GRASP heuristics for discrete and continuous global optimization

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

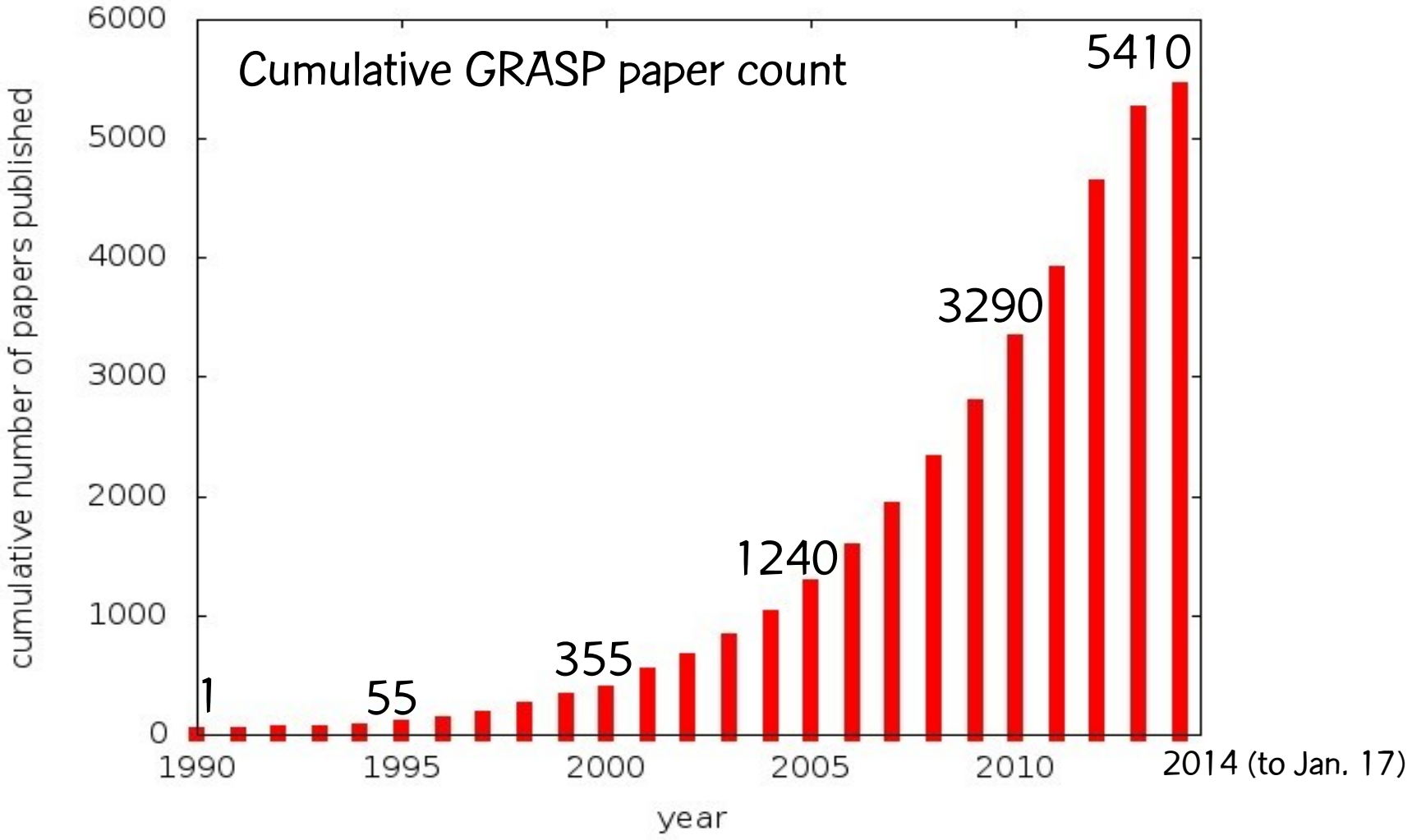
Talk given at Amazon.com, Inc.
Seattle, WA ✤ April 11, 2014

GRASP: The beginning

T.A. Feo & R.,"A probabilistic heuristic for a computationally difficult set covering problem," Oper. Res. Letters (1989)

T.A. Feo & R.,"Greedy randomized adaptive search procedures," J. of Global Opt. (1995)

Google Scholar Search: "greedy randomized adaptive search"
(http://scholar.google.com)



