Randomized heuristics for handover minimization

Mauricio G. C. Resende
AT&T Research

Joint work with Luis Morán-Mirabal, José Luis González-Velarde and Ricardo M. A. Silva

Talk given at Instituto de Informática
U. Federal do Rio Grande do Sul
Porto Alegre, RS - Brazil
July 6, 2012

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

Outline

- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

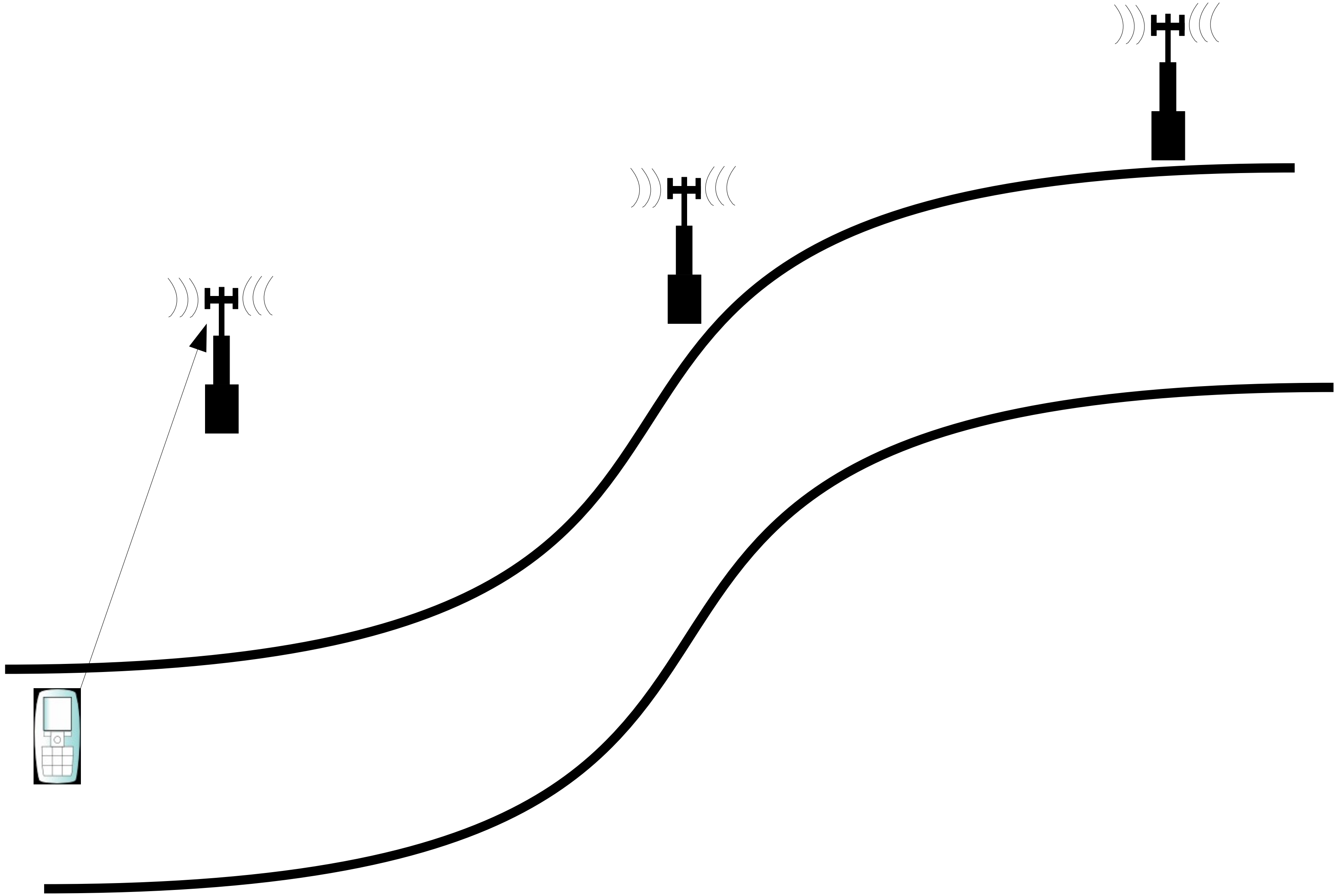
Outline

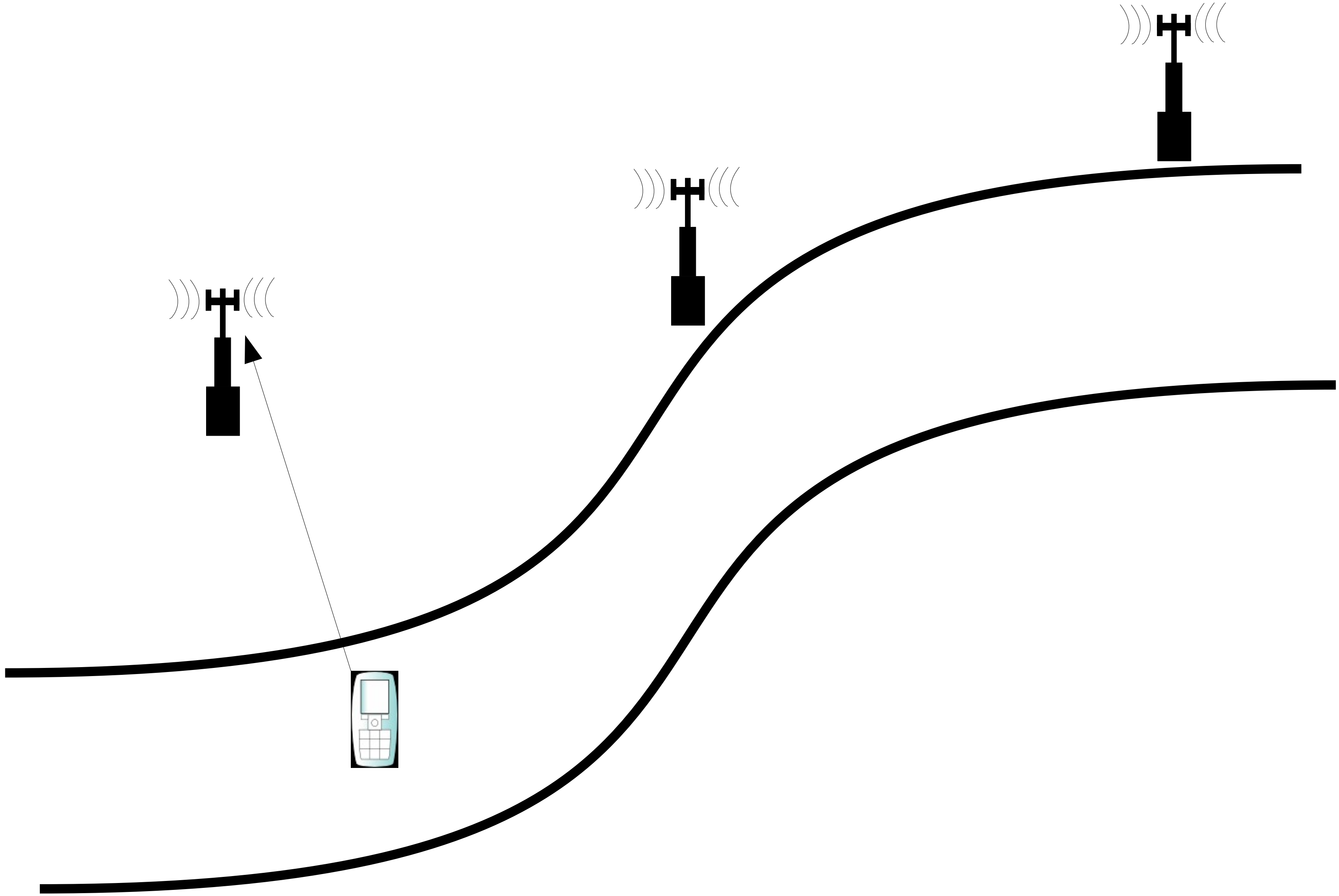
- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP with synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

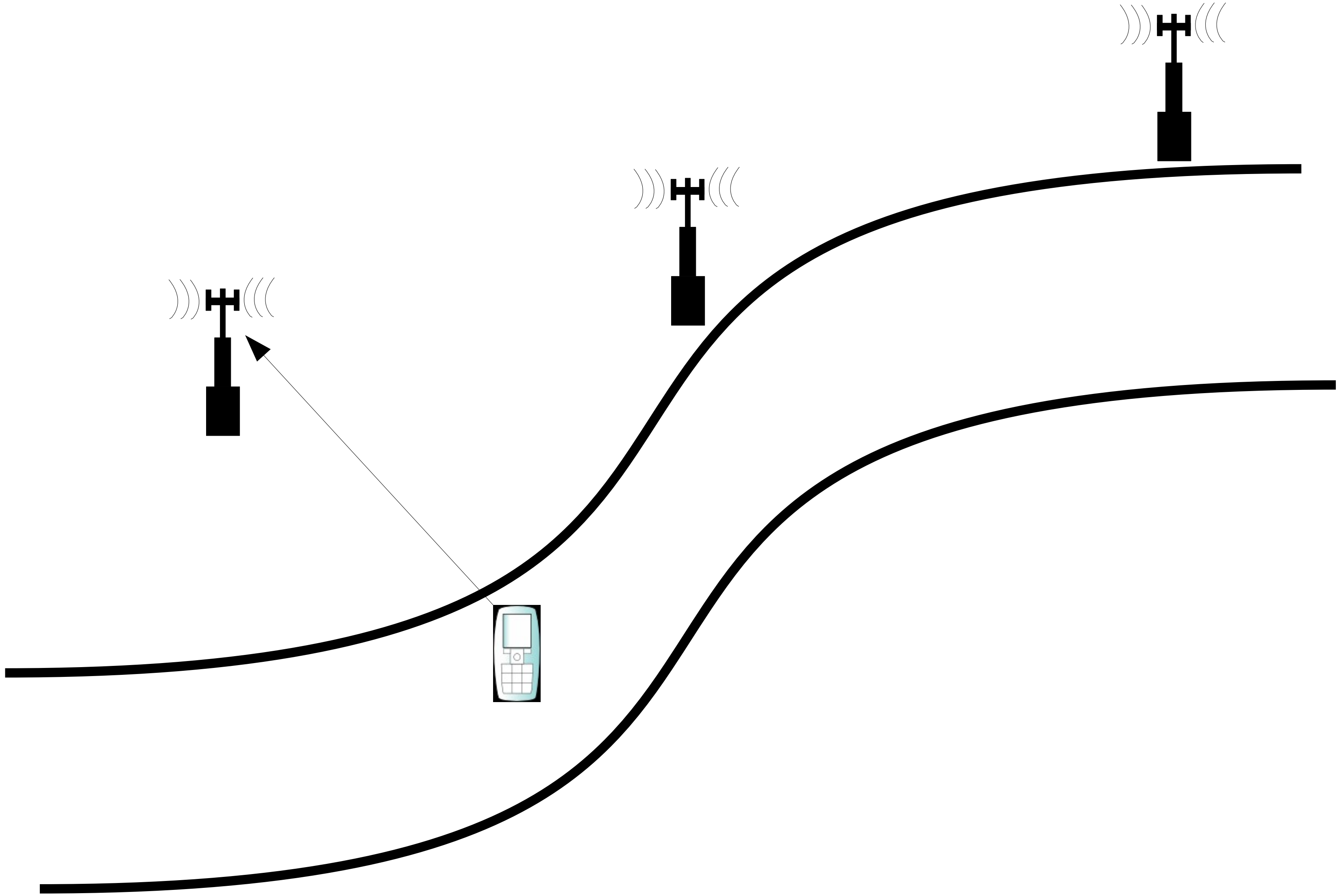
Outline

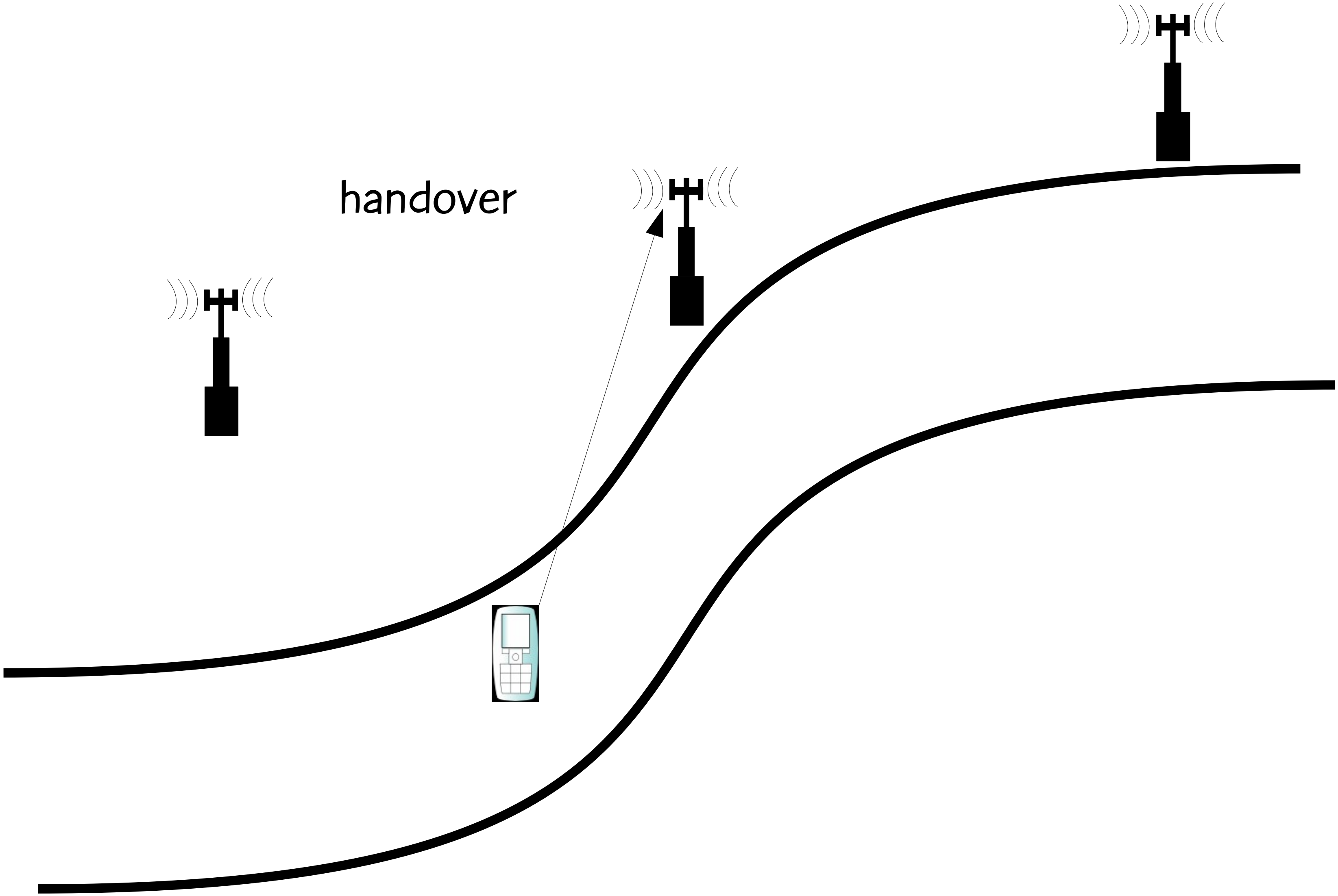
- Handover minimization problem (HMP)
- Generalized quadratic assignment problem (GQAP)
- GRASP with path-relinking for the GQAP
- HMP is a special case of GQAP
- Experiments with GRASP for GQAP on HMP on synthetic networks
- GRASP with evolutionary path-relinking for HMP with experiments
- Biased random-key genetic algorithm for HMP with experiments
- Concluding remarks

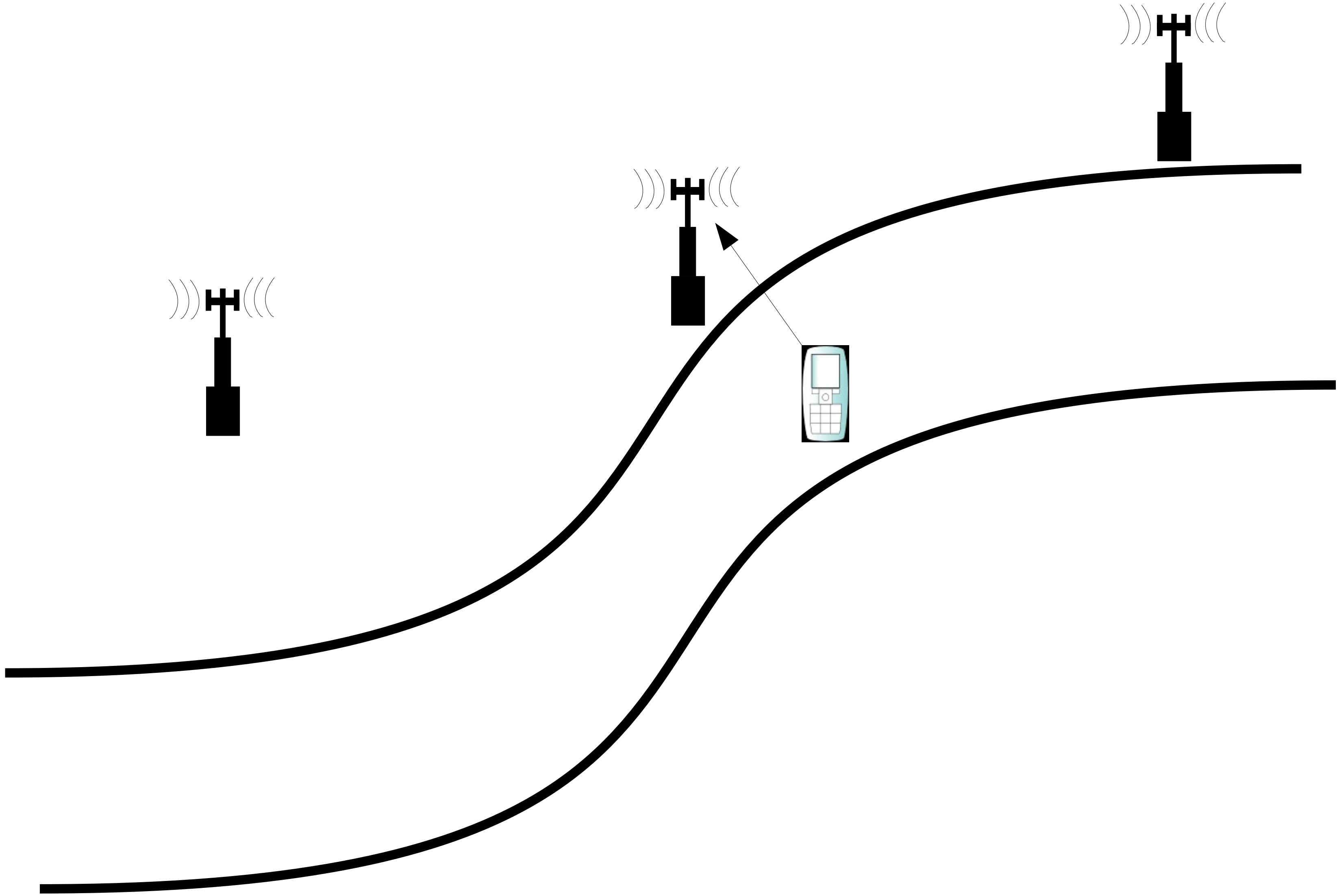
Handover minimization

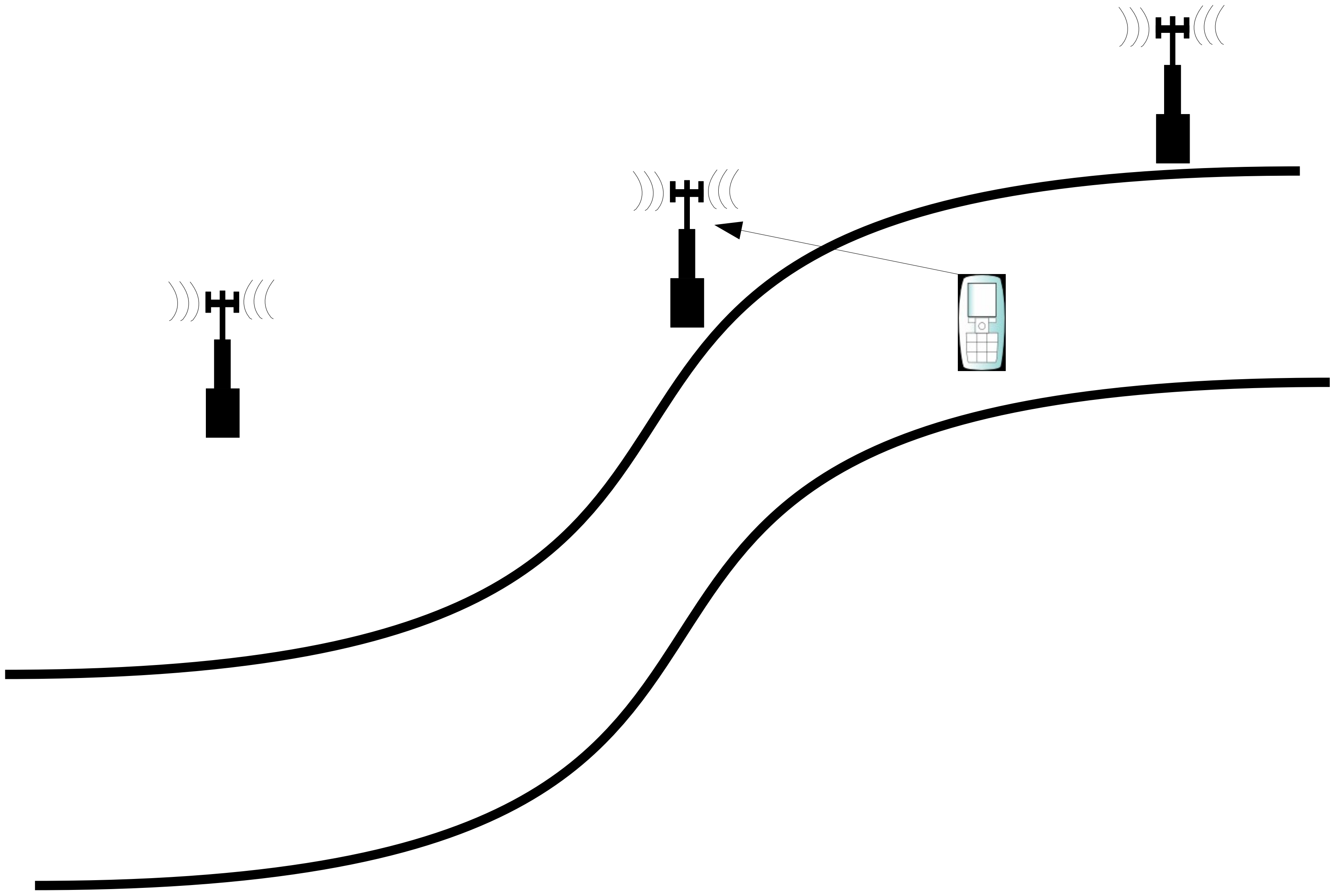


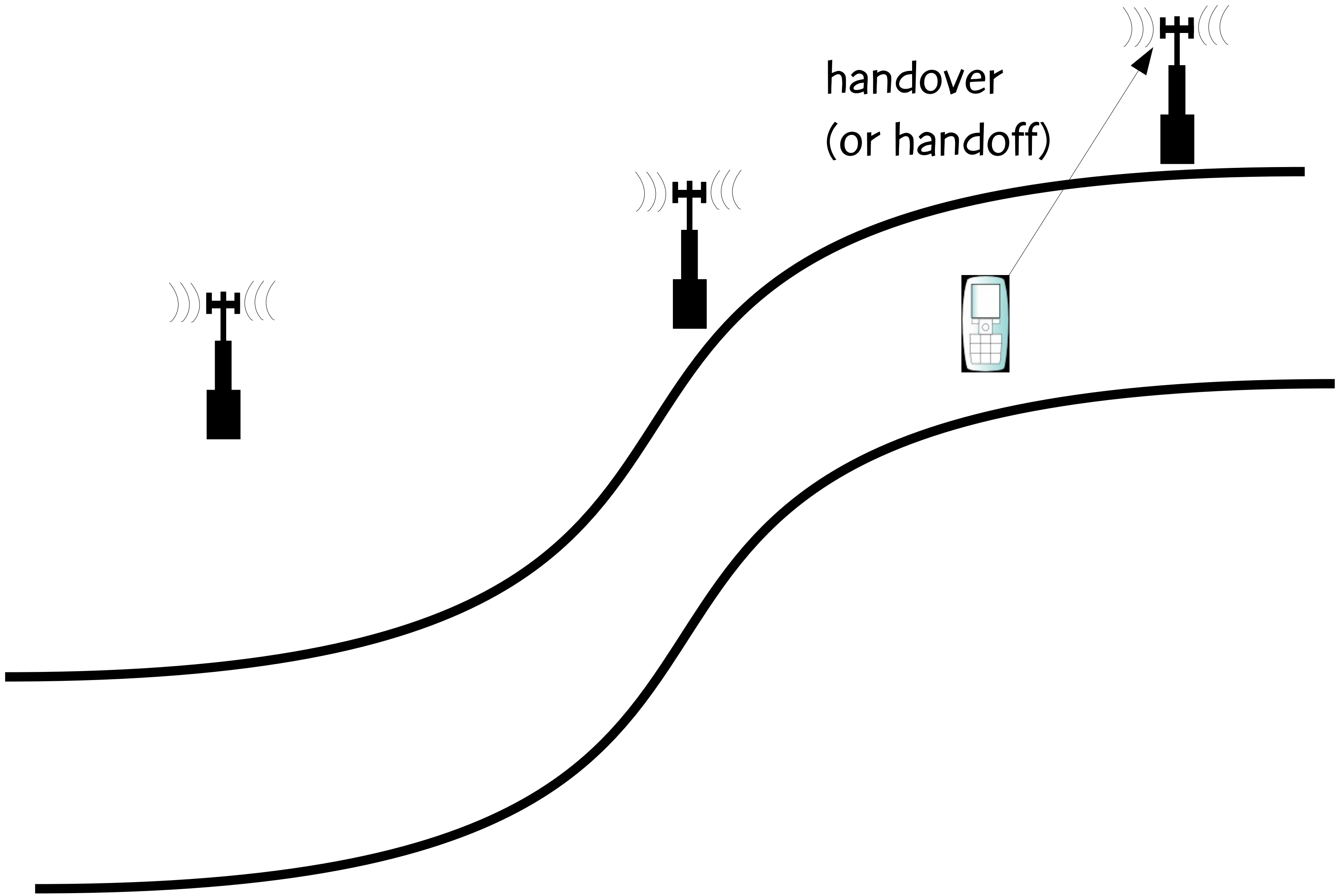


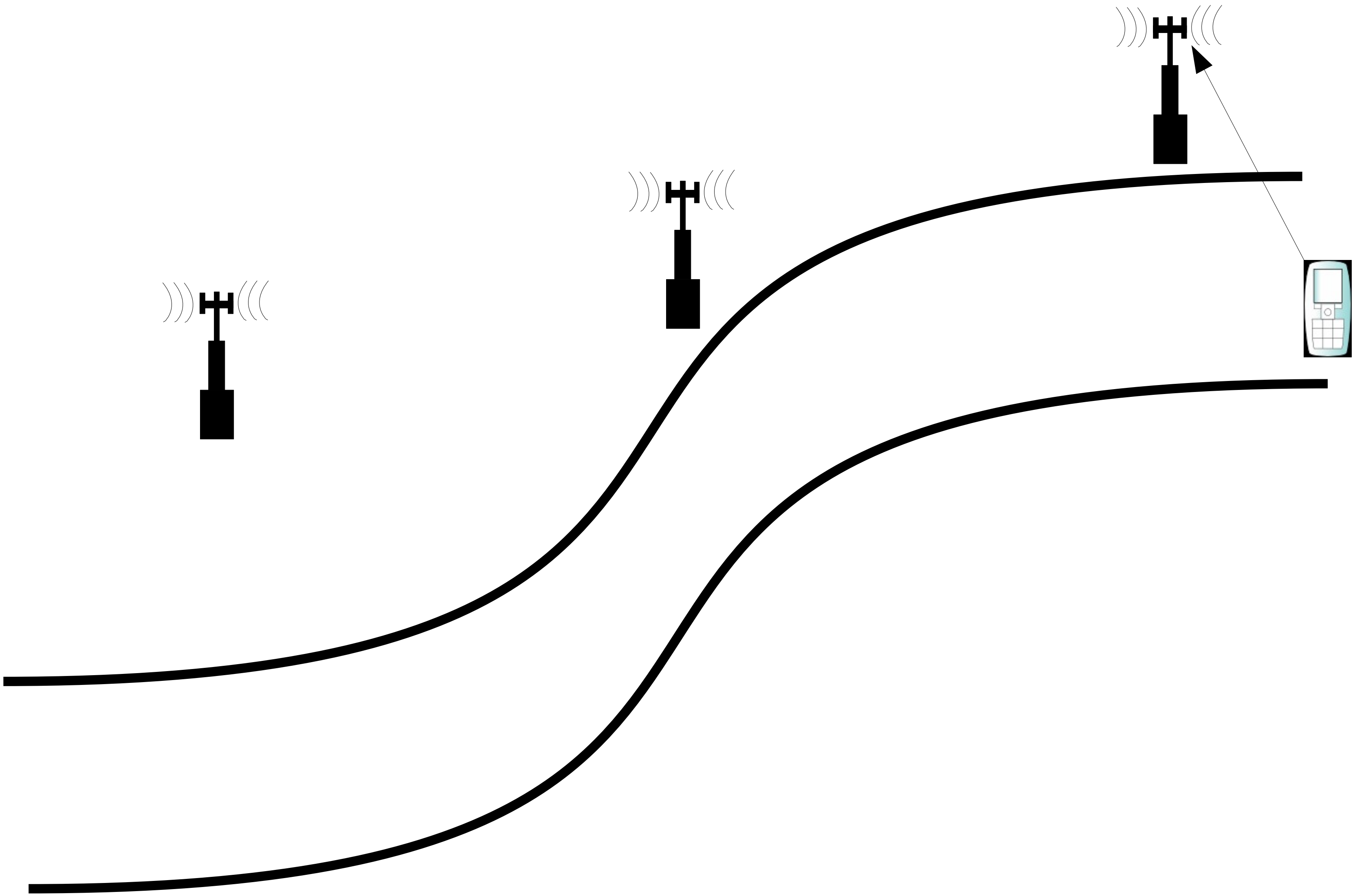


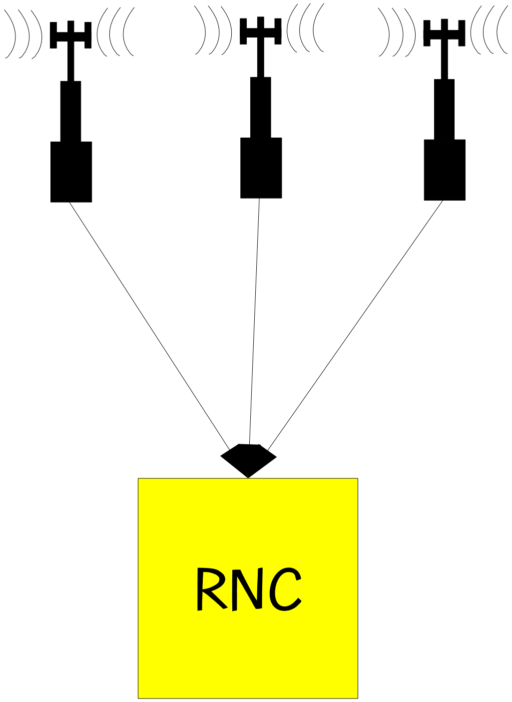

handover



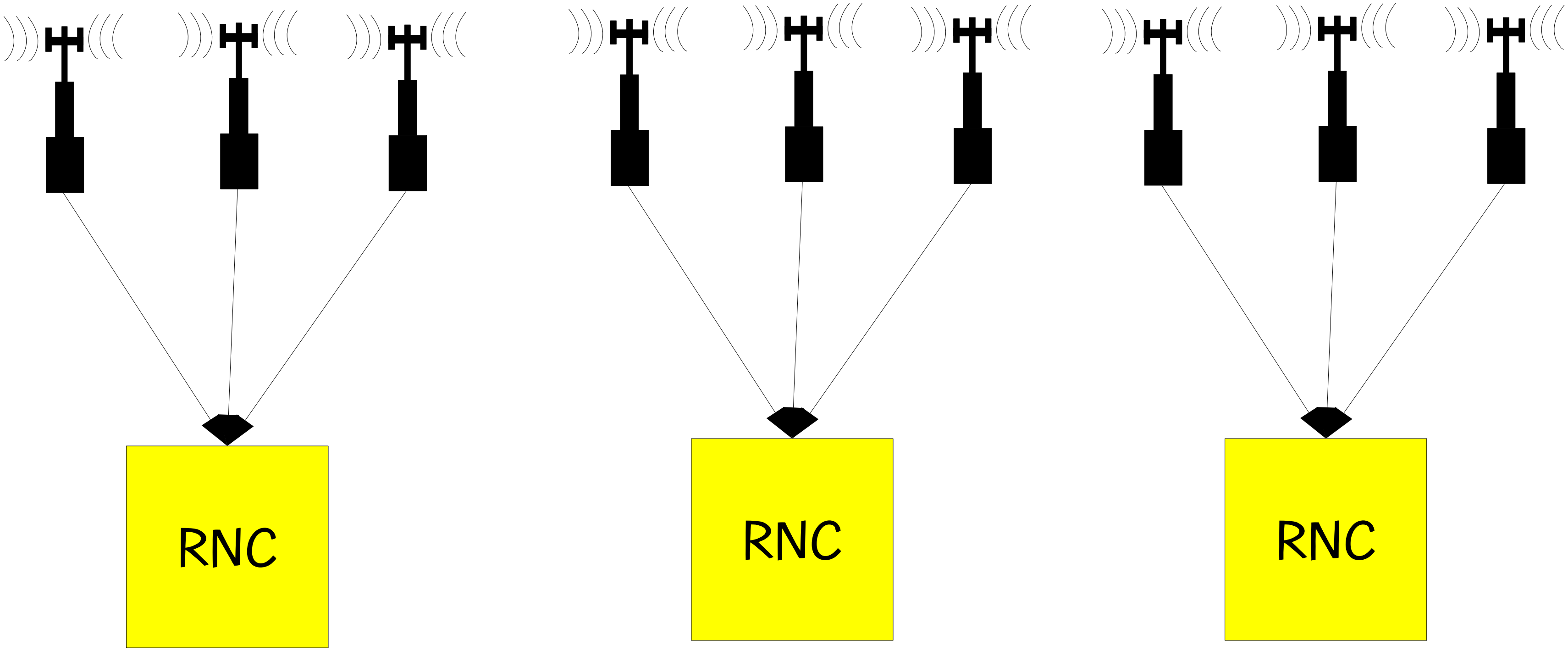



handover
(or handoff)

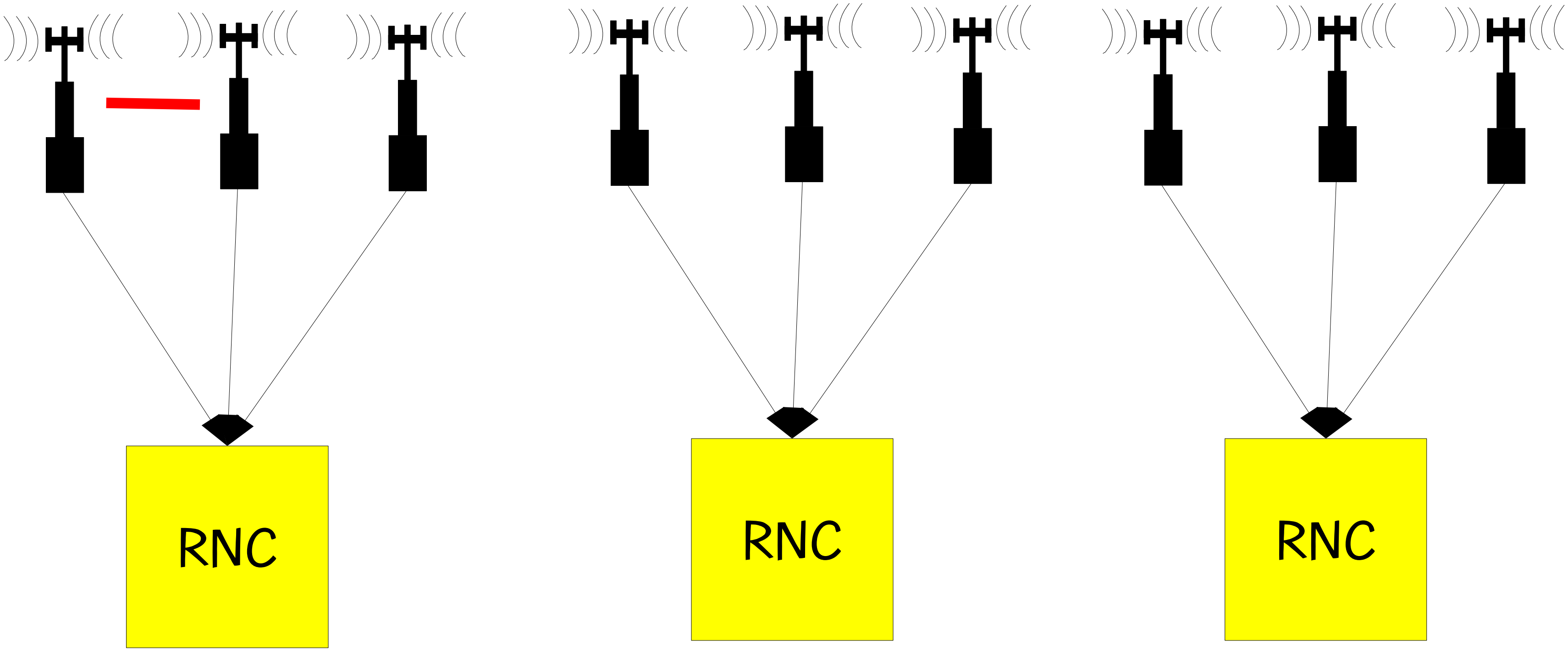




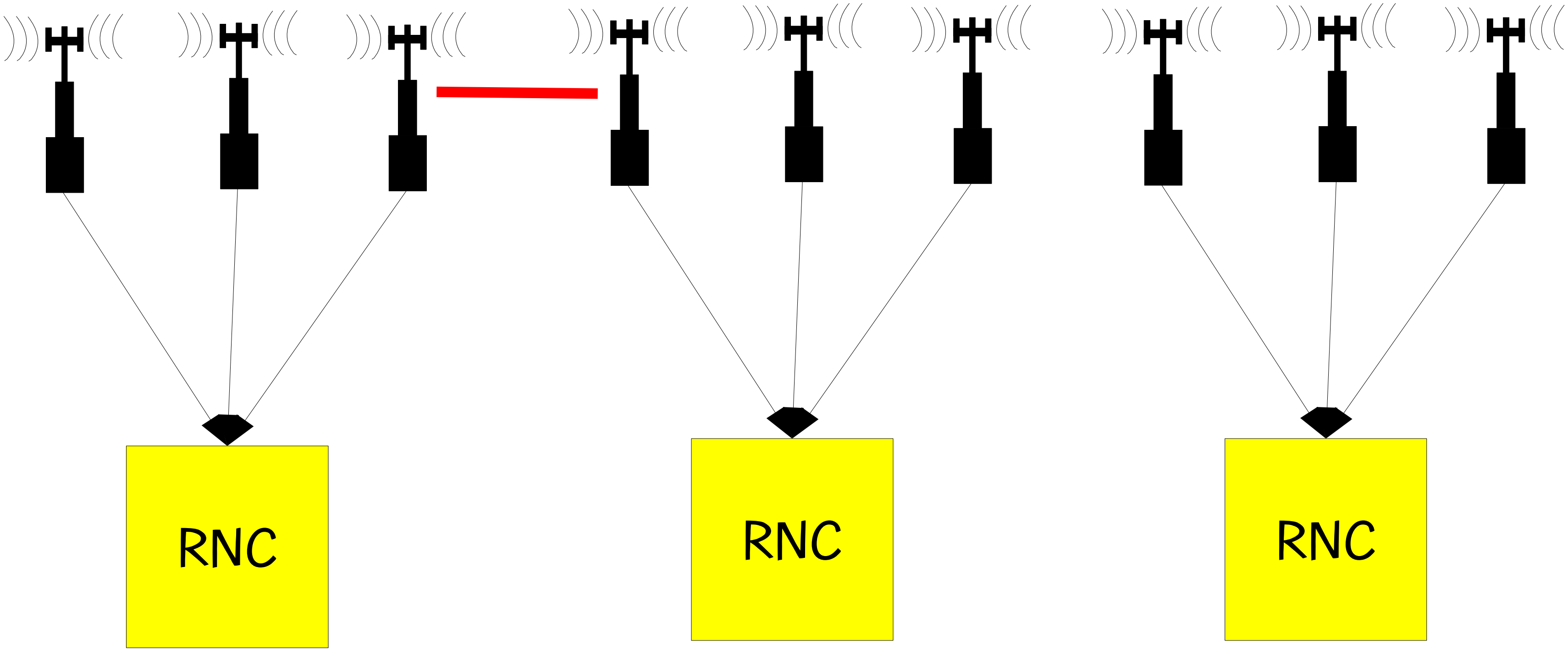
- Each cell tower has associated with it an amount of traffic.
- Each cell tower is connected to a Radio Network Controller (RNC).
- Each RNC can have one or more cell towers connected to it.
- Each RNC can handle a given amount of traffic ... this limits the subsets of cell towers that can be connected to it.
- An RNC controls the cell towers connected to it.



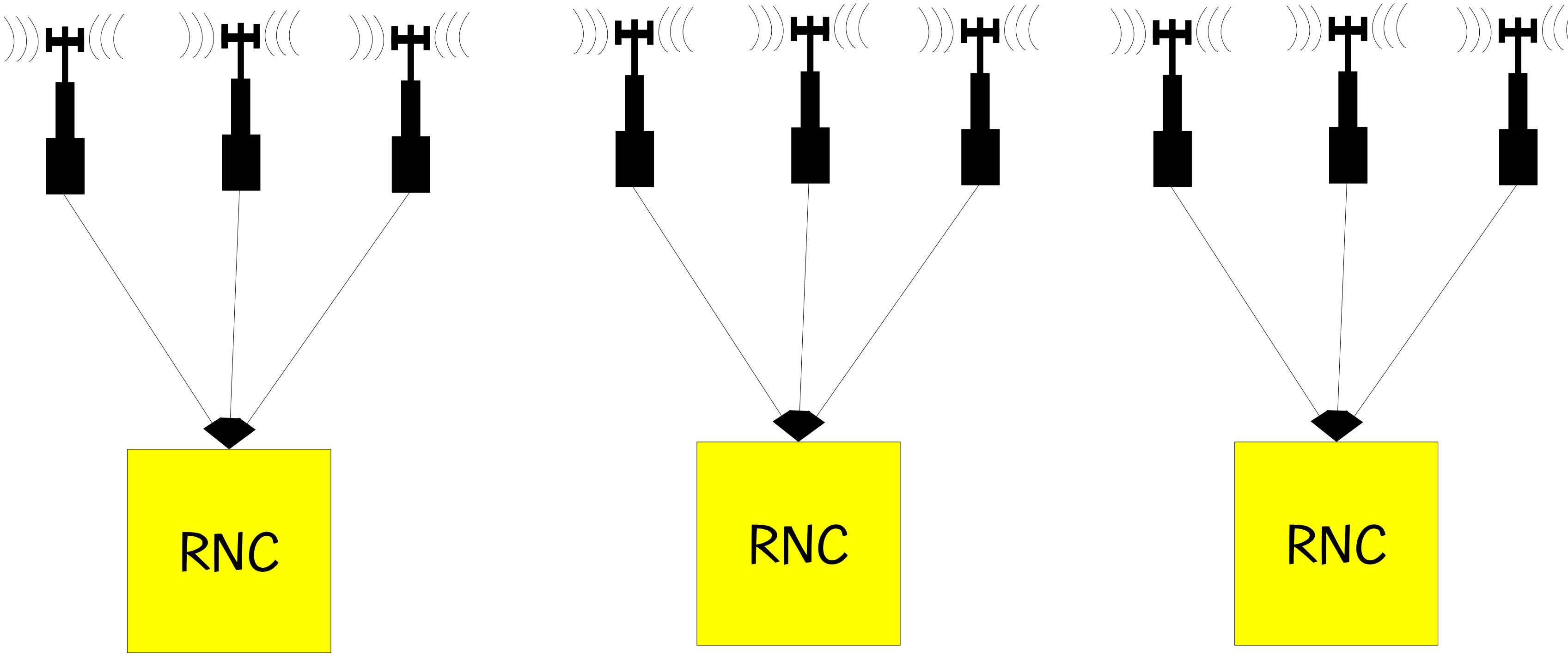
- Handovers can occur between towers



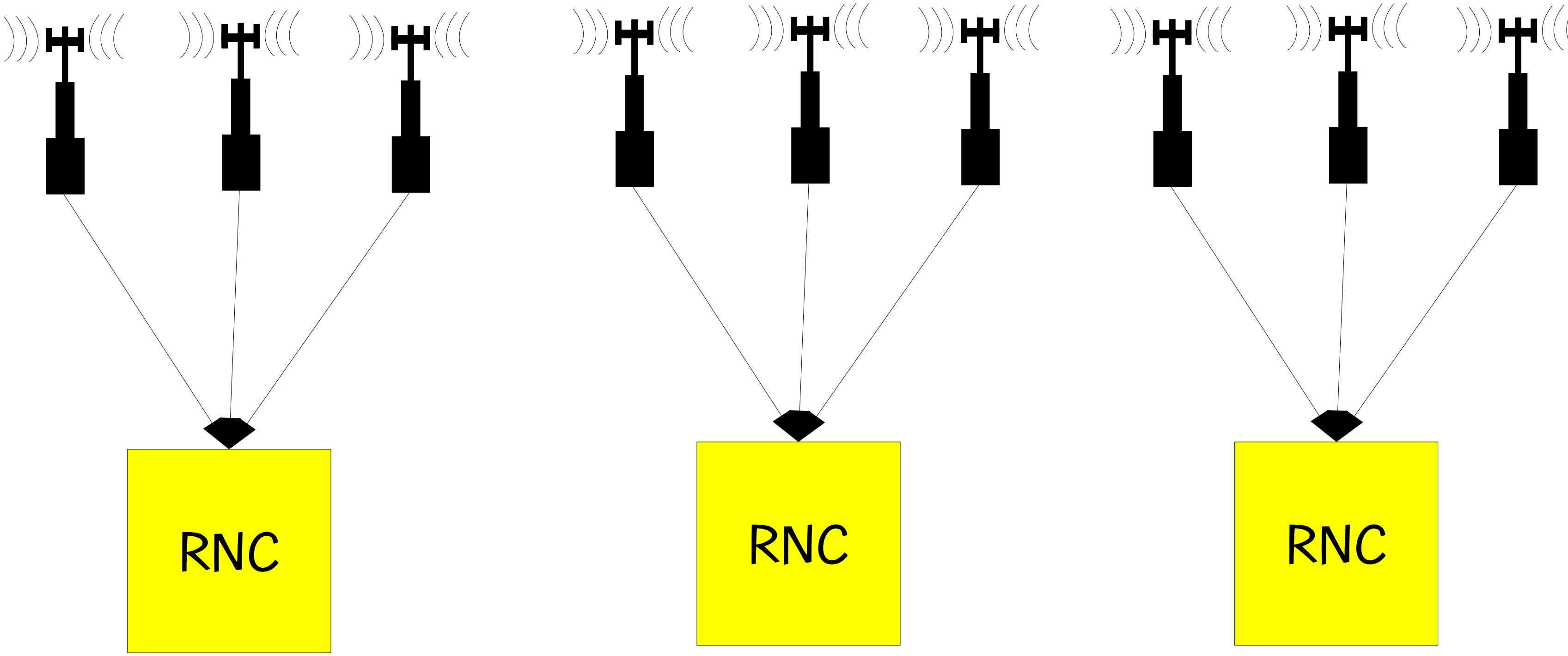
- Handovers can occur between towers
  - connected to the same RNC



- Handovers can occur between towers
  - connected to the same RNC
  - connected to different RNCs

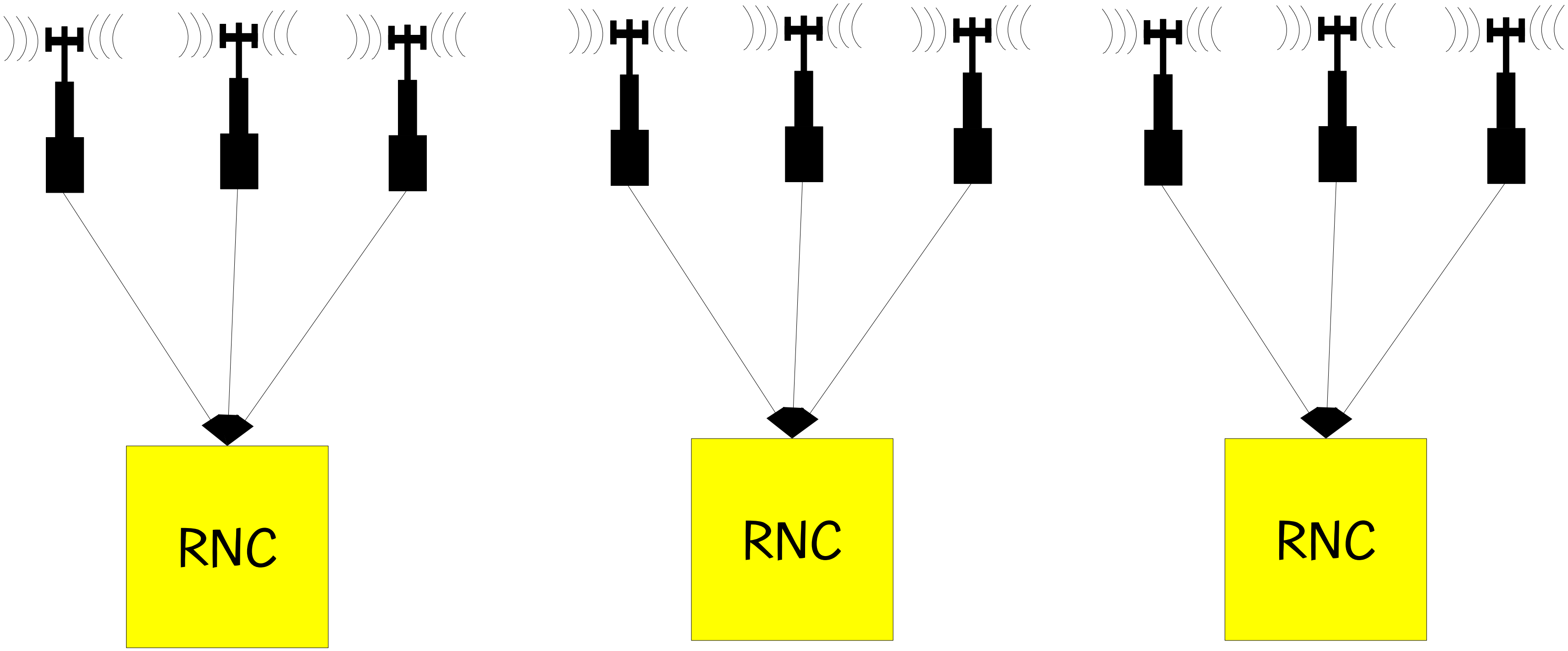


- Handovers between towers connected to different RNCs tend to fail more often than handovers between towers connected to the same RNC.
- Handover failure results in dropped call!

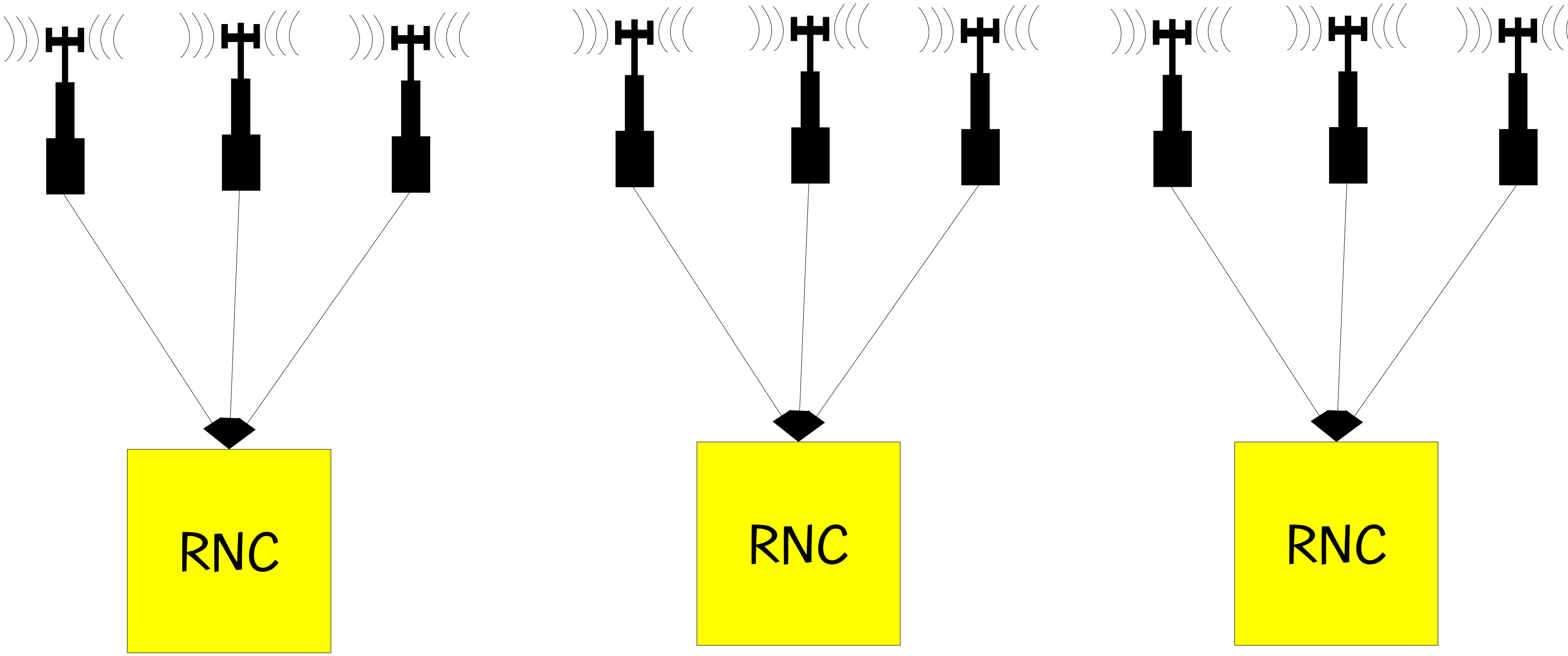


- If we minimize the number of handovers between towers connected to different RNCs we may be able to reduce the number of dropped calls.

at&t
Your world. Delivered.



- HANDOVER MINIMIZATION: Assign towers to RNCs such that RNC capacity is not violated and number of handovers between towers assigned to different RNCs is minimized.

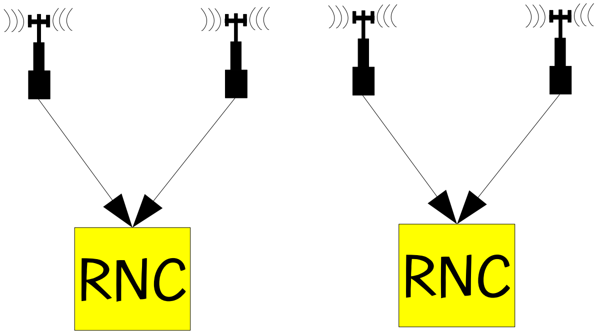


- HANDOVER MINIMIZATION: Assign towers to RNCs such that RNC capacity is not violated and number of handovers between towers assigned to different RNCs is minimized.

Node-capacitated graph partitioning problem

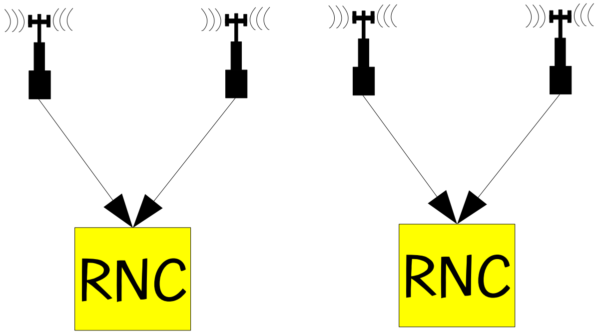
at&t
Your world. Delivered.

Example

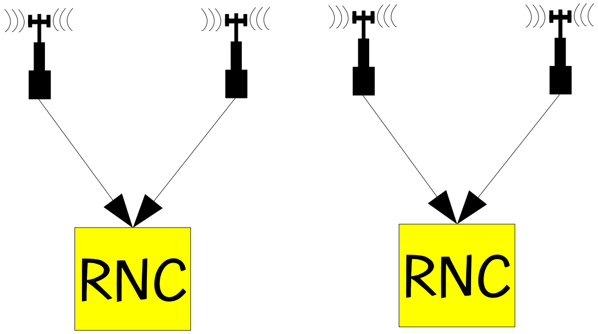


- 4 towers: t(1) = 25; t(2) = 15; t(3) = 35; t(4) = 25
- 2 RNCs: c(1) = 50; c(2) = 60
- Handover matrix:

	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

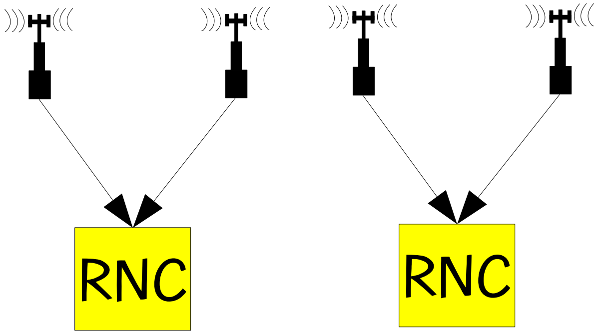


- 4 towers: t(1) = 25; t(2) = 15; t(3) = 35; t(4) = 25
- 2 RNCs: c(1) = 50; c(2) = 60
- Given this traffic profile and RNC capacities the feasible configurations are:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }
  - RNC(1): { 1, 4 }; RNC(2): { 2, 3 }



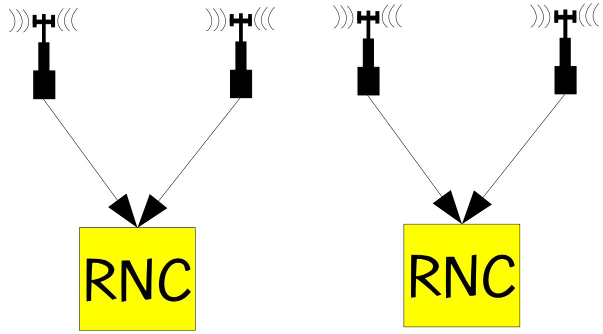
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:



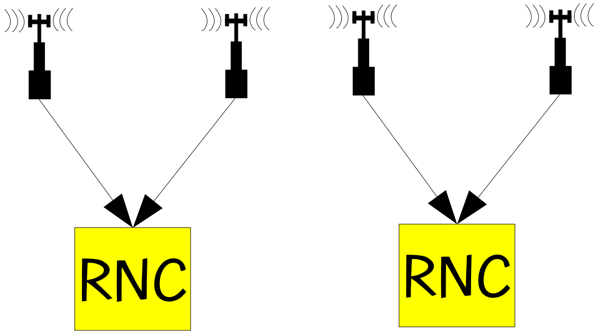
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = 260



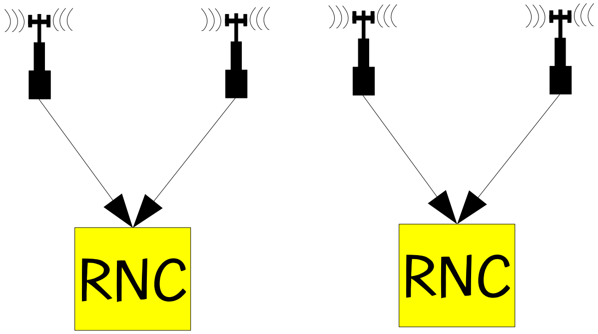
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = 260
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }: h(2,1) + h(2,4) + h(3,1) + h(3,4) = 100 + 50 + 10 + 500 = 660



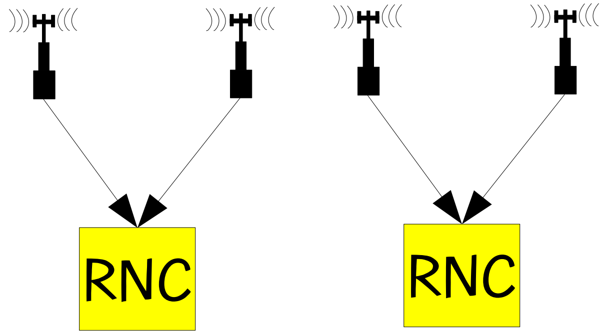
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = 260
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }: h(2,1) + h(2,4) + h(3,1) + h(3,4) = 100 + 50 + 10 + 500 = 660
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }: h(2,1) + h(2,3) + h(4,1) + h(4,3) = 100 + 200 + 0 + 500 = 800



	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = 260
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }: h(2,1) + h(2,4) + h(3,1) + h(3,4) = 100 + 50 + 10 + 500 = 660
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }: h(2,1) + h(2,3) + h(4,1) + h(4,3) = 100 + 200 + 0 + 500 = 800
  - RNC(1): { 1, 4 }; RNC(2): { 2, 3 }: h(1,2) + h(1,3) + h(4,2) + h(4,3) = 100 + 10 + 50 + 500 = 660

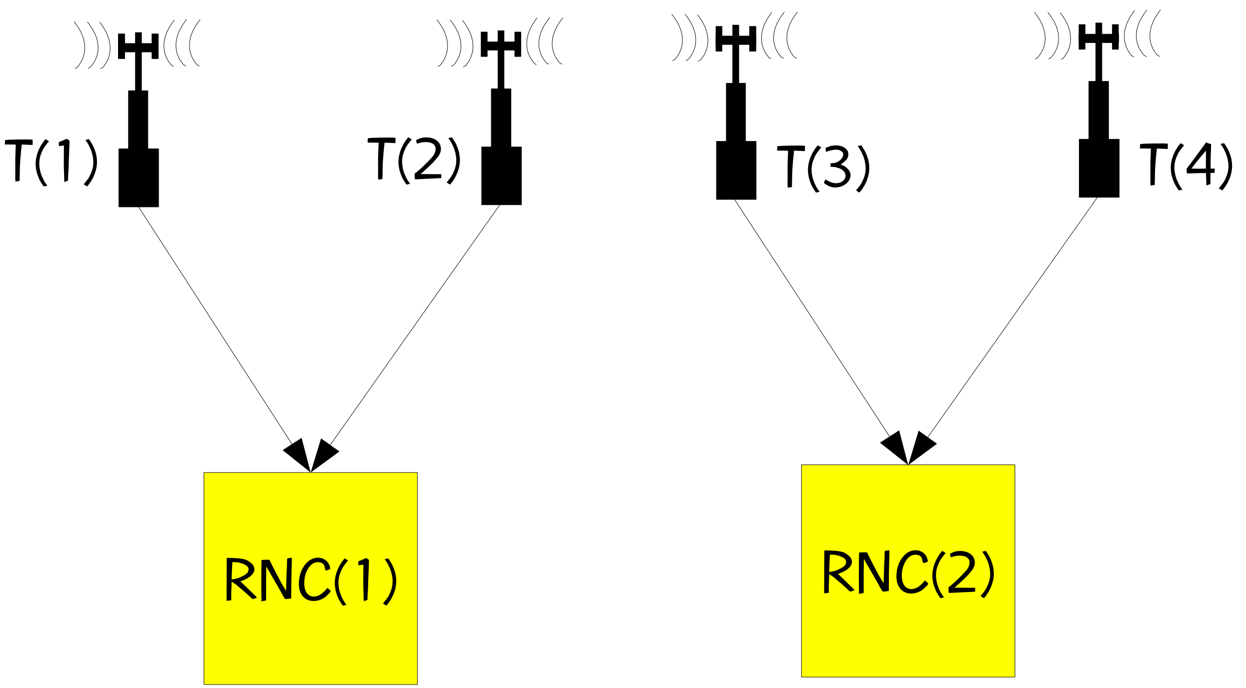


	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

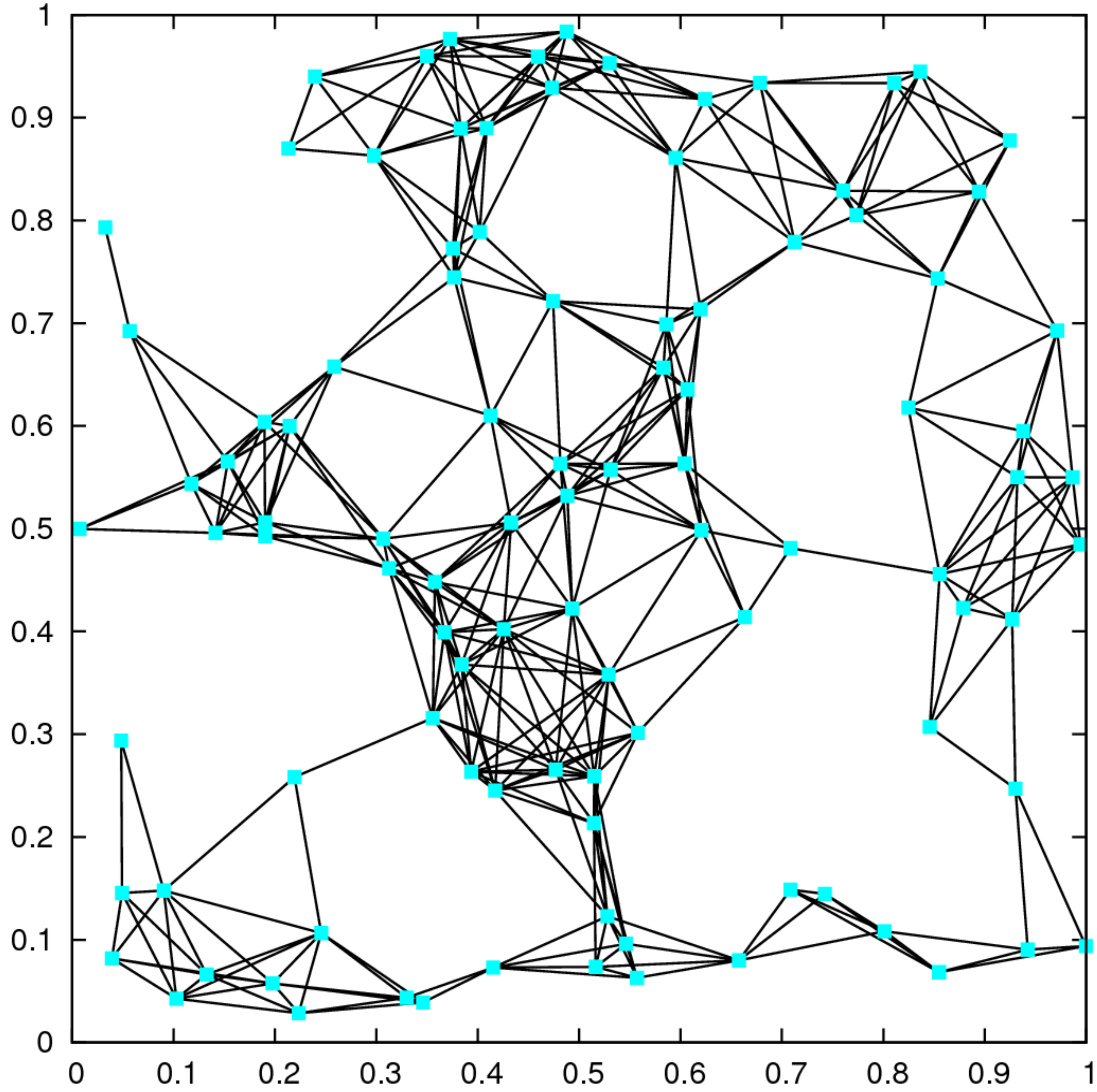
- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = **260**
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }: h(2,1) + h(2,4) + h(3,1) + h(3,4) = 100 + 50 + 10 + 500 = 660
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }: h(2,1) + h(2,3) + h(4,1) + h(4,3) = 100 + 200 + 0 + 500 = 800
  - RNC(1): { 1, 4 }; RNC(2): { 2, 3 }: h(1,2) + h(1,3) + h(4,2) + h(4,3) = 100 + 10 + 50 + 500 = 660

	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

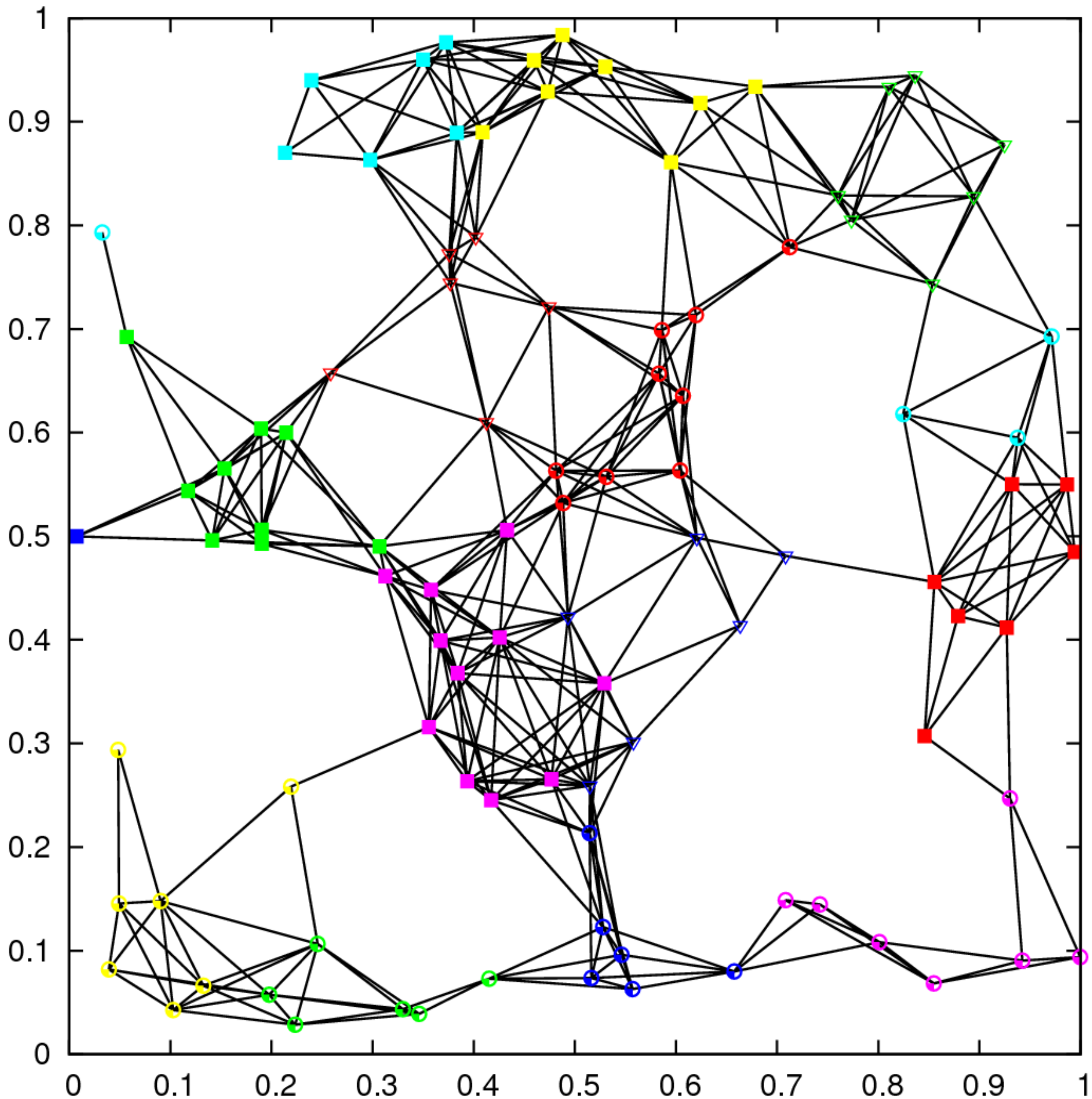
Optimal configuration:



G=(T,E) Nodeset T are the towers; Edgeset: $(i,j) \in E$ iff $h(i,j)+h(j,i) > 0$



Tower are assigned to RNCs indicated by distinct colors/shapes



at&t
Your world. Delivered.

Mixed integer programming formulation

- T is the set of towers
- R is the set of RNCs
- $x_{e,k} = 1$ if edge $e = (i,j)$ has both endpoints in RNC k
- $y_{i,k} = 1$ if tower i is assigned to RNC k

Mixed integer programming formulation

Each tower can only be assigned to one RNC:

$$\text{sum}_{\{k \in R\}}\, y_{i,k} = 1, \text{ for all } i \in T$$

Mixed integer programming formulation

Each $e=(i,j)$ cannot be in RNC k if either of its endpoints is not assigned to RNC k:

$$x_{e,k} \leq y_{i,k}\text{, for all } e=(i,j) \in E, k \in R$$

$$x_{e,k} \leq y_{j,k}\text{, for all } e=(i,j) \in E, k \in R$$

$$x_{e,k} \geq y_{i,k} + y_{j,k} - 1\text{, for all } e=(i,j) \in E, k \in R$$

Mixed integer programming formulation

Each RNC k can only accommodate c_k units of traffic:

$$\text{sum}_{\{i \in T\}} t_i y_{i,k} \leq c_k, \text{ for all } k \in R$$

Mixed integer programming formulation

Minimize handover between towers assigned to different RNCs is equivalent to maximize handover between towers assigned to the same RNC.

Objective function:

$$\max \{ \text{sum}_{\{k \in R\}} \{ \text{sum}_{\{e=(i,j) \in E\}} h(i,j)\ x_{e,k} \} \}$$

CPLEX MIP solver

Towers	RNCs	BKS	CPLEX	time (s)
20	10	7602	7602	18.80
30	15	18266	18266	25911.00
40	15	29700	29700	101259.91
100	15	19000	49270	1 day
100	25	36412	58637	1 day
100	50	60922	70740	1 day

at&t
Your world. Delivered.

CPLEX MIP solver

Towers	RNCs	BKS	CPLEX	time (s)
20	10	7602	7602	18.80
30	15	18266	18266	25911.00
40	15	29700	29700	101259.91
100	15	19000	49270	1 day
100	25	36412	58637	1 day
100	50	60922	70740	1 day

We would like to solve instances with 1000 towers.

