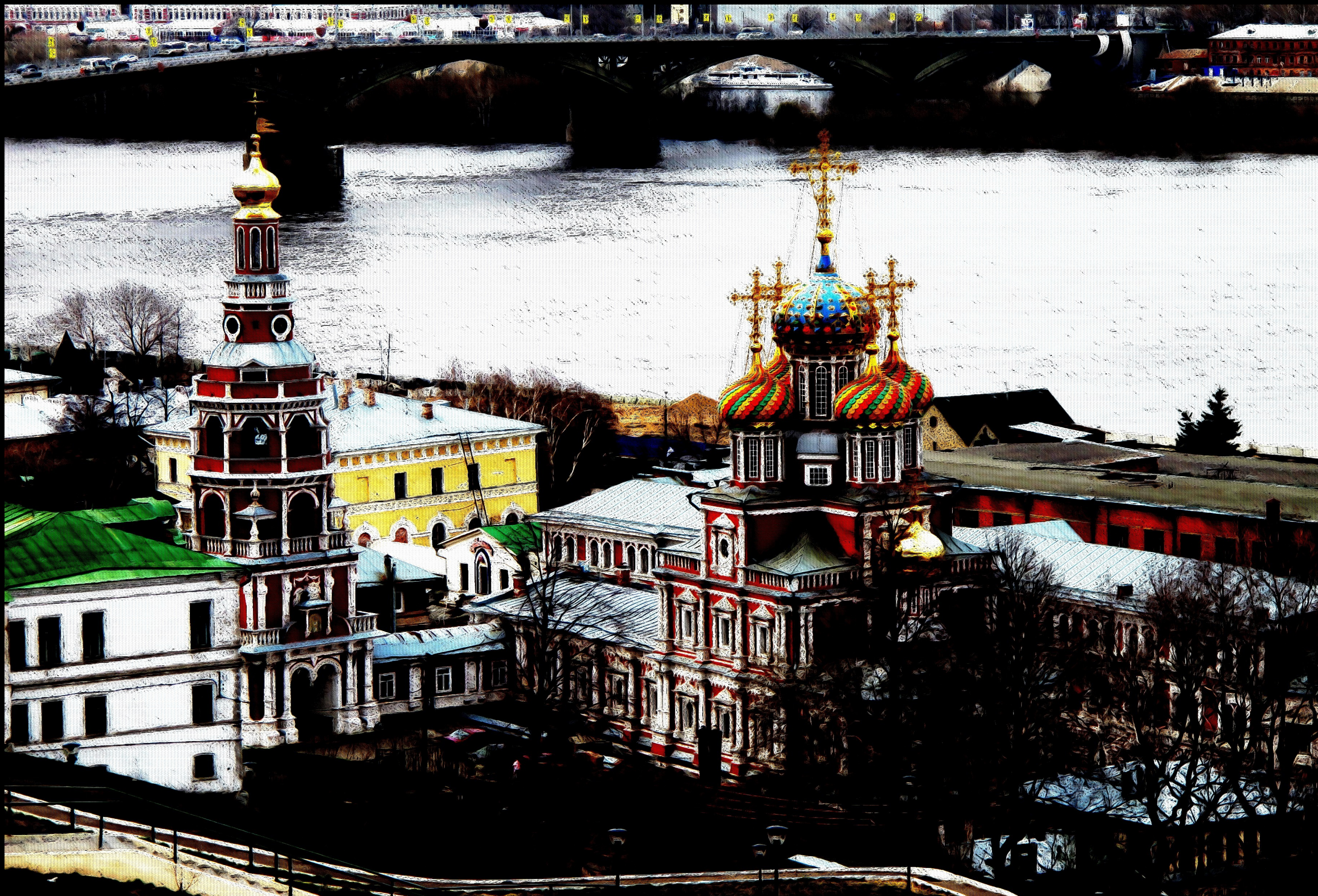
A biased random-key genetic algorithm for a prize-collecting directed Steiner forest network design problem

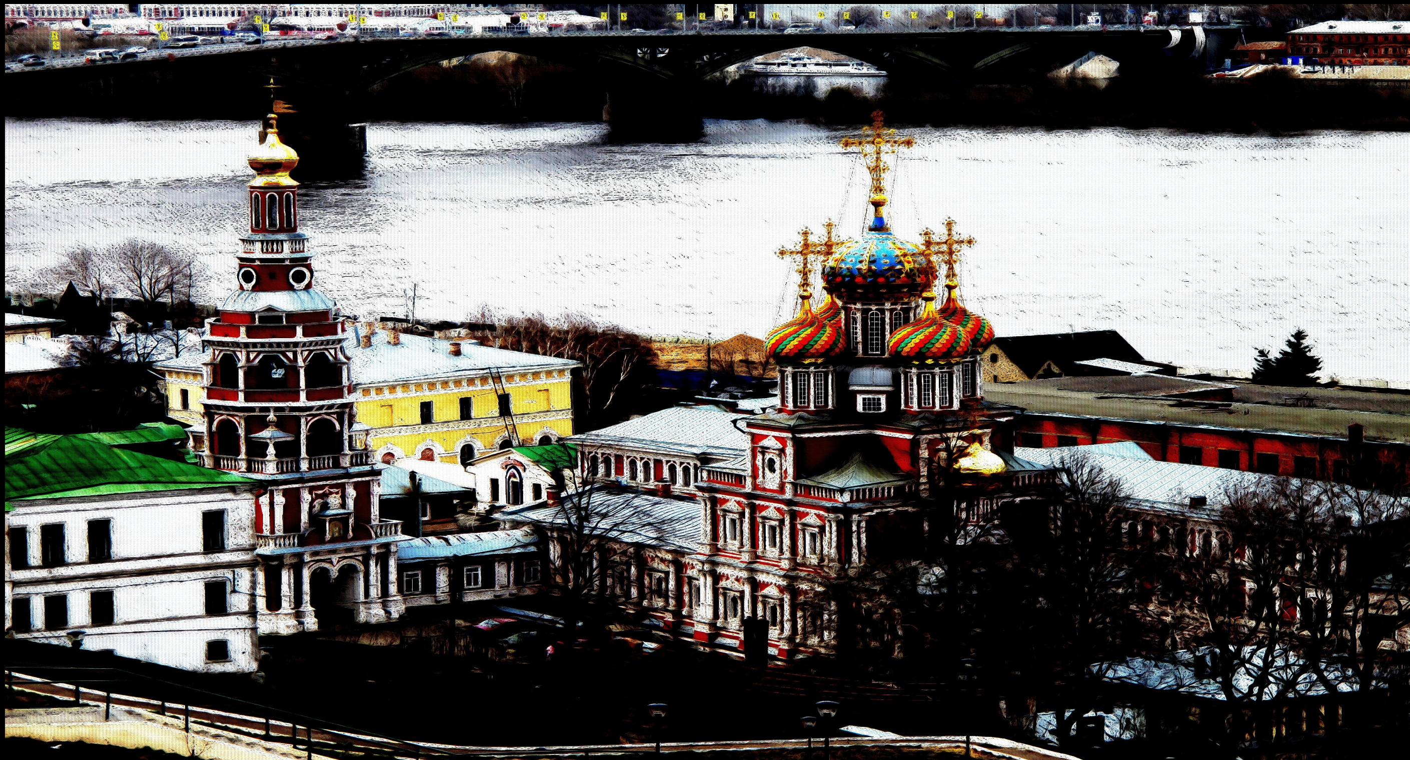
Mauricio G. C. Resende
AT&T Labs Research
Middletown, New Jersey
mgcr@research.att.com



Joint work with

C.E. de Andrade & F.K. Miyazawa
(UNICAMP, Brazil)

R.D. Doverspike, K. Reichmann,
R.K. Sinha & W. Zhang
(AT&T Labs Research, USA)



Summary

- Prize collecting directed k-hop Steiner forest (PCk-HSF) problem
- Wireless backhaul network planning as a PCk-HSF problem with additional constraints
- Biased random-key genetic algorithms (BRKGA)
- BRKGA for wireless backhaul network planning focusing on the decoder
- Application of the BRKGA to a "real" instance of wireless backhaul network planning
- Concluding remarks

Prize collecting directed Steiner Forest

- Let $G = (V,E)$ be a given directed graph
- Let V_r , V_s , and V_d partition the node set, i.e.

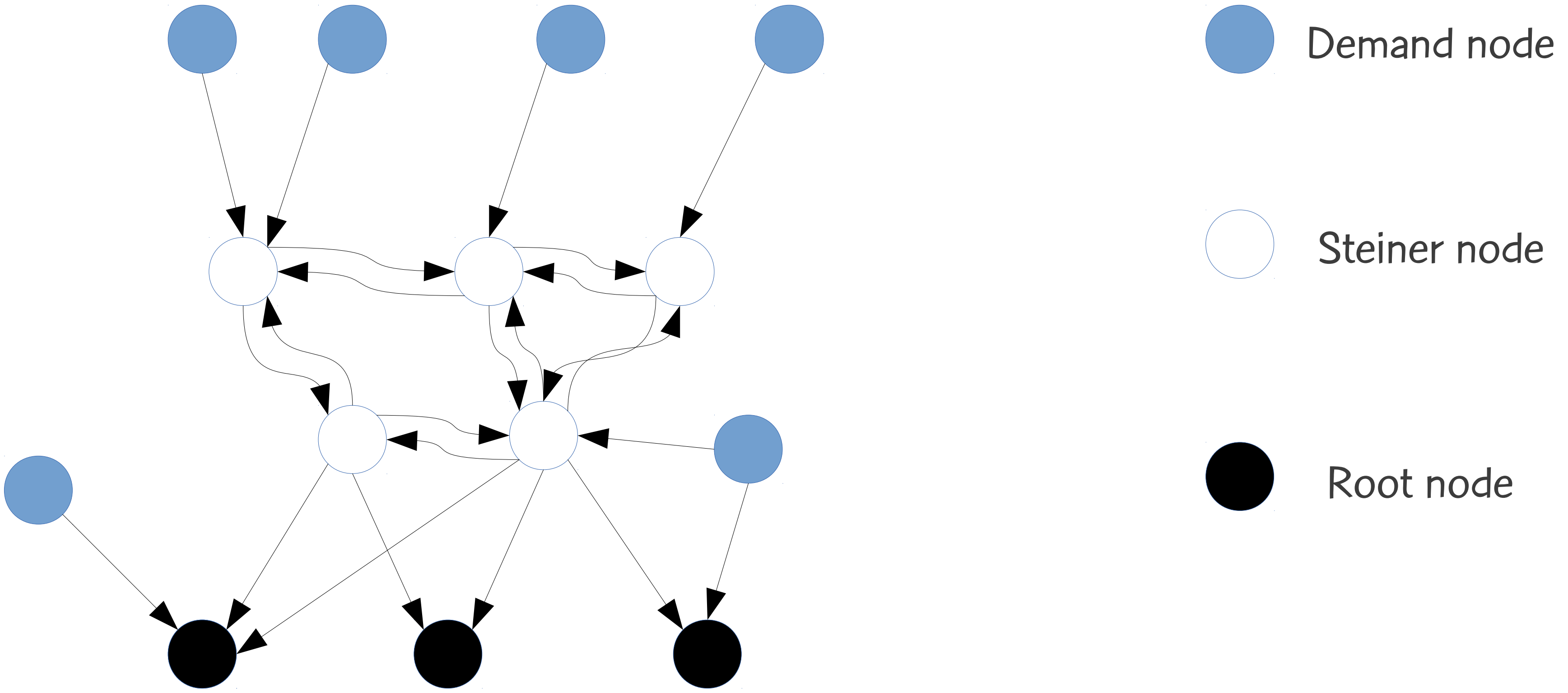
$$V_r \cup V_s \cup V_d = V \quad \text{and} \quad V_r \cap V_s \cap V_d = \emptyset$$

  - V_r – set of root nodes
  - V_s – set of Steiner nodes
  - V_d – set of demand nodes

Prize collecting directed Steiner Forest

- Let $G = (V,E)$ be a given directed graph
- Arcs are directed
  - From demand nodes to Steiner nodes and root nodes
  - From Steiner nodes to root nodes
- For each pair of nodes $v, u \in V_s$
  - $(u,v) \in E$
  - $(v,u) \in E$

Prize collecting directed Steiner Forest

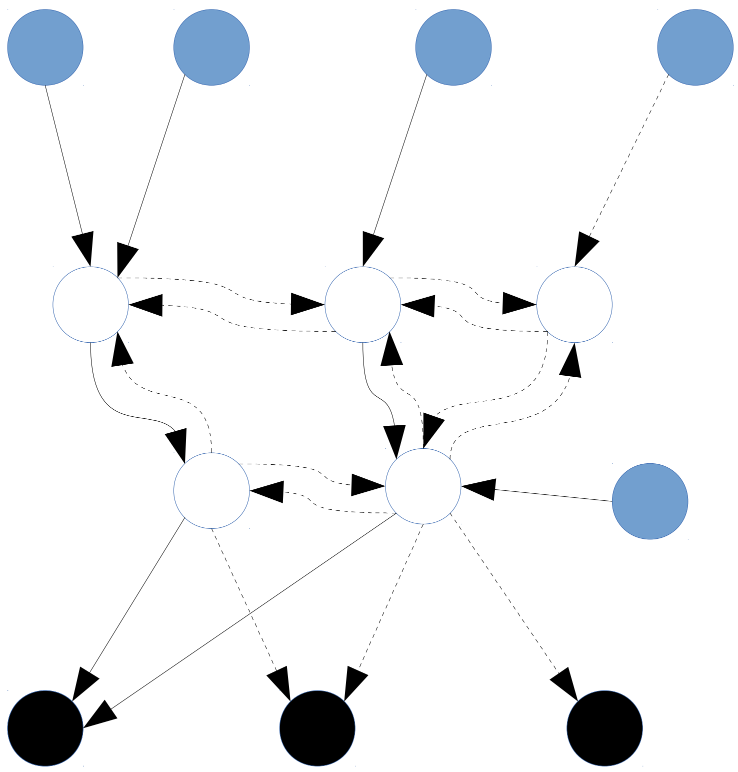


Prize collecting directed Steiner Forest

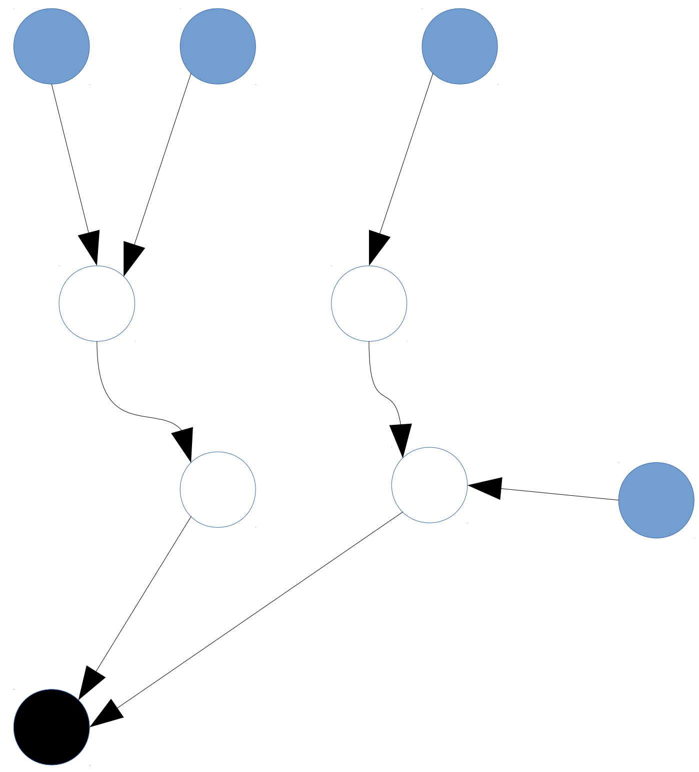
- A Directed Steiner Tree $T = (V[T] \subseteq V, E[T] \subseteq E)$ with root in $r \in V_r[T]$ is a loopless connected subgraph of G with a unique root r.
  - For each demand node $u \in V[T]$ there is a unique path in T from u to r.

Prize collecting directed Steiner Forest

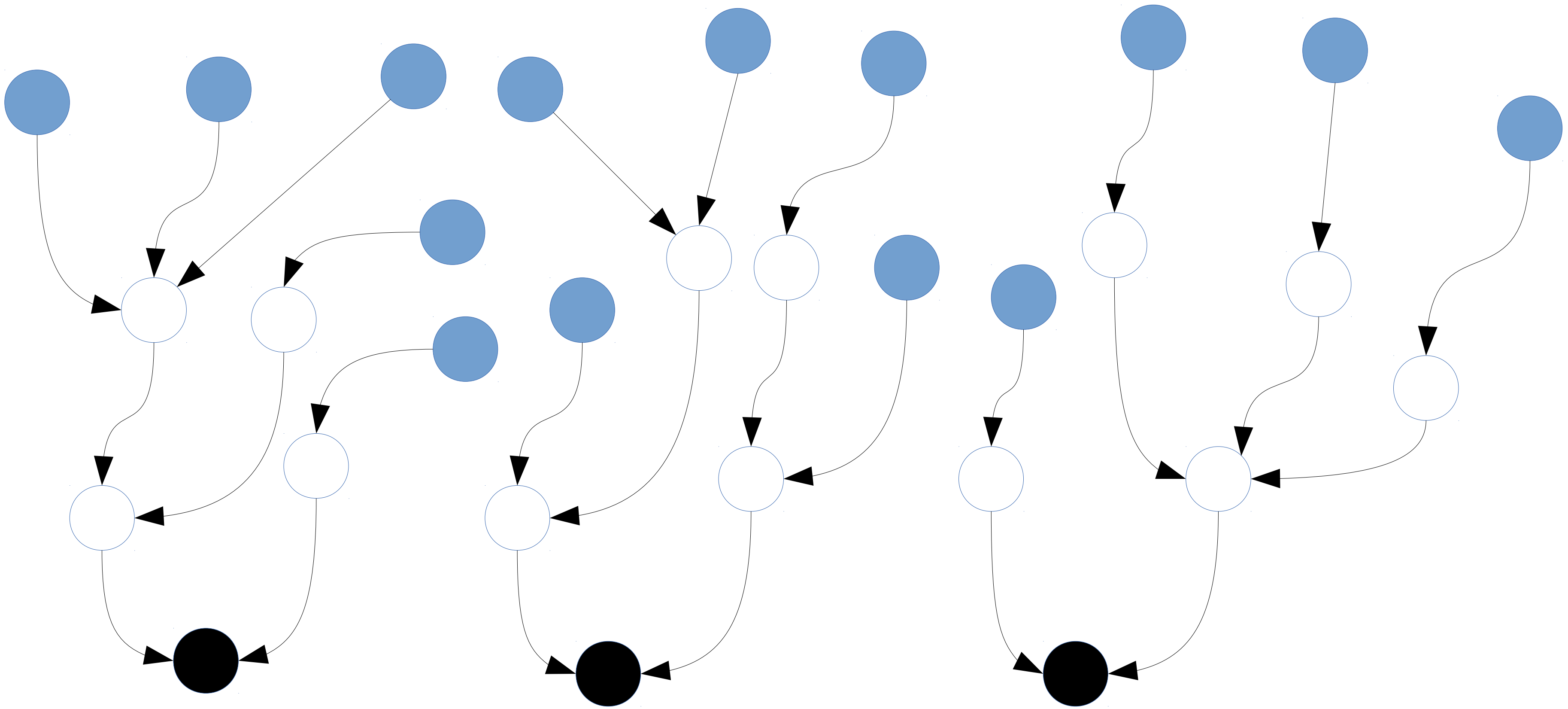
- A k-Hop Directed Steiner Tree is a directed Steiner tree (connected subgraph of G with a unique root r) such that:
  - Any path from a demand node u to the root r has no more than k+2 nodes, including u and r.
- A k-Hop Directed Steiner Forest is a collection of disjoint k-Hop Directed Steiner Trees



Acyclic directed graph



2-Hop Directed Steiner tree



2-Hop Directed Steiner Forest

Prize collecting directed Steiner Forest

- A demand function $d: V_d \rightarrow \mathbb{R}^+$ gives the prize to be collected from each demand node if it is a leaf on the tree.
- A cost function $c: V_s \rightarrow \mathbb{R}^+$ gives the cost of of each Steiner vertex.

Prize collecting directed Steiner Forest

- Given a directed graph $G = (V,E)$, vertex partition V_r , V_s , and V_d, hop parameter k, demand function d, and cost function c
- FIND: A k-hop directed Steiner forest F such that the profit

$$\text{sum} \, [\, v \in V_d[F] \,] \, d_v - \text{sum} \, [\, v \in V_s[F] \,] \, c_v$$

is maximized.

Wireless backhaul network planning

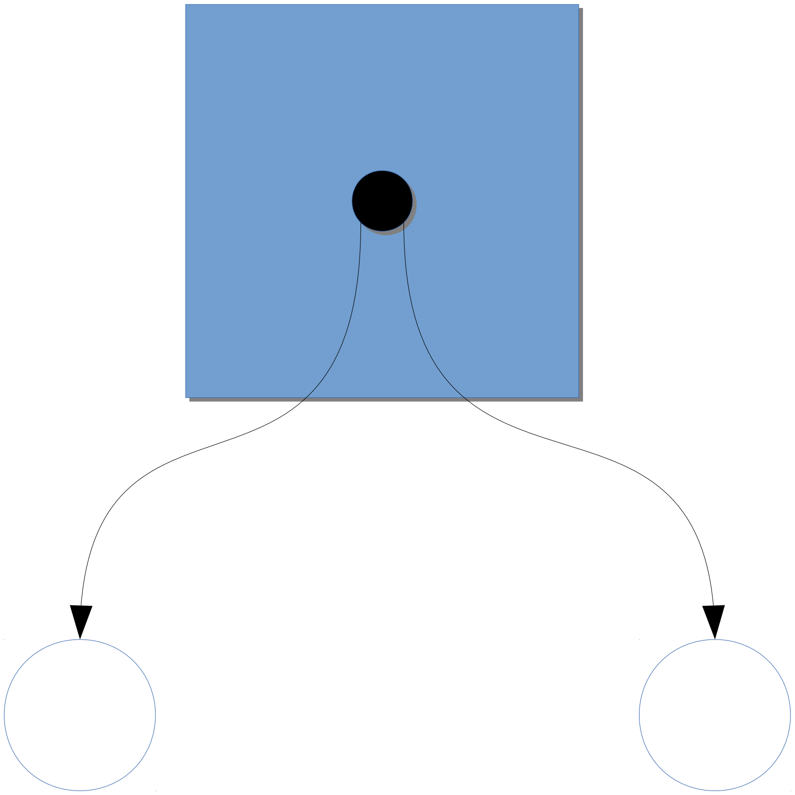
Given a geographical region where locations are represented as lat-long coordinates, where

- V_d is the set of nodes representing traffic origination points (demand points) in the region
- V_s is the set of nodes representing locations where equipment for traffic collection and routing is located (e.g. utility poles)
- V_r is the set of nodes representing fibered access points (FAP), e.g. remote terminals (RT), macrocell (MC), or central offices.

Wireless backhaul network planning

Constraints: Demand splitting

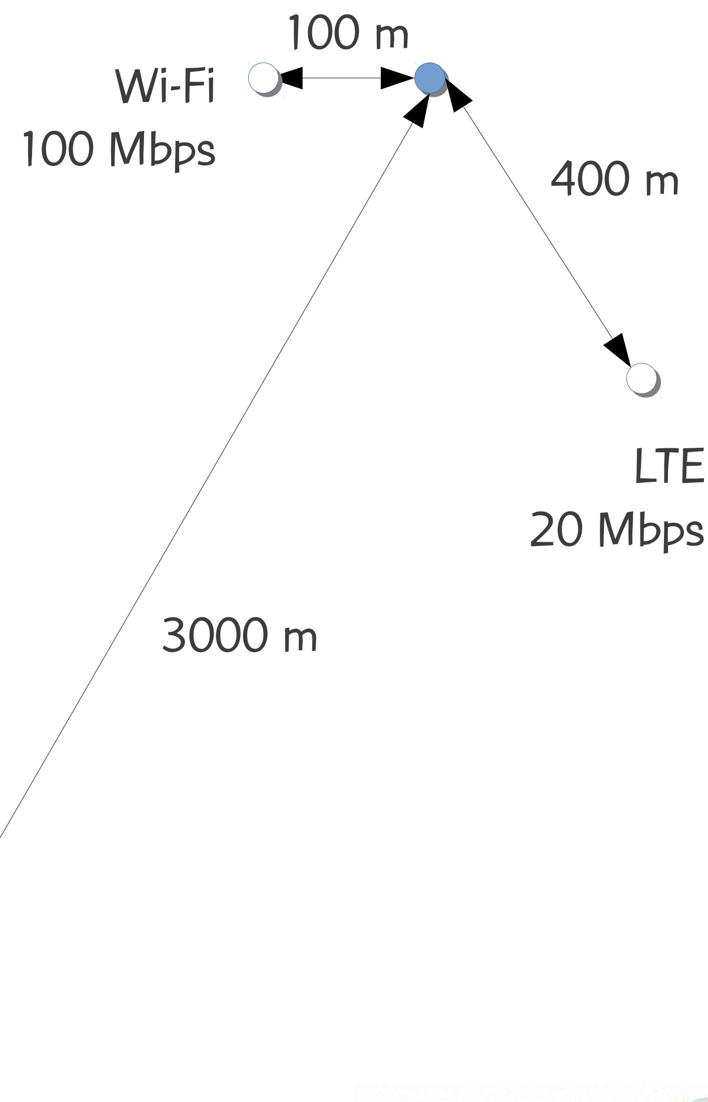
- Estimate for each block total demand is placed in center of block
- Block can be served by one or more antennae so demand can be split among them
- Because of this undirected cycles can be introduced resulting in a directed acyclic graph (DAG)



Wireless backhaul network planning

Constraints: Access equipment action radii & capacities

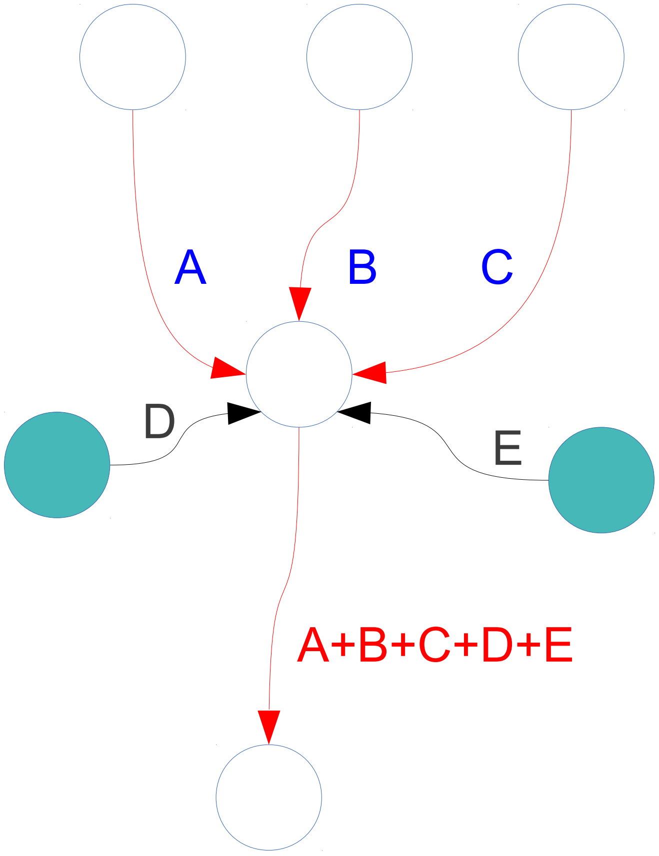
- Action radii
  - Wi-Fi: 100 m
  - LTE: 400 m
  - macrocell: 3 km
- Processing capacity
  - Wi-Fi: 100 Mbps
  - LTE: 20 Mbps
  - macrocell: 25 Mbps



Wireless backhaul network planning

Constraints: Retransmitter equipment capacity

- Action radii: 1 km
- Processing capacity
  - Flow that equipment receives from other wireless retransmitters plus flow it sends to other retransmitters is limited to at most 100 Mbps



A+B+C+A+B+C+D+E $\leqslant$100

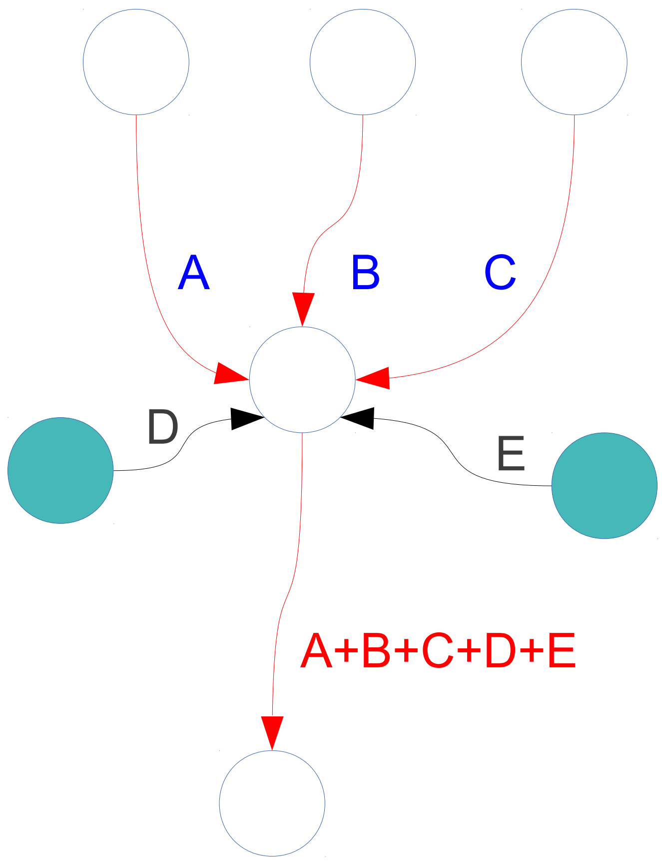
Wireless backhaul network planning

Constraints: Retransmitter equipment capacity

Fan-in constraint

- A limited number (5) of neighboring transmission equipment can flow traffic into equipment

In-degree = 3 $\leqslant$ 5



A+B+C+A+B+C+D+E $\leqslant$100

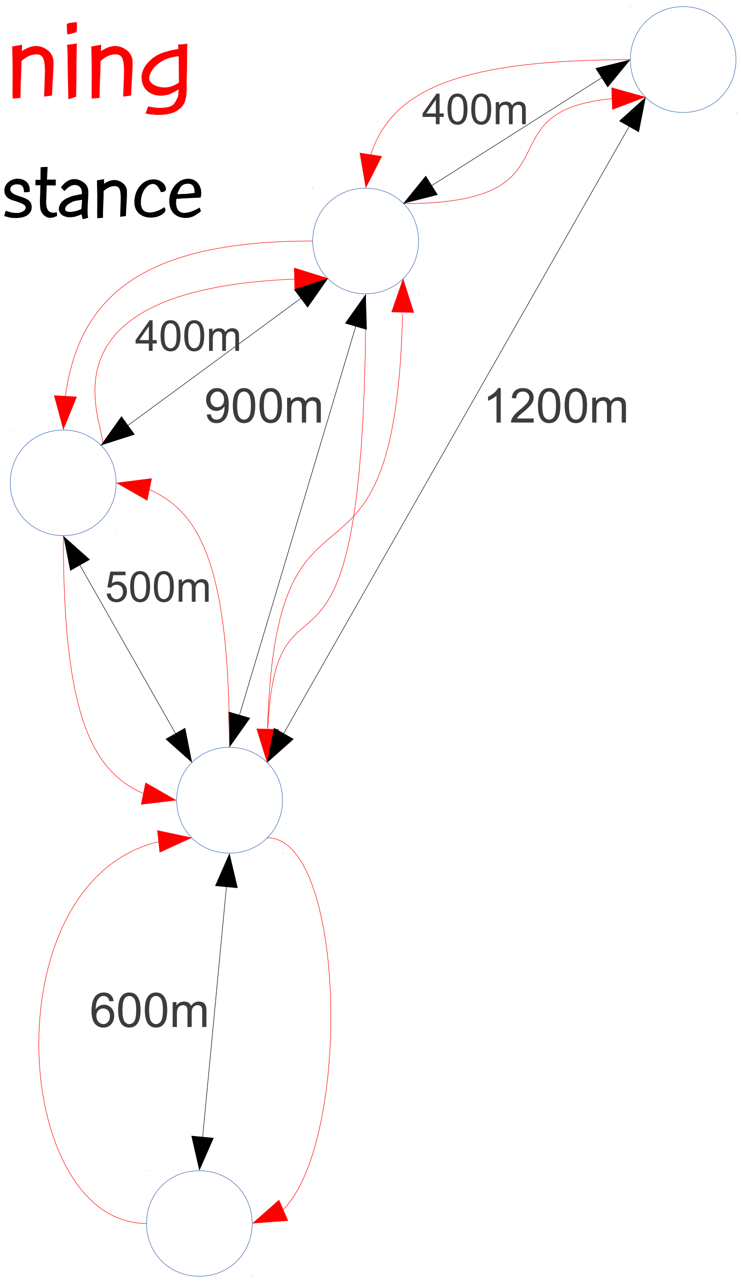
Wireless backhaul network planning

Constraints: Line of sight and minimum distance

PCSF Problem assumed, for each pair $u, v \in V_s$ that $(u,v) \in E$ and $(v,u) \in E$

Not so in wireless backhaul planning: $(u,v) \in E$ and $(v,u) \in E$ only if

- Poles u and v are within 1000 m of each other



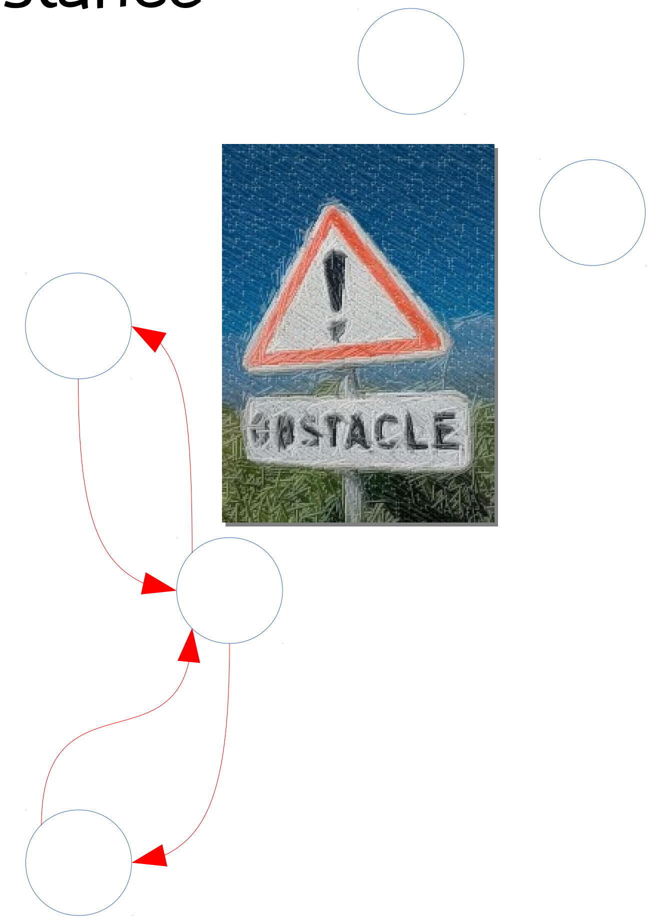
Wireless backhaul network planning

Constraints: Line of sight and minimum distance

PCSF Problem assumed, for each pair $u, v \in V_s$ that $(u,v) \in E$ and $(v,u) \in E$

Not so in wireless backhaul planning: $(u,v) \in E$ and $(v,u) \in E$ only if

- Poles u and v are within 1000 m of each other
- Poles u and v are in each other's line of sight

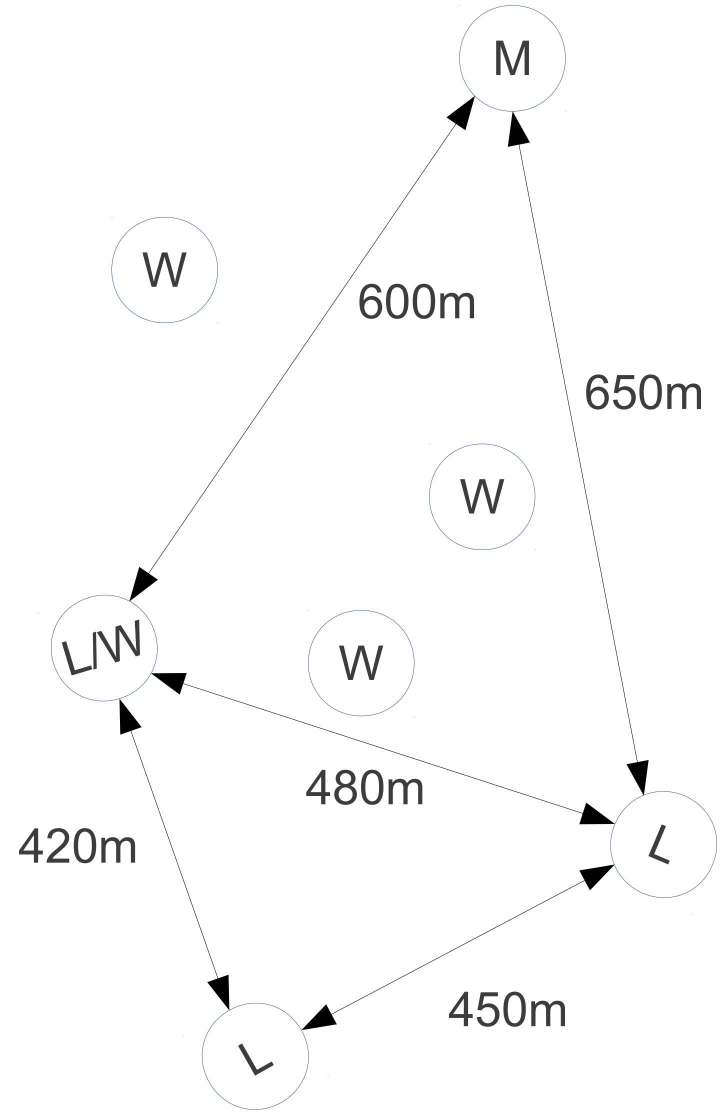


Wireless backhaul network planning

Constraints: Interference

LTE and MC use licensed spectrum and can interfere with each other.