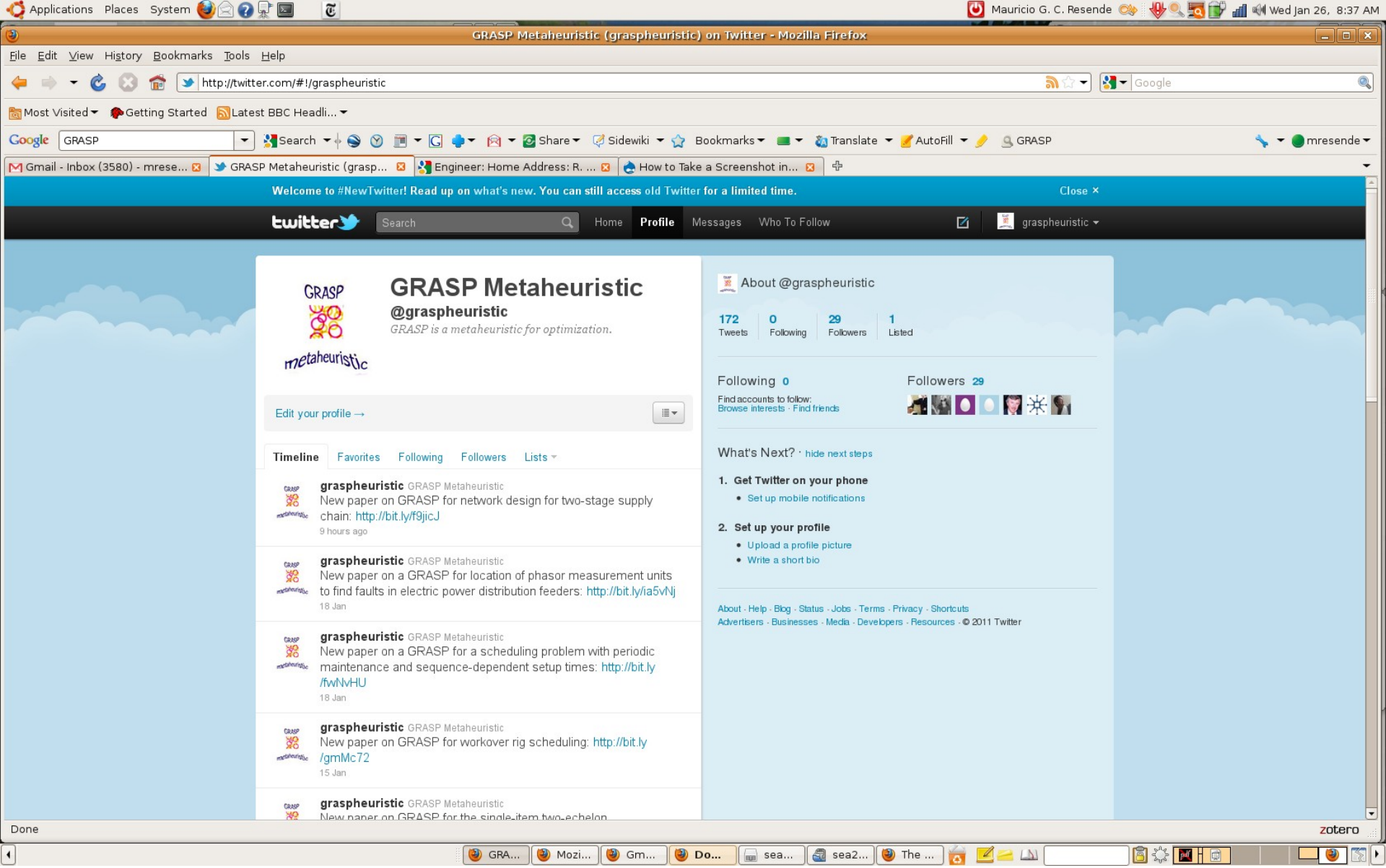
Amazon.com – April 11, 2014

Annotated bibliographies of GRASP

P. Festa and R., *GRASP: An annotated bibliography*, Essays and Surveys on Metaheuristics, C.C. Ribeiro and P. Hansen, Eds., Kluwer Academic Publishers, pp. 325-367, 2002

P. Festa and R., *An annotated bibliography of GRASP–Part I: Algorithms*, International Transactions in Operational Research, vol. 16, pp. 1-24, 2009.

P. Festa and R., *An annotated bibliography of GRASP–Part II: Applications*, International Transactions in Operational Research, vol. 16, pp. 131-172, 2009.



Follow GRASP on Twitter: http://twitter.com/graspheuristic

Summary

Combinatorial optimization and a review of GRASP

Neighborhoods, local search, greedy randomized construction and diversification

Hybrid construction

Other greedy randomized constructions, reactive GRASP, long-term memory in construction, biased sampling, cost perturbation

Summary

Hybridization with path-relinking

Elite sets, forward, backward, back and forward, mixed, greedy randomized adaptive path-relinking, evolutionary path-relinking

Some important developments not covered in talk

A recent real-world application of GRASP

Concluding remarks

Combinatorial Optimization

Combinatorial Optimization

Combinatorial optimization: process of finding the best, or optimal, solution for problems with a discrete set of feasible solutions.

