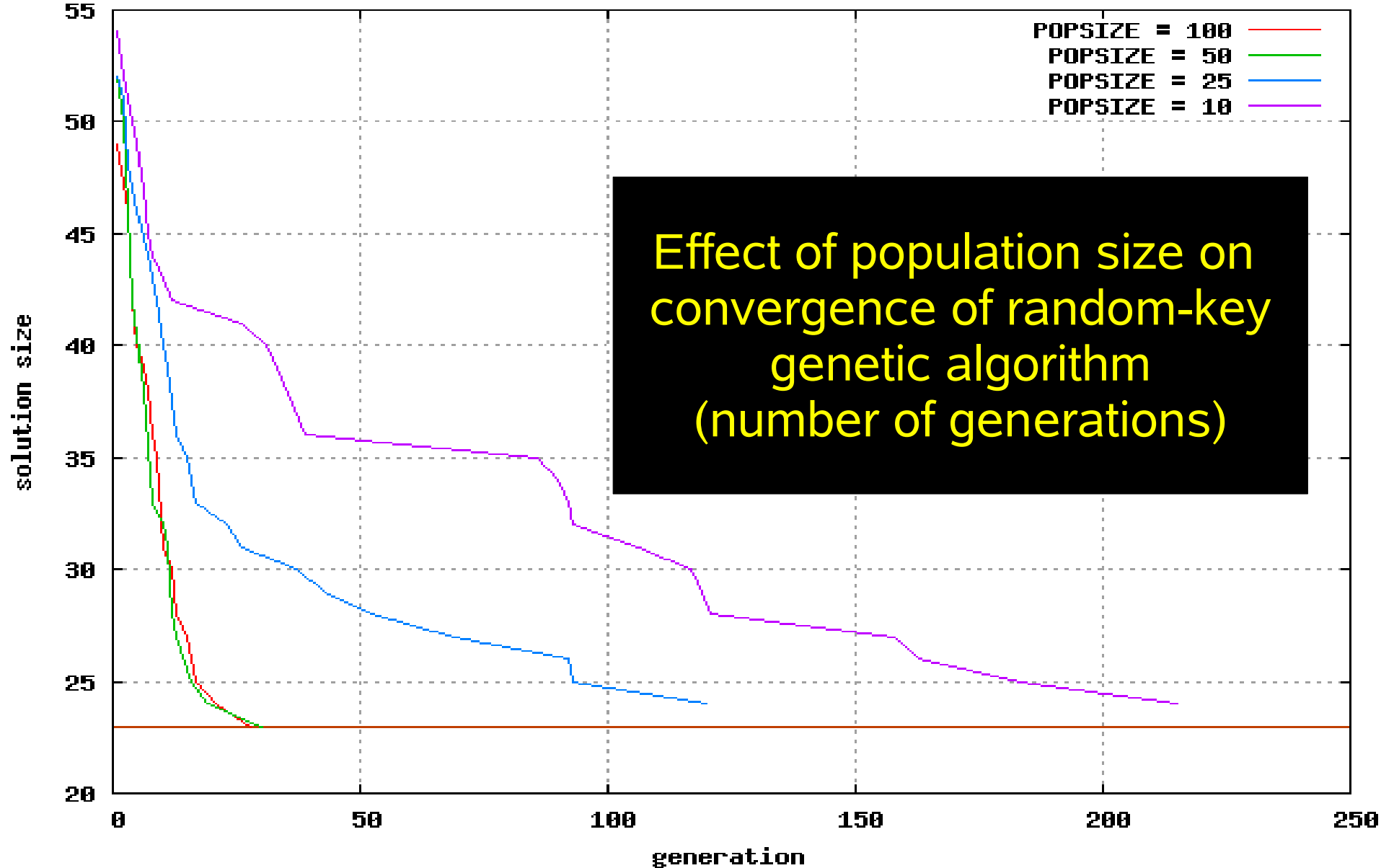


solution

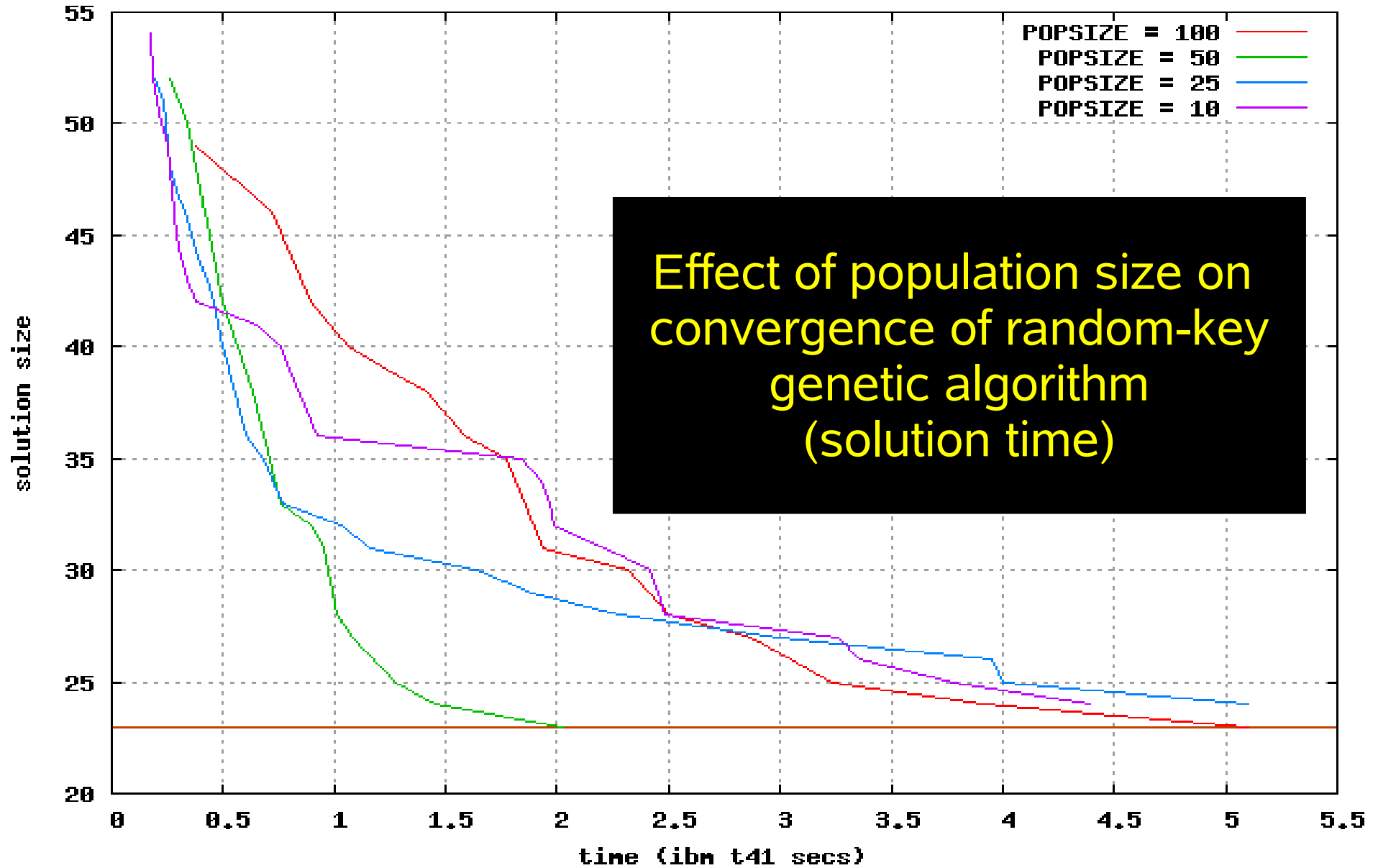
n100-i2-m100-b100: GA and random multi-start iterates