CPLEX MIP solver

Towers	RNCs	BKS	CPLEX	time (s)
20	10	7602	7602	18.80
30	15	18266	18266	25911.00
40	15	29700	29700	101259.91
100	15	19000	49270	1 day
100	25	36412	58637	1 day
100	50	60922	70740	1 day

We would like to solve instances with 1000 towers.

Need heuristics!

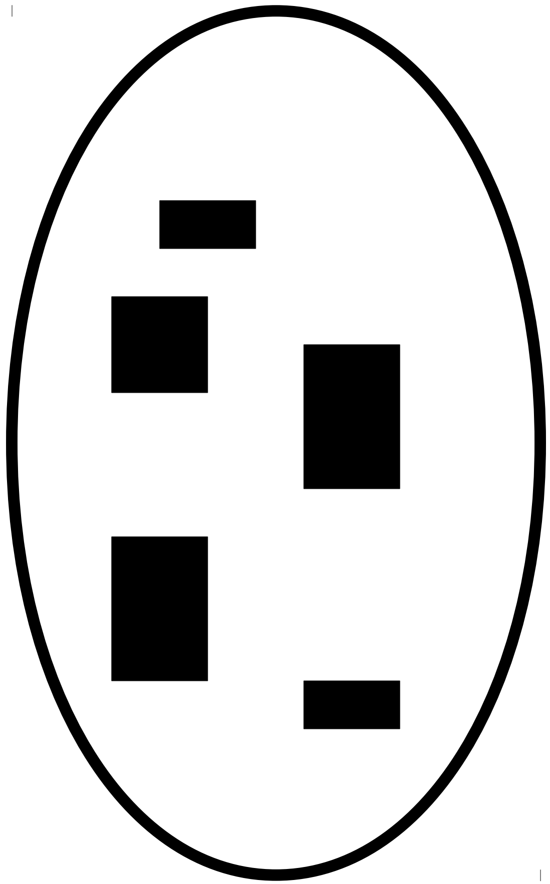
Generalized quadratic assignment problem

Generalized quadratic assignment

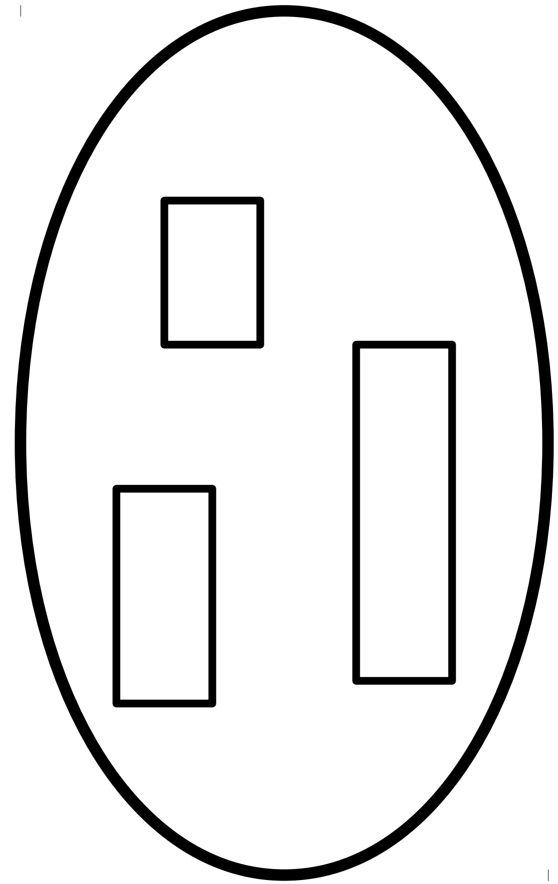
Generalization of the quadratic assignment problem (QAP).

Multiple facilities can be assigned to a single location as long as the capacity of the location allows.

N: set of n facilities



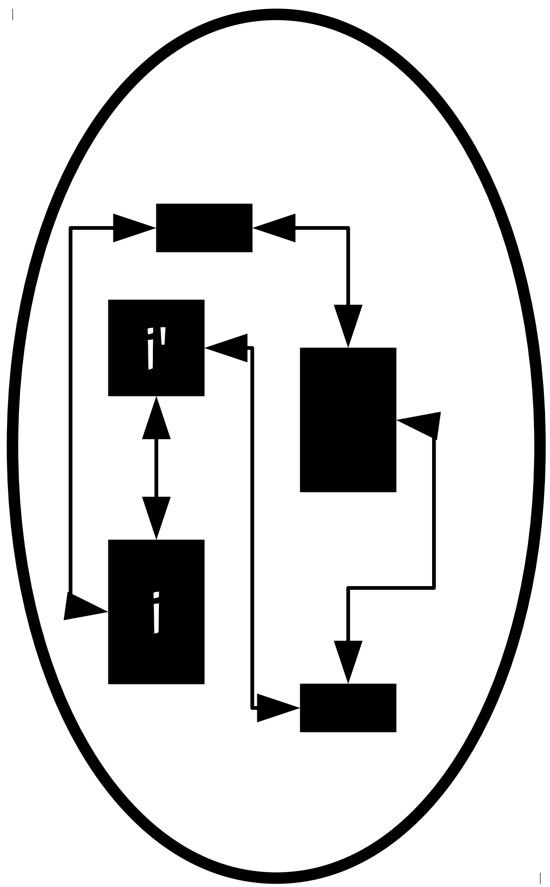
M: set of m locations



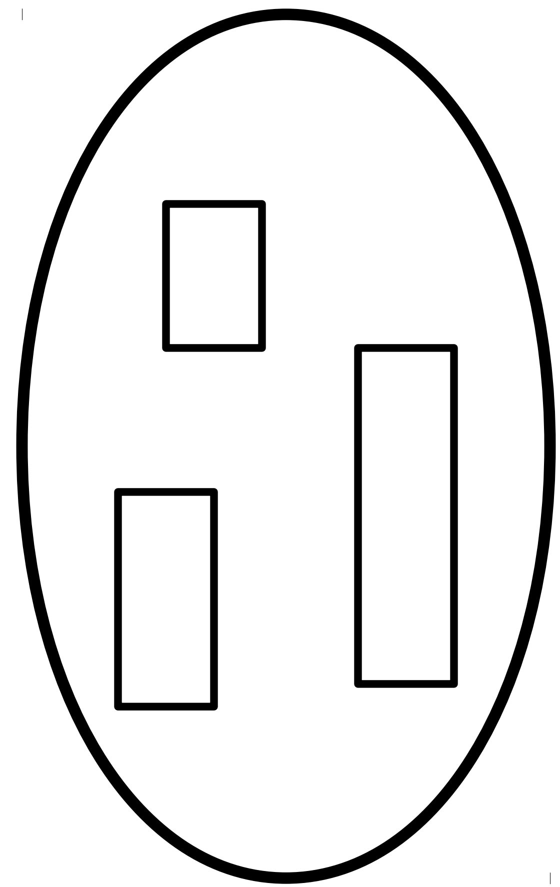
d_i : capacity demanded by facility $i \in N$

Q_j : capacity of location $j \in M$

N: set of n facilities



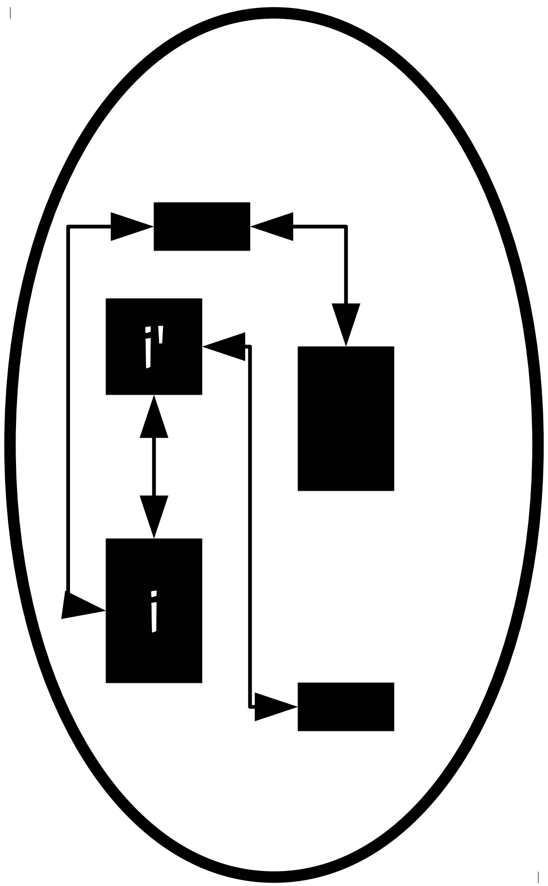
M: set of m locations

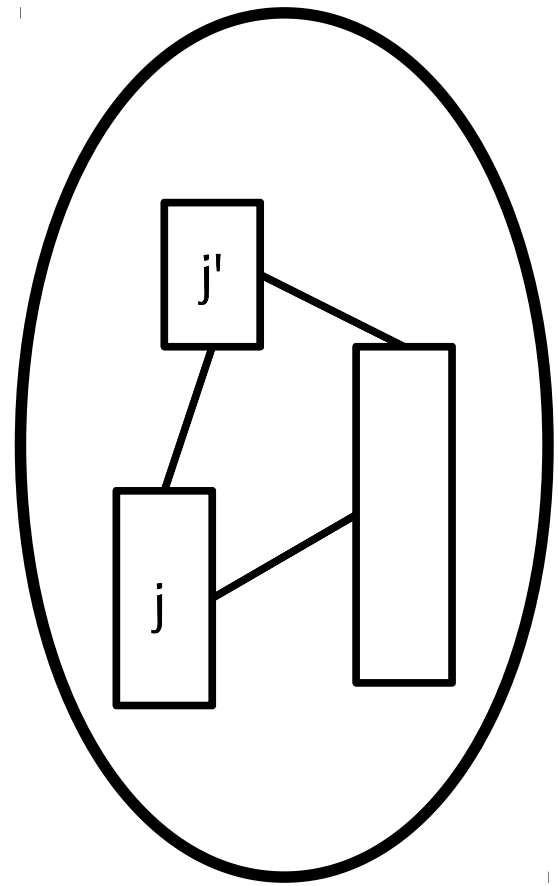


$A_{nxn}=(a_{ii'})$: flow between facilities

N: set of n facilities

M: set of m locations

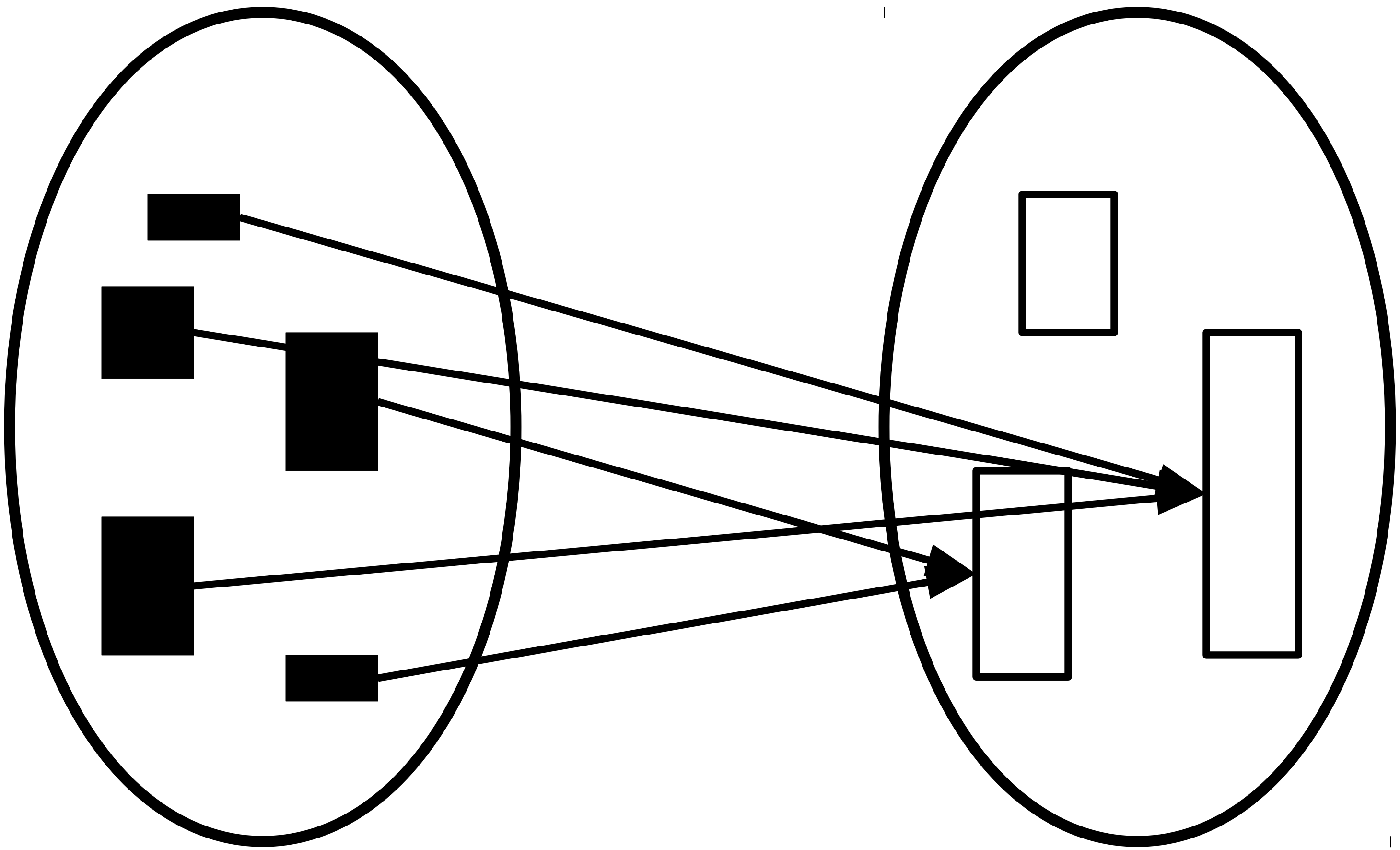




$A_{nxn} = (a_{ii'})$: flow between facilities

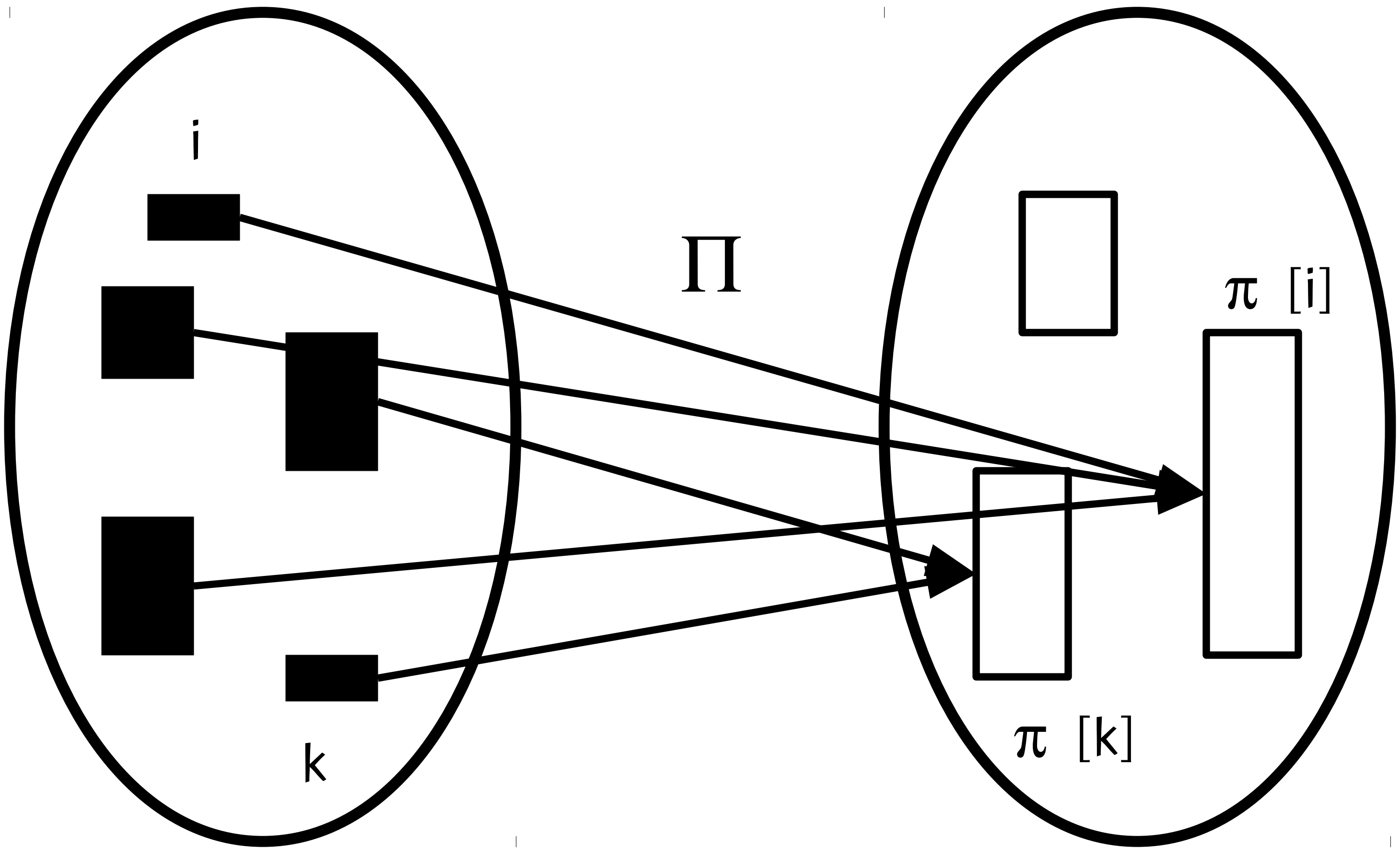
$B_{mxm} = (b_{jj'})$: distance between locations

The generalized quadratic assignment problem



GQAP seeks a assignment, without violating the capacities of locations, that minimizes the sum of products of flows and distances.

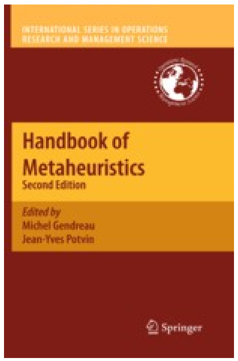
The generalized quadratic assignment problem



cost[Π] = sum(i=1,n) sum (i≠ k=1,n) F[i,k]*D[π [i],π [k]]

GRASP with path-relinking for GQAP

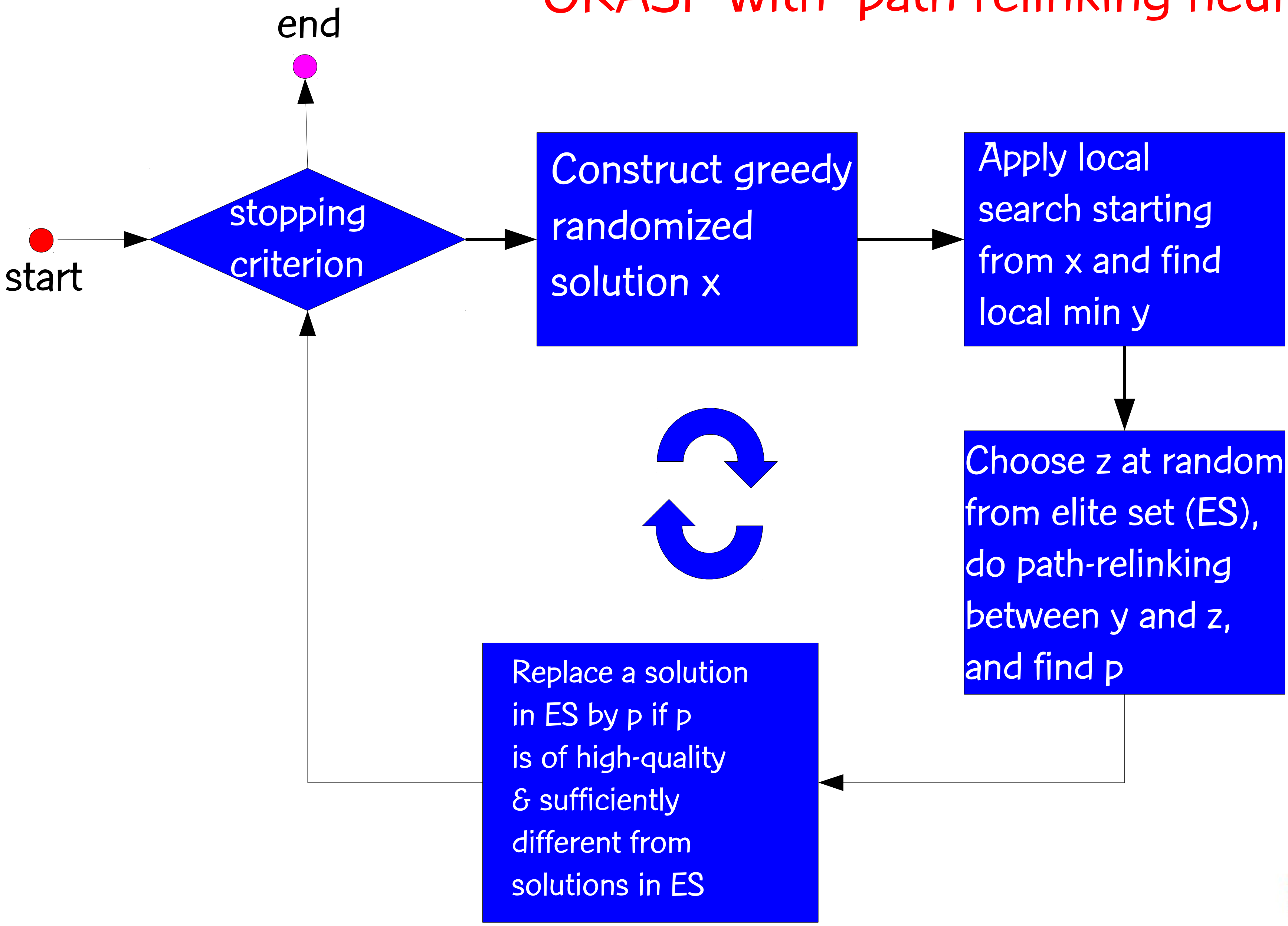
Recent survey of GRASP with path-relinking



M.G.C. Resende and C.C. Ribeiro, Greedy randomized adaptive search procedures: Fundamentals, advances, and applications, in Handbook of Metaheuristics, 2nd Edition, M. Gendreau and J.-Y. Potvin (Eds.), Springer, pp. 281-317, 2010.

http://www.research.att.com/~mgcr/doc/sgrasp2008a.pdf

GRASP with path-relinking heuristic



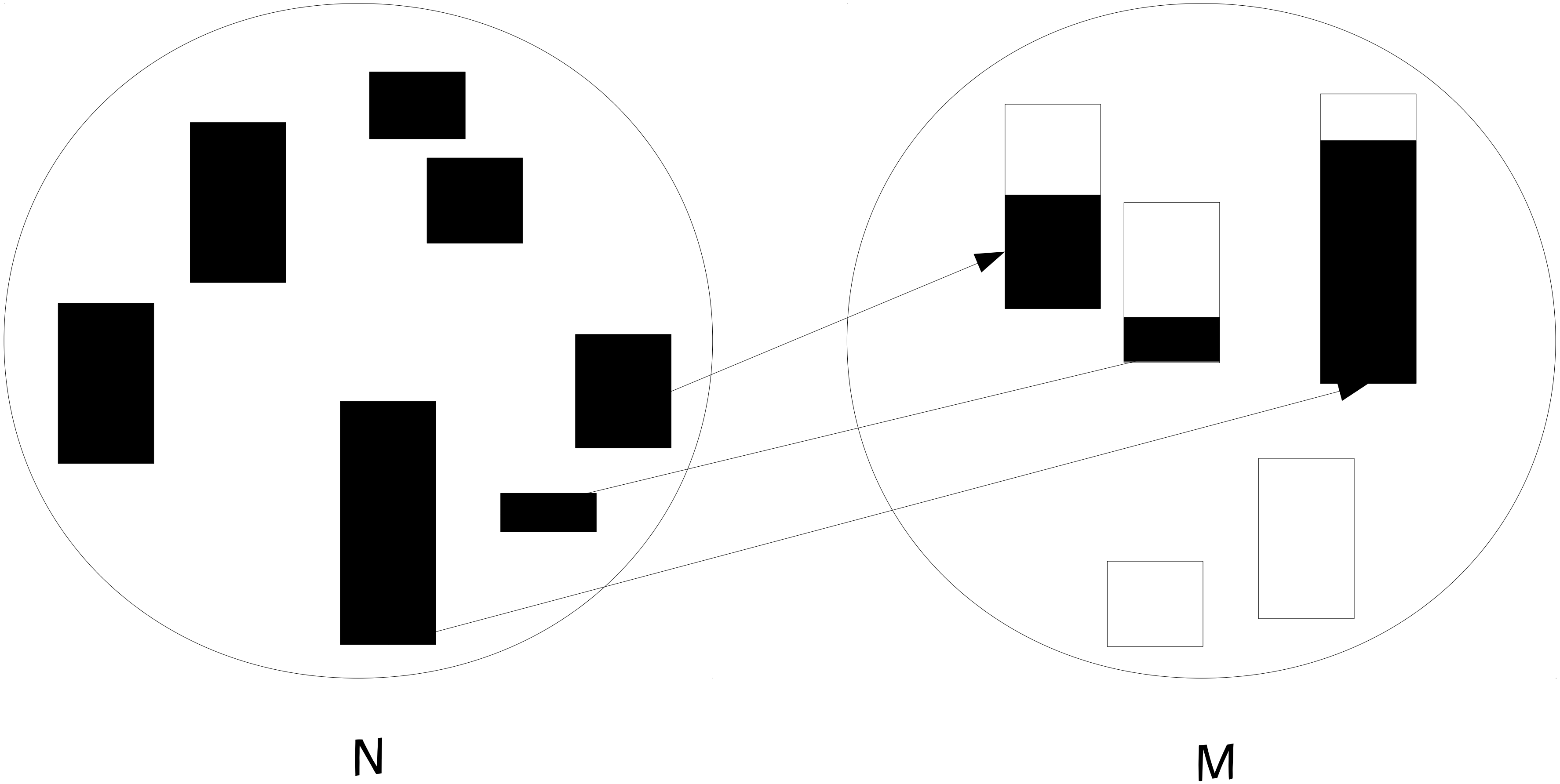
Components

Construction of greedy randomized solution

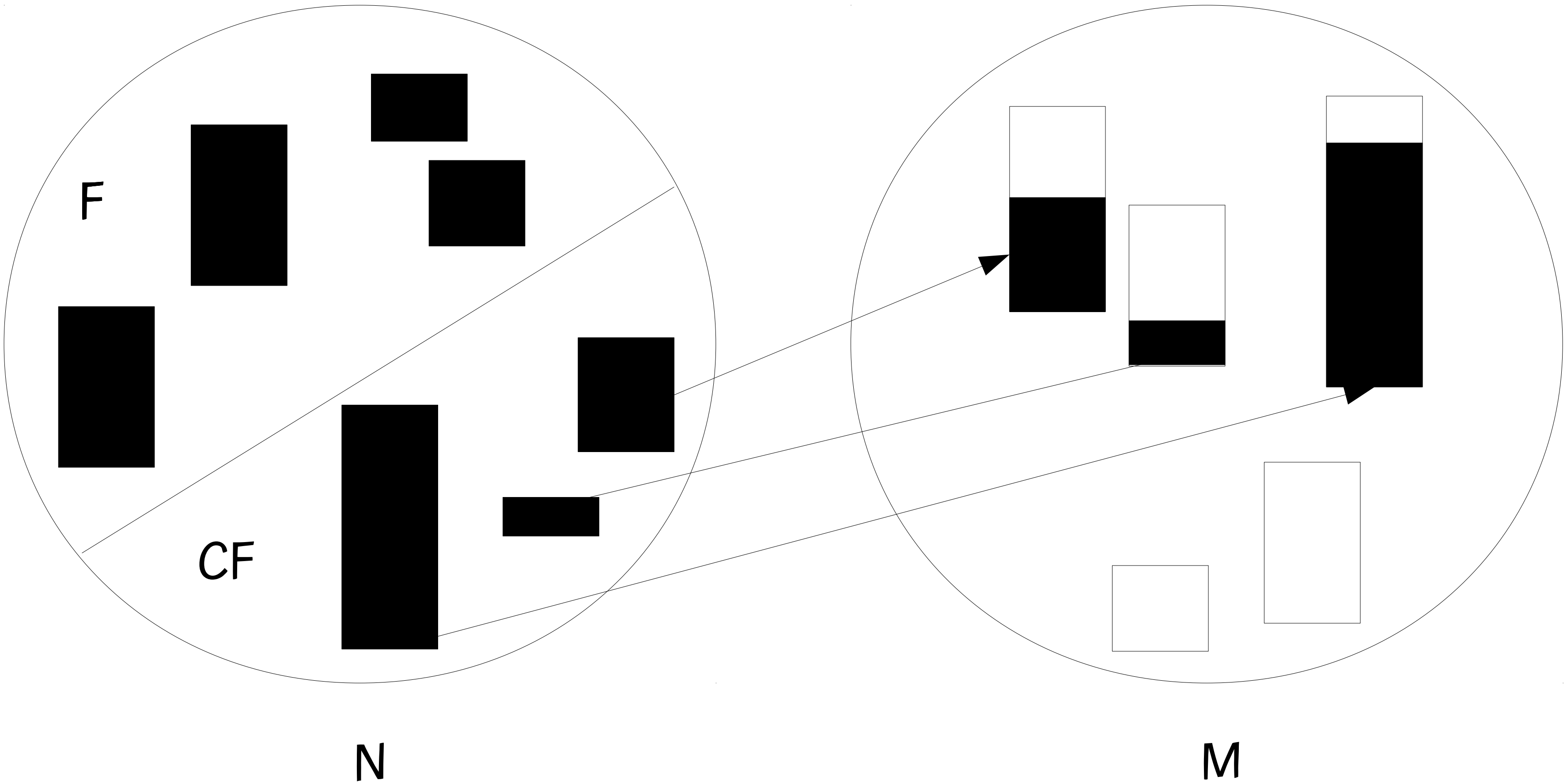
Local search

Path-relinking

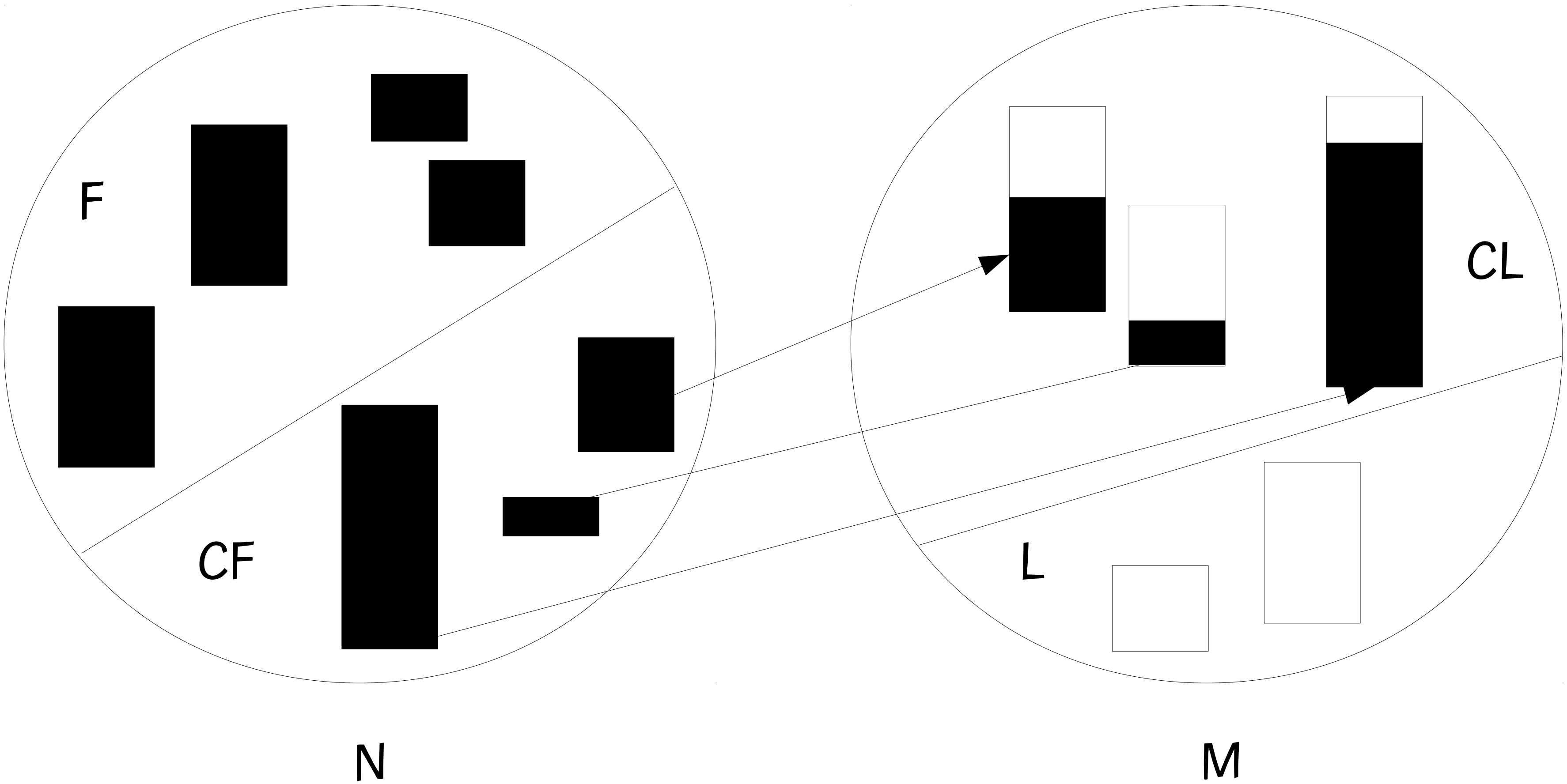
GRASP construction for GQAP



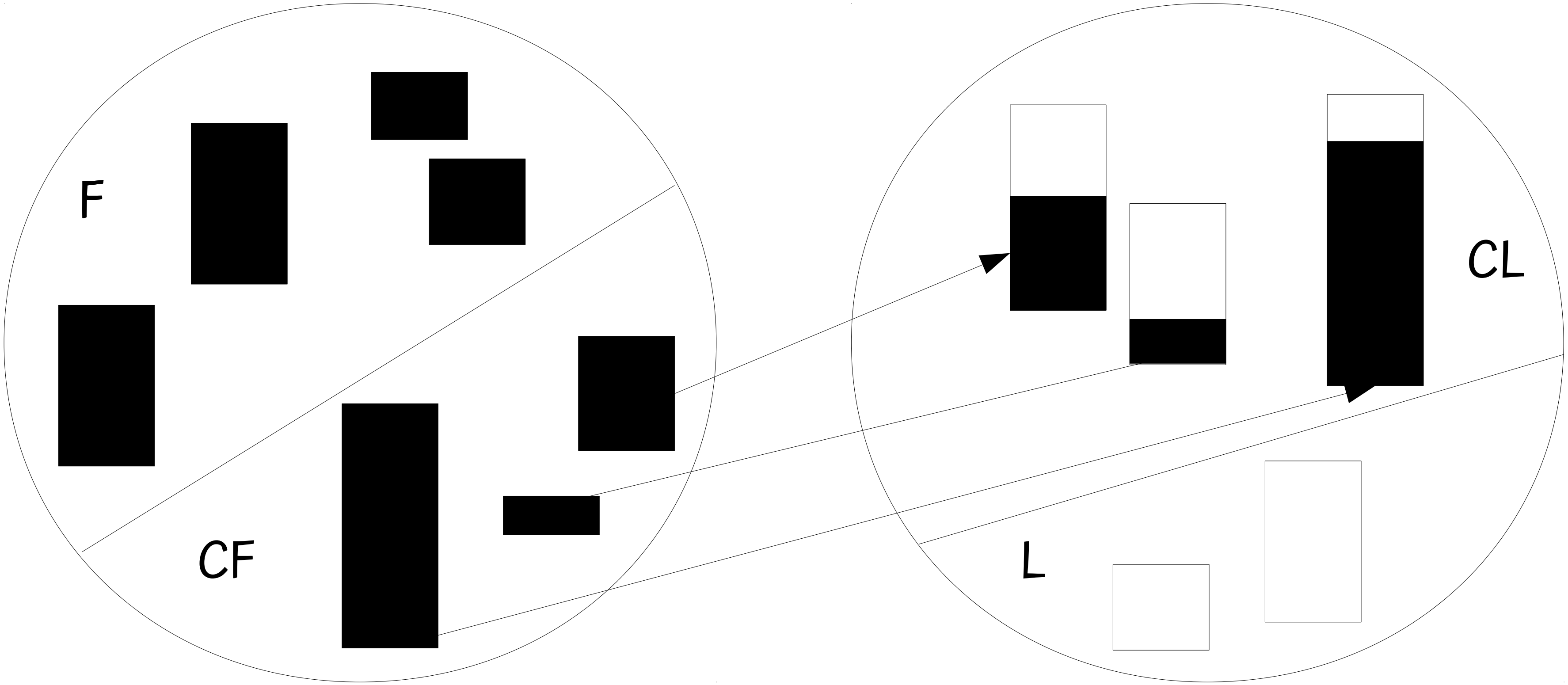
Suppose a number of assignments have already been made



$N = F \cup CF$, where CF is the set of assigned facilities and F the set of facilities not yet assigned to some location



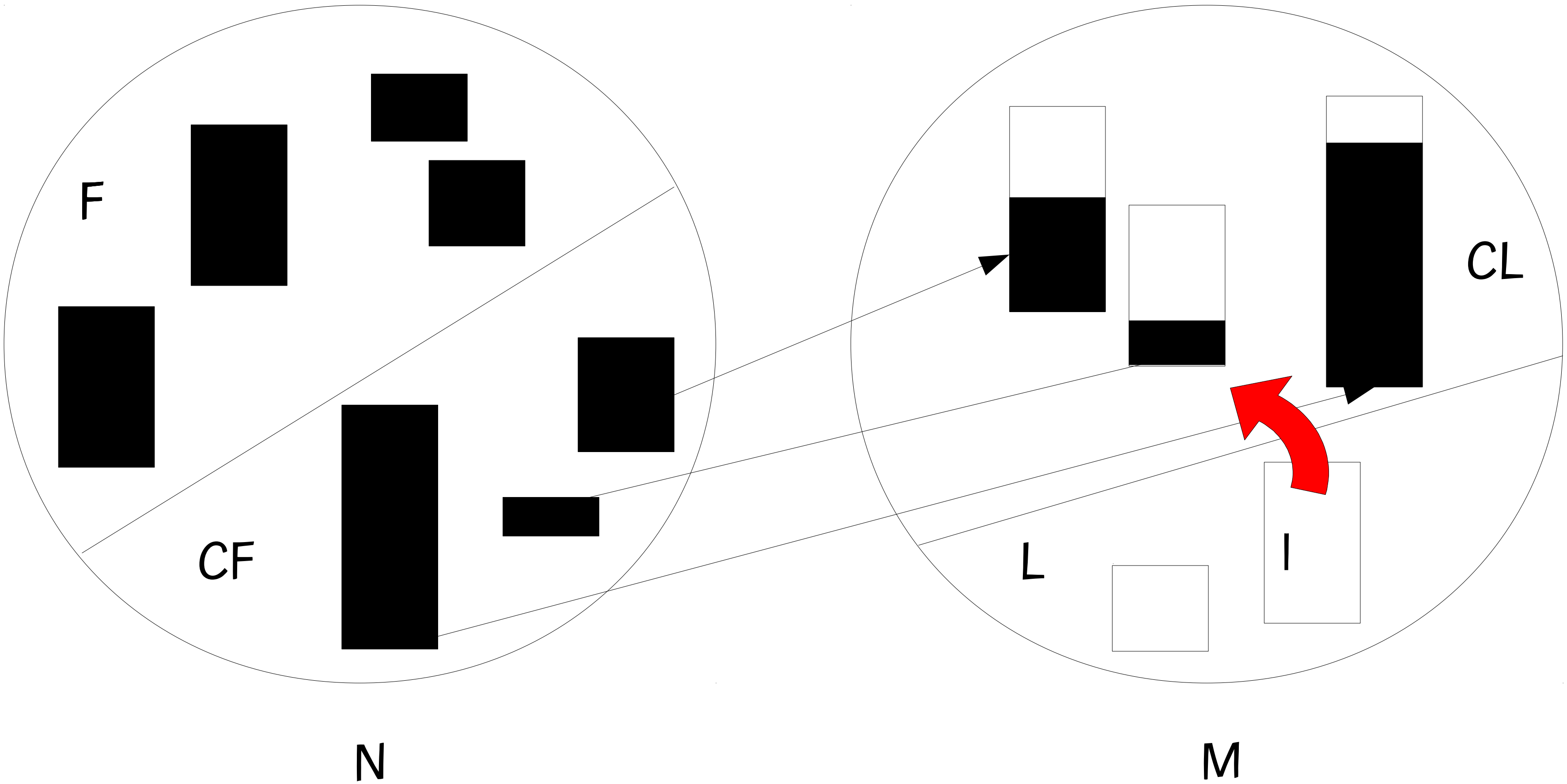
$M = L \cup CL$, where CL is the set of previously chosen locations and L the set of unselected locations.



Components of construction procedure:

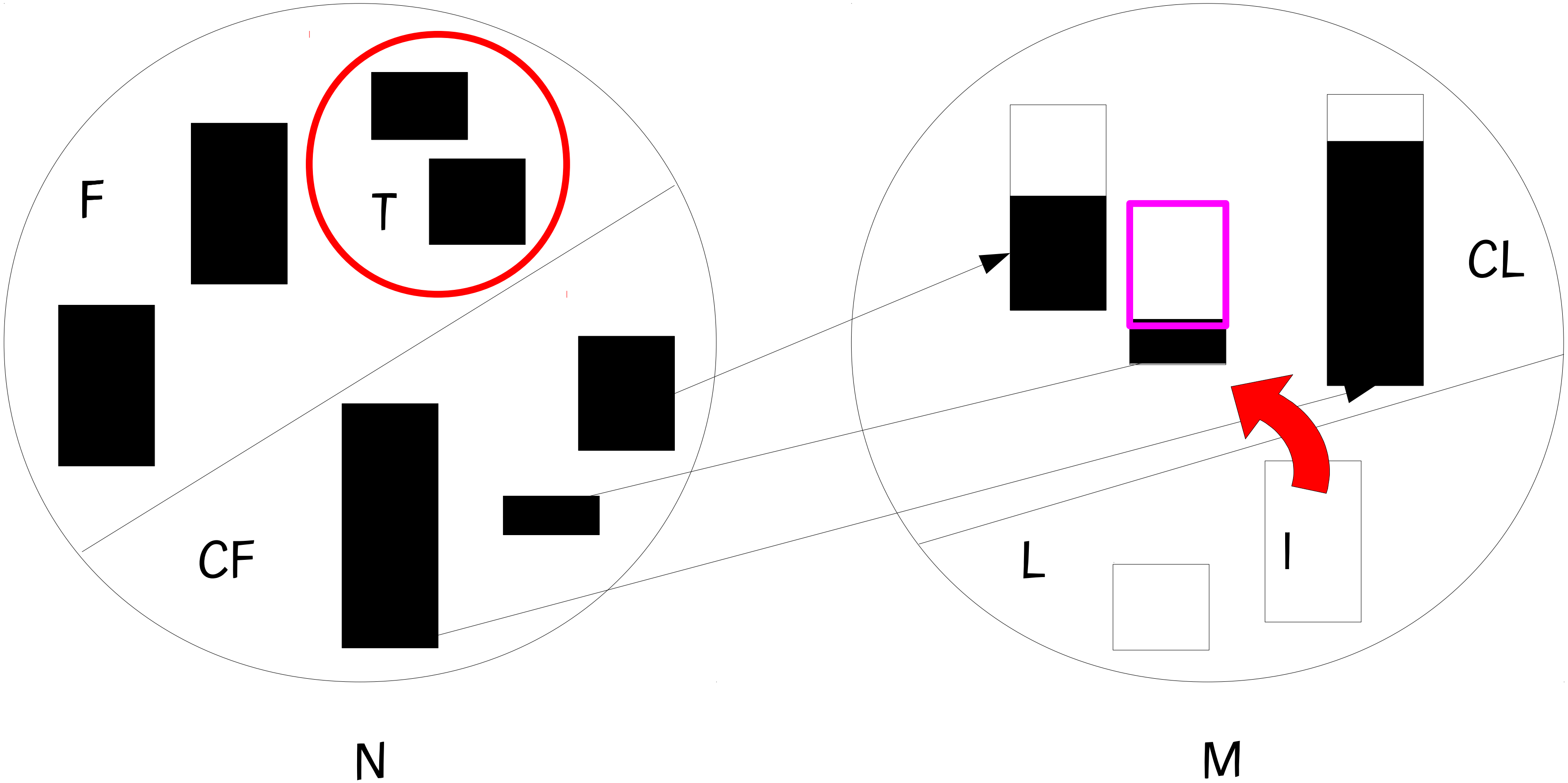
- procedure to select a NEW location from set L;
- procedure to select a facility from set F;
- procedure to select a location from set CL;

Procedure to select a NEW location from set L



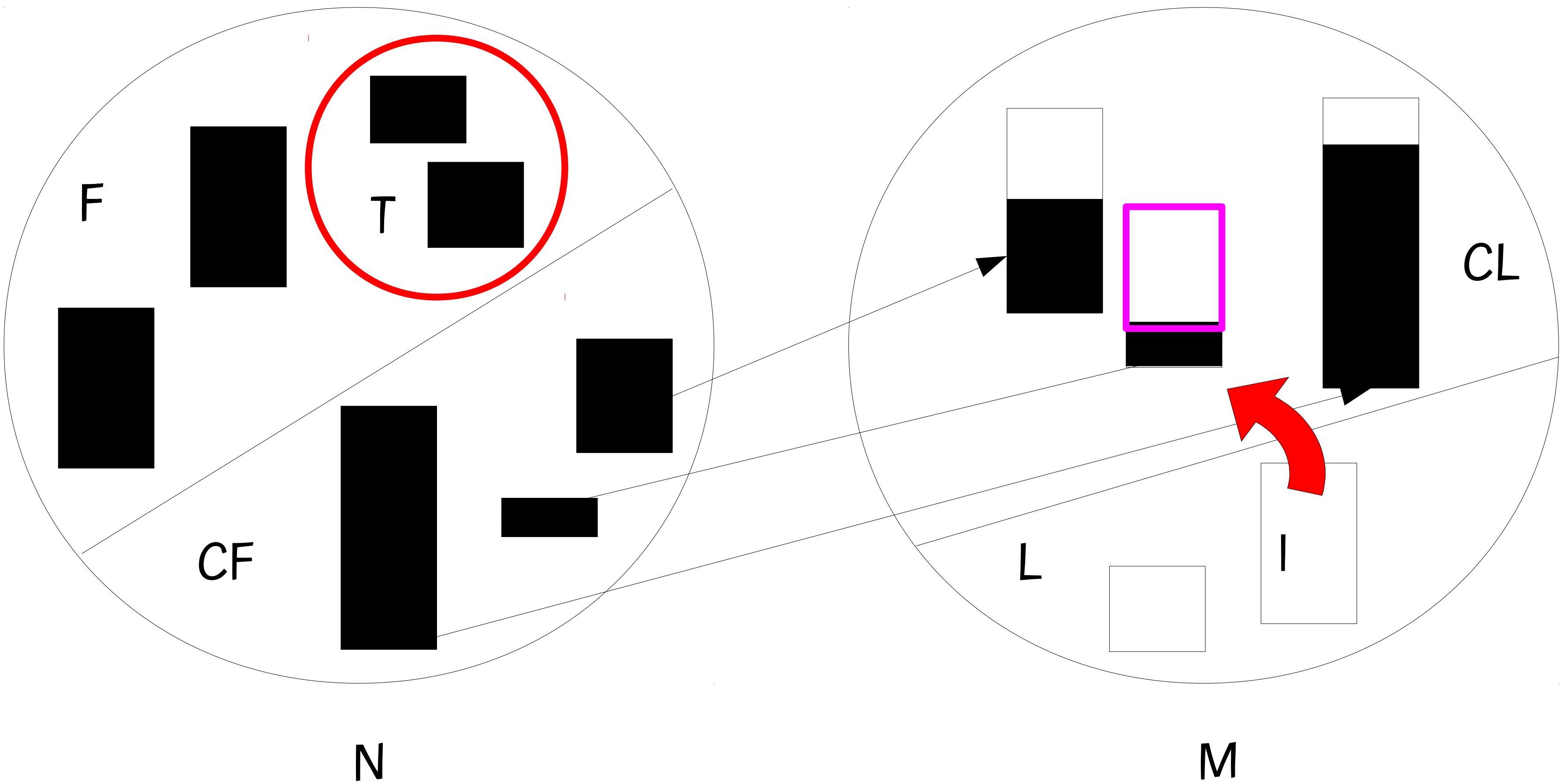
With probability P, randomly select a new location l from L, favoring those having high capacity and those close to all locations in CL, and move location l to CL.

Procedure to select a new location from set L



The probability P is equal to 1−(|T|/|F|), where the set T consists of all unassigned facilities with demands less than or equal to the maximum available capacity of locations in CL.

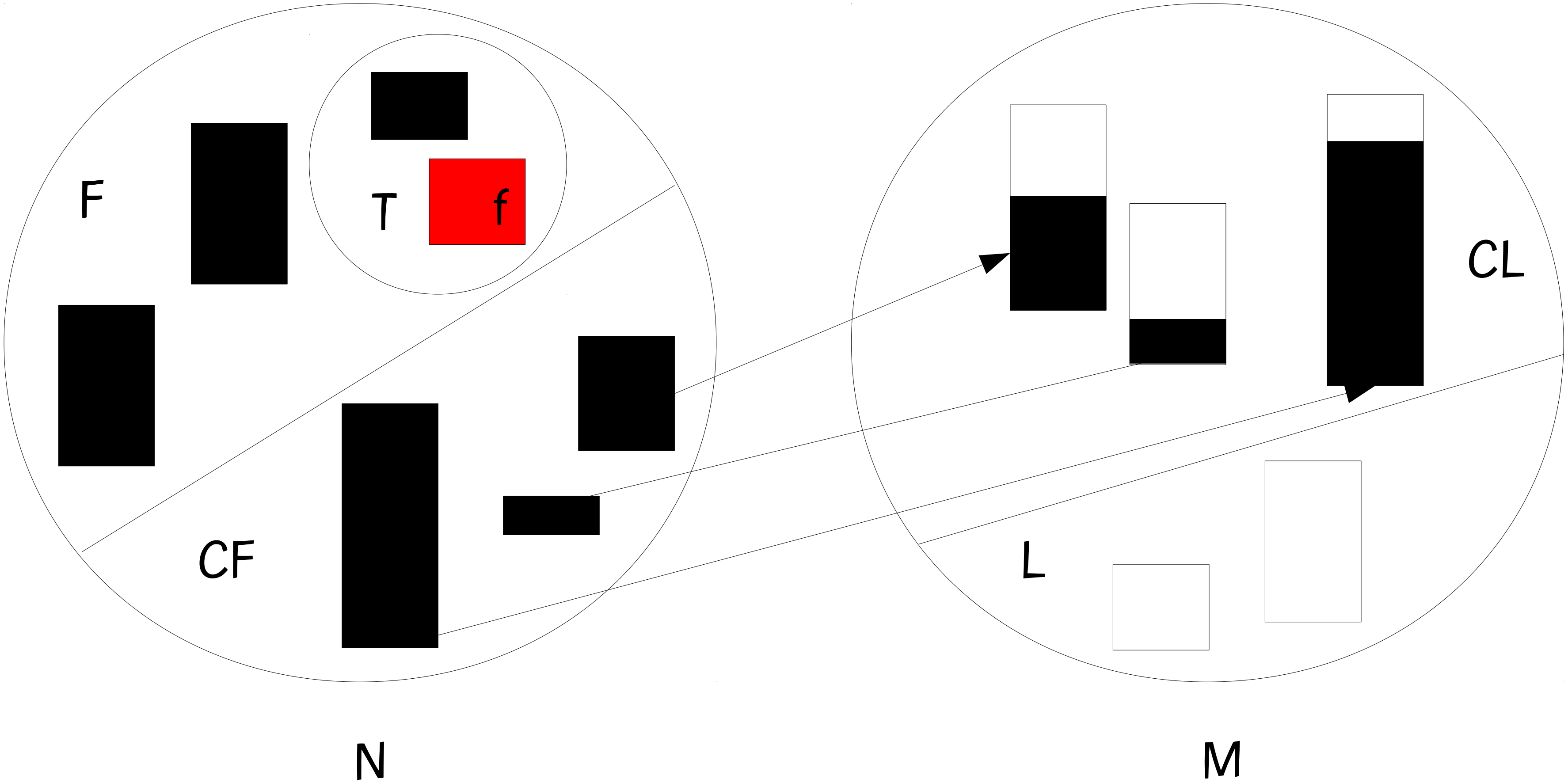
Procedure to select a new location from set L



With P = 1−(|T|/|F|):

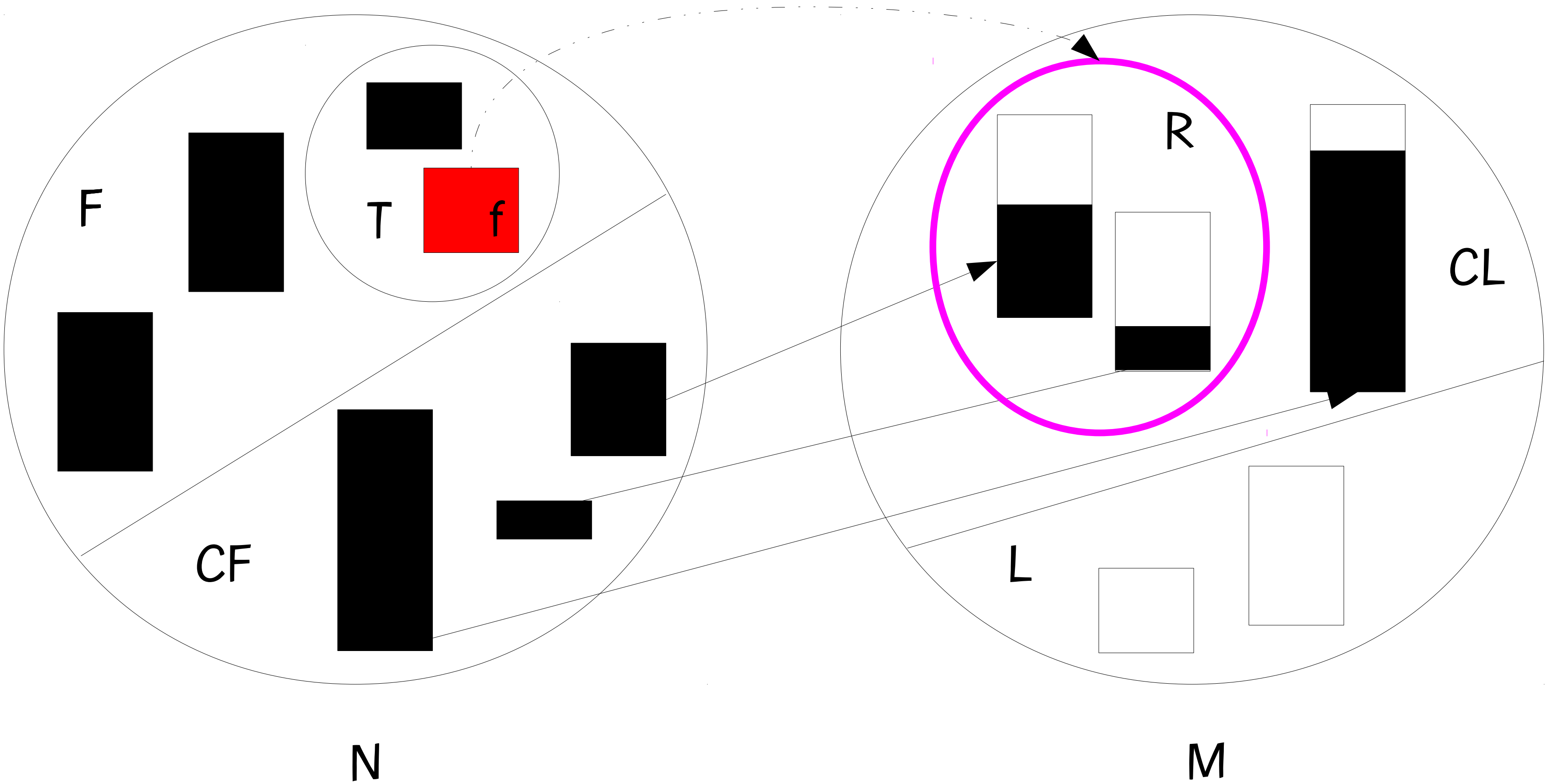
- if |T| is much less than |F|, then P tends to 1, which increases the need for a new location;
- if |T| tends to |F|, then P tends to 0, which reduces the need for a new location;

Facility selection procedure



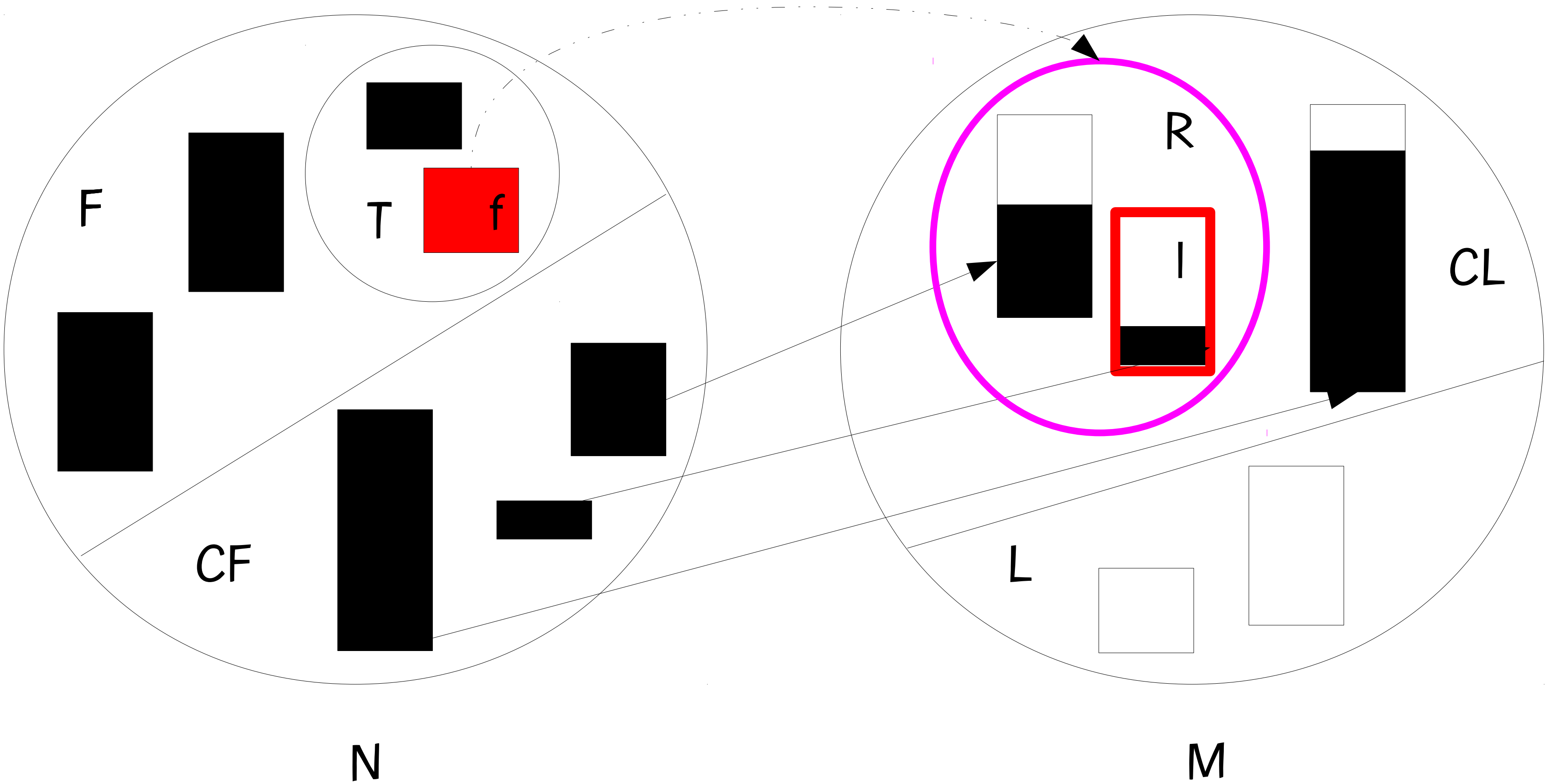
Randomly select a facility $f \in T$ favoring facilities that have high demand and high flows to other facilities.

Procedure to select a location from CL (step 1)



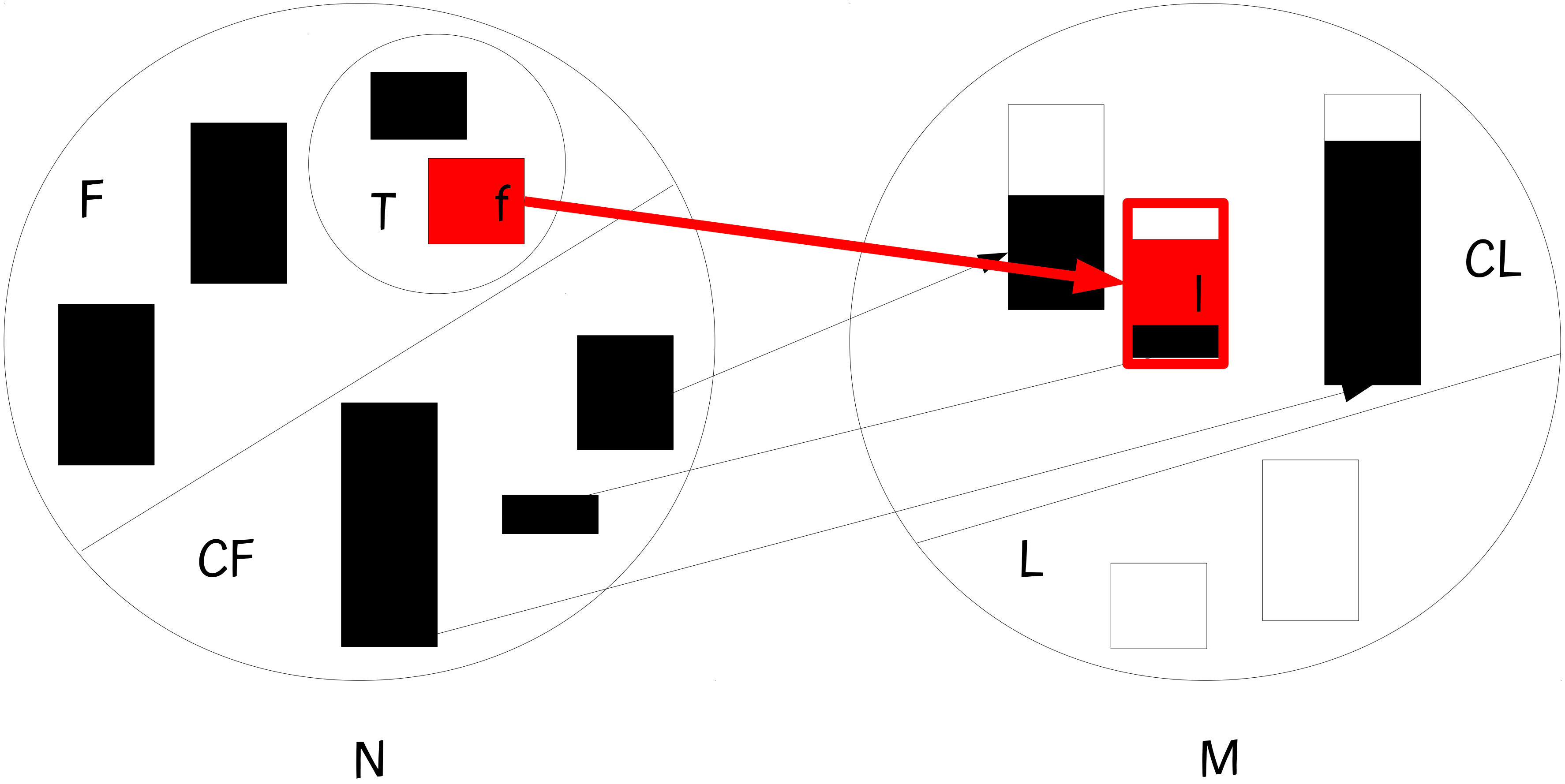
1. Let set R to be all locations in CL having slack greater than or equal to demand of facility f;

Procedure to select a location from CL (step 2)



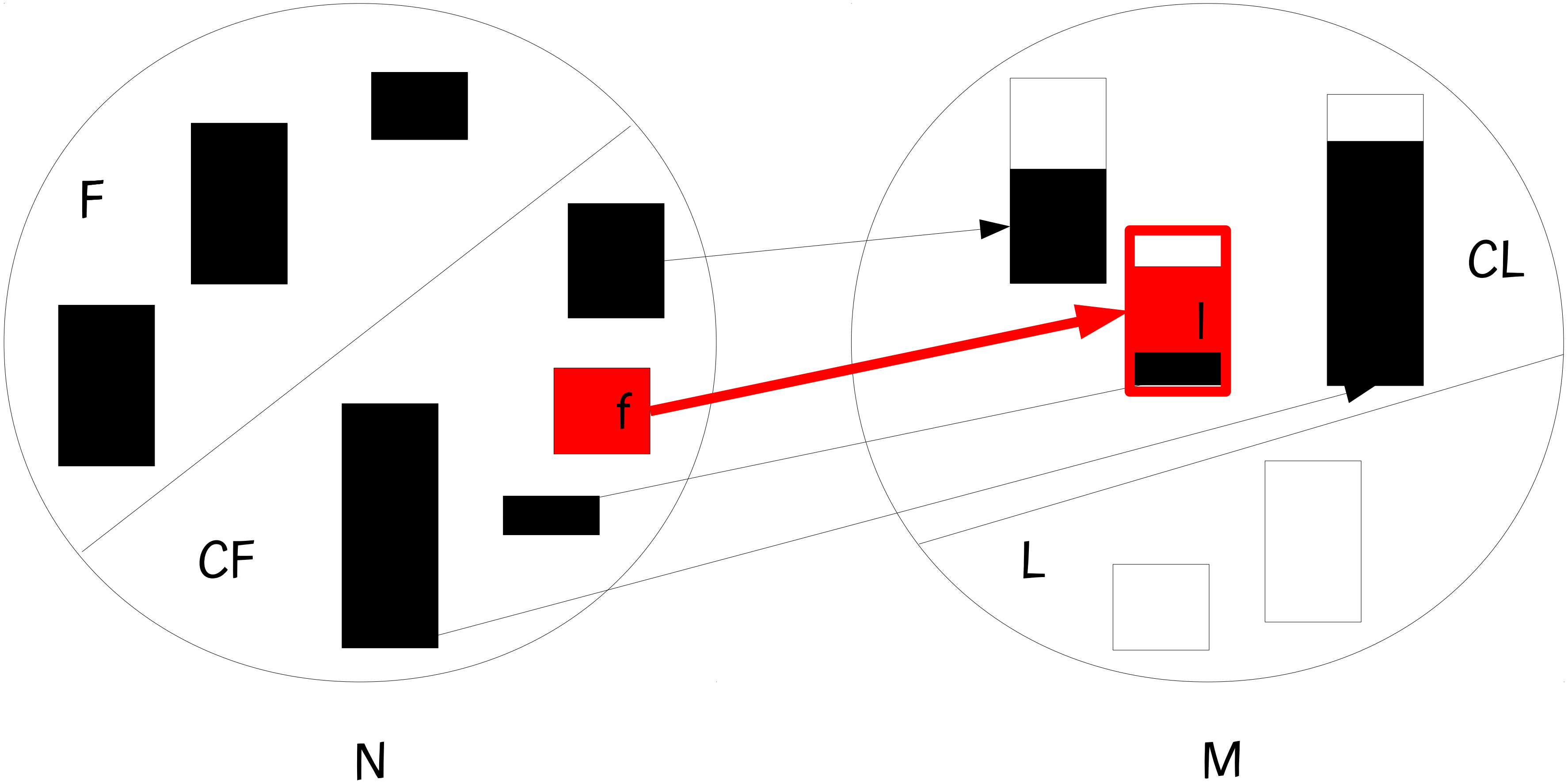
2. Randomly select a location l ∈ R favoring those having high available capacity and those close to high-capacity locations in CL;

Assignment procedure



Assign facility f to location l

Assignment procedure



Update sets F, CF, and slack of location l

Considerations about the construction procedure

The procedure is not guaranteed to produce a feasible solution.

To address this difficulty, the construction procedure is repeated a maximum number of times or until all facilities are assigned (i.e. until $F=\emptyset$).

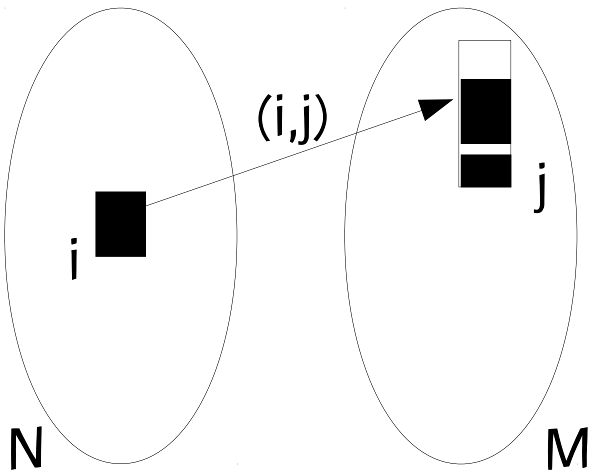
At start of construction, a location l is selected from the set L with probability proportional to its capacity. Location l is placed in CL.

Local search for GQAP

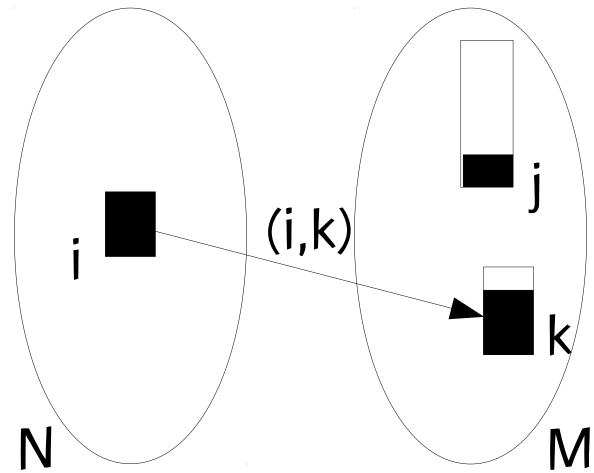
Local search

1-move and 2-move neighborhoods from solution p are used in our local search.

1-move: changing one facility-to-location assignment in p



solution p



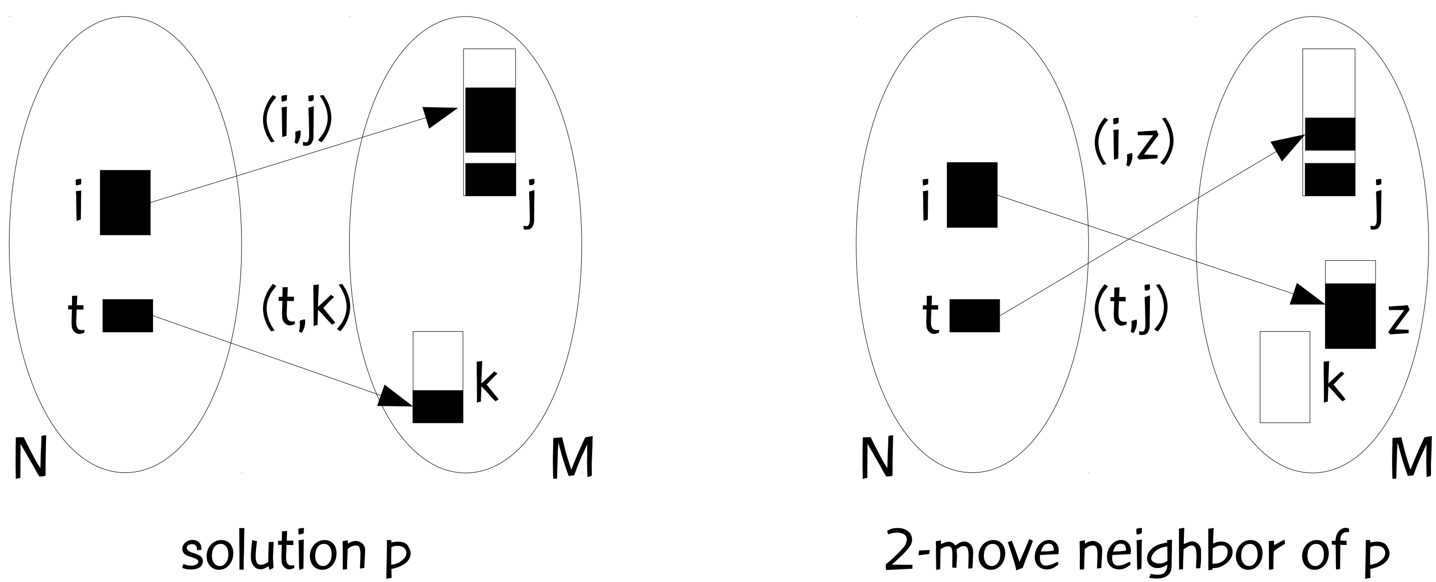
1-move neighbor of p

Local search

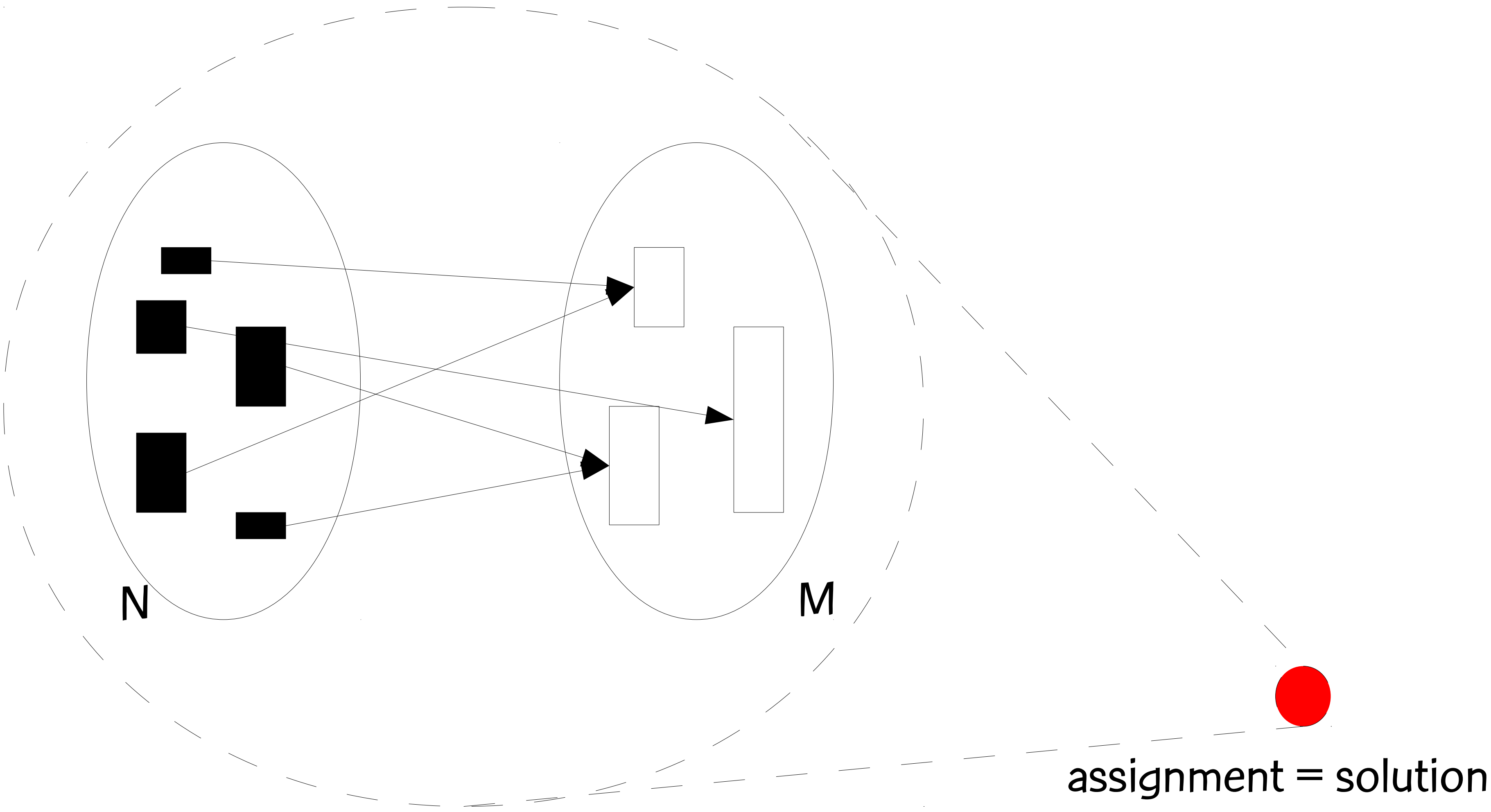
1-move and 2-move neighborhoods from solution p are used in our local search.