Applications: e.g. routing, scheduling, packing, inventory and production management, location, logic, and assignment of resources.

Economic impact: e.g. transportation (airlines, trucking, rail, and shipping), forestry, manufacturing, logistics, aerospace, energy (electrical power, petroleum, and natural gas), agriculture, biotechnology, financial services, and telecommunications.

Combinatorial Optimization

Given:

discrete set of solutions X

objective function $f(x)$: $x \in X \rightarrow R$

Objective (minimization):

find $x \in X : f(x) \leq f(y), \forall\ y \in X$

Combinatorial Optimization

Much progress in recent years on finding exact (provably optimal) solutions: dynamic programming, cutting planes, branch and cut, ...

Many hard combinatorial optimization problems are still not solved exactly and require good solution methods.

Combinatorial Optimization

Approximation algorithms are guaranteed to find in polynomial-time a solution within a given factor of the optimal.

Combinatorial Optimization

Approximation algorithms are guaranteed to find in polynomial-time a solution within a given factor of the optimal.

Sometimes the factor is too big, i.e. guaranteed solutions are far from optimal

Some optimization problems (e.g. max clique, covering by pairs) cannot have approximation schemes unless P=NP

Combinatorial Optimization

Aim of heuristic methods for combinatorial optimization is to quickly produce good-quality solutions, without necessarily providing any guarantee of solution quality.

Metaheuristics

Metaheuristics are heuristics to devise heuristics.

Examples: simulated annealing, genetic algorithms, tabu search, scatter search, ant colony optimization, variable neighborhood search, and GRASP.

Metaheuristics

Metaheuristics are high level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristics alone.

Examples: simulated annealing, genetic algorithms, tabu search, scatter search, ant colony optimization, variable neighborhood search, and GRASP.

Metaheuristics

Metaheuristics are high level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristics alone.

Examples: simulated annealing, genetic algorithms, tabu search, scatter search, ant colony optimization, variable neighborhood search, and GRASP.

Review of GRASP: Local Search

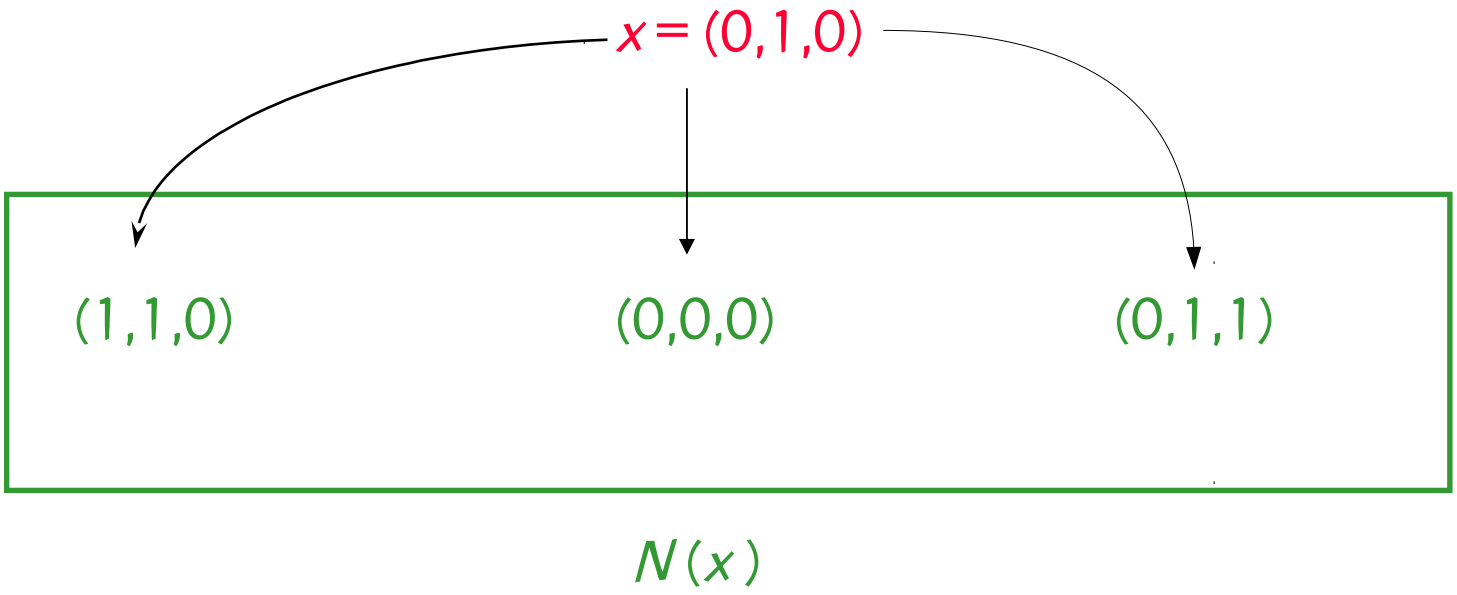
Local Search

To define local search, one needs to specify a local neighborhood structure.