- Pairs of LTE antennae must be separated by at least 400m
- LTE and macrocells by at least 500m
- No constraint on Wi-Fi exists since it uses non-licensed spectrum



Wireless backhaul network planning

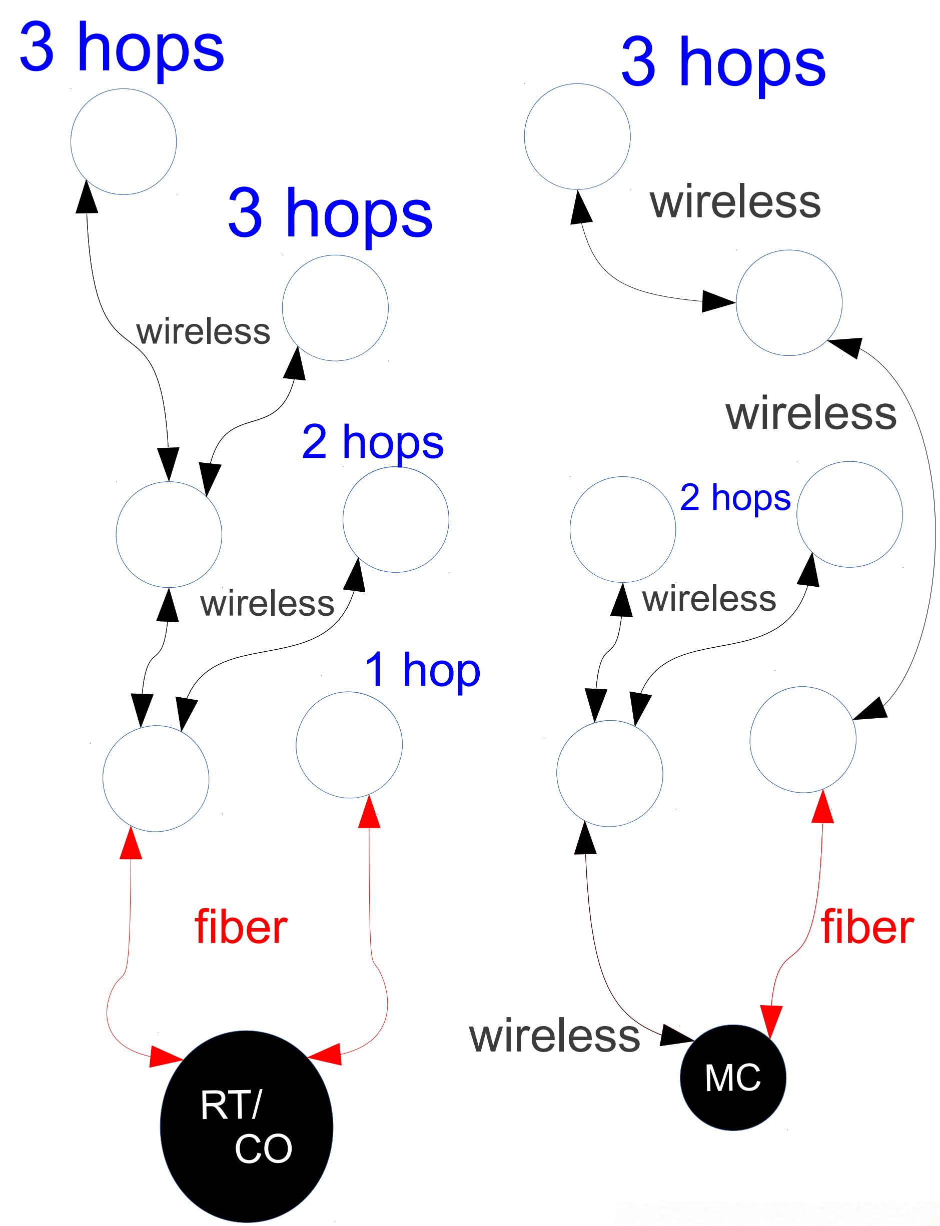
Constraints: k-Hops

First hop is from FAP (root) to pole

- If root is Central Office (CO) or RT link must use fiber
- If root is macrocell link can be fiber or wireless

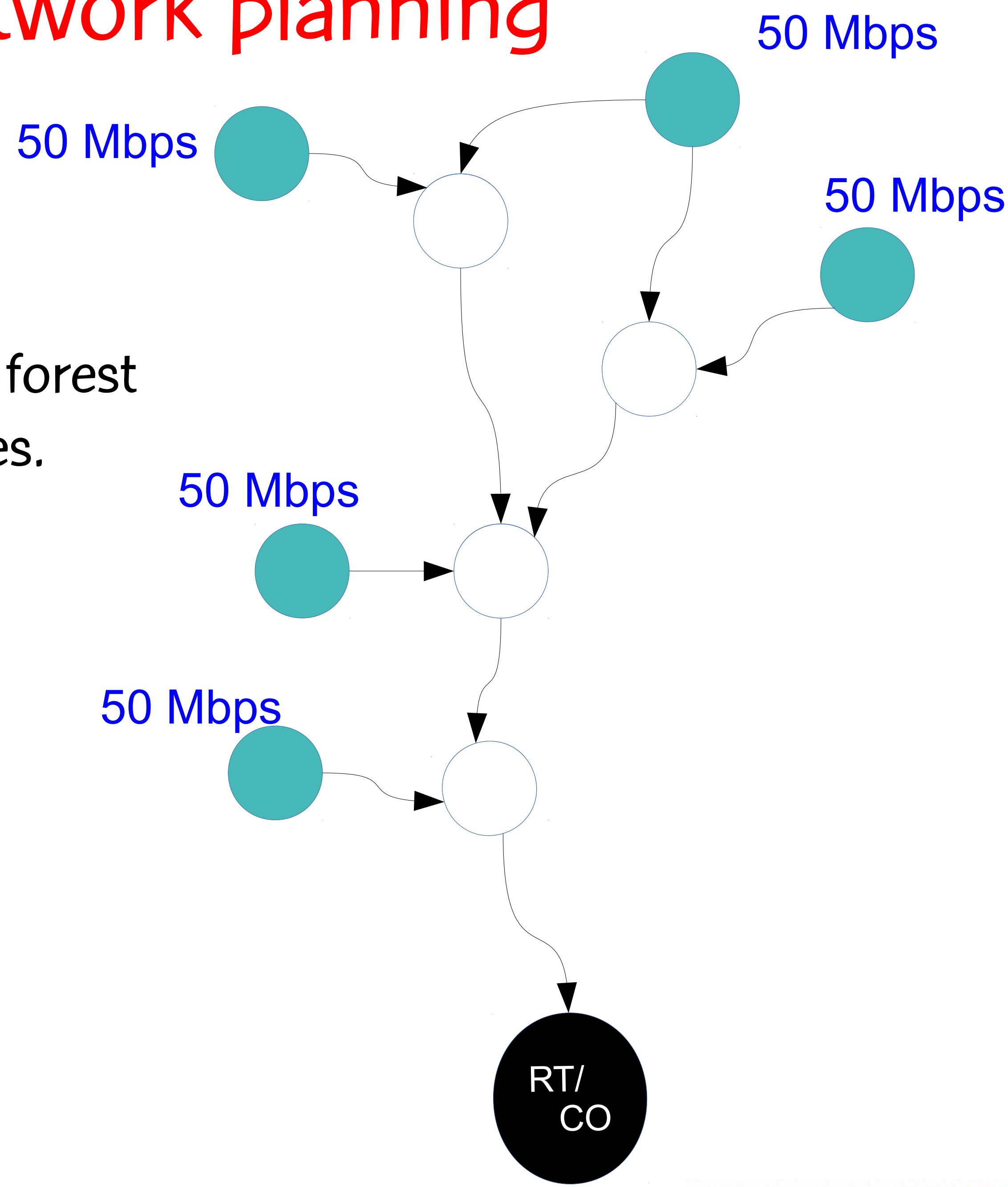
All other links are wireless

Number of hops is limited to $k = 2$ or 3 (in case first link is fiber)



Wireless backhaul network planning

Constraints: Traffic flow



Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

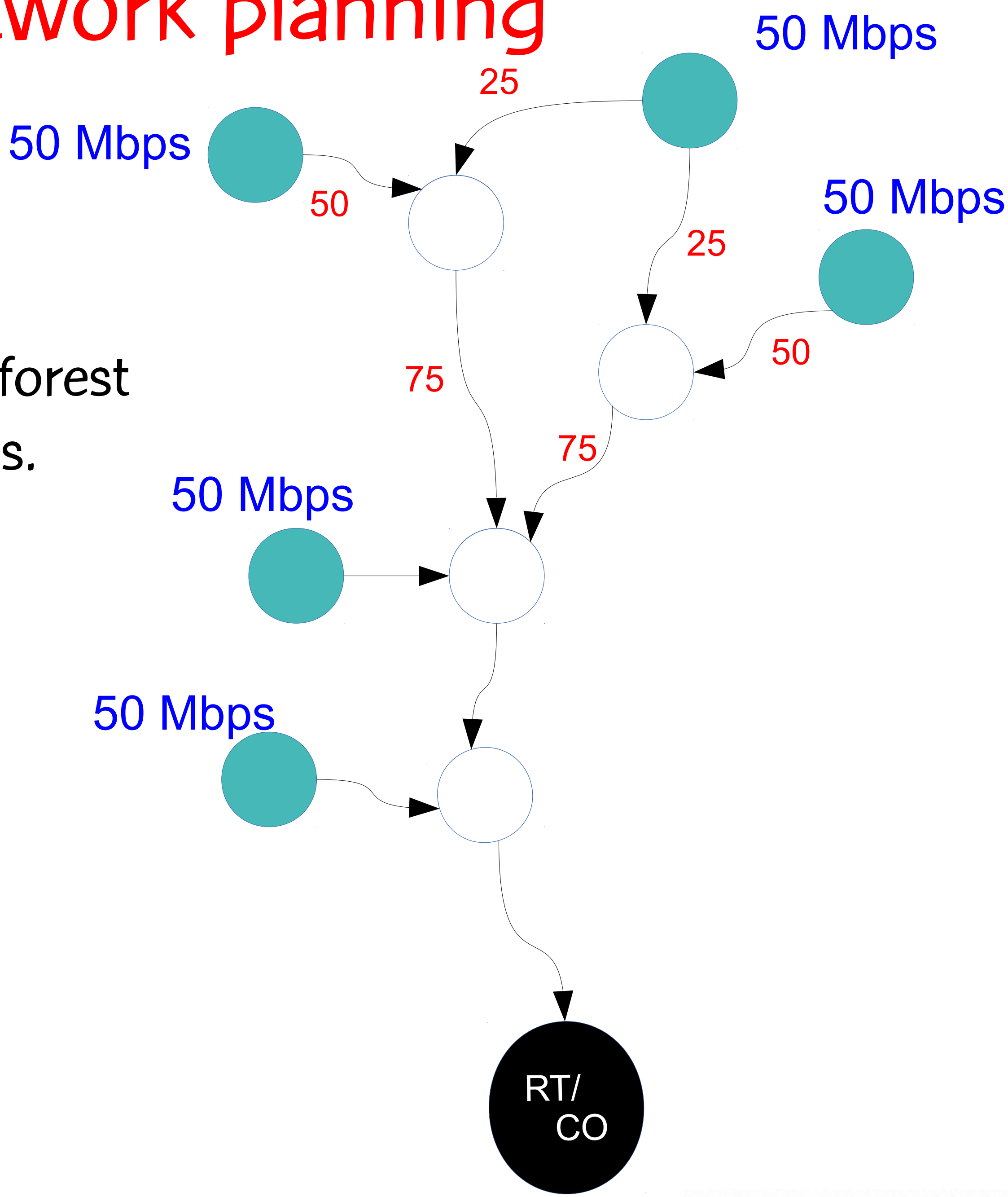
Only traffic that reaches roots is counted as revenue.

Wireless backhaul network planning

Constraints: Traffic flow

Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.

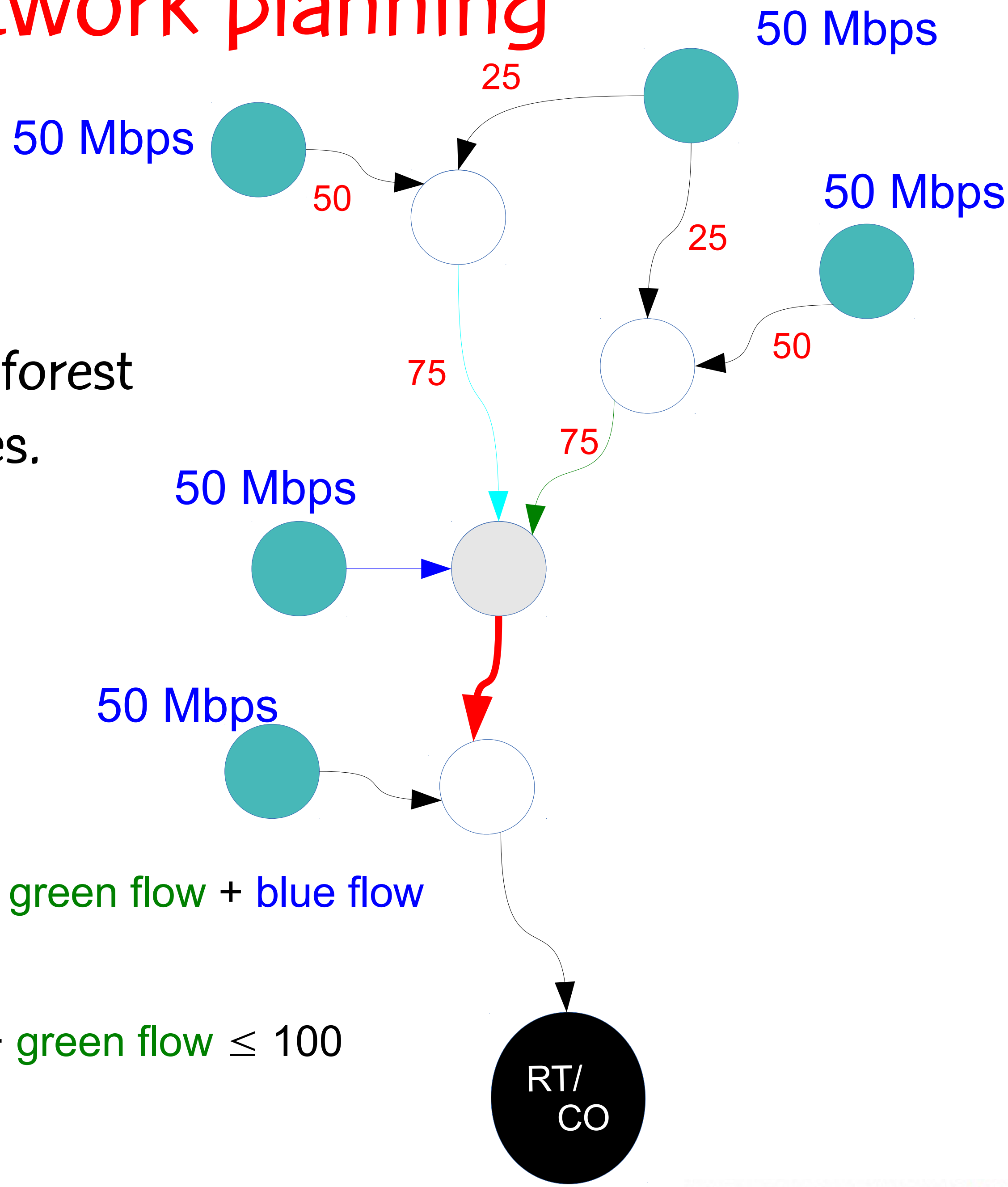


Wireless backhaul network planning

Constraints: Traffic flow

Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.



Flow conservation: red flow = cyan flow + green flow + blue flow

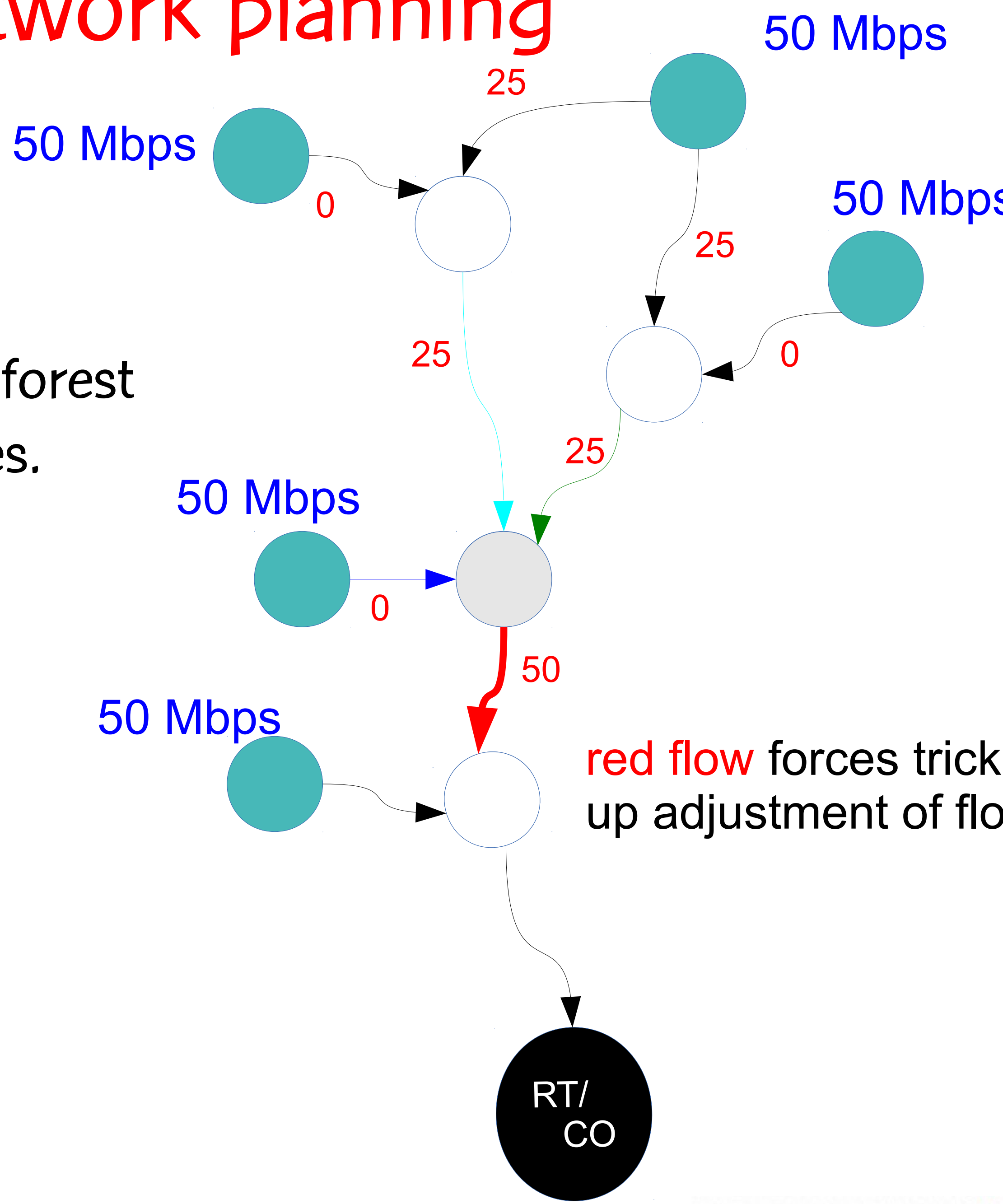
Capacity constraint: red flow + cyan flow + green flow $\leq$ 100

Wireless backhaul network planning

Constraints: Traffic flow

Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.



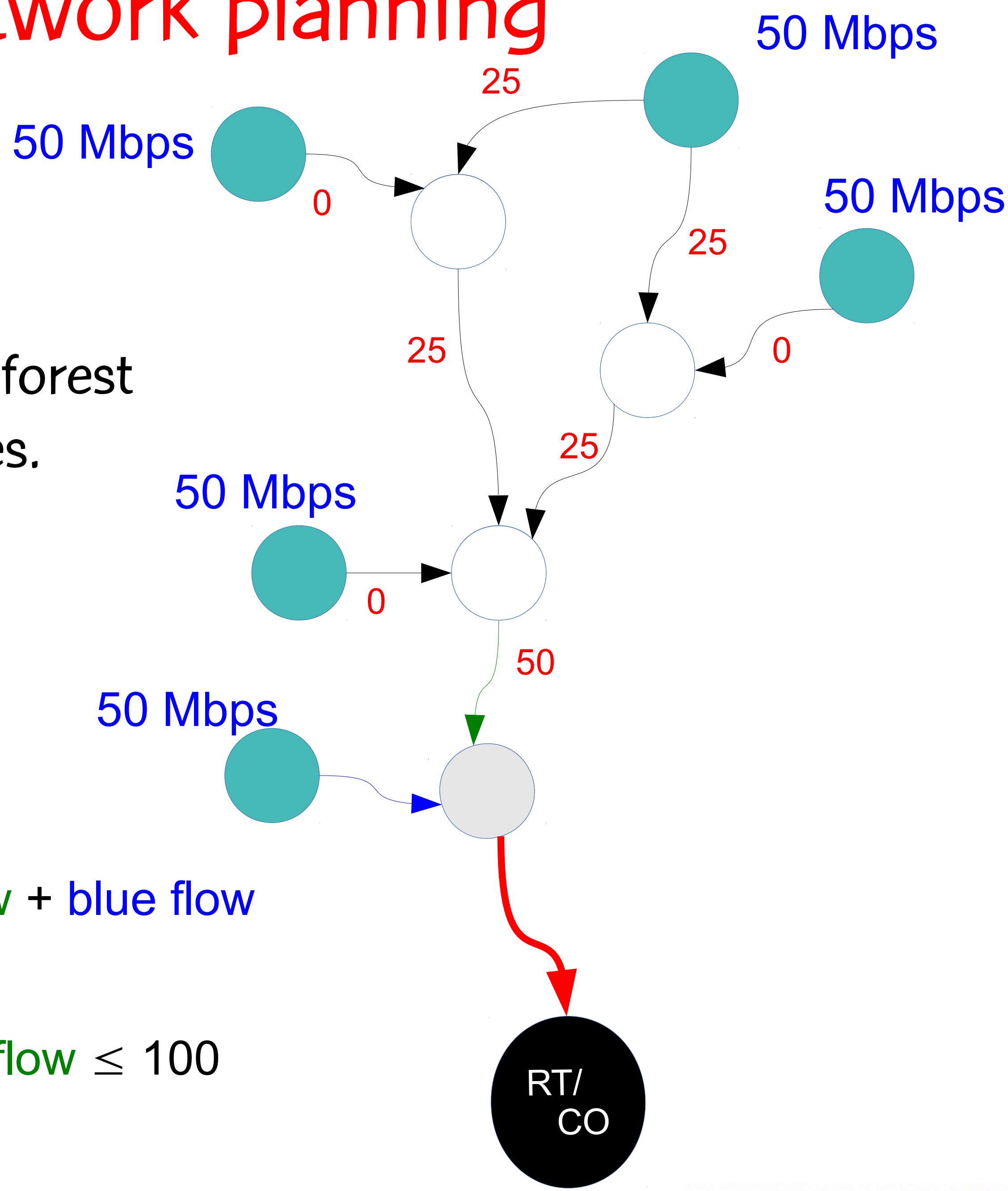
red flow forces trickle up adjustment of flows

Wireless backhaul network planning

Constraints: Traffic flow

Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.

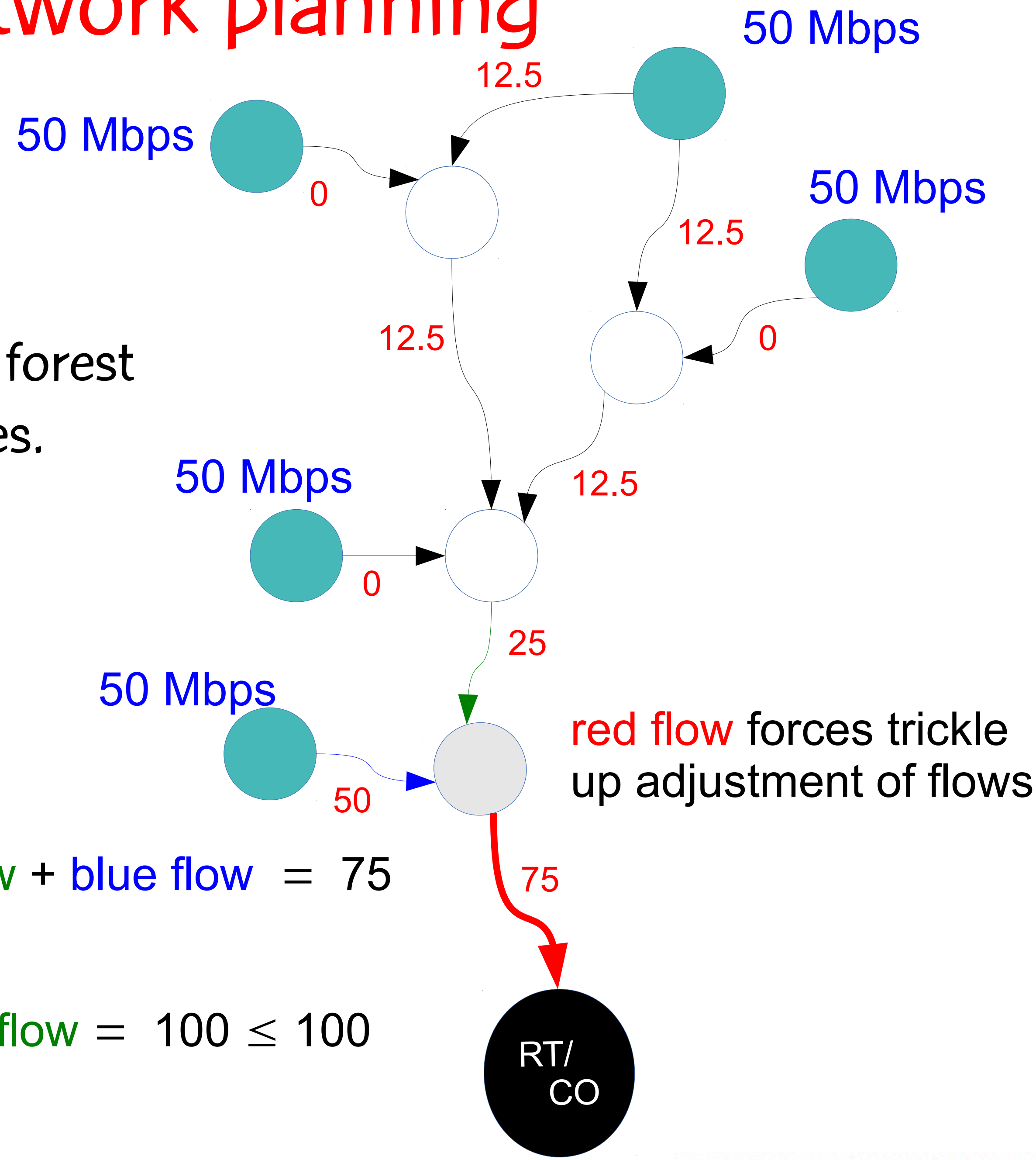


Flow conservation: red flow = green flow + blue flow

Capacity constraint: red flow + green flow $\leq$ 100

Wireless backhaul network planning

Constraints: Traffic flow



Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.