n100-i2-m100-b100.dat with POPSIZE = 100, 50, 25, 10

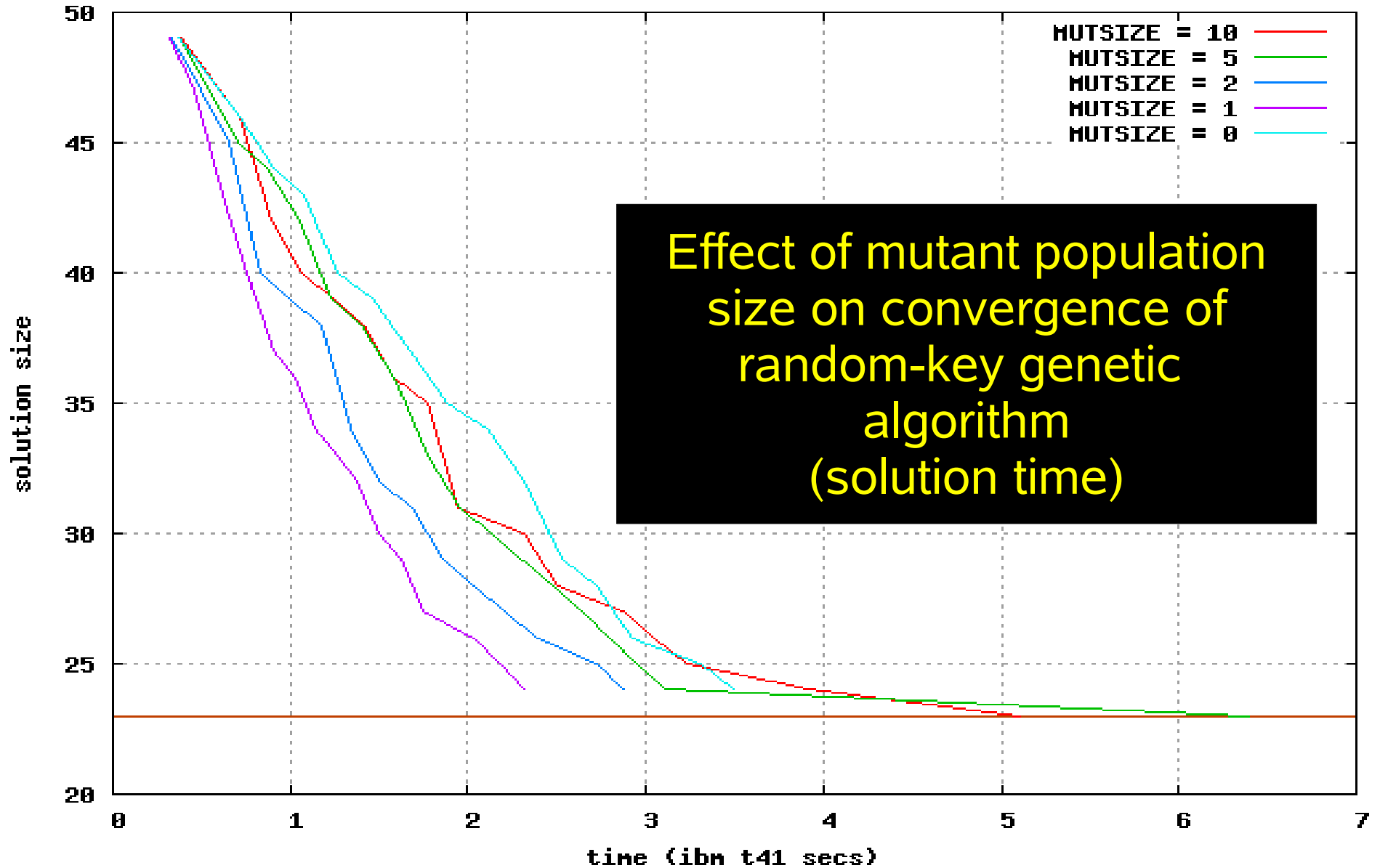


n100-i2-m100-b100.dat with POPSIZE = 100, 50, 25, 10

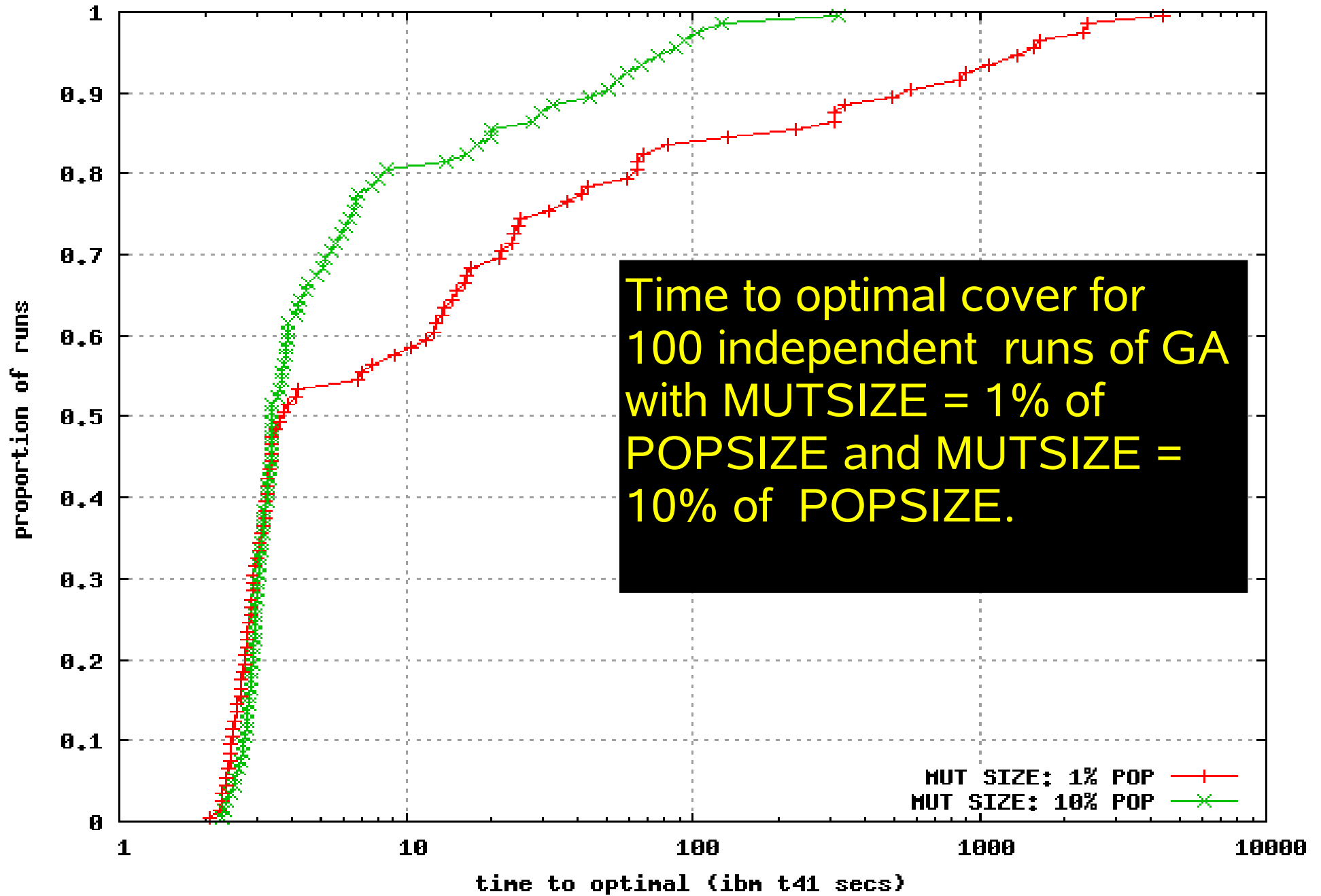


Effect of population size on convergence of random-key genetic algorithm (solution time)

n100-i2-n100-b100.dat with MUTSIZE = 10, 5, 2, 1, 0



Effect of mutant population size on convergence of random-key genetic algorithm (solution time)



# Experimental results

- 560 instances, with 25, 50, 100, 190, 220, 250, 300, and 558 nodes.
- 324 of these 560 were solved optimally with CPLEX. Running GA a single time, we found optimal solutions in 318 of these instances.
- Of the 236 that CPLEX could not solve, GA matched a lower bound in 166.
- In all, the GA found optimal solutions for 484 of the 560 instances (86.4%)

# Experimental results

- The paper describes the double hitting set heuristic (HH) proposed by H. Karloff. This heuristic makes use of the OSPF paths and is very fast and effective.
- In 482 of the 560 instances (86.1%) the GA and HH found solutions with the same cost.
- In 68 instances (12.1%) GA found a better solution than HH.
- In 10 instances (2%) HH found a better solution than GA.
- In only 12 instances (2.1%) was the solution found by GA not minimal.

# Routing in IP networks

# Routing in IP networks

- Protocol: In OSPF, traffic is routed on shortest weight path from origination router to destination router.
- Splitting: If more than one link out of a router is on a shortest weight path, traffic is evenly distributed on those links.
- Weight setting problem: Determine OSPF weights such that if traffic is routed according to OSPF protocol, network congestion is minimized.