Given a solution x , the elements of the neighborhood N(x) of x are those solutions y that can be obtained by applying an elementary modification (often called a move) to x.

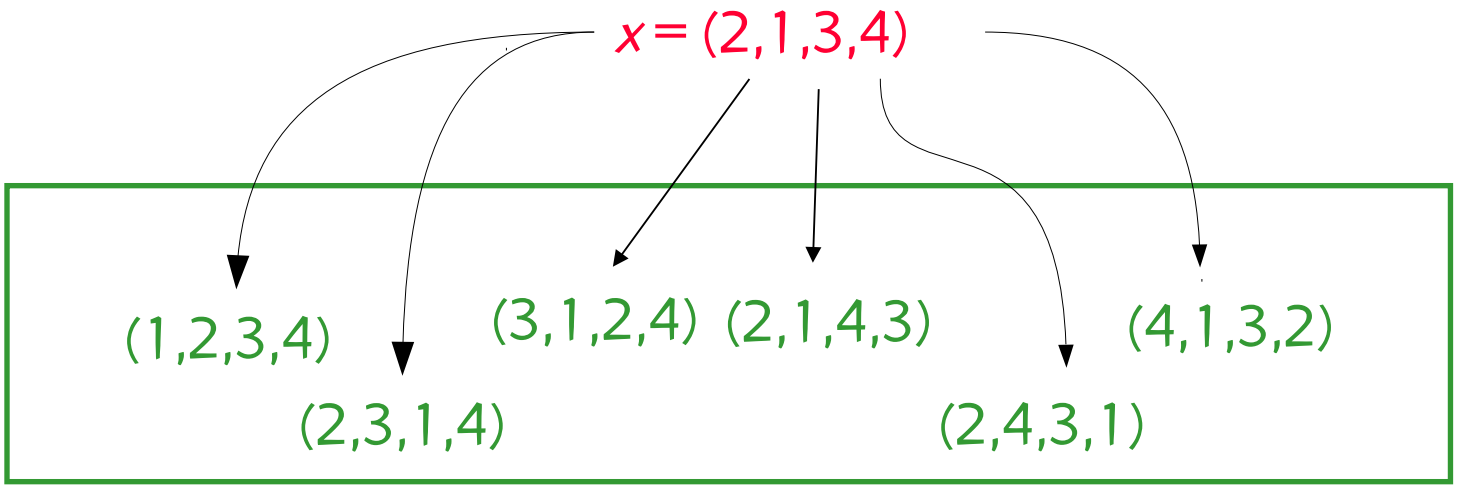
Local Search Neighborhoods

Consider $x = (0,1,0)$ and the 1-flip neighborhood of a 0/1 array.



Local Search Neighborhoods

Consider $x = (2,1,3,4)$ and the 2-swap neighborhood of a permutation array.



$N(x) = C(4,2) = 6$

Local Search

Given an initial solution x_0, a neighborhood N(x), and function f(x) to be minimized:

$x = x_0$;

while ($\exists$ y $\in$ N(x) | f(y) < f(x)) {

    x = y ;