Flow conservation: red flow = green flow + blue flow $= 75$

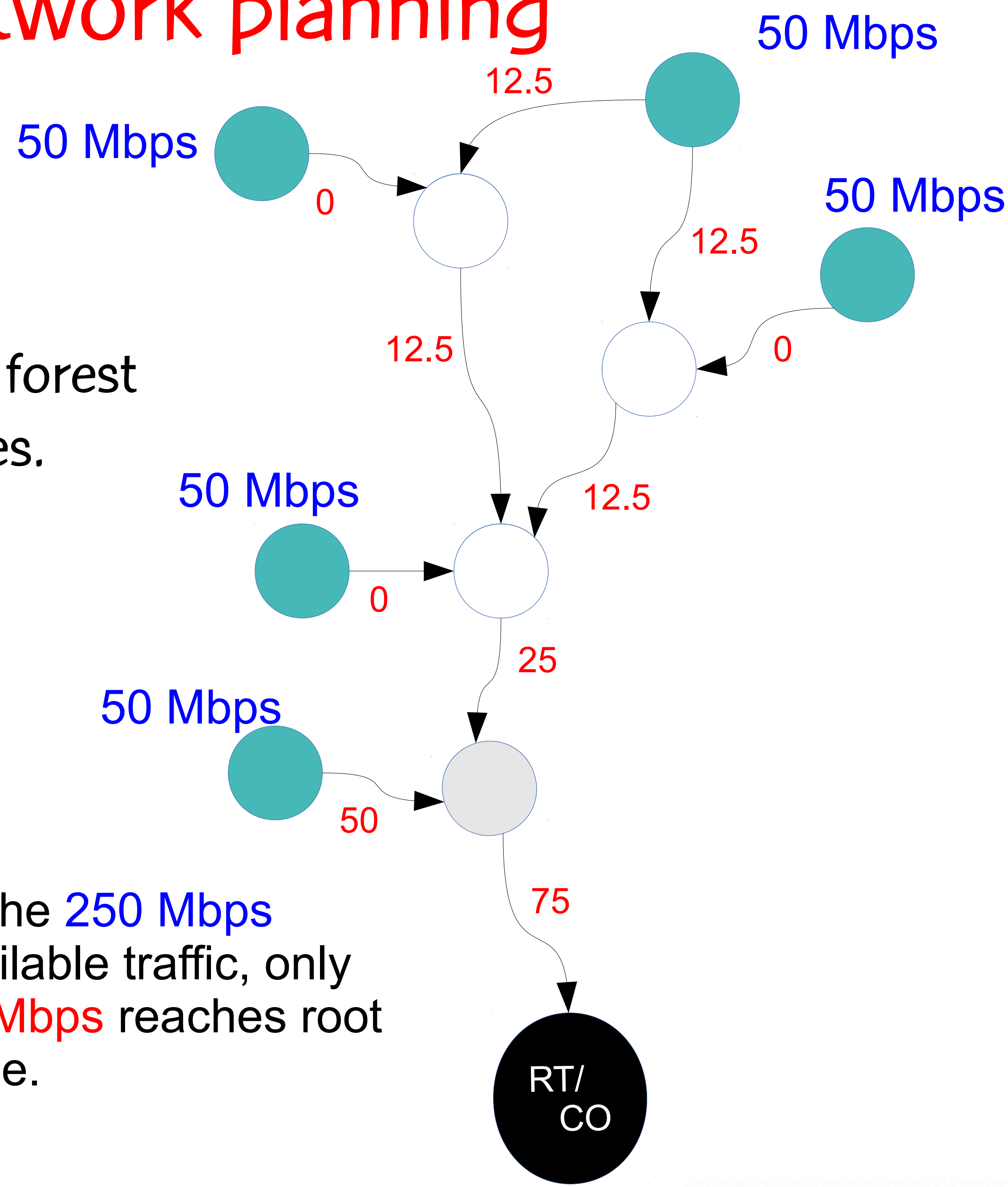
Capacity constraint: red flow + green flow $= 100 \leq 100$

Wireless backhaul network planning

Constraints: Traffic flow

Traffic that is backhauled from demand points to root nodes of forest is limited by equipment capacities.

Only traffic that reaches roots is counted as revenue.

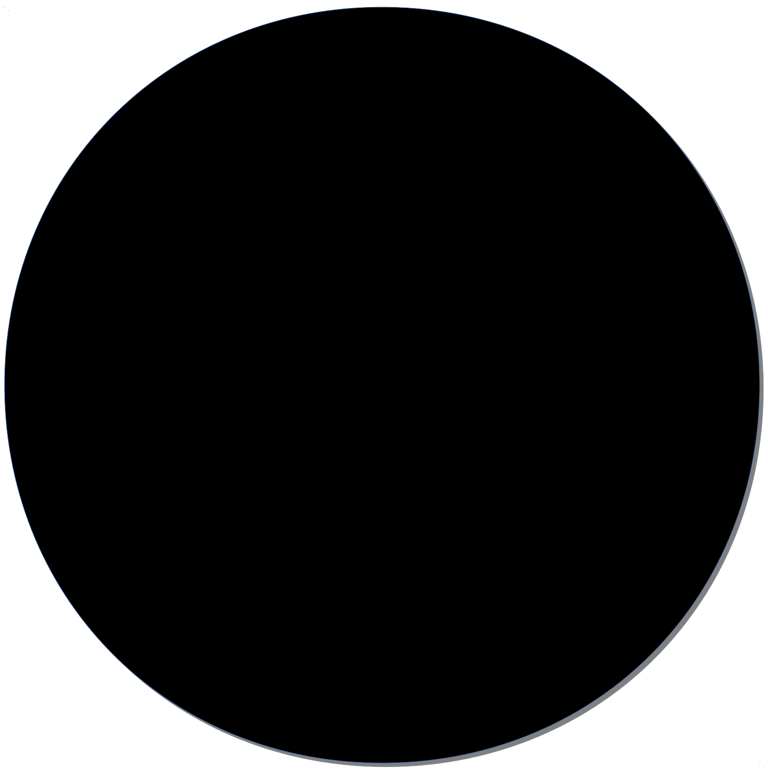


Of the 250 Mbps available traffic, only 75 Mbps reaches root node.

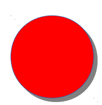
Genetic algorithms

Genetic algorithms

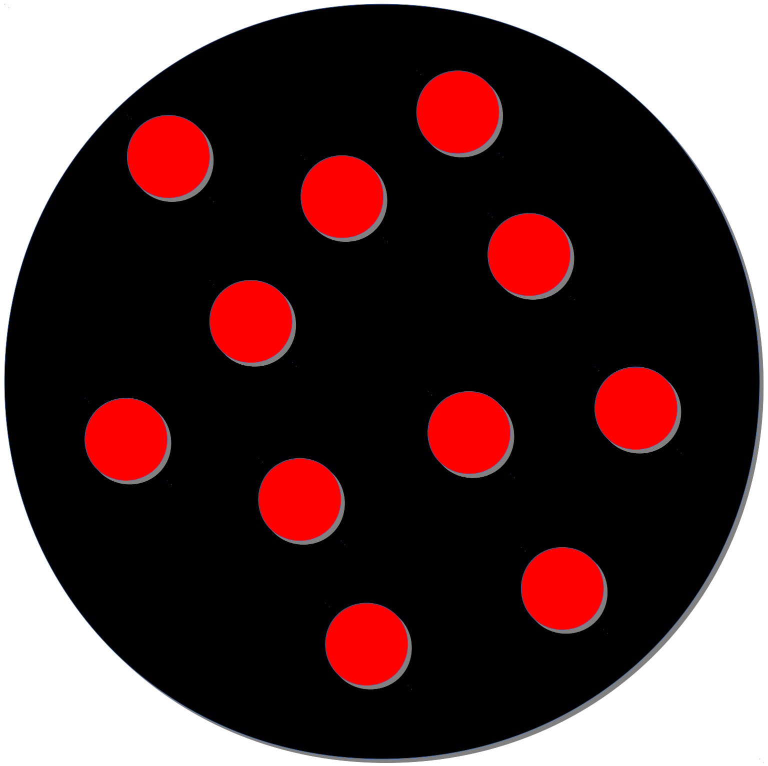
Holland (1975)



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

Individual: solution 

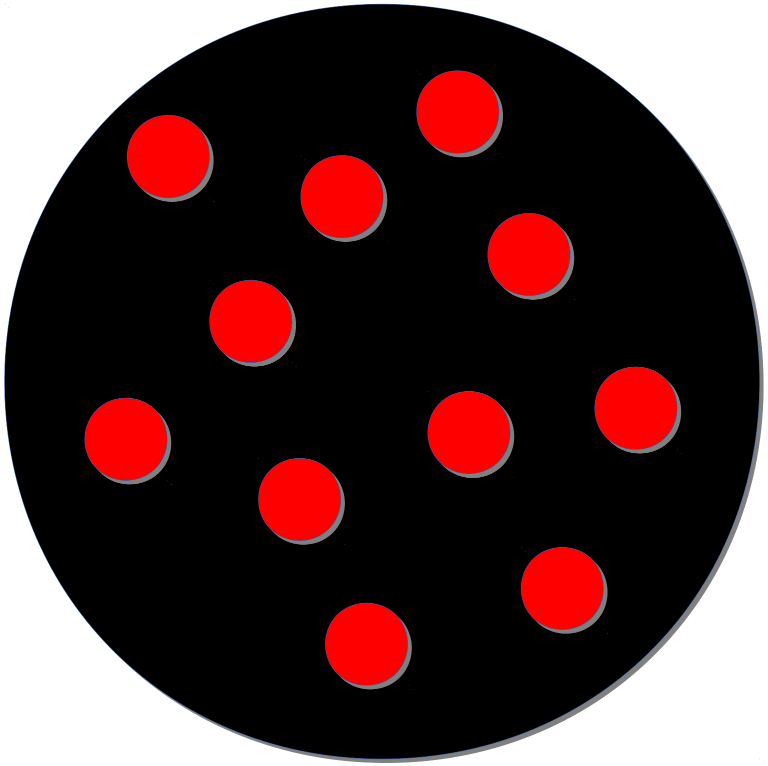
Genetic algorithms



Individual: solution (chromosome = string of genes)

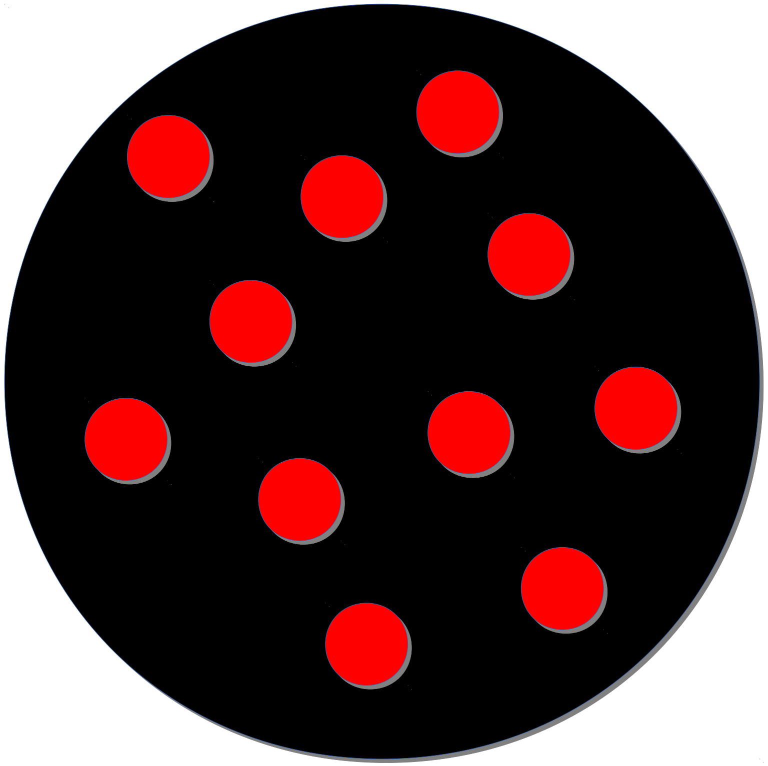
Population: set of fixed number of individuals

Genetic algorithms



Genetic algorithms evolve population applying Darwin's principle of survival of the fittest.

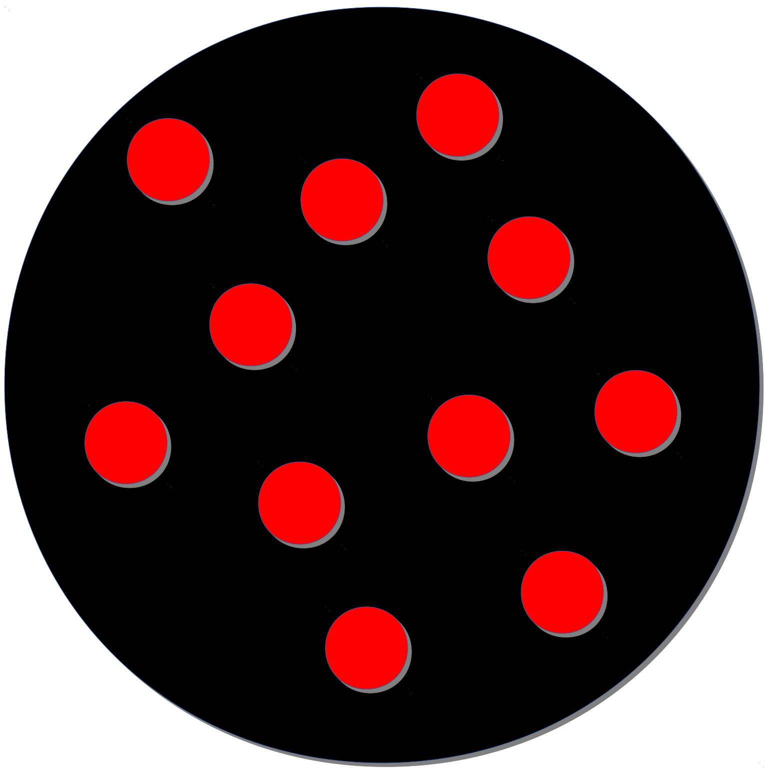
Genetic algorithms



Genetic algorithms evolve population applying Darwin's principle of survival of the fittest.

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

Genetic algorithms

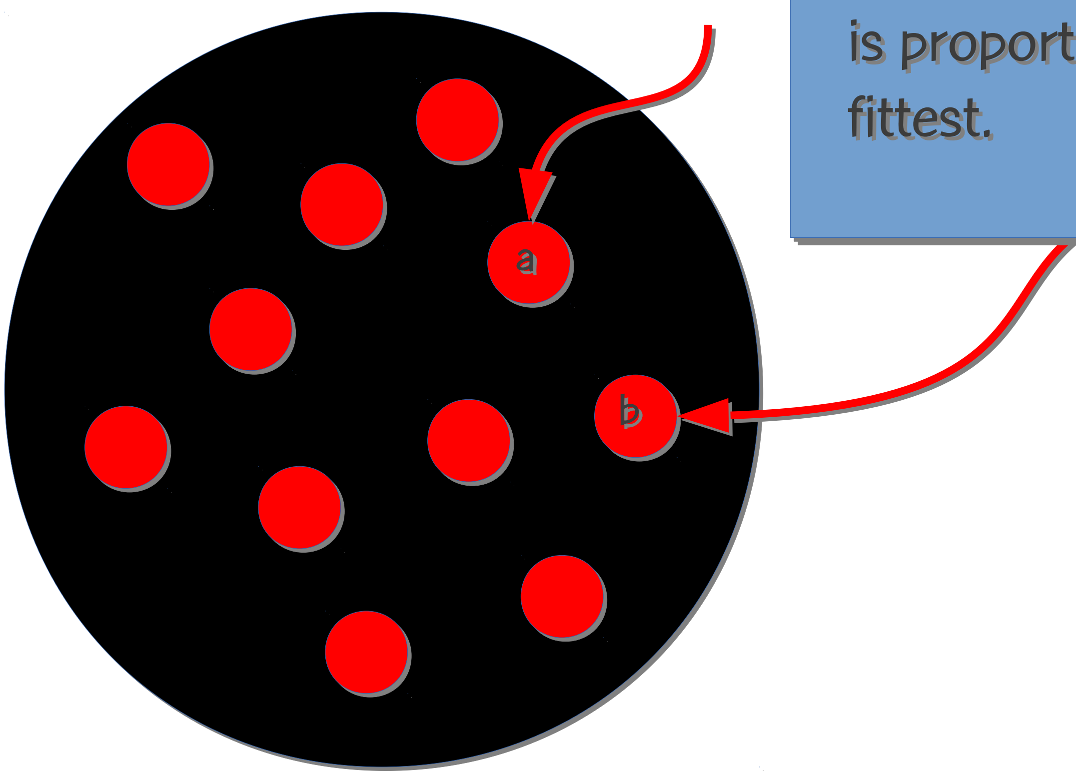


Genetic algorithms evolve population applying Darwin's principle of survival of the fittest.

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

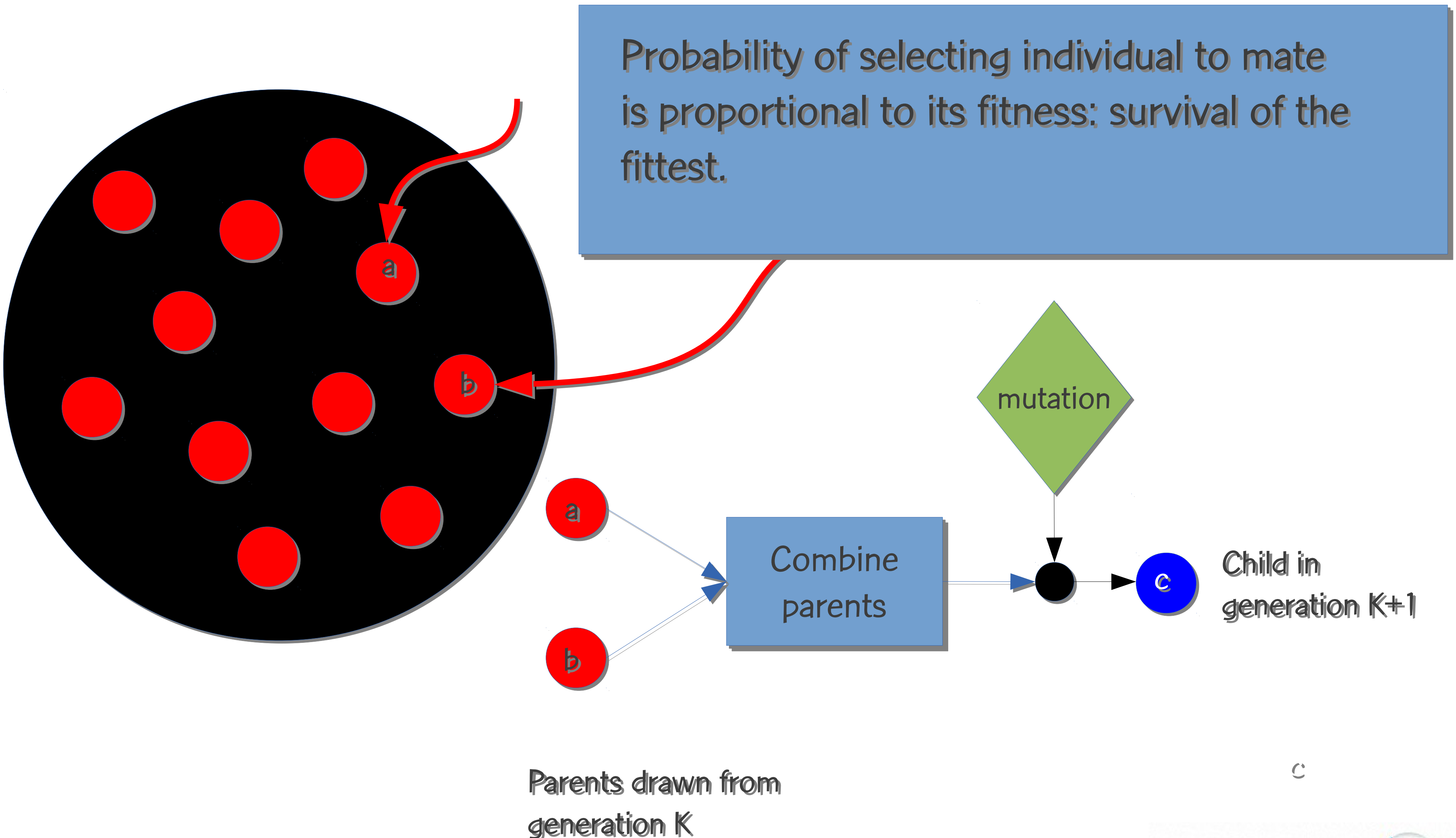
Individuals from one generation are combined to produce offspring that make up next generation.

Genetic algorithms

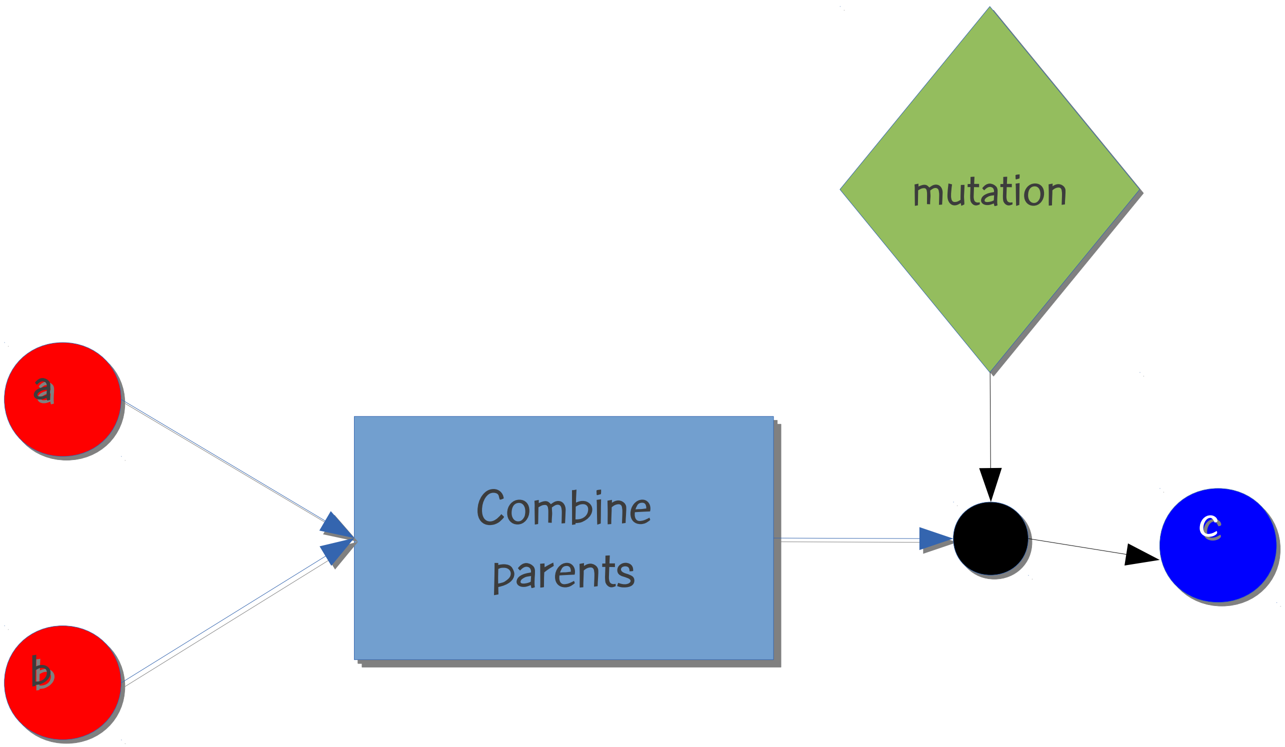


Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.

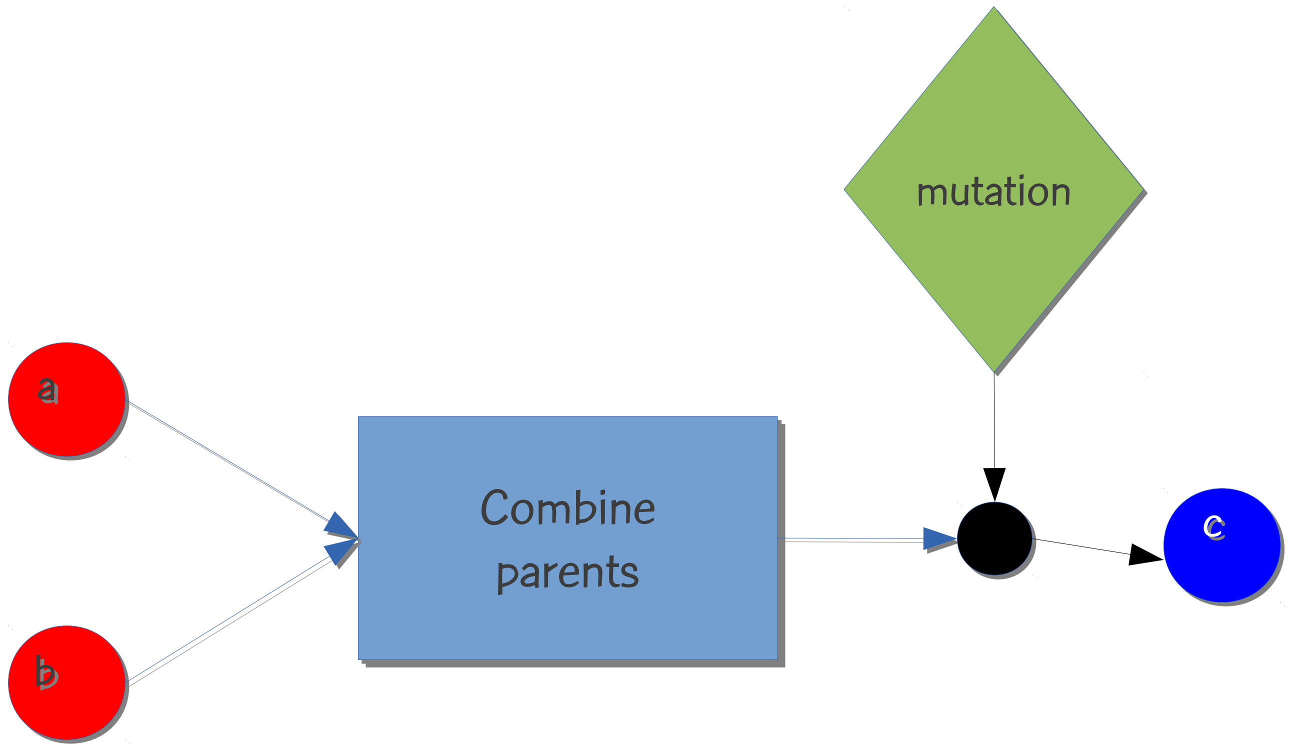
Genetic algorithms



Crossover and mutation



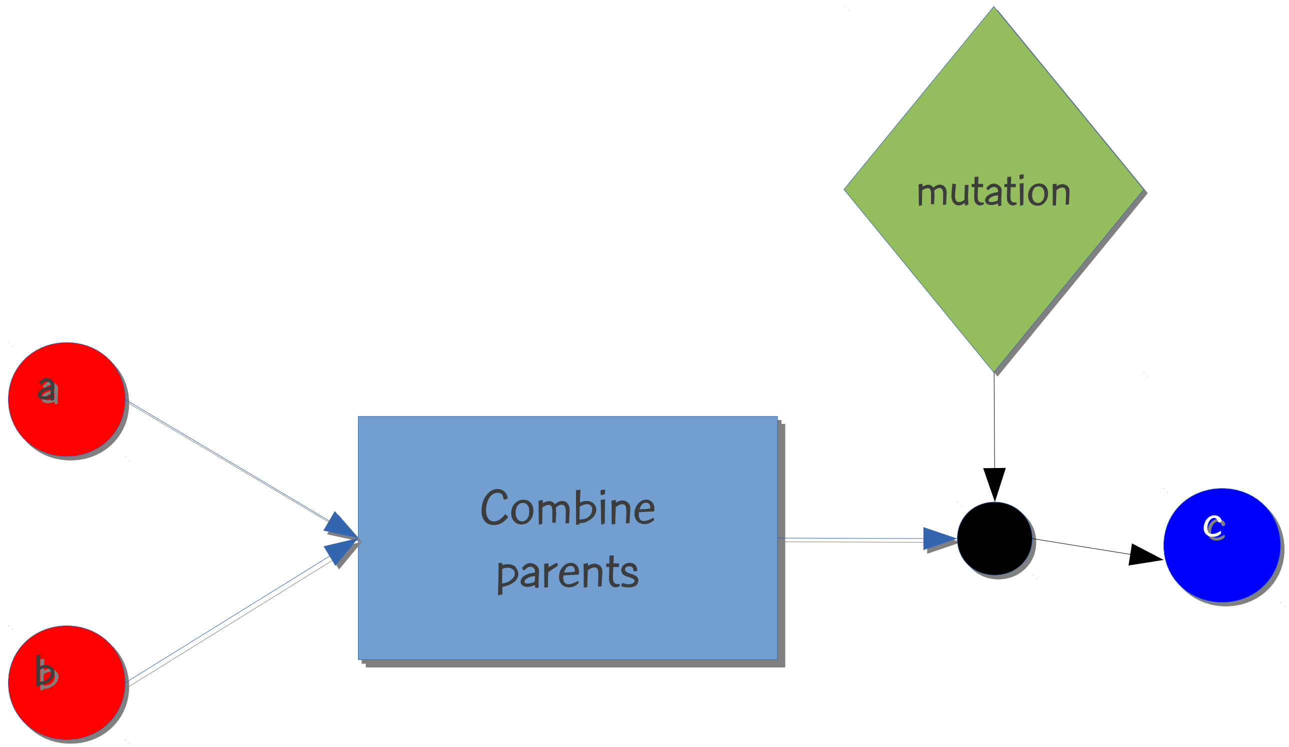
Crossover and mutation



Crossover: Combines parents ... passing along to offspring characteristics of each parent ...