1-move: changing one facility-to-location assignment in p

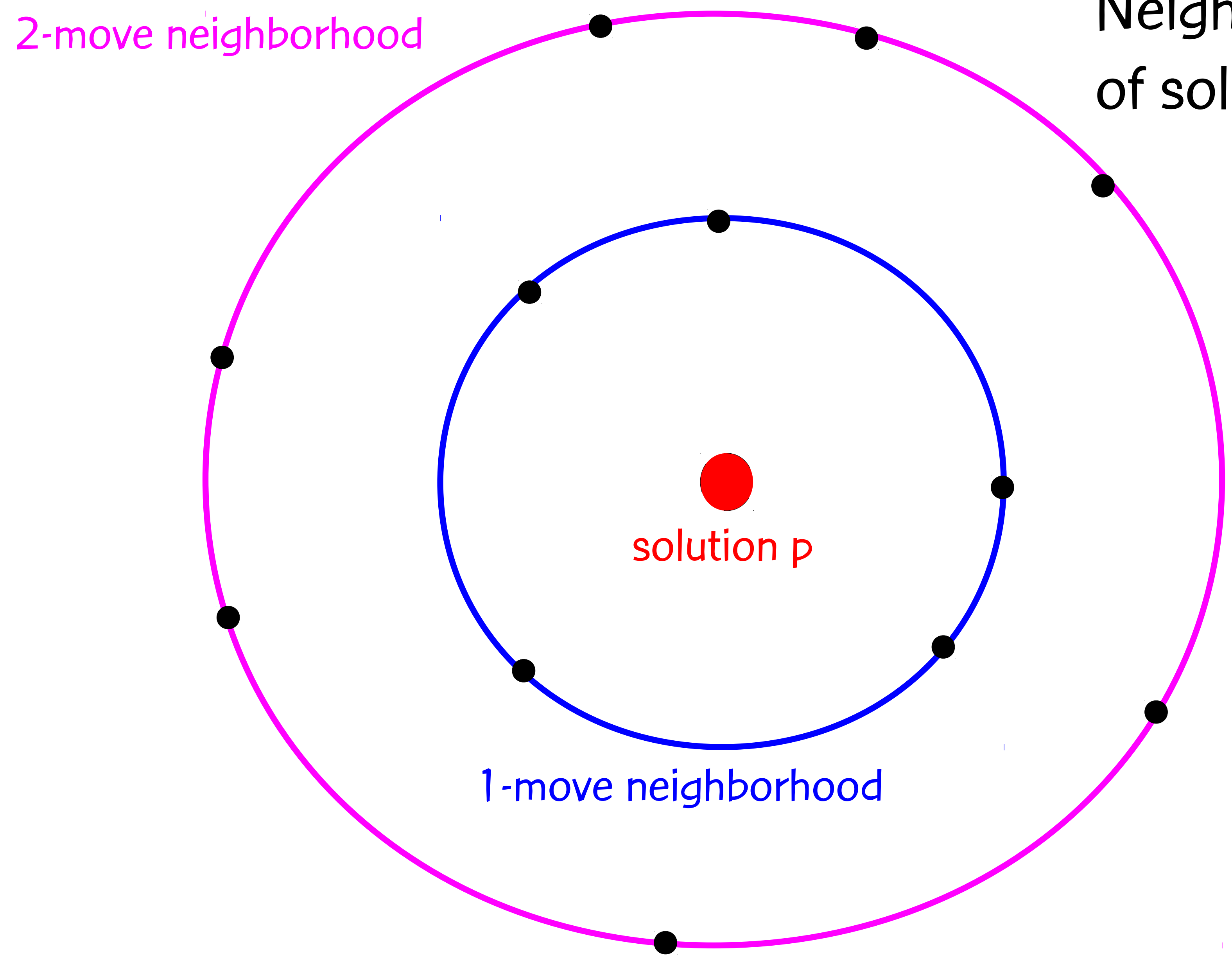
2-move: changing two facility-to-location assignment in p.



Assignment representation



Neighborhood of solution p



Traditional local search approaches

Best improving approach:

Evaluate all 1-move and 2-move neighborhood solutions and select the best improving solution

First improving approach:

1: From solution p, to evaluate its 1-move neighbors until the first improving solution q is found.

2: If q does not exist, continue search in the 2-move neighborhood.

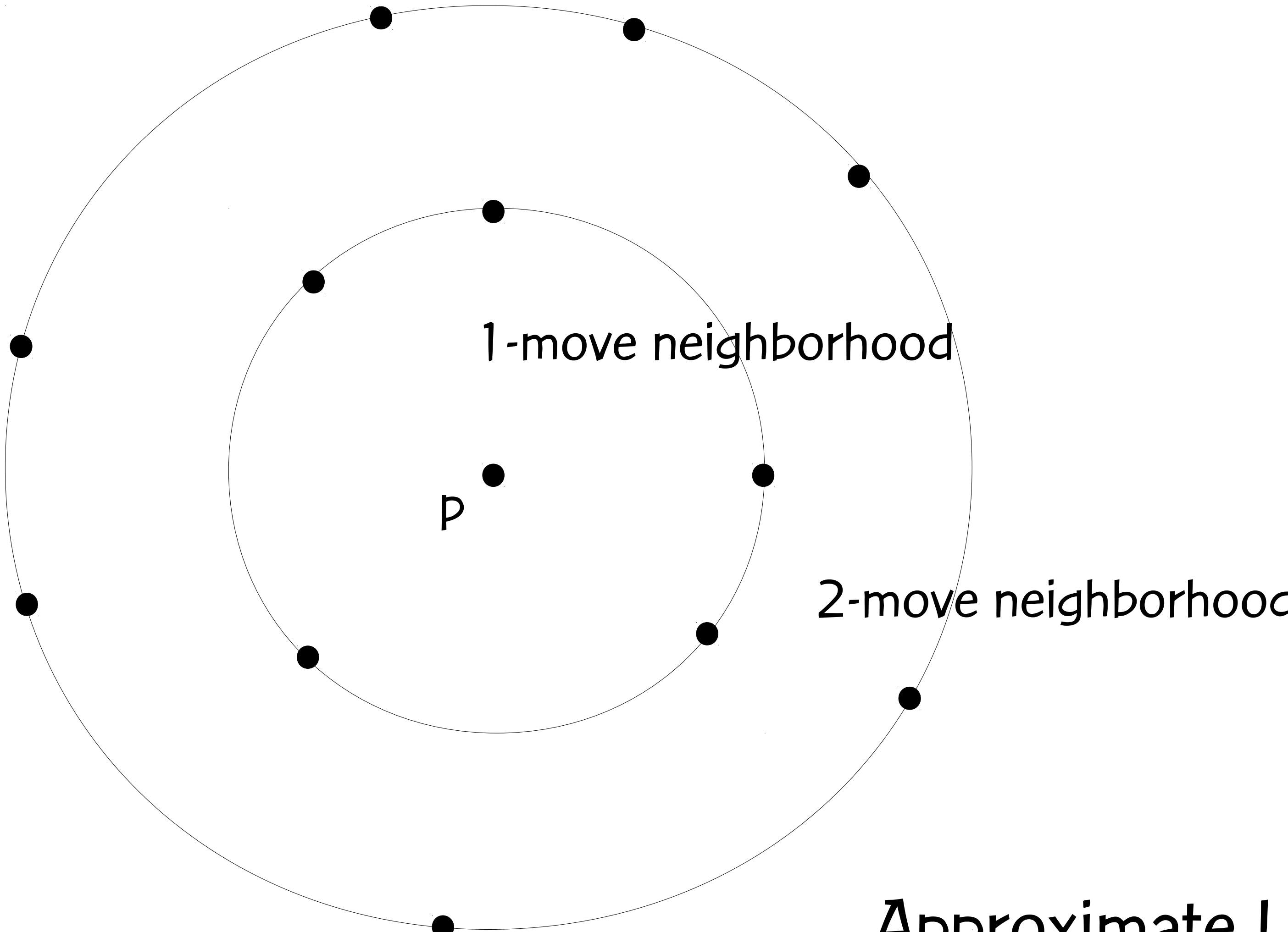
3: If q does not exist in the 2-move neighborhood, stop. Otherwise, assign $p = q$ and go to step 1.

Approximate local search

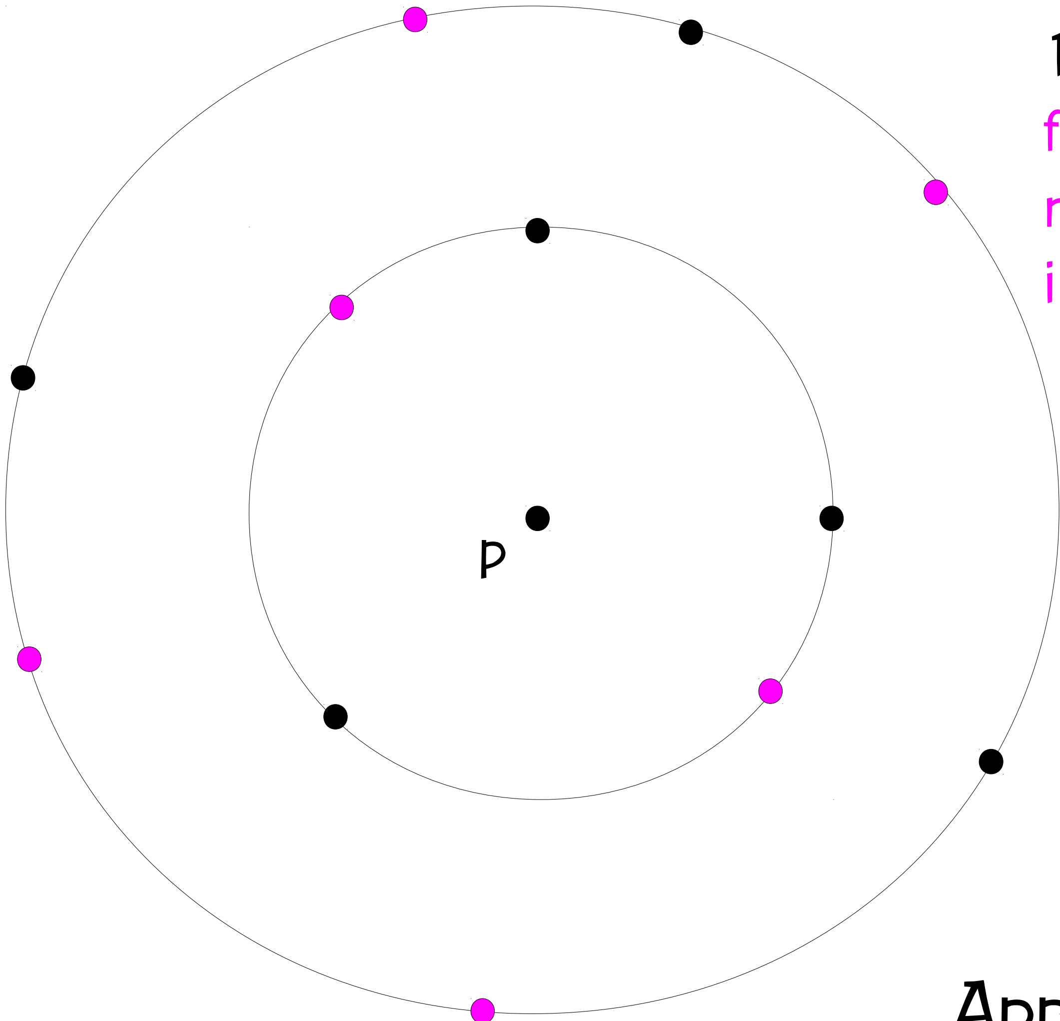
Tradeoff between best & first improvement: sample the neighborhood of solution p.

Neighborhoods can be very large for best improvement

Local search can take very long

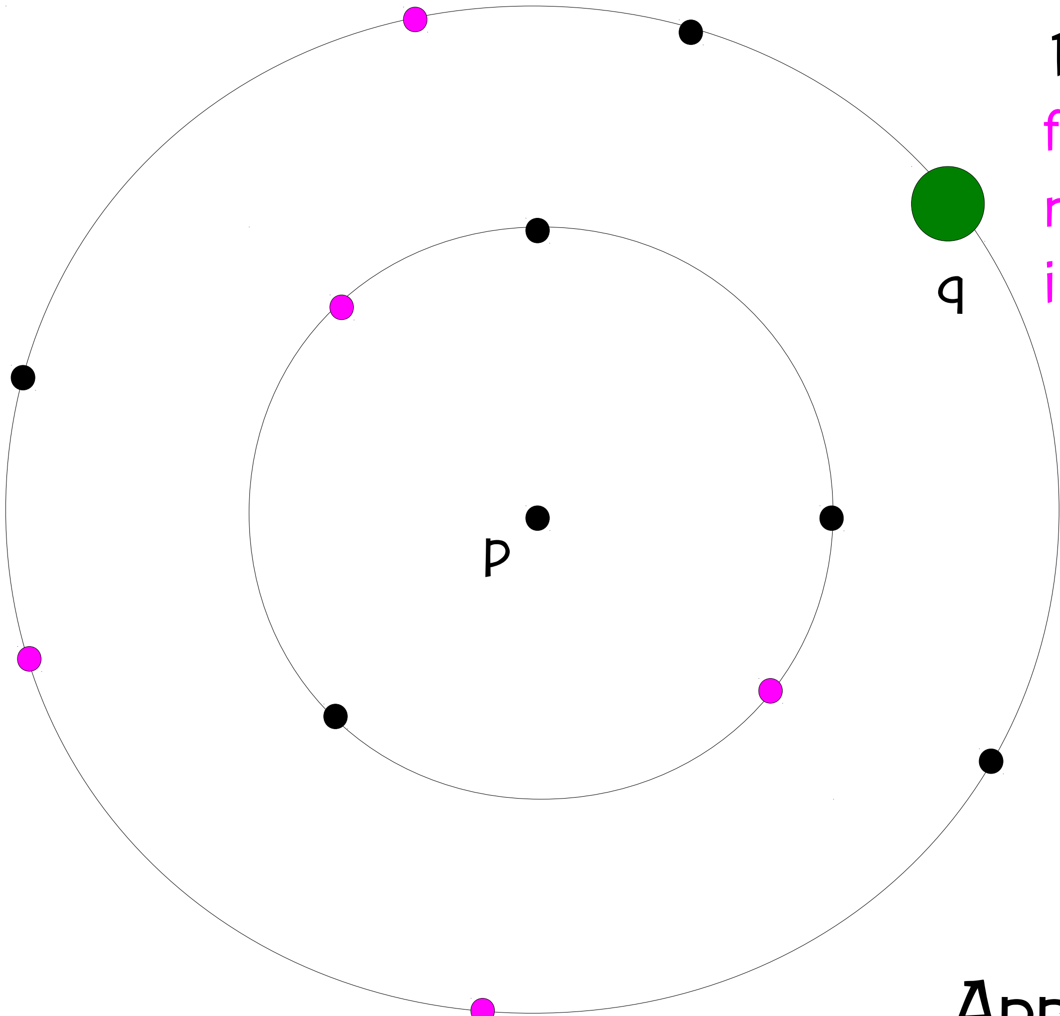


Approximate Local Search



1. Sample k improving solutions from 1-move and 2-move neighborhood of p and place them in an elite set E.

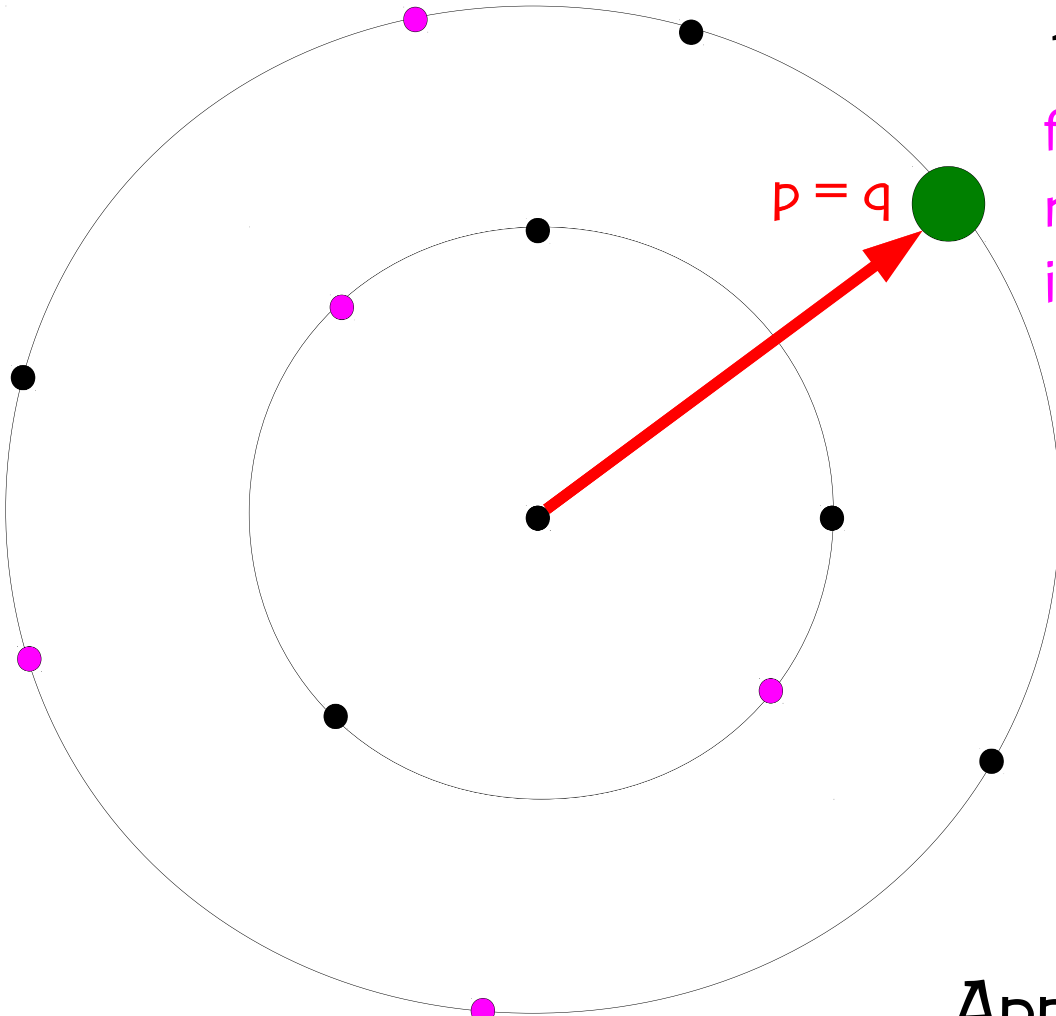
Approximate Local Search



1. Sample k improving solutions from 1-move and 2-move neighborhood of p and place them in an elite set E.

2. Select the best solution **q** from elite set E.

Approximate Local Search



1. Sample k improving solutions from 1-move and 2-move neighborhood of p and place them in an elite set E.

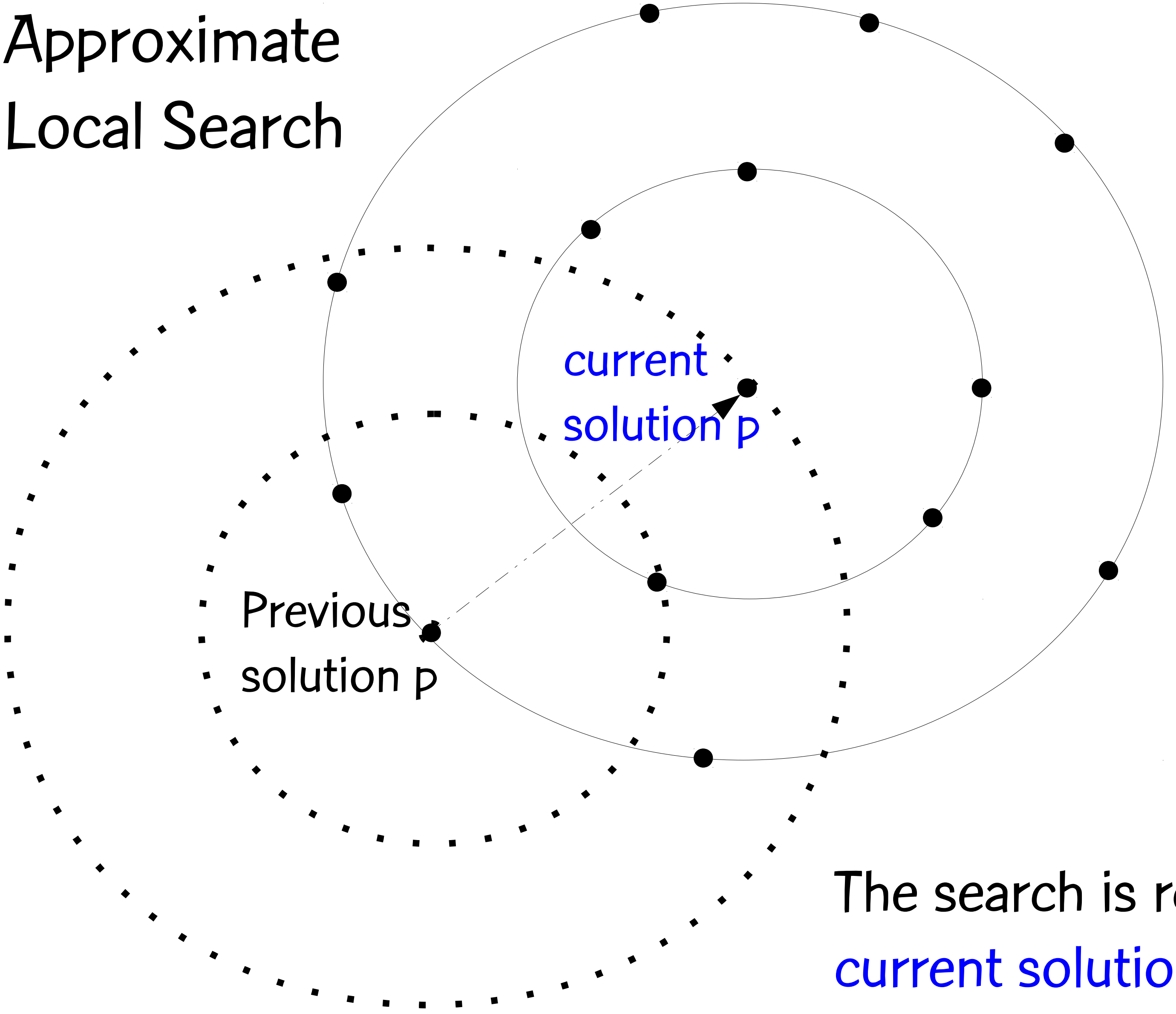
2. Select the best solution **q** from elite set E.

3. Update p = q

Approximate Local Search

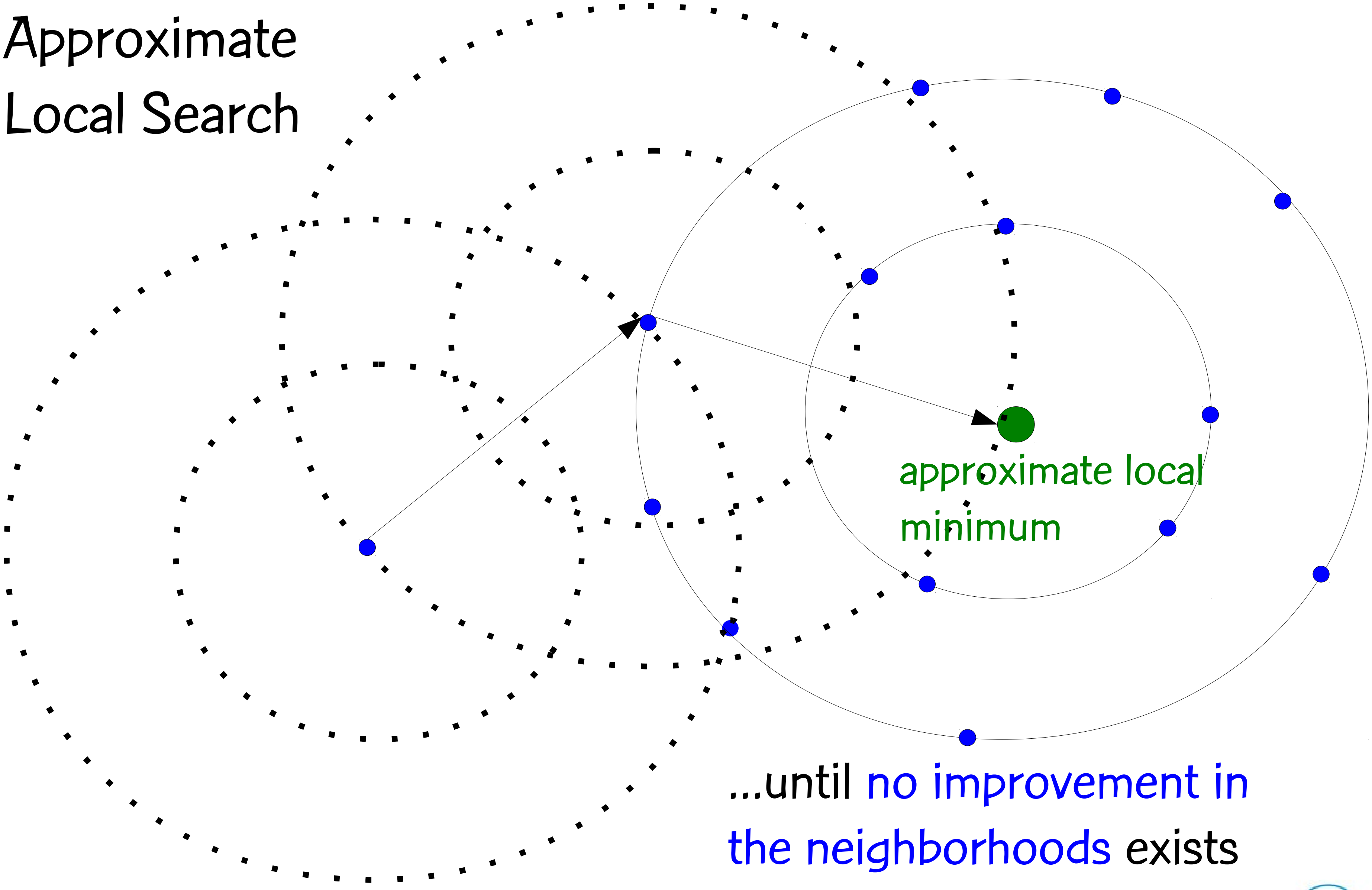
Your world. Delivered.

Approximate Local Search



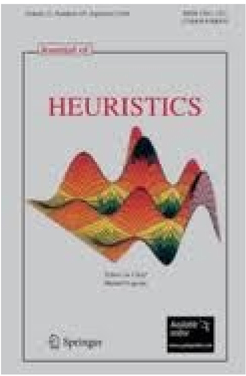
The search is repeated from current solution p until ….

Approximate Local Search



...until no improvement in the neighborhoods exists

Paper and java code



G.R. Mateus, R.M.A. Silva, and M.G.C. Resende, GRASP with path-relinking for the generalized quadratic assignment problem, J. of Heuristics 17 (527-565) 2011

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

We developed a Java implementation of the algorithm.

Handover minimization is a special case of the GQAP

- Towers $\leftrightarrow$ Facilities
  - tower traffic is facility demand
- RNCs $\leftrightarrow$ Locations
  - RNC capacity is Location capacity
- Handovers between towers $\leftrightarrow$ Flows between facilities
- Distance between each pair of RNC = 1

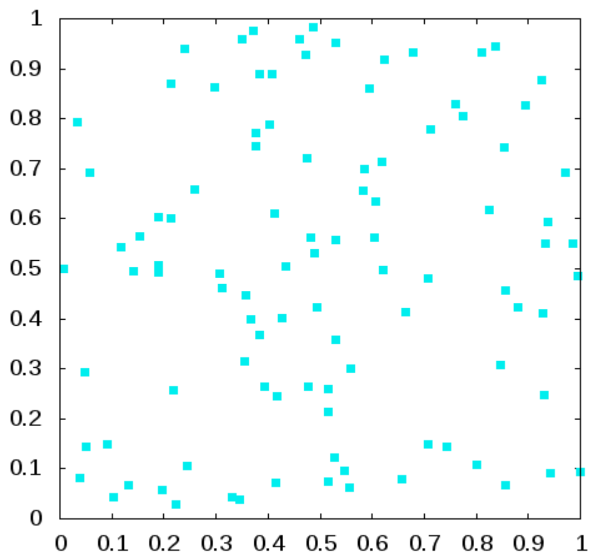
Experiments with GRASP with PR for GQAP

Random instance generator

- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.

Random instance generator

- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.
- Generate T random points (towers) on the unit square.



Random instance generator

- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.
- Generate T random points (towers) on the unit square.
- For each tower i, traffic[i] = randunif(l(t), u(t))

Random instance generator

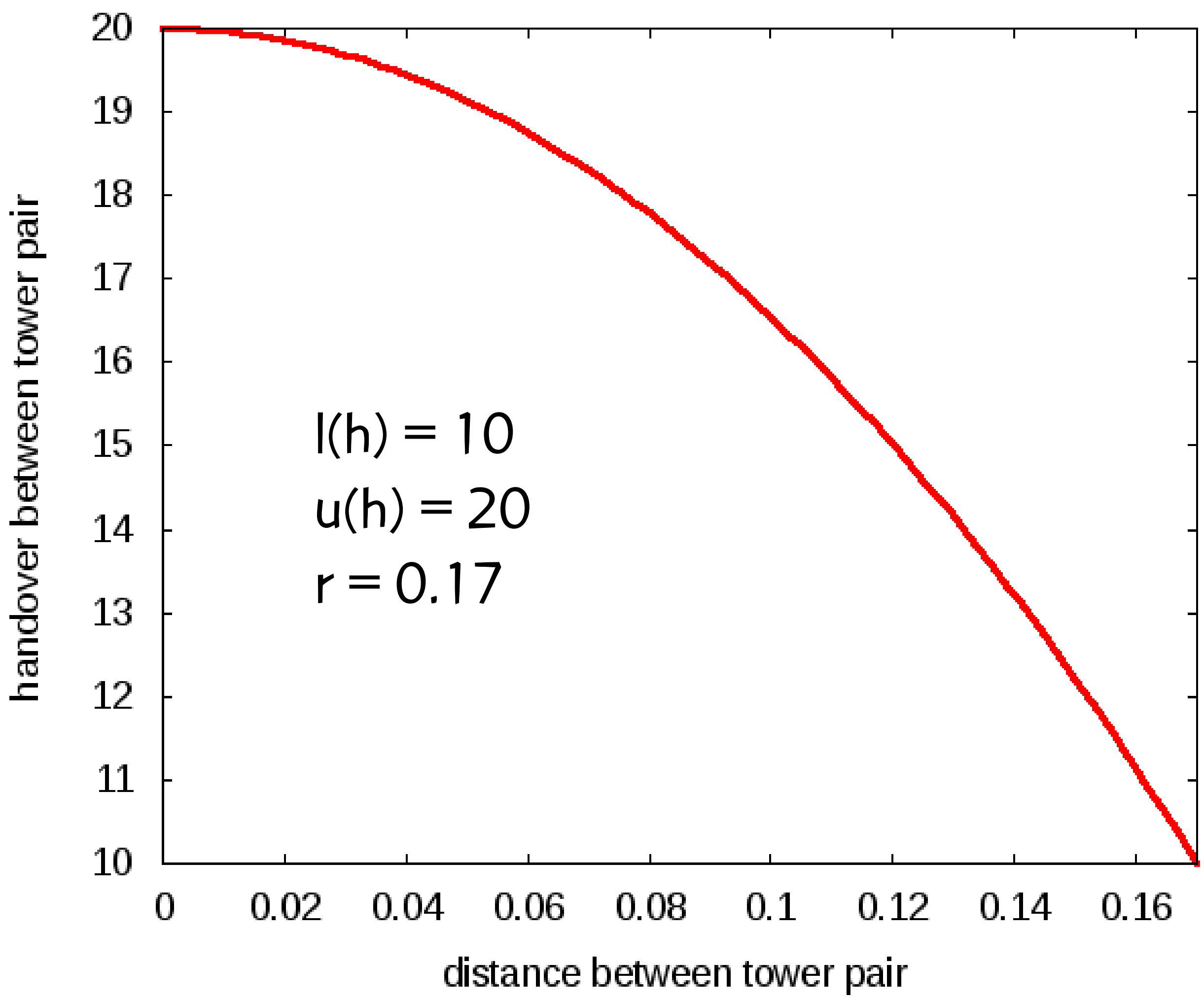
- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.
- Generate T random points (towers) on the unit square.
- For each tower i, traffic[i] = randunif(l(t), u(t))
- avg-traffic is sum of traffic[i]/T over all towers

Random instance generator

- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.
- Generate T random points (towers) on the unit square.
- For each tower i, traffic[i] = randunif(l(t), u(t))
- avg-traffic is sum of traffic[i]/T over all towers
- For each pair of towers {i, j}, if dist(i,j) < r, then handover[i,j] = $[l(h) - u(h)]/r^2 \times d^2 + u(h)$

For each pair of towers $\{i, j\}$, if $dist(i,j) < r$, then

$$handover[i,j] = [l(h) - u(h)]/r^2 \times d^2 + u(h)$$



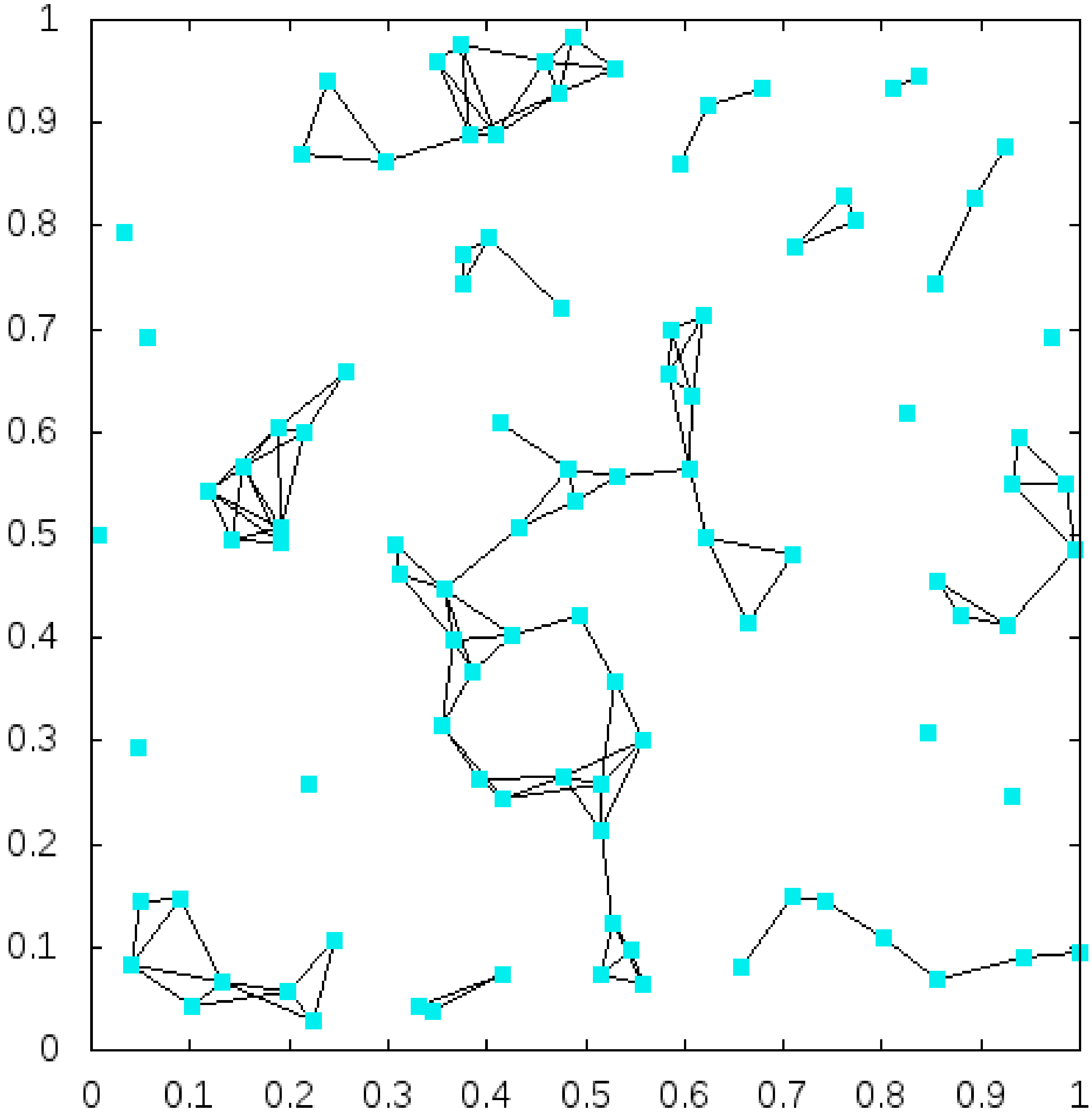
Random instance generator

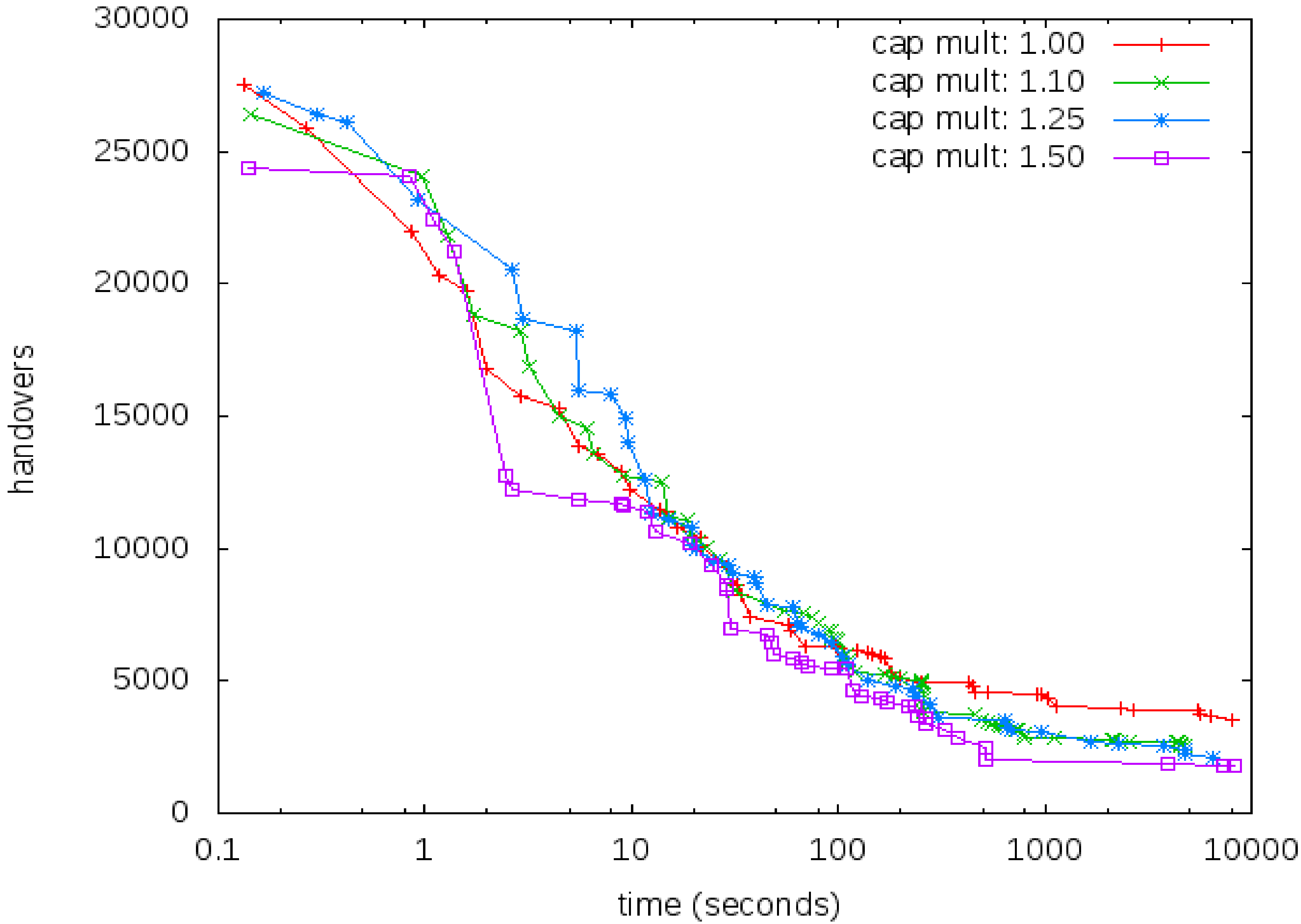
- Input T (number of towers), R (number of RNCs), r (max handover distance), and lower and upper bounds on traffic, handover, and capacity slack.
- Generate T random points (towers) on the unit square.
- For each tower i, traffic[i] = randunif(l(t), u(t))
- avg-traffic is sum of traffic[i]/T over all towers
- For each pair of towers {i, j}, if dist(i,j) < r, then handover[i,j] = $[l(h) - u(h)]/r^2 \times d^2 + u(h)$
- For each RNC j, capacity[j] = randunif(l(c), u(c)) * avg-traffic, where u(c) > l(c) > 1.

Three synthetic instances for experiments with GRASP with PR for GQAP

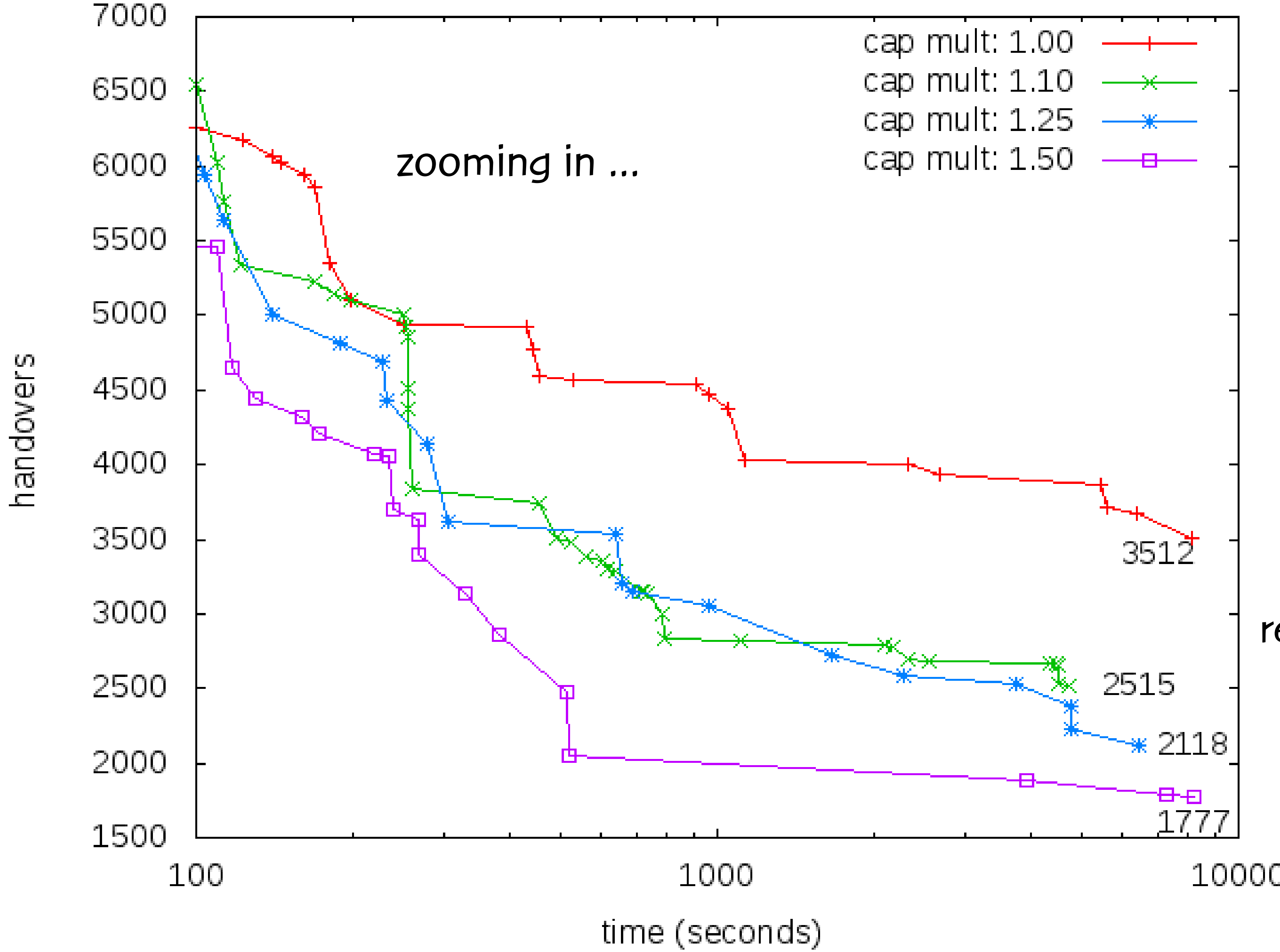
- Number of towers: 100
- Number of RNCs: two instances with 15 and one with 15, 17, 19, 21, 23, 25, 27, and 29
- Tower traffic bounds: [5, 50]
- Handover bounds: [5, 200]
- RNC capacity slack bounds: [1.05, 1.15]
- Three values of max handover distance: 0.1, 0.17, and 0.25

15 RNCs
100 towers
r = 0.1





cap mult: 1.00
cap mult: 1.10
cap mult: 1.25
cap mult: 1.50
handovers
time (seconds)
30000
25000
20000
15000
10000
5000
0
0.1
1
10
100
1000
10000



reductions

28%

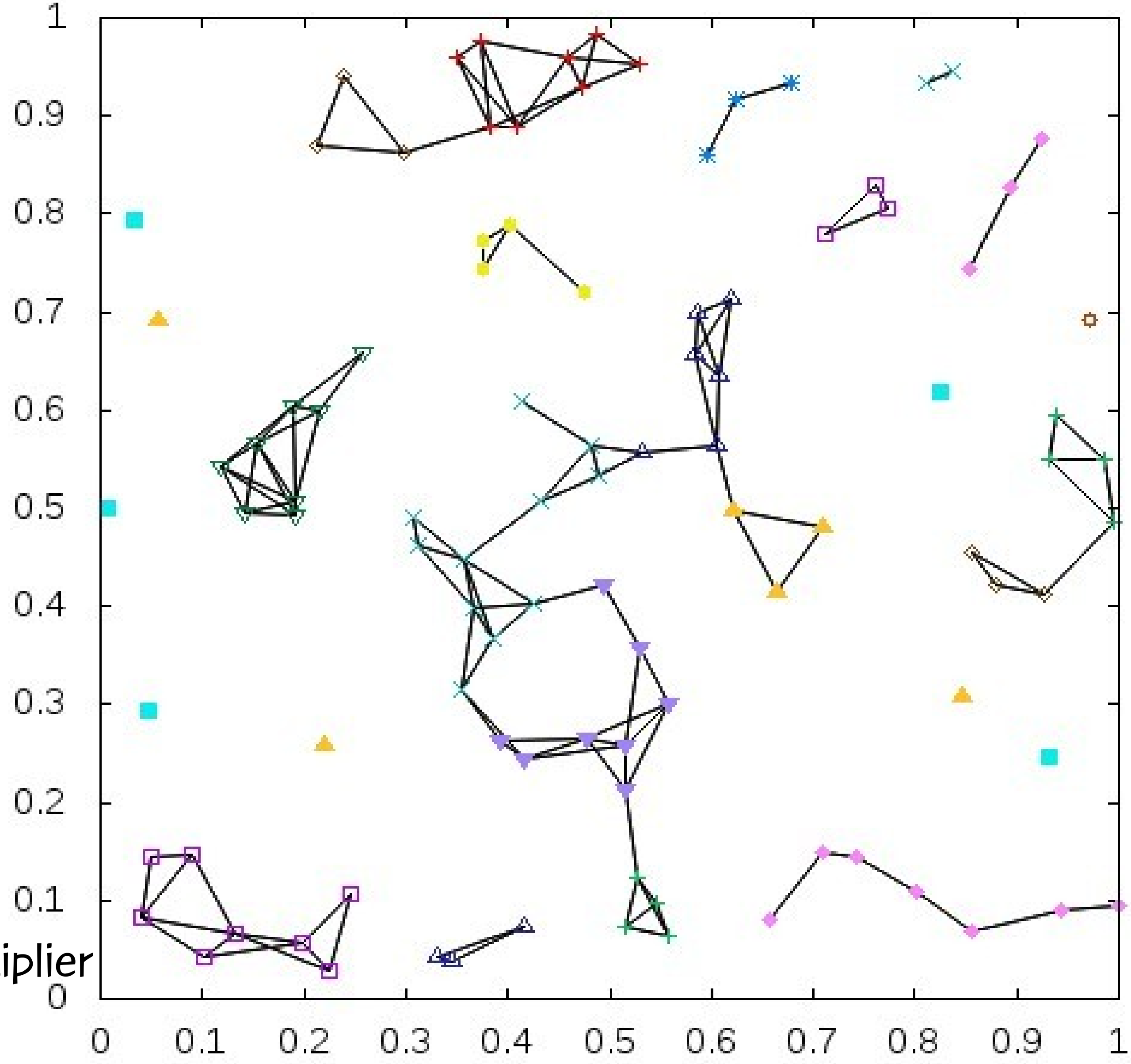
40%

49%

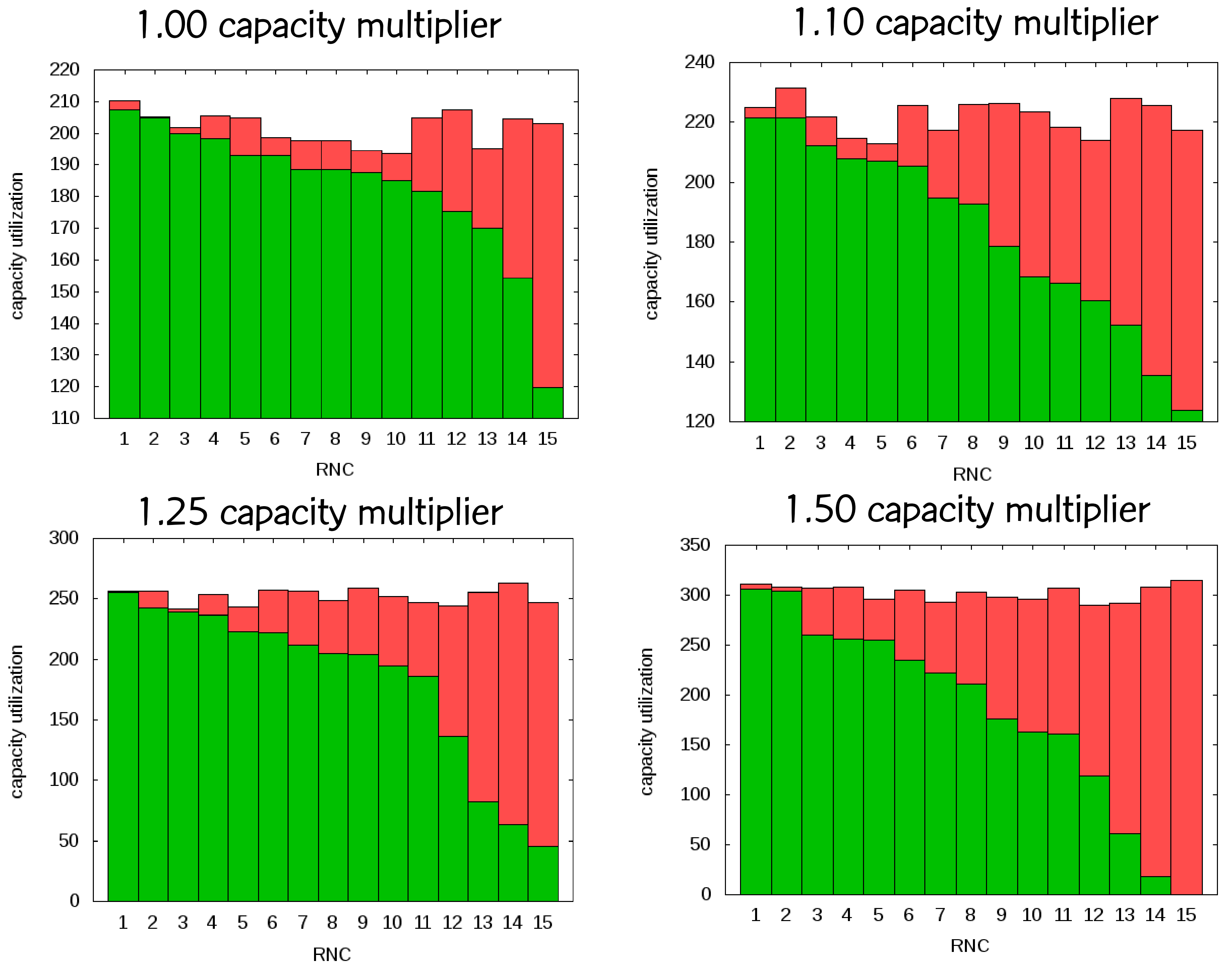
at&t Your world. Delivered.

15 RNCs
100 towers
r = 0.1

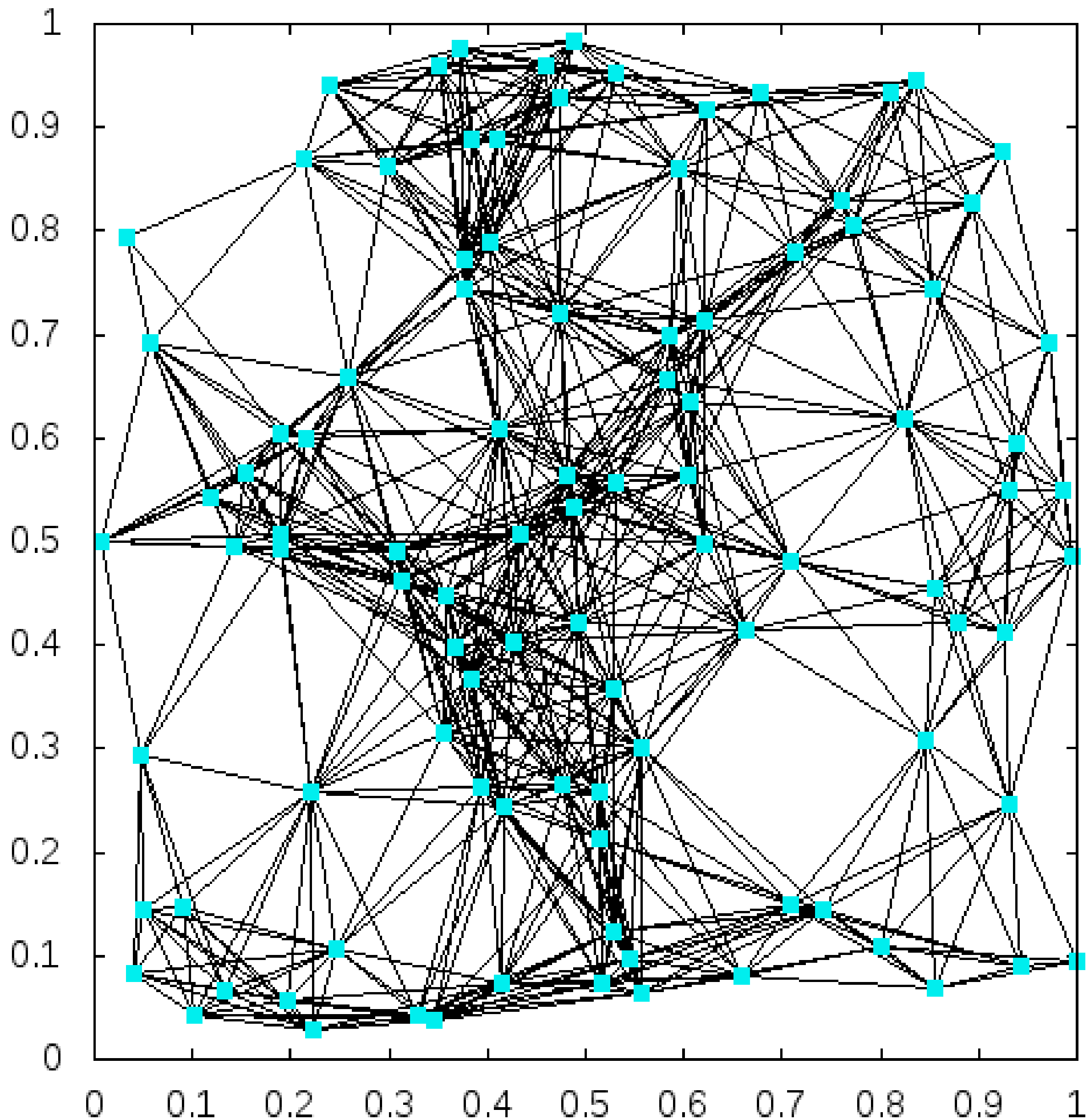
Solution with 1777 handovers

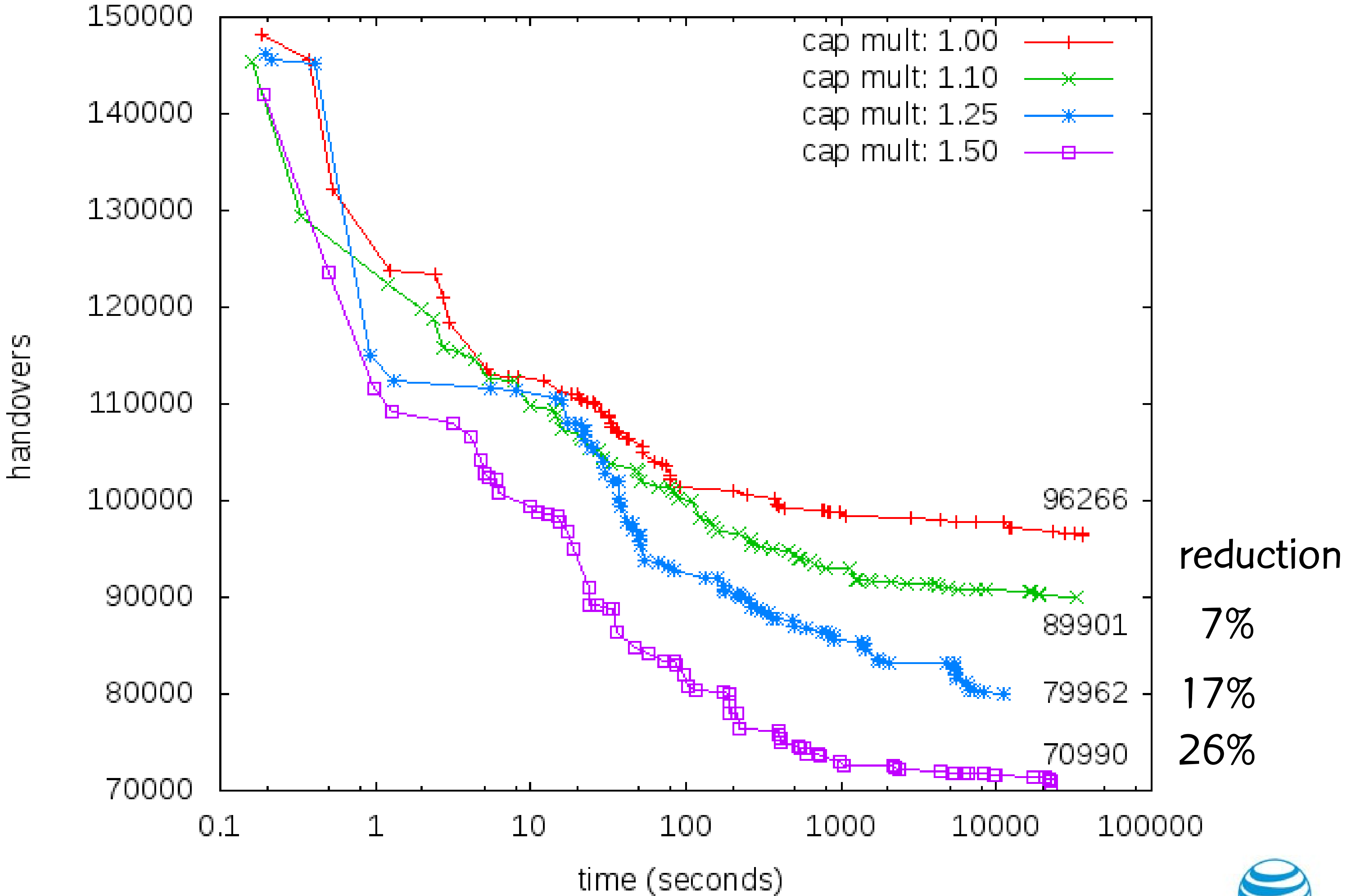


1.50 capacity multiplier


1.00 capacity multiplier
1.10 capacity multiplier
1.25 capacity multiplier
1.50 capacity multiplier
capacity utilization
RNC
220
210
200
190
180
170
160
150
140
130
120
110
240
300
250
100
50
0
350
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

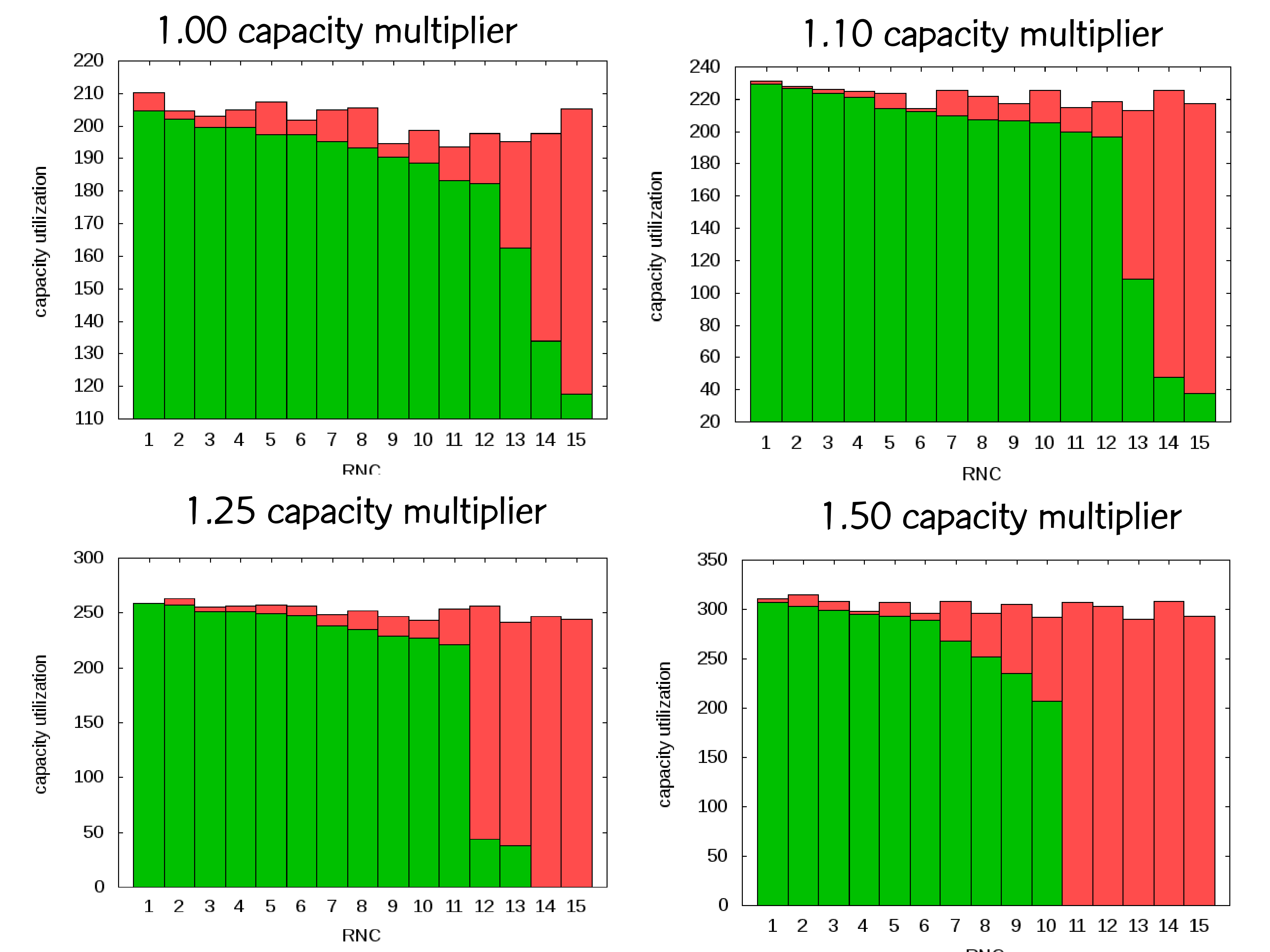
15 RNCs
100 towers
r = 0.25





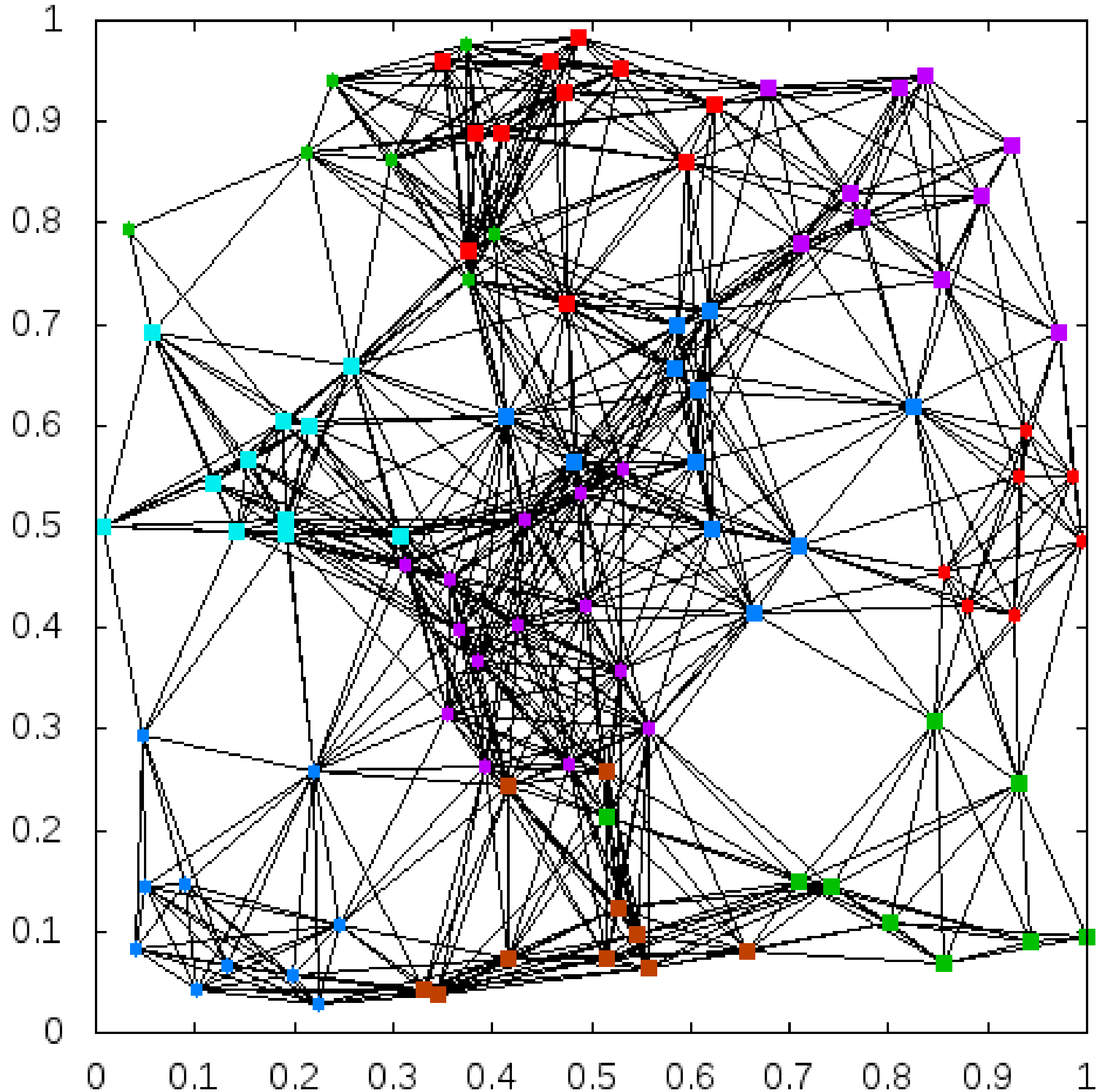
at&t

Your world. Delivered.