}

check for better solution in neighborhood of *x*

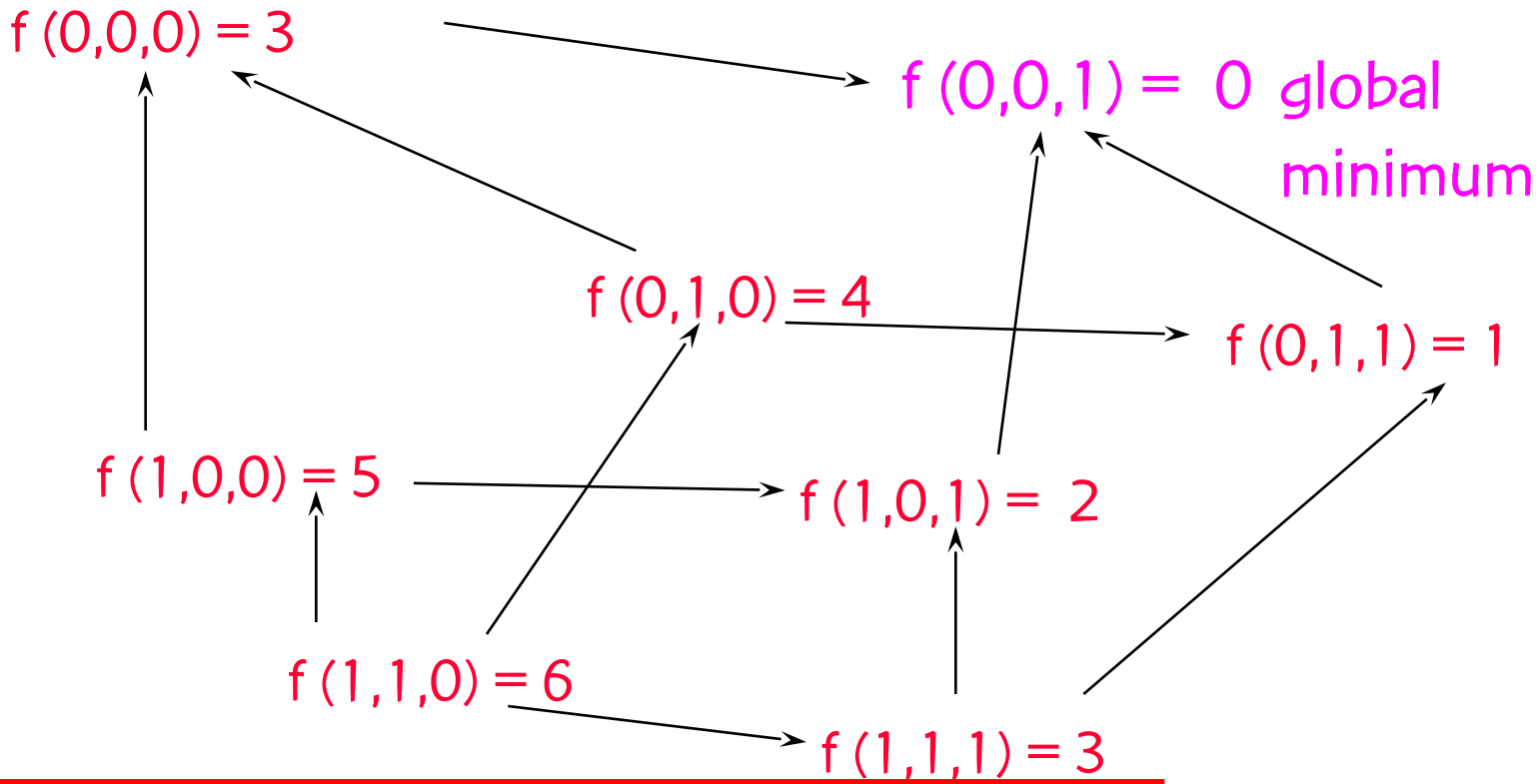
move to better solution *y*

Time complexity of local