Intensification of search

Crossover and mutation

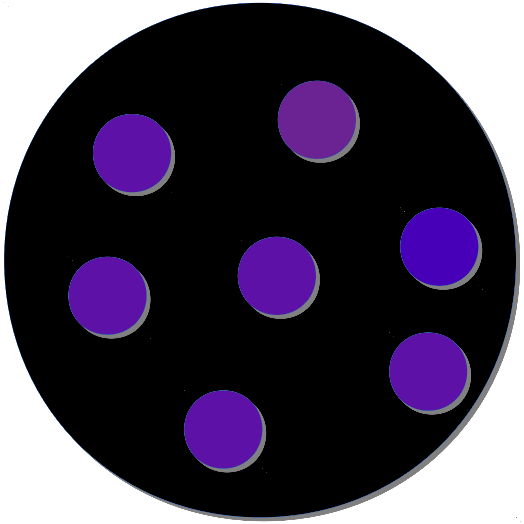


Mutation: Randomly changes chromosome of offspring …

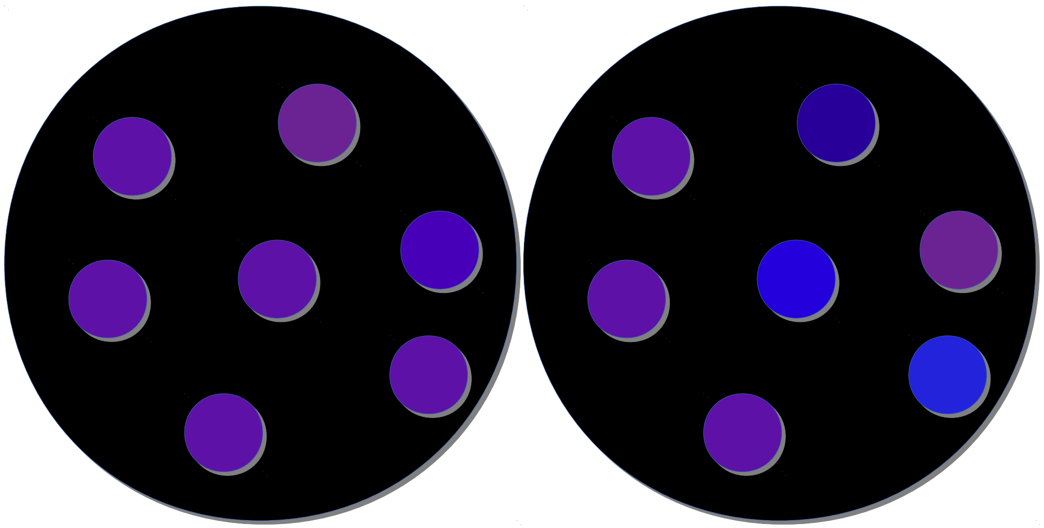
Driver of evolutionary process …

Diversification of search

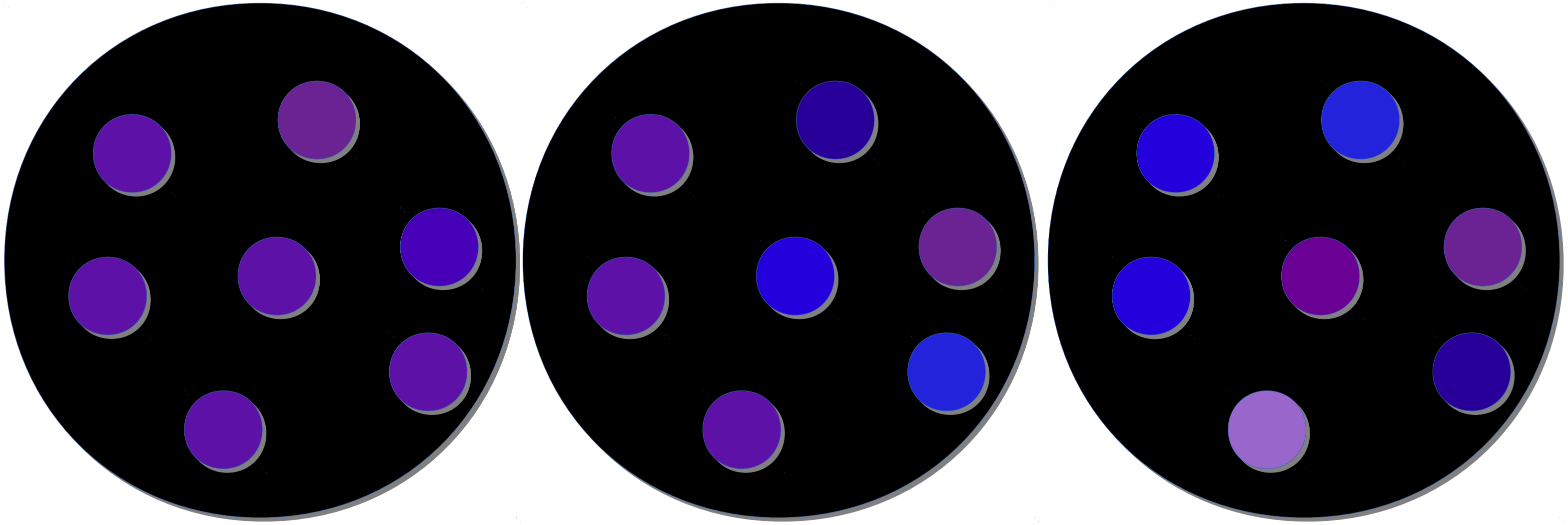
Evolution of solutions



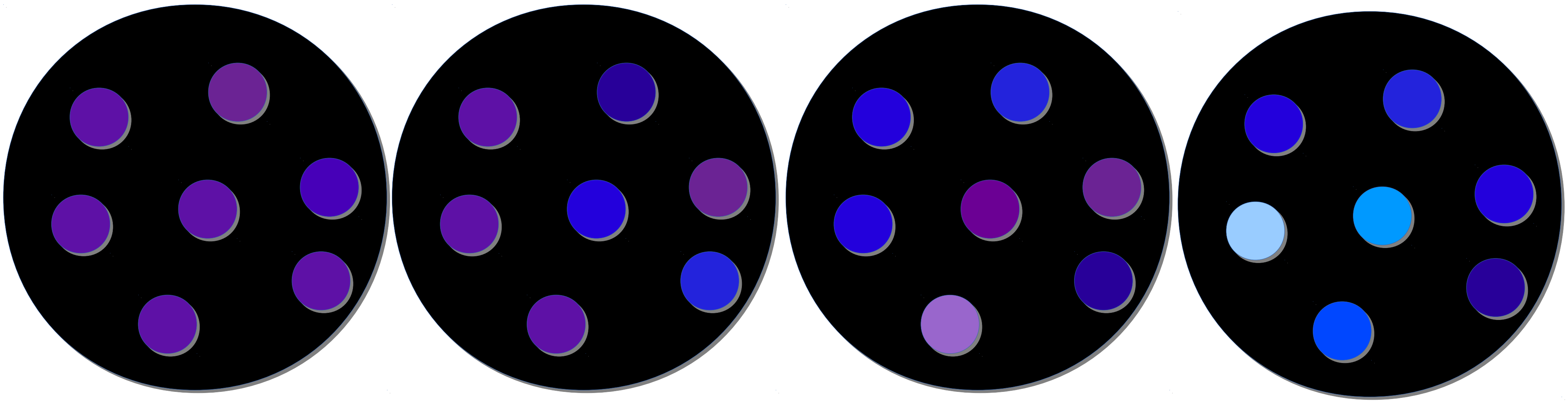
Evolution of solutions



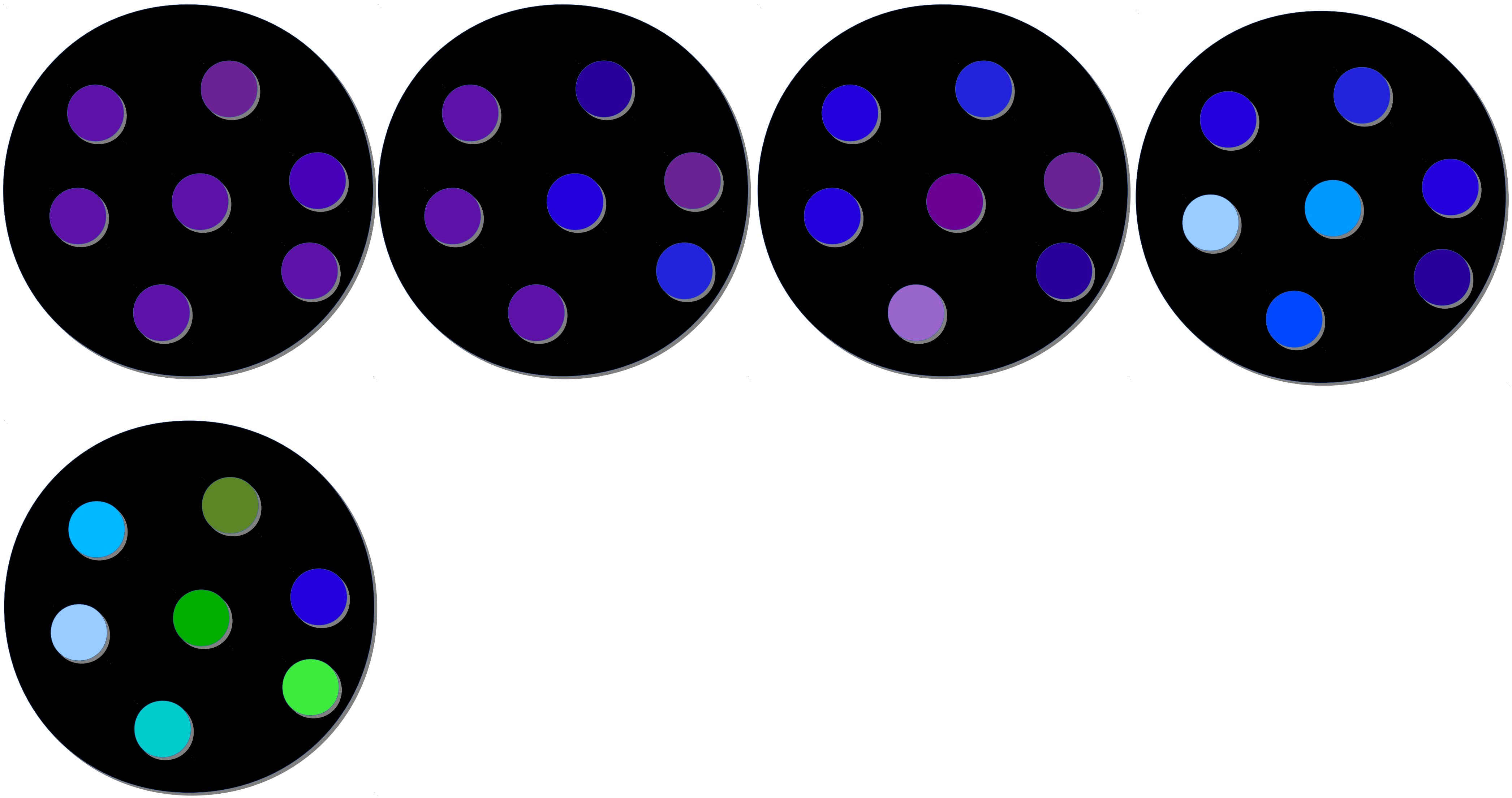
Evolution of solutions



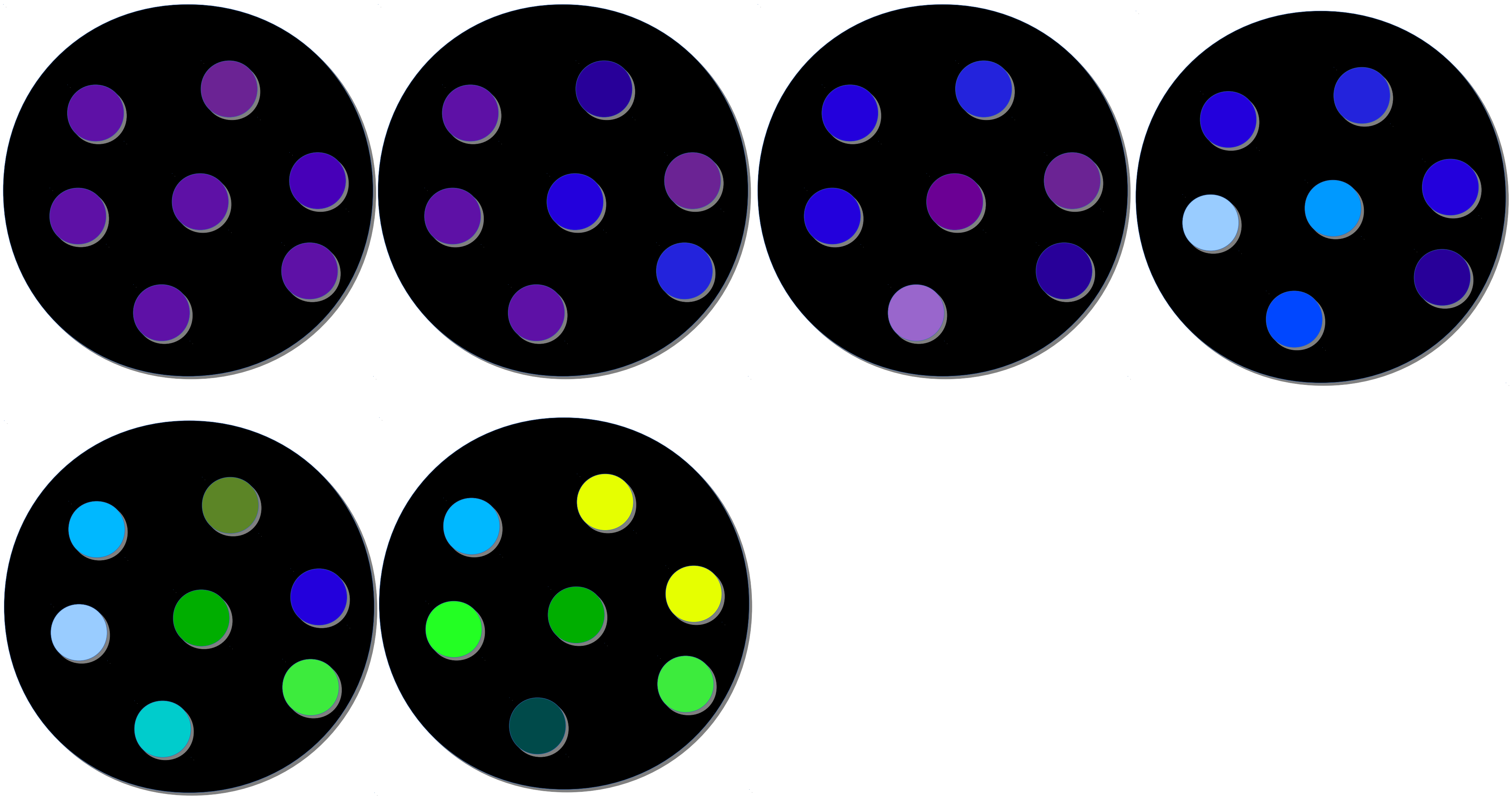
Evolution of solutions



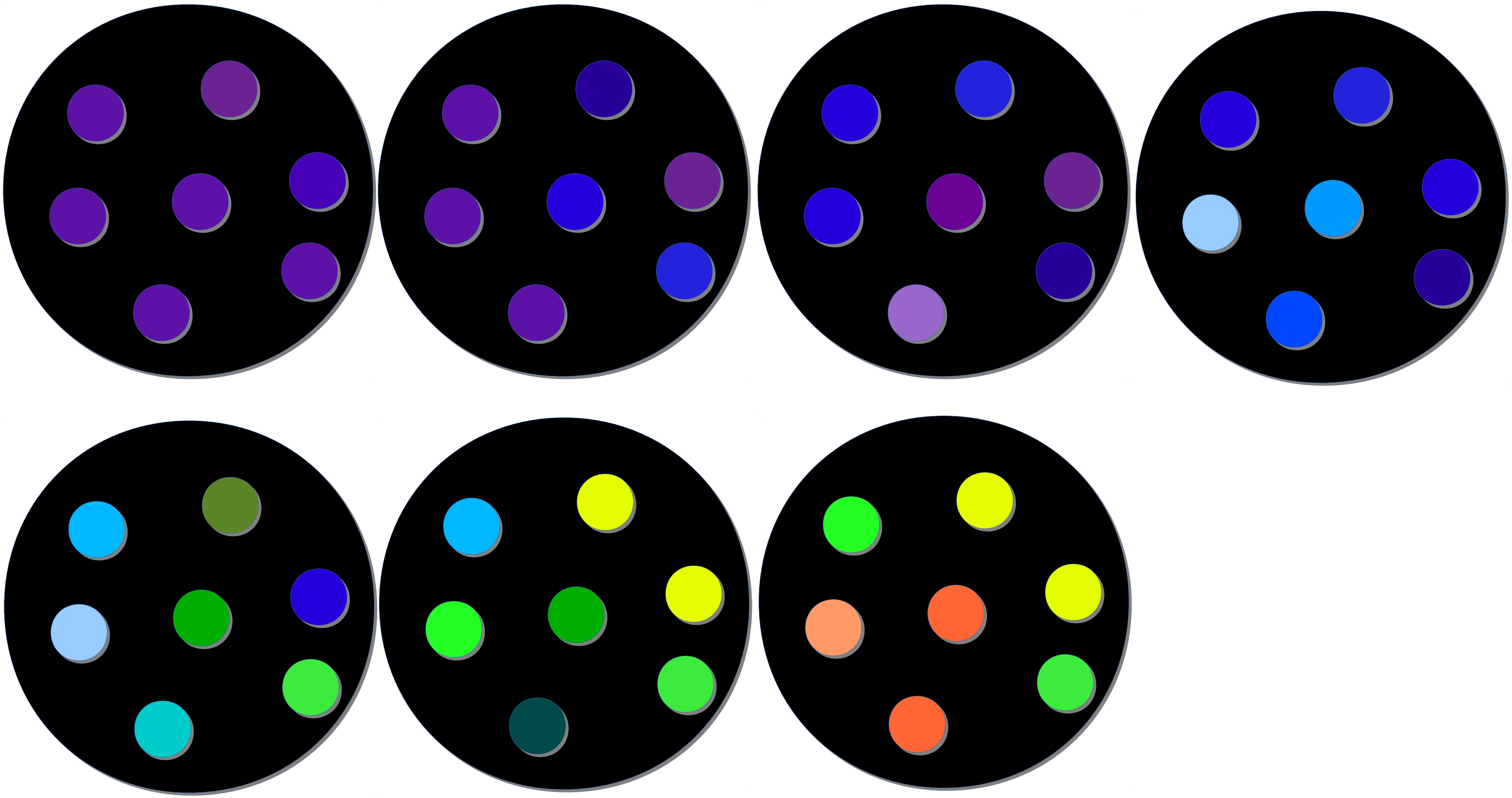
Evolution of solutions



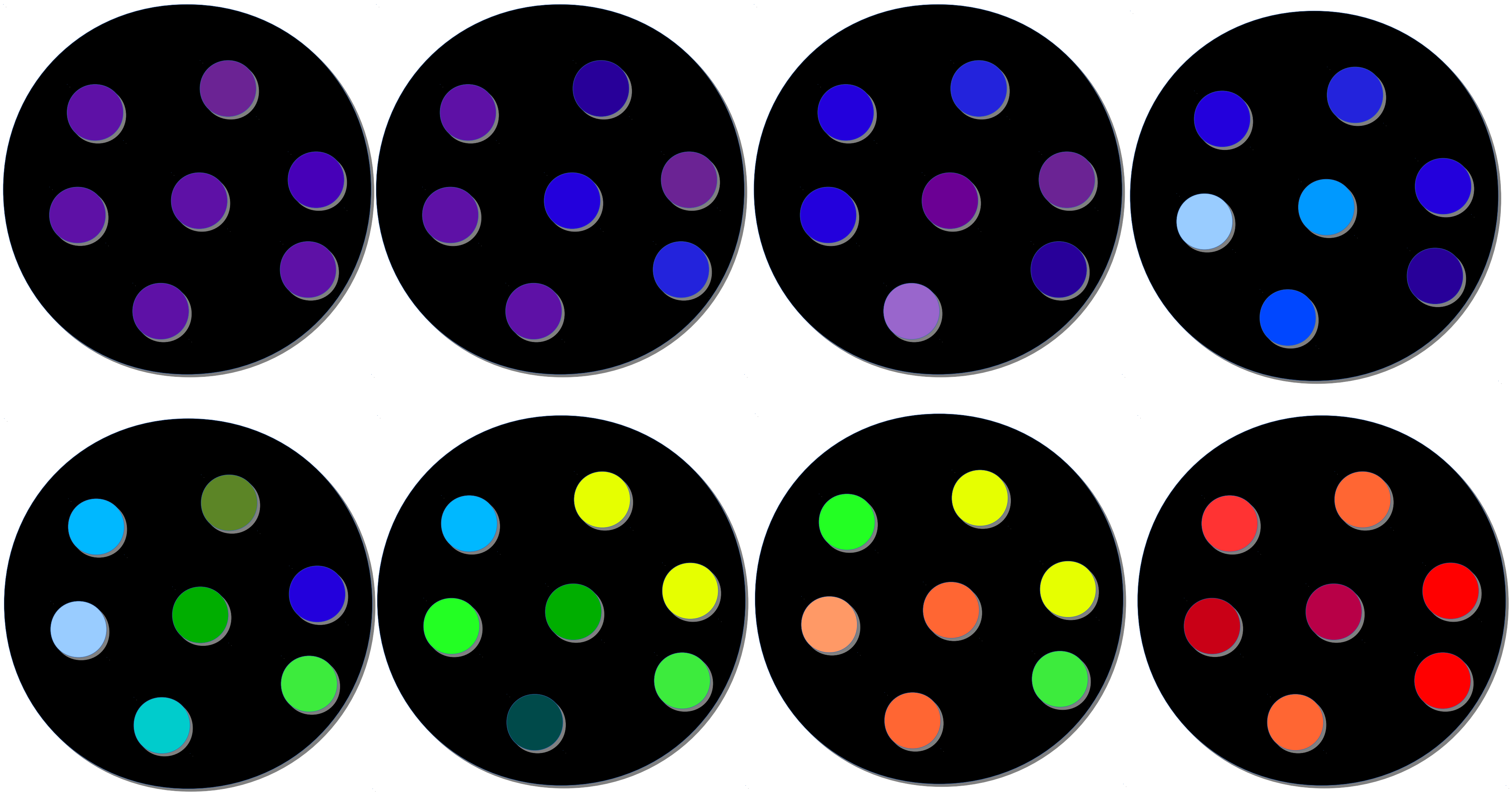
Evolution of solutions



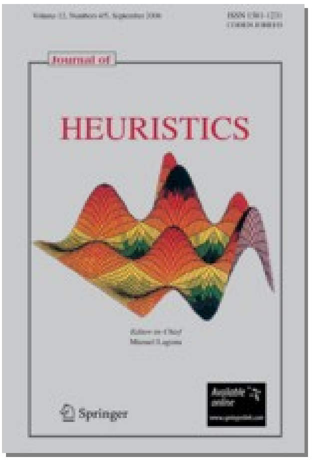
Evolution of solutions



Evolution of solutions



Reference



J.F. Gonçalves and M.G.C.R., "Biased random-key genetic algorithms for combinatorial optimization," J. of Heuristics, vol.17, pp. 487-525, 2011.

Tech report version:

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

Genetic algorithms and 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

- 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 (key, or value of gene) 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 (key, or value of gene) 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 (key, or value of gene) 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 (key, or value of gene) 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 (key, or value of gene) 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 (key, or value of gene) 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 (key, or value of gene) 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

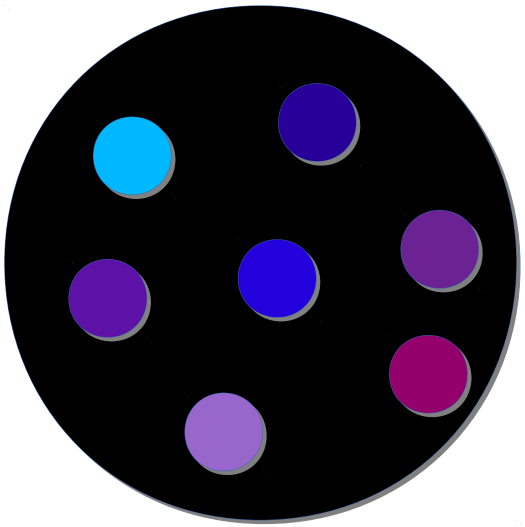
- Mating is done using parametrized uniform crossover (Spears & DeJong , 1990)
- For each gene, flip a biased coin to choose which parent passes the allele (key, or value of gene) to the child.

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.

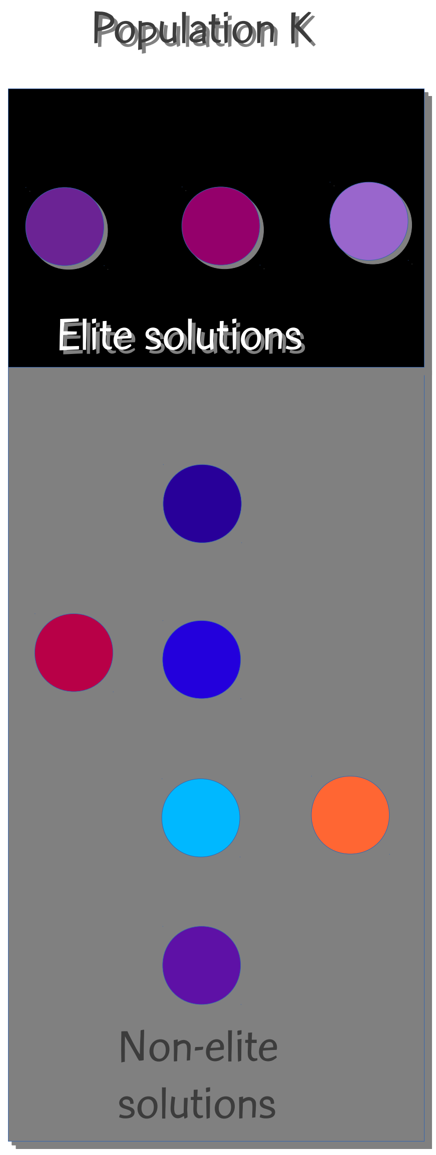
GAs and random keys

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).



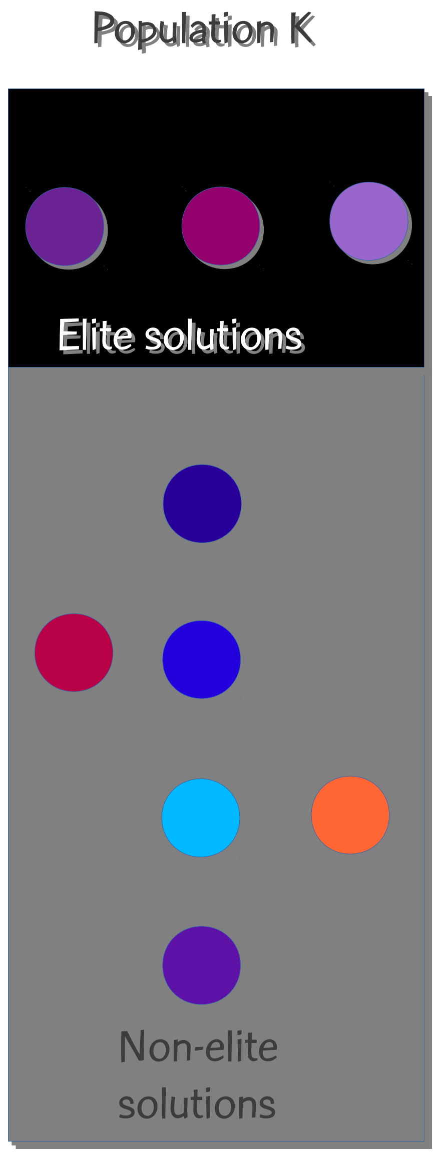
GAs and random keys

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



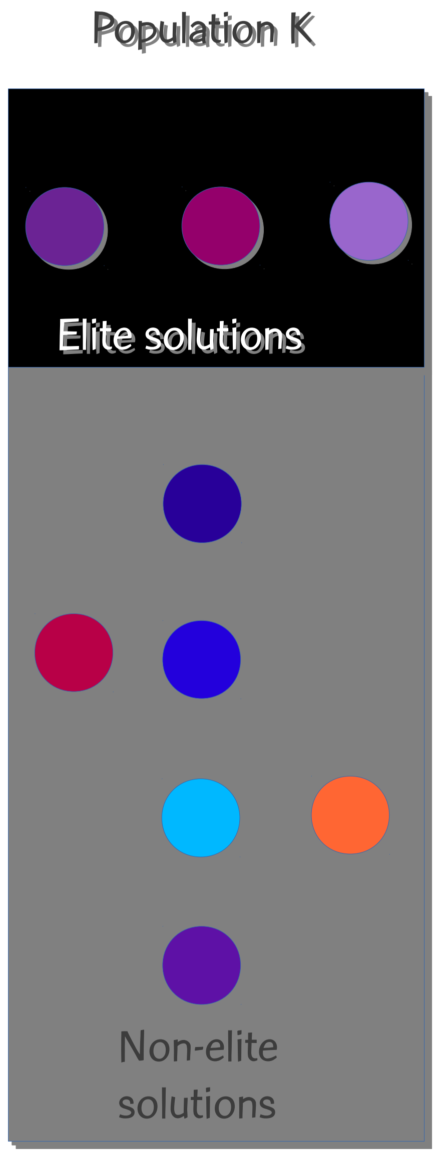
GAs and random keys

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



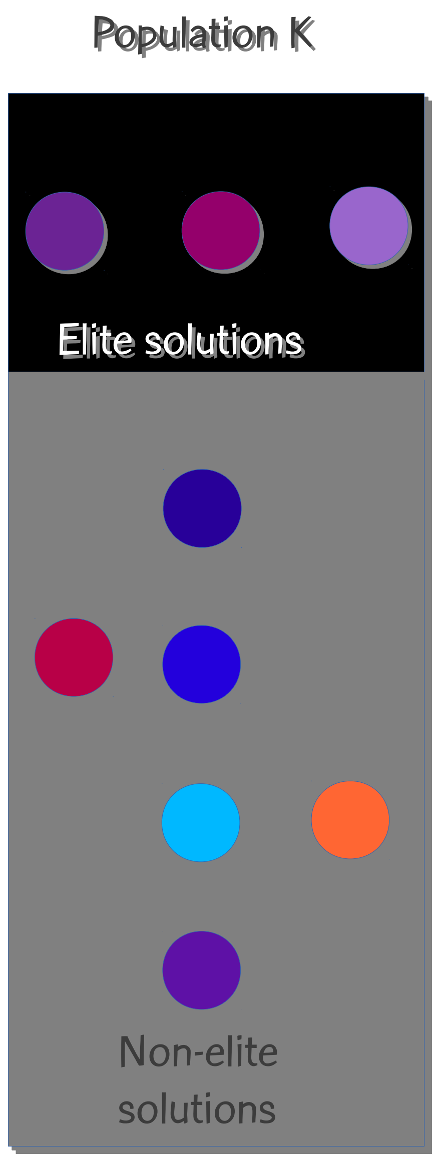
GAs and random keys

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



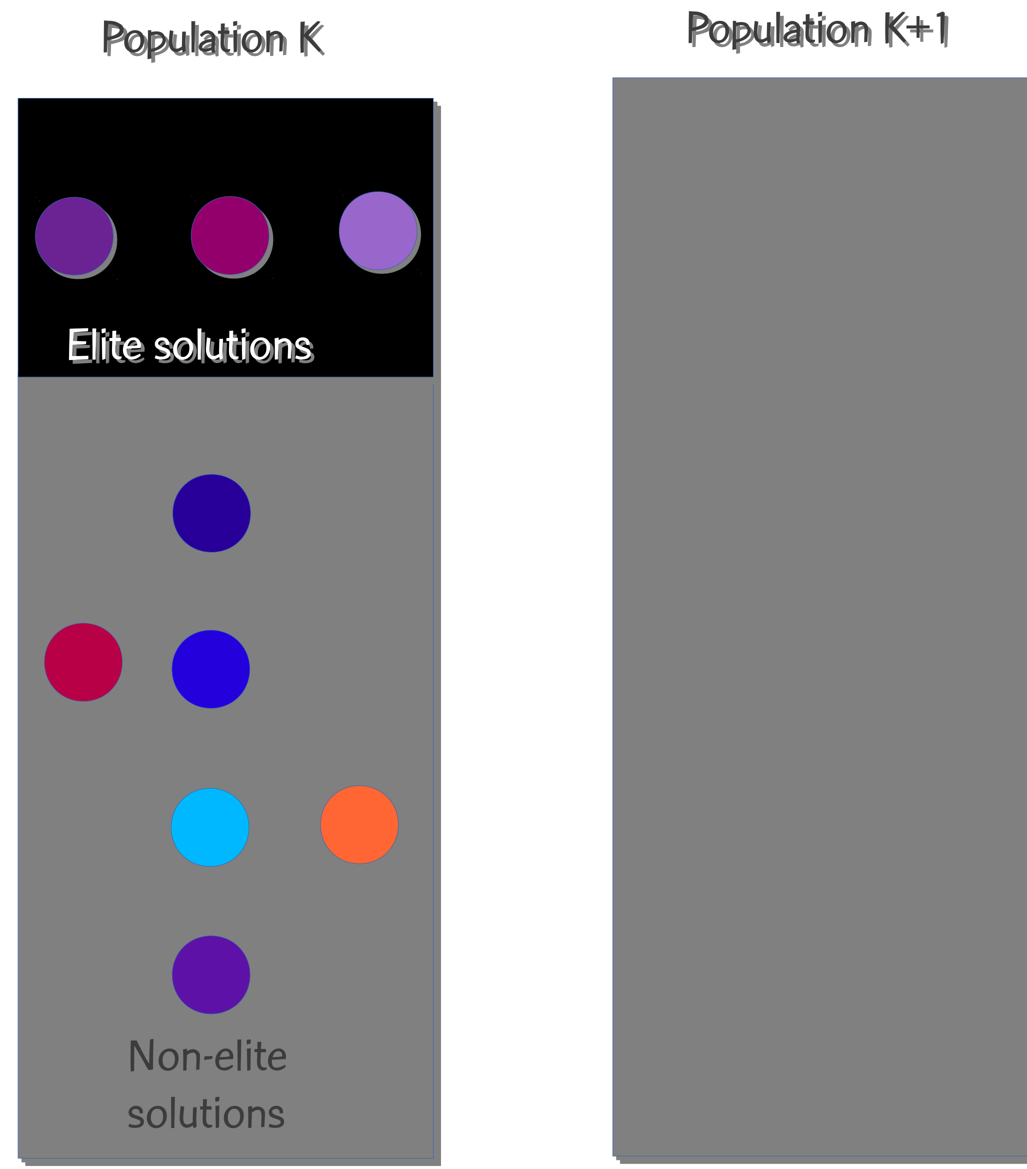
GAs and random keys

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

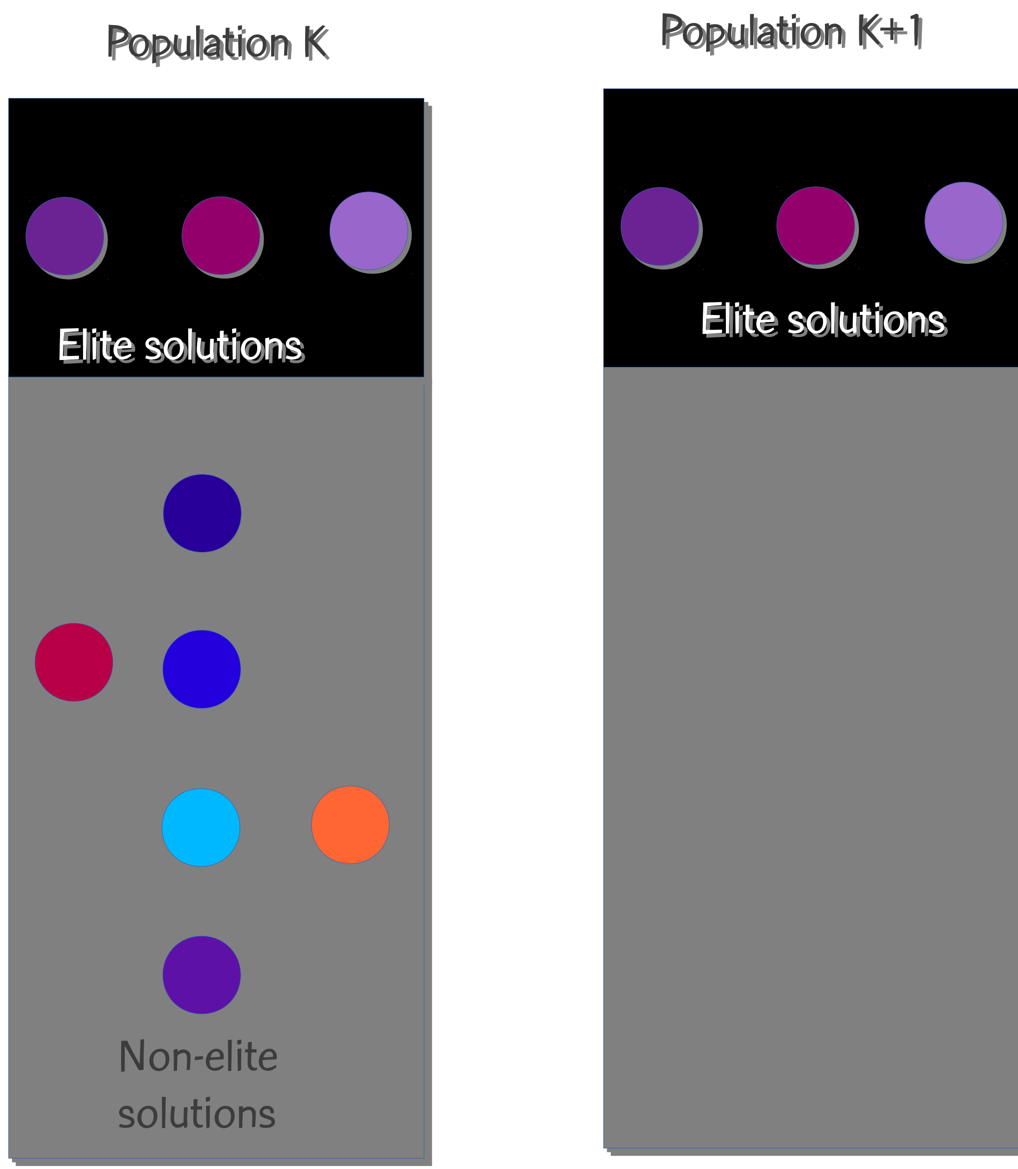
Evolutionary dynamics



GAs and random keys

Evolutionary dynamics

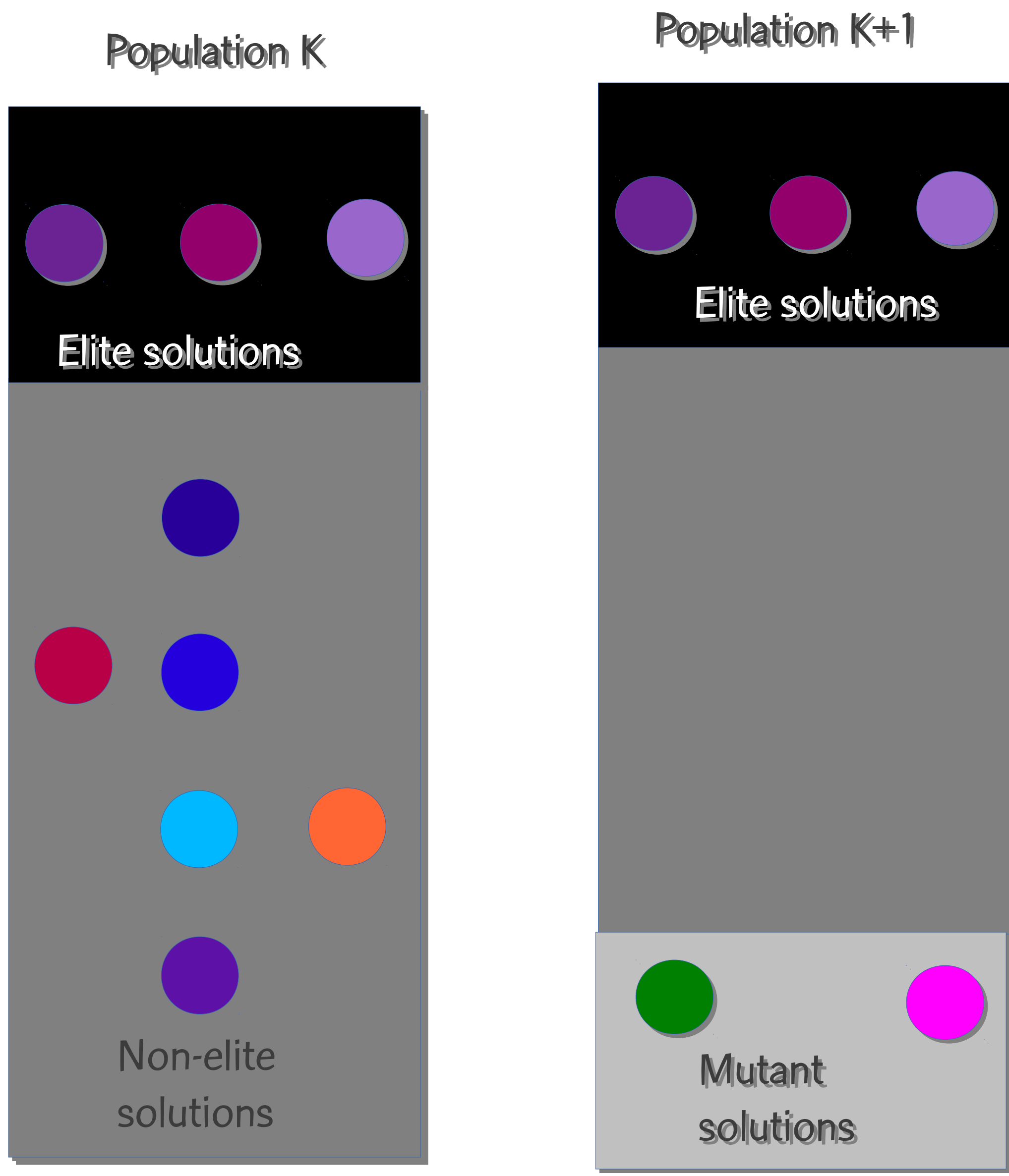
- Copy elite solutions from population K to population K+1