1.00 capacity multiplier
capacity utilization
RNC
1.10 capacity multiplier
capacity utilization
RNC
1.25 capacity multiplier
capacity utilization
RNC
1.50 capacity multiplier
capacity utilization

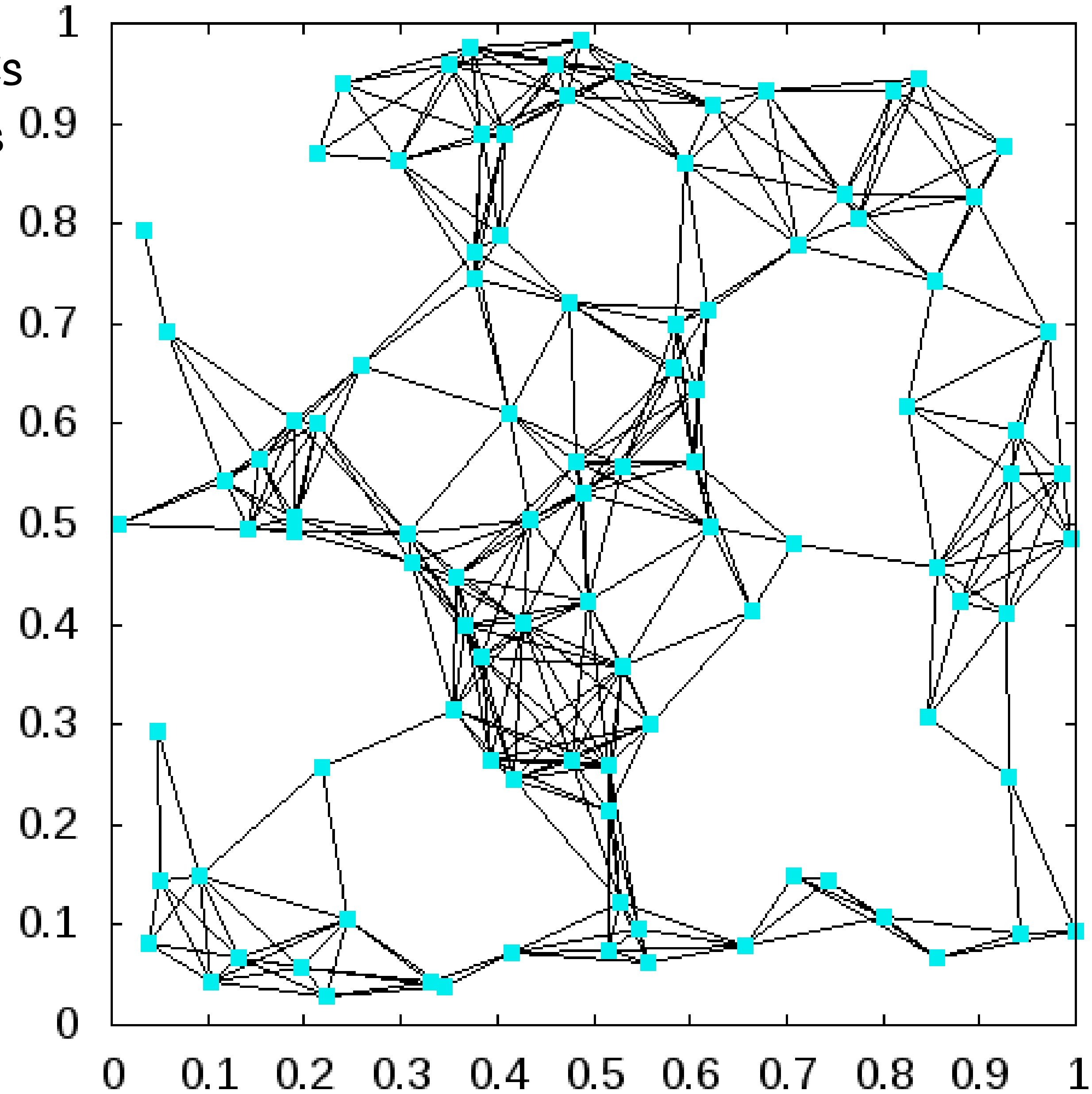
15 RNCs
100 towers
r = 0.25

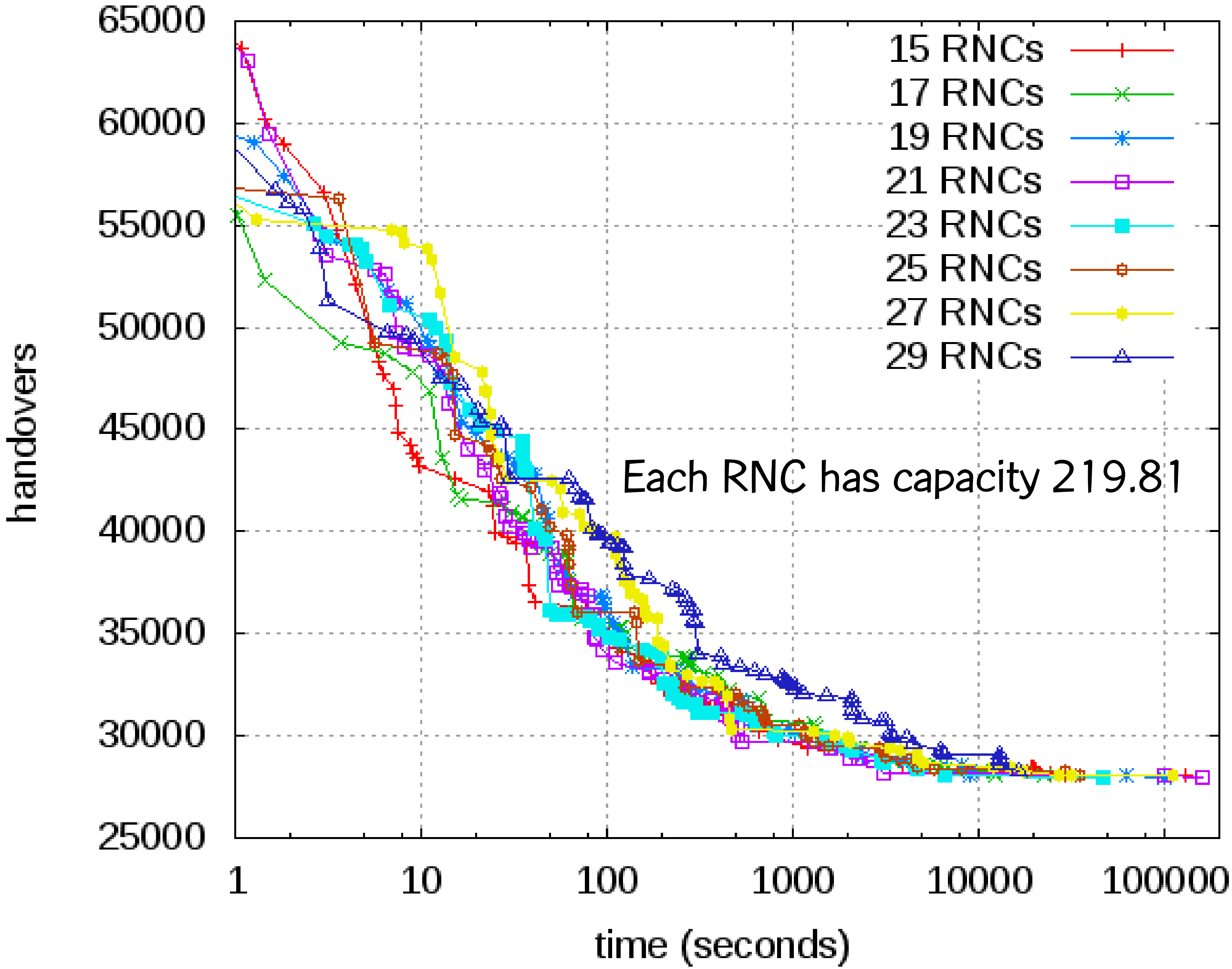
Solution with 70990 handovers

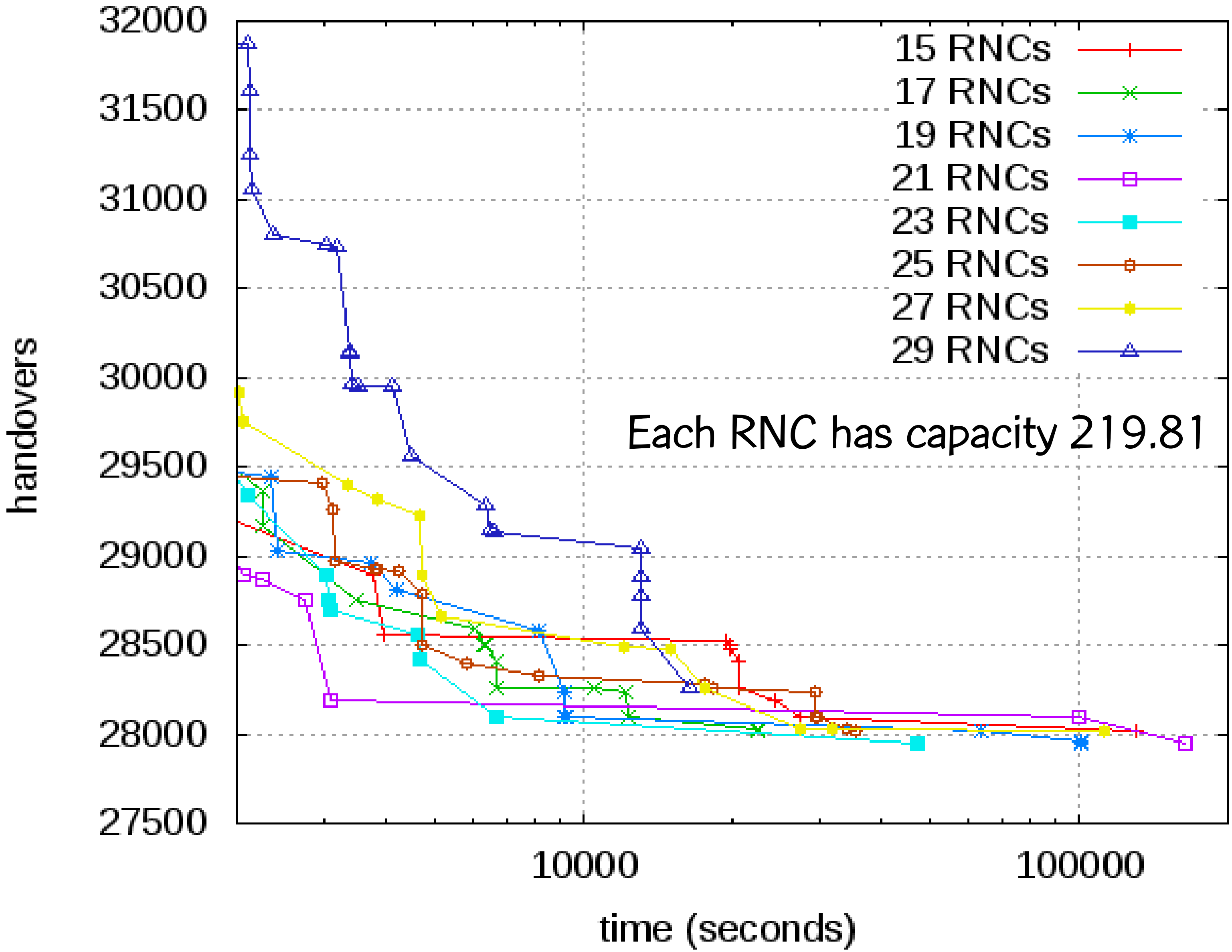


1.50 capacity multiplier

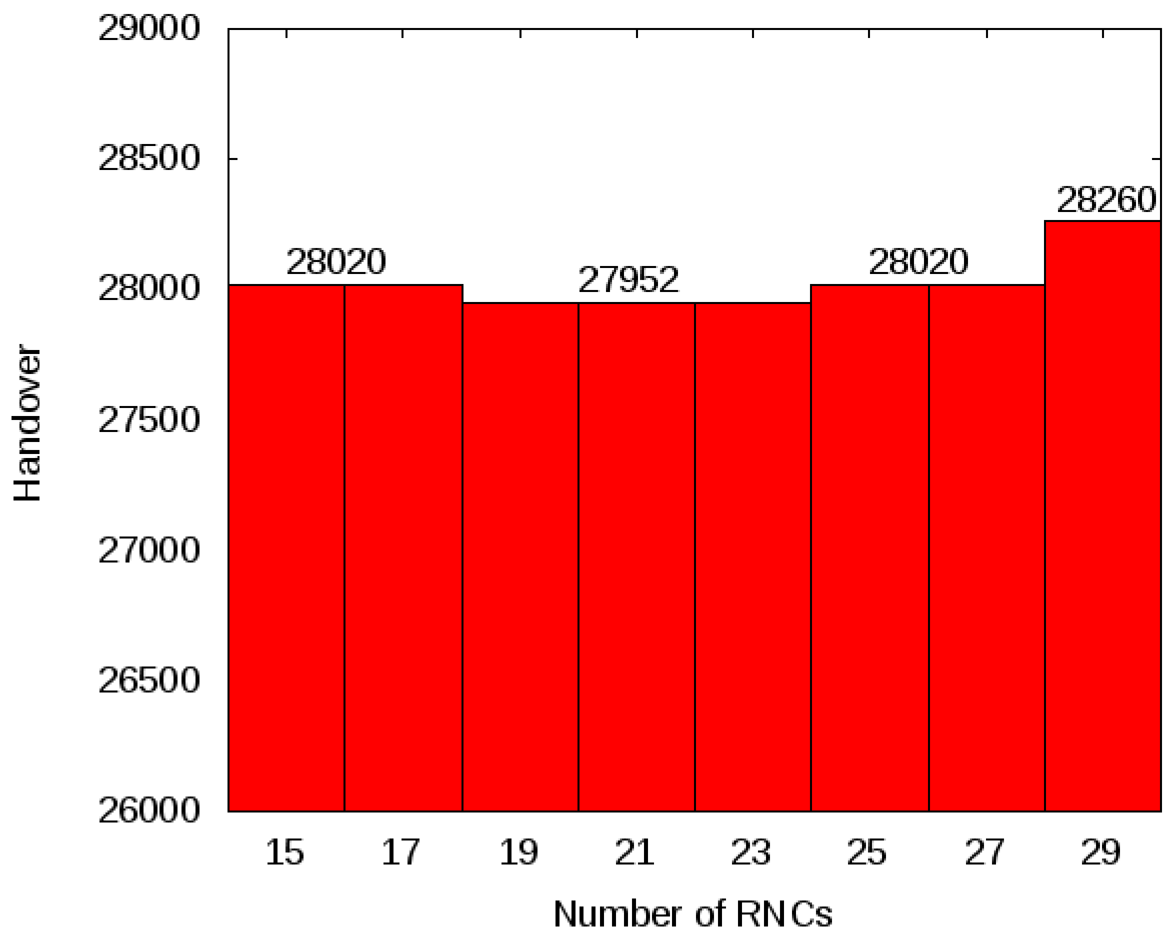
15-29 RNCs
100 towers
r = 0.17



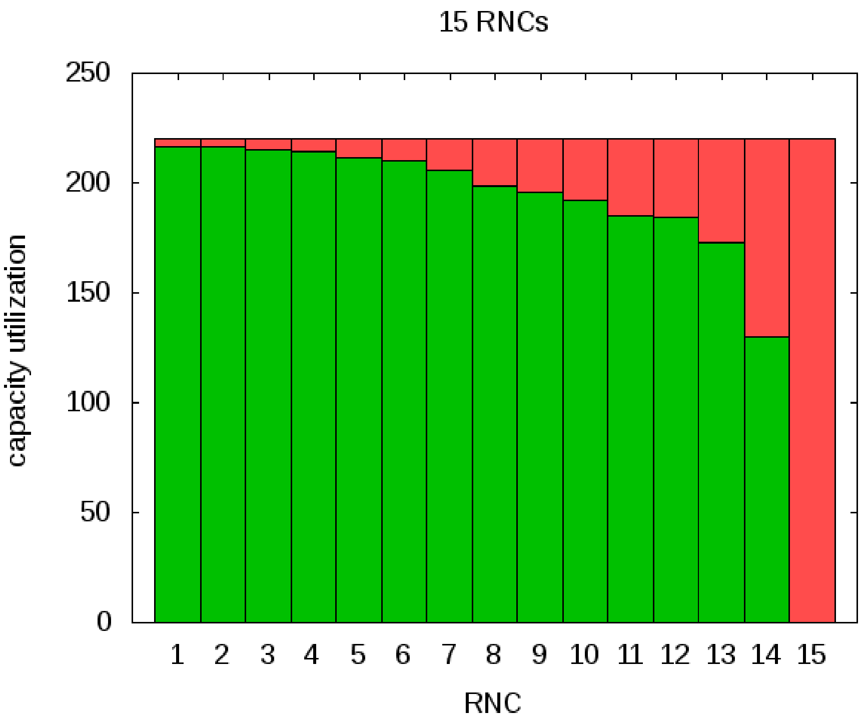

65000
60000
55000
50000
45000
40000
35000
30000
25000
handovers
1
10
100
1000
10000
100000
time (seconds)
15 RNCs
17 RNCs
19 RNCs
21 RNCs
23 RNCs
25 RNCs
27 RNCs
29 RNCs
Each RNC has capacity 219.81

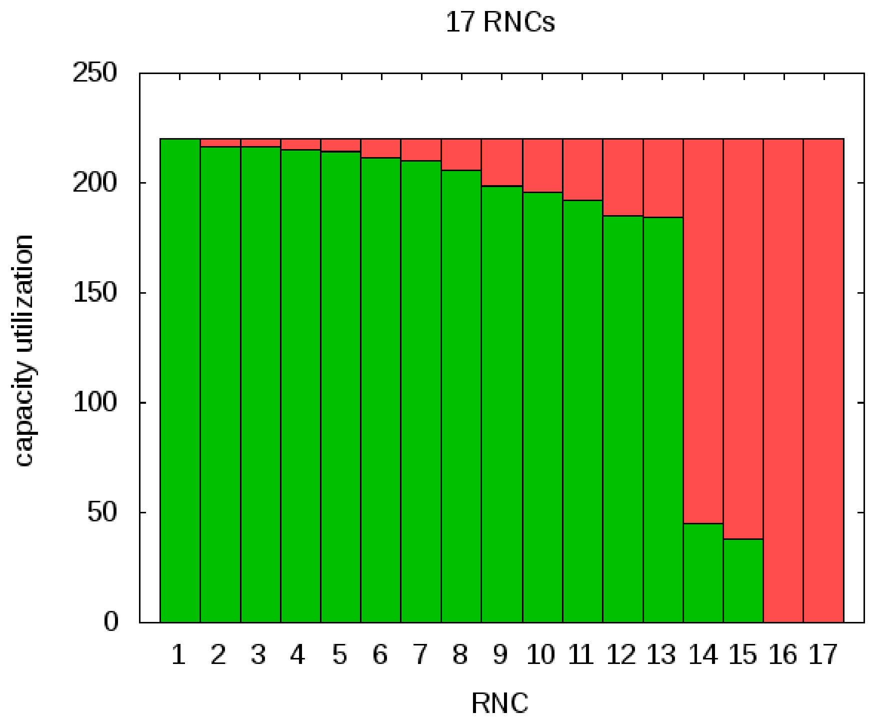


32000
31500
31000
30500
30000
29500
29000
28500
28000
27500
handovers
10000
100000
time (seconds)
15 RNCs
17 RNCs
19 RNCs
21 RNCs
23 RNCs
25 RNCs
27 RNCs
29 RNCs
Each RNC has capacity 219.81

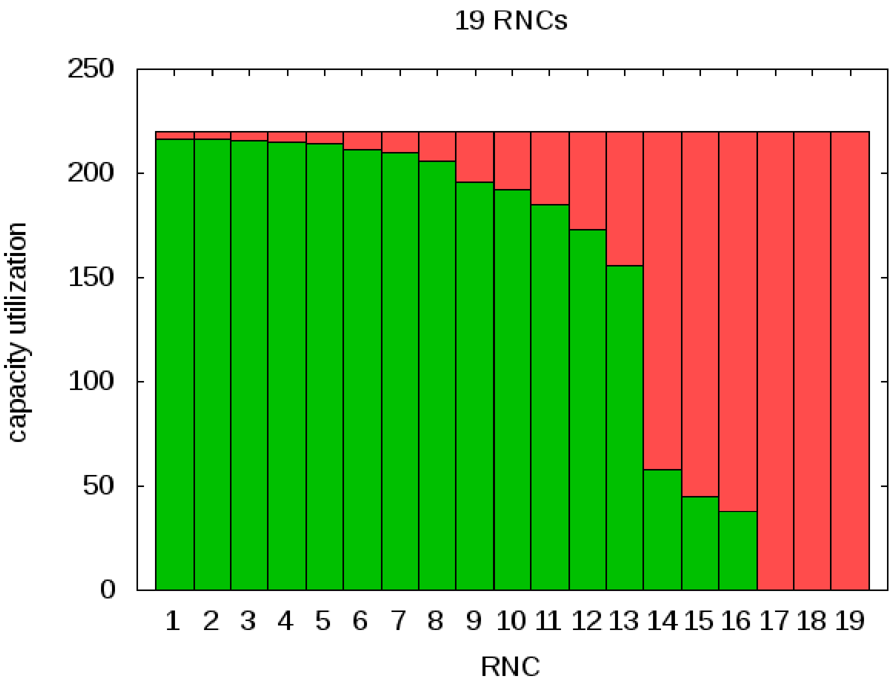


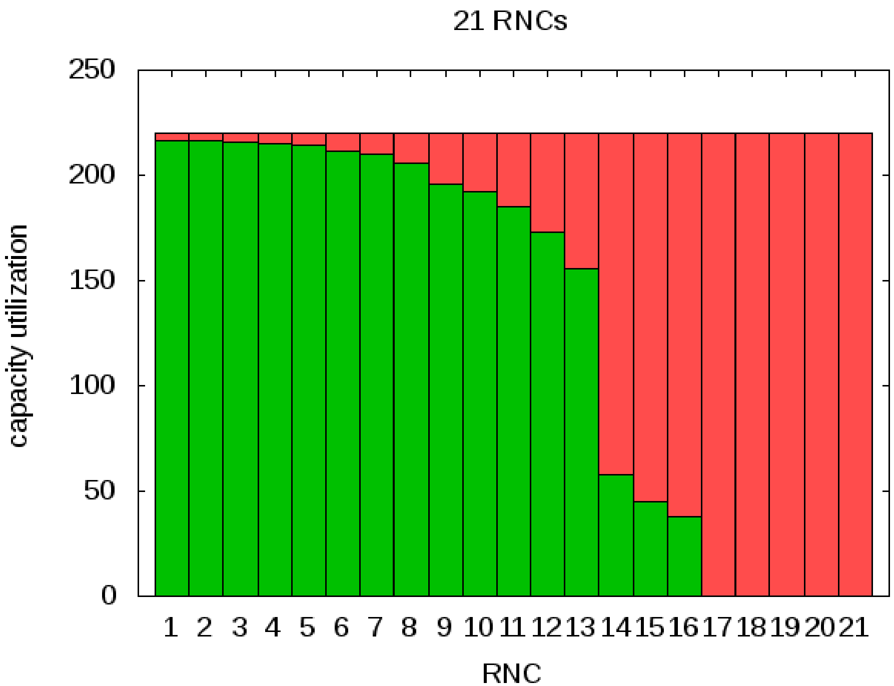
29000
28500
28000
27500
27000
26500
26000
Handover
28020
27952
28020
28260
15
17
19
21
23
25
27
29
Number of RNCs

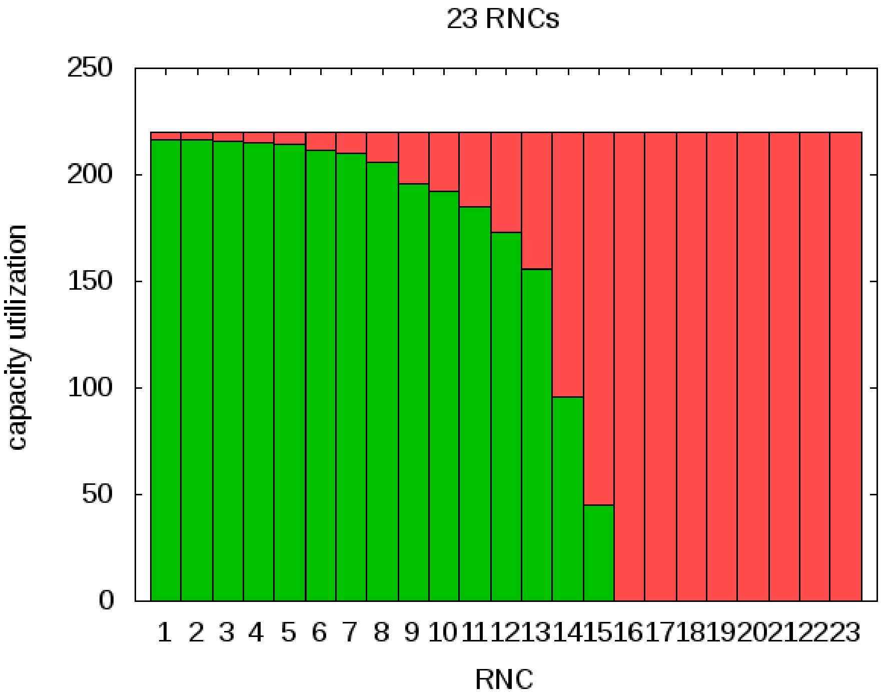




28020 handovers



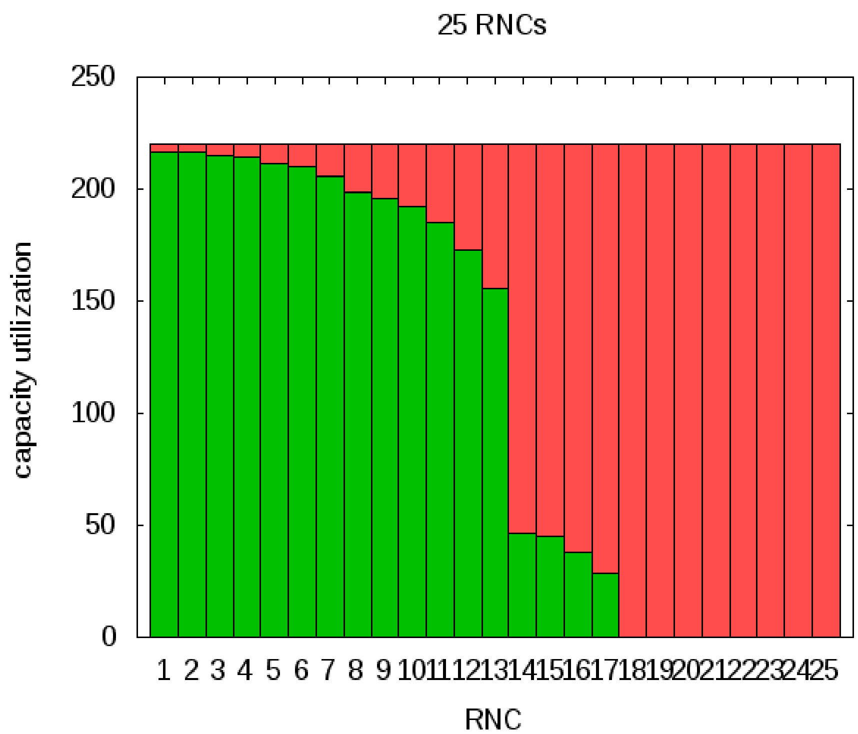


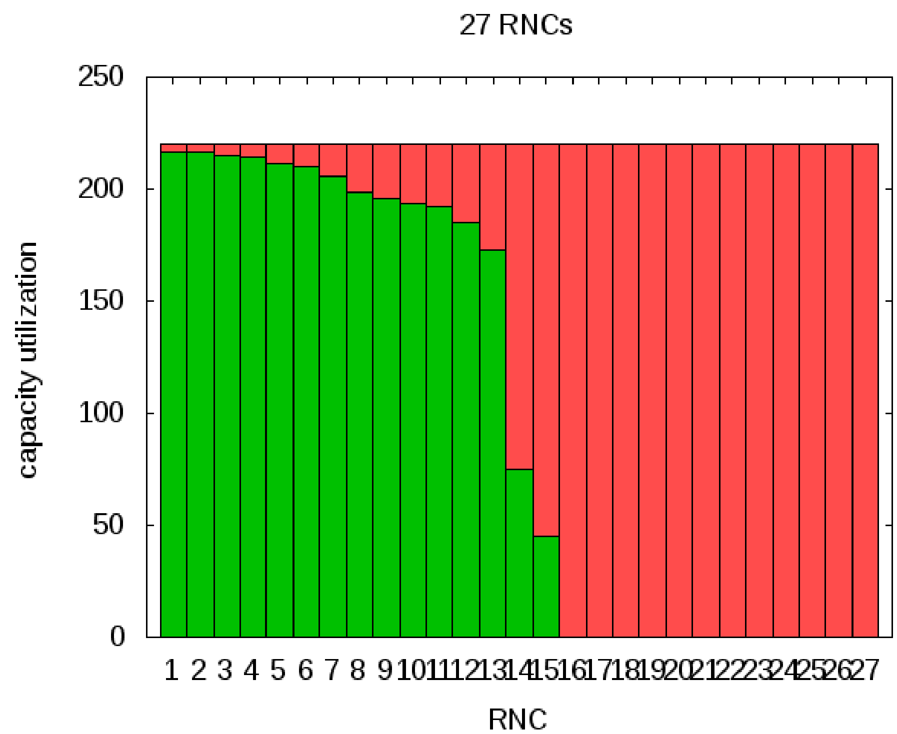


27952 handovers

at&t

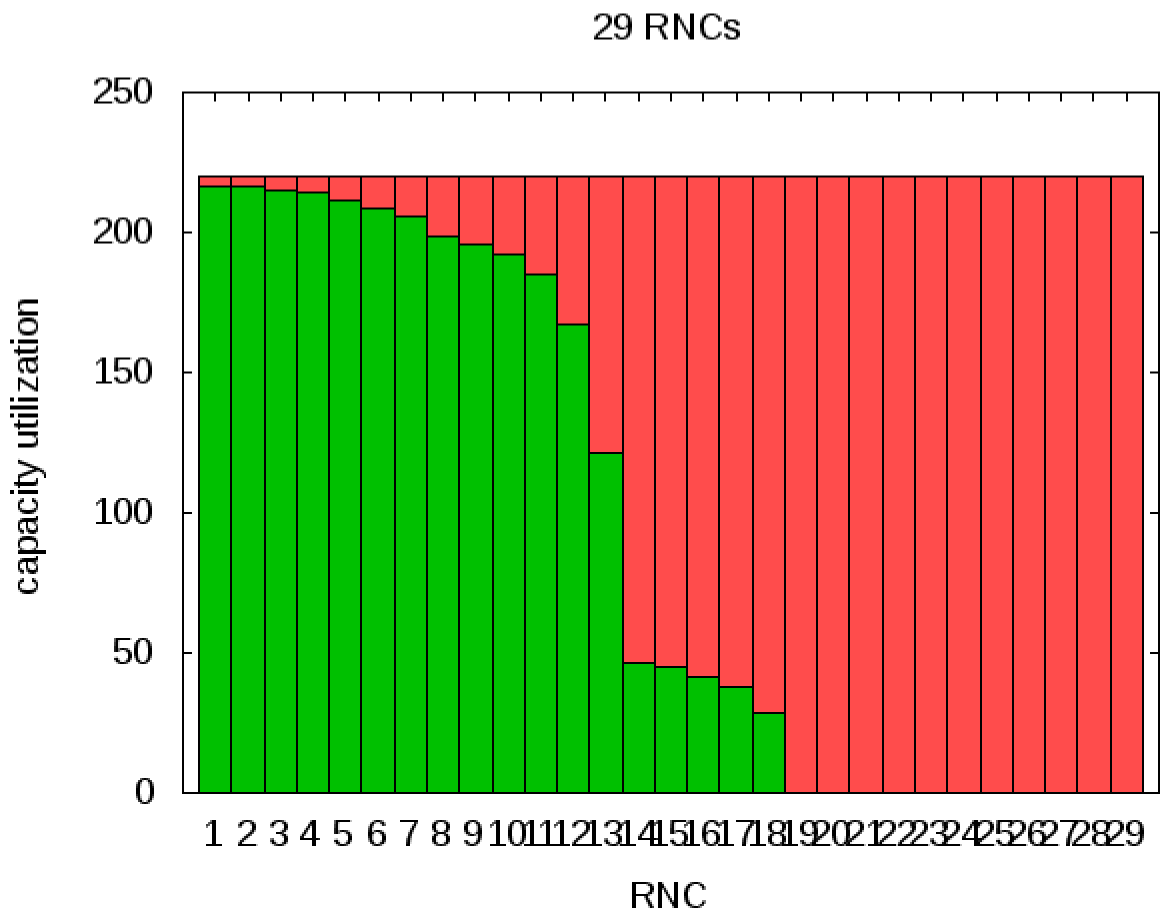
Your world. Delivered.



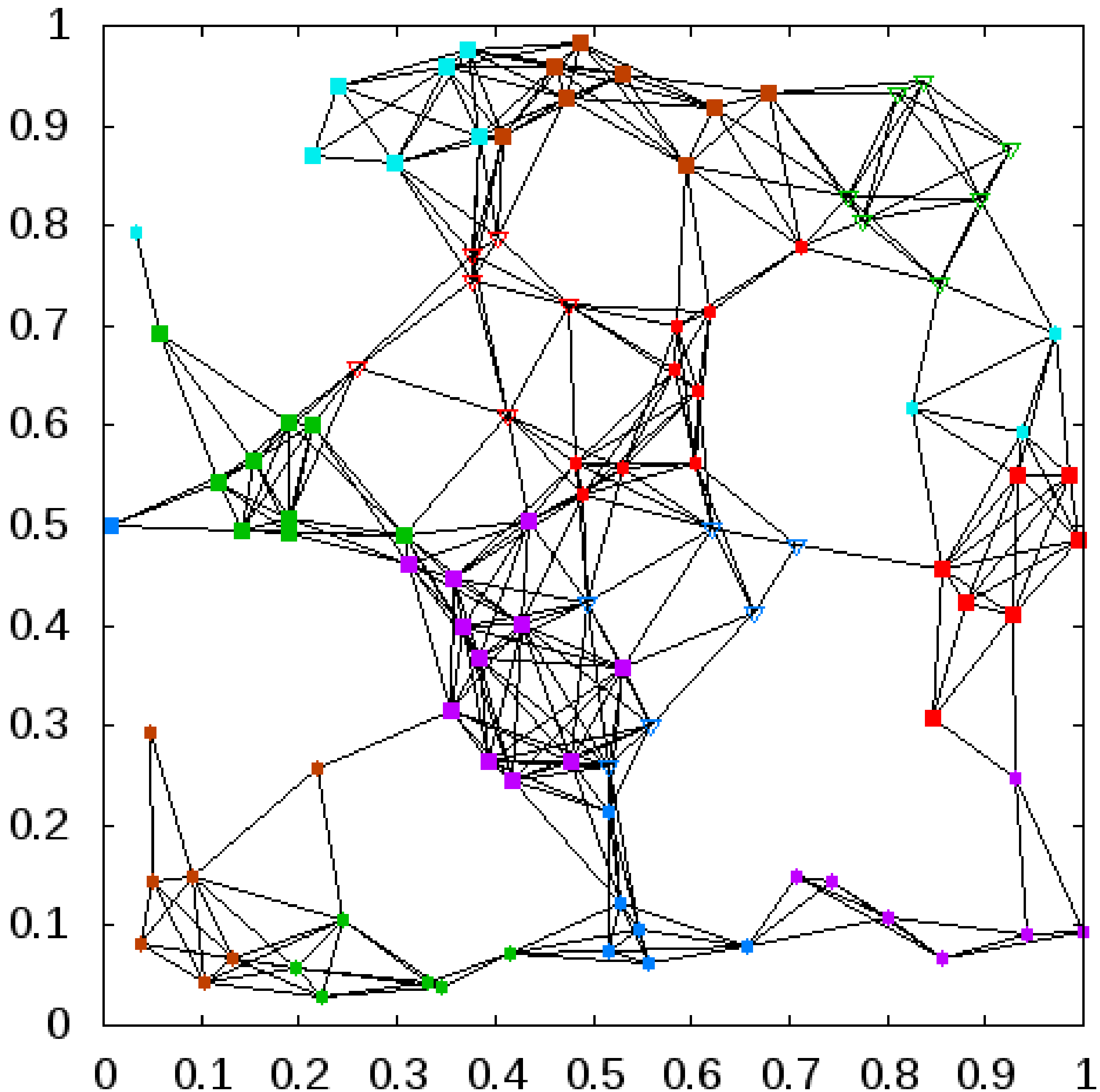


28020 handovers

28260 handovers



15 RNCs
100 towers
r = 0.17



27952
handovers

15 RNCs

Solution from run with 23 RNCs

GRASP with evolutionary path-relinking for handover minimization

GRASP with evolutionary path-relinking

- Algorithm maintains an elite set of diverse good-quality solutions found during search
- Repeat
  - build tower-to-RNC assignment π' using a randomized greedy algorithm
  - apply local search to find local min assignment π near π'
  - select assignment π' from elite pool and apply path-relinking operator between π' and π and attempt to add result to elite set
- Apply evolutionary path-relinking to elite set once in while during search

Randomized greedy construction

- Open one RNC at a time …
  - use heuristic A to assign first tower to RNC
  - while RNC can accommodate an unassigned tower
    - use heuristic B to assign next tower to RNC
- If all available RNCs have been opened and some tower is still unassigned, open one or more artificial RNCs having capacity equal to the max capacity over all real RNCs

Randomized greedy construction:
Heuristic A to assign first tower to RNC

- Let $H(i) = \text{sum}_{(j=1,\ldots,T)}\ h(i,j) + h(j,i)$
- Let Ω be the set of unassigned towers that fit in RNC
- Choose tower i from Ω with probability proportional to its H(i) value and assign i to RNC

at&t Your world. Delivered.

Randomized greedy construction: Heuristic B to assign remaining towers to RNC

- Let $g(i) = \text{sum}_{(j \in RNC)} h(i,j) + h(j,i)$
- Let Ω be the set of unassigned towers that fit in RNC
- Select tower i from Ω with probability proportional to its g(i) value and assign i to RNC

Local search

- Repeat until no improving reassignment of tower to RNC exists:
  - Let { i, j, k } be such that tower i is assigned to RNC j, RNC k has available capacity to accommodate tower i and moving i from RNC j to RNC k reduces the number of handovers between towers assigned to different RNCs
  - If { i, j, k } exists, then move tower i from RNC j to RNC k

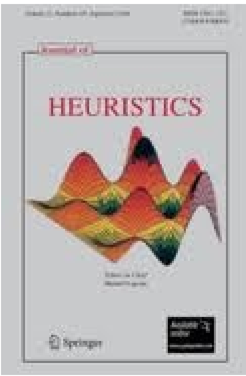
Path-relinking

Intensification strategy exploring trajectories connecting elite solutions (Glover, 1996)

Originally proposed in the context of tabu search and scatter search.

Paths in the solution space leading to other elite solutions are explored in the search for better solutions.

Recent survey paper on path-relinking

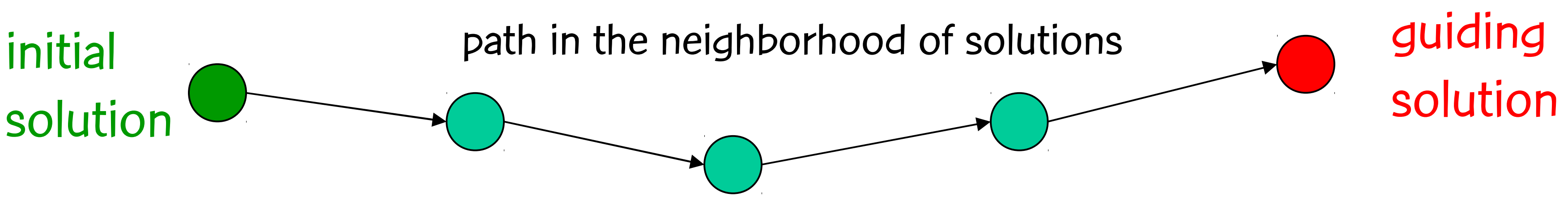


C.C. Ribeiro and M.G.C. Resende, Path-relinking intensification methods for stochastic local search algorithms, J. of Heuristics, vol. 18, pp. 193-214, 2012.

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

Path-relinking

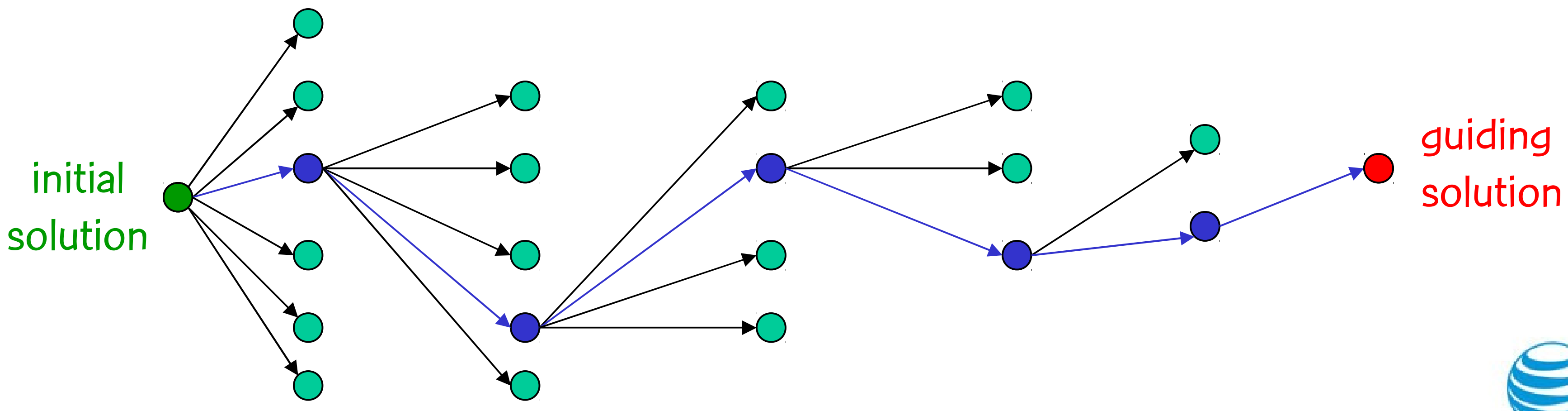
Exploration of trajectories that connect high quality (elite) solutions:



Path-relinking

Path is generated by selecting moves that introduce in the initial solution attributes of the guiding solution.

At each step, all moves that incorporate attributes of the guiding solution are evaluated and the best move is selected:



PR example

starting solution

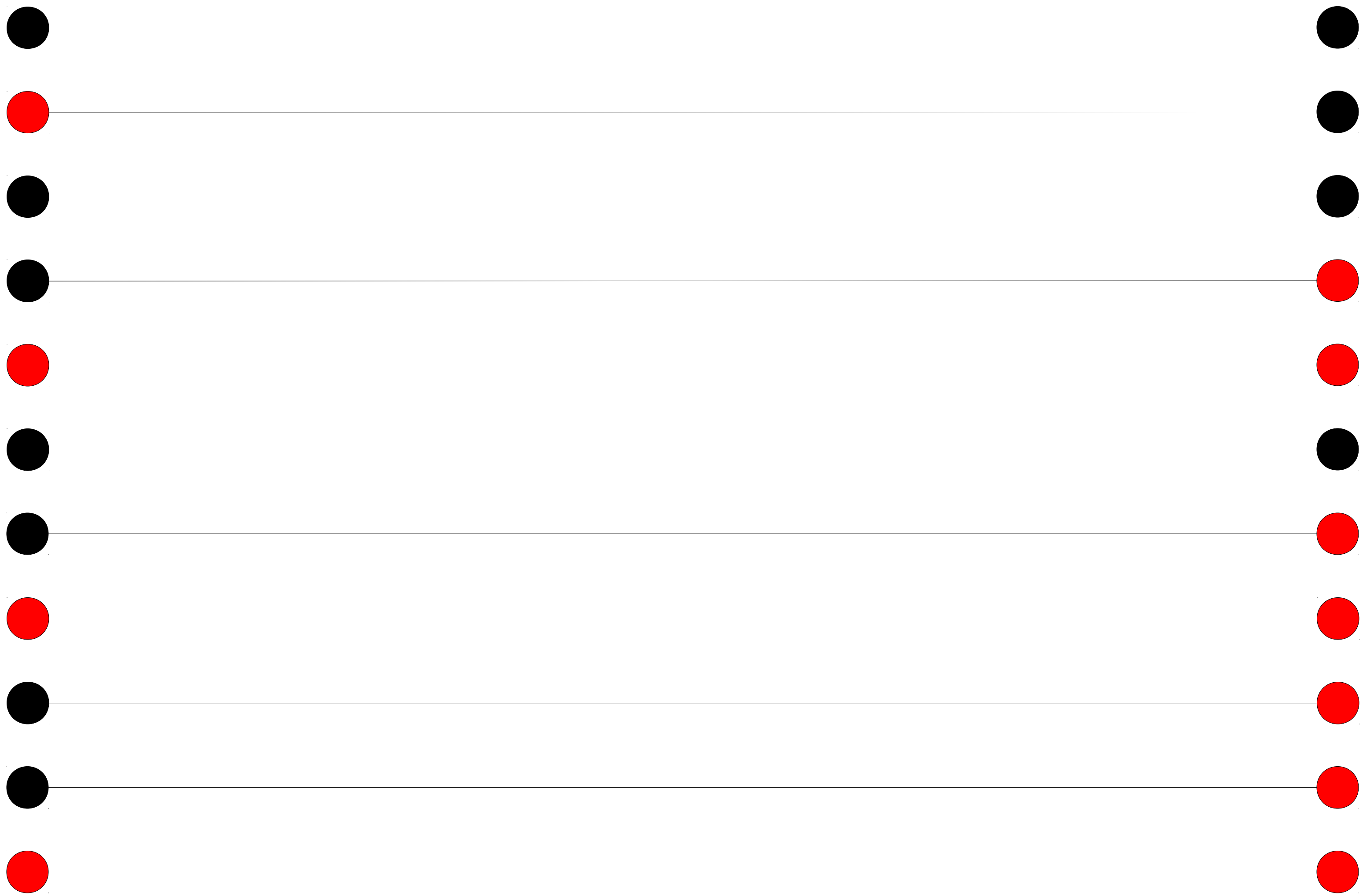
guiding solution

Example with two RNCs

starting solution x

PR example

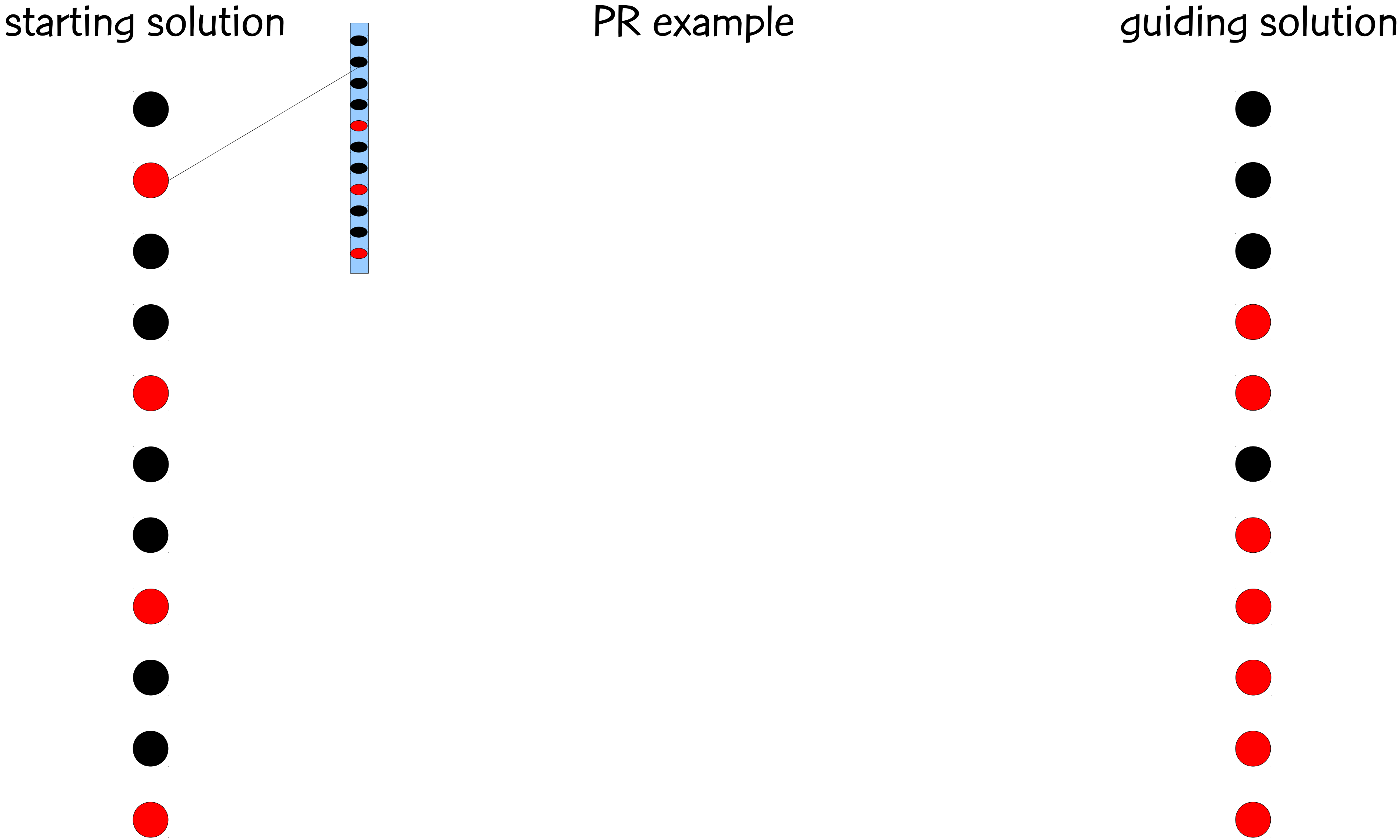
guiding solution y



$|\Delta\ (x,y)| = 5$

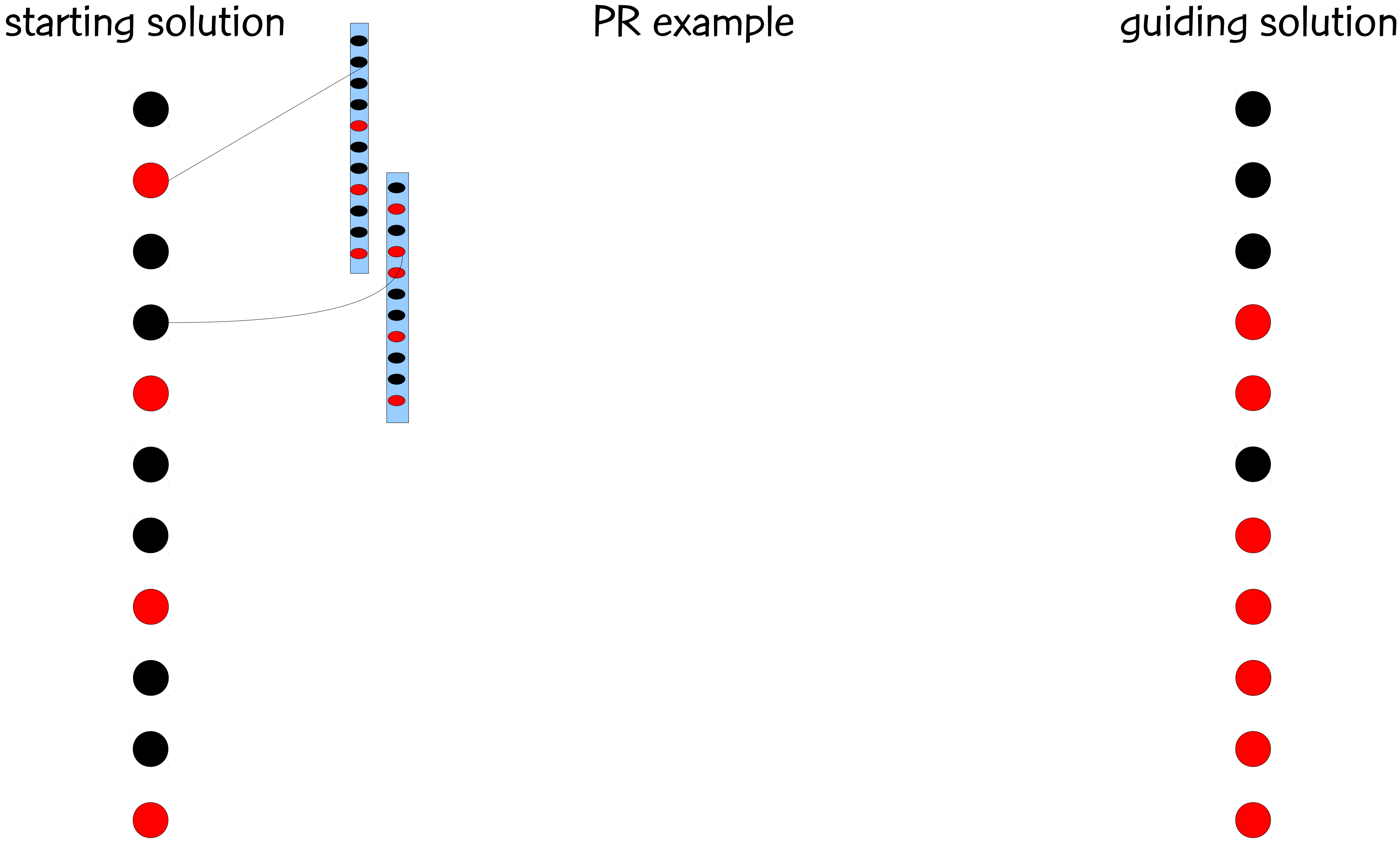
Example with two RNCs

PR example



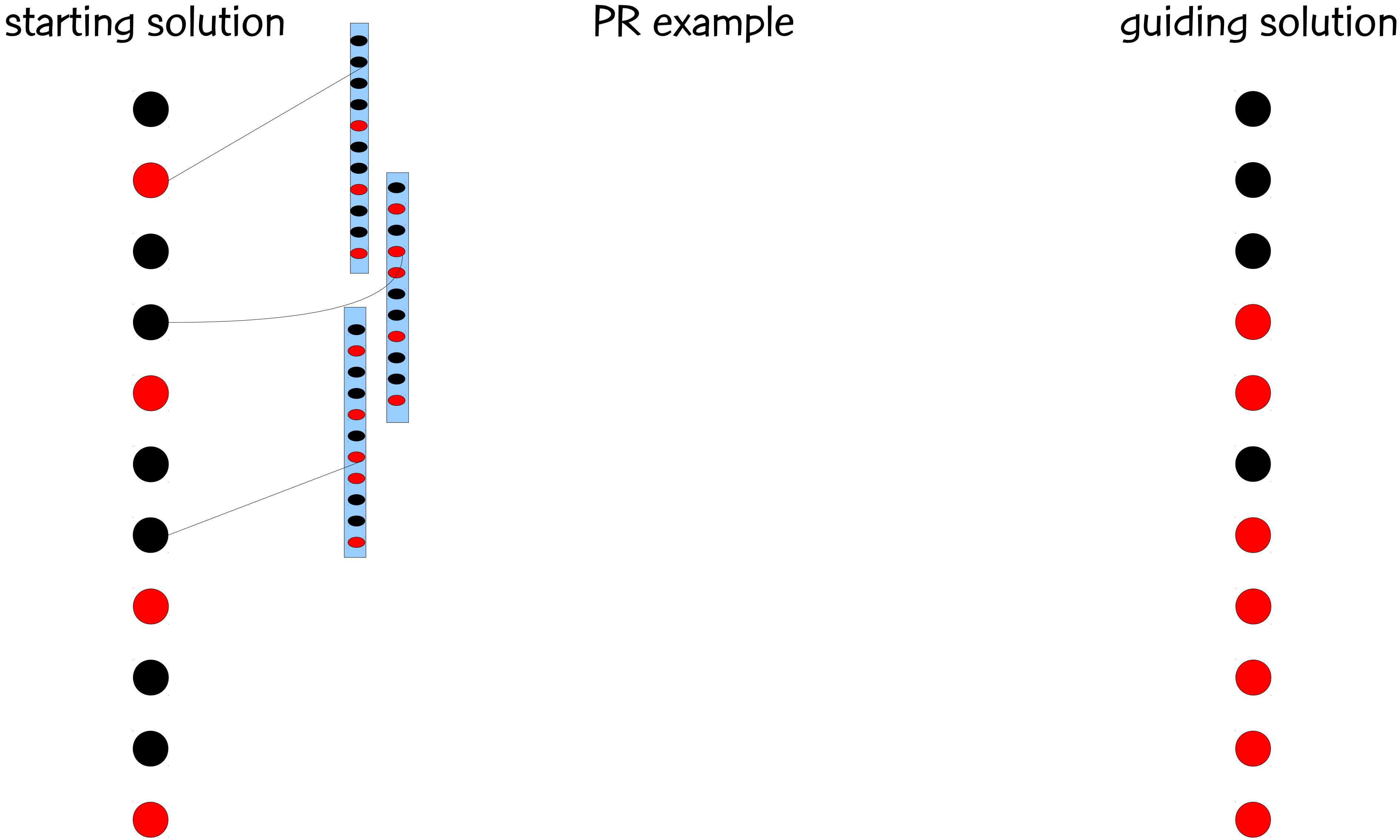
Example with two RNCs

PR example



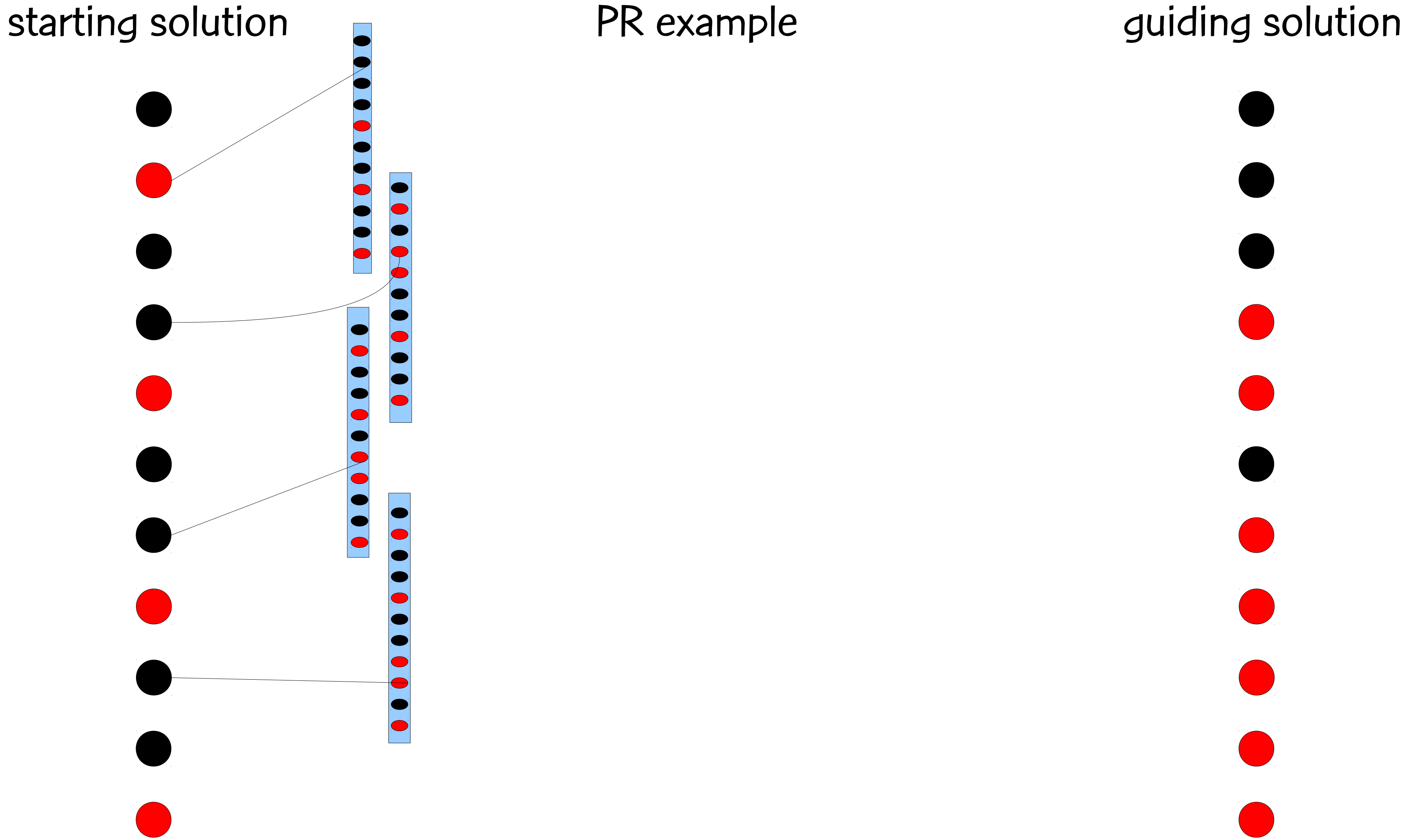
Example with two RNCs

PR example



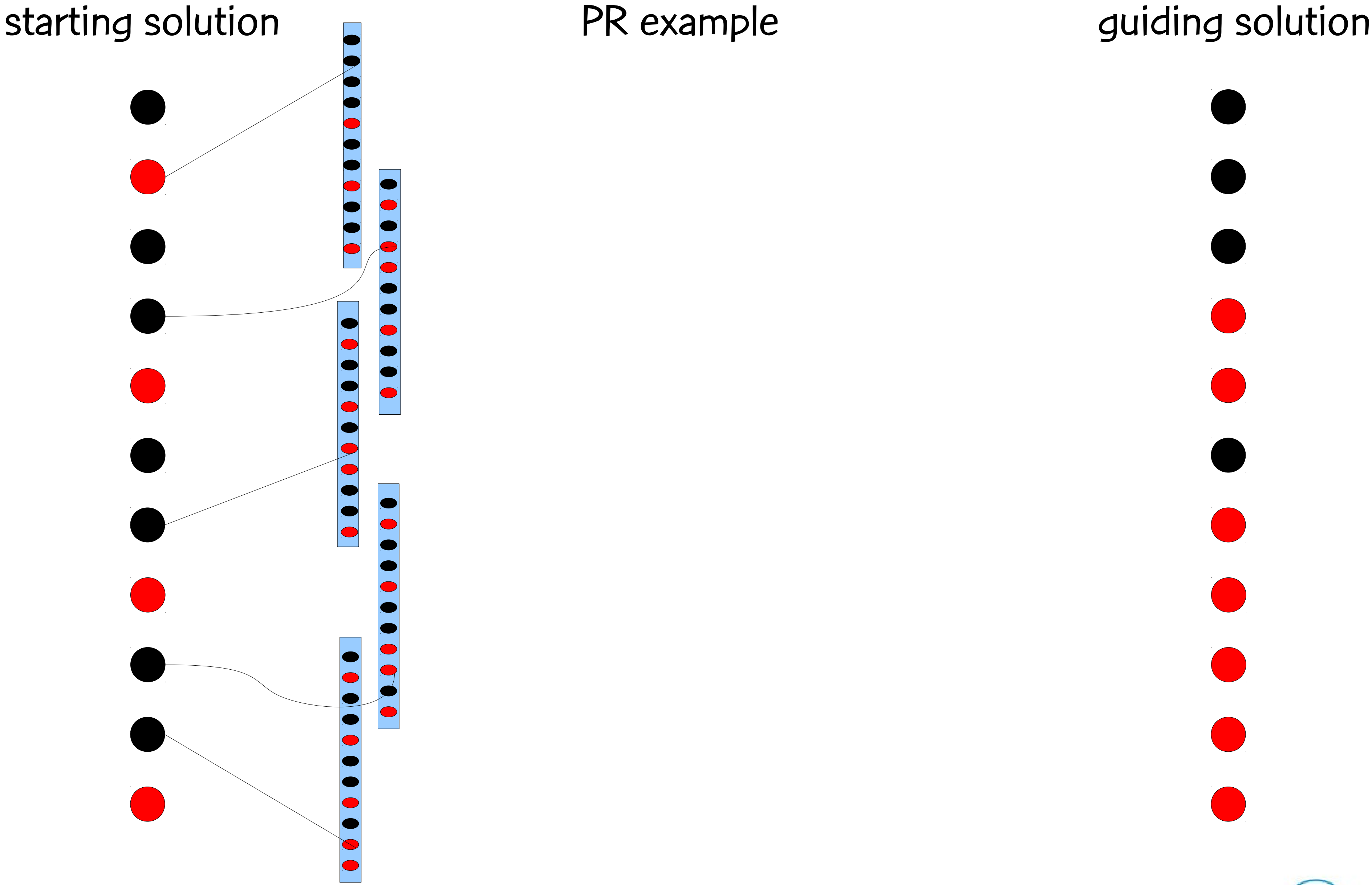
Example with two RNCs

PR example



Example with two RNCs

PR example

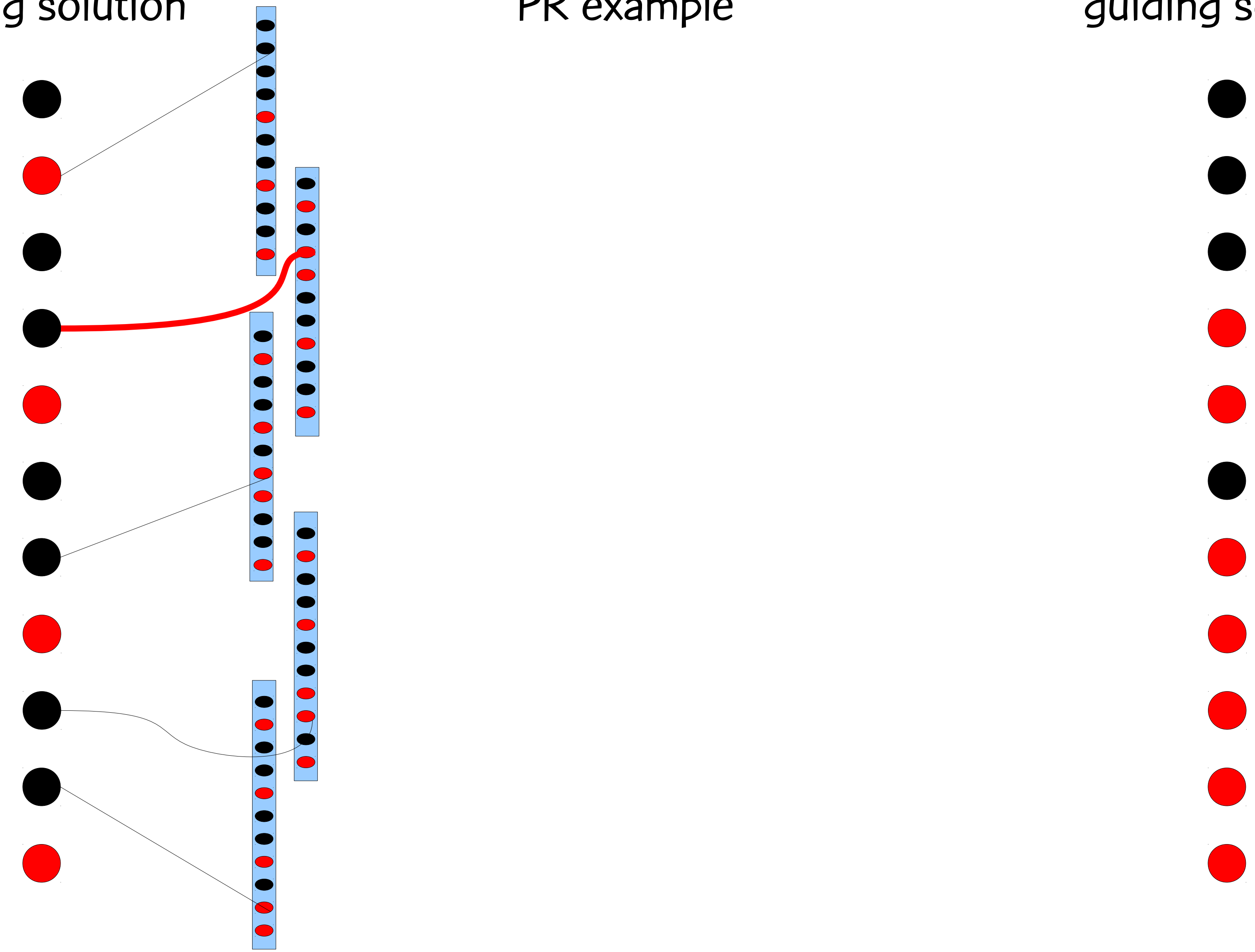


Example with two RNCs

PR example

starting solution

guiding solution

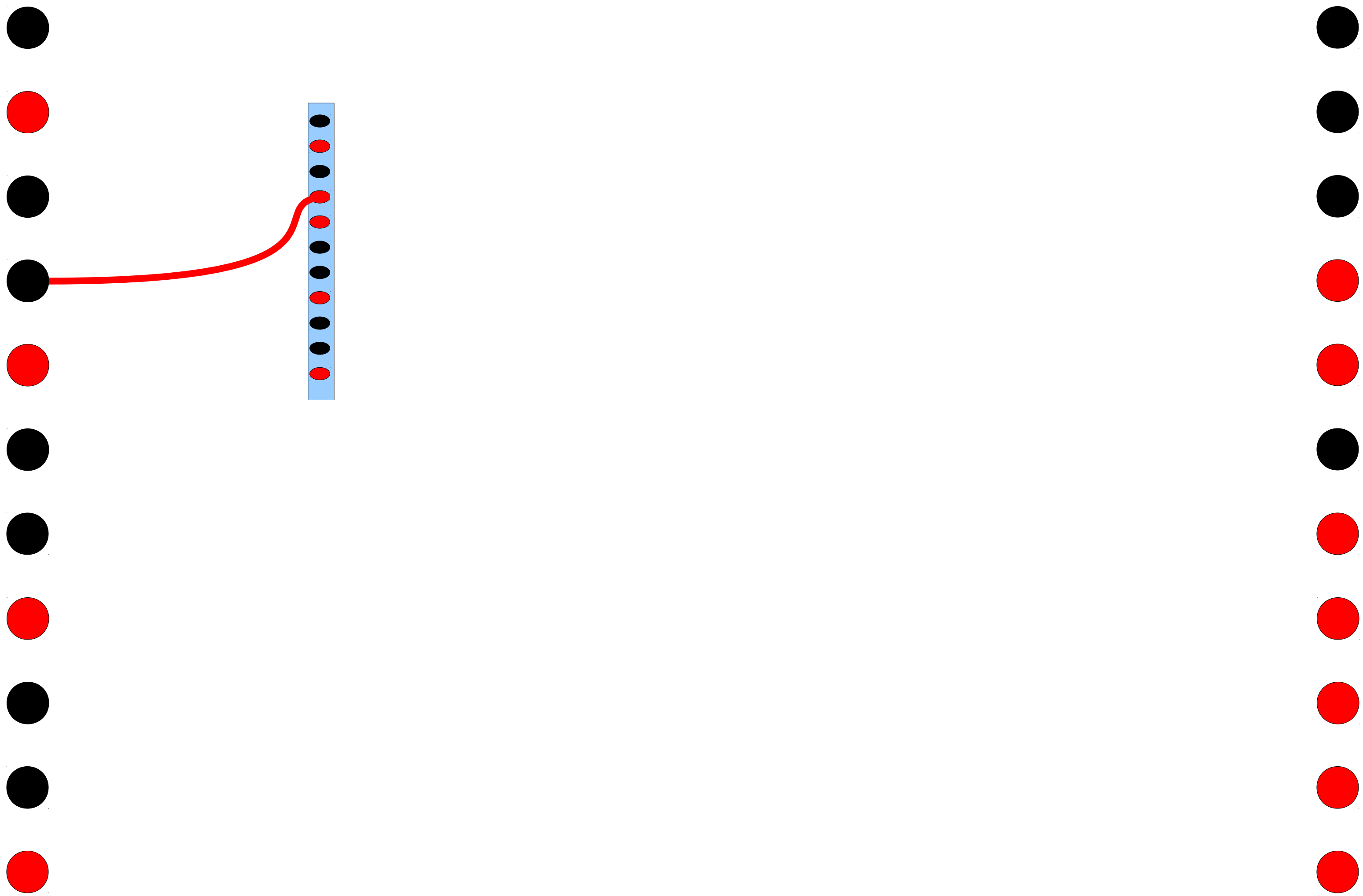


Example with two RNCs

starting solution

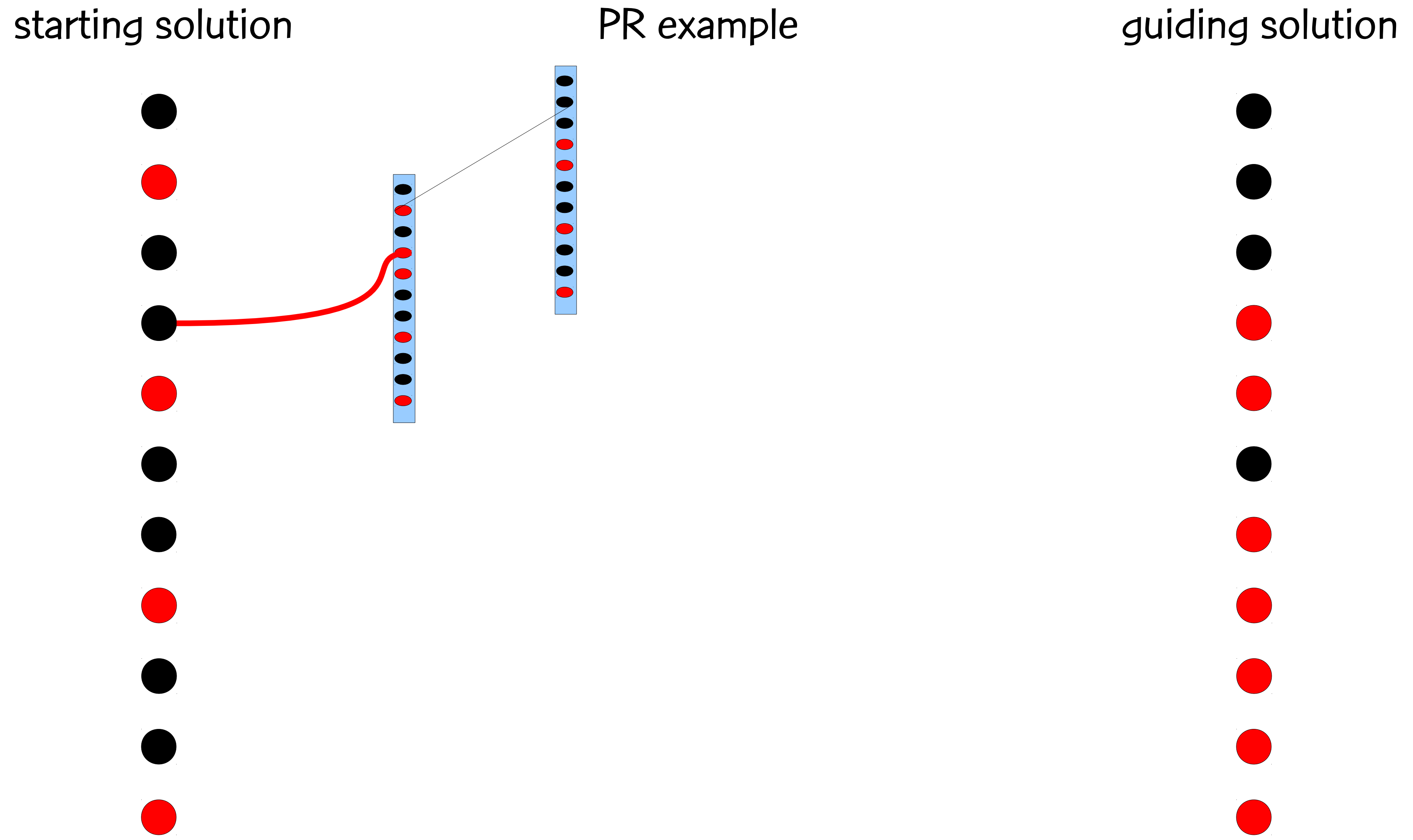
PR example

guiding solution



Example with two RNCs

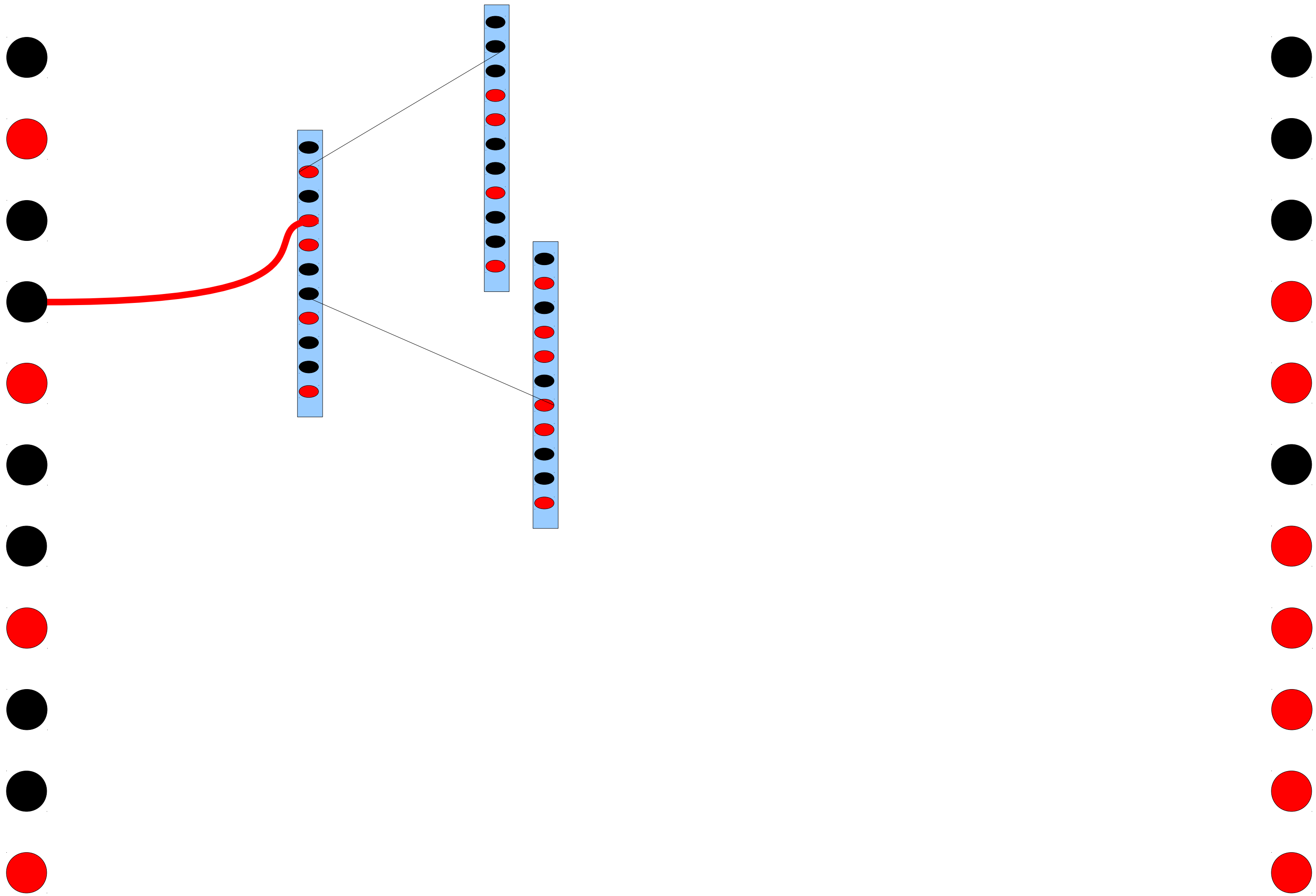
PR example



Example with two RNCs

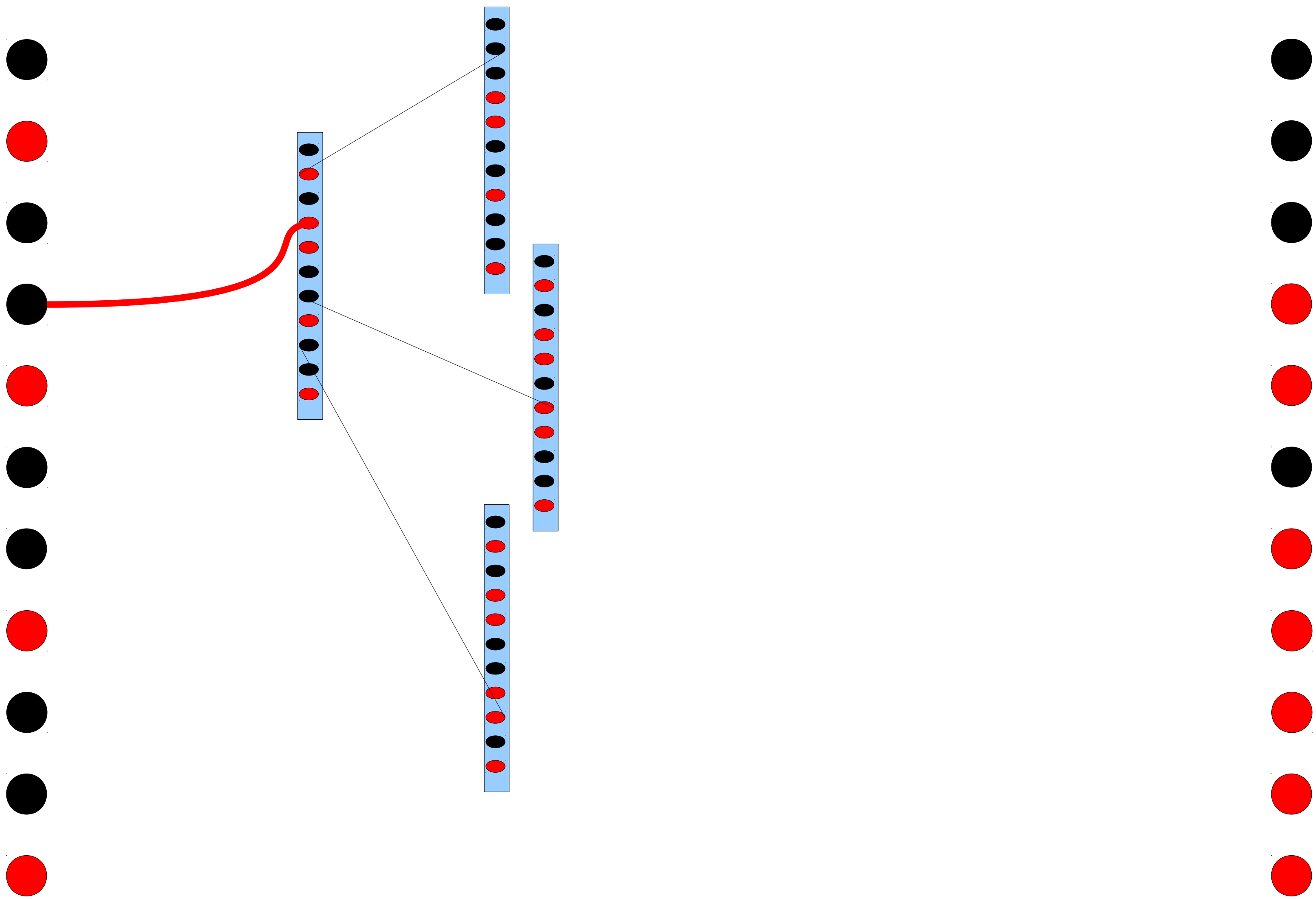
PR example

starting solution | guiding solution



Example with two RNCs

starting solution PR example guiding solution

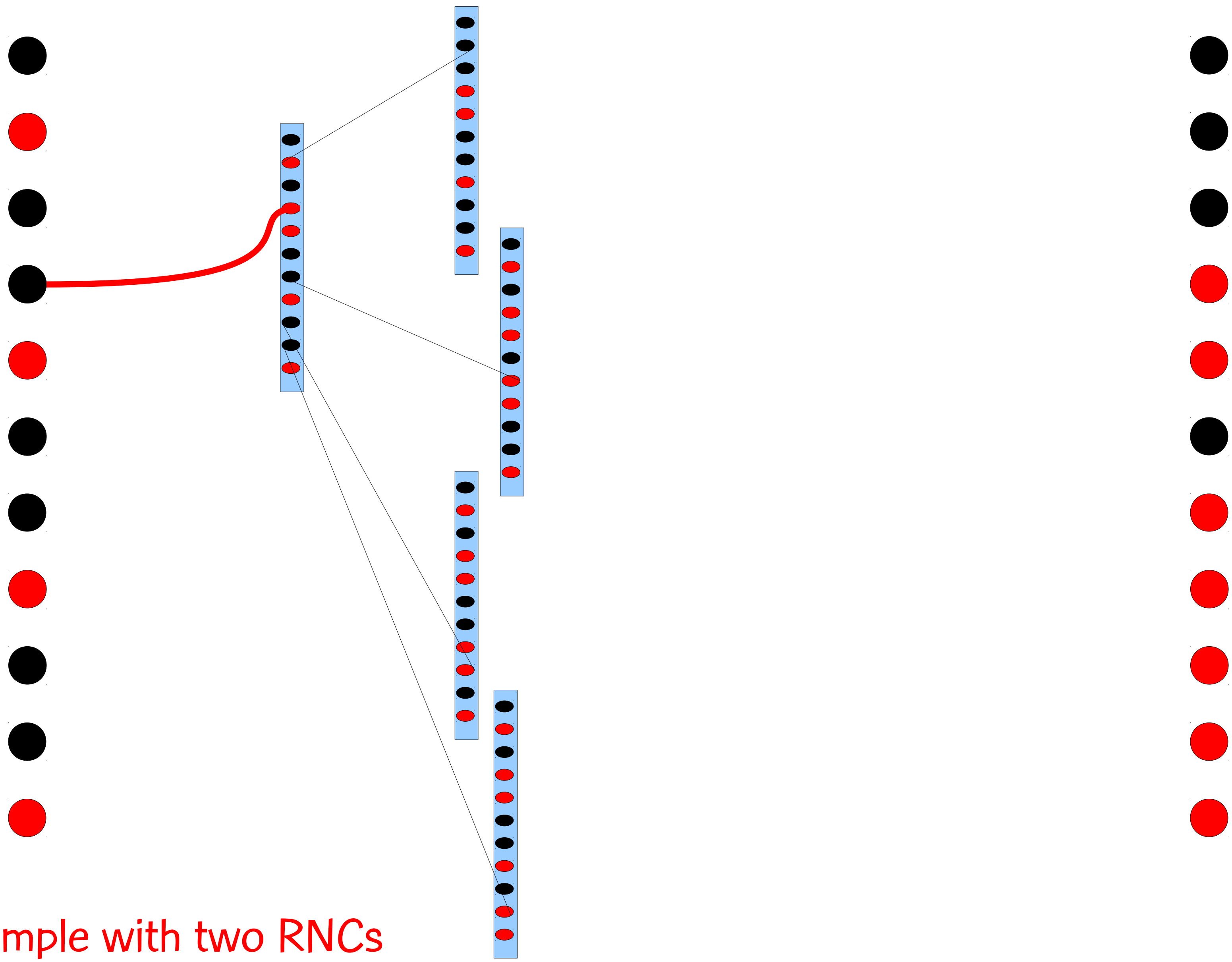


Example with two RNCs

PR example

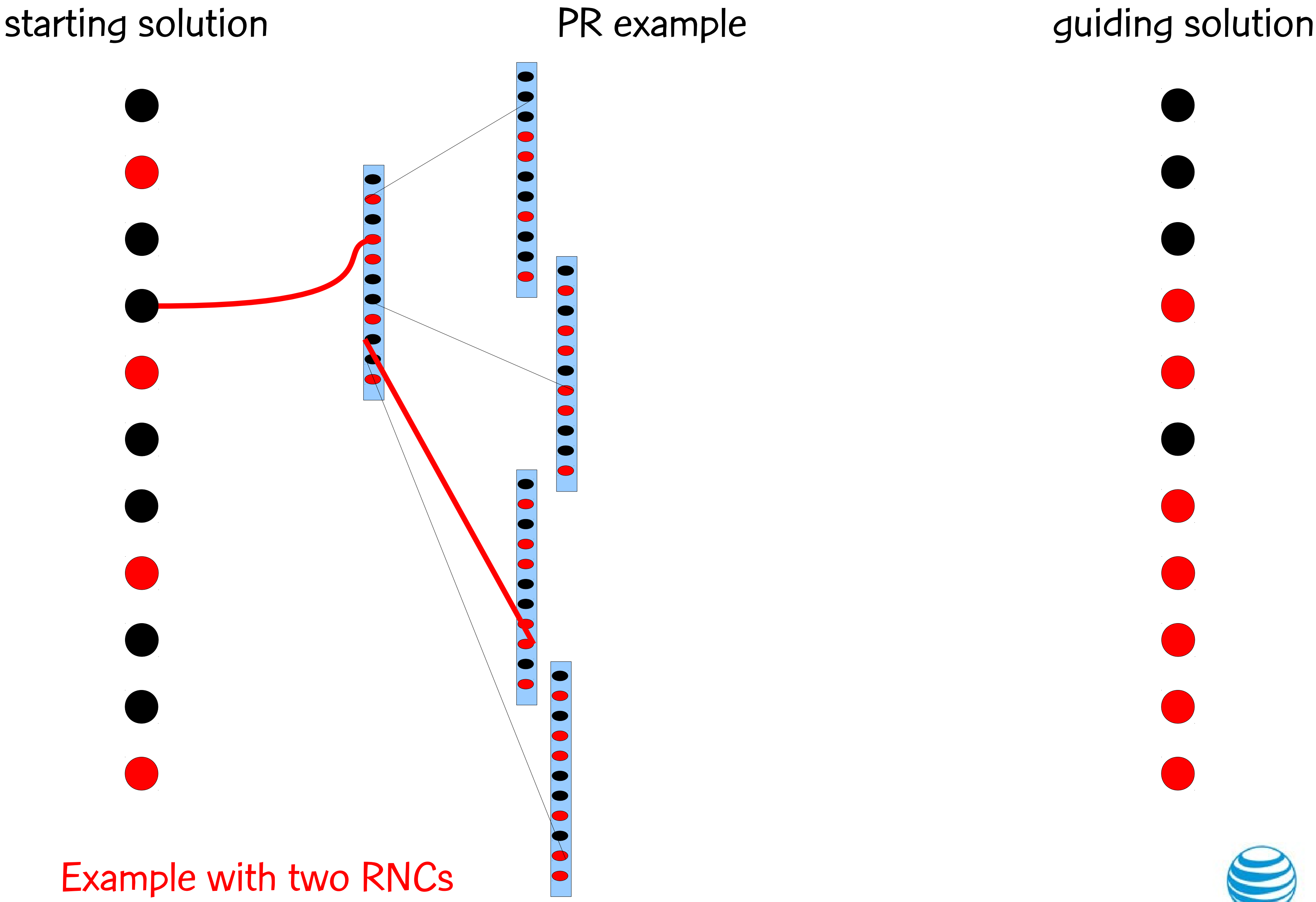
starting solution

guiding solution



Example with two RNCs

PR example

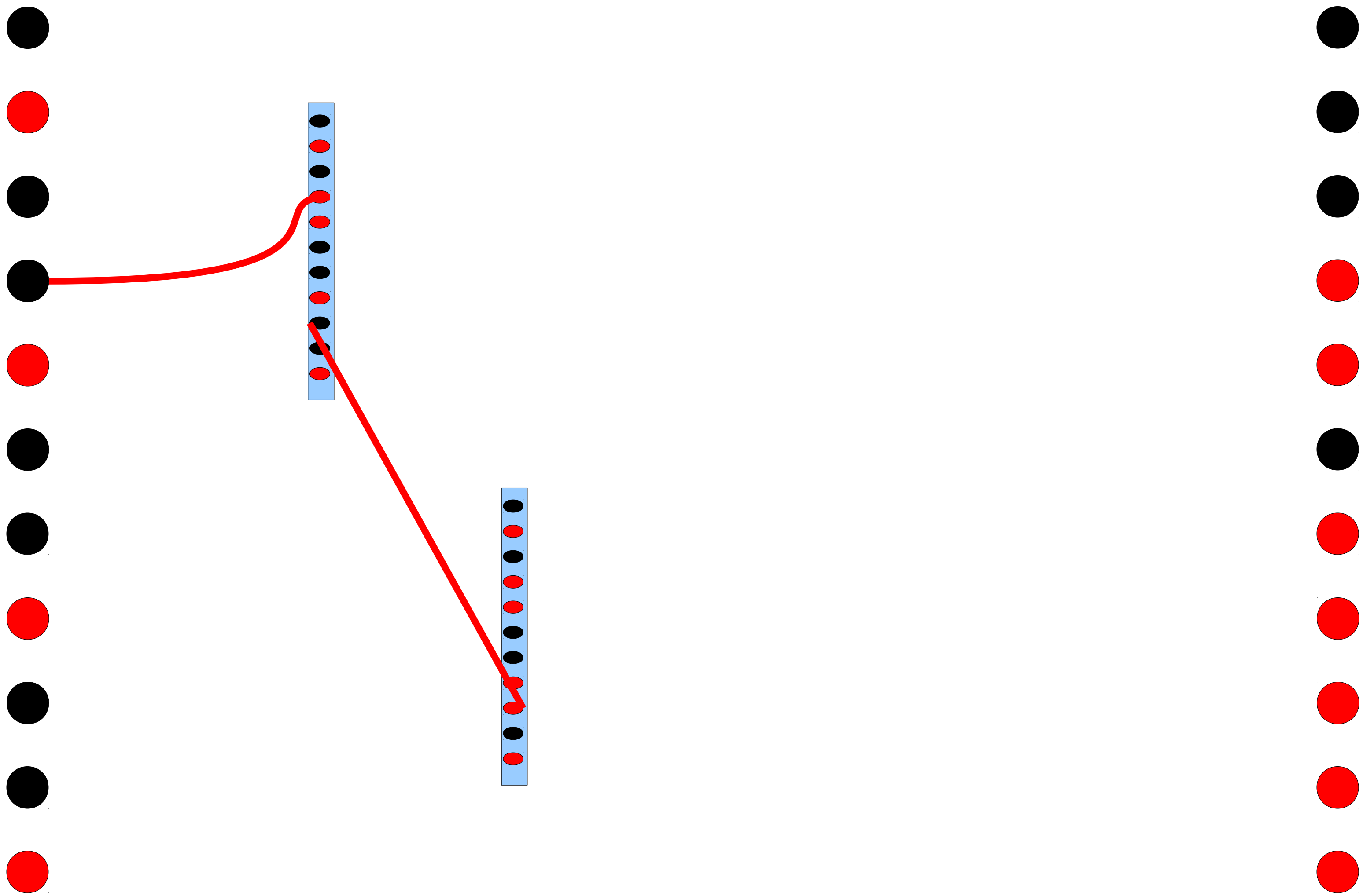


Example with two RNCs

PR example

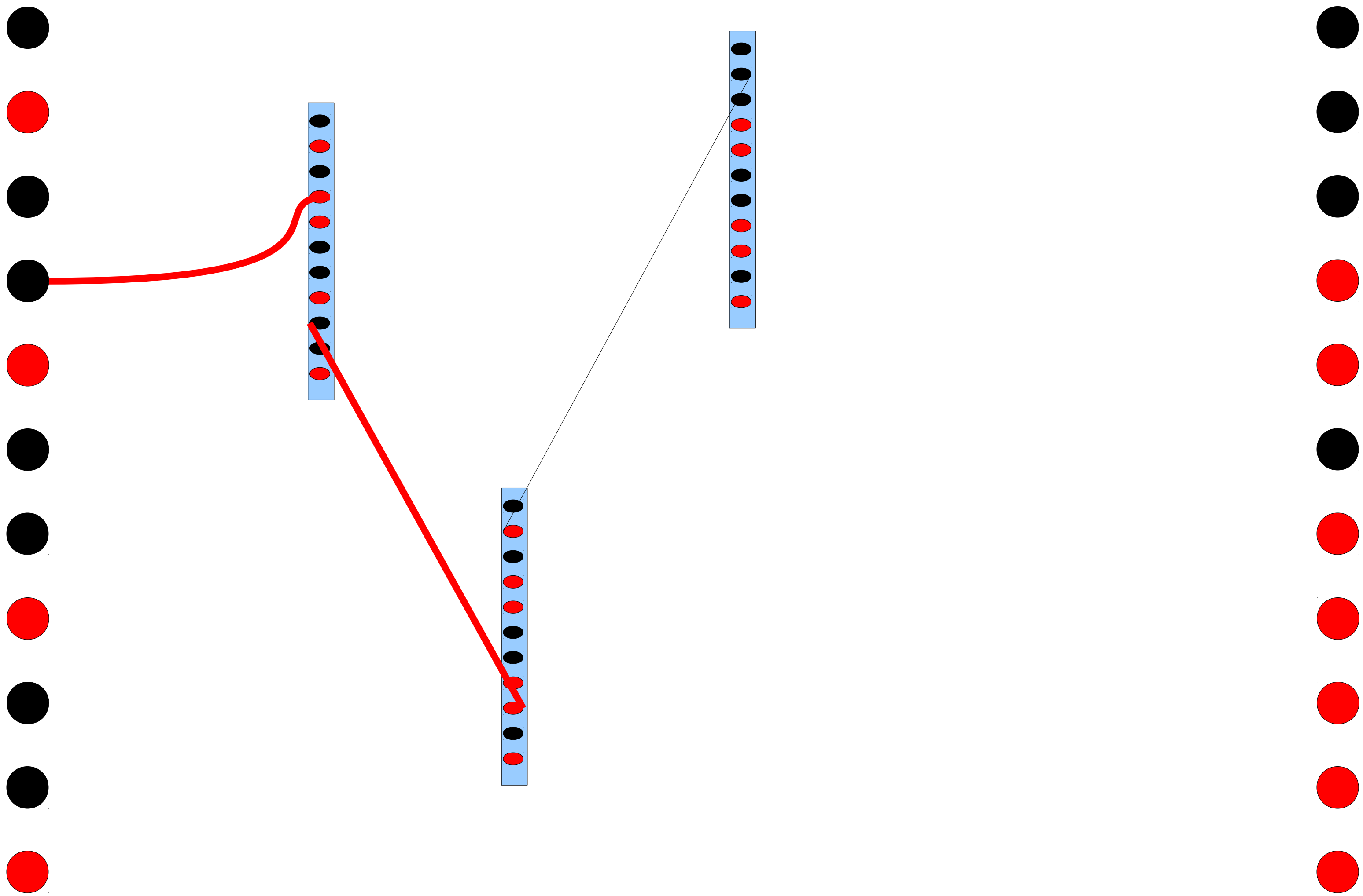
starting solution

guiding solution



Example with two RNCs

starting solution PR example guiding solution

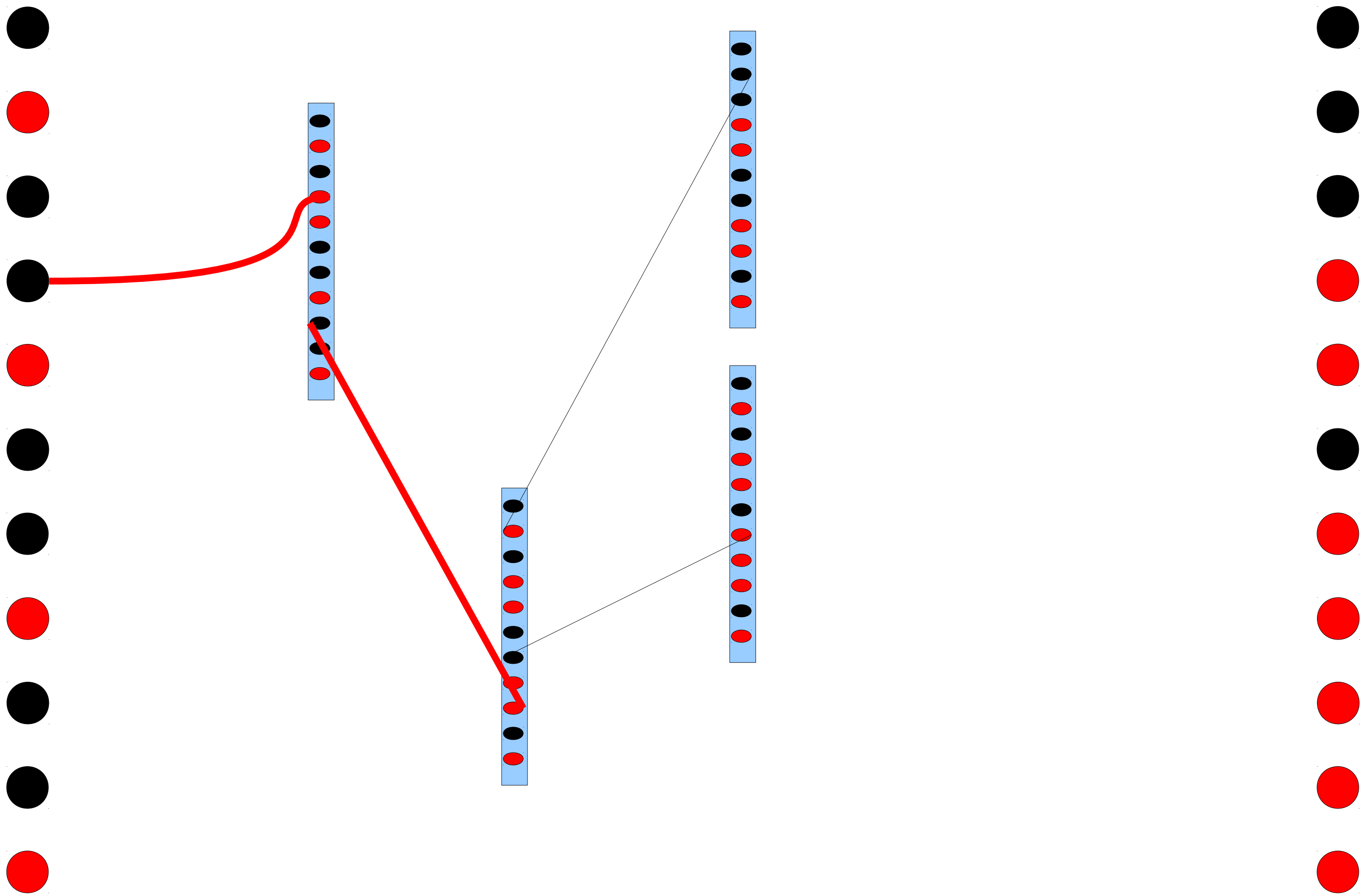


Example with two RNCs

starting solution

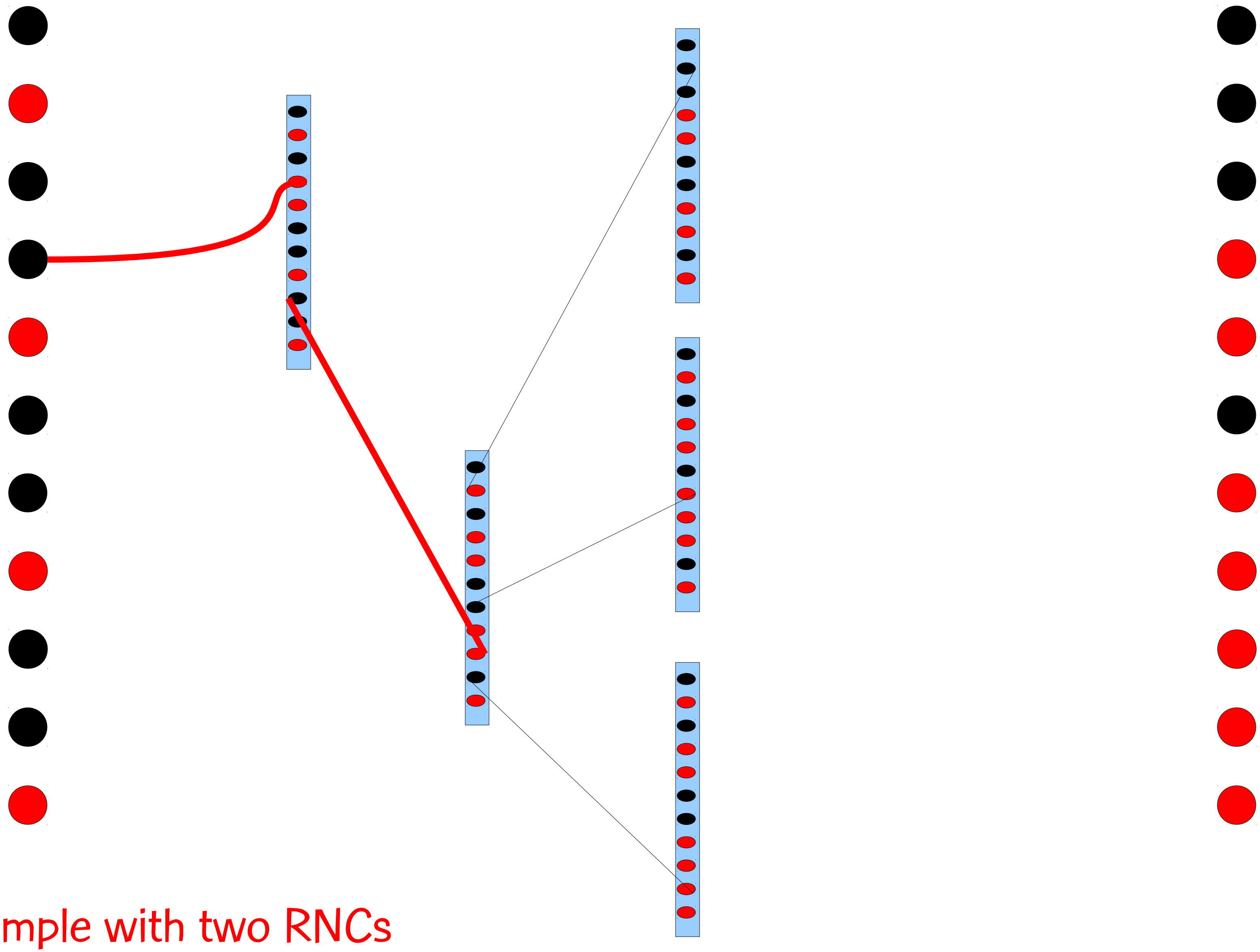
PR example

guiding solution



Example with two RNCs

starting solution PR example guiding solution

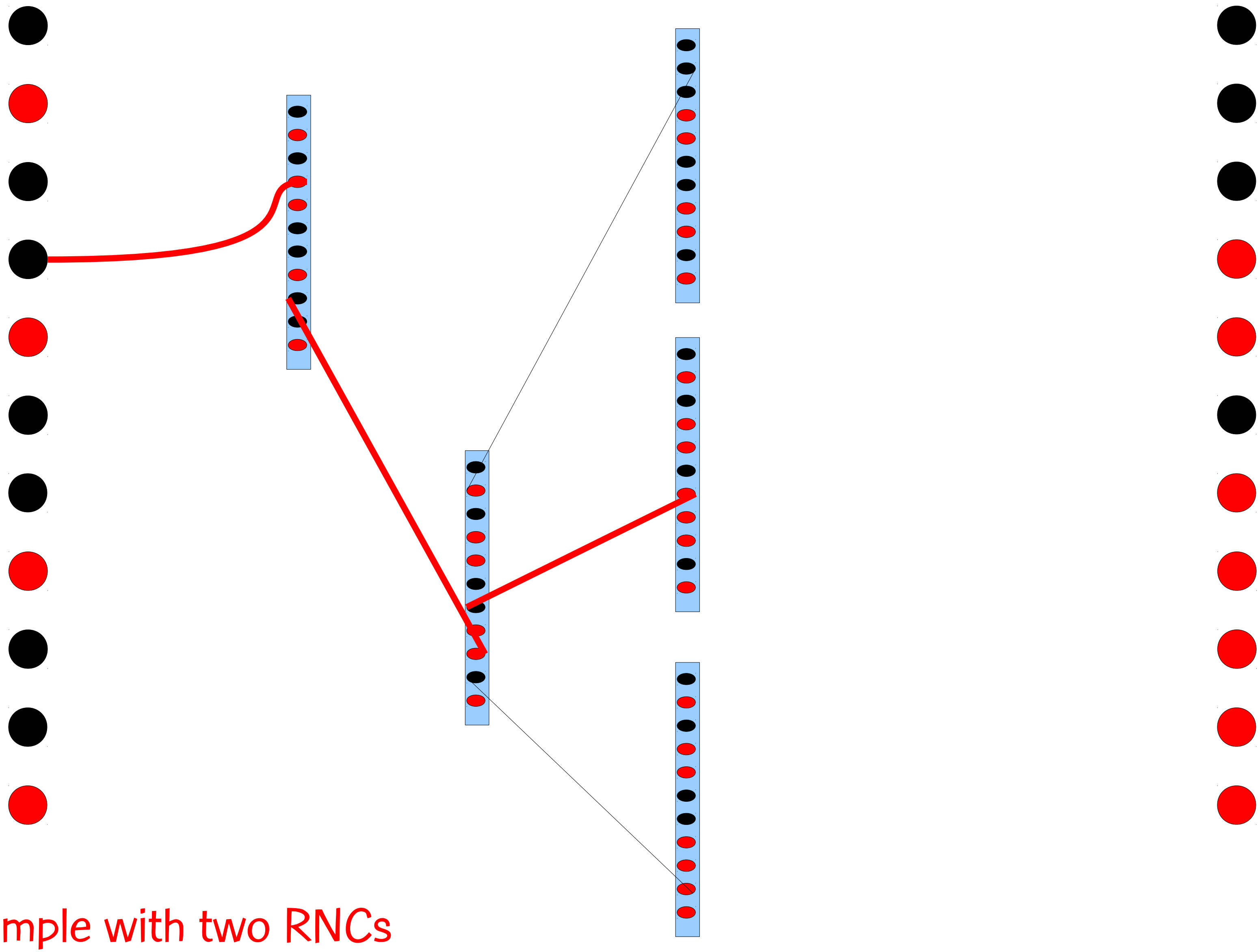


Example with two RNCs

starting solution

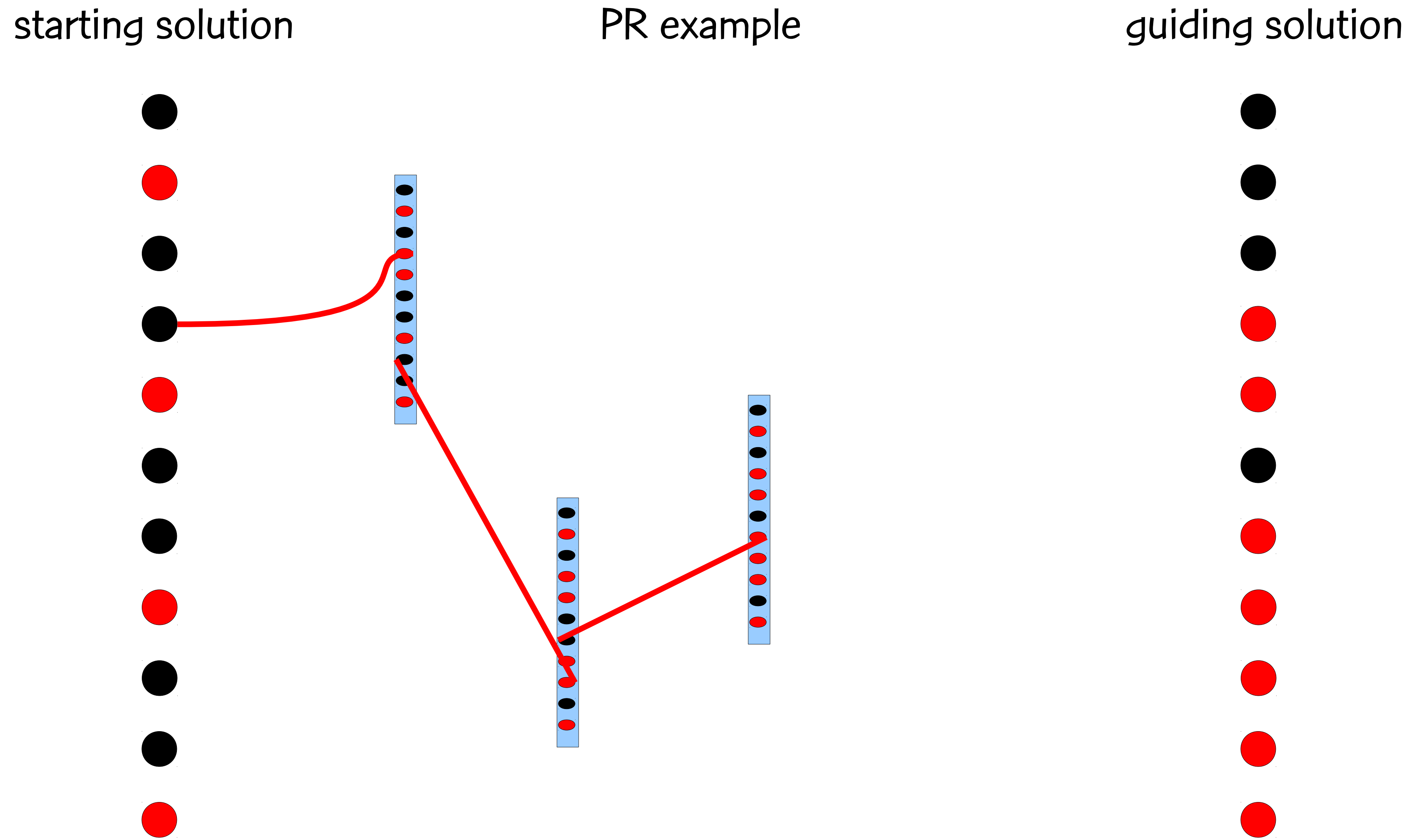
PR example

guiding solution



Example with two RNCs

PR example

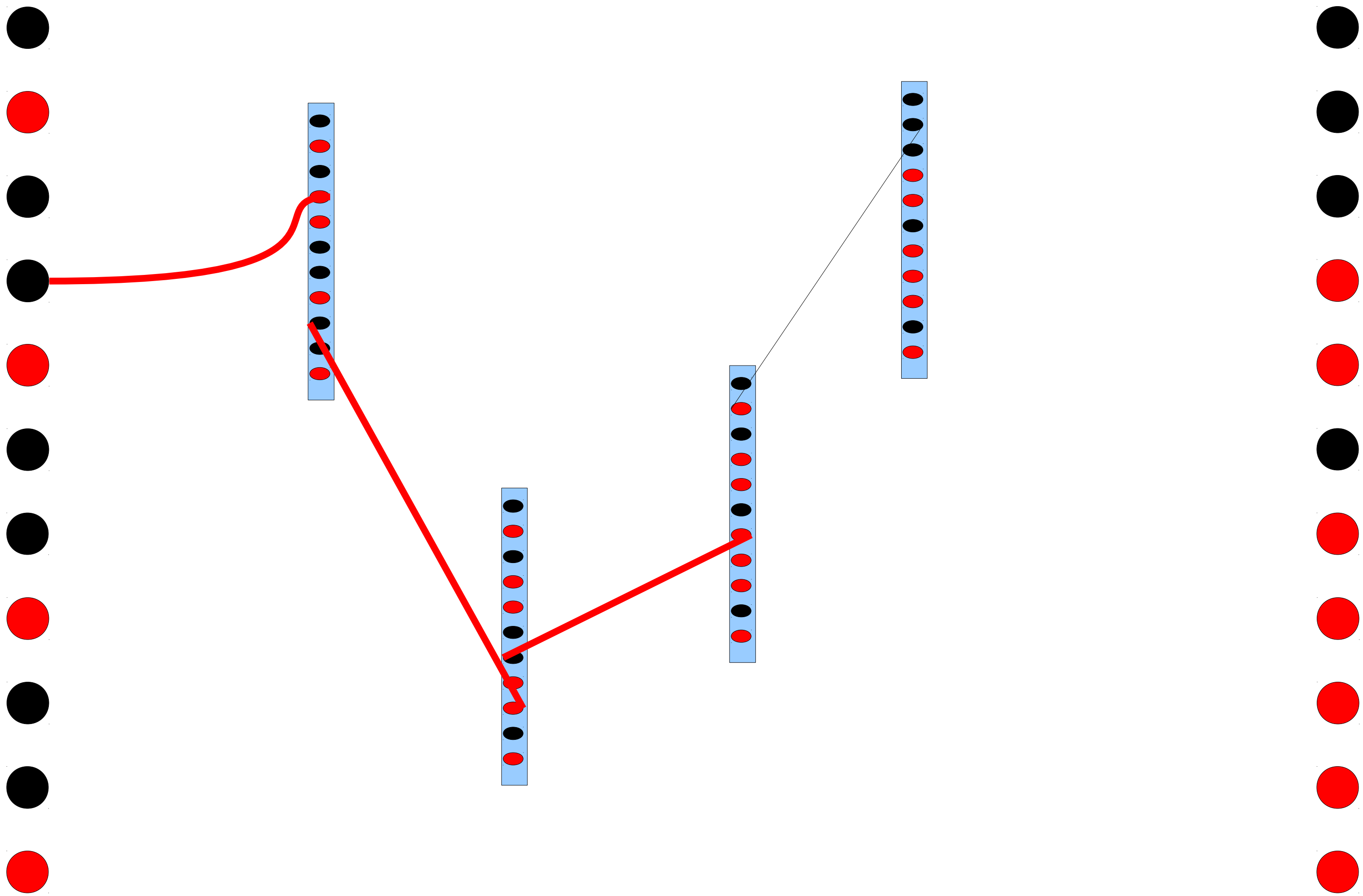


Example with two RNCs

starting solution

PR example

guiding solution

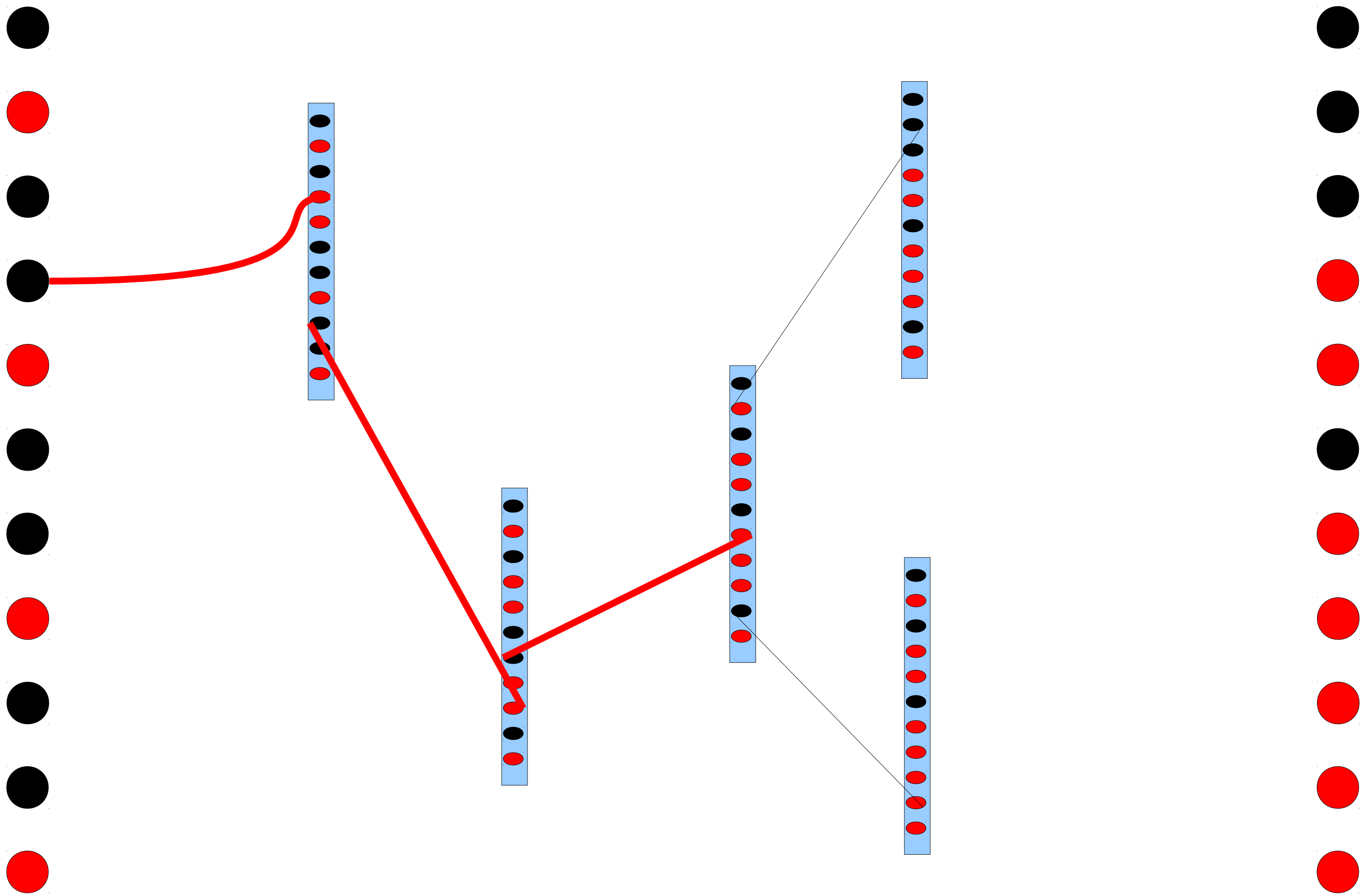


Example with two RNCs

starting solution

PR example

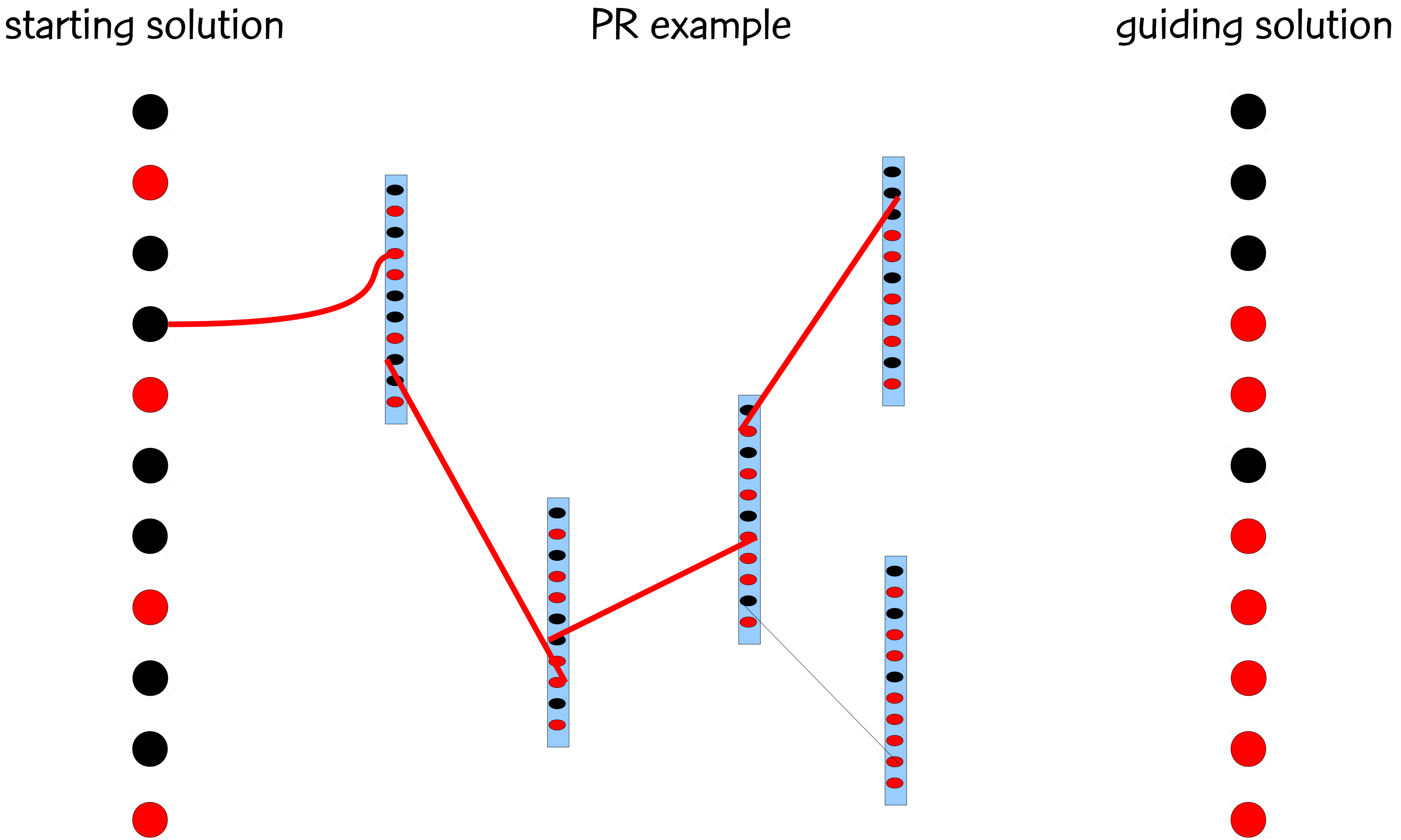
guiding solution



Example with two RNCs

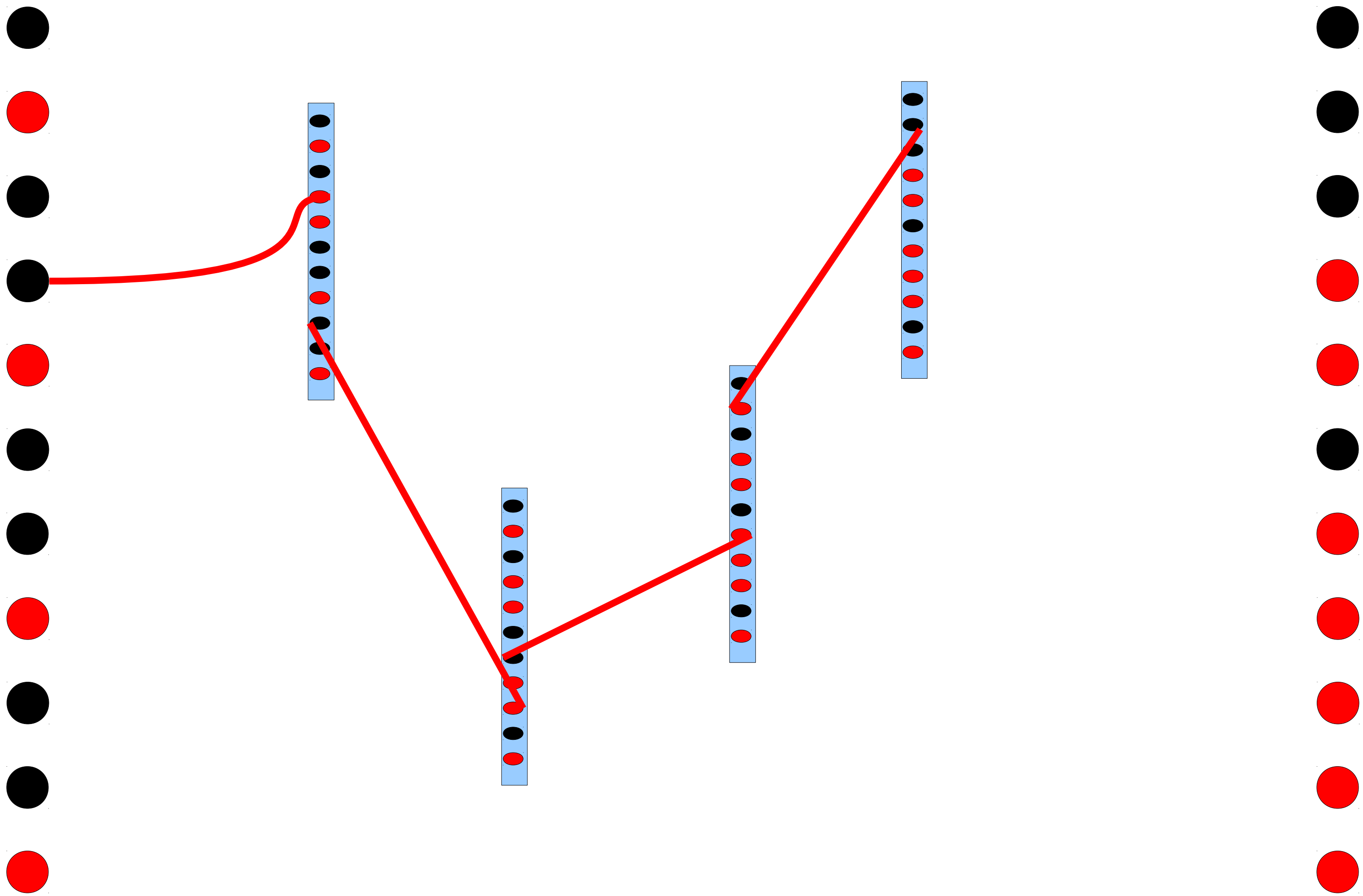
at&t
Your world. Delivered.

PR example



Example with two RNCs

starting solution PR example guiding solution

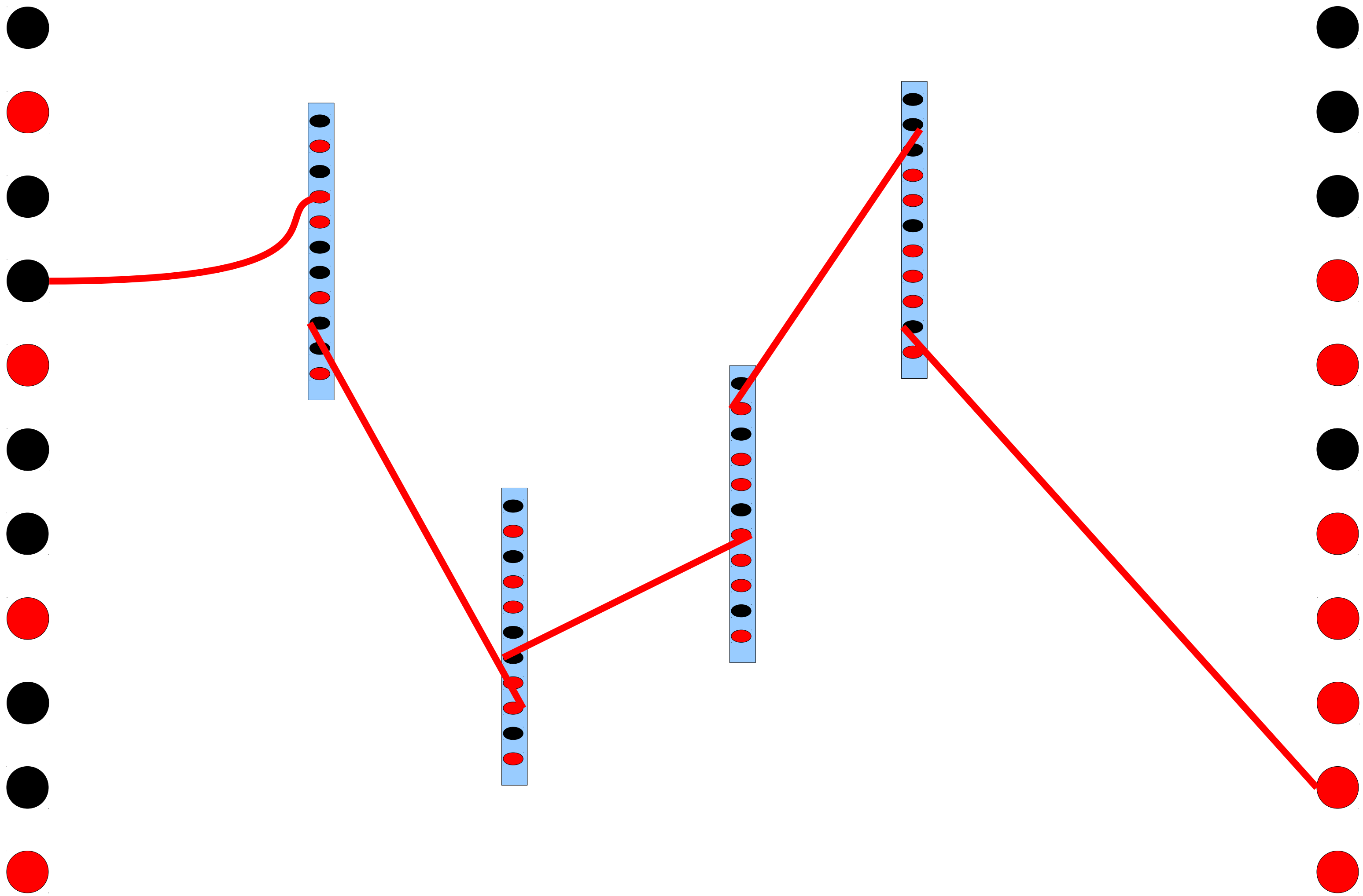


Example with two RNCs

starting solution

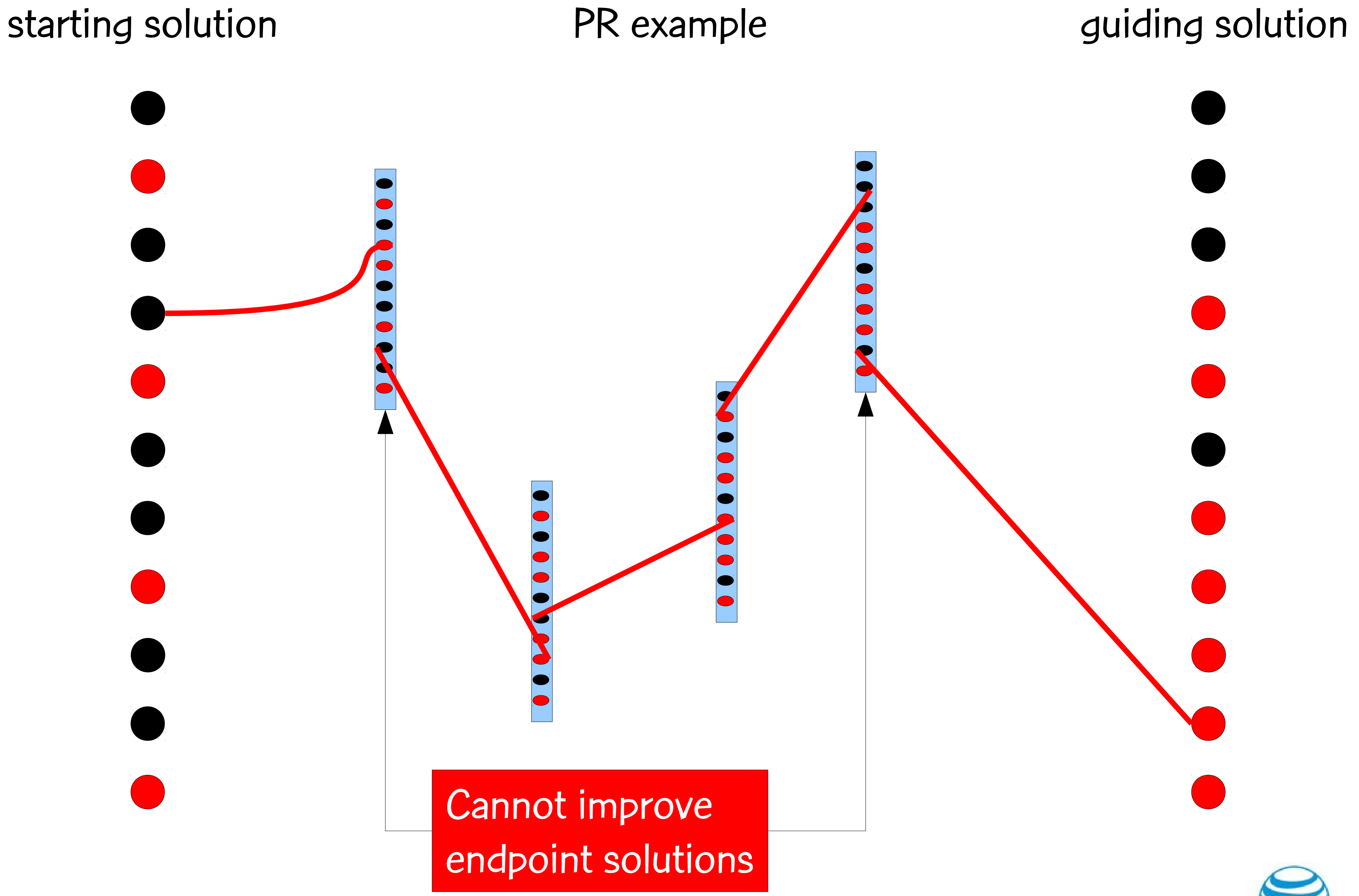
PR example

guiding solution



Example with two RNCs

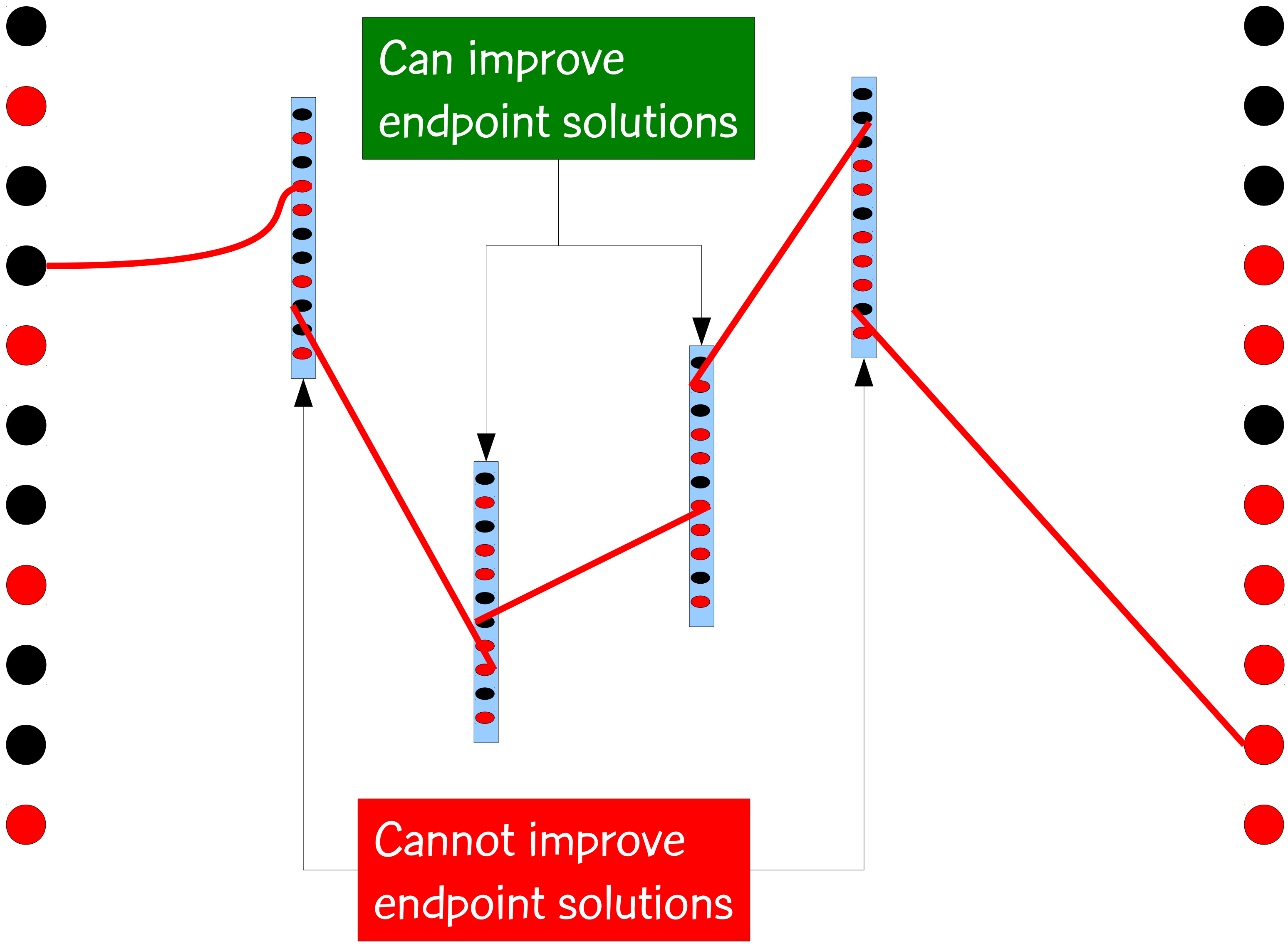
PR example


starting solution
guiding solution
Cannot improve
endpoint solutions

starting solution

PR example

guiding solution



GRASP with path-relinking:
Pool management

P is a set (pool, or set) of elite solutions.

Ideally, pool has a set of good diverse solutions.

Mechanisms are needed to guarantee that pool is made up of those kinds of solutions.

GRASP with path-relinking:
Pool management

Each iteration of first |P| GRASP iterations adds one solution to P (if different from others).

After that: solution x is promoted to P if:

x is better than best solution in P.

x is not better than best solution in P, but is better than worst and is sufficiently different from all solutions in P.

GRASP with path-relinking:
Pool management

GRASP with PR works best when paths in PR are long, i.e. when the symmetric difference between the initial and guiding solutions is large.

Given a solution to relink with an elite solution, which elite solution to choose?

Choose at random with probability proportional to the symmetric difference.

GRASP with path-relinking:
Pool management

Solution quality and diversity are two goals of pool design.

Given a solution X to insert into the pool, which elite solution do we choose to remove?

Of all solutions in the pool with worse solution than X, select to remove the pool solution most similar to X, i.e. with the smallest symmetric difference from X.

GRASP with path-relinking

Repeat GRASP with PR loop	1) Construct randomized greedy X 2) Y = local search to improve X 3) Path-relinking between Y and pool solution Z 4) Update pool

Evolutionary path-relinking (EvPR)

Evolutionary path-relinking

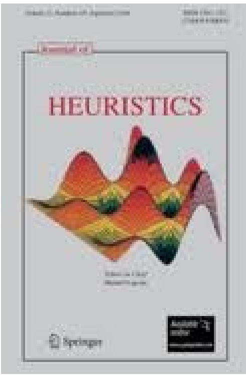
Evolutionary path-relinking "evolves" the pool, i.e. transforms it into a pool of diverse elements whose solution values are better than those of the original pool.

Evolutionary path-relinking can be used as

1) an intensification procedure at certain points of the solution process;

2) a post-optimization procedure at the end of the solution process.

Evolutionary path-relinking proposed in



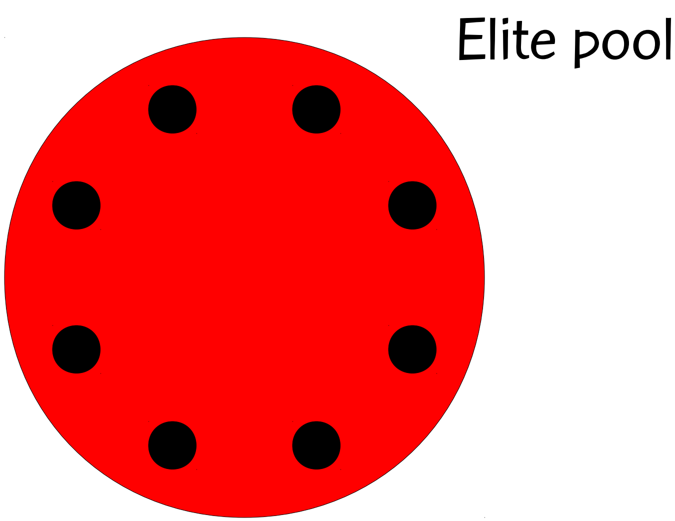
M.G.C. Resende and R.F. Werneck, A hybrid heuristic for the p-median problem, J. of Heuristics, vol. 10, pp. 59-88, 2004.

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

M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte, GRASP and path relinking for the max-min diversity problem, Computers & Operations Research, vol. 37, pp. 498-508, 2010.

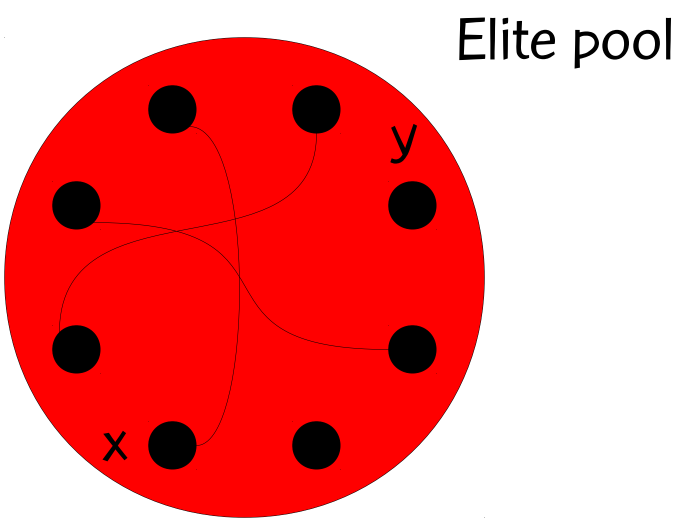
http://www.research.att.com/~mgcr/doc/gpr-maxmindiv.pdf

Evolutionary path-relinking (EvPR)



Start with current elite set.

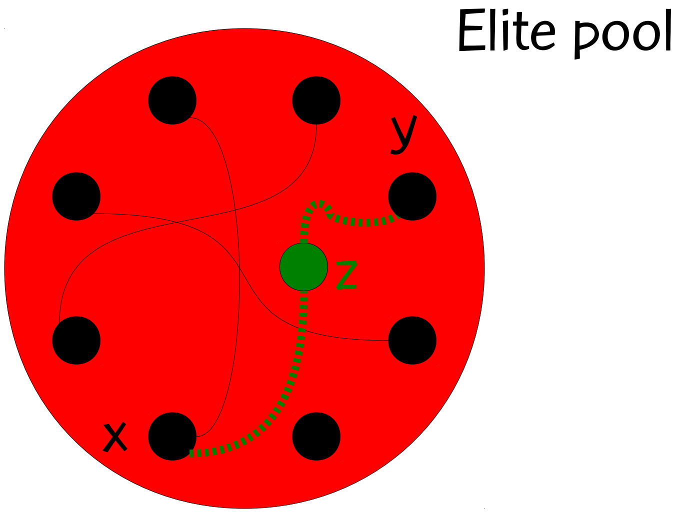
Evolutionary path-relinking (EvPR)



Start with current elite set.

While there is a pair {x,y} of pool solutions that has not yet been relinked:

Evolutionary path-relinking (EvPR)



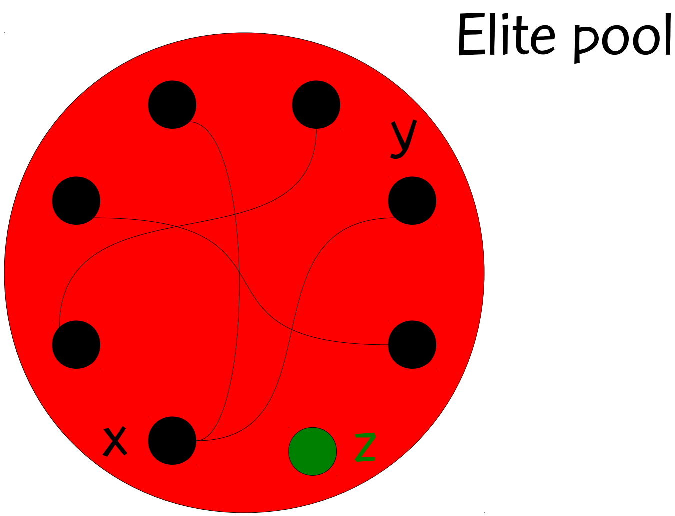
Start with current elite set.

While there is a pair {x,y} of pool solutions that has not yet been relinked:

Relink the pair

$z = PR(x,y)$

Evolutionary path-relinking (EvPR)



Start with current elite set.

While there is a pair {x,y} of pool solutions that has not yet been relinked:

Relink the pair

$z = PR(x,y)$

and attempt to insert z into the pool, replacing some other pool solution.

GRASP with evolutionary path-relinking

As post-optimization

Repeat GRASP with PR loop

1) Construct greedy randomized
2) Local search
3) Path-relinking
4) Update pool

Evolutionary-PR

During GRASP + PR

Repeat outer loop

Repeat inner loop

1) Construct greedy randomized
2) Local search
3) Path-relinking
4) Update pool

Evolutionary-PR

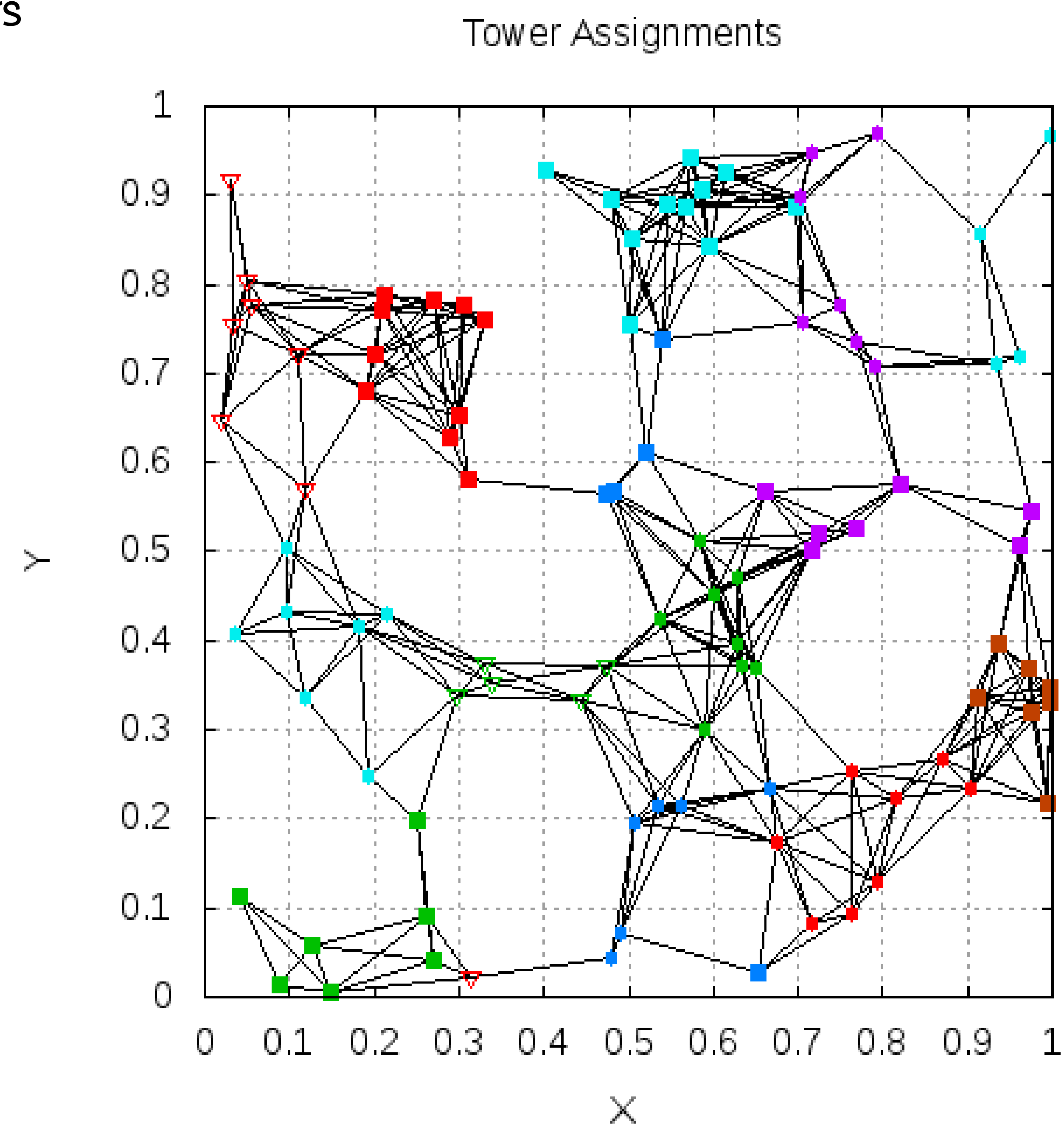
(Resende & Werneck, 2004, 2006)

Experiments with GRASP with evPR for HMP

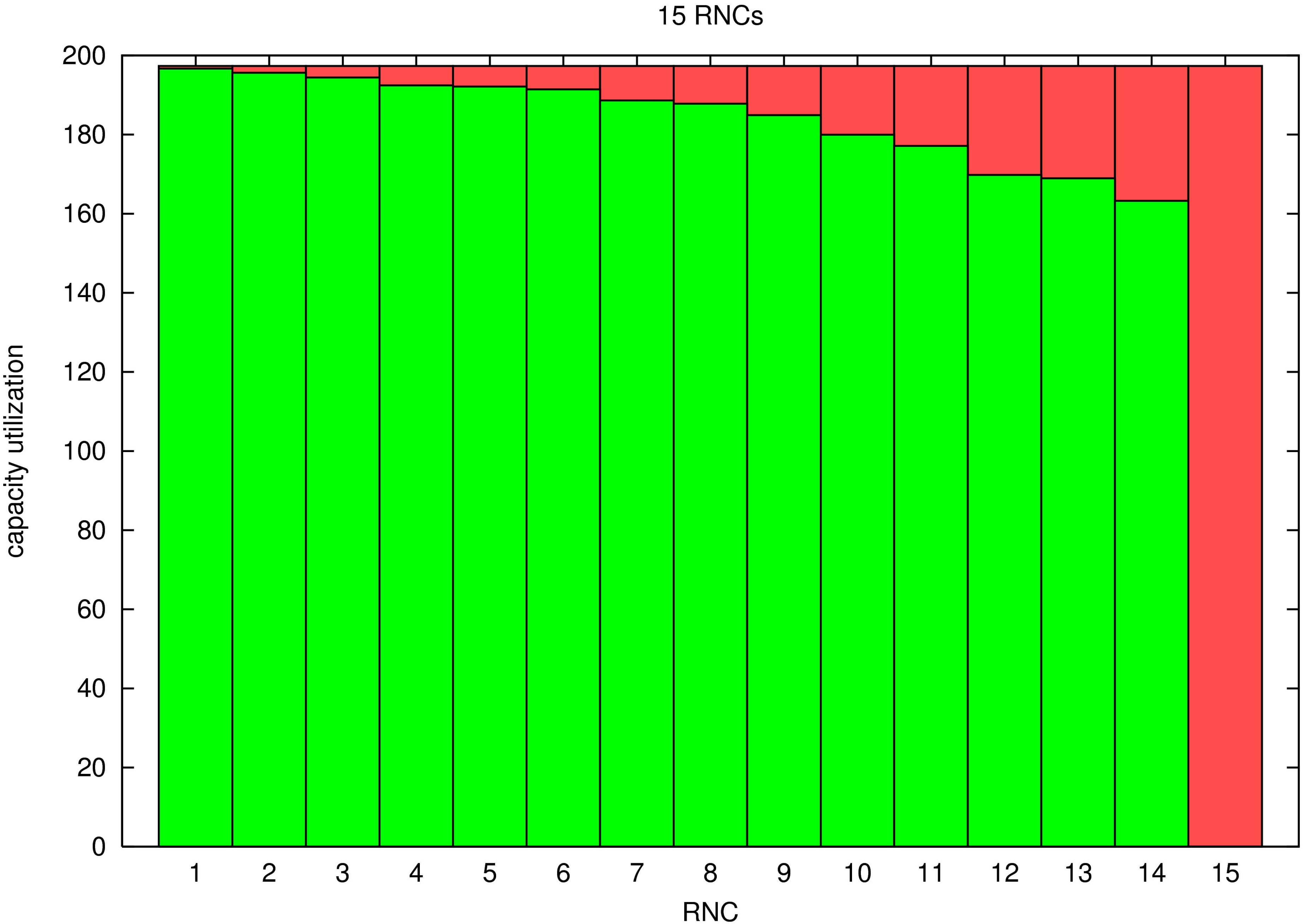
Your world. Delivered.