# Minimization of congestion

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

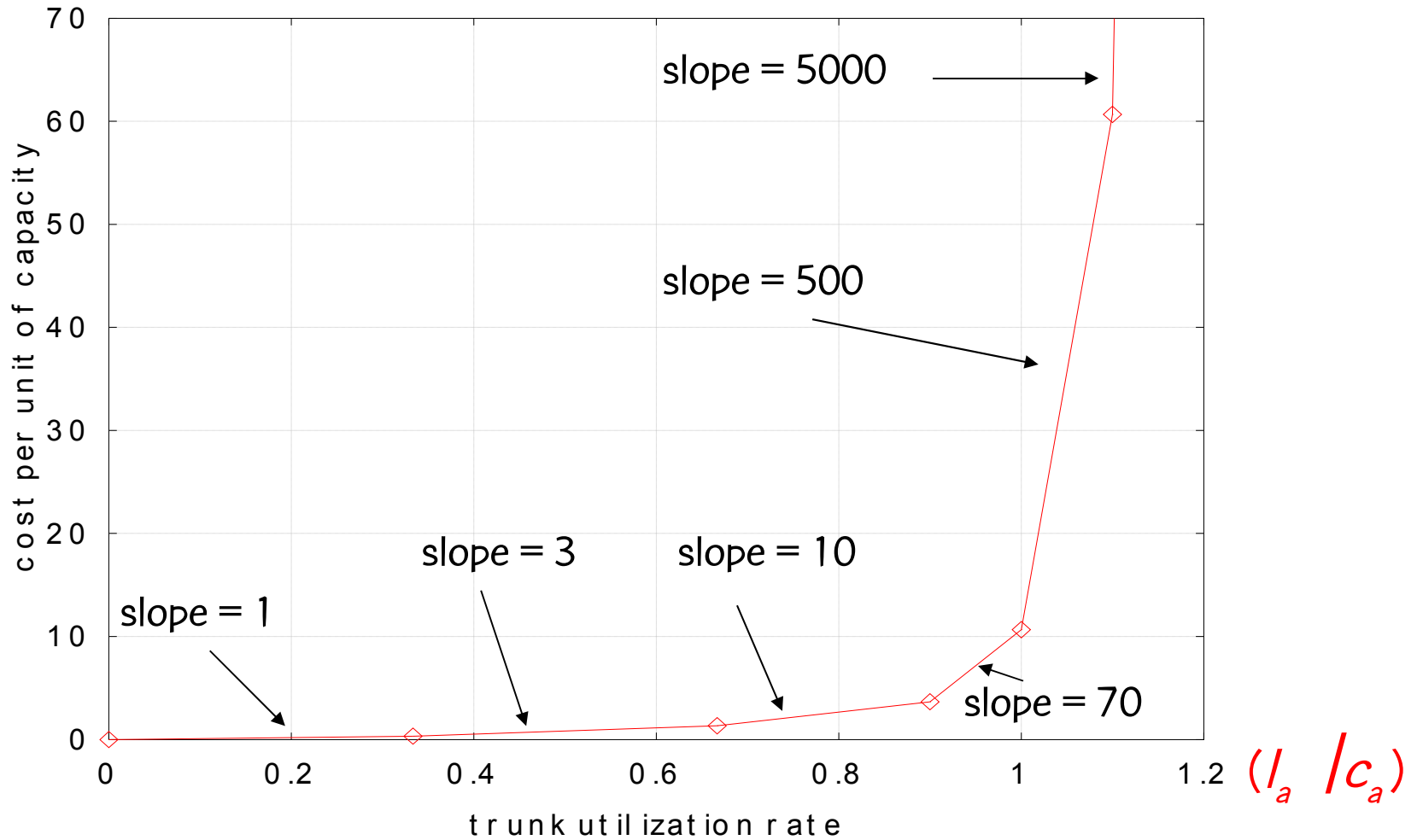
$$\Phi = \Phi_1(l_1) + \Phi_2(l_2) + \dots + \Phi_{|\bar{A}|}(l_{|\bar{A}|})$$