GAs and random keys

Evolutionary dynamics

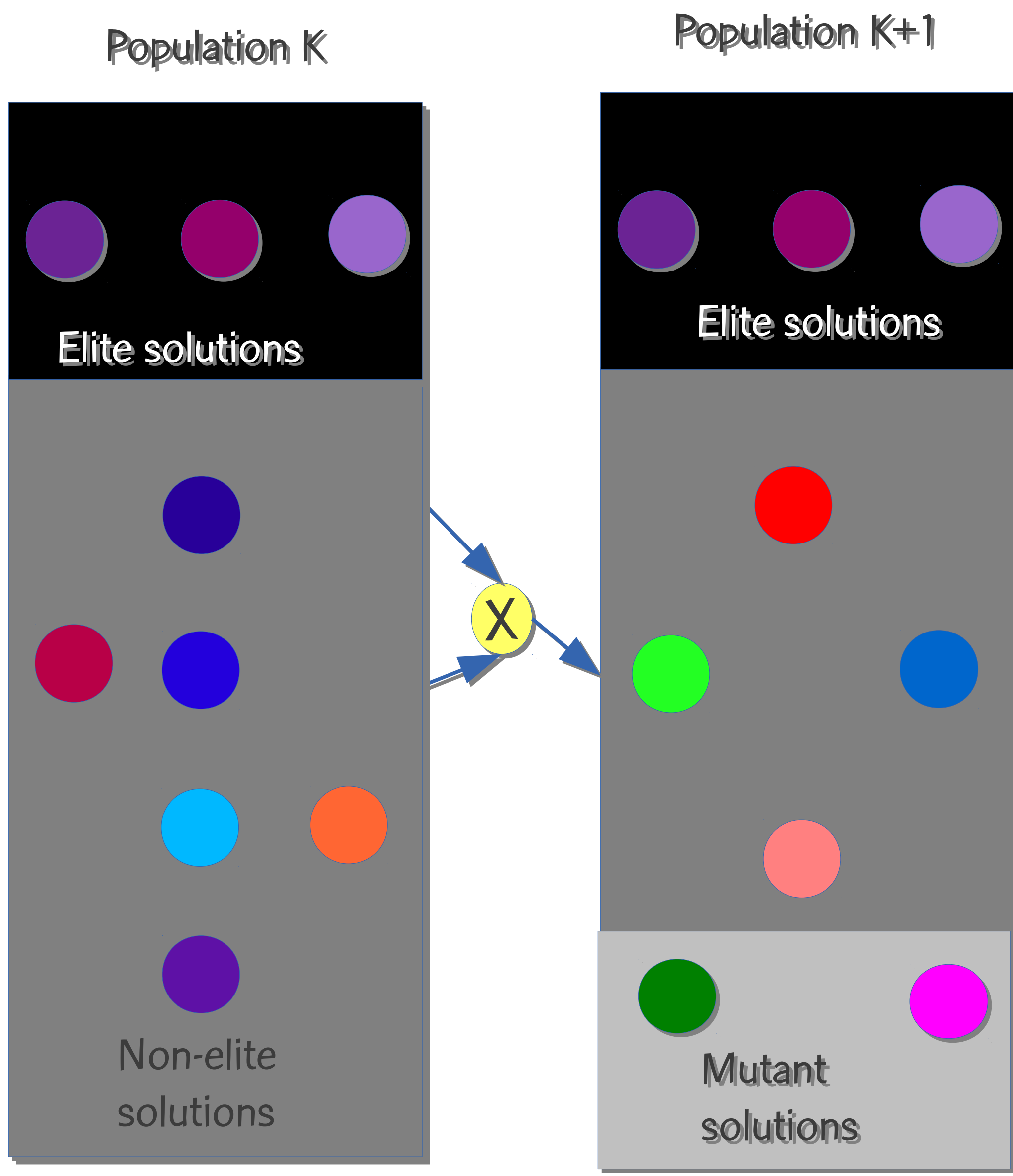
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1



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
- 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 genetic algorithm

- A biased random key genetic algorithm (BRKGA) is a random key genetic algorithm (RKGA).

Biased random key genetic algorithm

- A biased random key genetic algorithm (BRKGA) is a random key genetic algorithm (RKGA).
- BRKGA and RKGA differ in how mates are chosen for crossover and how parametrized uniform crossover is applied.

How RKGA & BRKGA differ

RKGA

both parents chosen at random from entire population

BRKGA

How RKGA & BRKGA differ

RKGA

both parents chosen at random from entire population

BRKGA

both parents chosen at random but one parent chosen from population of elite solutions

How RKGA & BRKGA differ

RKGA

both parents chosen at random from entire population

either parent can be parent A in parametrized uniform crossover

BRKGA

both parents chosen at random but one parent chosen from population of elite solutions

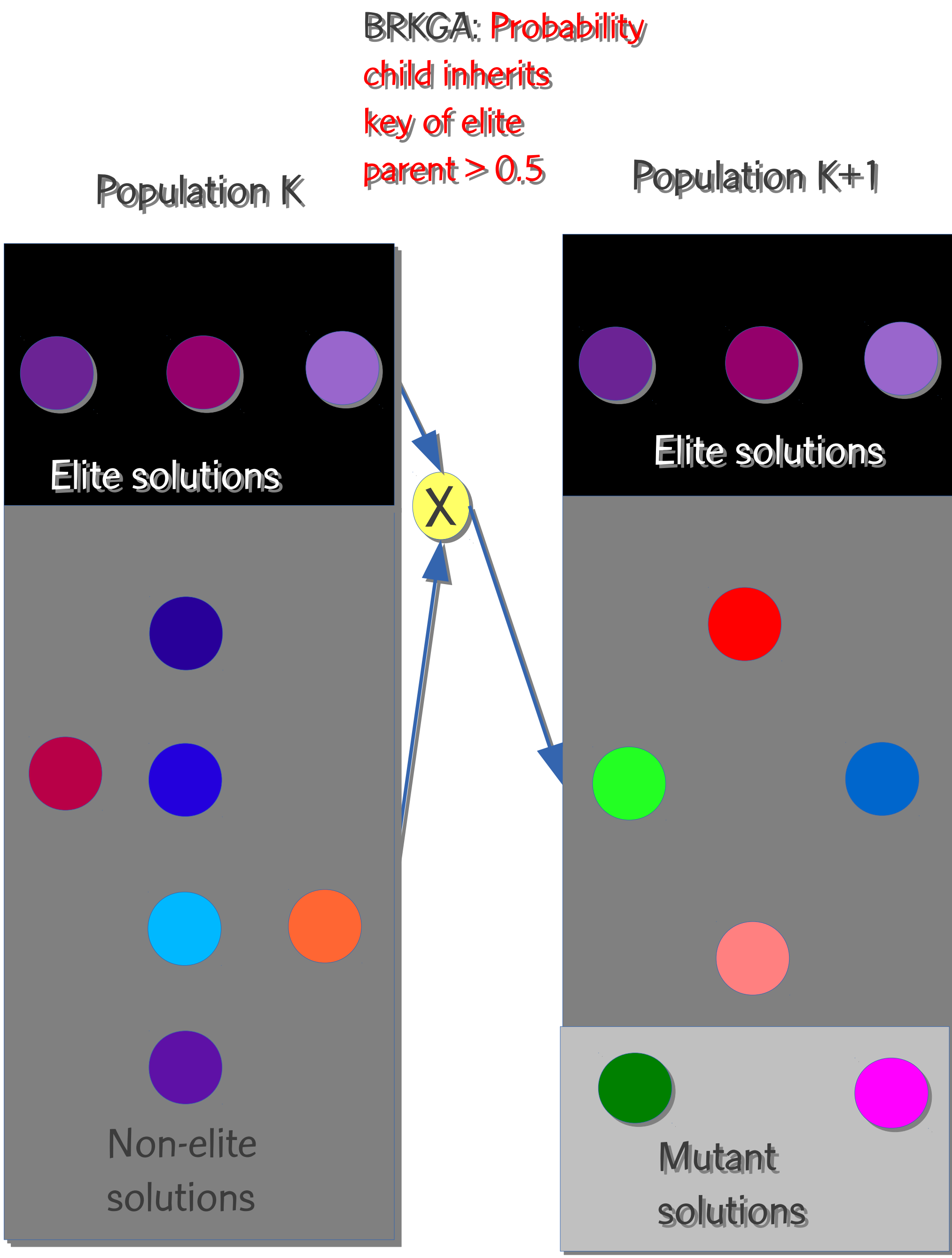
How RKGA & BRKGA differ

RKGA	BRKGA
both parents chosen at random from entire population	both parents chosen at random but one parent chosen from population of elite solutions
either parent can be parent A in parametrized uniform crossover	best fit parent is parent A in parametrized uniform crossover

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 other solution 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 vectors of random keys

Observations

- Random method: keys are randomly generated so solutions are always vectors of random keys
- Elitist strategy: best solutions are passed without change from one generation to the next (incumbent is kept)

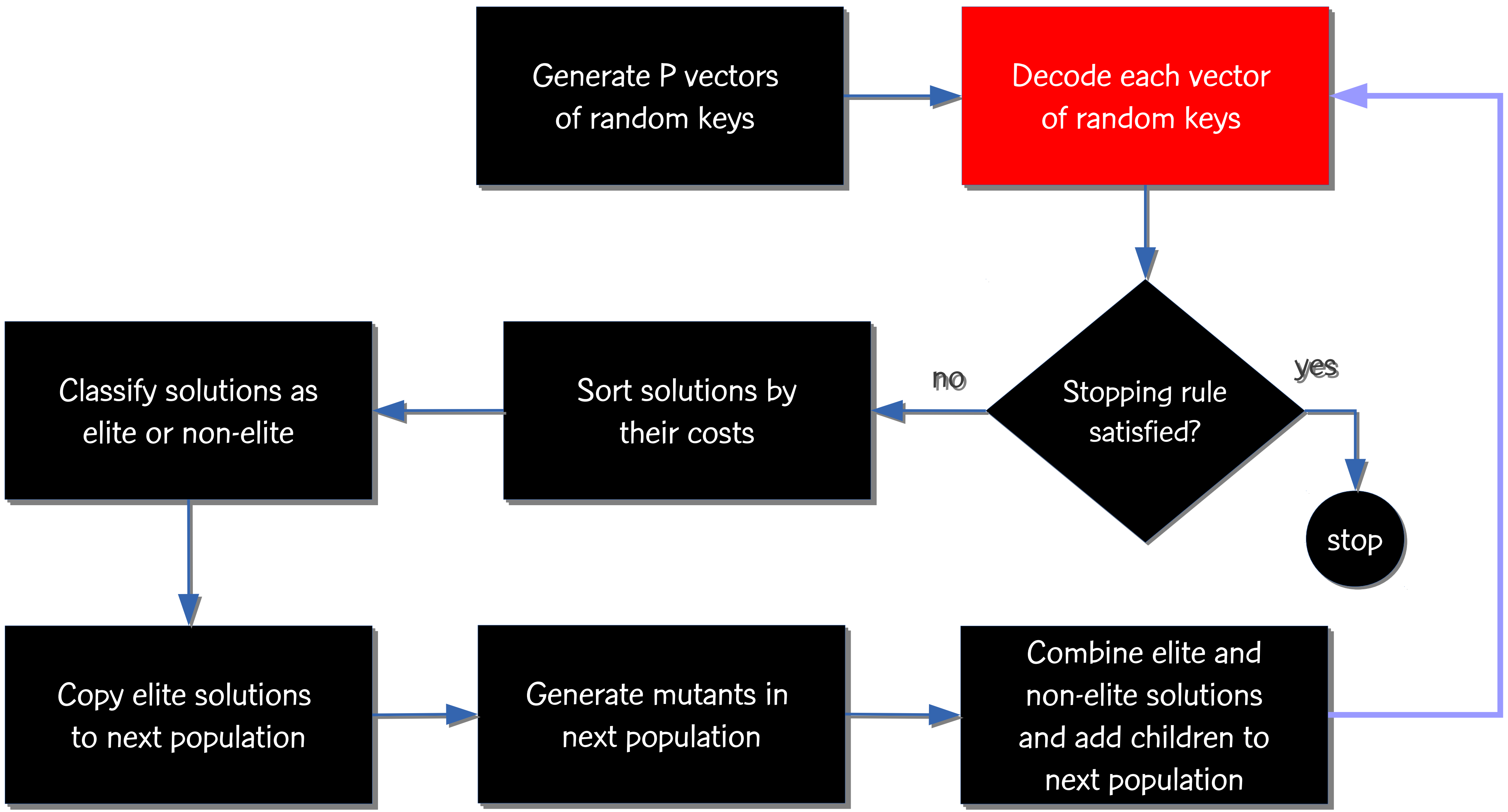
Observations

- Random method: keys are randomly generated so solutions are always vectors of random keys
- Elitist strategy: best solutions are passed without change from one generation to the next (incumbent is kept)
- 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 Not so in the RKGA of Bean.

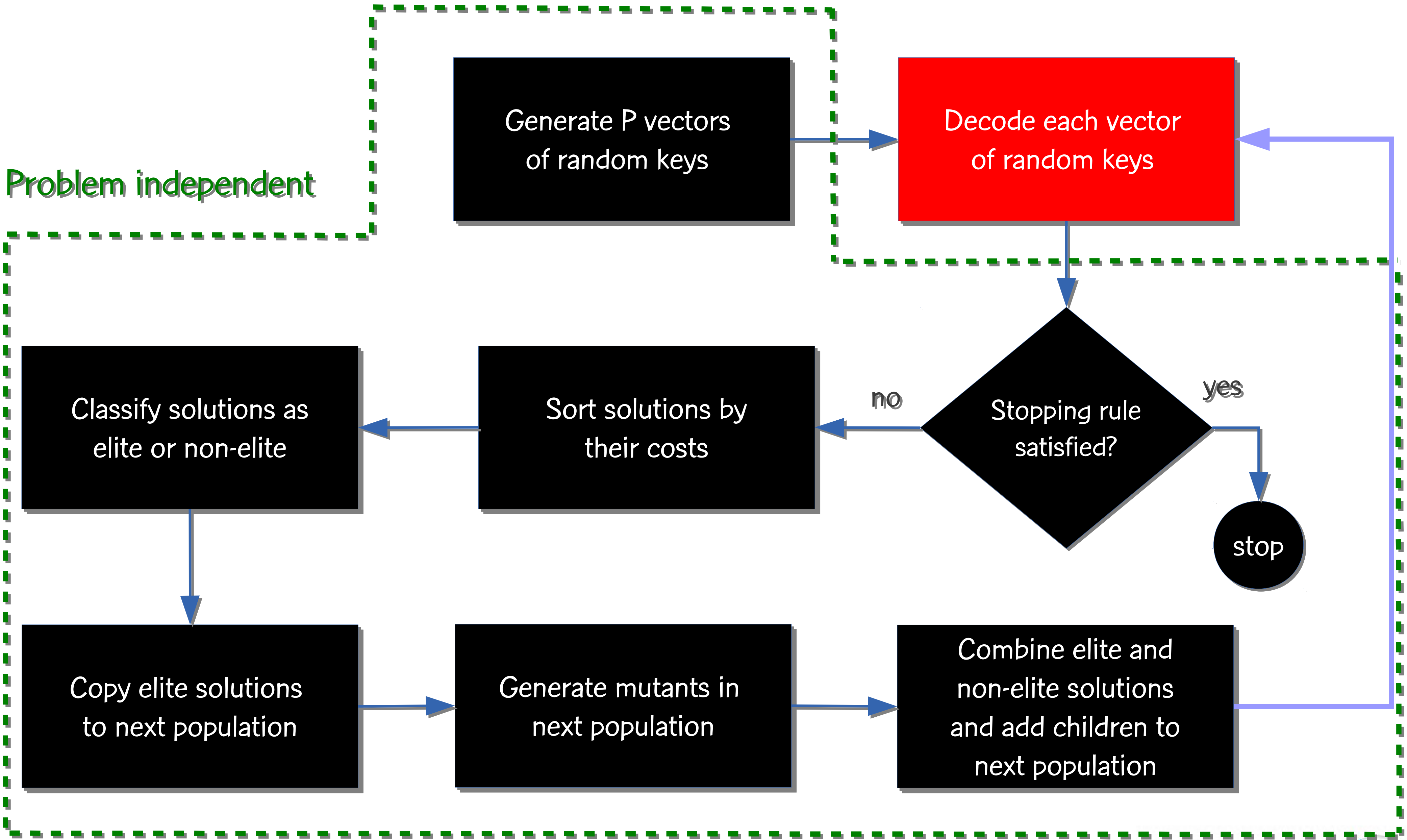
Observations

- Random method: keys are randomly generated so solutions are always vectors of random keys
- Elitist strategy: best solutions are passed without change from one generation to the next (incumbent is kept)
- 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 Not so in the RKGA of Bean.
- No mutation in crossover: mutants are used instead (they play same role as mutation in GAs … help escape local optima)

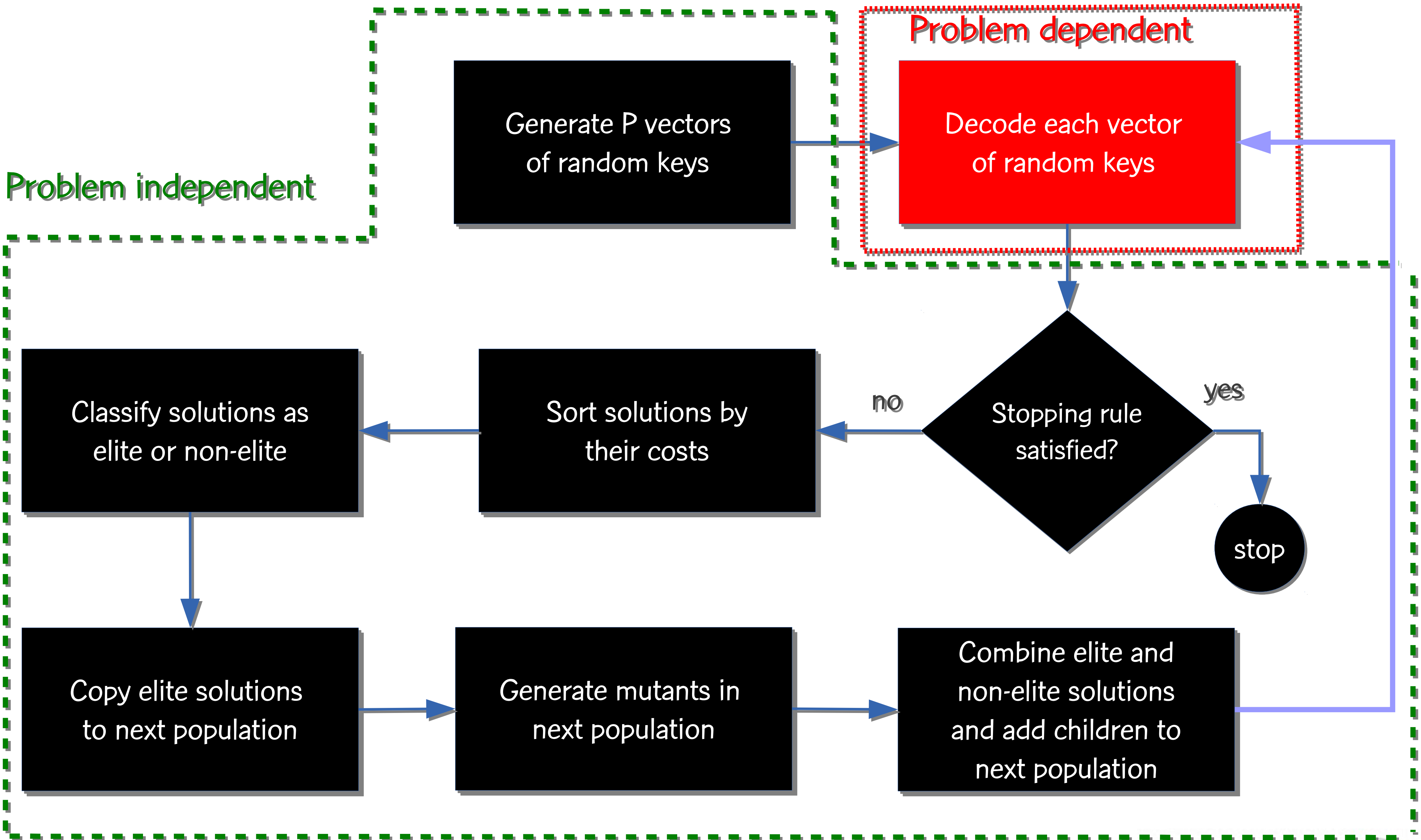
Framework for biased random-key genetic algorithms



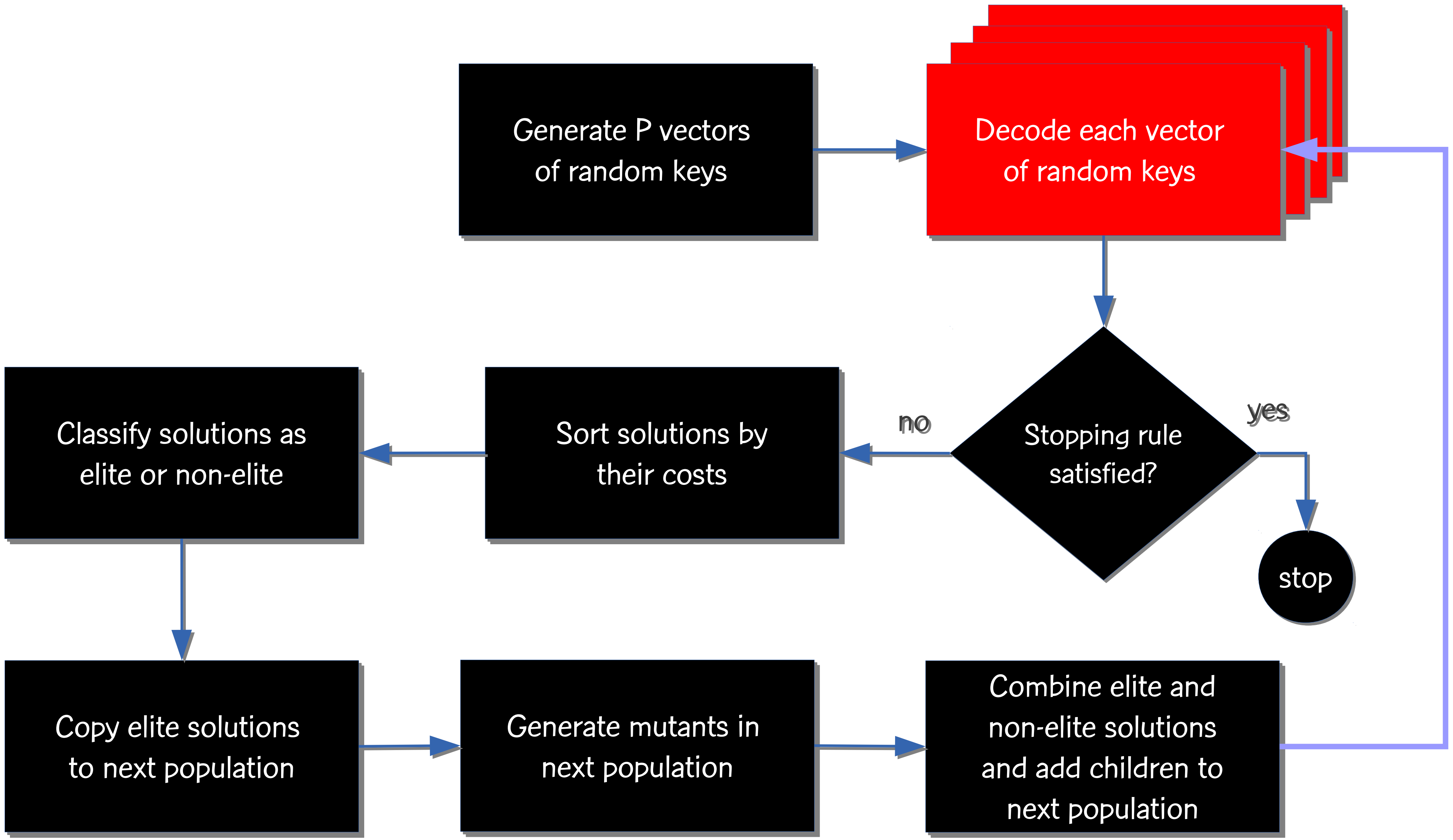
Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Decoding of random key vectors can be done in parallel



Specifying a BRKGA

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)

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)

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

Specifying a biased random-key GA

Parameters:

- Size of population
- Size of elite partition
- Size of mutant set
- Child inheritance probability
- Restart strategy parameter
- Stopping criterion

Specifying a biased random-key GA

Parameters:

- Size of population: a function of N, say N or 2N
- Size of elite partition
- Size of mutant set
- Child inheritance probability
- Restart strategy parameter
- Stopping criterion

Specifying a biased random-key GA

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
- Restart strategy parameter
- Stopping criterion

Specifying a biased random-key GA

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
- Restart strategy parameter
- Stopping criterion

Specifying a biased random-key GA

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
- Restart strategy parameter
- Stopping criterion

Specifying a biased random-key GA

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
- Restart strategy parameter: a function of N, say 2N or 10N
- Stopping criterion

Specifying a biased random-key GA

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
- Restart strategy parameter: a function of N, say 2N or 10N
- Stopping criterion: e.g. time, # generations, solution quality, # generations without improvement

brkgaAPI: A C++ API for BRKGA

- Efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.

brkgaAPI: A C++ API for BRKGA

- Efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics

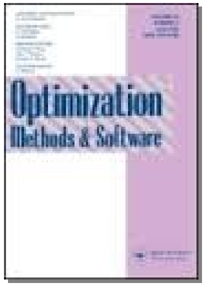
brkgaAPI: A C++ API for BRKGA

- Efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics
- Implemented in C++ and may benefit from shared-memory parallelism if available.

brkgaAPI: A C++ API for BRKGA

- Efficient and easy-to-use object oriented application programming interface (API) for the algorithmic framework of BRKGA.
- Cross-platform library handles large portion of problem independent modules that make up the framework, e.g.
  - population management
  - evolutionary dynamics
- Implemented in C++ and may benefit from shared-memory parallelism if available.
- User only needs to implement problem-dependent decoder.

brkgaAPI: A C++ API for BRKGA



Paper: Rodrigo F. Toso and M.G.C.R., "A C++ Application Programming Interface for Biased Random-Key Genetic Algorithms," Optimization Methods & Software, published online 13 March 2014.