search can be exponential.

At the end, x is a local minimum of f(x) .

Local Search

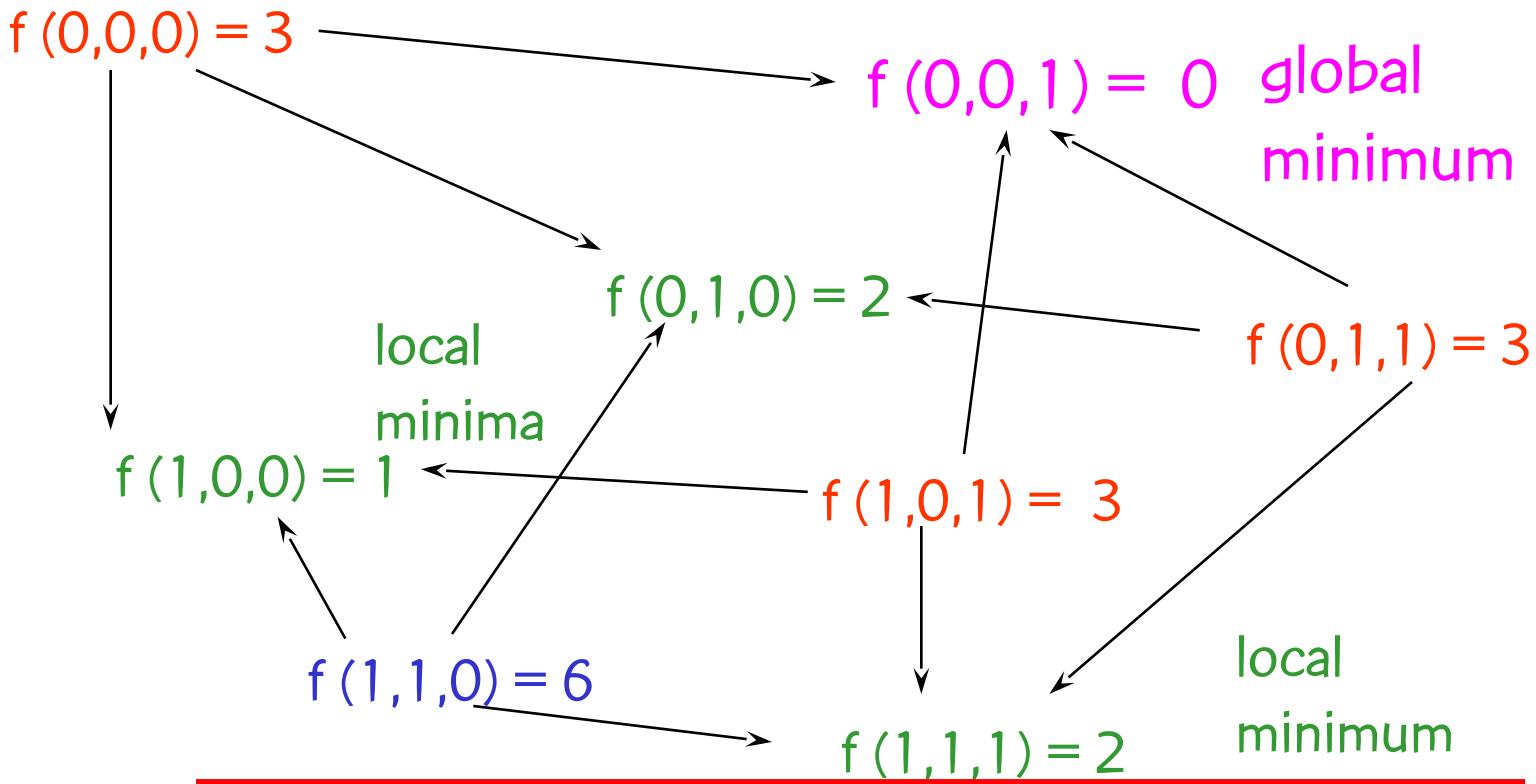
(ideal situation)