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

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

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

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

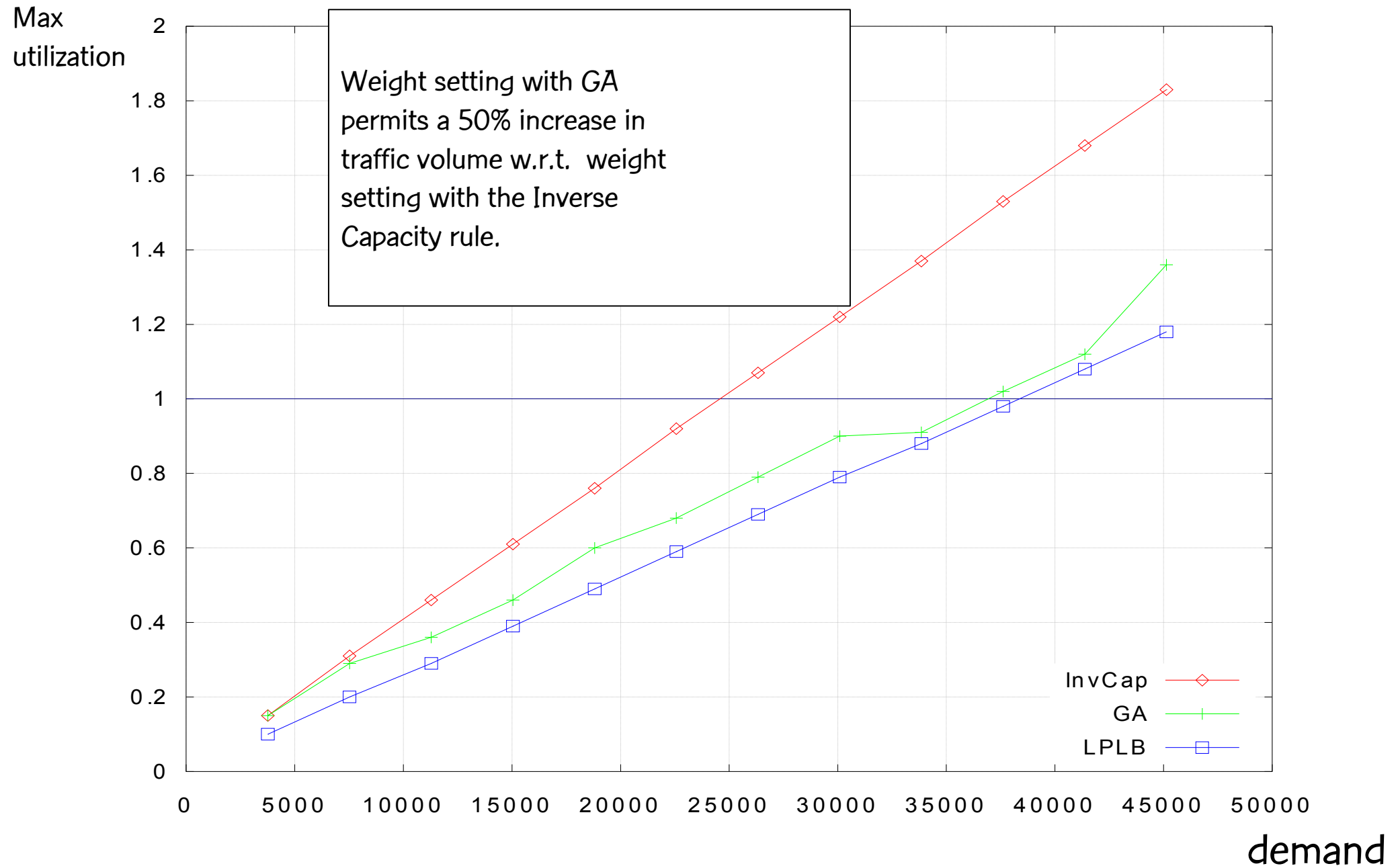


# Routing in IP networks

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

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

# AT&T Worldnet backbone network (90 routers, 274 links)



# Routing in IP networks

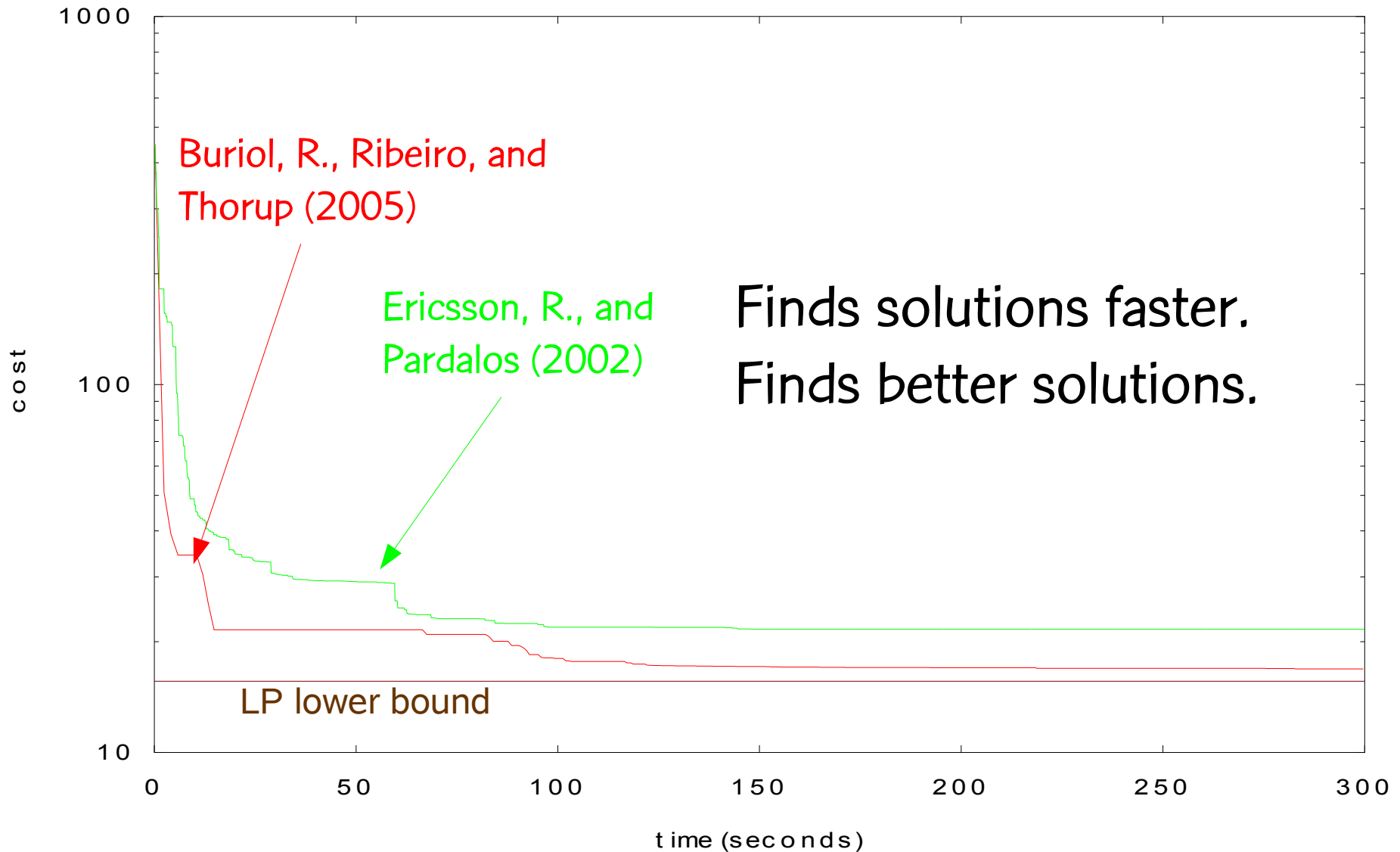
Buriol, R., Ribeiro, and Thorup (Networks, 2005)

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

# Fast local search

- Let  $A^*$  be the set of five arcs  $a \in A$  having largest  $\Phi_a$  values.
- Scan arcs  $a \in A^*$  from largest to smallest  $\Phi_a$ :
  - 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  $\Phi$  is reduced, restart local search.

# Effect of decoder with fast local search



# Routing and wavelength assignment

Noronha, R., and Ribeiro (MIC 2007)

- RWA consists in routing connections and assigning wavelengths to connections such that if two connections share an edge, they must be assigned different wavelengths. We want to minimize the number of wavelengths needed.
- The BFD-RWA (N. Skorin-Kapov, 2007) heuristic to date has the best results in the literature for this RWA problem when used in a multi-start process. It is an extension of the best fit heuristic for bin packing.
- Our random-key GA uses a chromosome where each gene corresponds to a connection and decodes by adding the key to the shortest hop count of the connections and using the resulting permutation as input to the BFD-RWA heuristic.
- On average, the GA found the best known solutions 24% faster than the multi-start BFD-RWA.