Software: http://www.research.att.com/~mgcr/src/brkgaAPI

Decoder for wireless backhaul planning

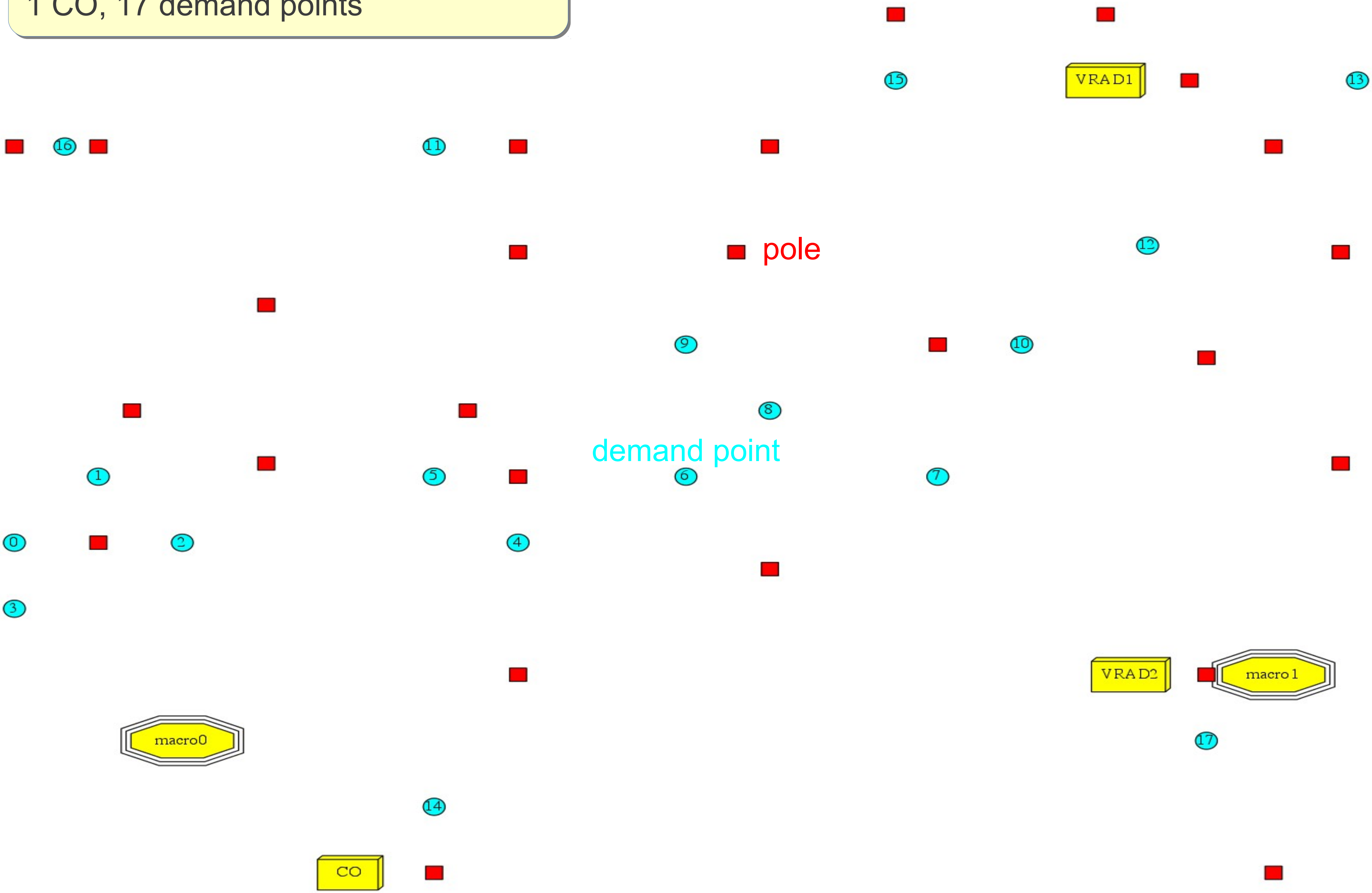
- Biased Random-Key Genetic Algorithm
  - Learn the best network layout and equipment placement
- A solution is encoded by a vector $x \in [0,1]^n$
  - Where $n = 5 \times \#$ of poles

Activation order	Activation parameters	Evaluation order	Neighborhood evaluation order	Minimum tree level of pole

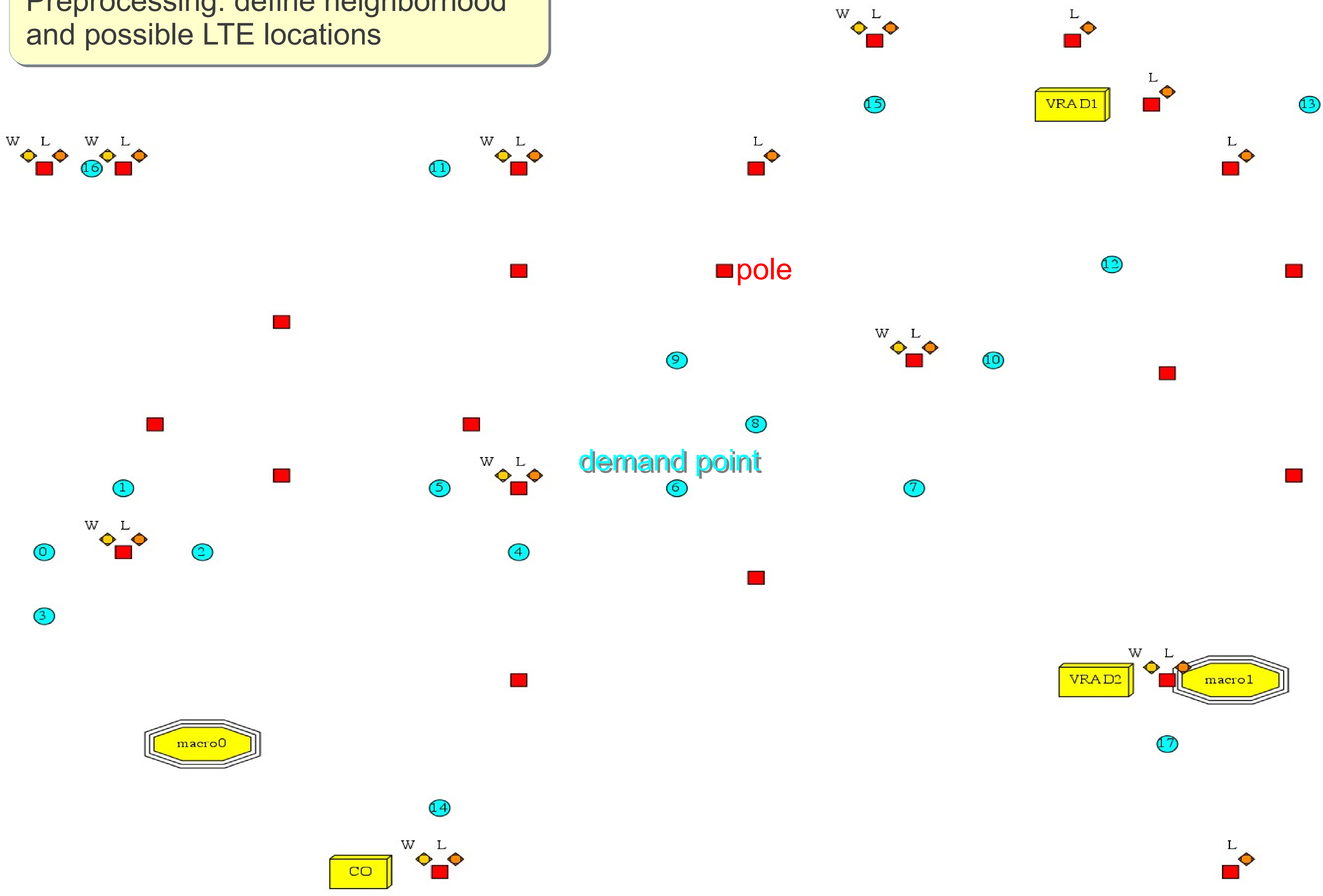
Decoder

- Define activation order & install LTE on poles
- Build backhaul graph
- Remove unused equipment
- Compute maximum flow from demand points to FAPs
- Remove unused equipment & poles
- Compute cost and revenue and return objective function value

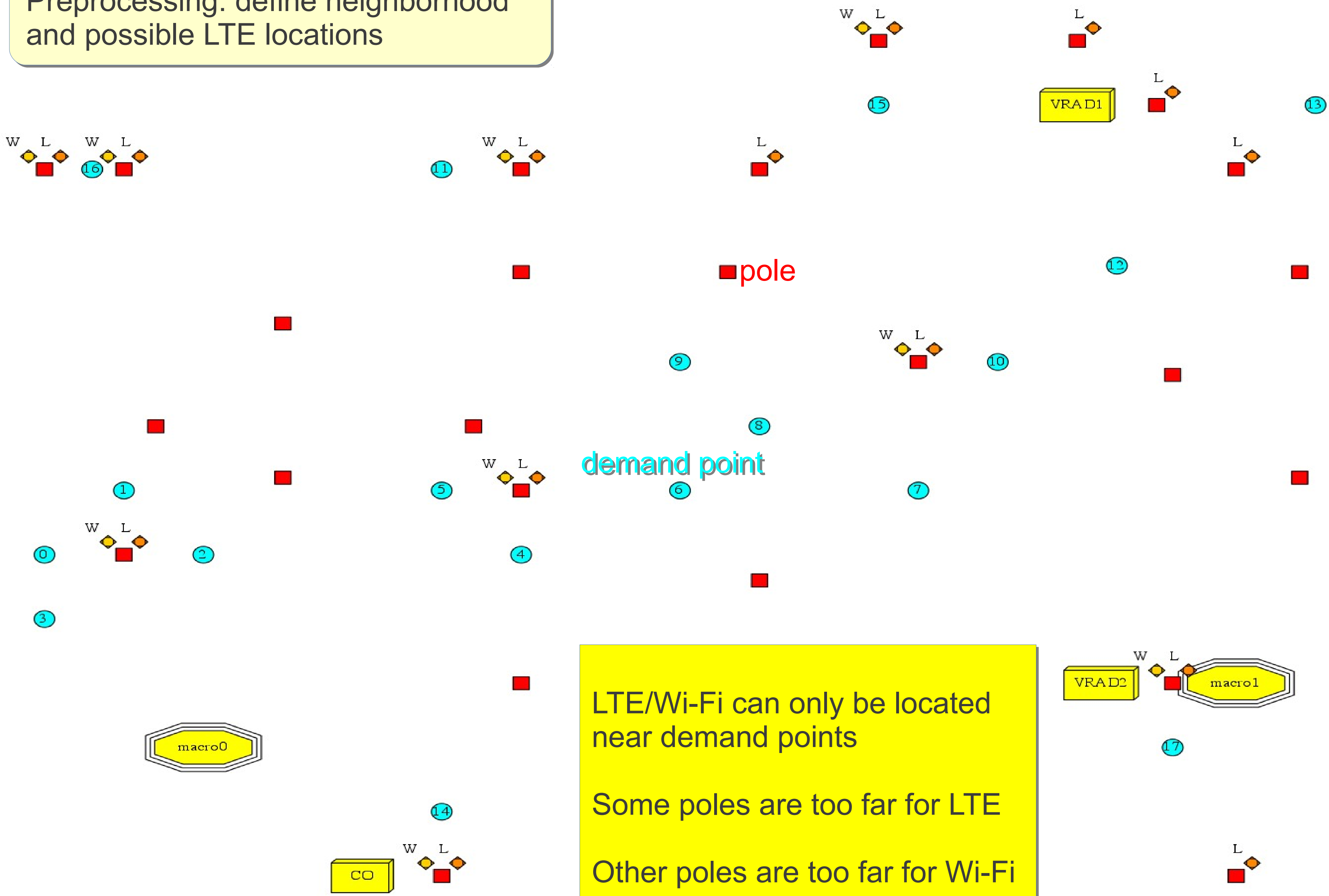
Example: 25 poles, 2 macros, 2 RTs, 1 CO, 17 demand points



Preprocessing: define neighborhood and possible LTE locations



Preprocessing: define neighborhood and possible LTE locations



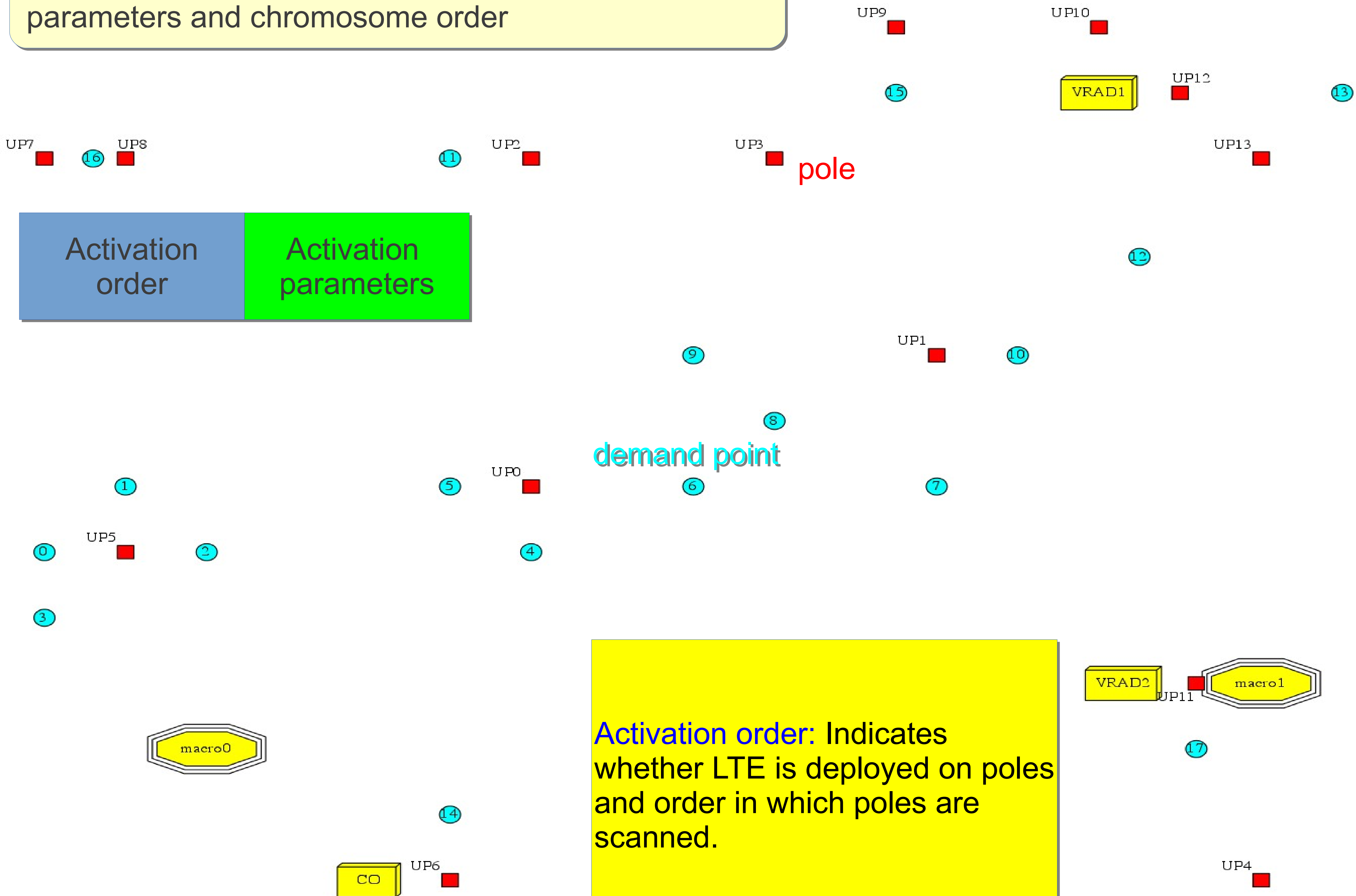
LTE/Wi-Fi can only be located near demand points

Some poles are too far for LTE

Other poles are too far for Wi-Fi

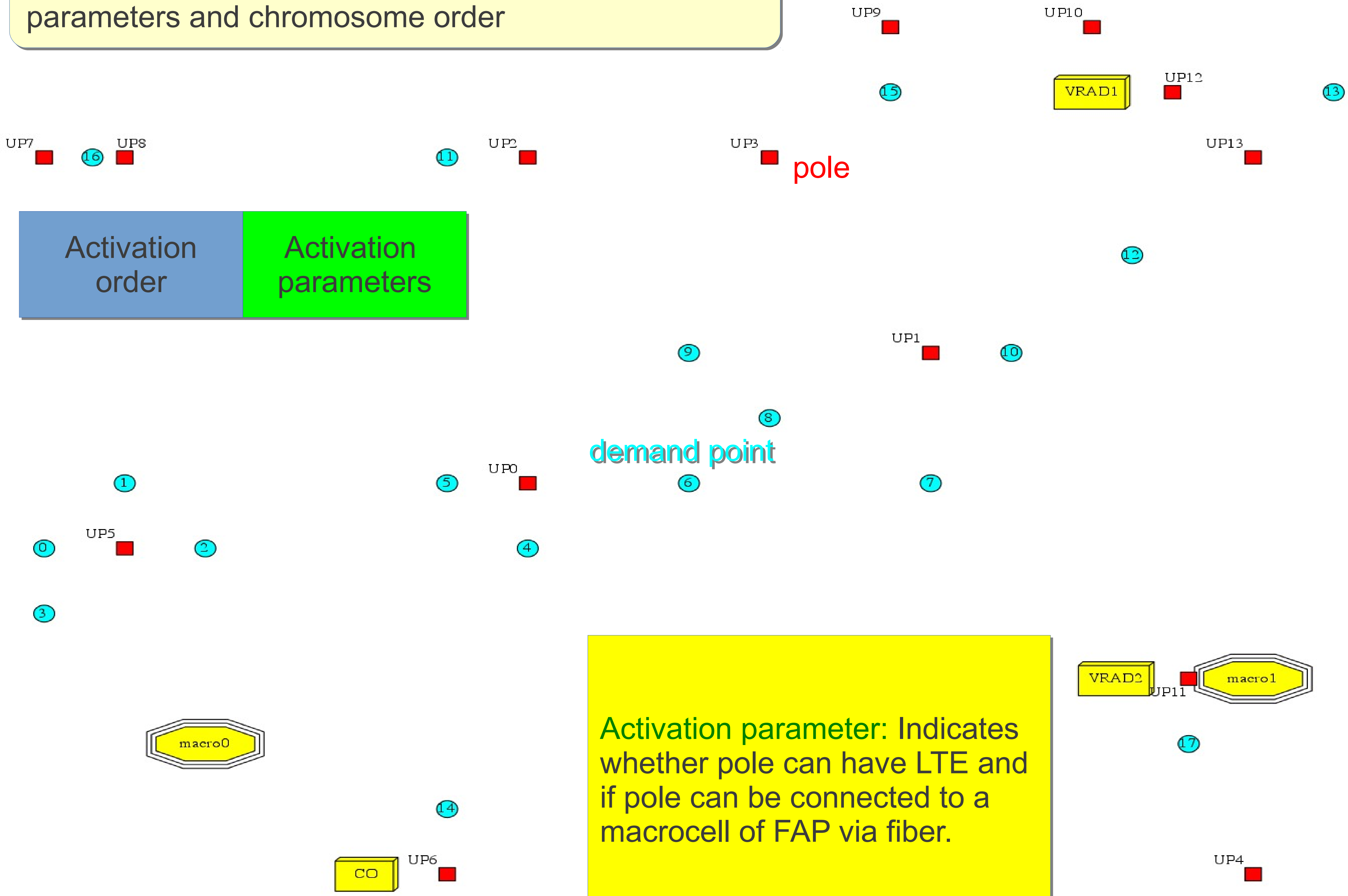
Activation: LTE deployment according to activation parameters and chromosome order

Activation order	Activation parameters



Activation order: Indicates whether LTE is deployed on poles and order in which poles are scanned.

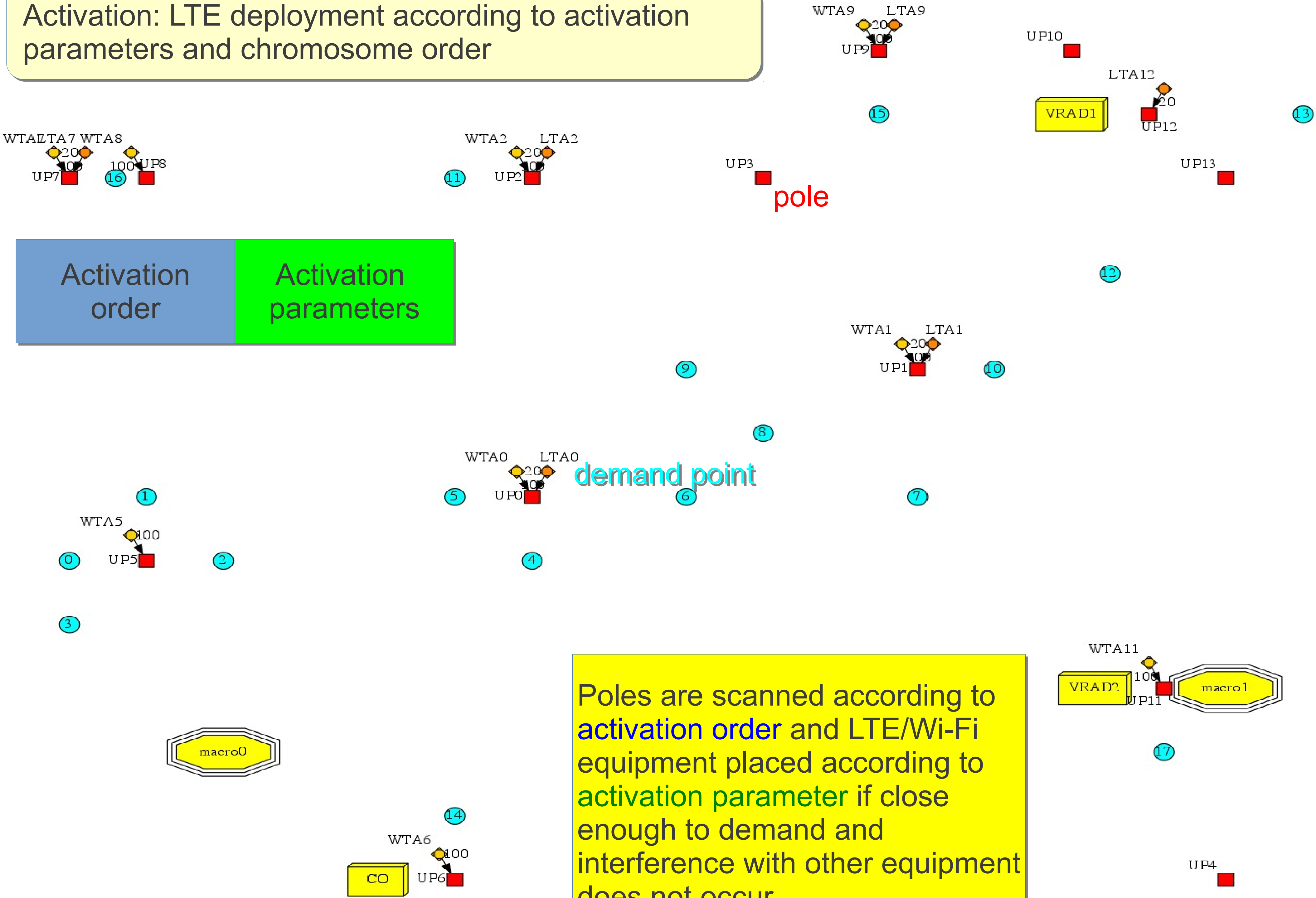
Activation: LTE deployment according to activation parameters and chromosome order



Activation parameter: Indicates whether pole can have LTE and if pole can be connected to a macrocell of FAP via fiber.

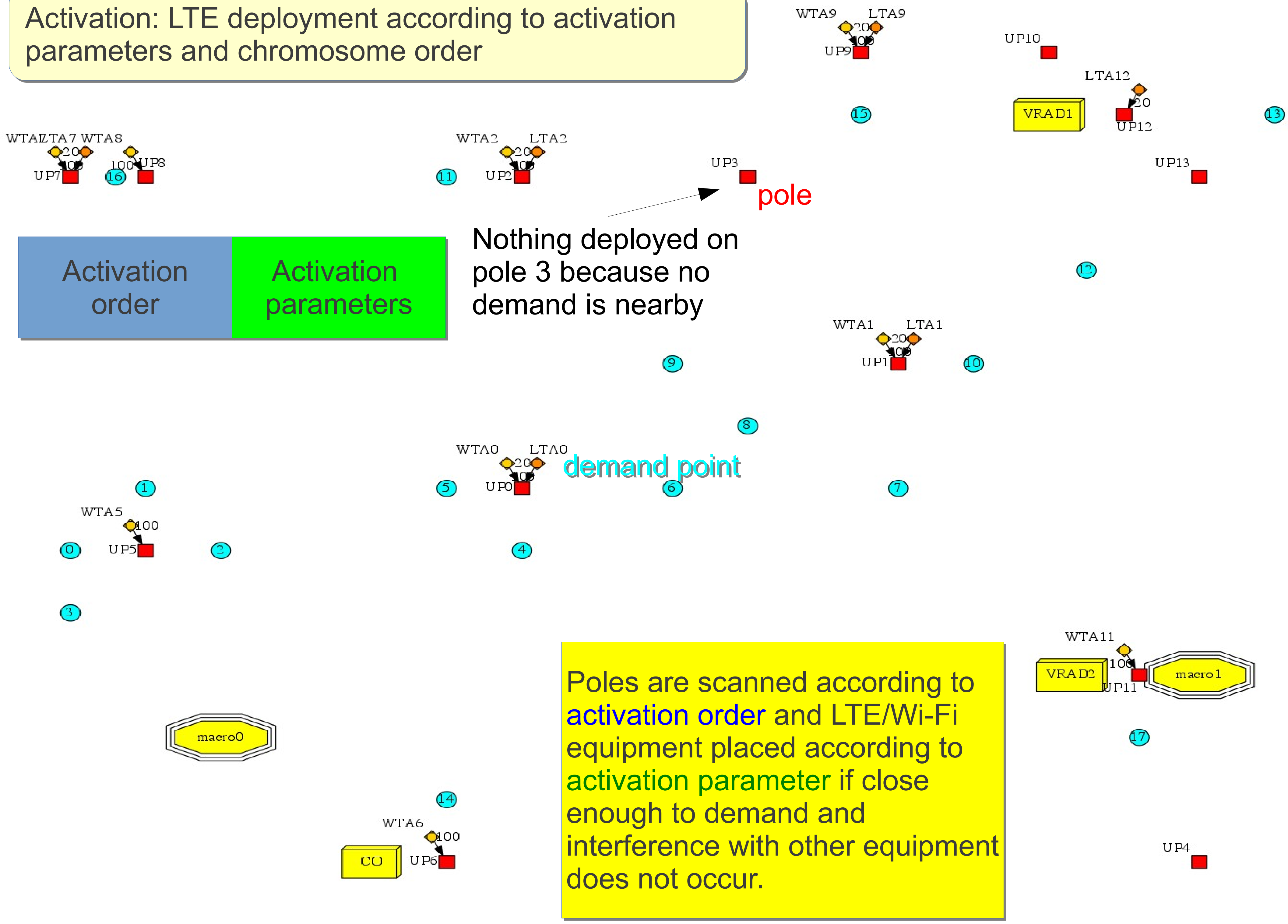
Activation: LTE deployment according to activation parameters and chromosome order

Activation order	Activation parameters

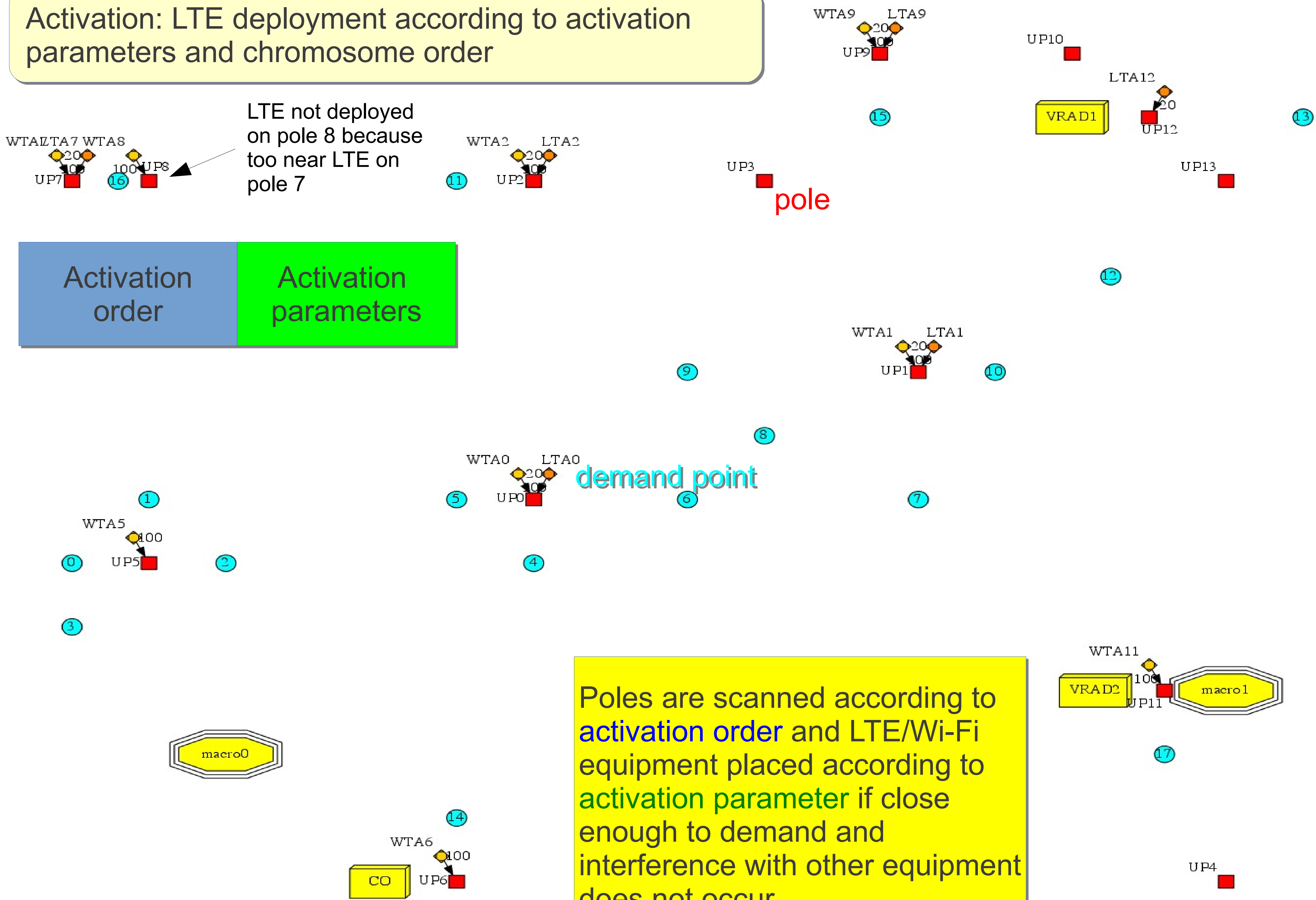


Poles are scanned according to activation order and LTE/Wi-Fi equipment placed according to activation parameter if close enough to demand and interference with other equipment does not occur.

Activation: LTE deployment according to activation parameters and chromosome order



Activation: LTE deployment according to activation parameters and chromosome order

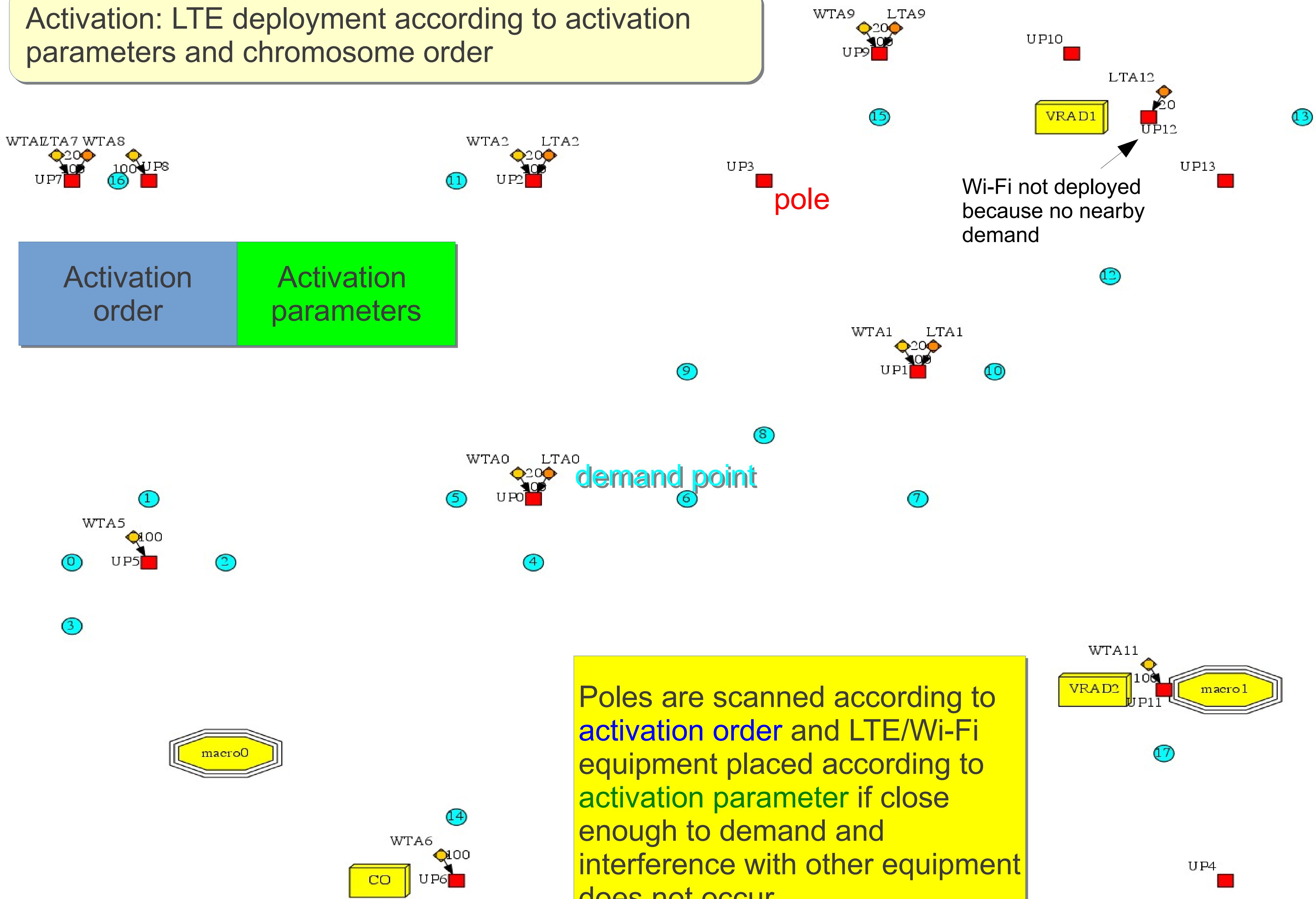


Activation order	Activation parameters

Poles are scanned according to activation order and LTE/Wi-Fi equipment placed according to activation parameter if close enough to demand and interference with other equipment does not occur.

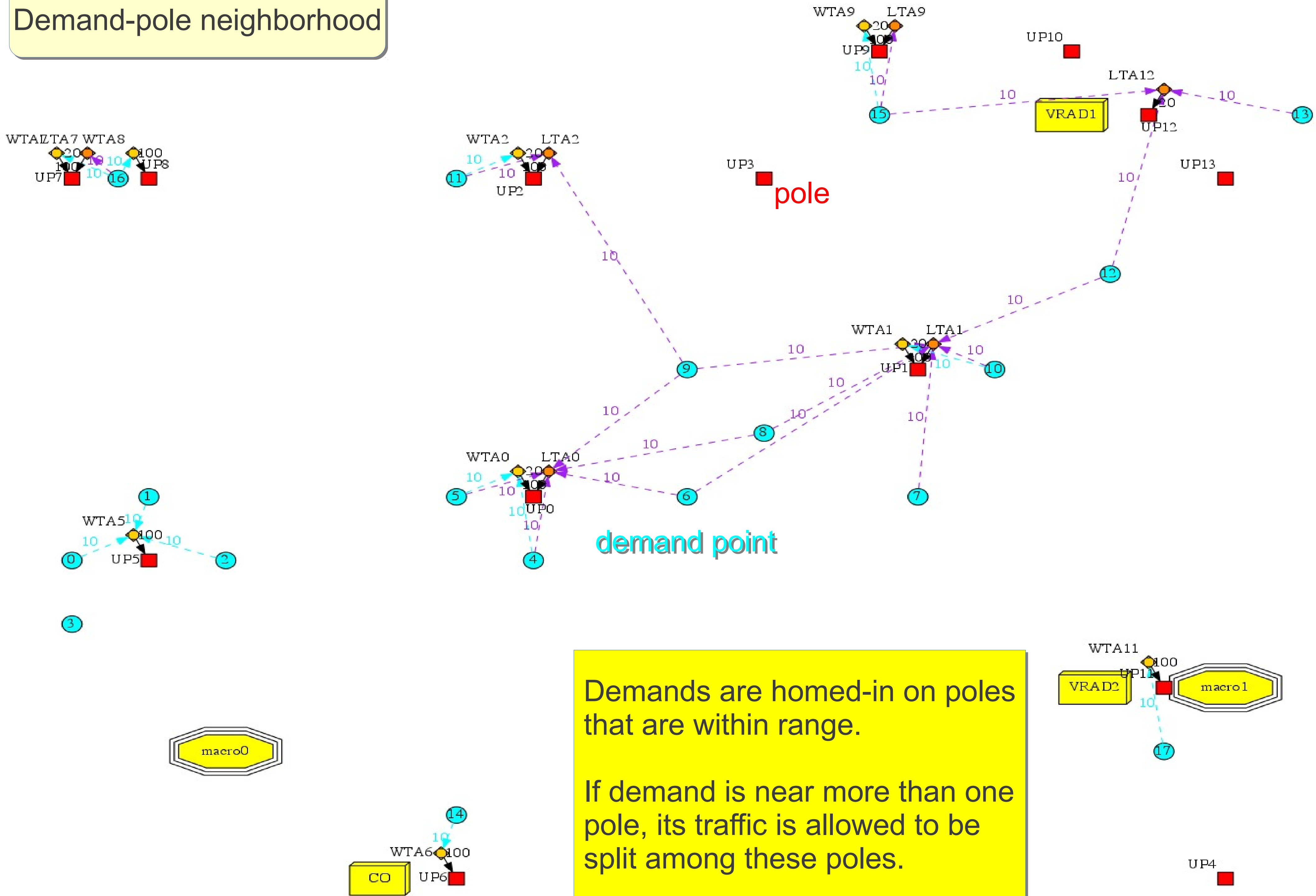
Activation: LTE deployment according to activation parameters and chromosome order

Activation order	Activation parameters



Poles are scanned according to activation order and LTE/Wi-Fi equipment placed according to activation parameter if close enough to demand and interference with other equipment does not occur.