100 towers

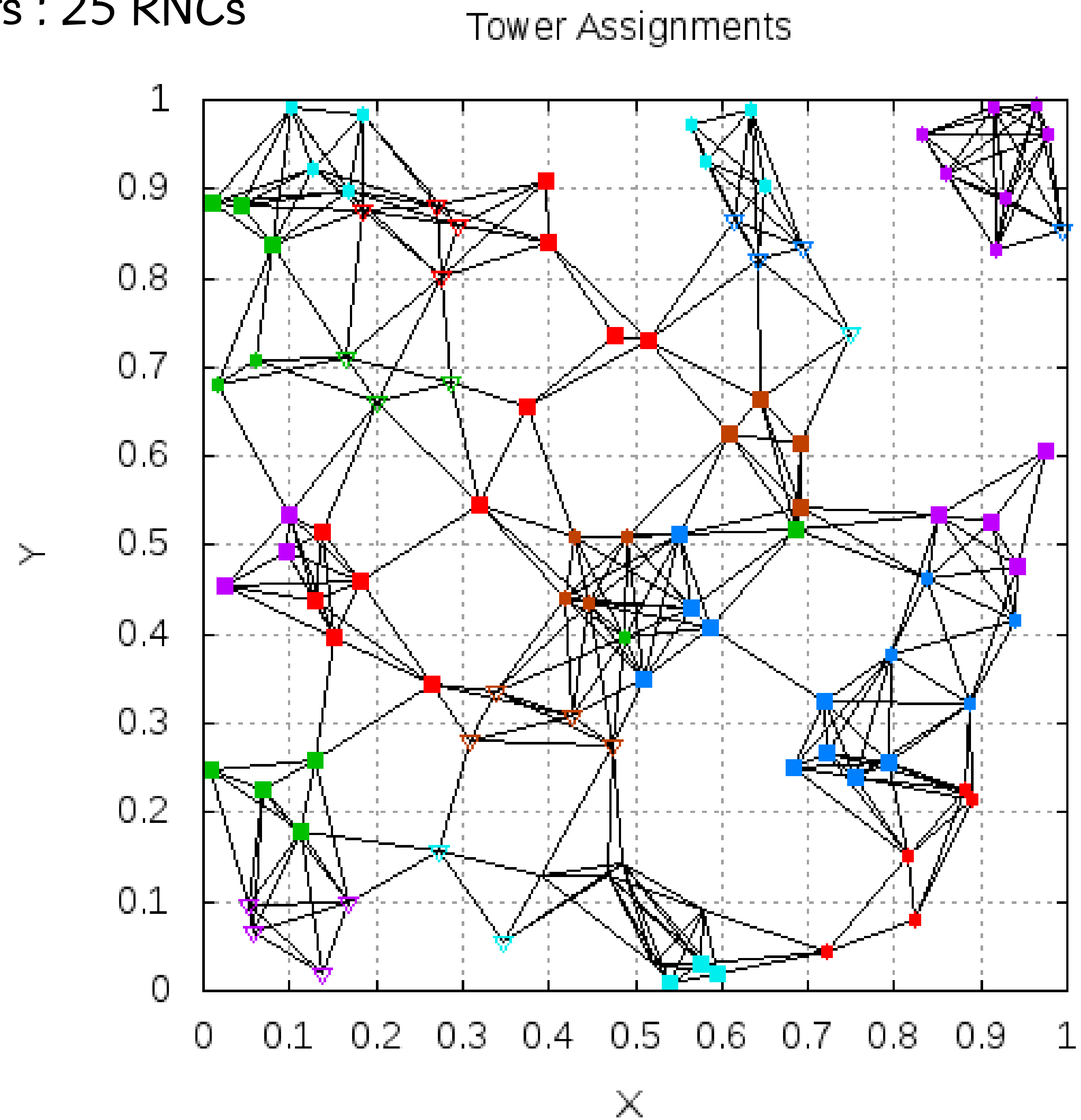
15 RNCs



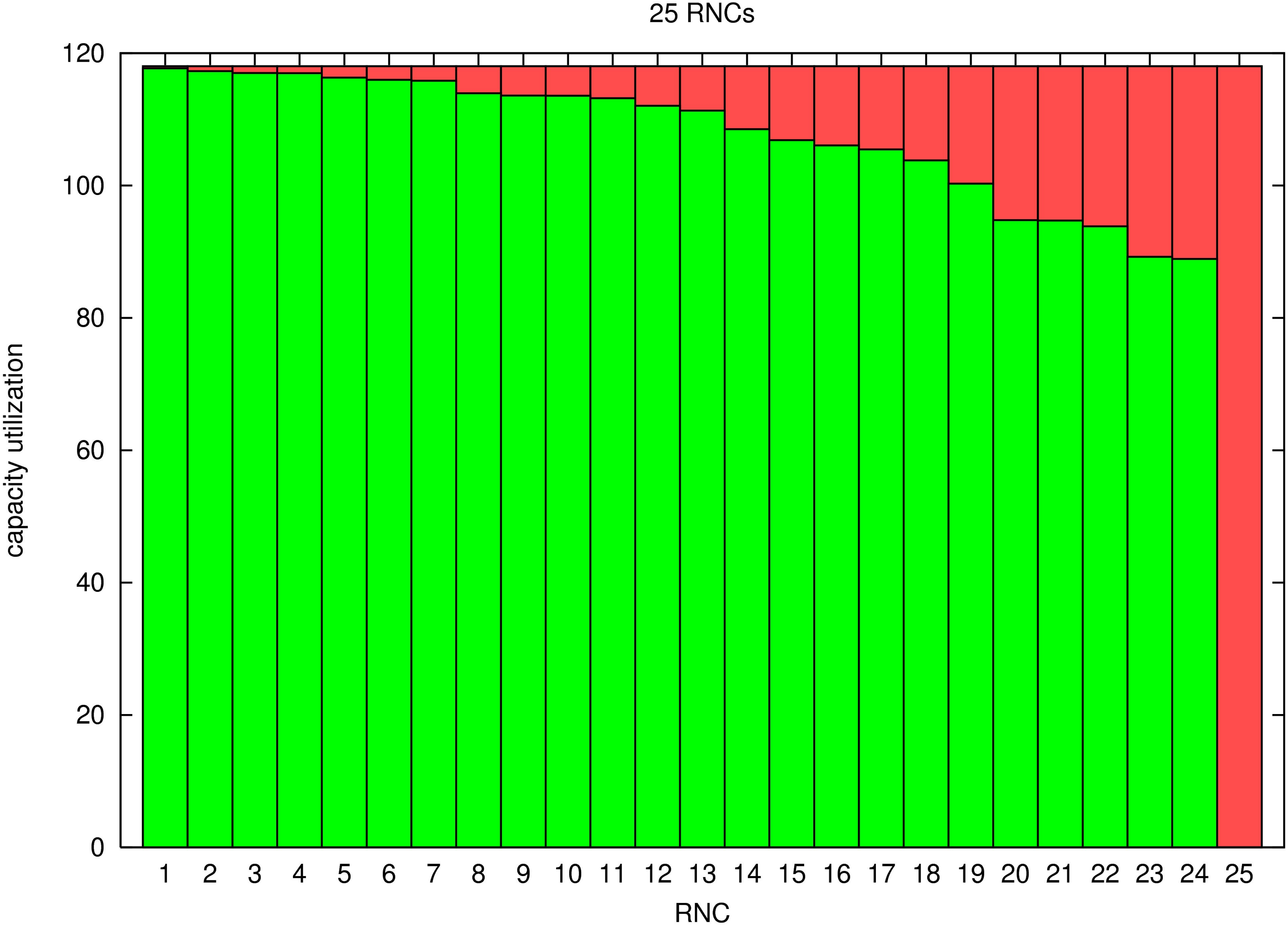
100 towers : 15 RNCs



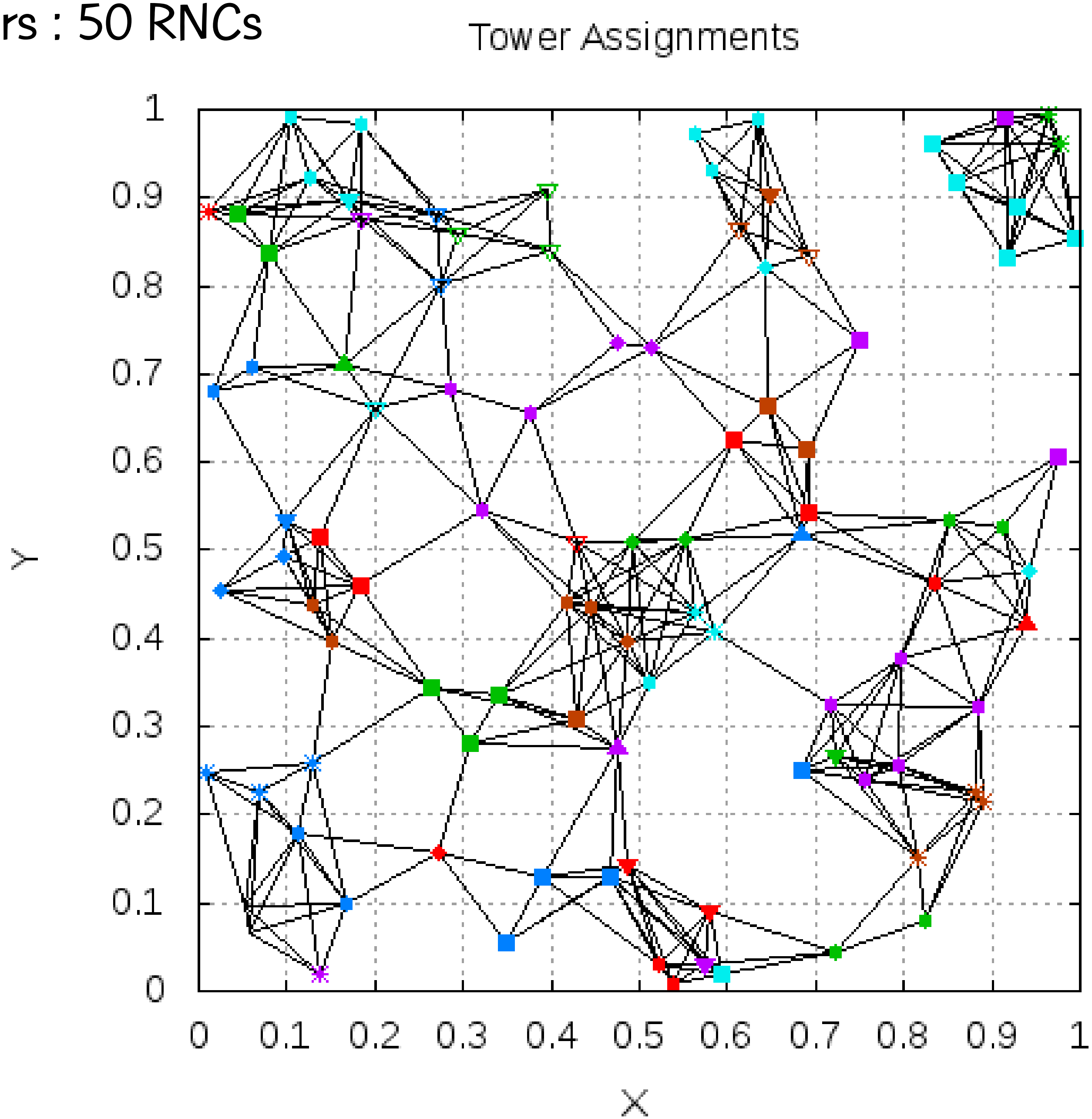
100 towers : 25 RNCs



100 towers : 25 RNCs

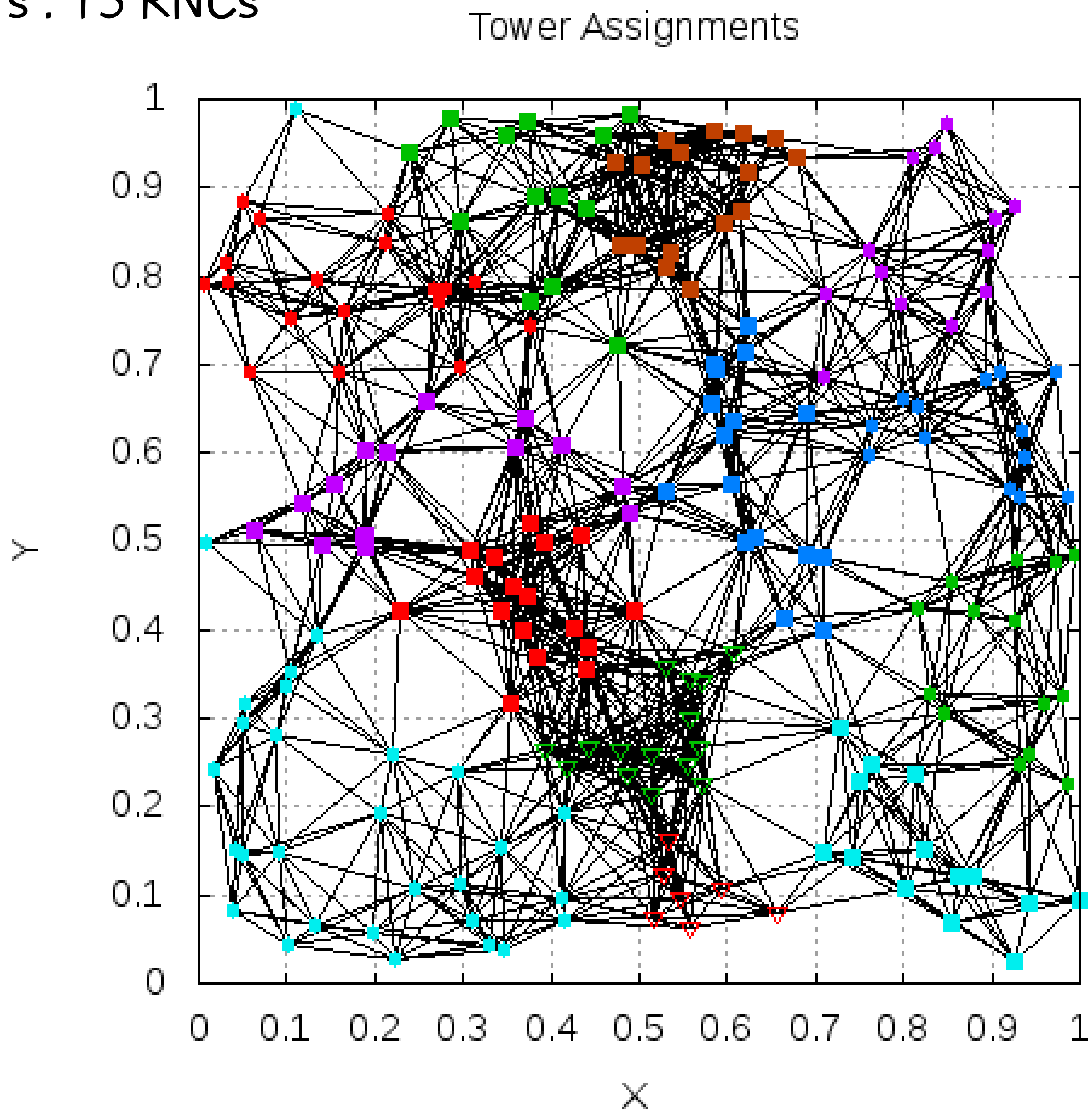


100 towers : 50 RNCs

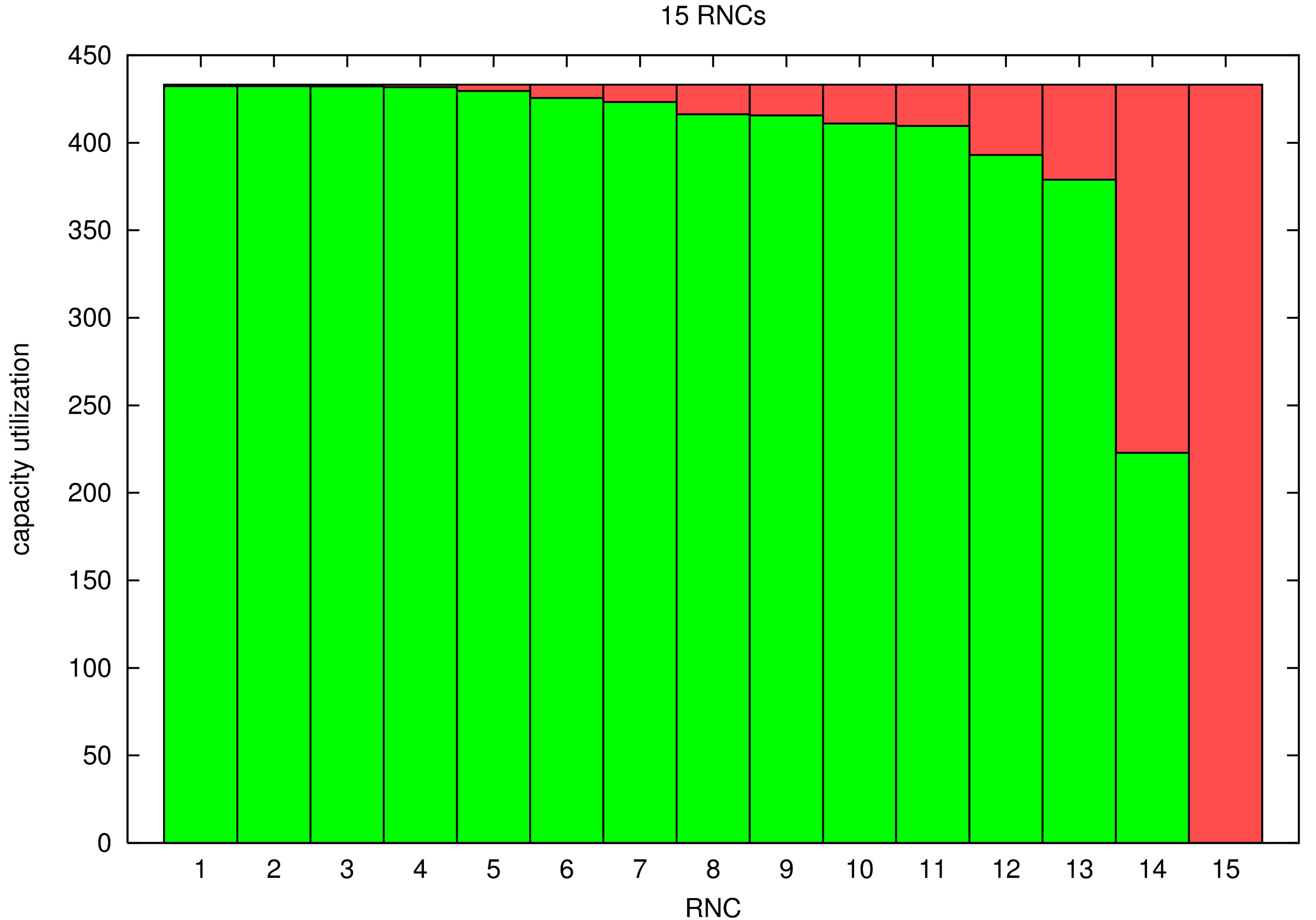


100 towers : 50 RNCs

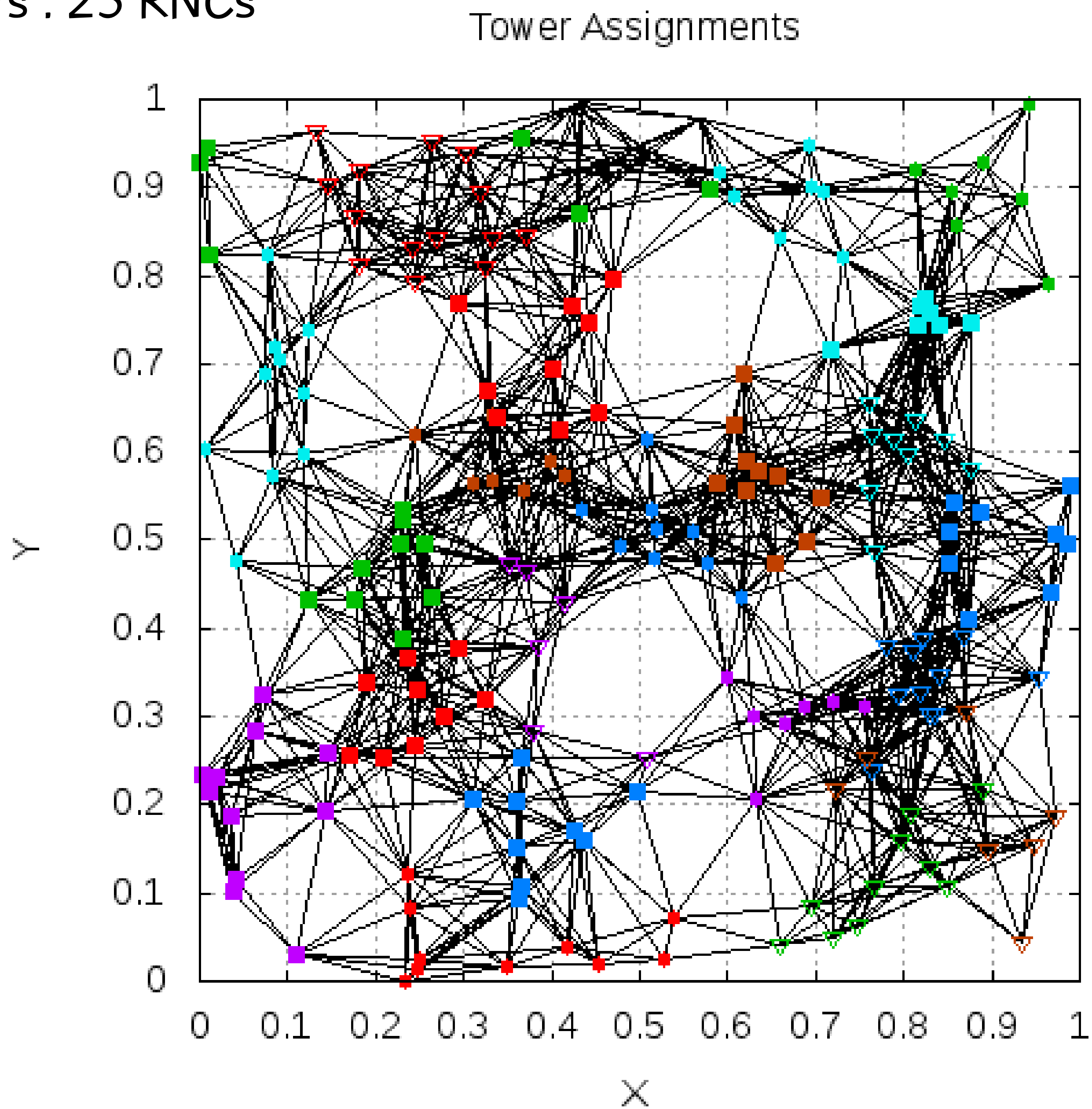
200 towers : 15 RNCs



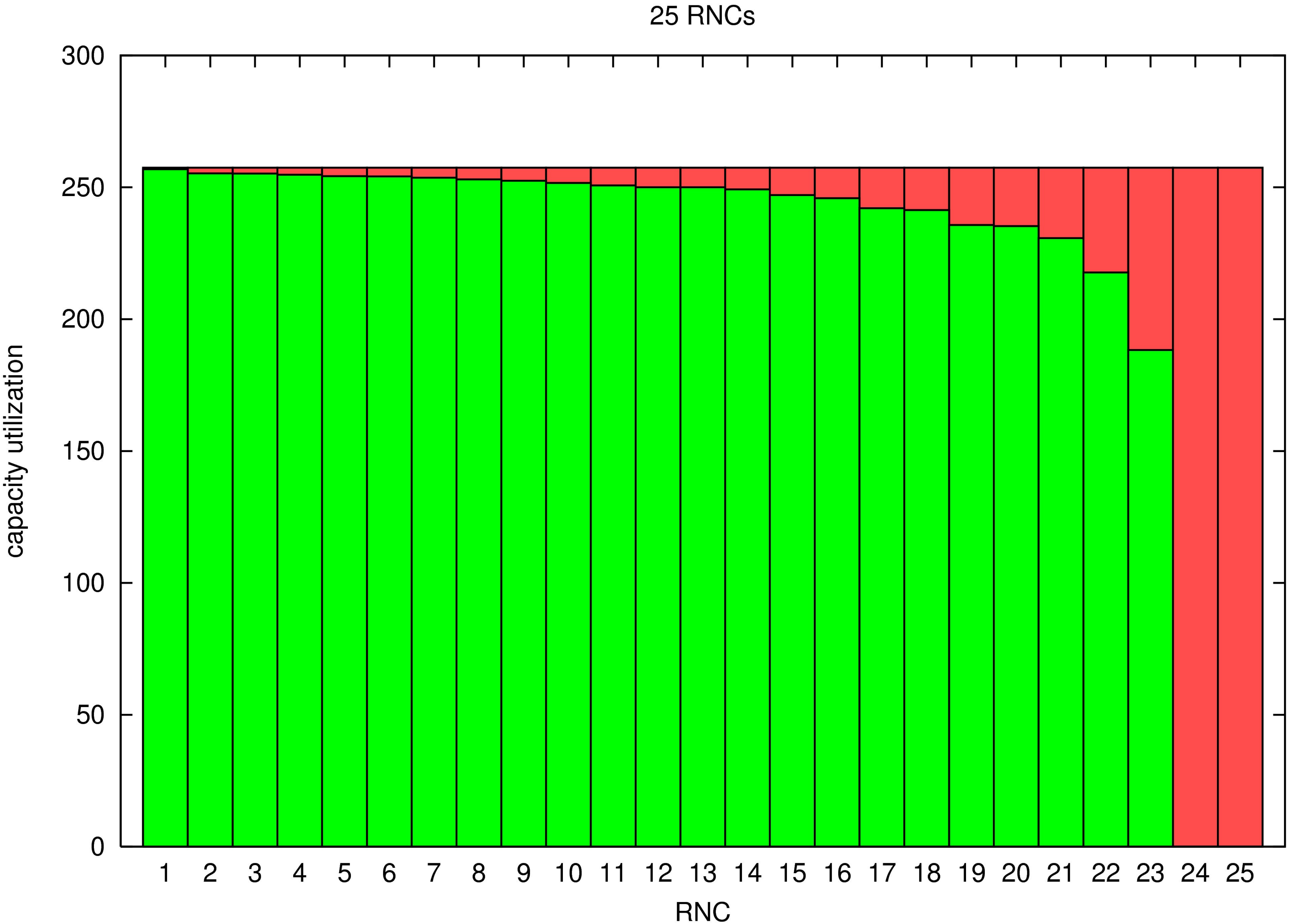
200 towers : 15 RNCs



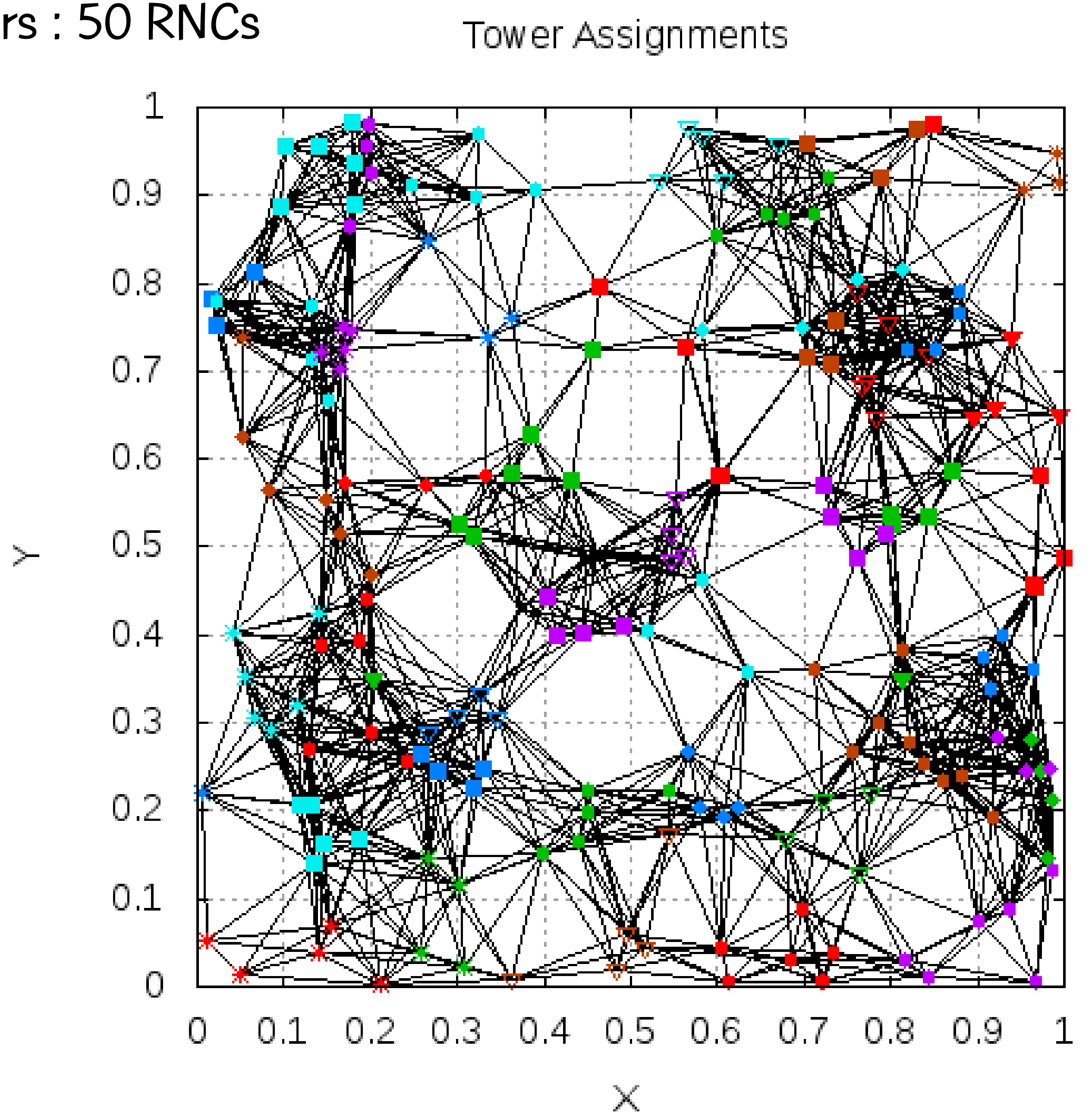
200 towers : 25 RNCs



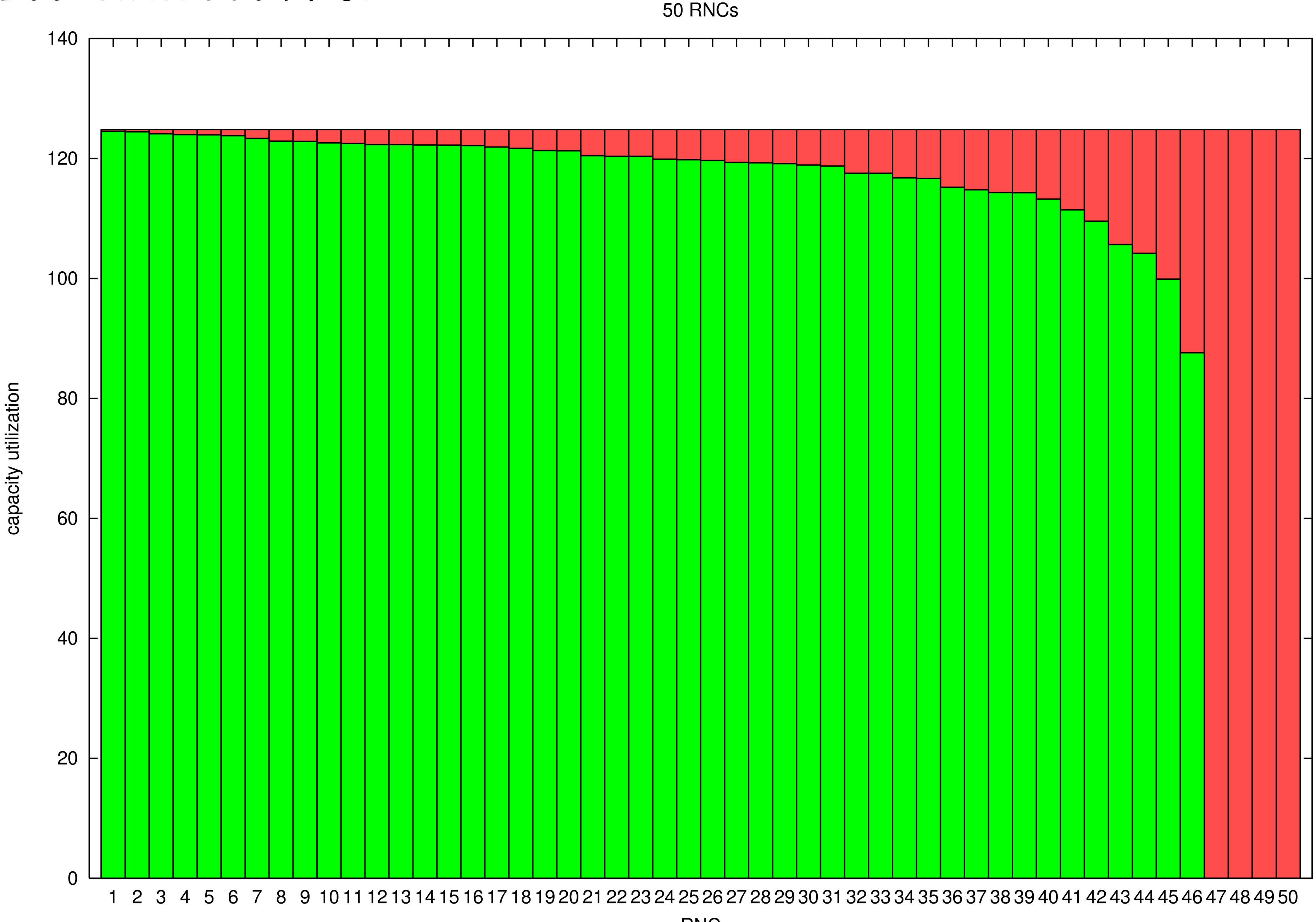
200 towers : 25 RNCs



200 towers : 50 RNCs



200 towers : 50 RNCs

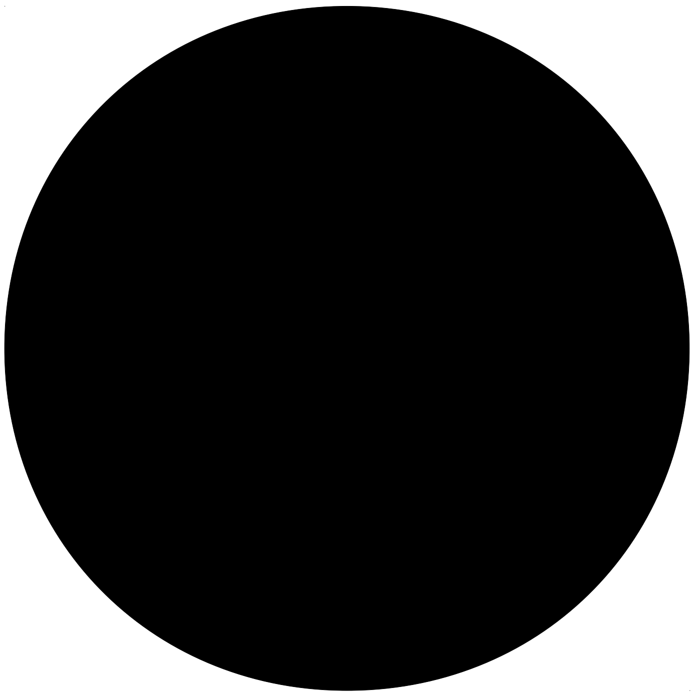


Your world. Delivered.

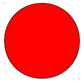
Biased random-key genetic algorithms

Genetic algorithms

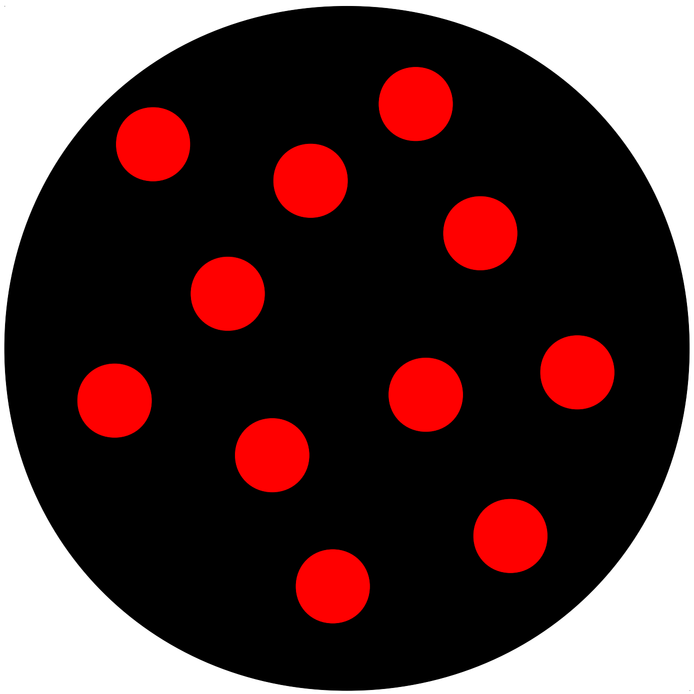
Holland (1975)



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

Individual: solution 

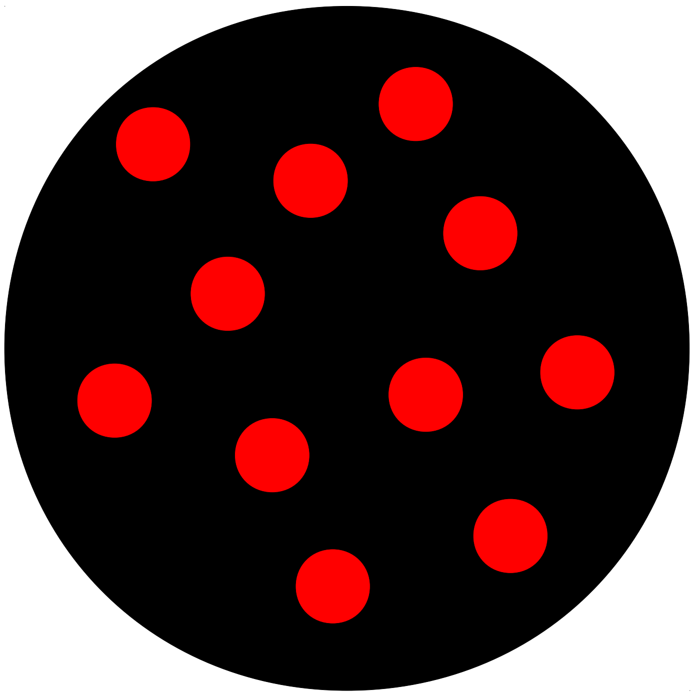
Genetic algorithms



Individual: solution

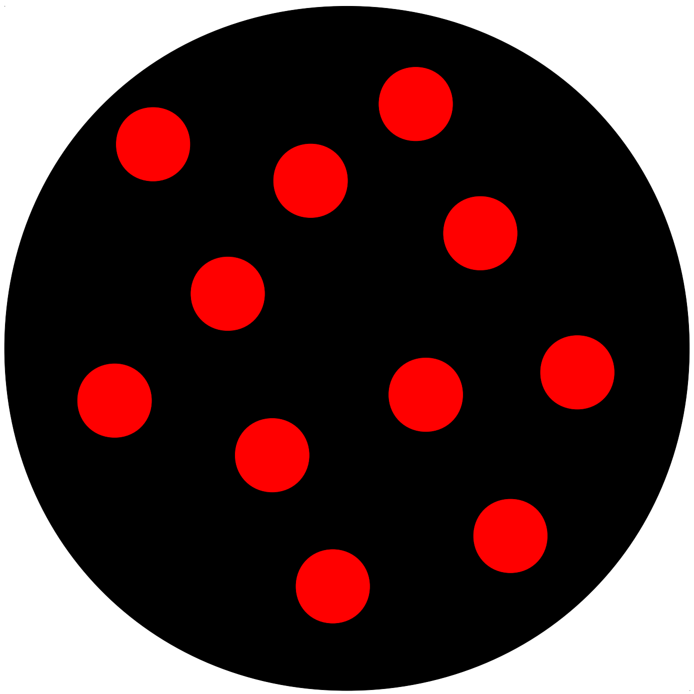
Population: set of fixed number of individuals

Genetic algorithms



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

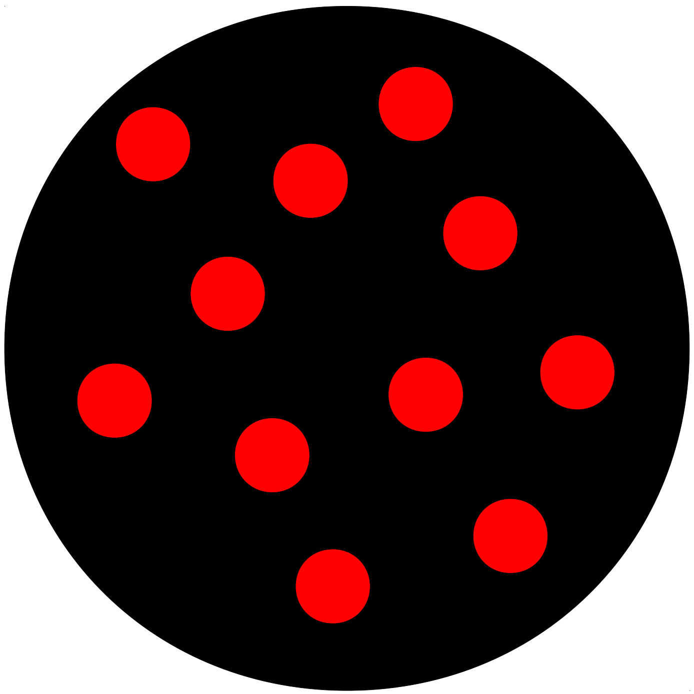
Genetic algorithms



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

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

Genetic algorithms

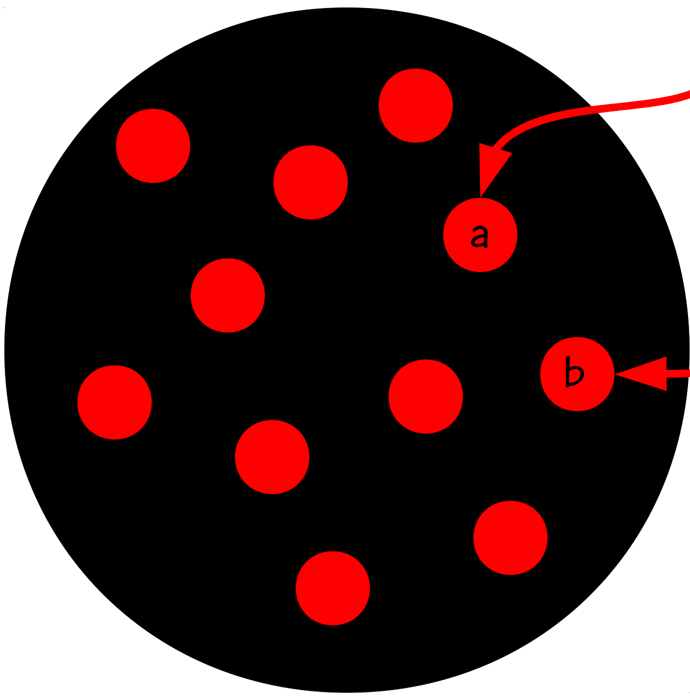


Genetic algorithms evolve population applying the principle of survival of the fittest.

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

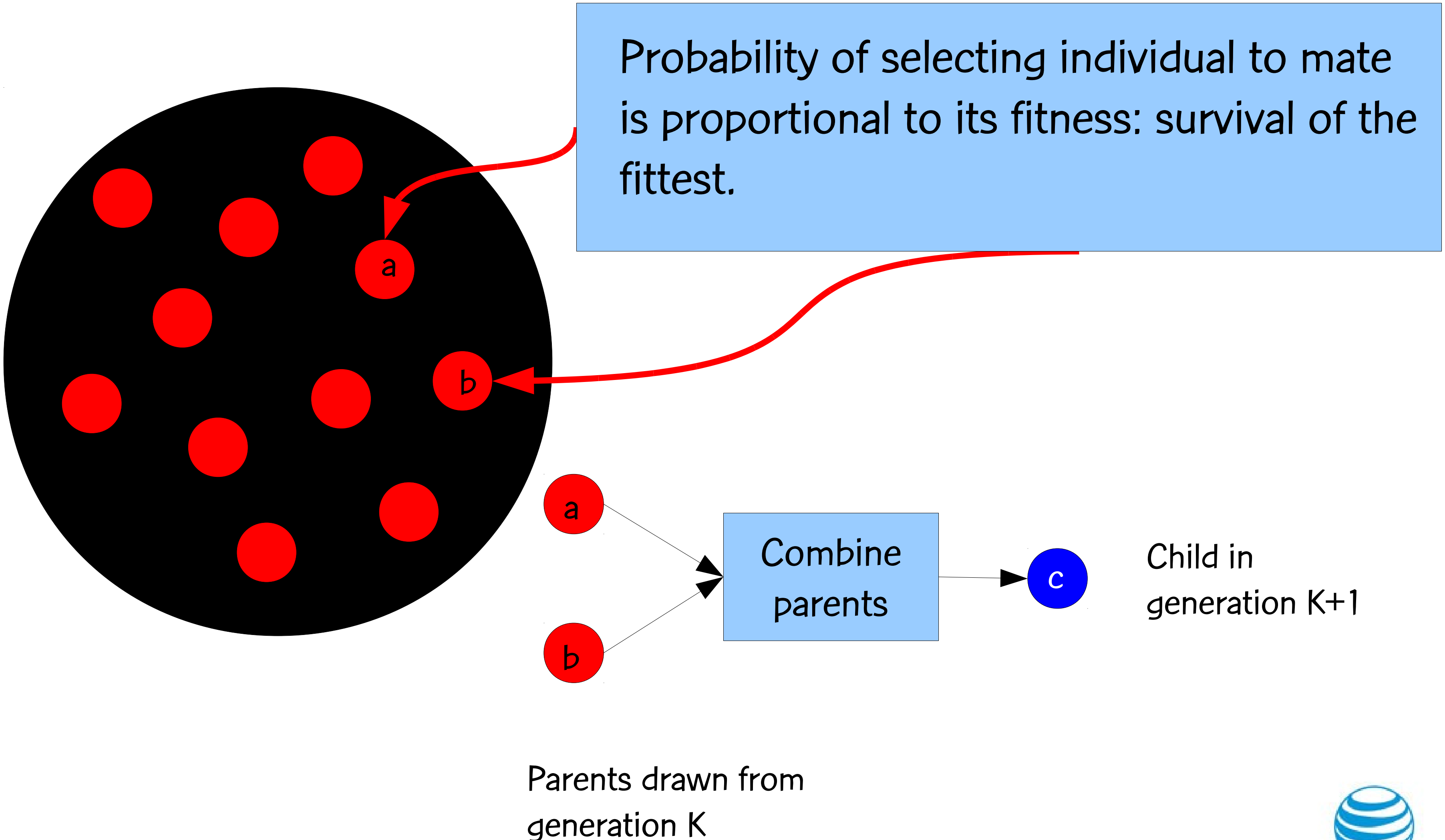
Individuals from one generation are combined to produce offspring that make up next generation.

Genetic algorithms

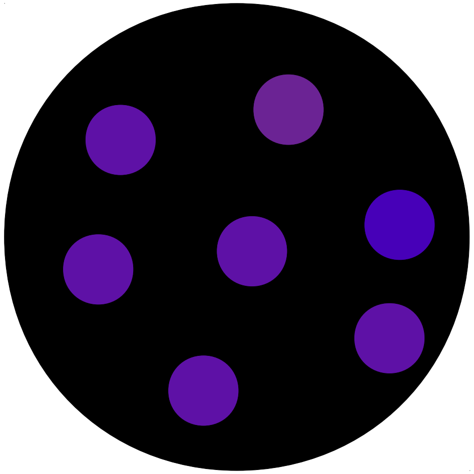


Probability of selecting individual to mate is proportional to its fitness: survival of the fittest.

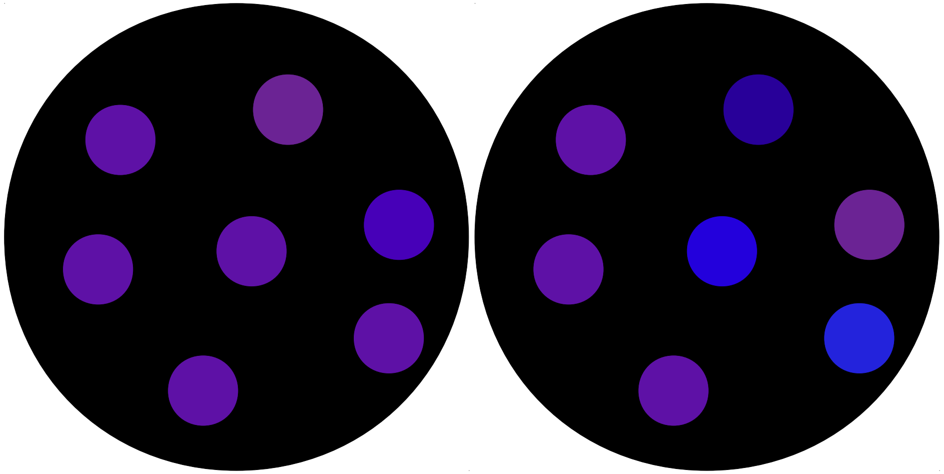
Genetic algorithms



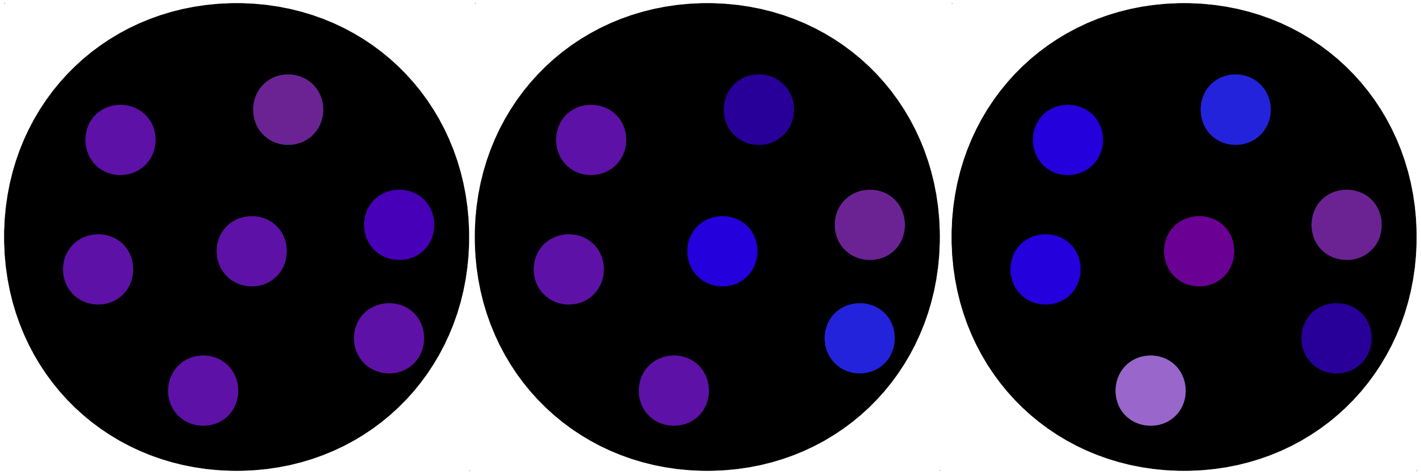
Evolution of solutions



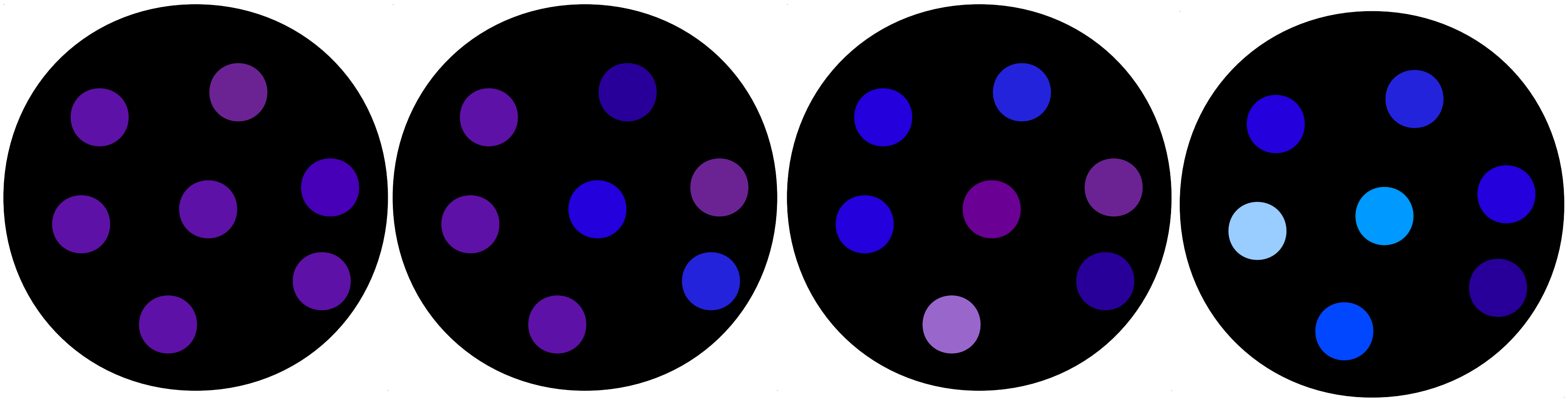
Evolution of solutions



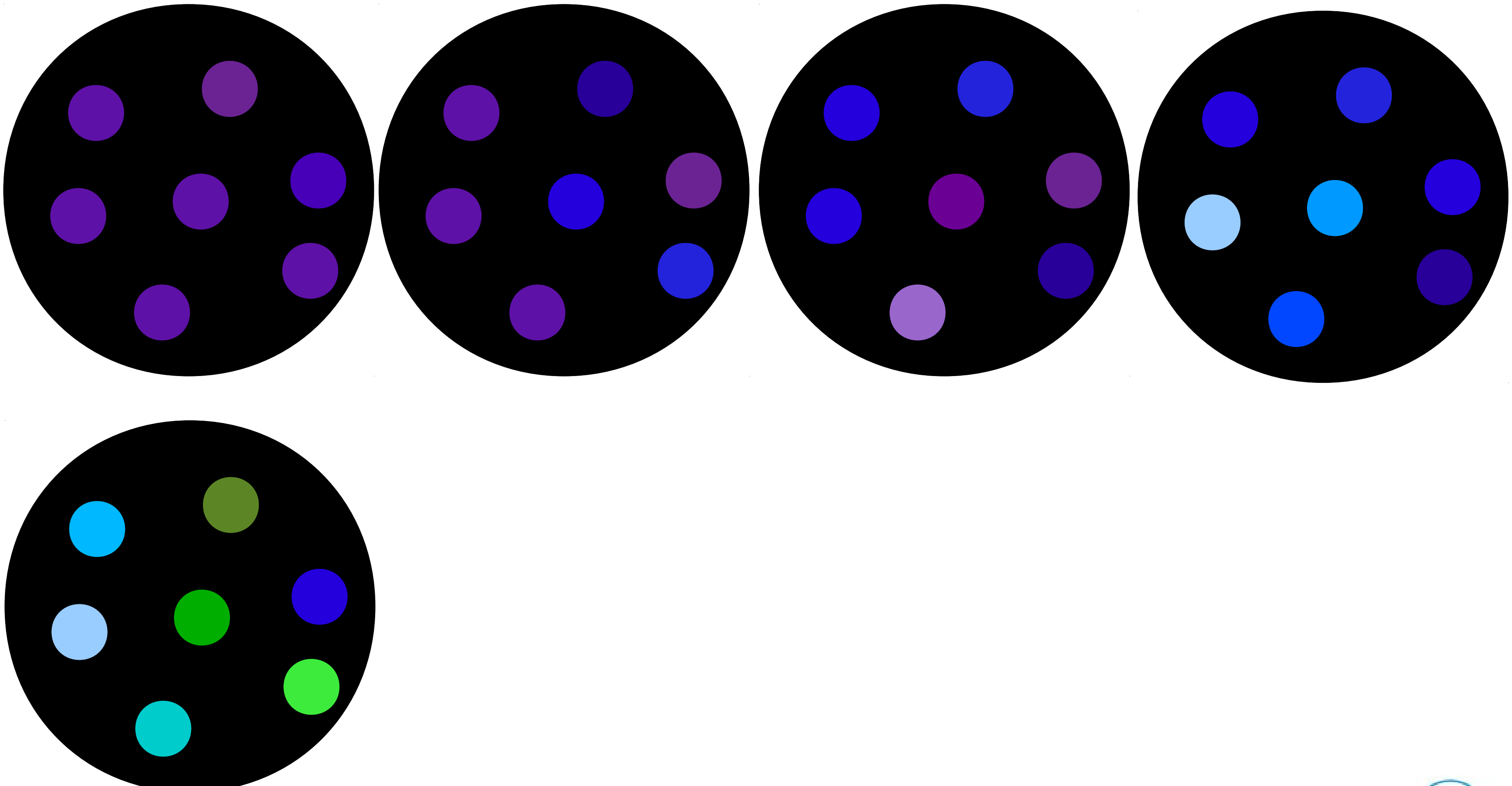
Evolution of solutions



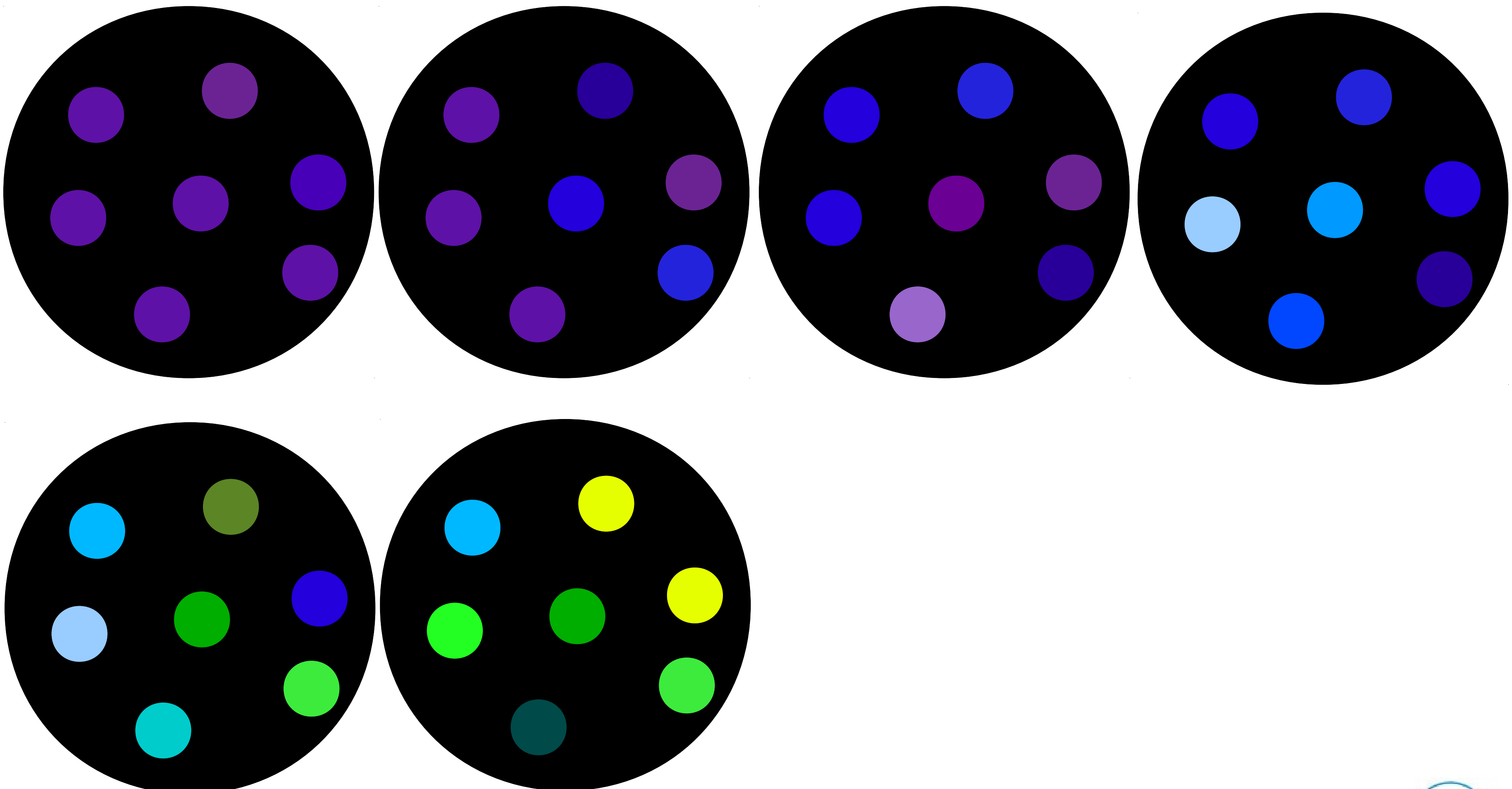
Evolution of solutions



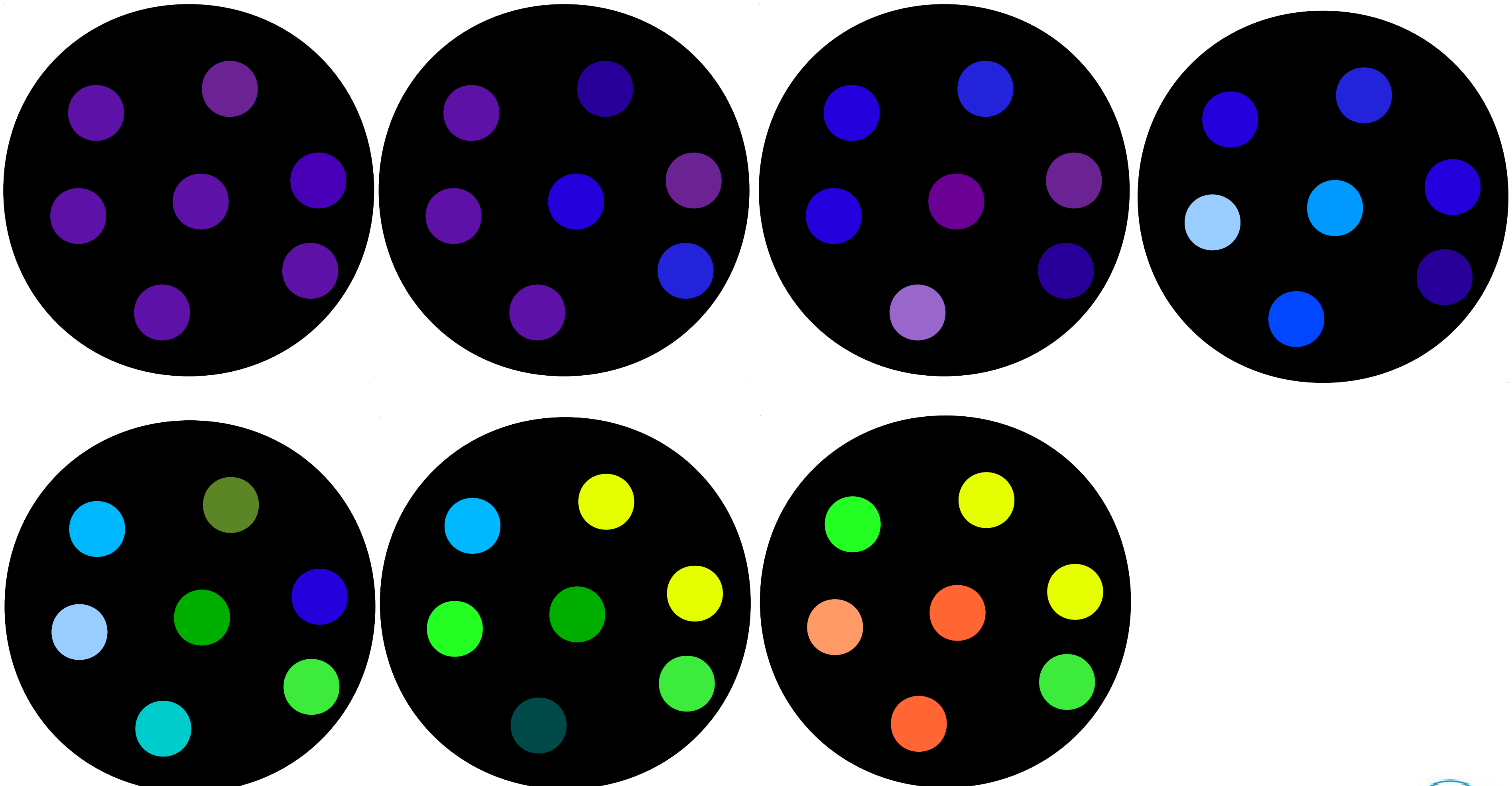
Evolution of solutions



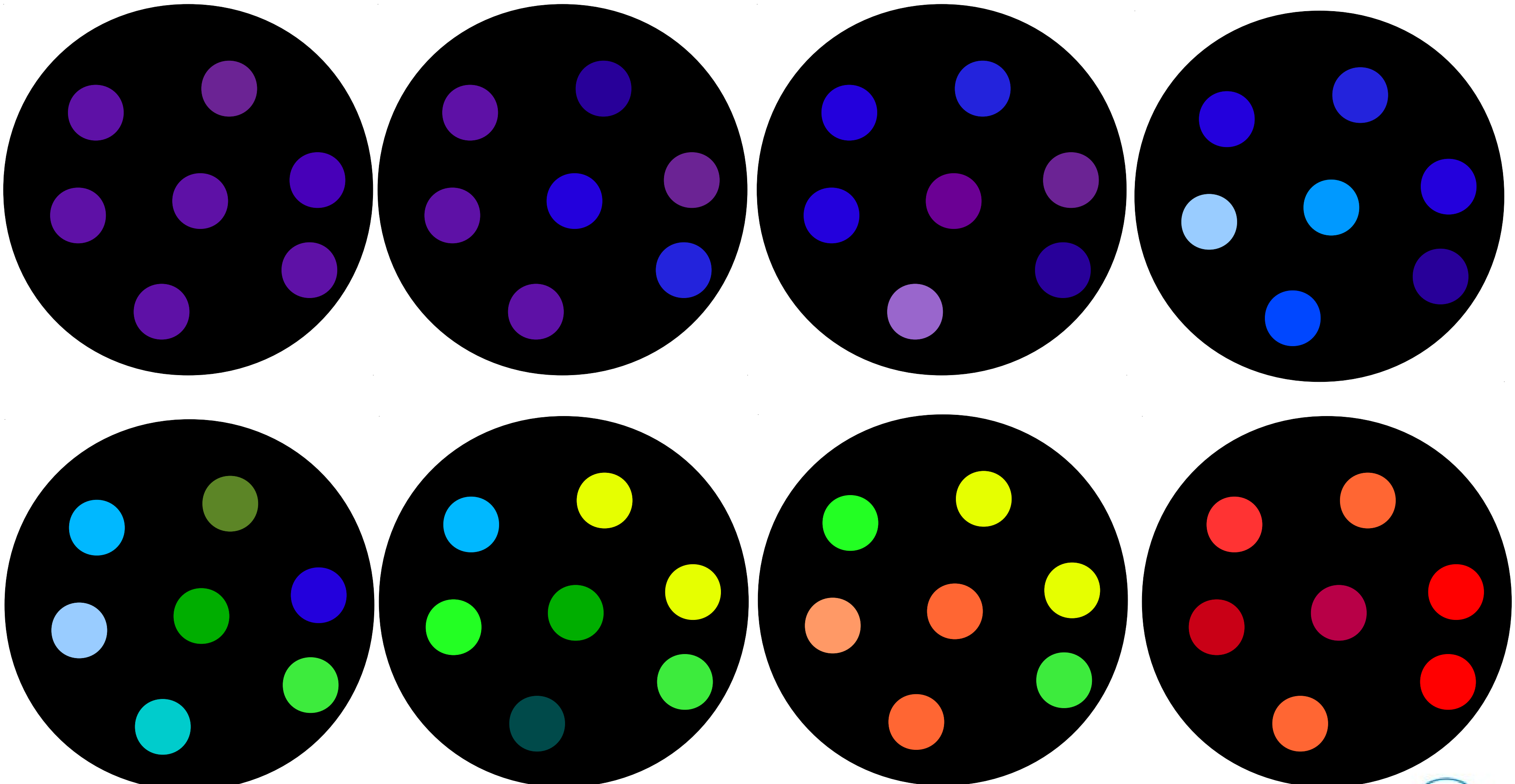
Evolution of solutions



Evolution of solutions

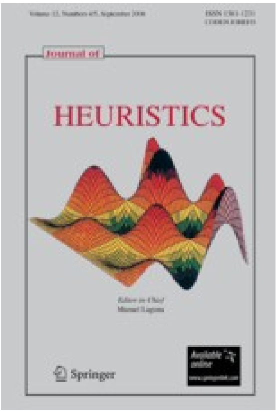


Evolution of solutions



Genetic algorithms with random keys

Survey paper on BRKGA



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.

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

GAs and random keys

- Introduced by Bean (1994) for sequencing problems.

GAs and random keys

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

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

GAs and random keys

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

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

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

Sequence: 4 – 2 – 1 – 3 – 5

GAs and random keys

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

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

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

GAs and random keys

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

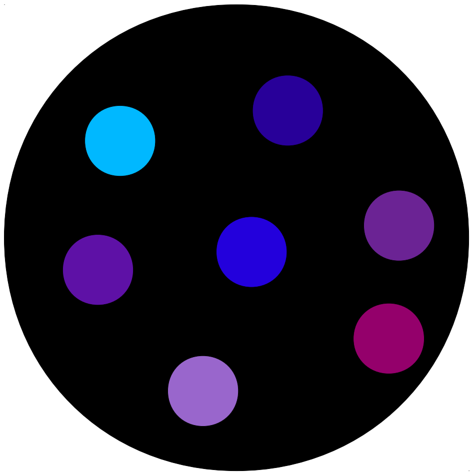
a = (0.25, 0.19, 0.67, 0.05, 0.89)
b = (0.63, 0.90, 0.76, 0.93, 0.08)
c = (0.25, 0.90, 0.76, 0.05, 0.89)

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

at&t Your world. Delivered.

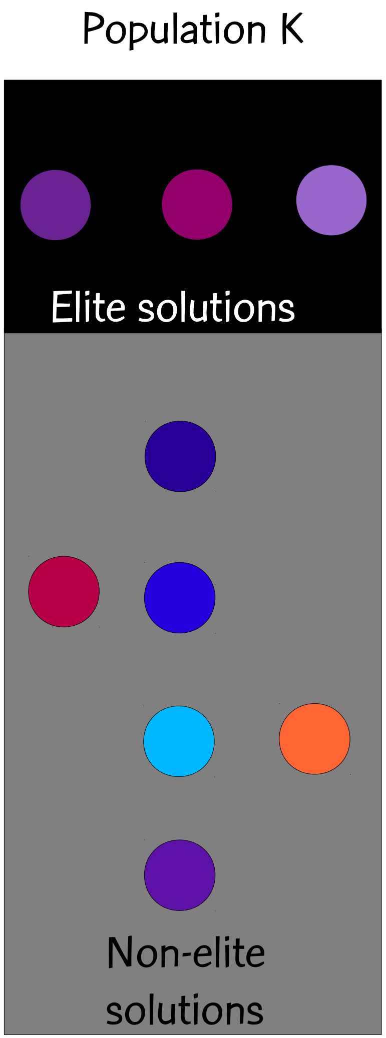
GAs and random keys

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].