# Routing with distributed exponential flow splitting (DEFT)

R., Reis, Ritt, and Buriol (INFORMS Telecom. Conf., 2008)

- DEFT has been recently proposed by Xu, Chiang, and Rexford (2007).
- Load at a given node is split among all of its outgoing links (and not only the shortest path links as in OSPF). Moreover, the load is not evenly split among all links as is the case in OSPF. DEFT applies an exponential penalty to longer paths between origin-destination pairs nodes such that more load is routed through links that lead to shorter paths.
- DEFT with real-valued weights was shown to outperform OSPF with integer weights and often obtains optimal or near-optimal congestion.
- We propose a random-key GA for integer weight setting in DEFT routing and show that congestion is reduced when compared to OSPF routing.

# Survivable IP network design

# Survivable IP network design

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

- Given

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

- Determine

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

# Survivable IP network design

- Chromosome:

- A vector  $X$  of  $N$  random keys, where  $N$  is the number of links. The  $i$ -th random key corresponds to the  $i$ -th link weight.

- Decoder:

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

- For each failure mode: route demand according to OSPF and for each link  $i \in A$  determine load on link  $i$ .

- For each link  $i \in A$ , compute multiplicity  $M(i)$  needed to accommodate maximum load over all failure modes.

iterate

- Network design cost =  $\sum_{i \in A} M(i) \times K(i)$ .

# Survivable composite link IP network design

Andrade, Buriol, R., & Thorup (INFORMS Telecom. Conf., 2006)

- Given a load  $L(a)$  on arc  $a$ , we can compose several different link types that sum up to the needed capacity  $c(a) \geq L(a)$ :
  - $c(a) = \sum_{t \text{ used in arc } a} M(t,a) \times \gamma(t)$ ,  
where
    - $M(t,a)$  is the multiplicity of link type  $t \in \{1, 2, \dots, T\}$  on arc  $a$
    - $\gamma(t)$  is the capacity of link type  $t$ :  $\{\gamma(1), \gamma(2), \dots, \gamma(T)\} : \gamma(i) < \gamma(i+1)$
- Assumptions
  - Prices / unit length =  $\{p(1), p(2), \dots, p(T)\} : p(i) < p(i+1)$
  - $[p(T)/\gamma(T)] < [p(T-1)/\gamma(T-1)] < \dots < [p(1)/\gamma(1)]$ : economies of scale
  - $\gamma(i) = \alpha \times \gamma(i-1)$ , for  $\alpha \in \mathbb{N}$ ,  $\alpha > 1$ , e.g.
    - $\gamma(\text{OC192}) = 4 \times \gamma(\text{OC48})$
    - $\gamma(\text{OC48}) = 4 \times \gamma(\text{OC12})$
    - $\gamma(\text{OC12}) = 4 \times \gamma(\text{OC3})$

# Survivable composite link IP network design

- Chromosome:

- A vector  $X$  of  $N$  random keys, where  $N$  is the number of links. The  $i$ -th random key corresponds to the  $i$ -th link weight.

- Decoder:

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

- For each failure mode: route demand according to OSPF and for each arc  $i \in A$  determine the load on arc  $i$ .

- For each arc  $i \in A$ , determine the multiplicity  $M(t,i)$  for each link type  $t$  using the maximum load for that arc over all failure modes. iterate

- Network design cost =  $\sum_{i \in A} \sum_{t \text{ used in arc } i} M(t,i) \times p(t)$

# Concluding remarks

# Concluding remarks

- We described a general framework to construct heuristics for finding near-optimal or optimal solutions to combinatorial optimization problems.
- Besides the networking applications discussed, we have applied this technique to other problems, including:
  - Manufacturing cell formation (Gonçalves and R., 2004)
  - 2-dim packing (Gonçalves and R., 2006)
  - Job-shop, project, and multi-project scheduling (Gonçalves, Mendes, and R., 2005, 2007, 2007)
  - Congestion pricing in transportation networks (Buriol, Hirsch, Pardalos, Querido, and R., 2008)

# The End