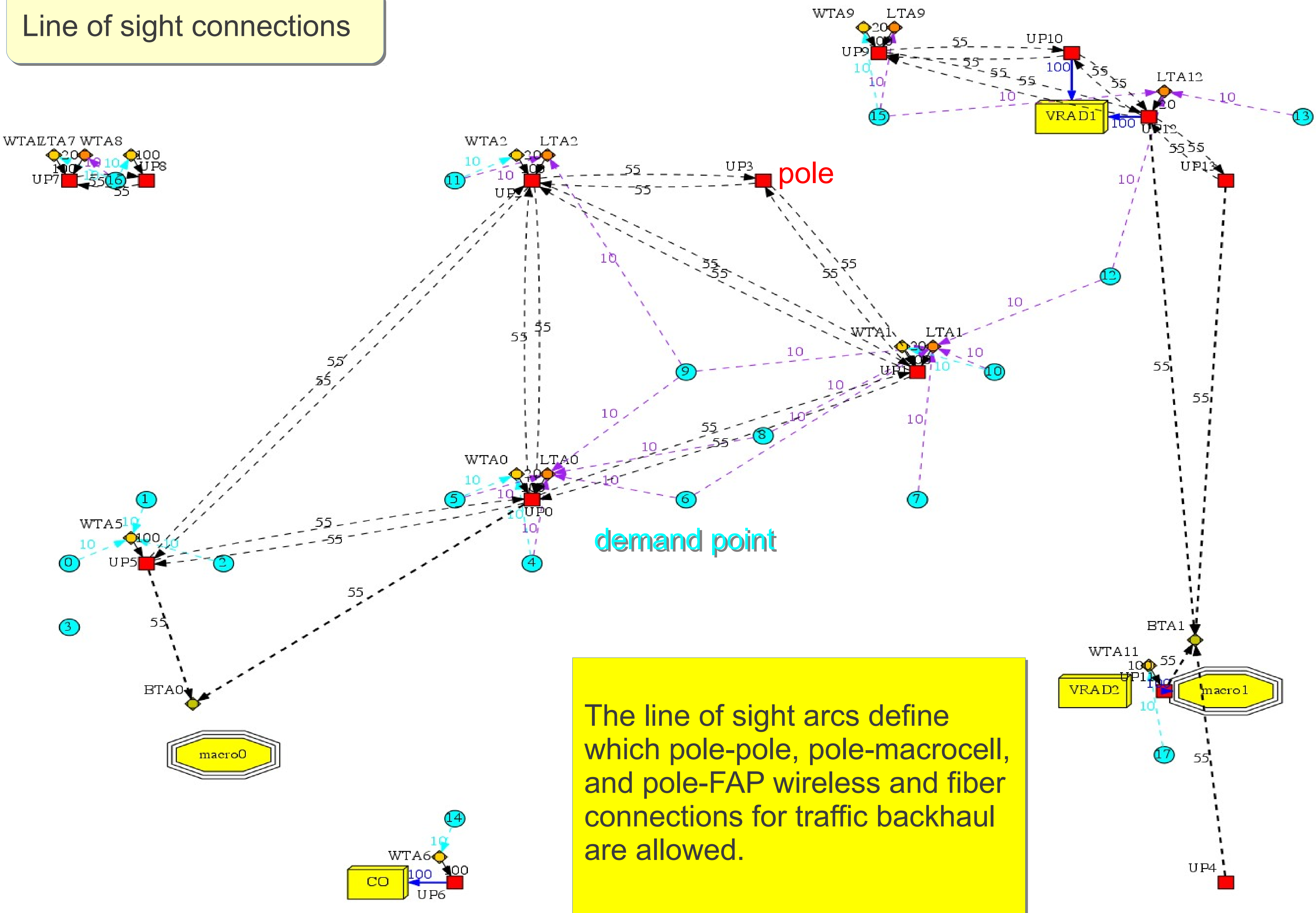
Demand-pole neighborhood



Demands are homed-in on poles that are within range.

If demand is near more than one pole, its traffic is allowed to be split among these poles.

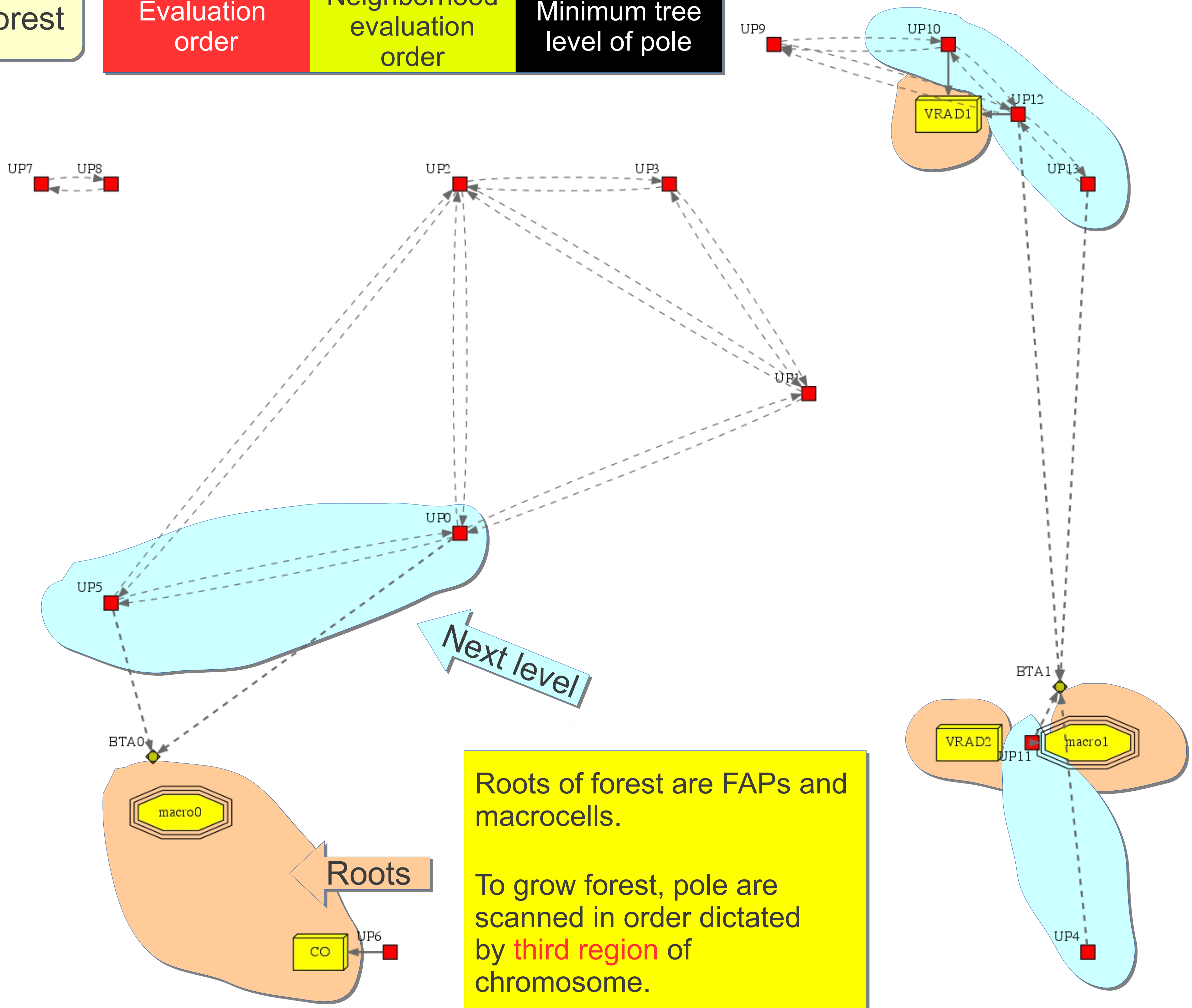
Line of sight connections



The line of sight arcs define which pole-pole, pole-macrocell, and pole-FAP wireless and fiber connections for traffic backhaul are allowed.

Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole

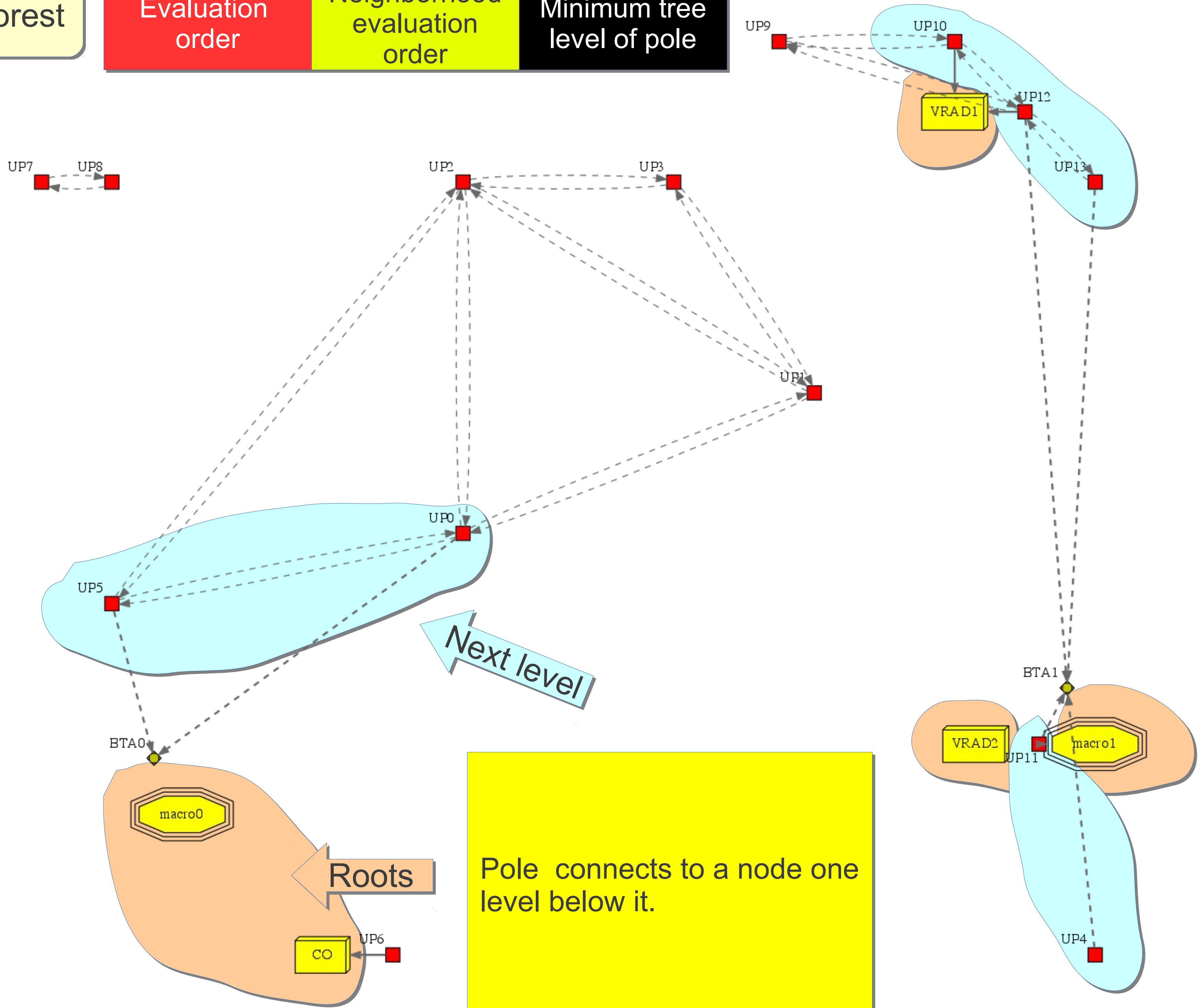


Roots of forest are FAPs and macrocells.

To grow forest, pole are scanned in order dictated by third region of chromosome.

Growing forest

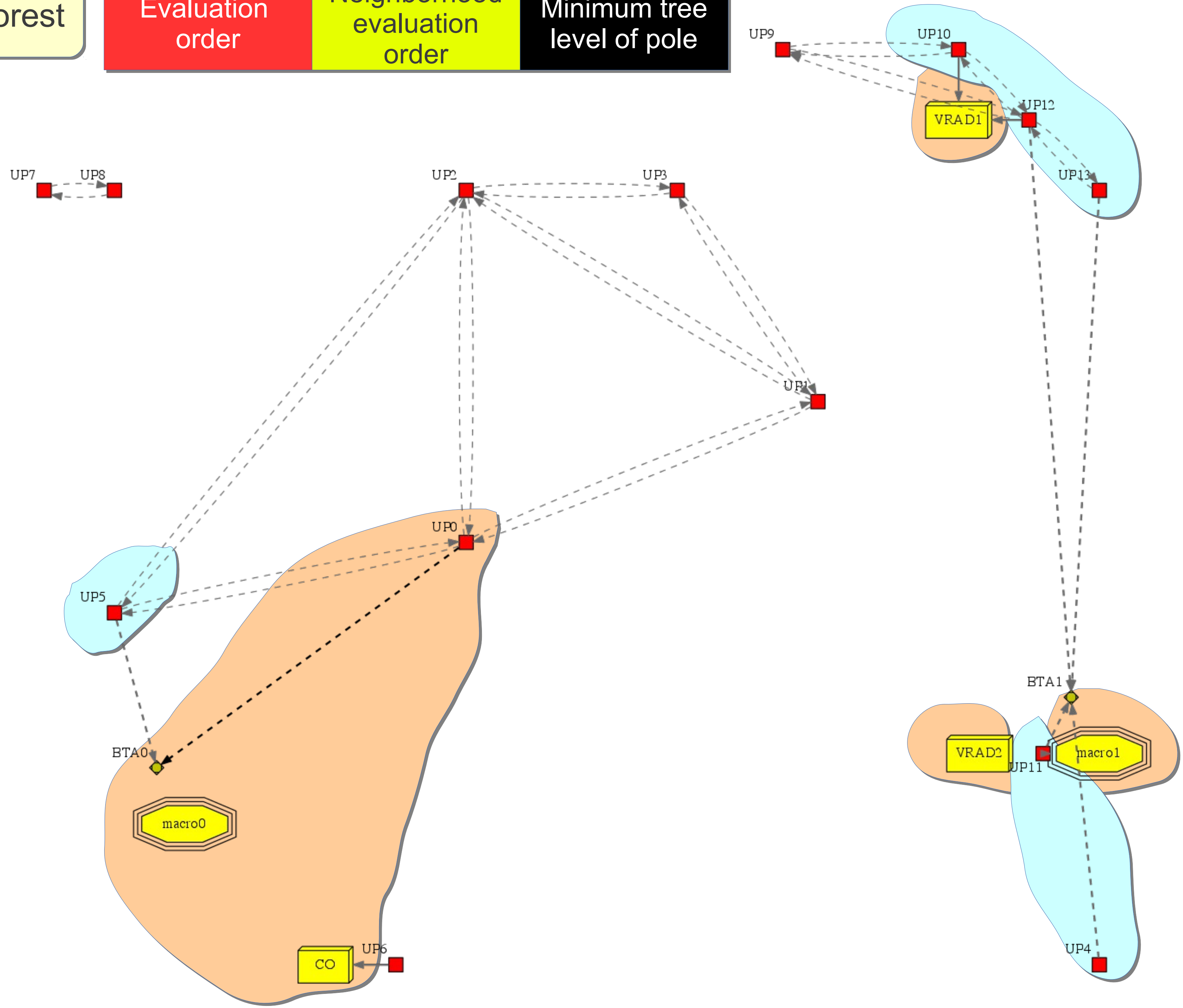
Evaluation order	Neighborhood evaluation order	Minimum tree level of pole



Pole connects to a node one level below it.

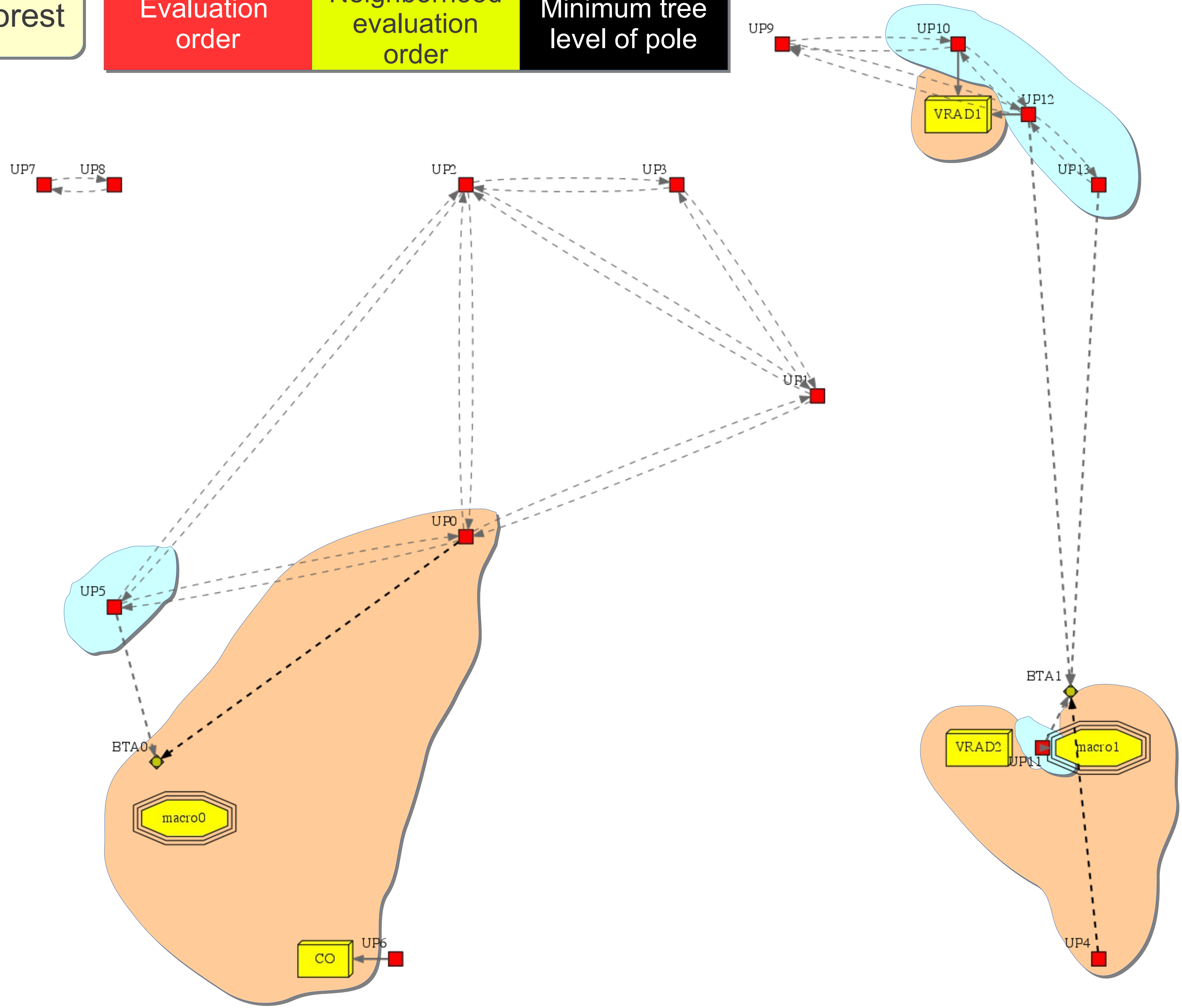
Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole



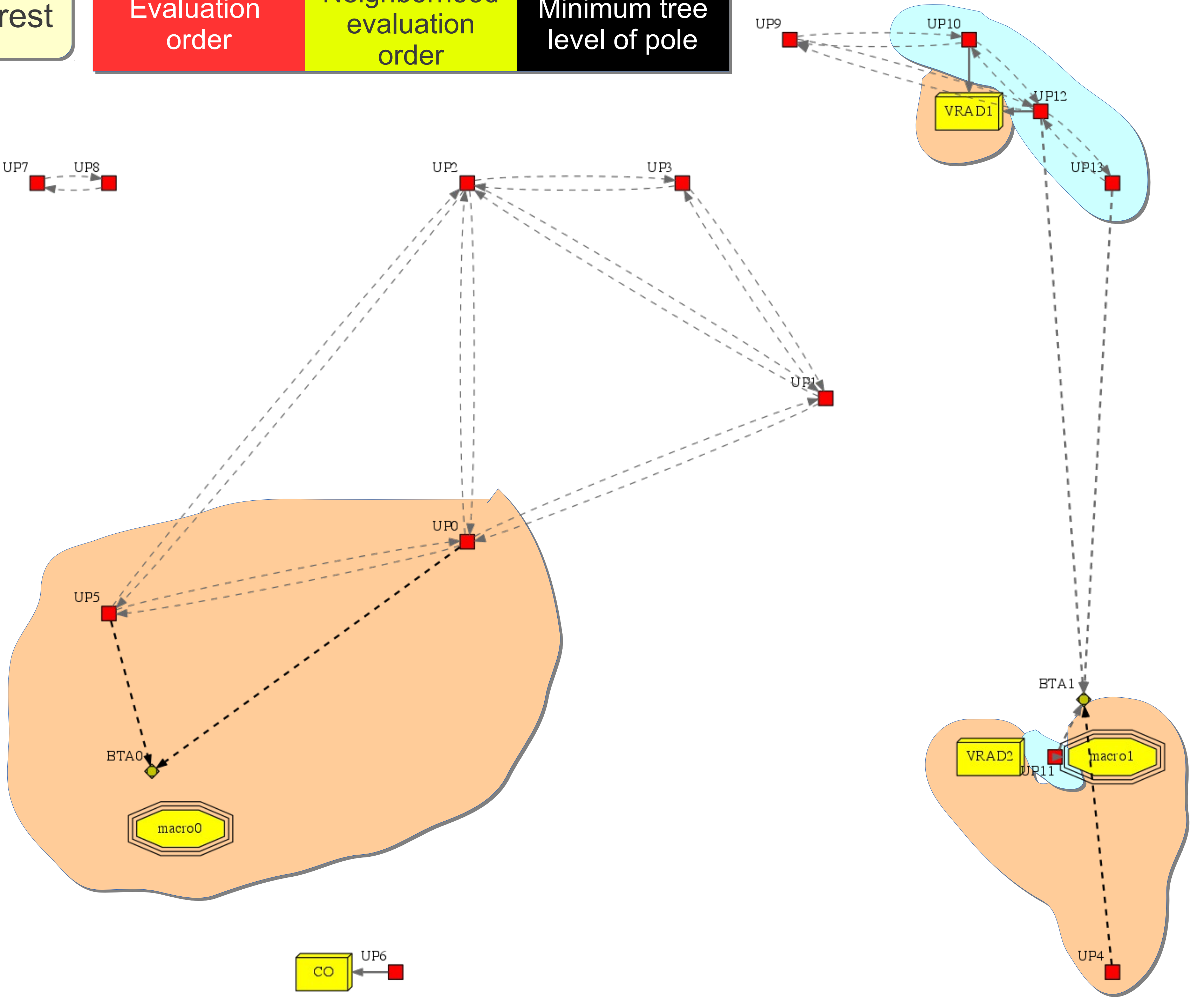
Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole



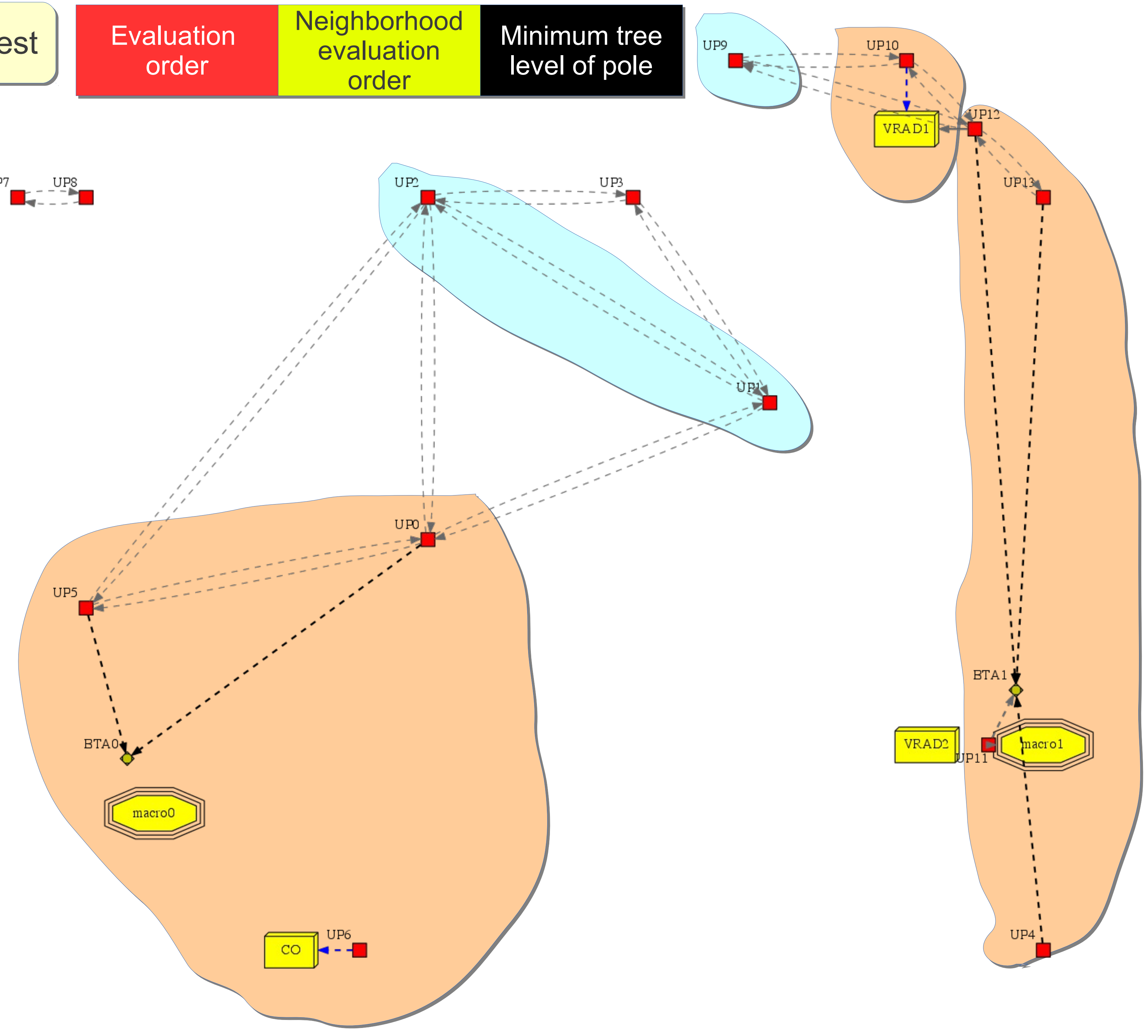
Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole



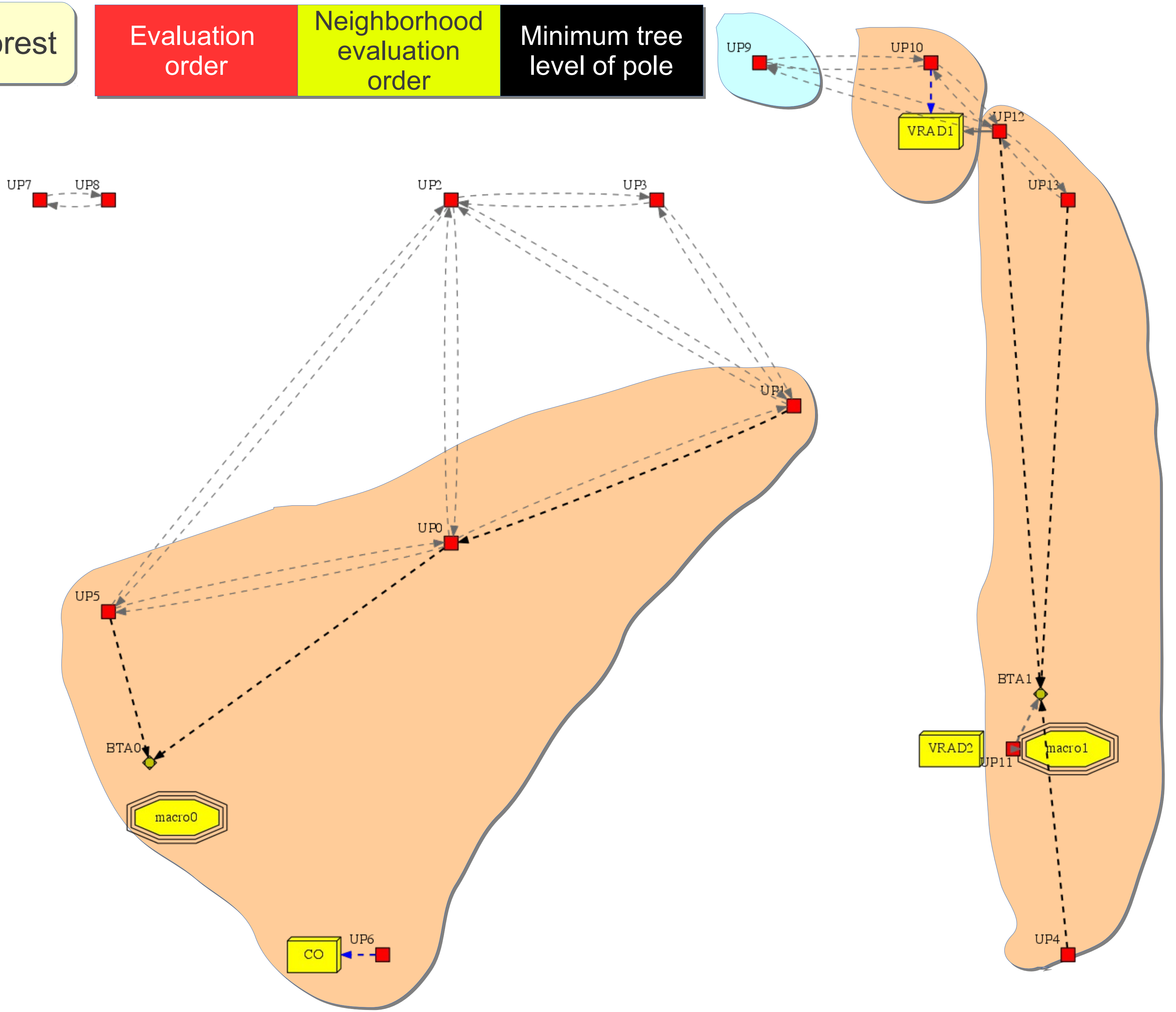
Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole