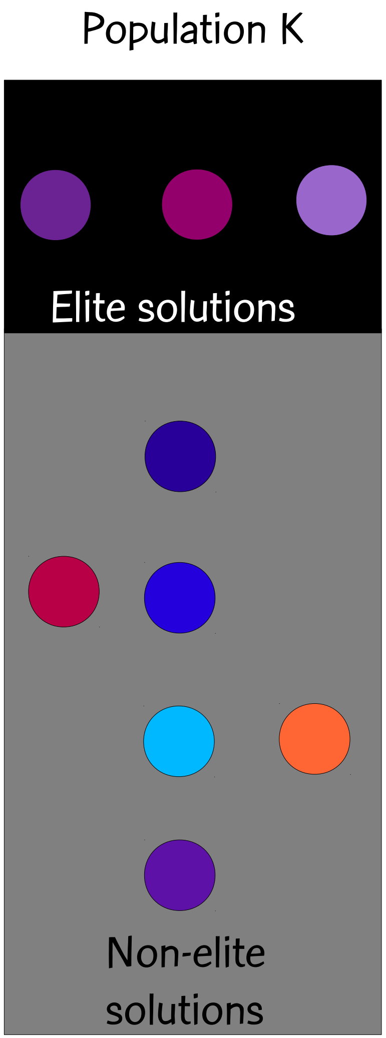
GAs and random keys

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



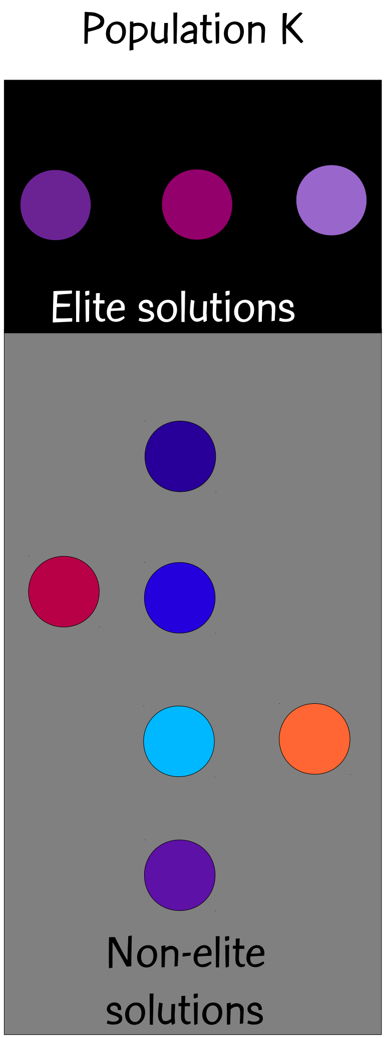
GAs and random keys

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



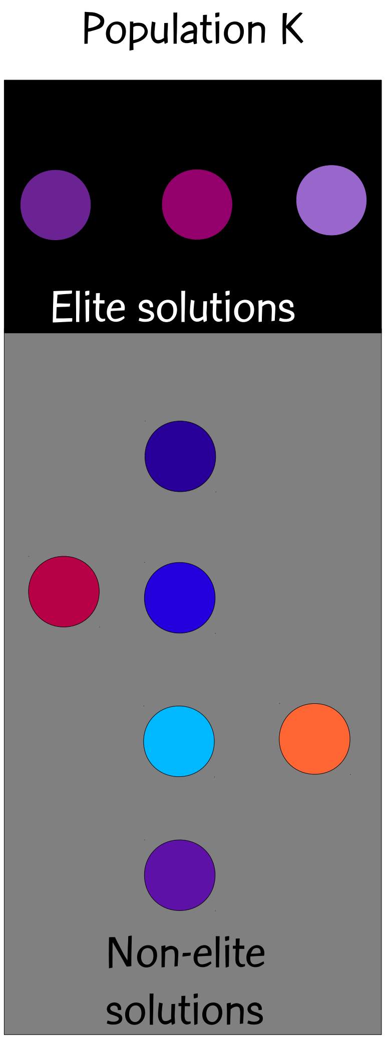
GAs and random keys

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



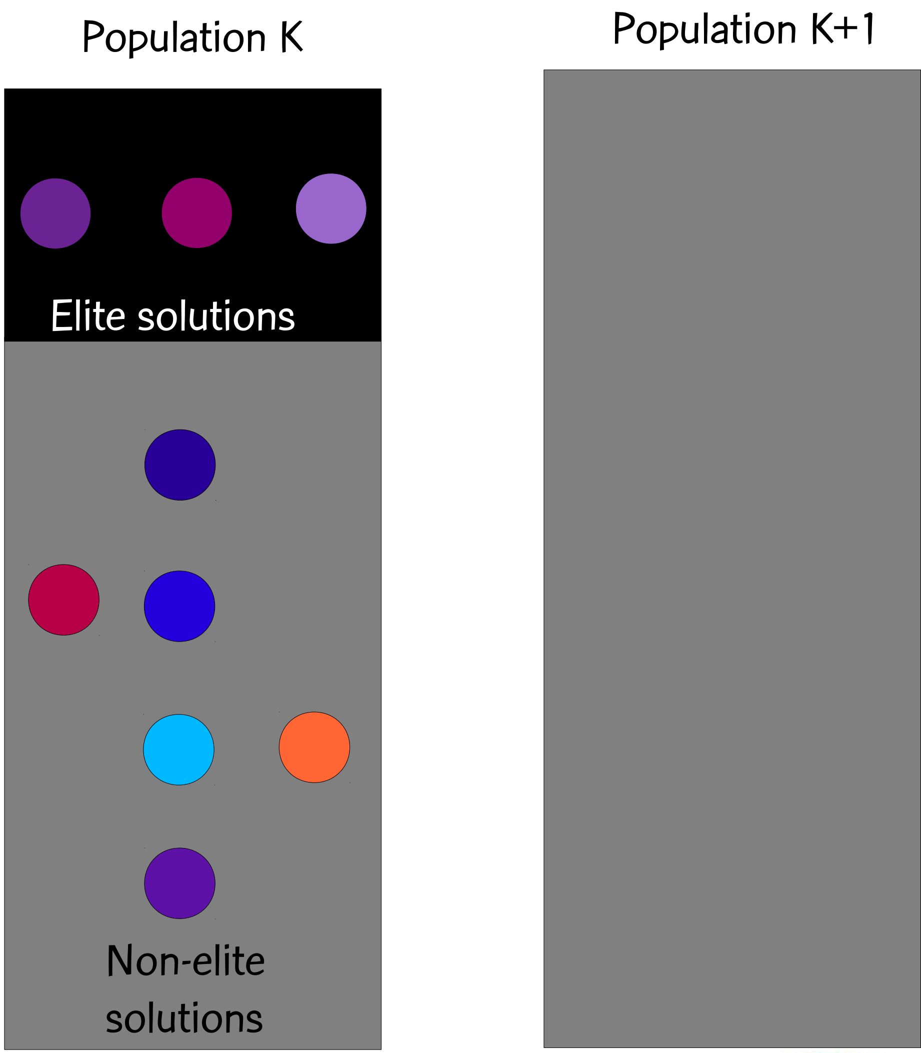
GAs and random keys

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

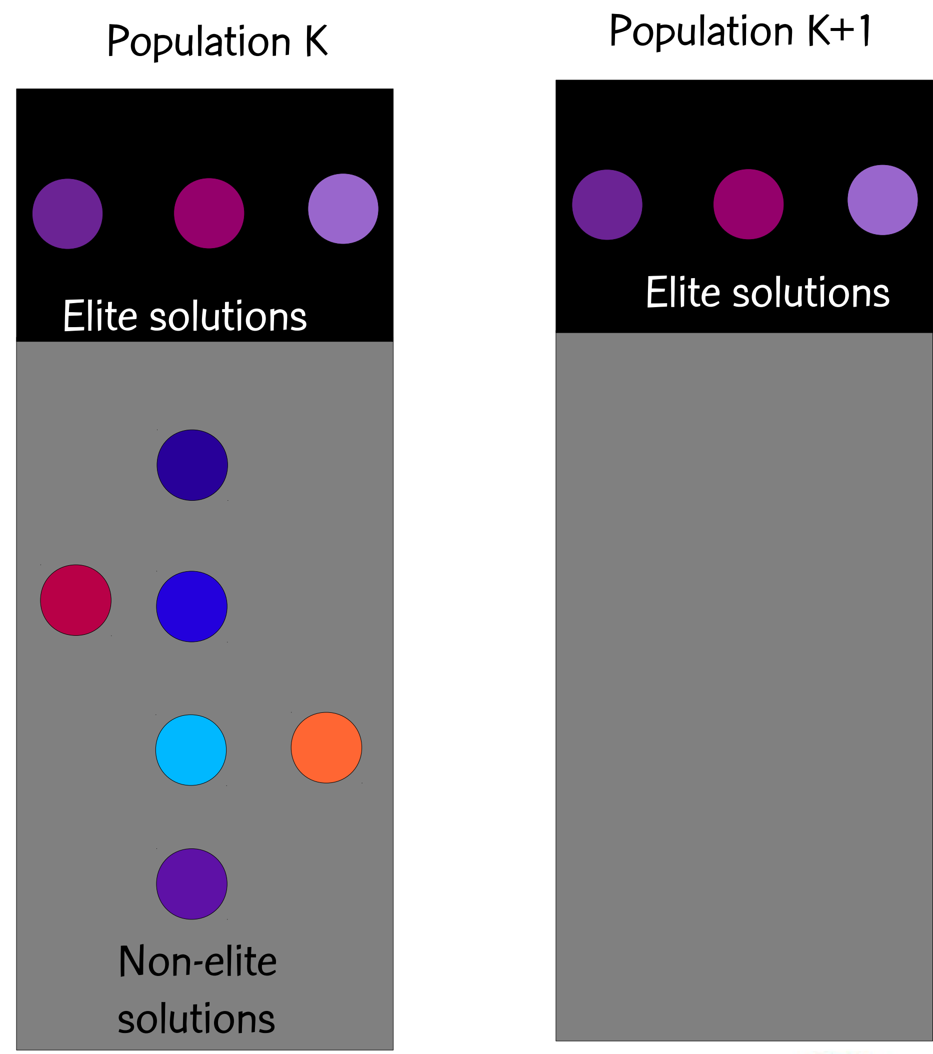
Evolutionary dynamics



GAs and random keys

Evolutionary dynamics

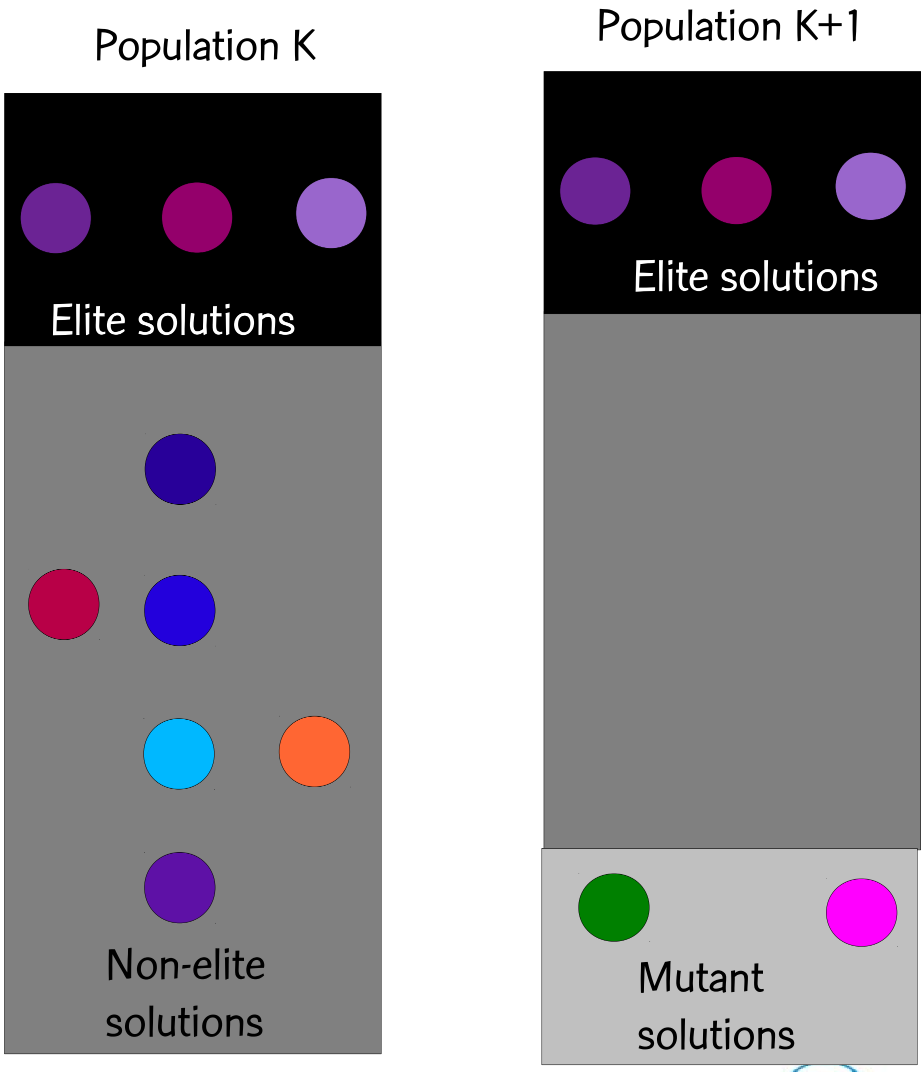
– Copy elite solutions from population K to population K+1



GAs and random keys

Evolutionary dynamics

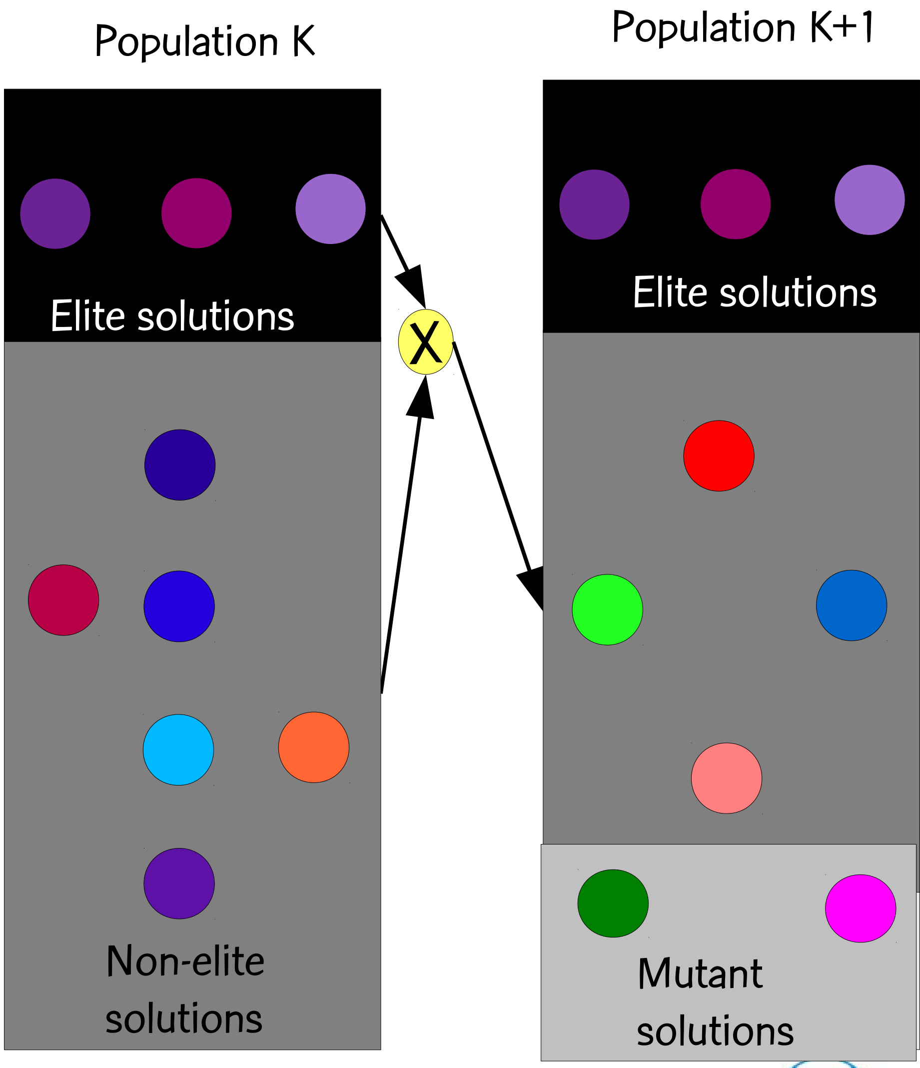
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1



Biased random key GA

Evolutionary dynamics

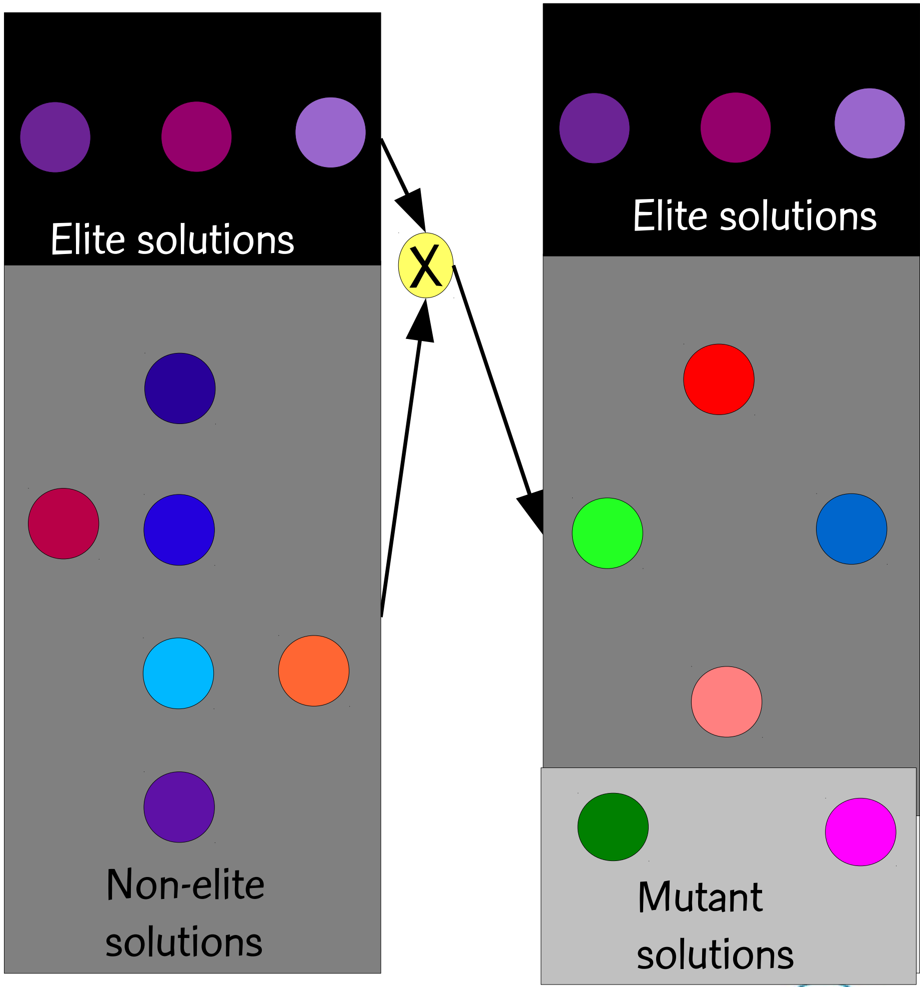
- Copy elite solutions from population K to population K+1
- Add R random solutions (mutants) to population K+1
- While K+1-th population < P
  - RANDOM-KEY GA: Use any two solutions in population K to produce child in population K+1. Mates are chosen at random.



Biased random key GA

Evolutionary dynamics

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

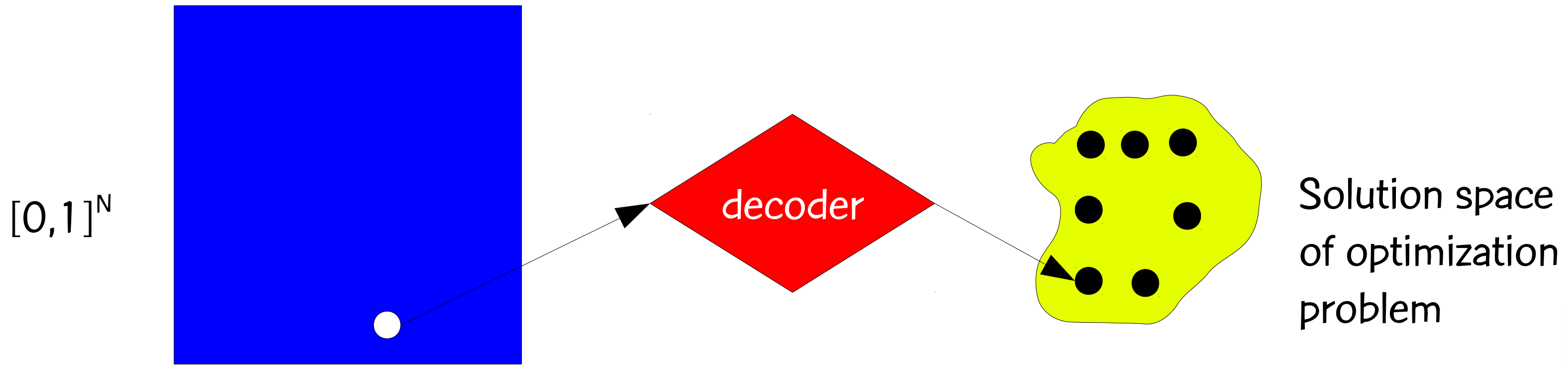


Observations

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

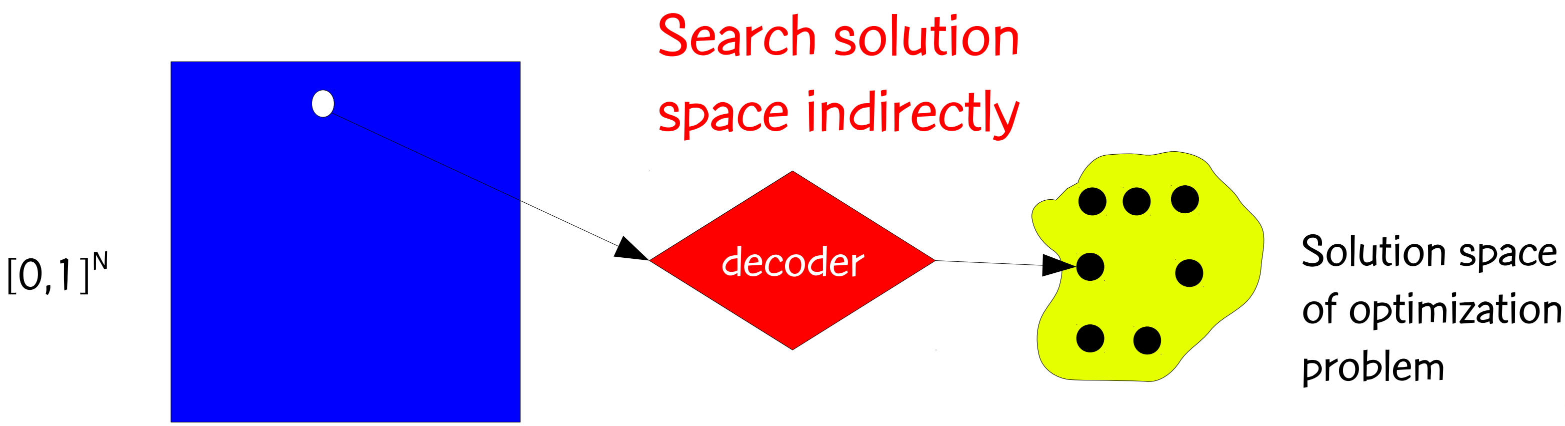
Decoders

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



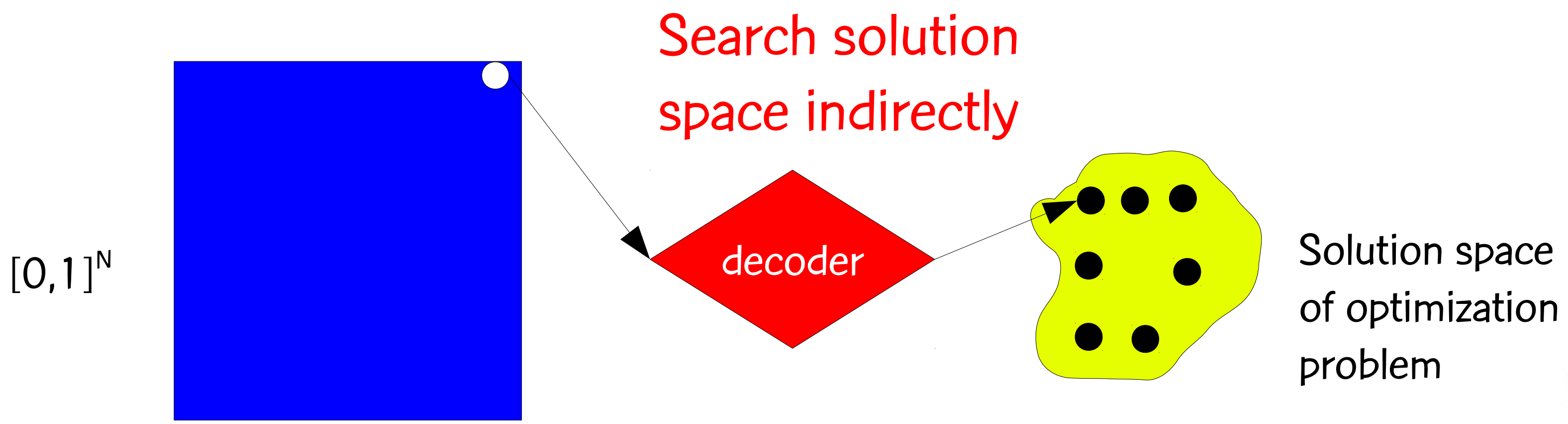
Decoders

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



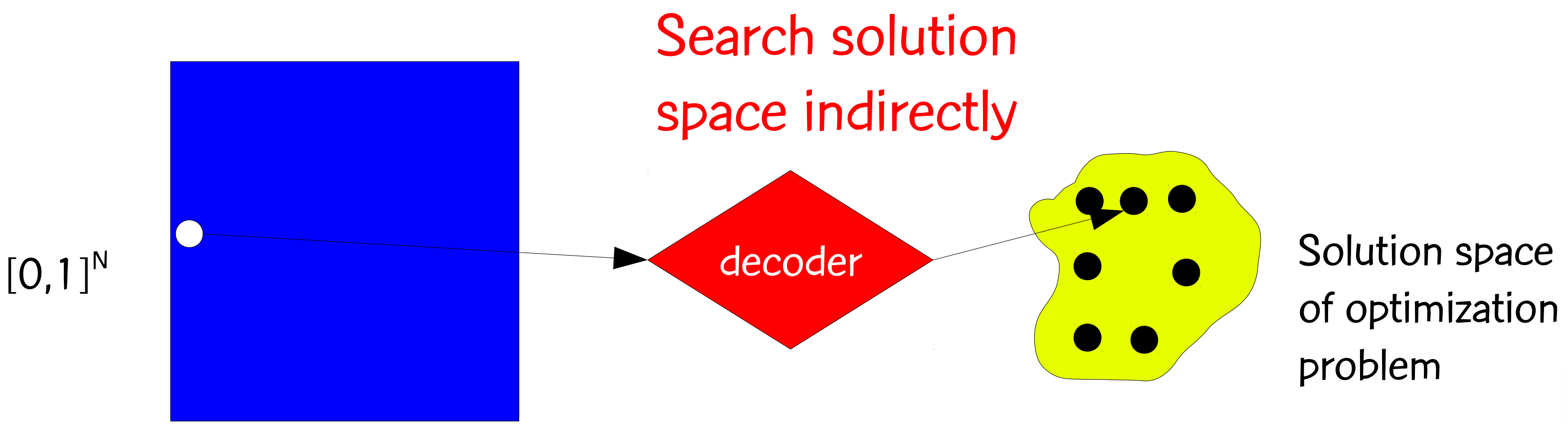
Decoders

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

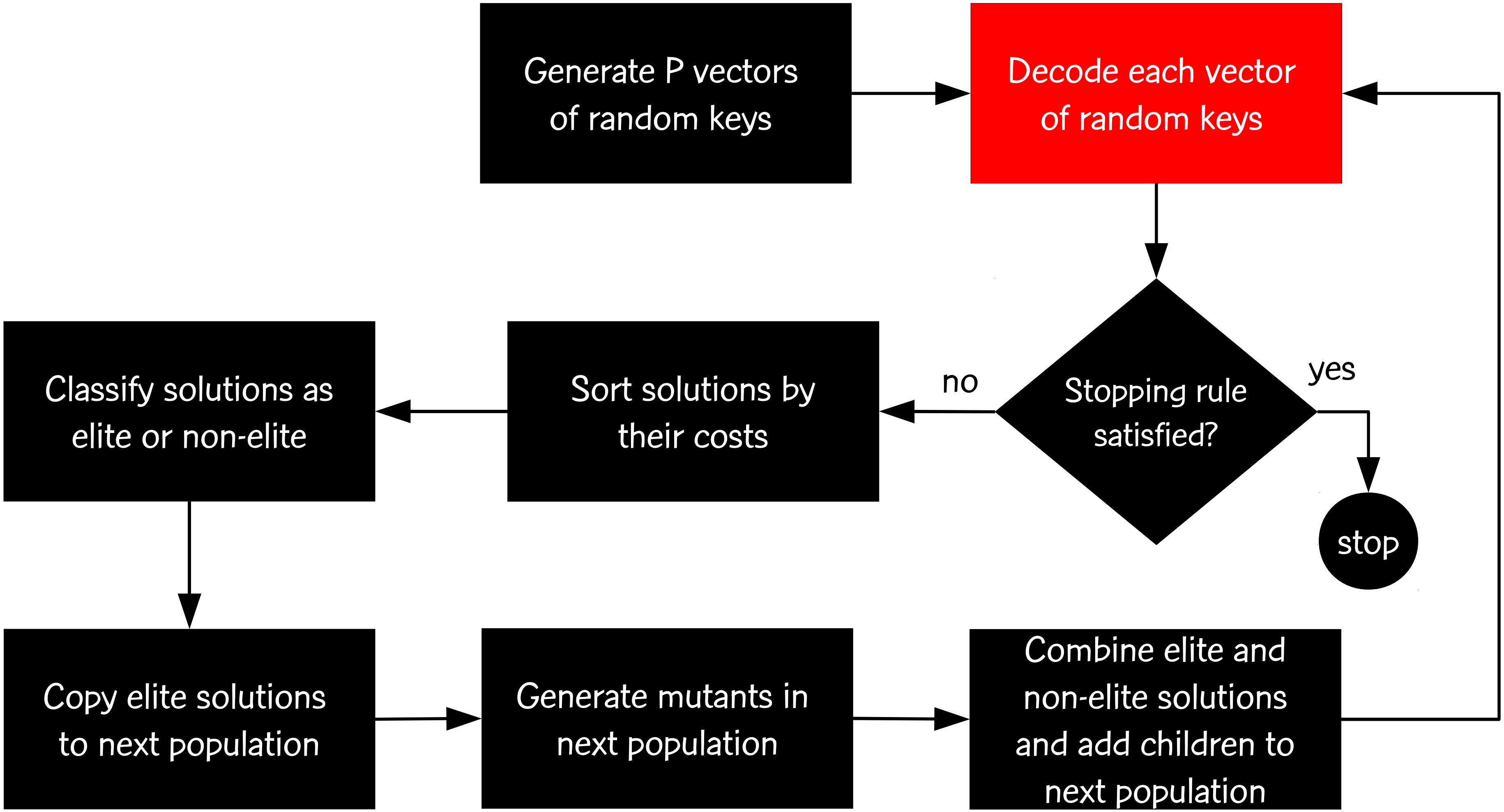


Decoders

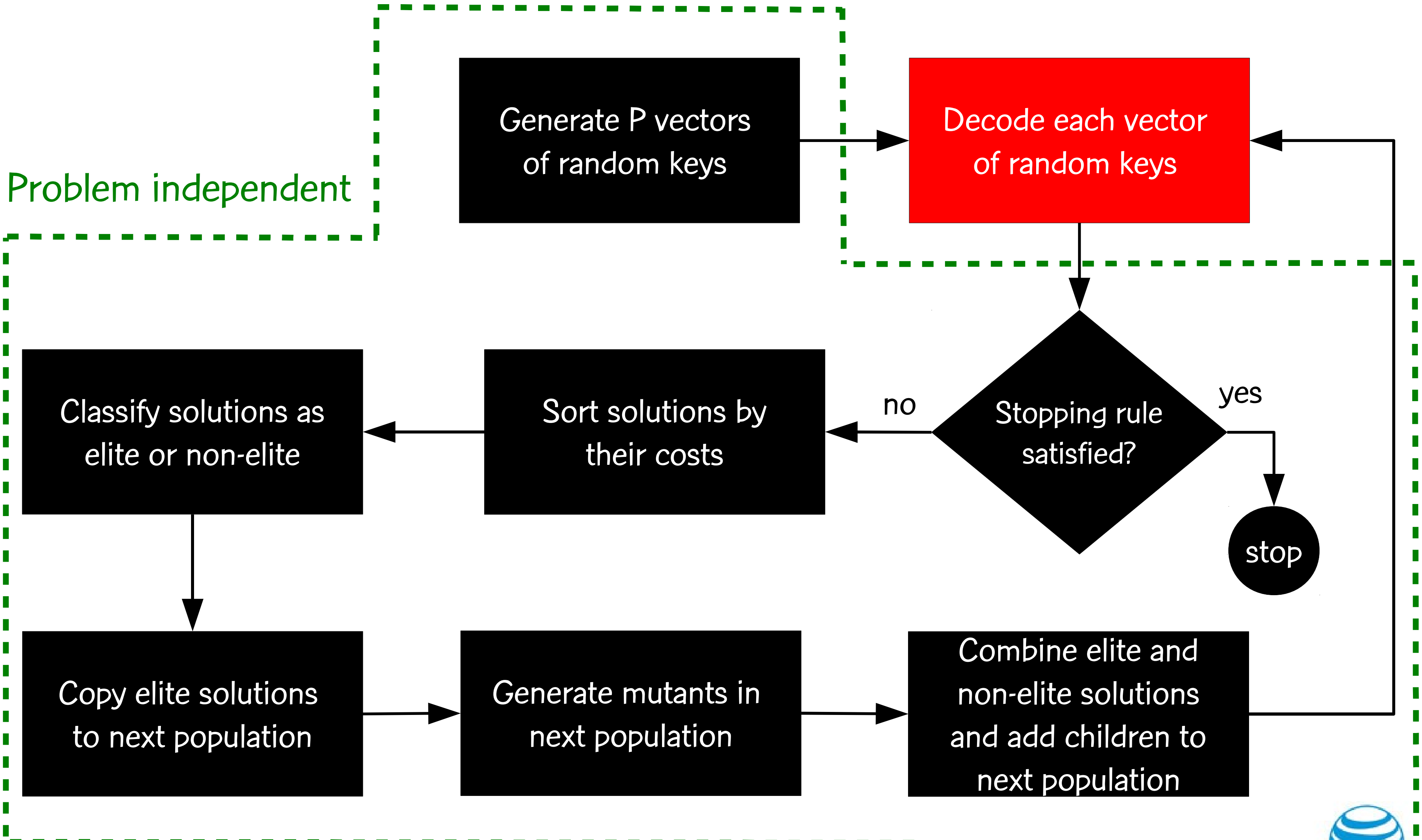
- A decoder is a deterministic algorithm that takes as input a random-key vector and returns a feasible solution of the optimization problem and its cost.
- Bean (1994) proposed decoders based on sorting the random-key vector to produce a sequence.
- A random-key GA searches the solution space indirectly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



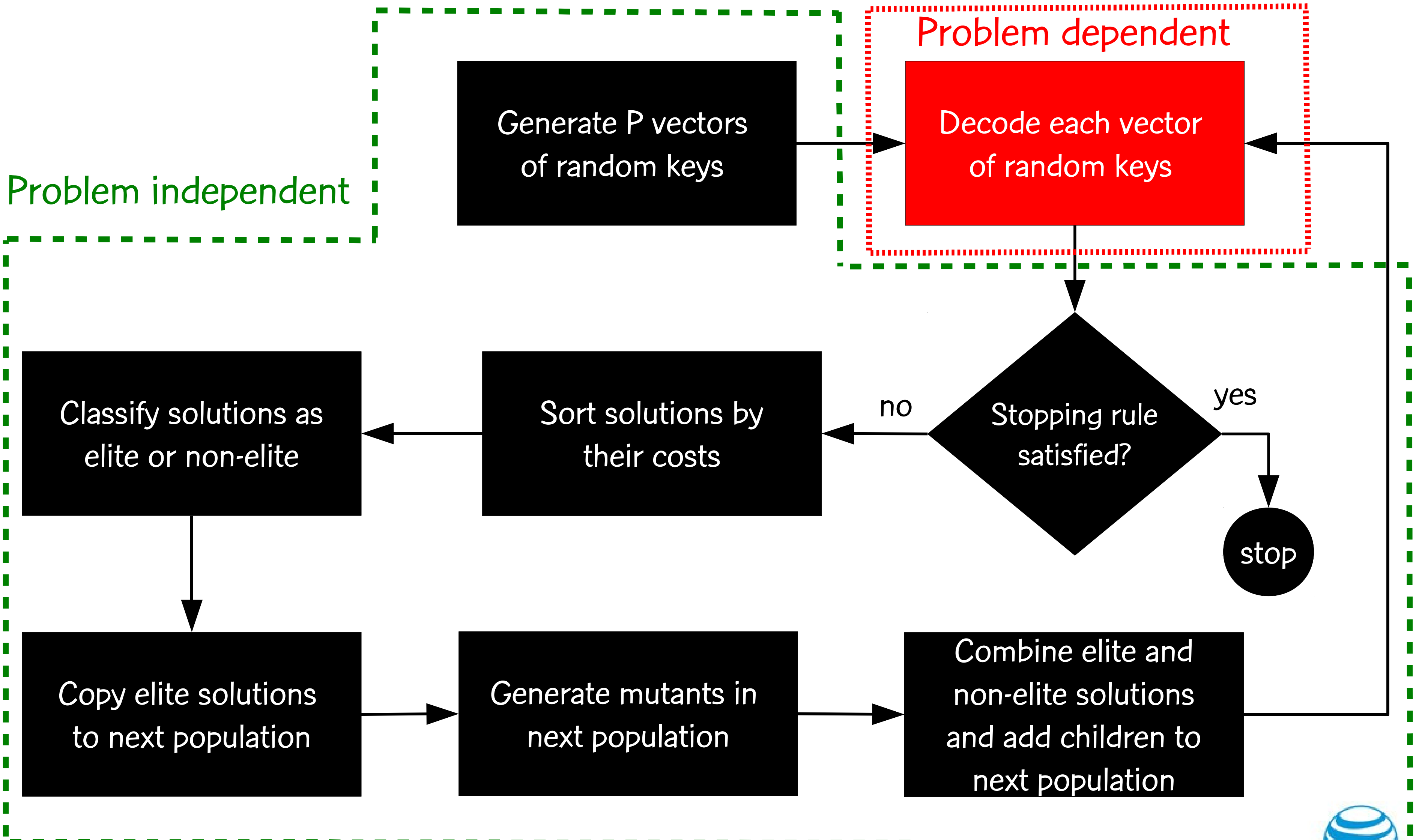
Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Framework for biased random-key genetic algorithms



Paper on API for BRKGA

R.F. Toso and M.G.C. Resende, A C++ application programming interface for biased random-key genetic algorithms, AT&T Labs Research Technical Report, 2012

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

Specifying a biased random-key GA

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

Specifying a biased random-key GA

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

Specifying a biased random-key GA

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

Specifying a biased random-key GA

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

Specifying a biased random-key GA

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

Specifying a biased random-key GA

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

A simple BRKGA for HMP

Encoding

Each solution is encoded as a vector of $|T|$ random keys, where $|T|$ is the number of towers

Decoding

Decoder takes input a vector of $|T|$ random keys and outputs a tower-to-RNC assignment:

1) sort vector resulting in ordering of towers

2) scan towers in order ...

- place tower in RNC with available capacity with which the tower has greatest number of handovers with other towers already assigned to RNC
- if RNC with available capacity does not exist, open a new artificial RNC with capacity max $\{ c_i \mid i \in \text{open RNCs} \}$

3) apply move-based local search (like one used in GRASP) to produce local minimum

Another BRKGA for HMP

Encoding

Each solution is encoded as a vector of $2\ |T|$ random keys, where $|T|$ is the number of towers

Decoding

Decoder takes input a vector of $2\ |T|$ random keys and outputs a tower-to-RNC assignment:

1) sort first $|T|$ keys resulting in ordering of towers

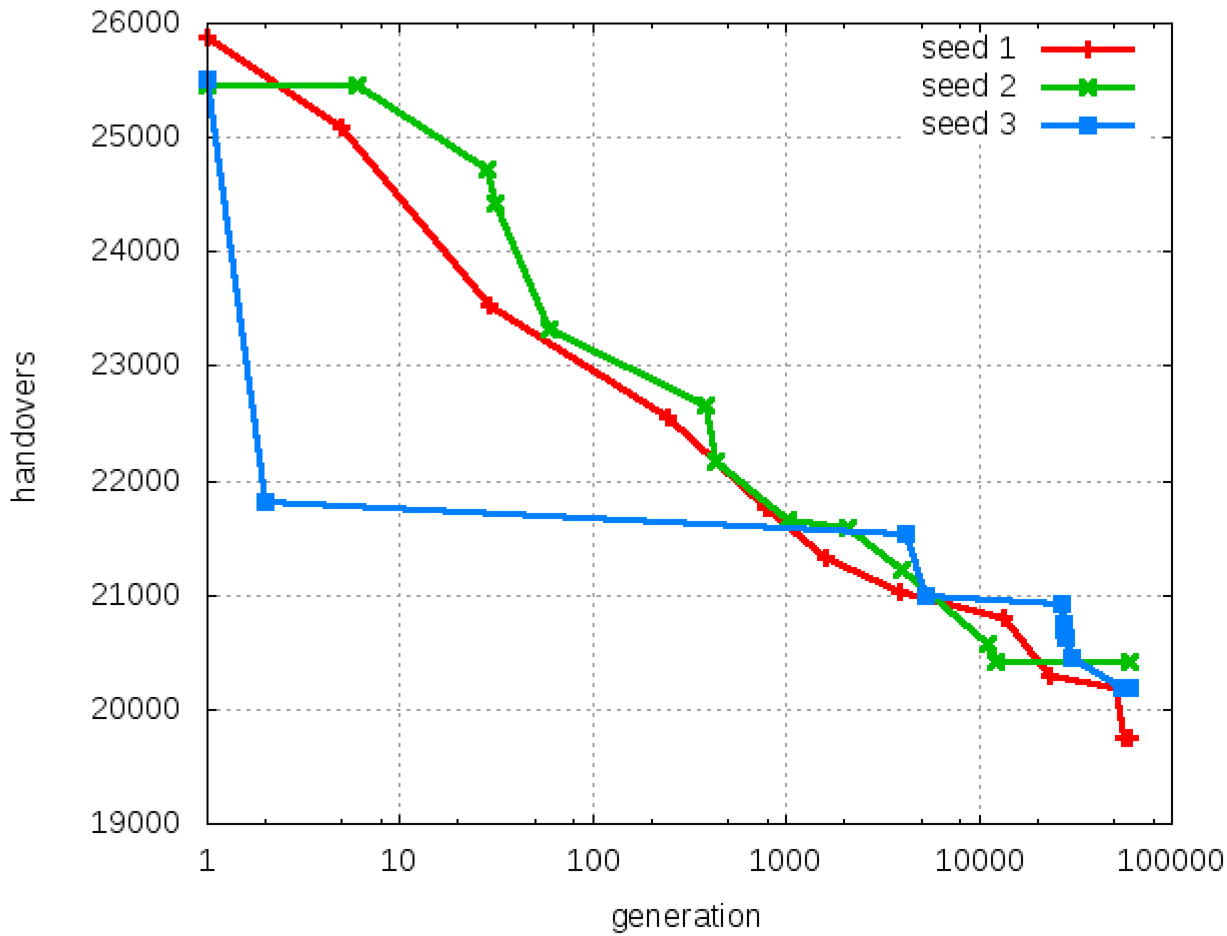
2) scan towers in order ...

- place tower in RNC with available capacity as indicated by mapping (0,1] to [1, 2, .., |RNCs|] from second $|T|$ keys
- scan unassigned towers in order and place them in RNC with available capacity maximizing handover count with tower assigned there
- if RNC with available capacity does not exist, assign tower to RNC with maximum handover count w.r.t. to tower

3) apply move-based local search (like one used in GRASP) to produce local minimum

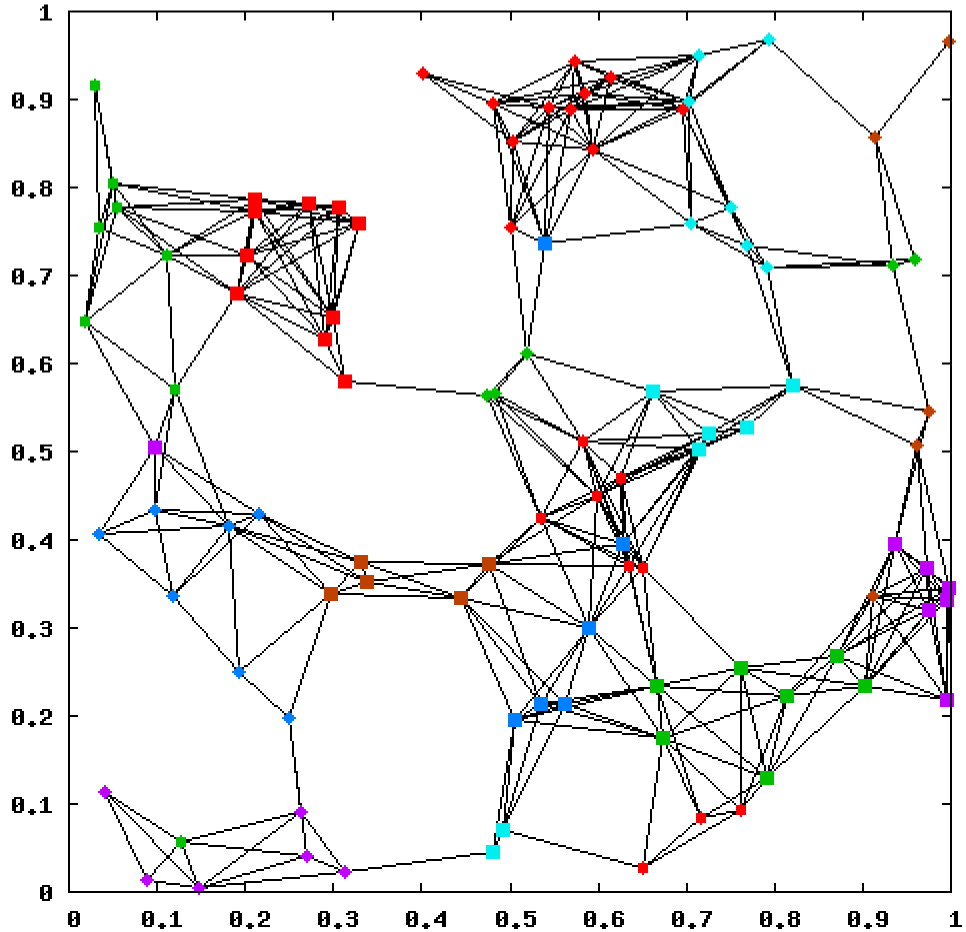
Experiments with BRKGA-1 for HMP

BRKGA: 100 towers : 14 RNCs



BRKGA: 100 towers : 14 RNCs

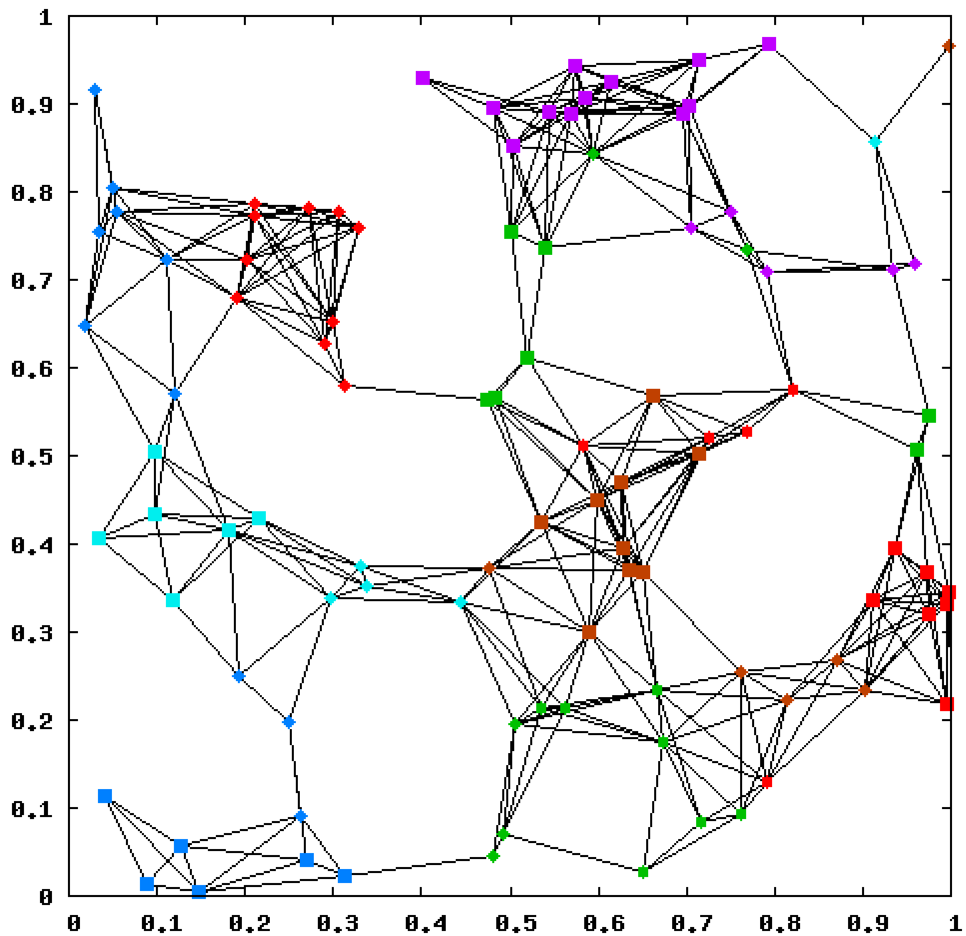
Generation: 1
Handovers: 25872



BRKGA: 100 towers : 14 RNCs

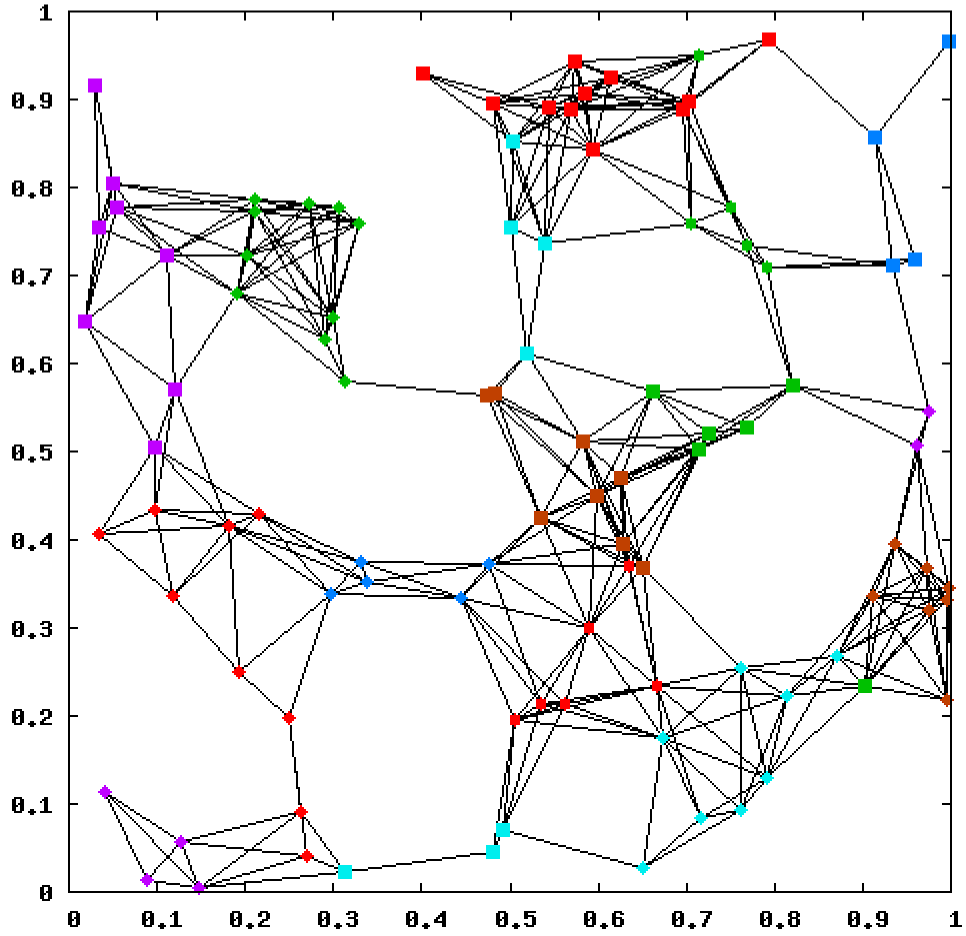
Generation: 5

Handovers: 25086



BRKGA: 100 towers : 14 RNCs

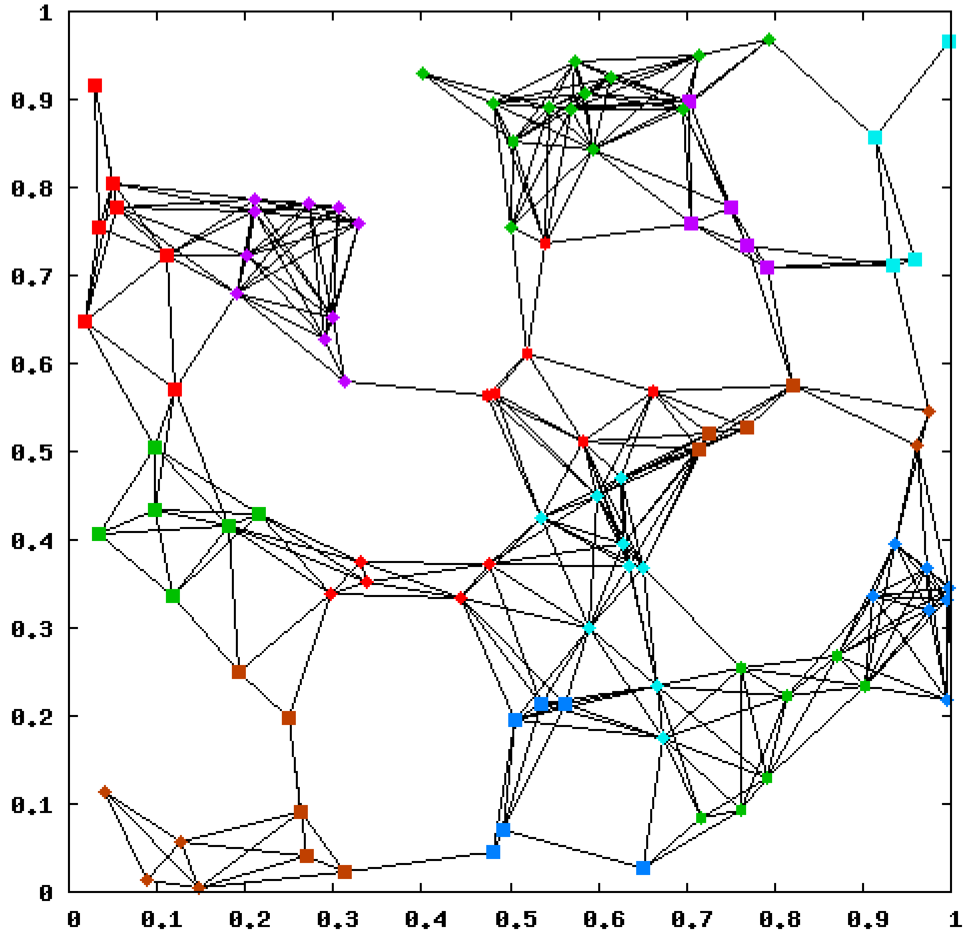
Generation: 29
Handovers: 23524



BRKGA: 100 towers : 14 RNCs

Generation: 241

Handovers: 22544



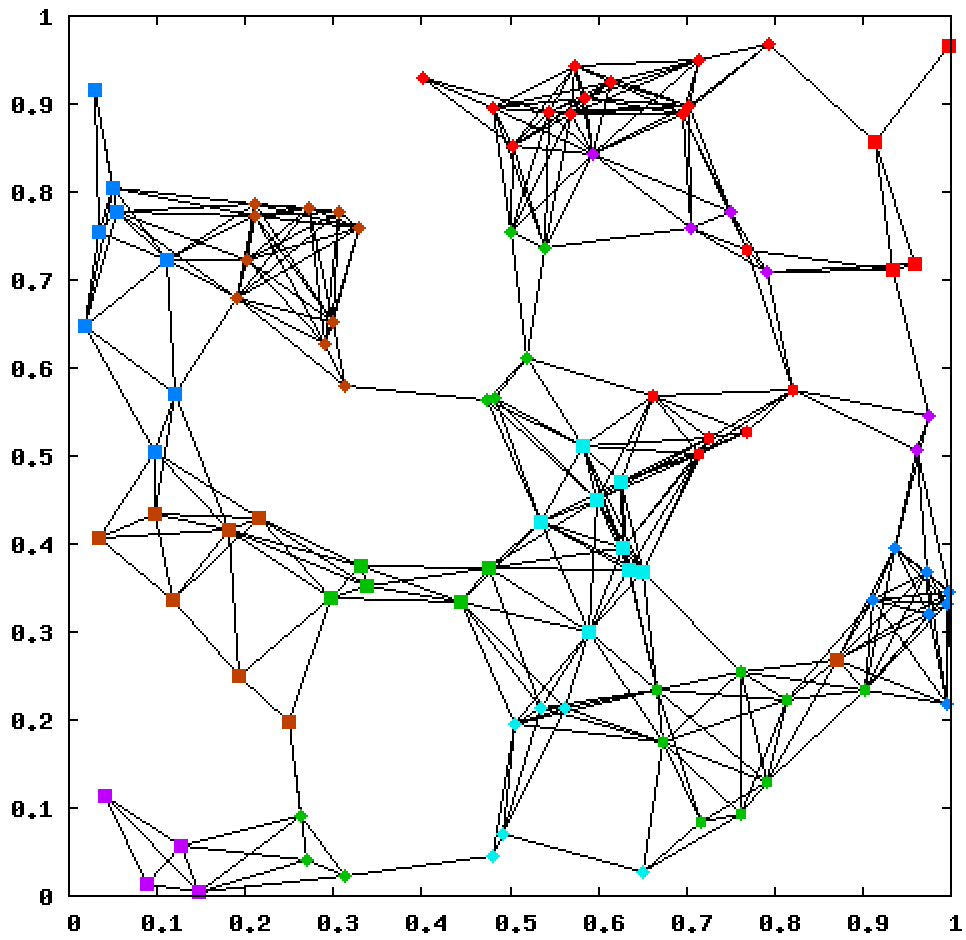
at&t

Your world. Delivered.

BRKGA: 100 towers : 14 RNCs

Generation: 777

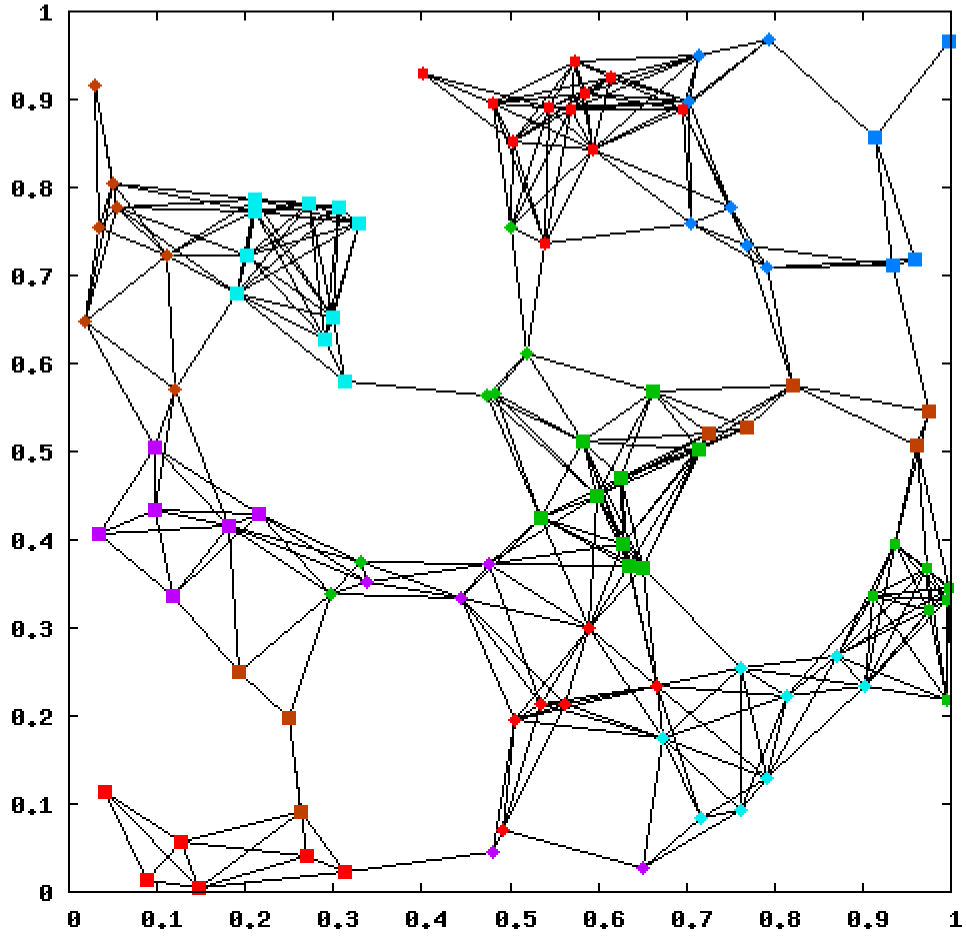
Handovers: 21766



at&t Your world. Delivered.

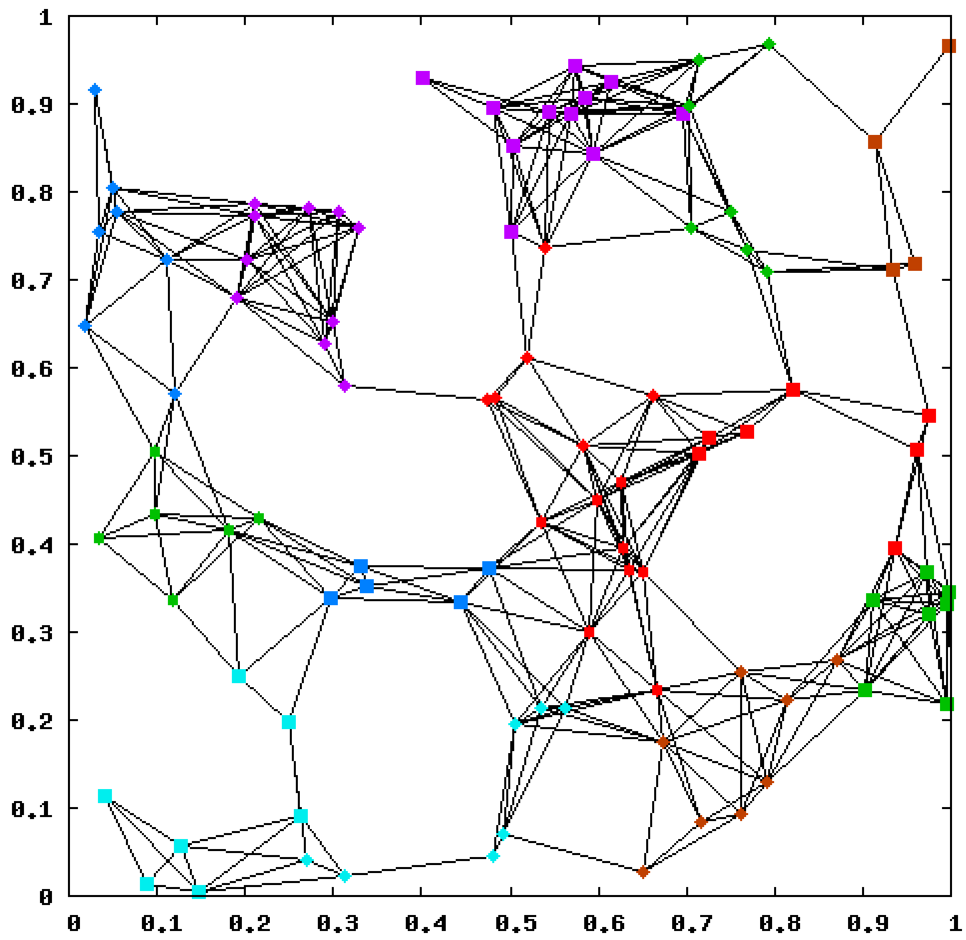
BRKGA: 100 towers : 14 RNCs

Generation: 1616
Handovers: 21336



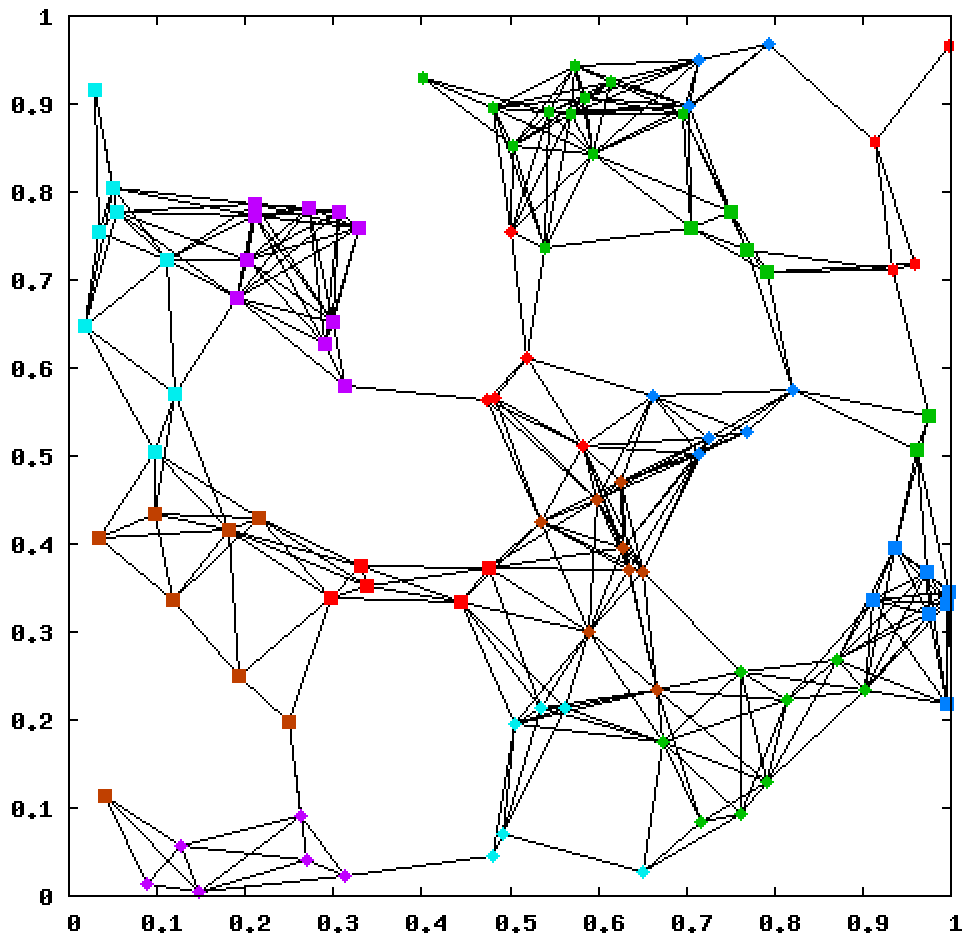
BRKGA: 100 towers : 14 RNCs

Generation: 3894
Handovers: 21022



BRKGA: 100 towers : 14 RNCs

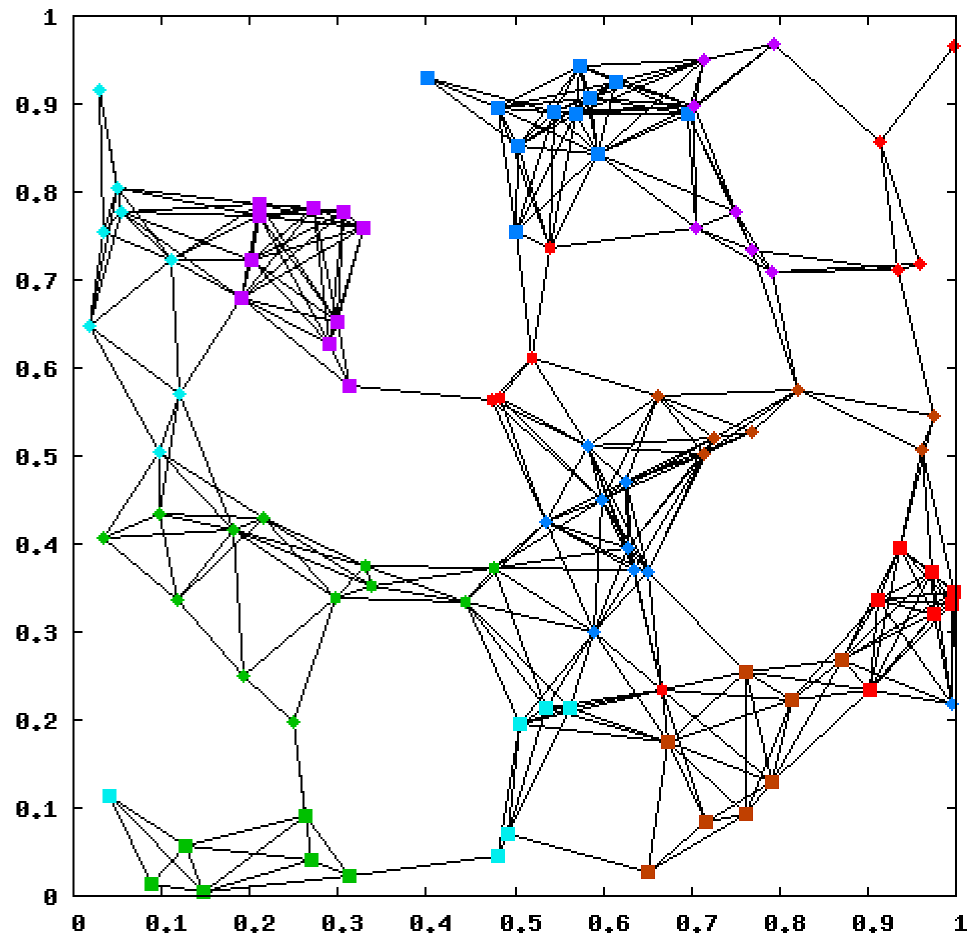
Generation: 13502
Handovers: 20806



BRKGA: 100 towers : 14 RNCs

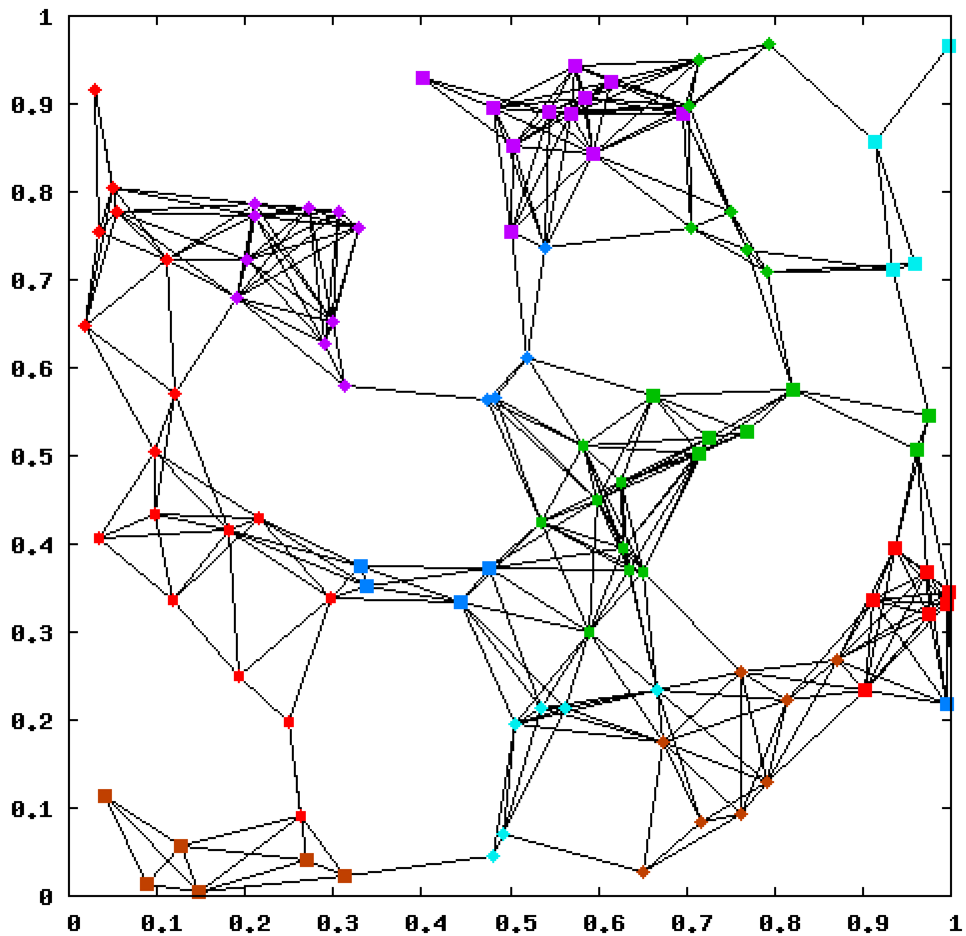
Generation: 23221

Handovers: 20288



BRKGA: 100 towers : 14 RNCs

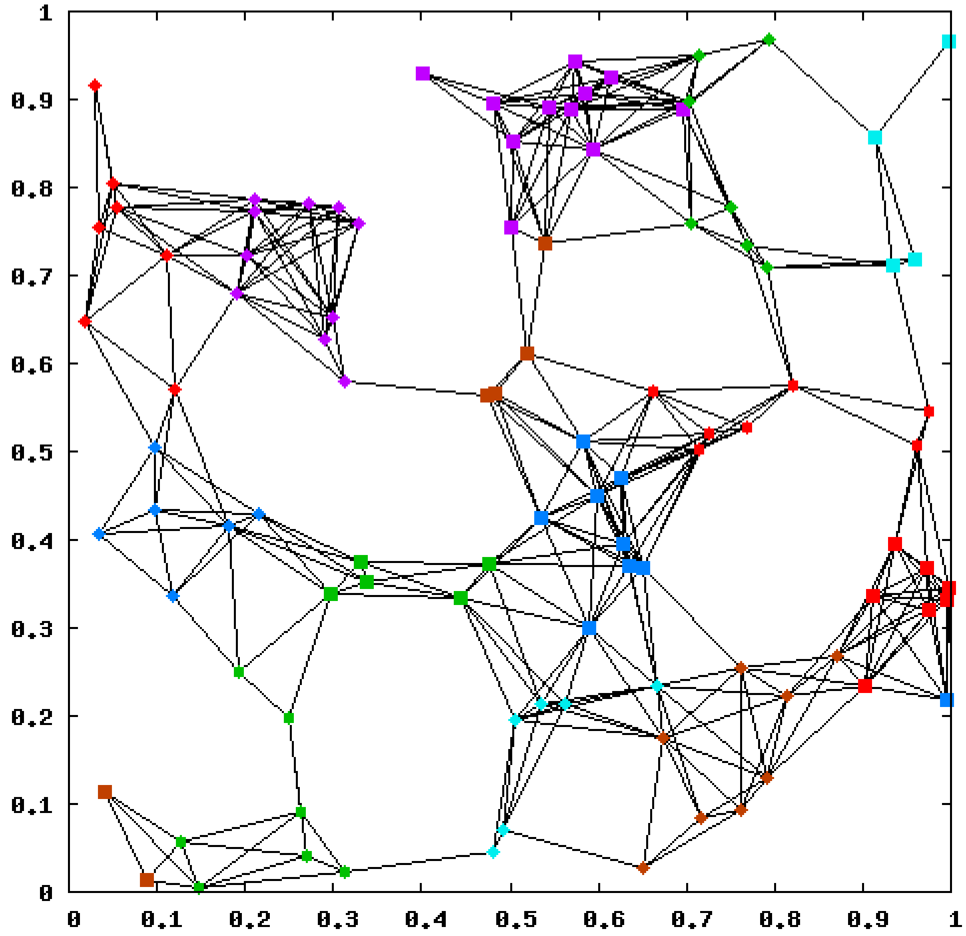
Generation: 51359
Handovers: 20186

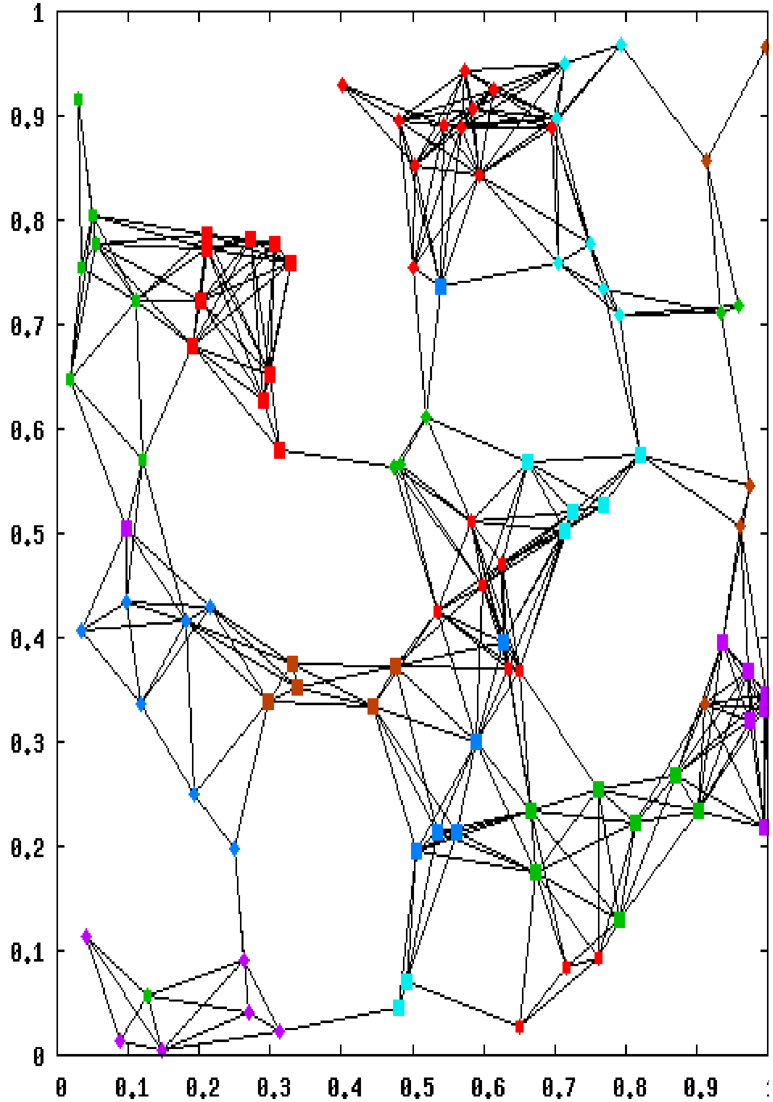


Your world. Delivered.