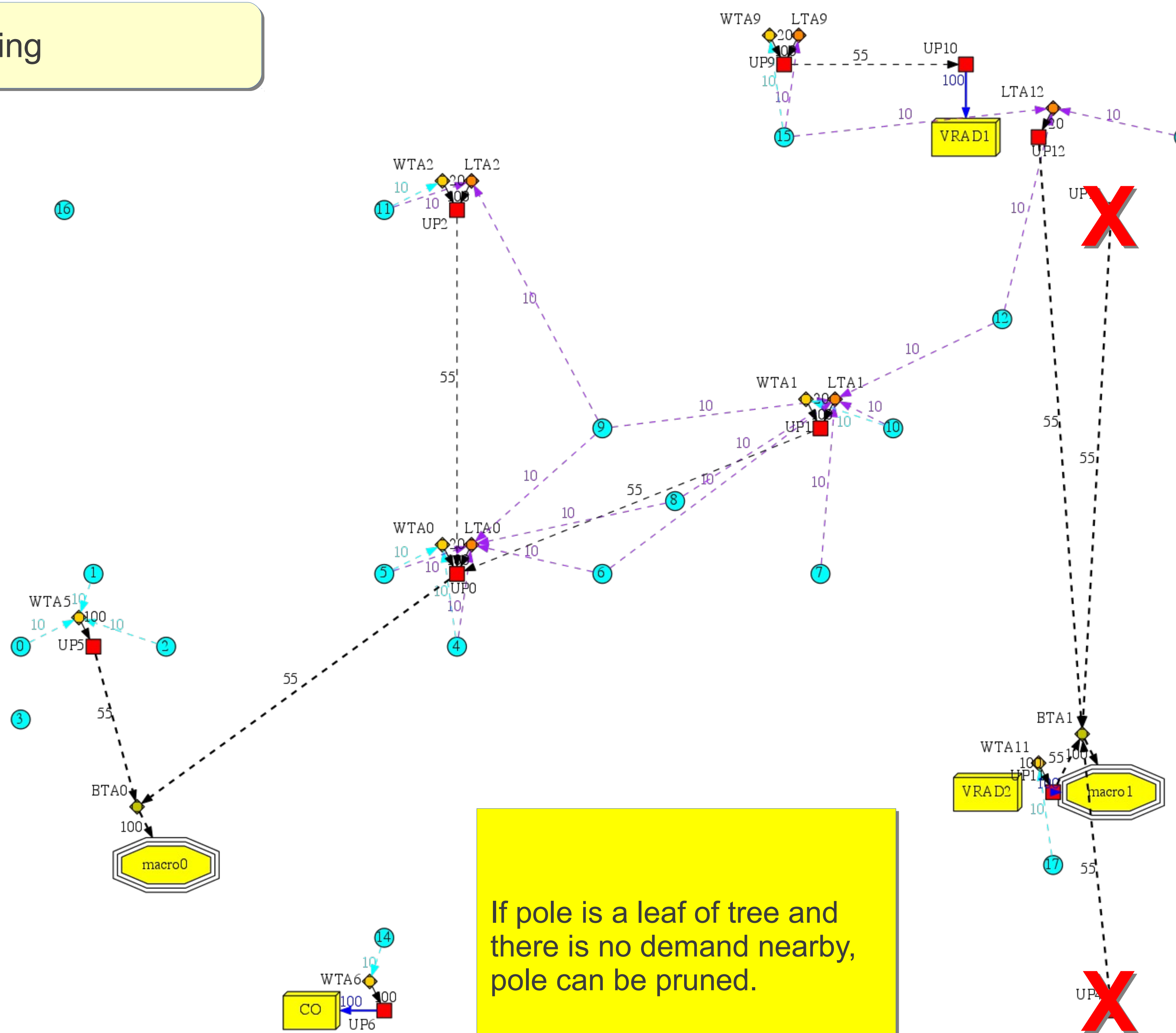


Growing forest

Evaluation order	Neighborhood evaluation order	Minimum tree level of pole

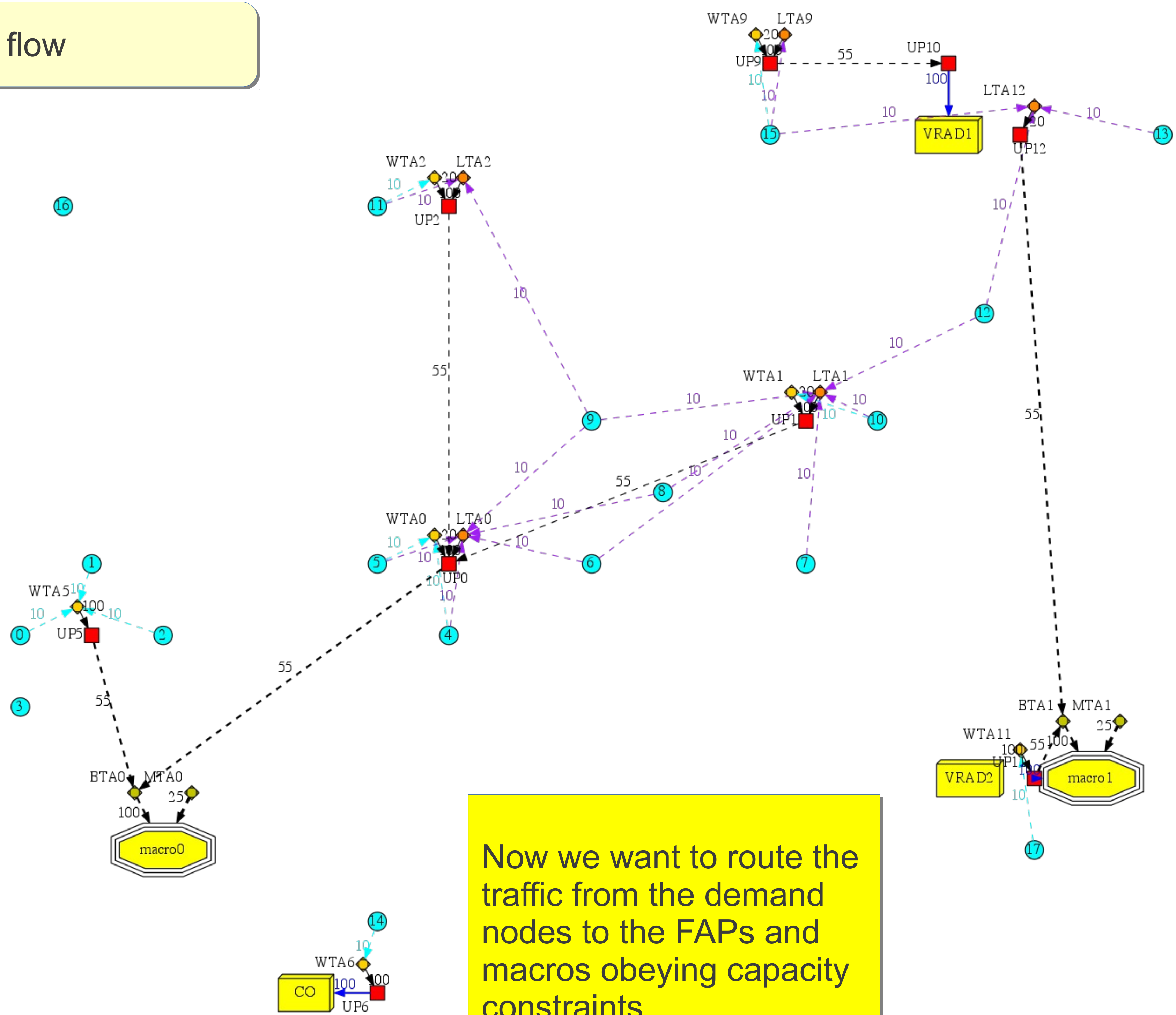


First Pruning



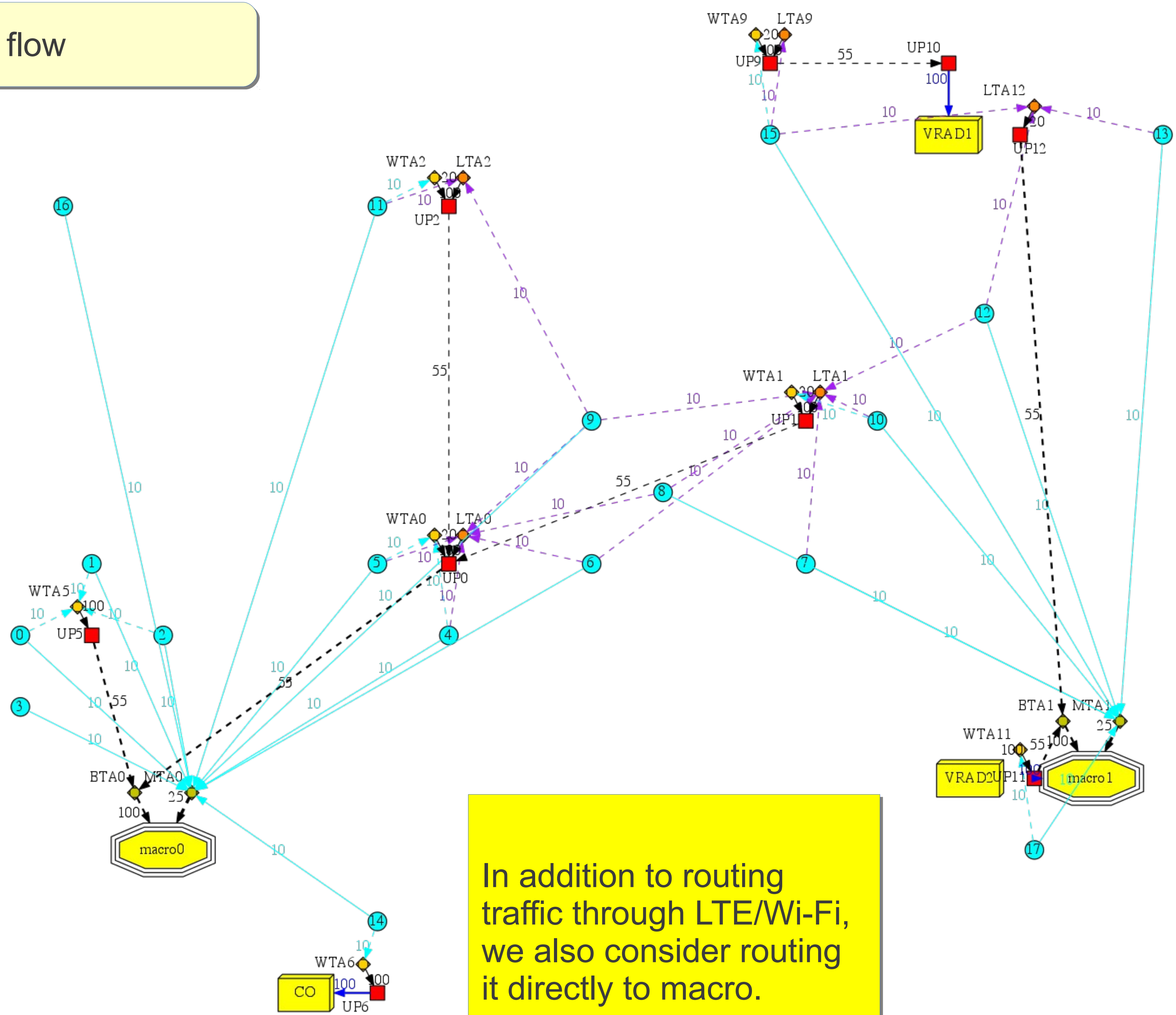
If pole is a leaf of tree and there is no demand nearby, pole can be pruned.

Maximum flow



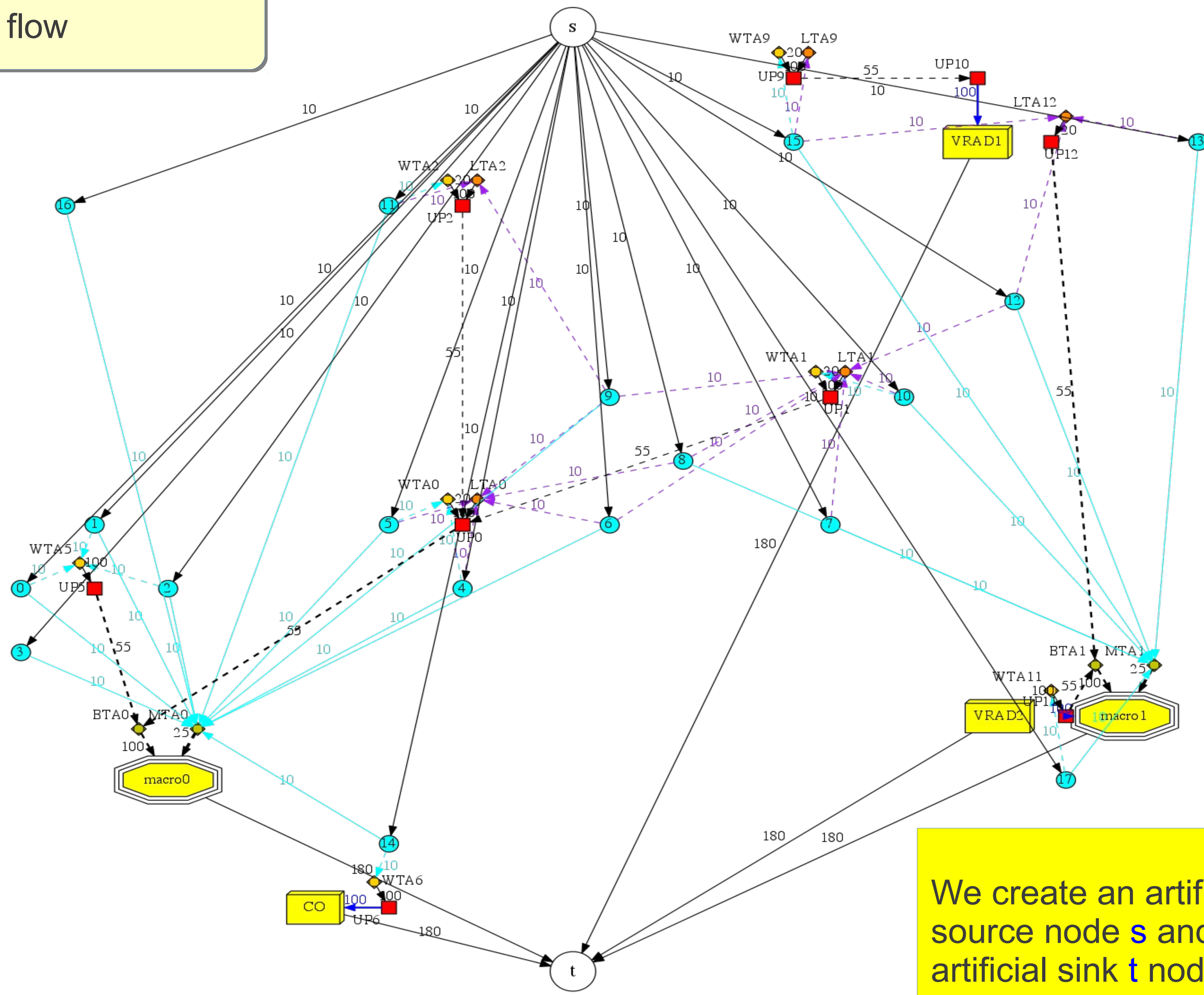
Now we want to route the traffic from the demand nodes to the FAPs and macros obeying capacity constraints.

Maximum flow



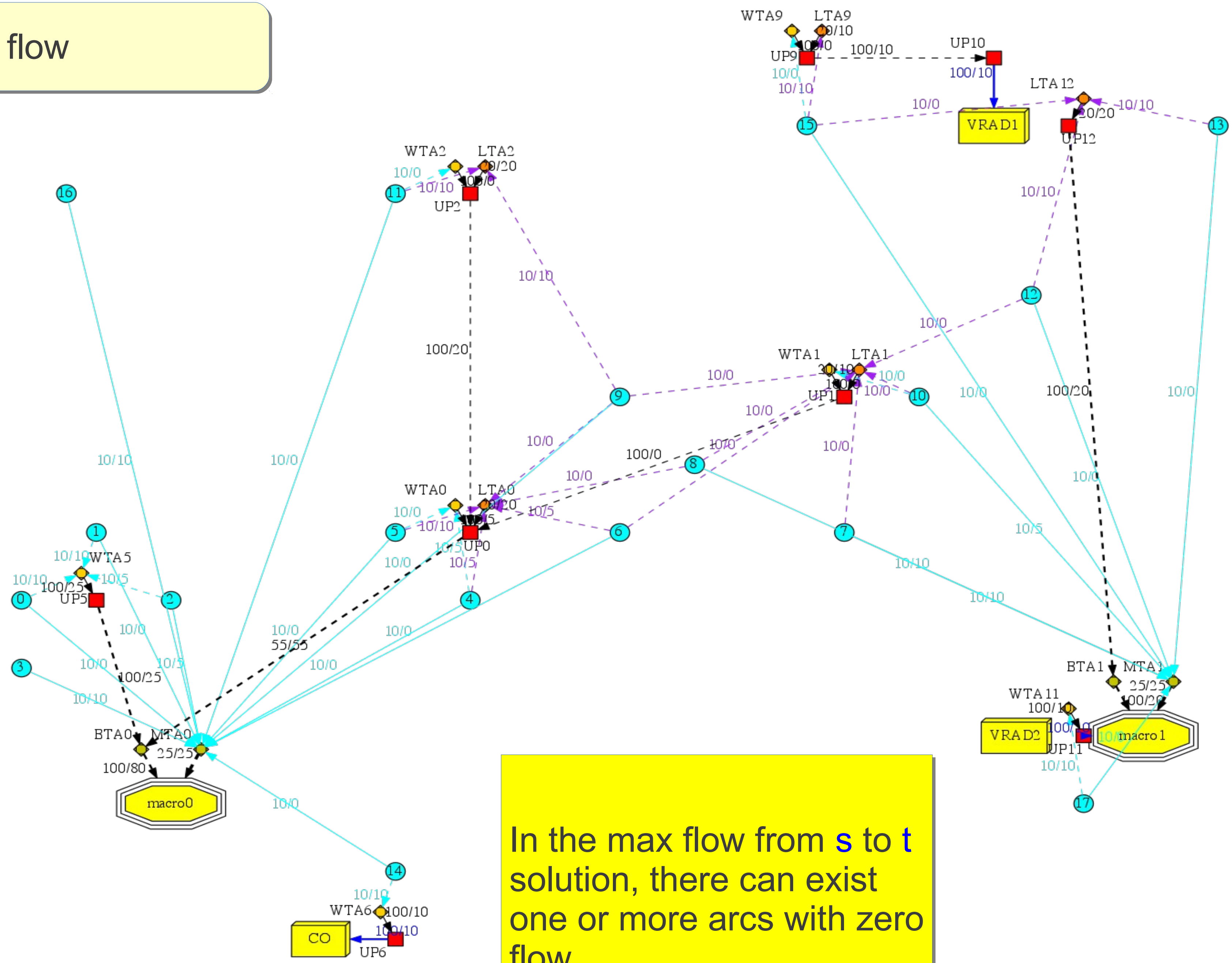
In addition to routing traffic through LTE/Wi-Fi, we also consider routing it directly to macro.

Maximum flow



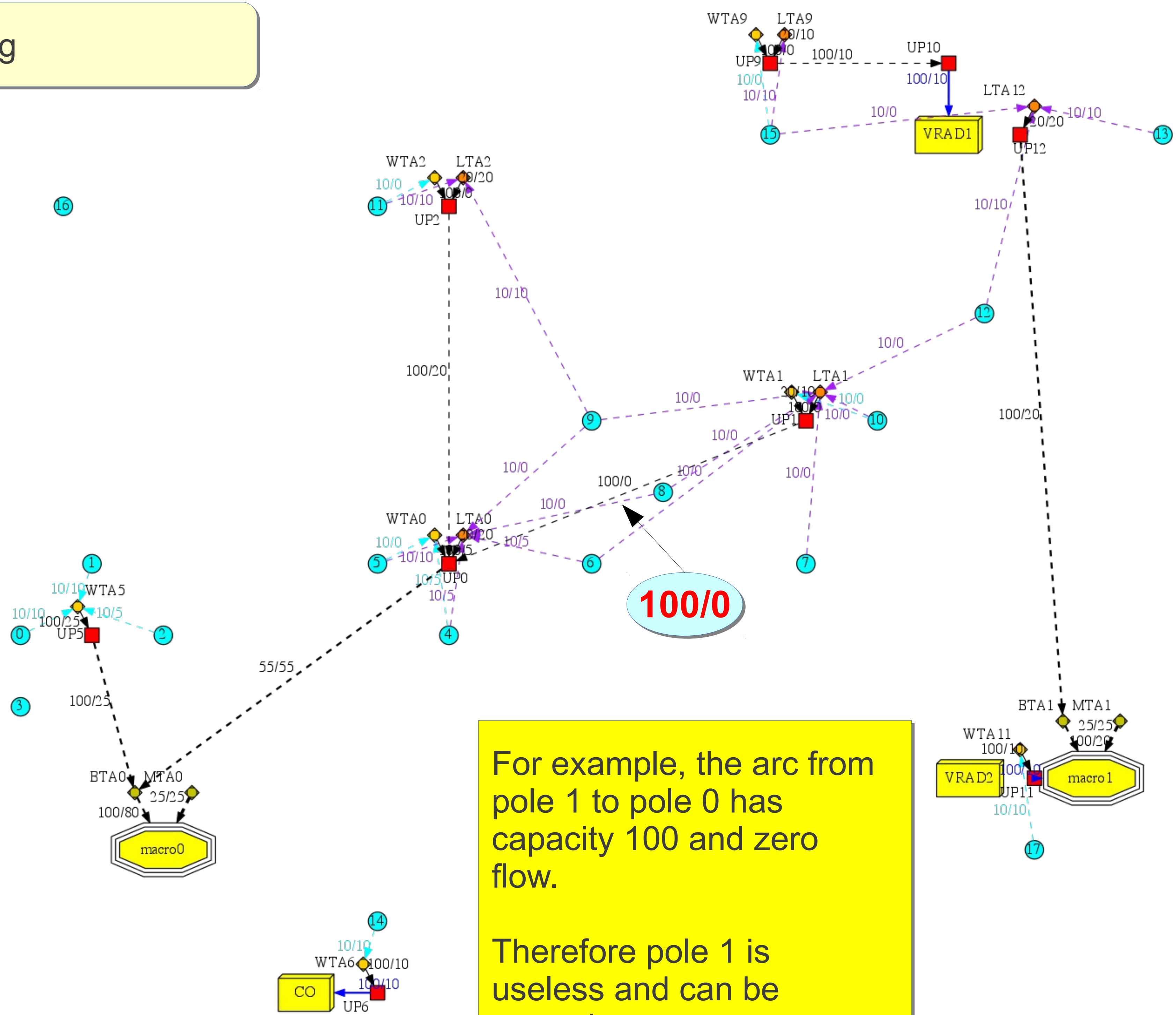
We create an artificial source node s and an artificial sink t node and compute the max flow from s to t.

Maximum flow



In the max flow from s to t solution, there can exist one or more arcs with zero flow.

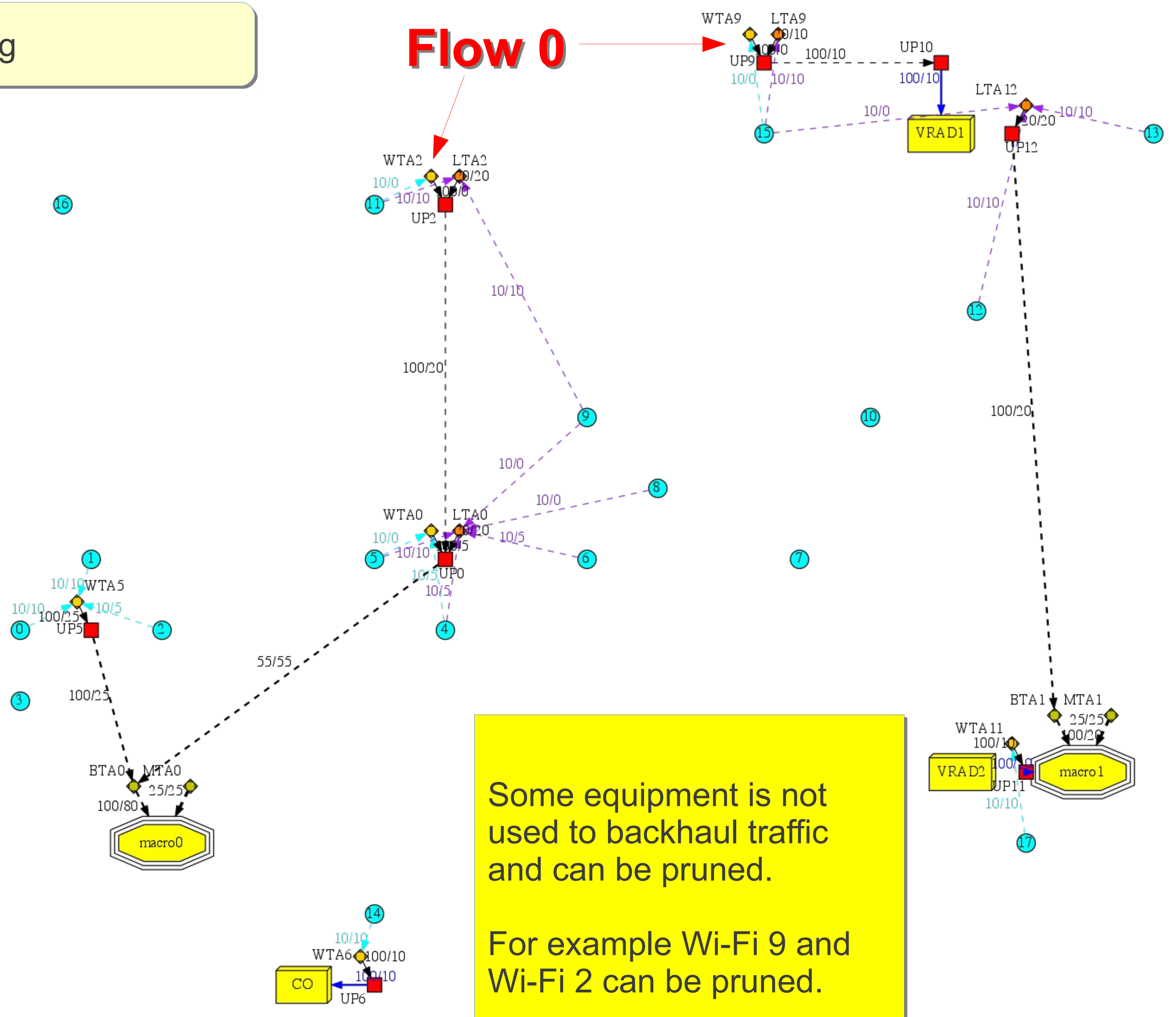
2nd Pruning



For example, the arc from pole 1 to pole 0 has capacity 100 and zero flow.

Therefore pole 1 is useless and can be pruned.

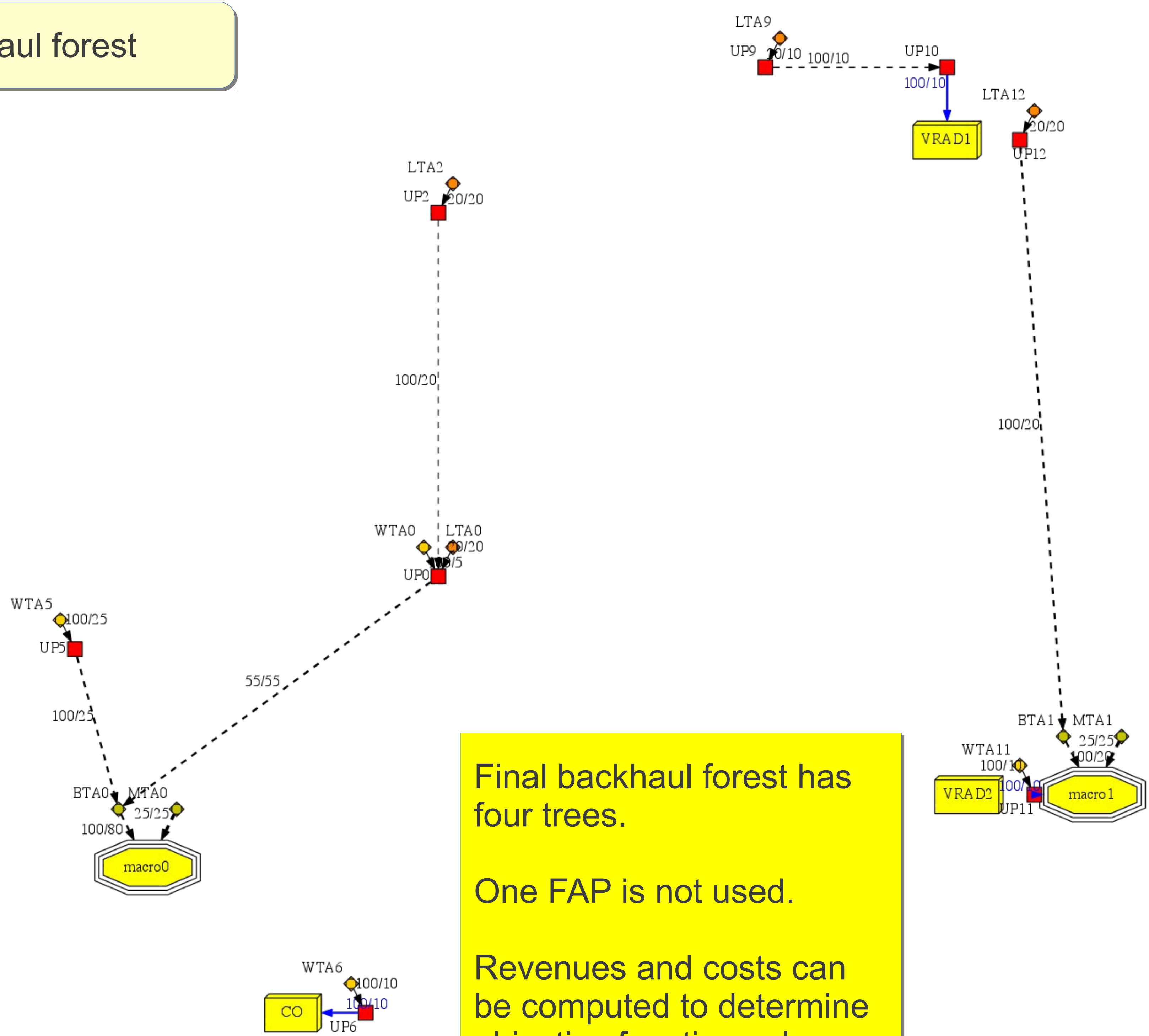
2nd Pruning



Some equipment is not used to backhaul traffic and can be pruned.

For example Wi-Fi 9 and Wi-Fi 2 can be pruned.

Final backhaul forest



Final backhaul forest has four trees.

One FAP is not used.

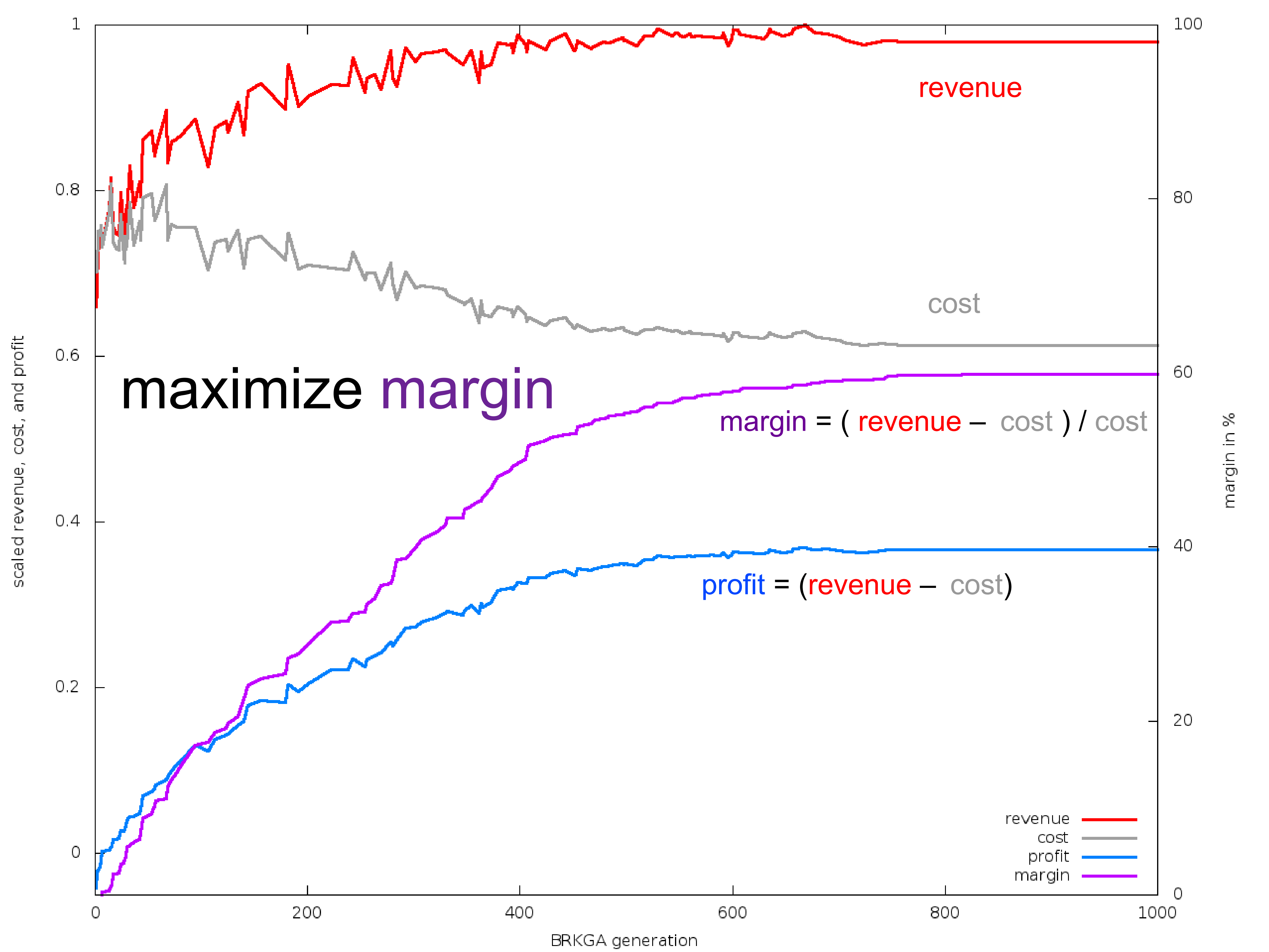
Revenues and costs can be computed to determine objective function value.

An example optimization

We ran the BRKGA on a "real" instance from a large Tier 1 Internet Service Provider

- Number of utility poles: 4395
- Number of demand points: 3236
- Number of fibered access points (FAP) excluding macrocells: 482
- Number of macrocells: 31

• Objective function: maximize margin = (revenue - cost)/cost



maximize margin
revenue
cost
margin = (revenue – cost) / cost
profit = (revenue – cost)
scaled revenue, cost, and profit
margin in %
BRKGA generation
0
200
400
600
800
1000
1
0.8
0.6
0.4
0.2
0
100
80
60
40
20
0
revenue
cost
profit
margin

Concluding remarks

- We described the prize collecting directed k-hop Steiner forest (PCk-HSF) problem
- We modeled a wireless backhaul network planning problem as a PCk-HSF problem with additional constraints
- We described a biased random-key genetic algorithm (BRKGA) for the wireless backhaul network planning problem focusing on the decoder
- We applied the BRKGA to a “real” instance of the wireless backhaul network planning problem

The End

These slides are available at http://mauricioresende.com