BRKGA: 100 towers : 14 RNCs

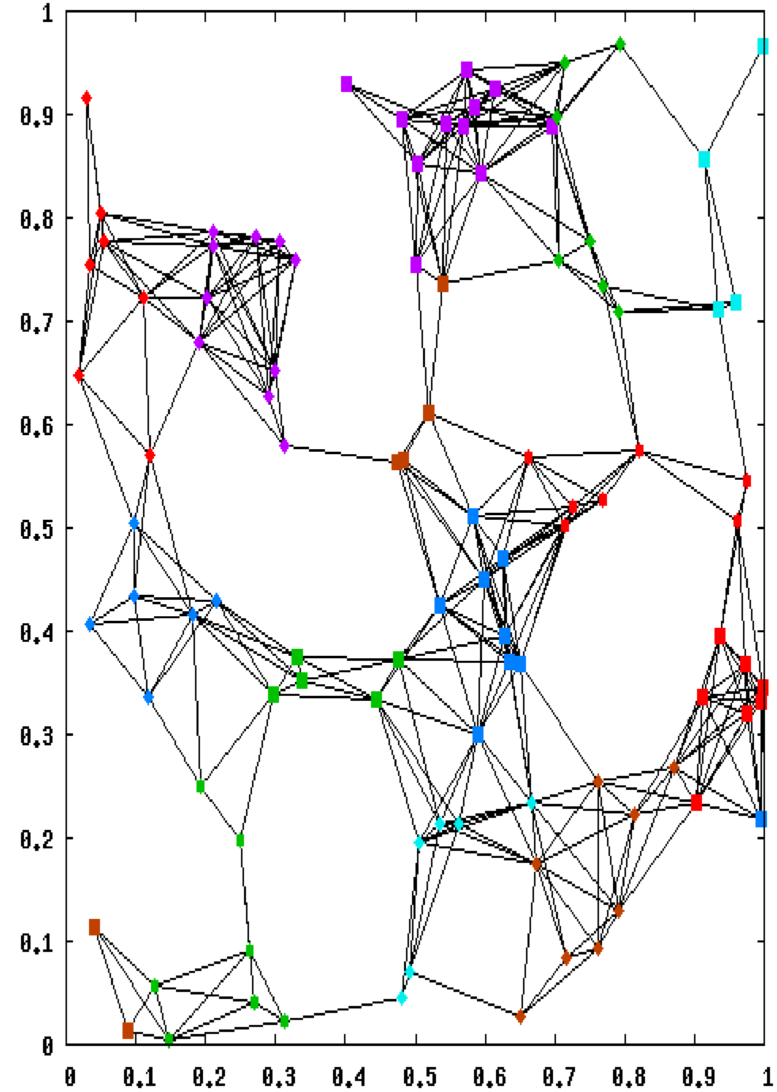
Generation: 56324
Handovers: 19750





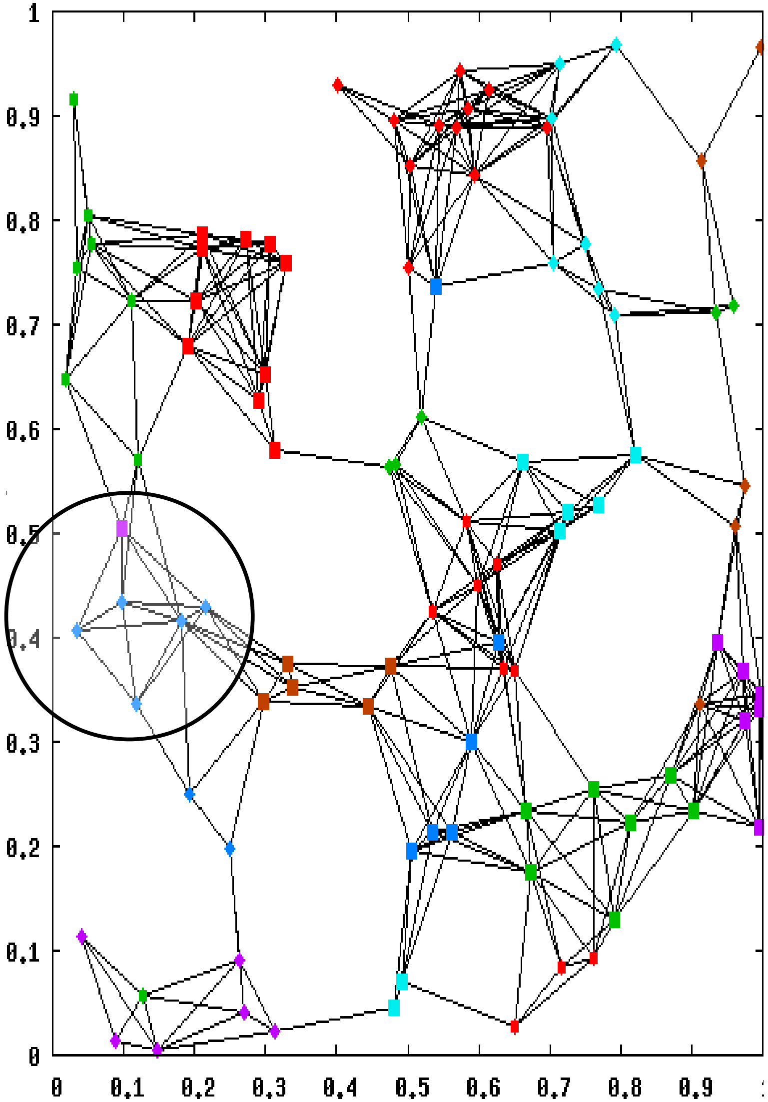
Generation: 1

Handovers: 25872



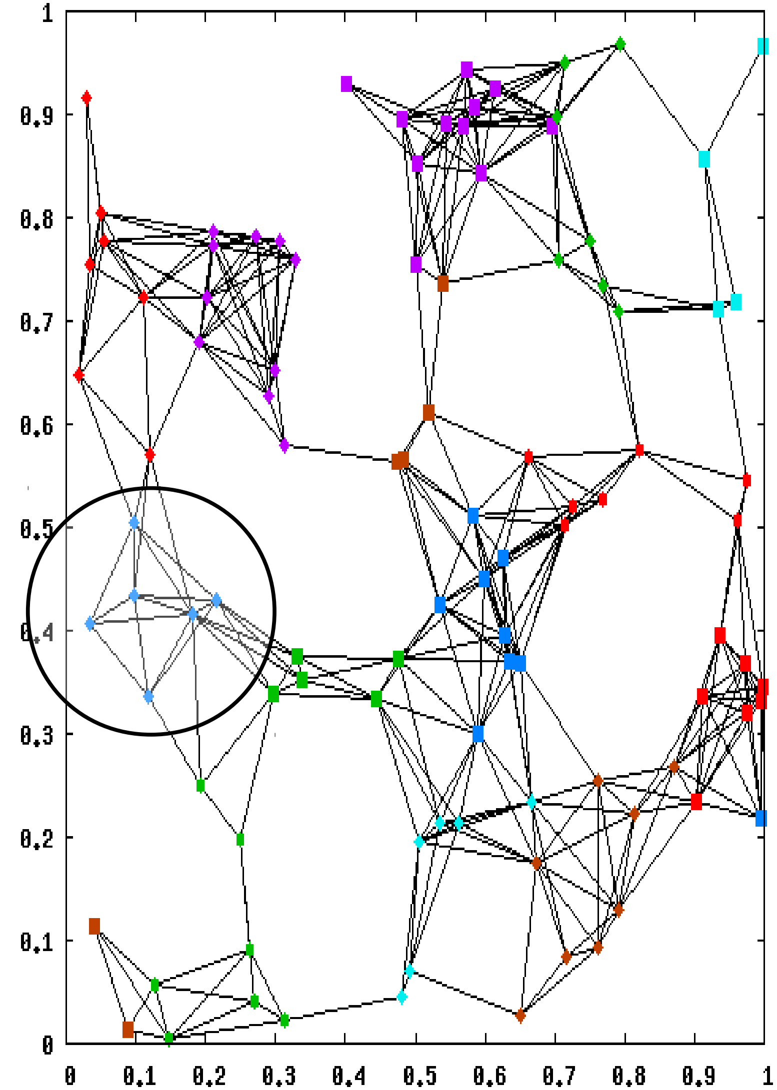
Generation: 56324

Handovers: 19750



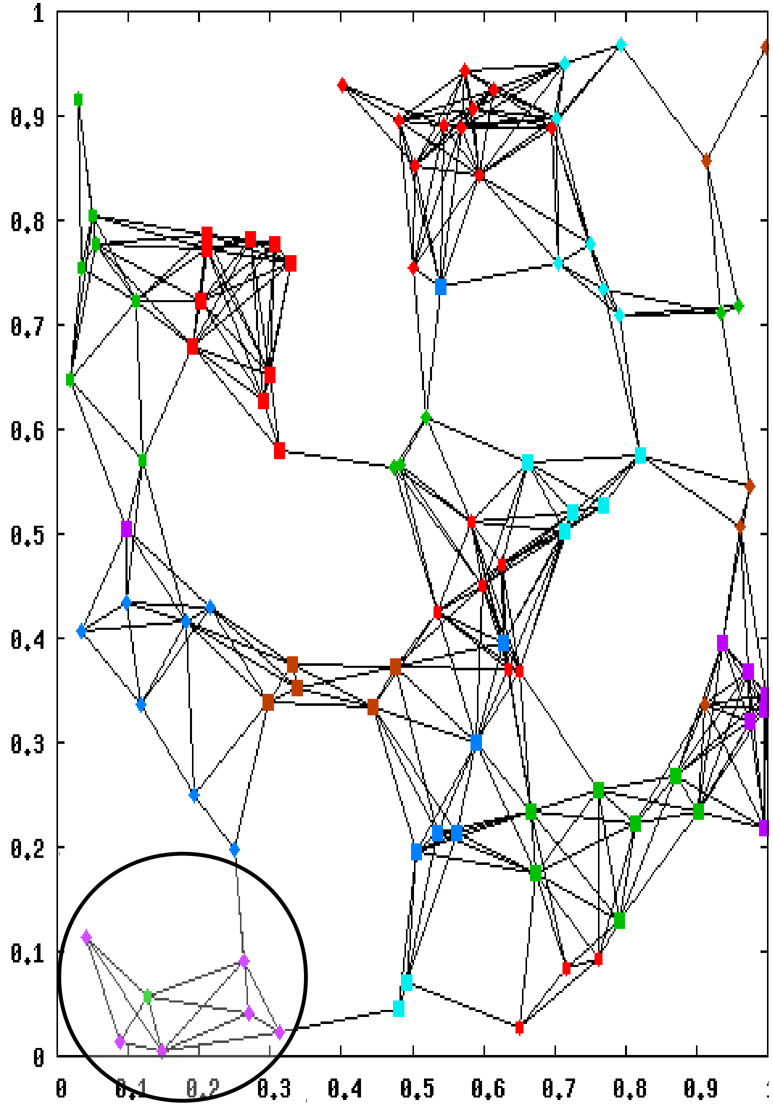
Generation: 1

Handovers: 25872

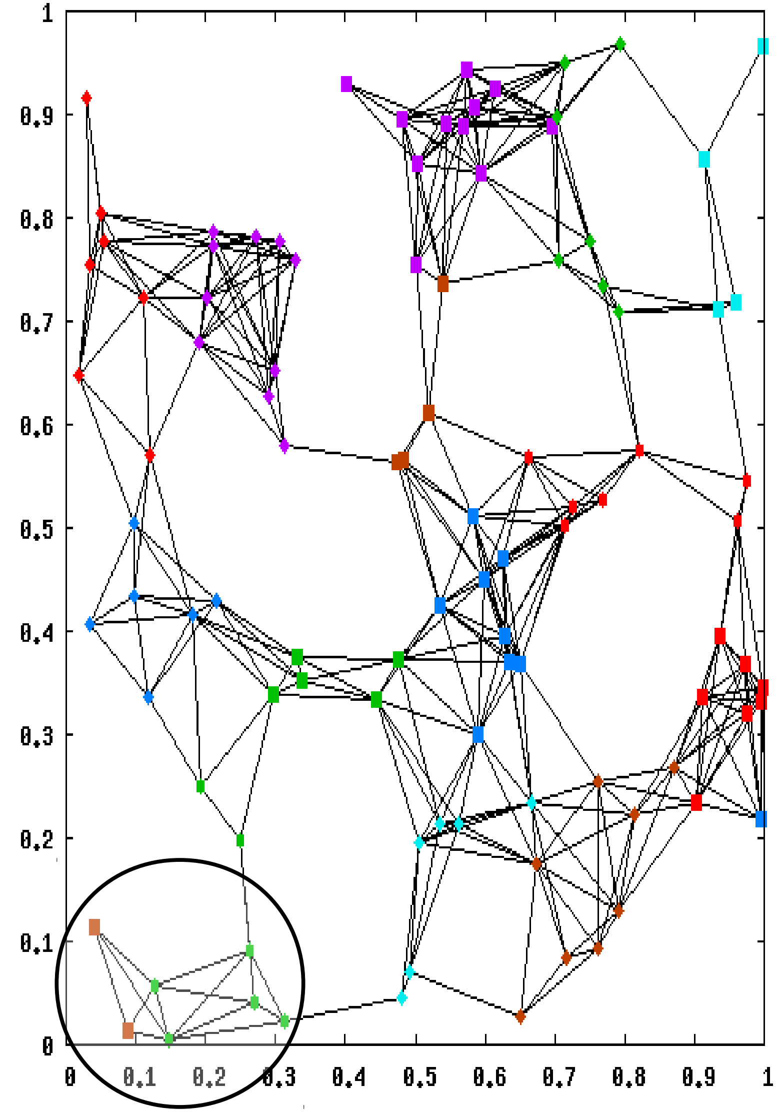


Generation: 56324

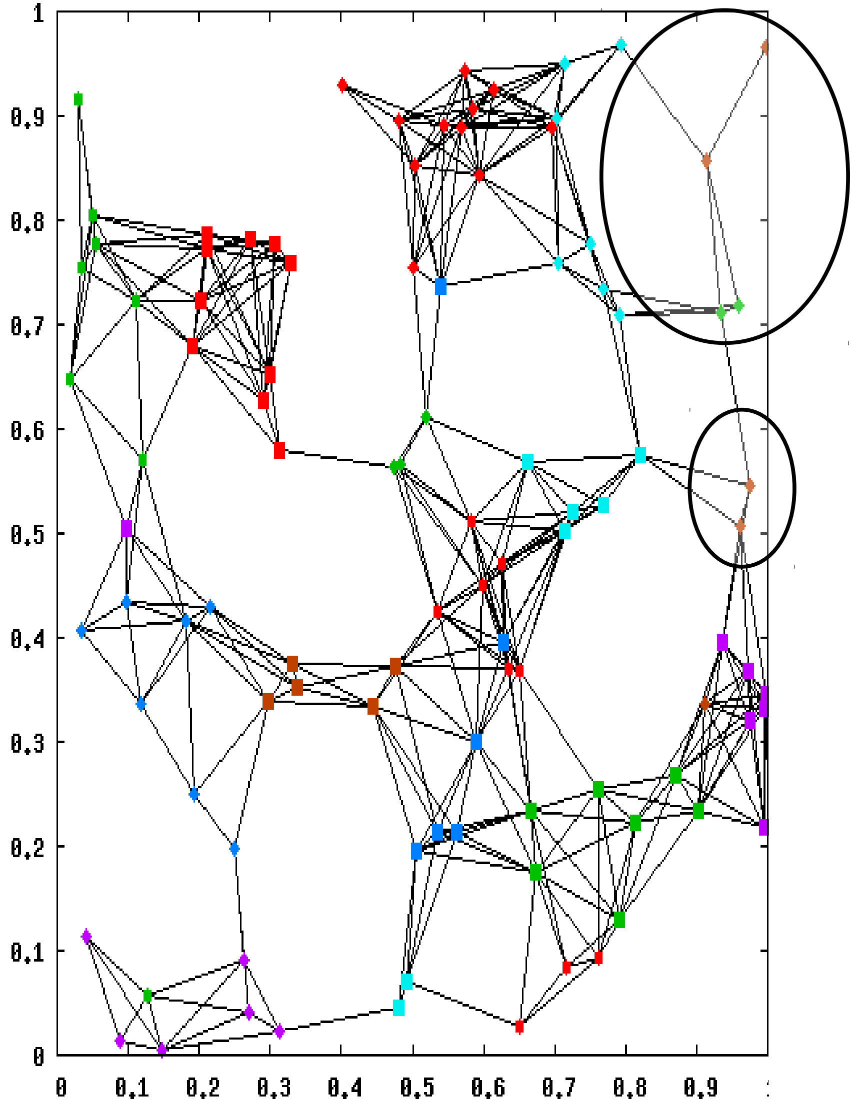
Handovers: 19750



Generation: 1
Handovers: 25872

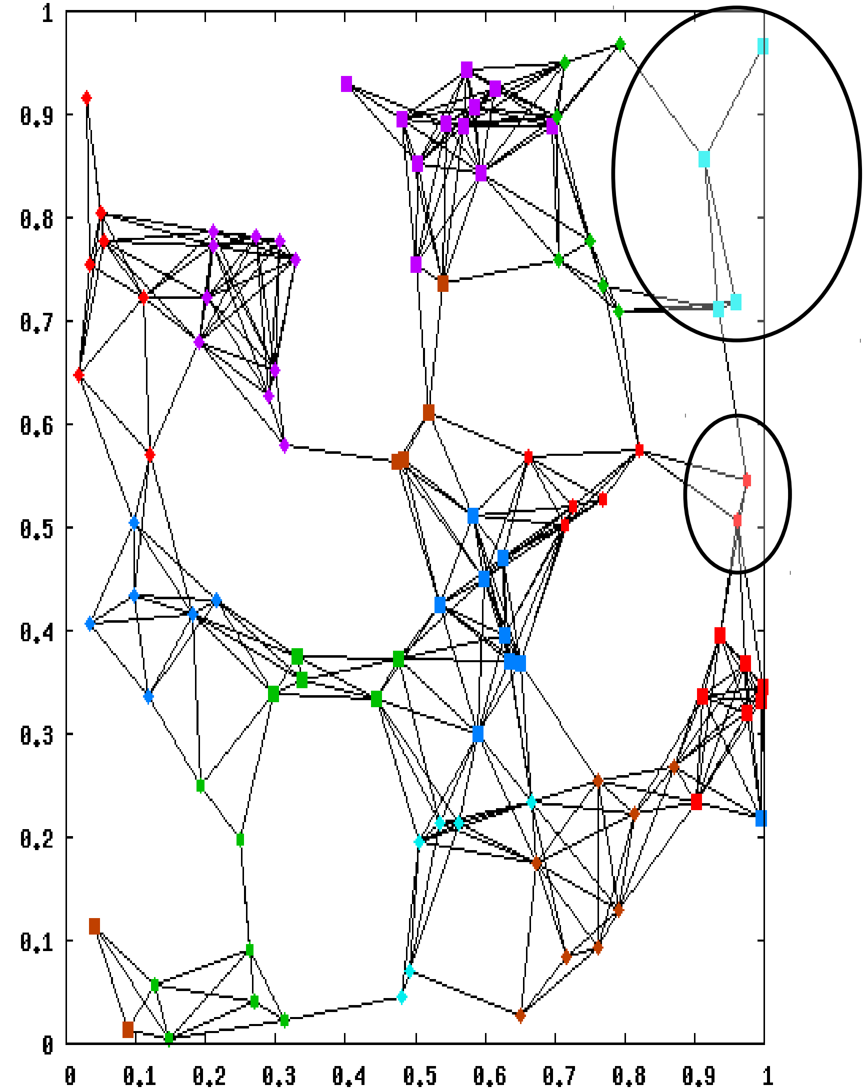


Generation: 56324
Handovers: 19750



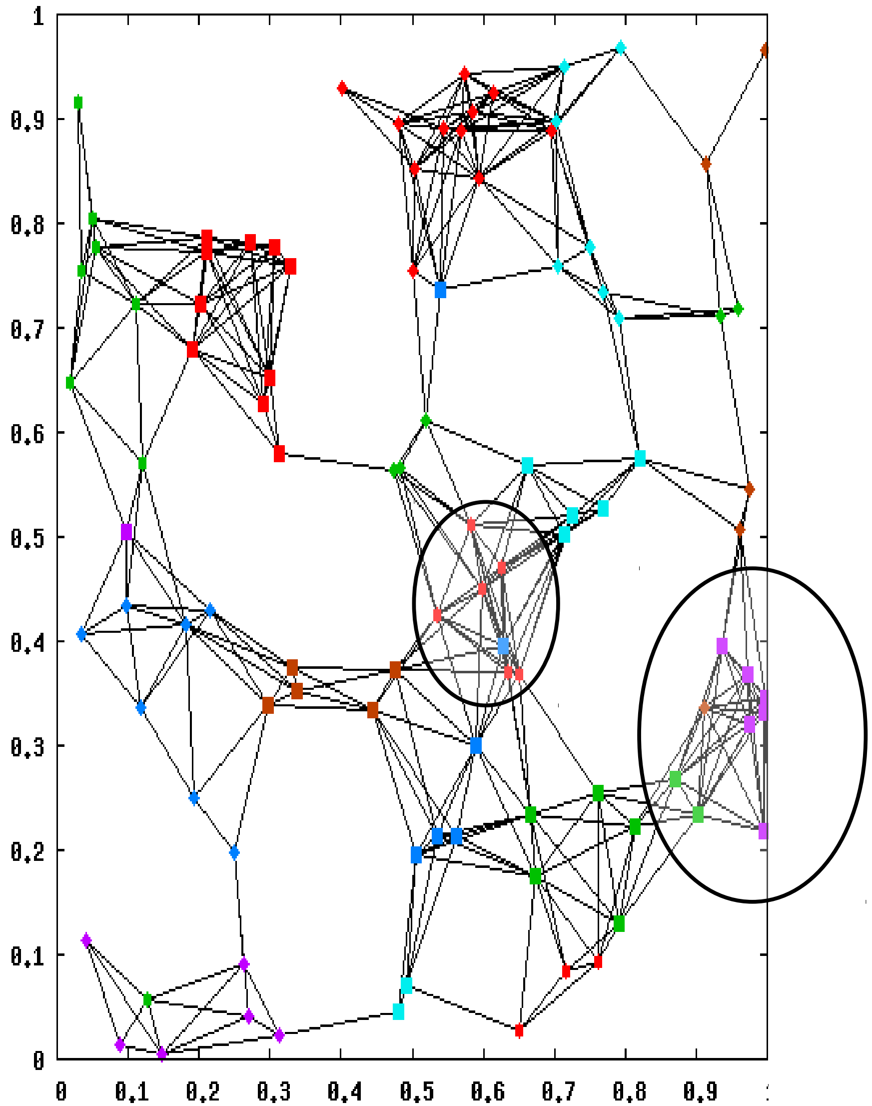
Generation: 1

Handovers: 25872

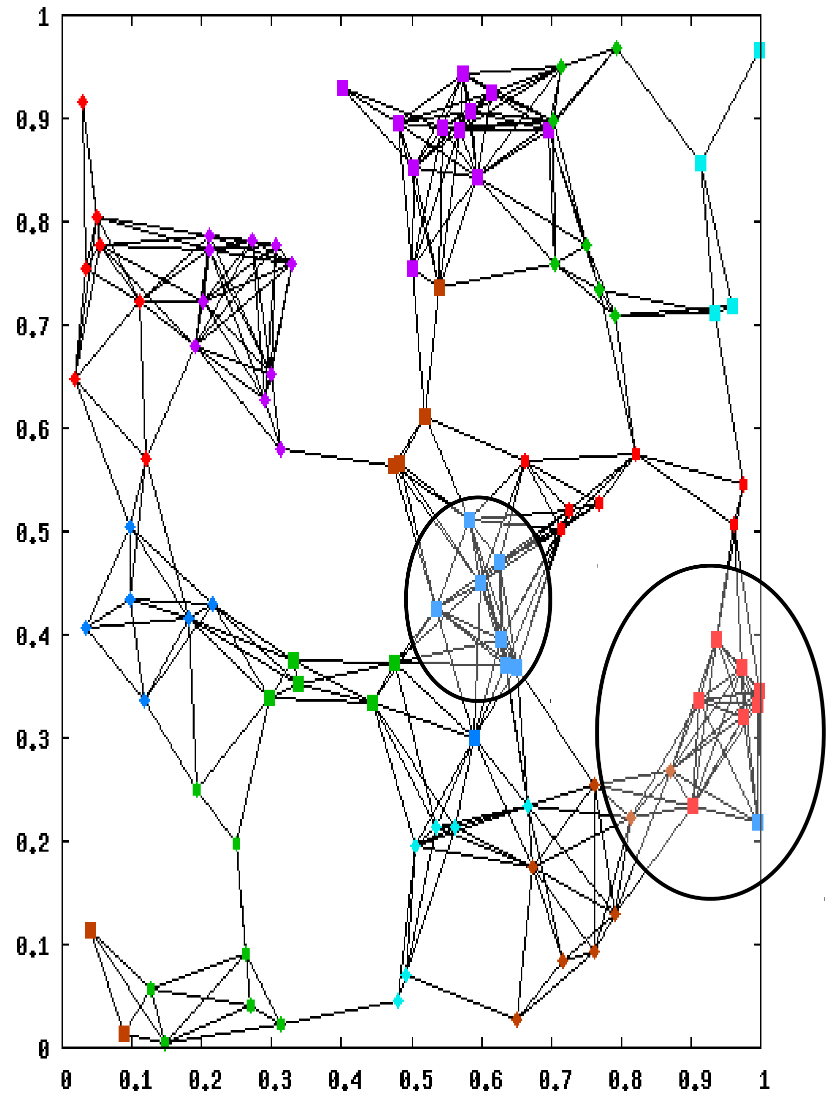


Generation: 56324

Handovers: 19750

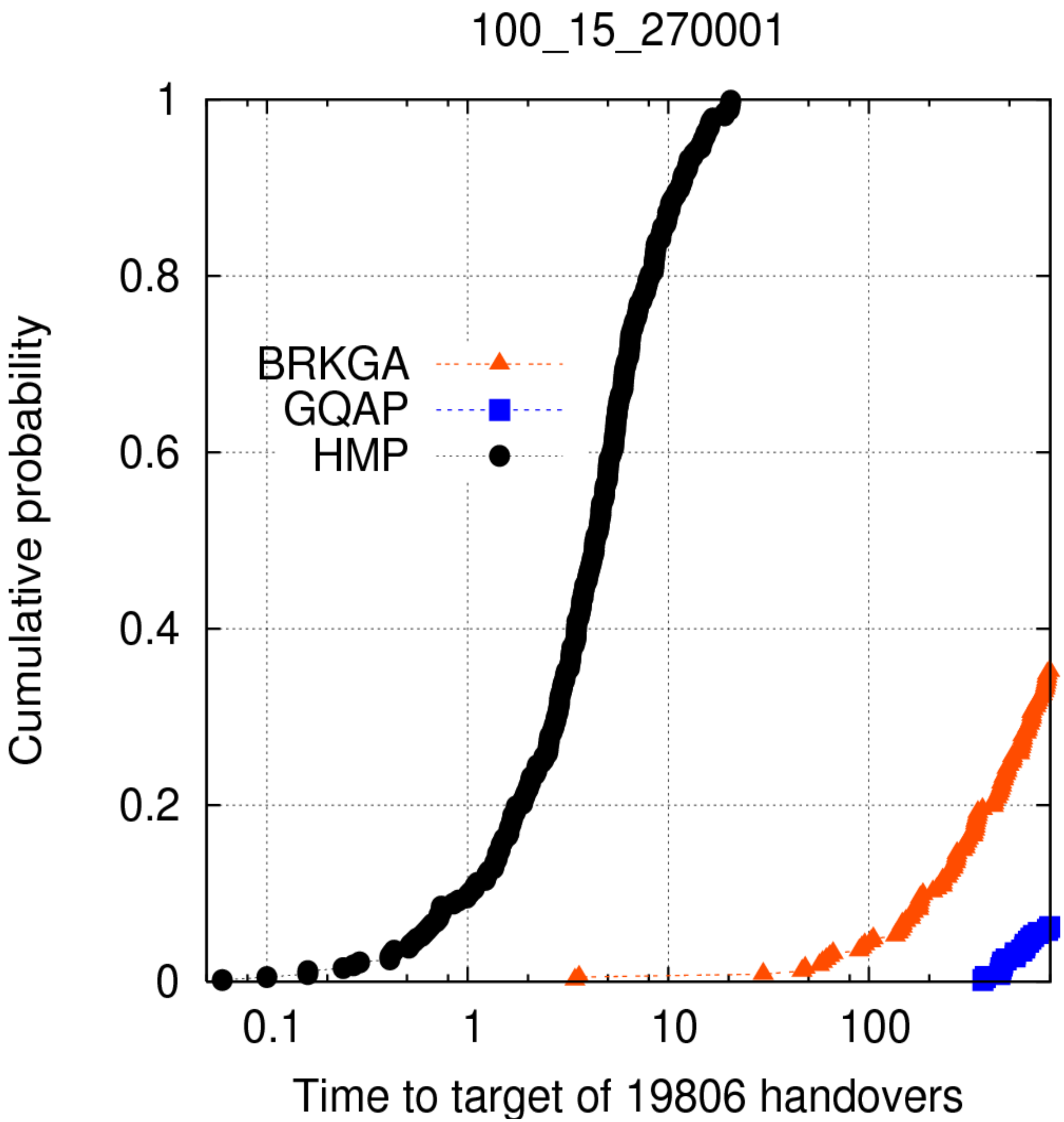


Generation: 1
Handovers: 25872

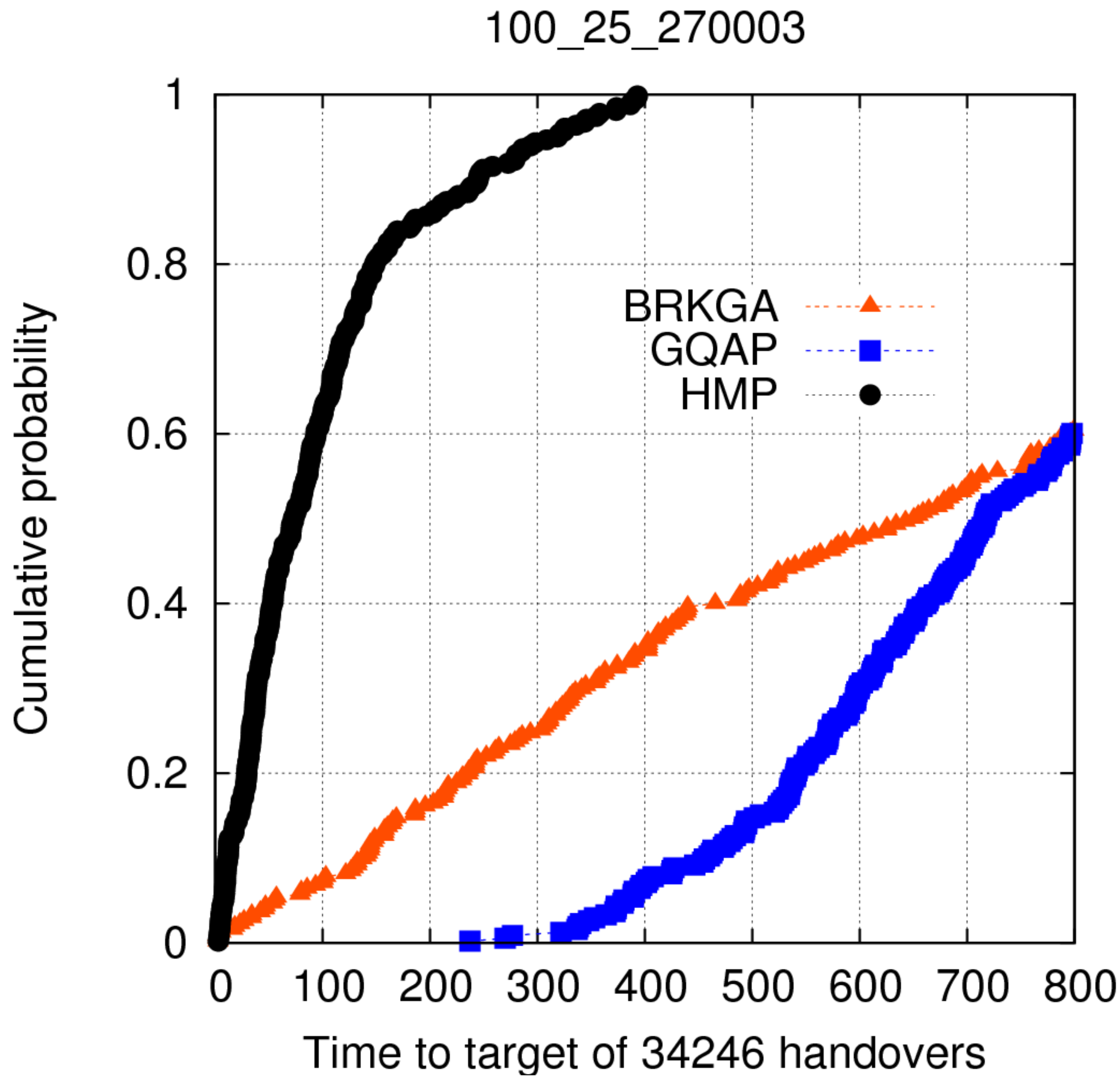


Generation: 56324
Handovers: 19750

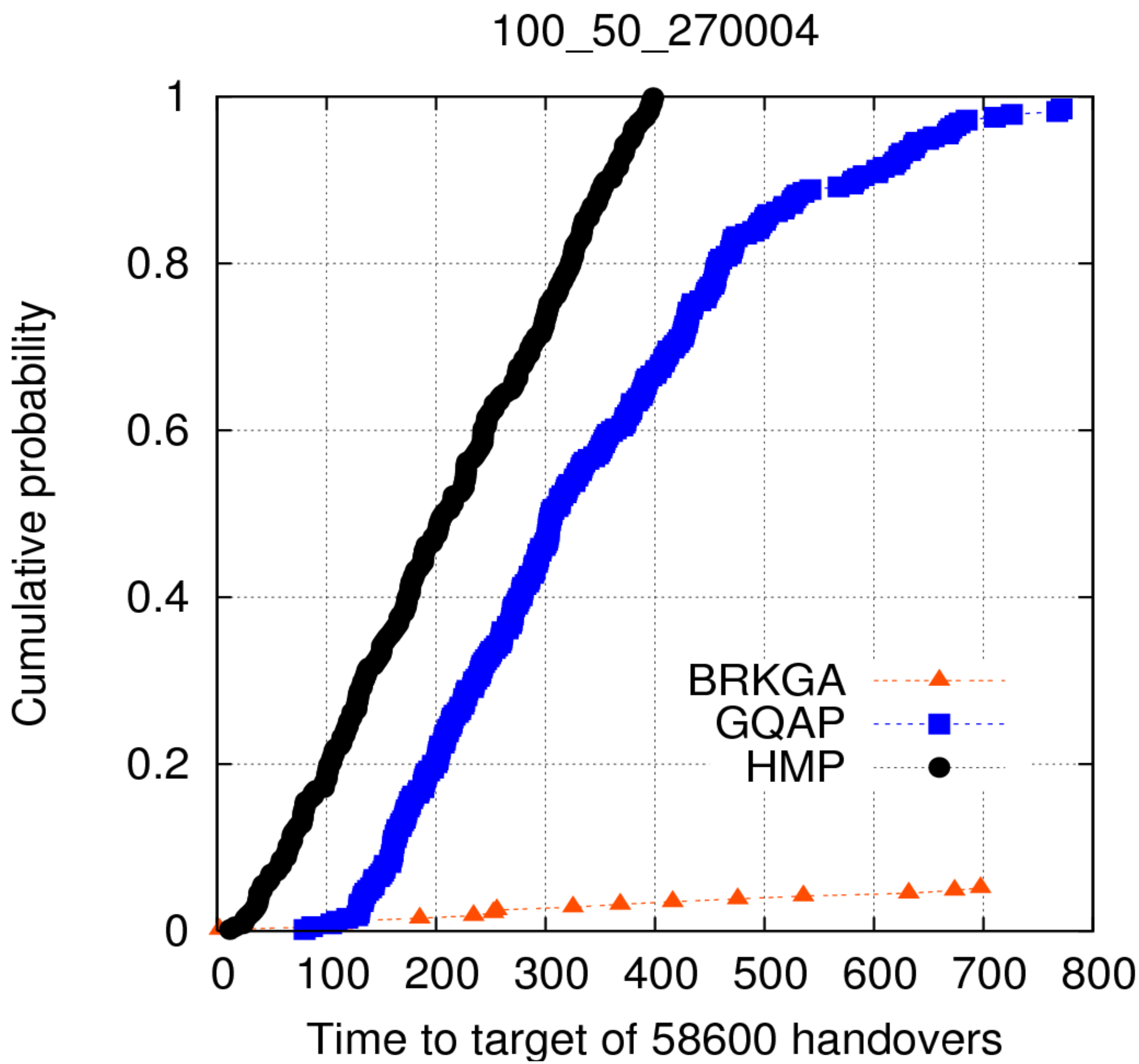
at&t
Your world. Delivered.



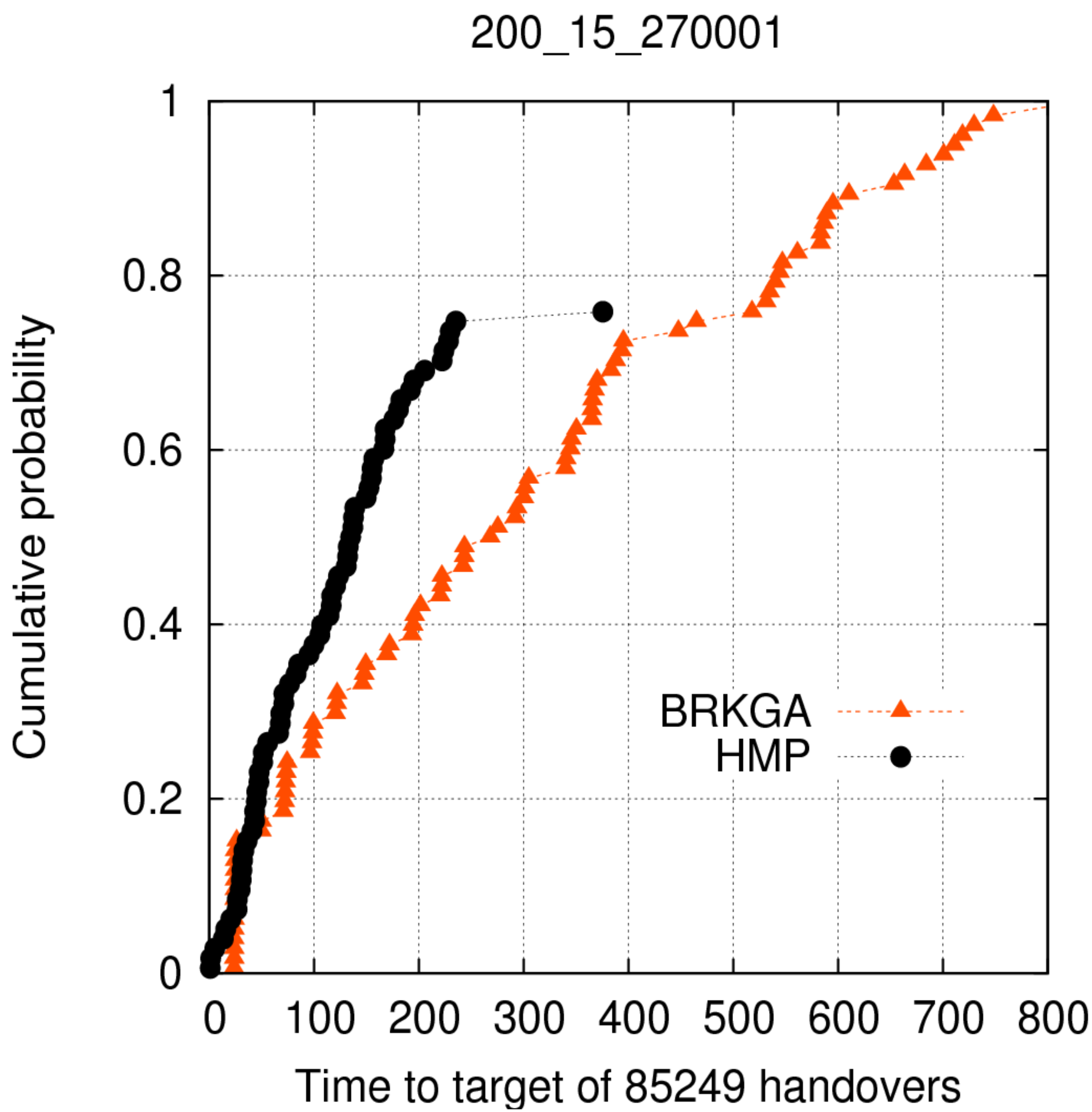
100 trials for each heuristic stopping when target solution was found or after 800s



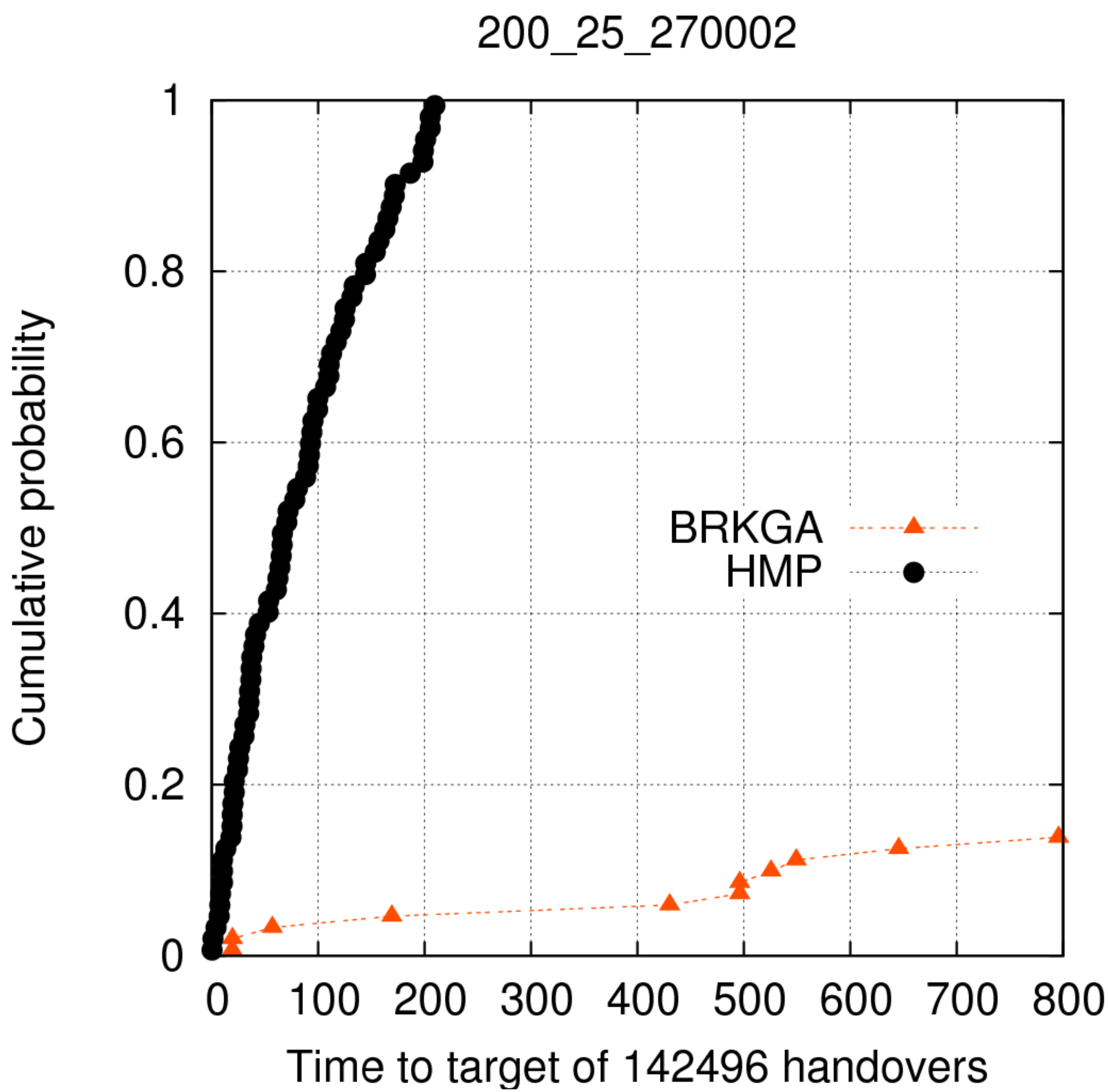
100 trials for each heuristic stopping when target solution was found or after 800s



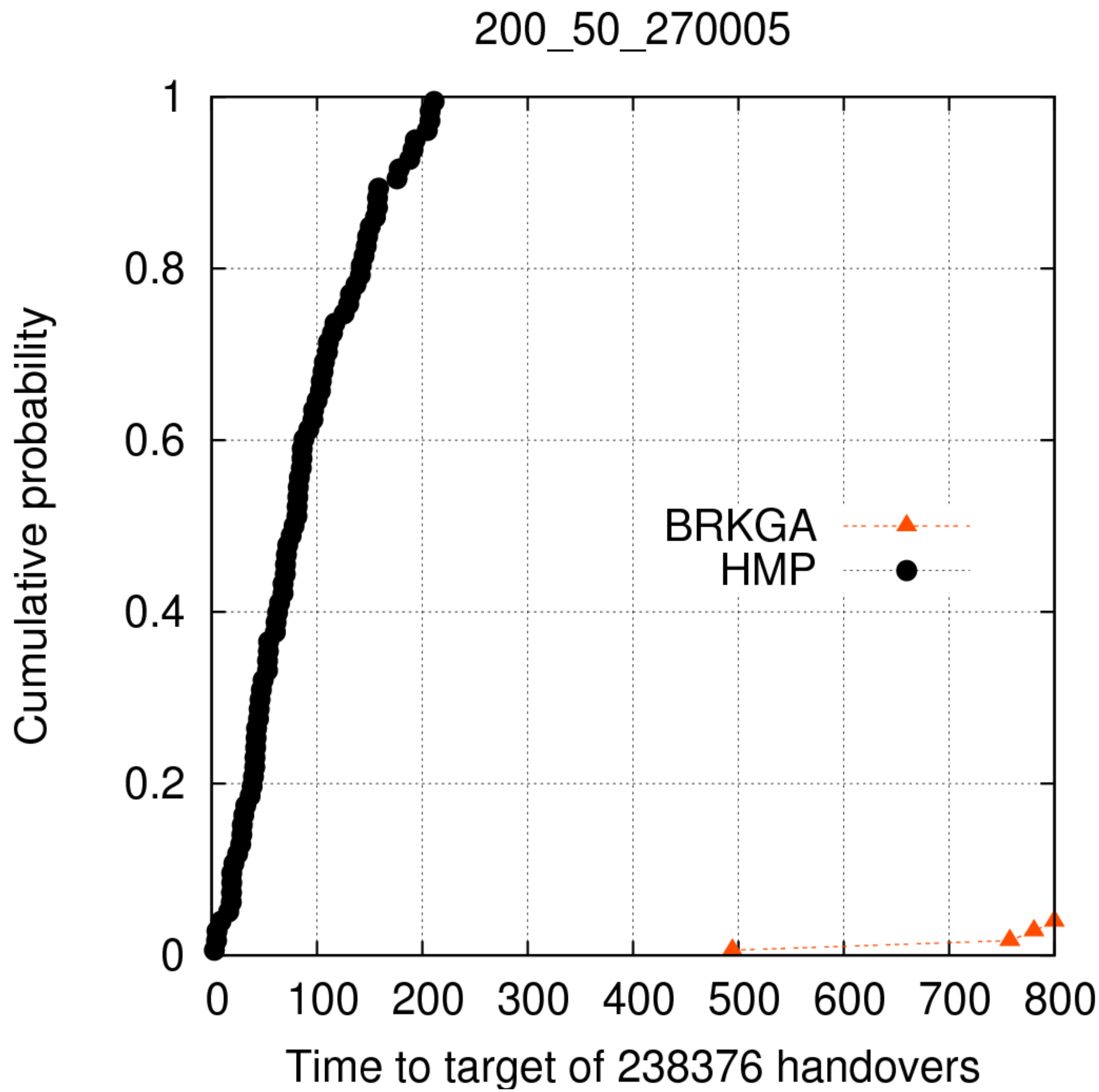
100 trials for each heuristic stopping when target solution was found or after 800s



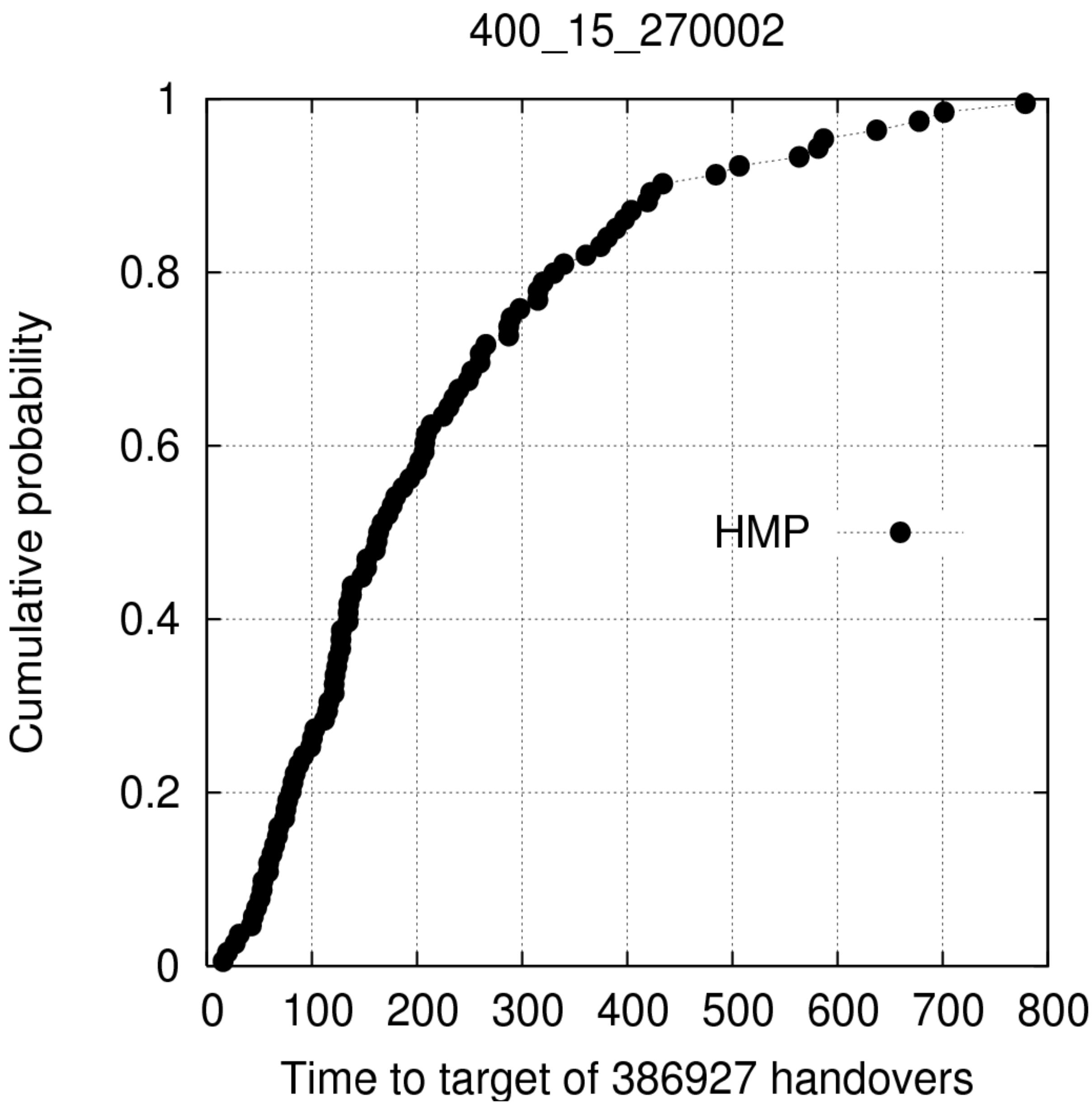
100 trials for each heuristic stopping when target solution was found or after 800s



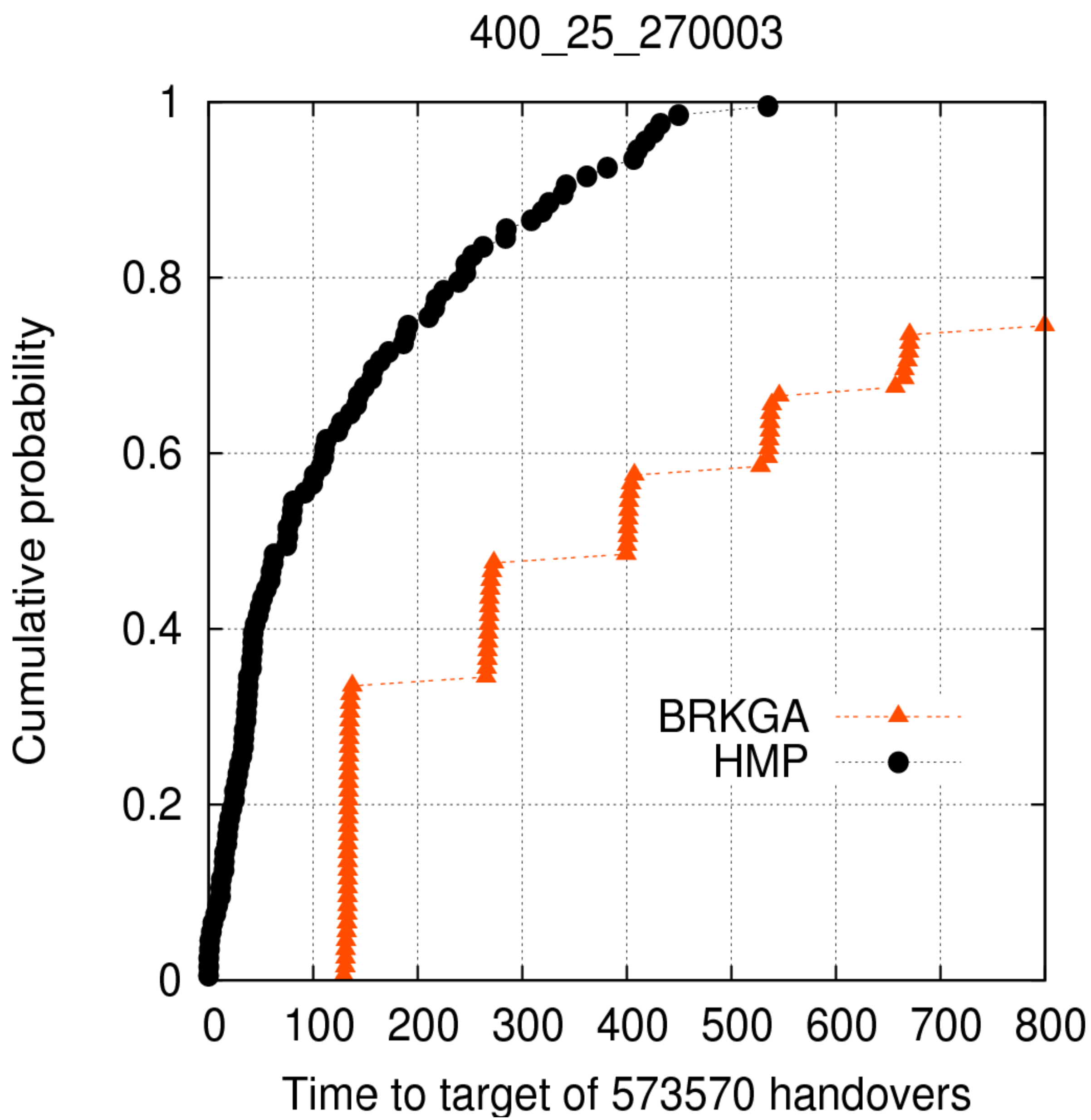
100 trials for each heuristic stopping when target solution was found or after 800s



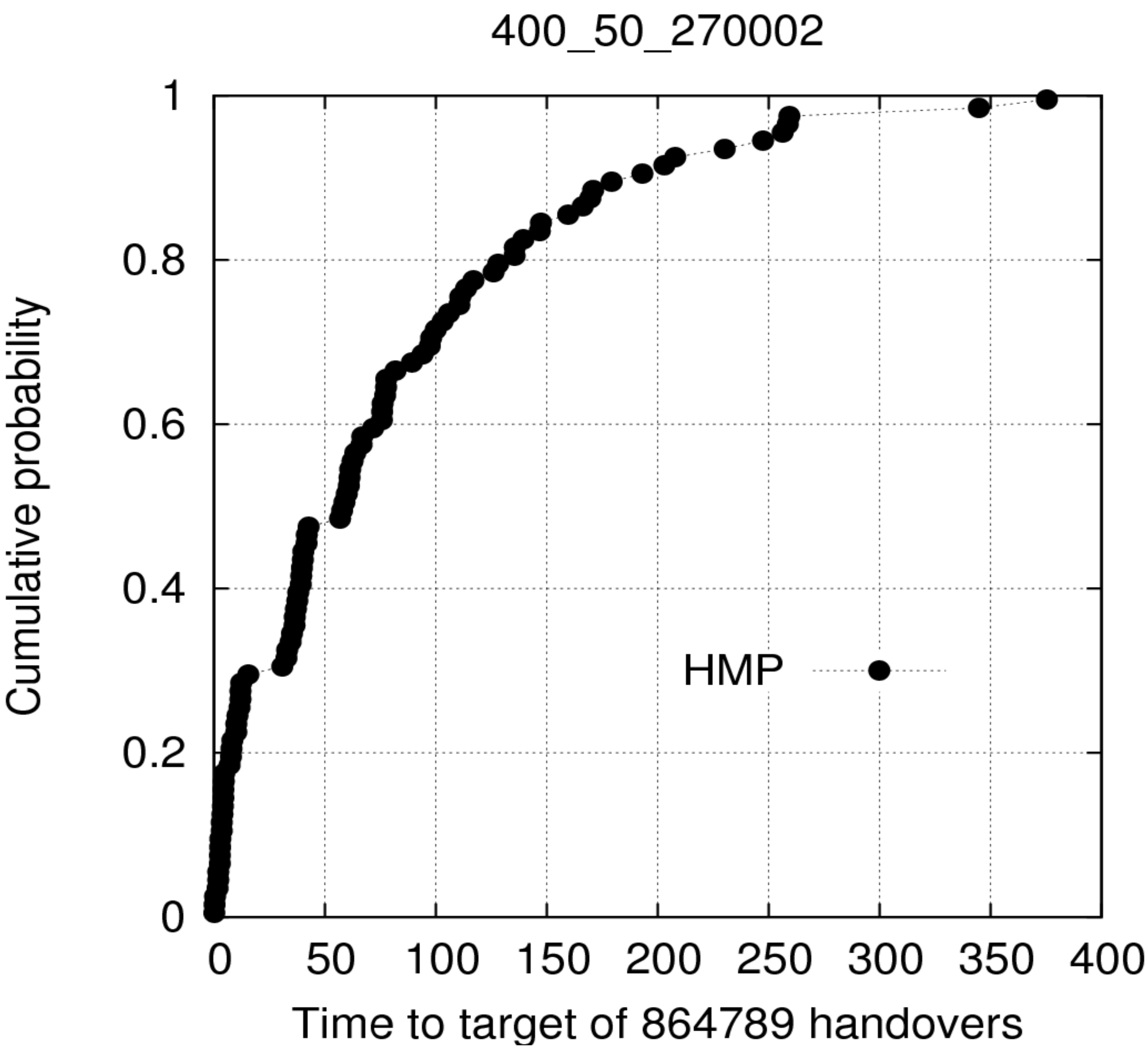
100 trials for each heuristic stopping when target solution was found or after 800s



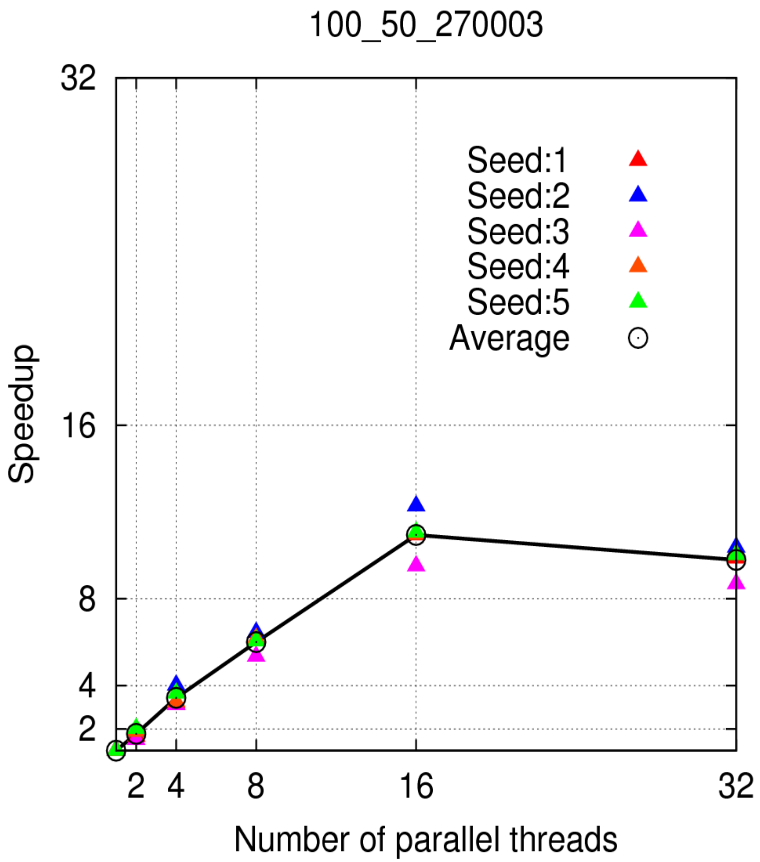
100 trials for each heuristic stopping when target solution was found or after 800s

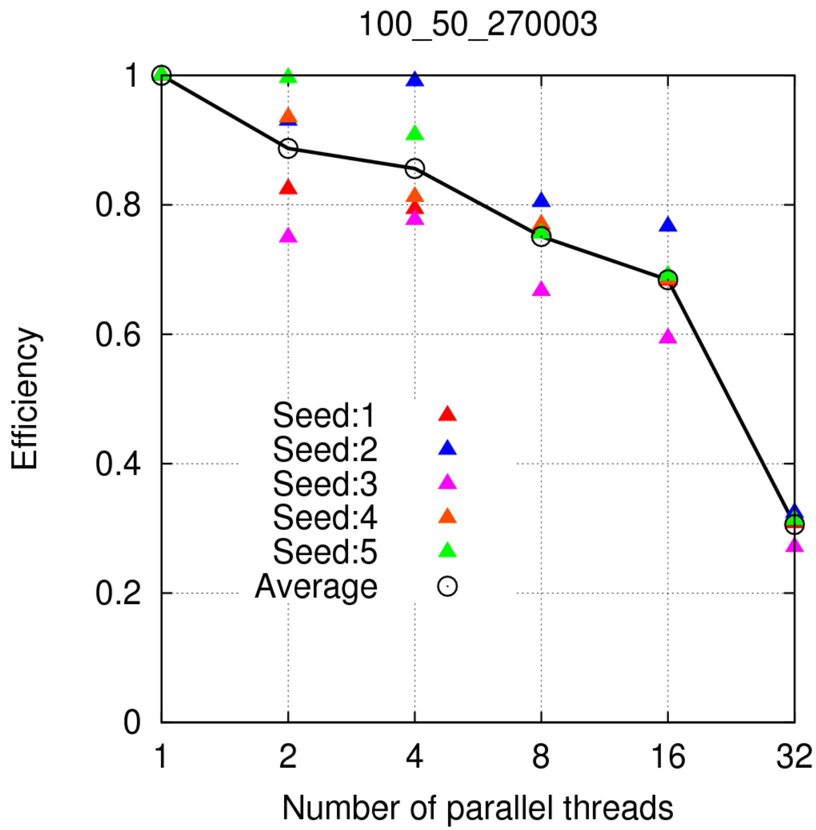


100 trials for each heuristic stopping when target solution was found or after 800s



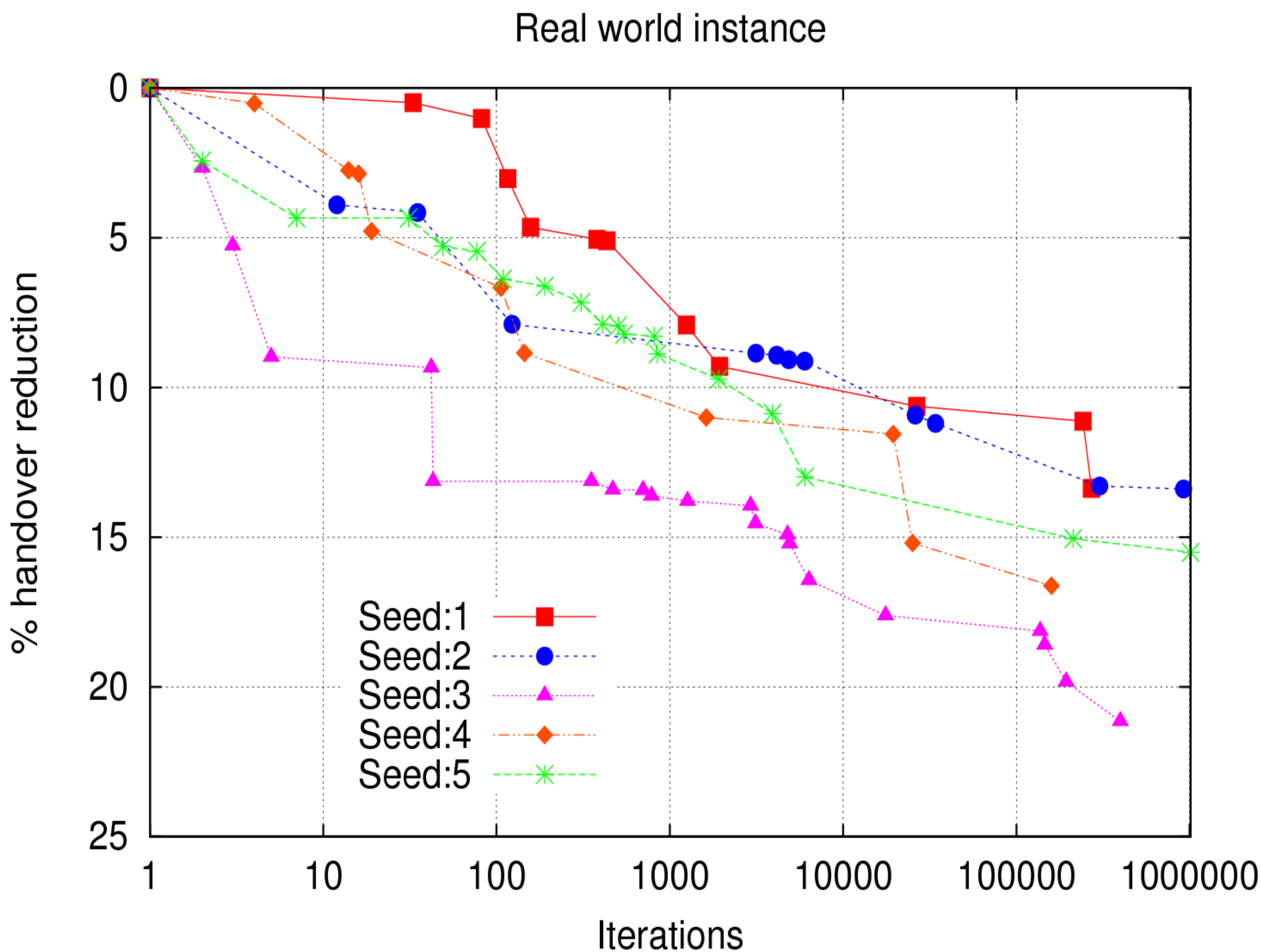
100 trials for each heuristic stopping when target solution was found or after 800s





Parallel decoding in BRKGA

Your world. Delivered.


Real world instance
0
5
10
15
20
25
% handover reduction
1
10
100
1000
10000
100000
1000000
Iterations
Seed:1
Seed:2
Seed:3
Seed:4
Seed:5

Concluding remarks

- We described the handover minimization problem (HMP).
- Objective of handover minimization is to reduce number of dropped calls in a cellular network.
- The HMP is a special case of the generalized quadratic assignment problem (GQAP).
- We described three randomized heuristics for the HMP and applied them on synthetic instances of the problem and one real instance.

Concluding remarks

- We described the handover minimization problem (HMP).
- Objective of handover minimization is to reduce number of dropped calls in a cellular network.
- The HMP is a special case of the generalized quadratic assignment problem (GQAP).
- We described three randomized heuristics for the the HMP and applied them on synthetic instances of the problem and one real instance. GRASP with evolutionary PR turns out to be the best (w.r.t to solution quality x solution time) so far ...

Thanks!

These slides as well as related technical reports are available at

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

Thanks!

Technical report: L.F. Morán-Mirabal, J.L. González-Velarde, MGCR, & R.M.A. Silva, "Randomized heuristics for handover minimization in mobiity networks" will be shortly available online at

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

at&t
Your world. Delivered.