With any starting solution Local Search finds the global optimum.

Local Search

(more realistic situation)



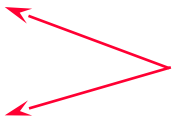
But some starting solutions lead Local Search to a local minimum.

Local Search

Effectiveness of local search depends on several factors:

neighborhood structure

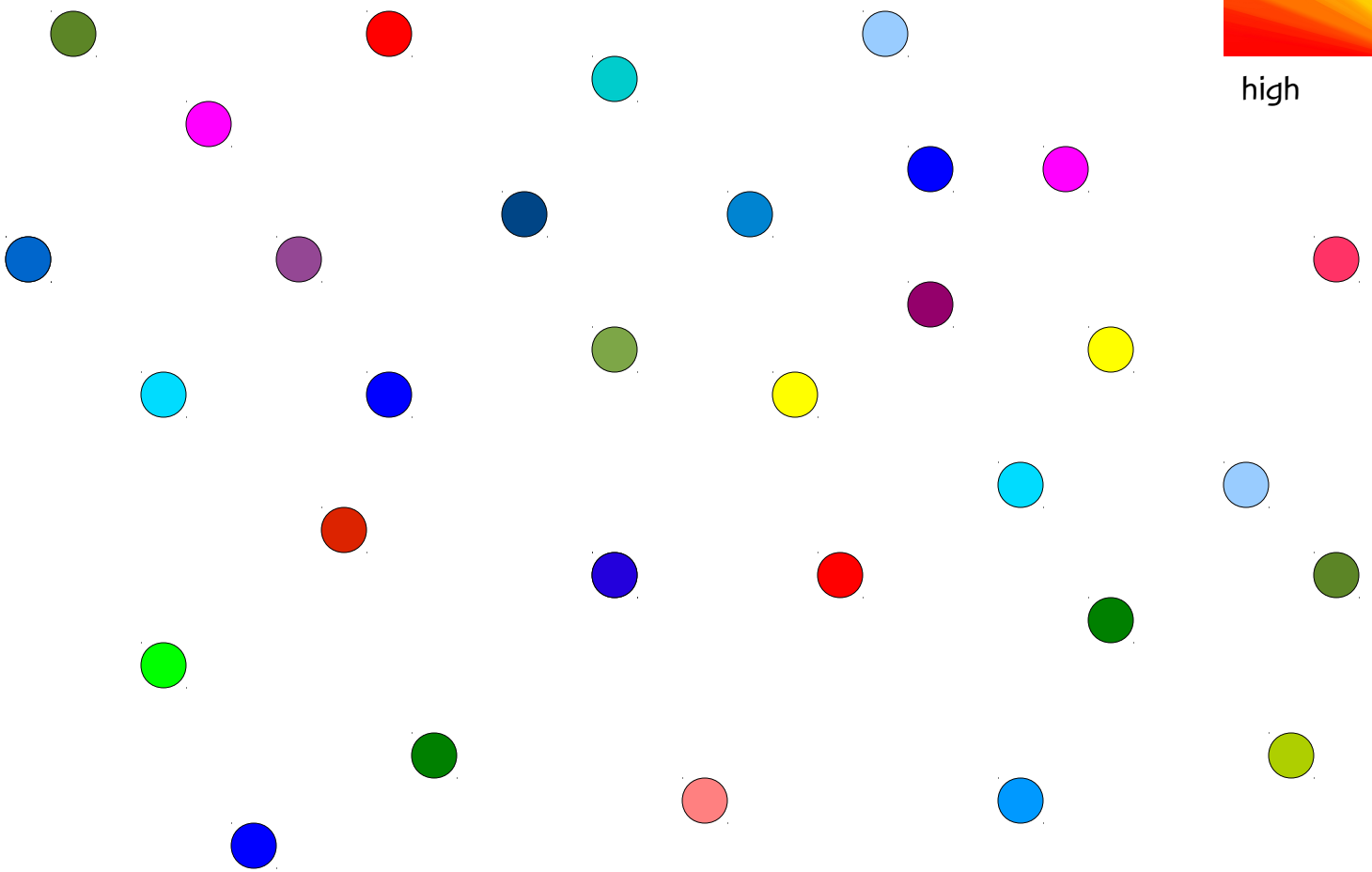
function to be minimized

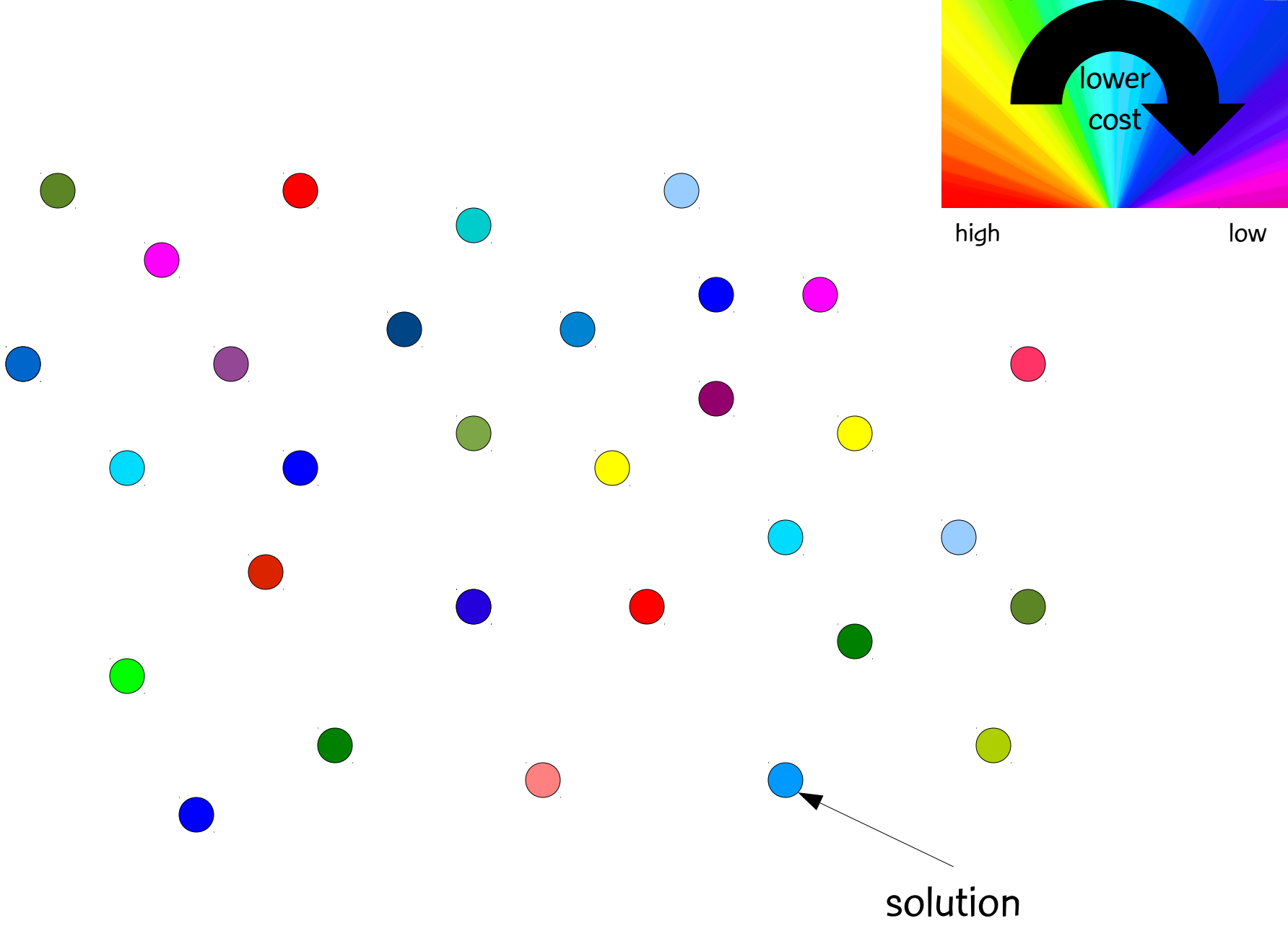


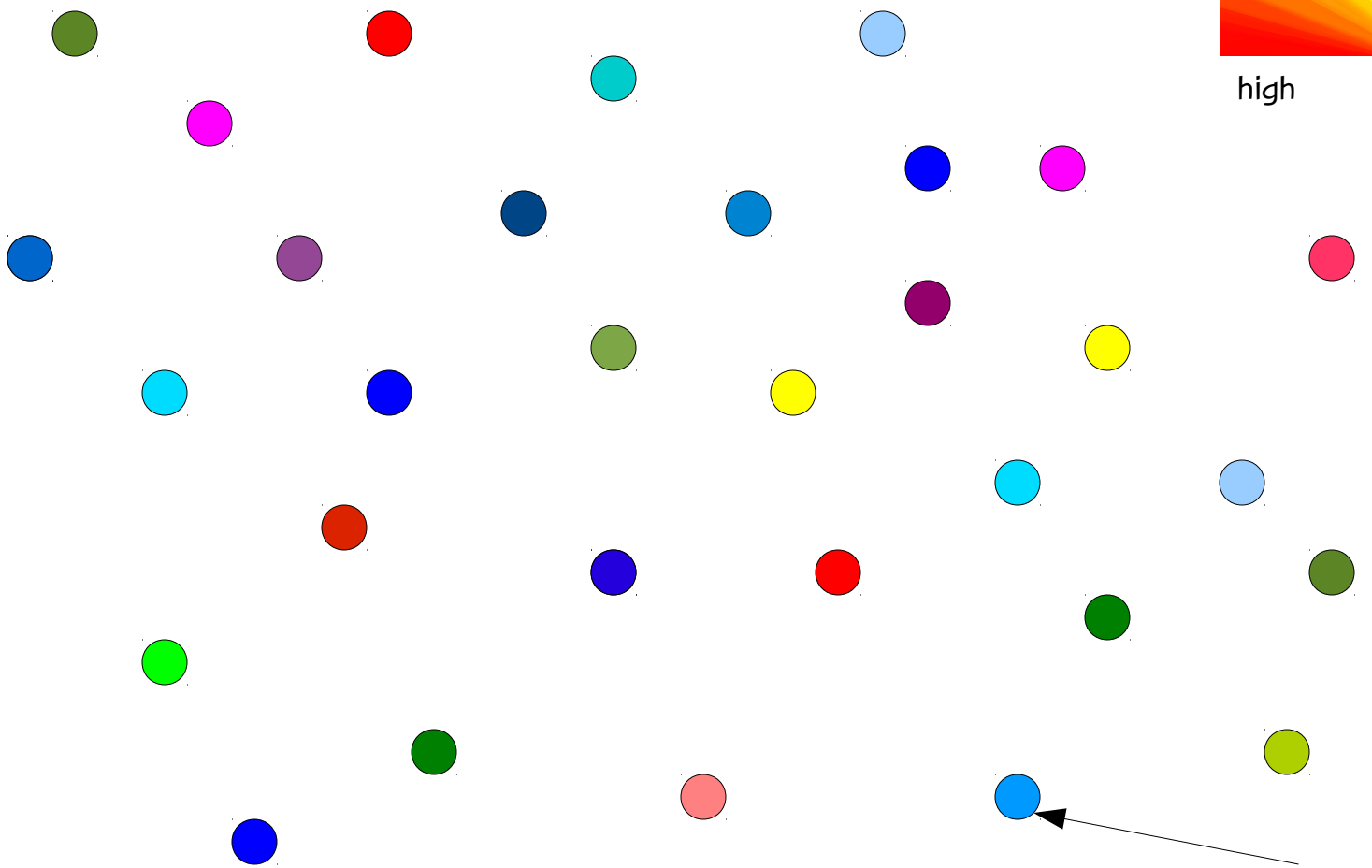
usually pre-determined

starting solution

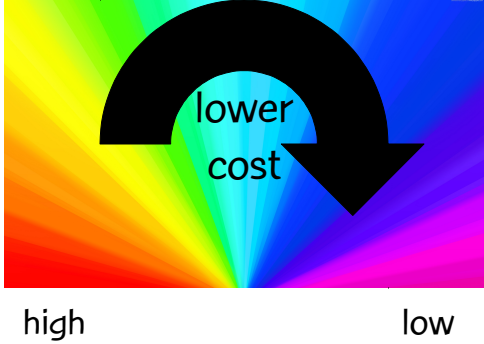
usually easier to control

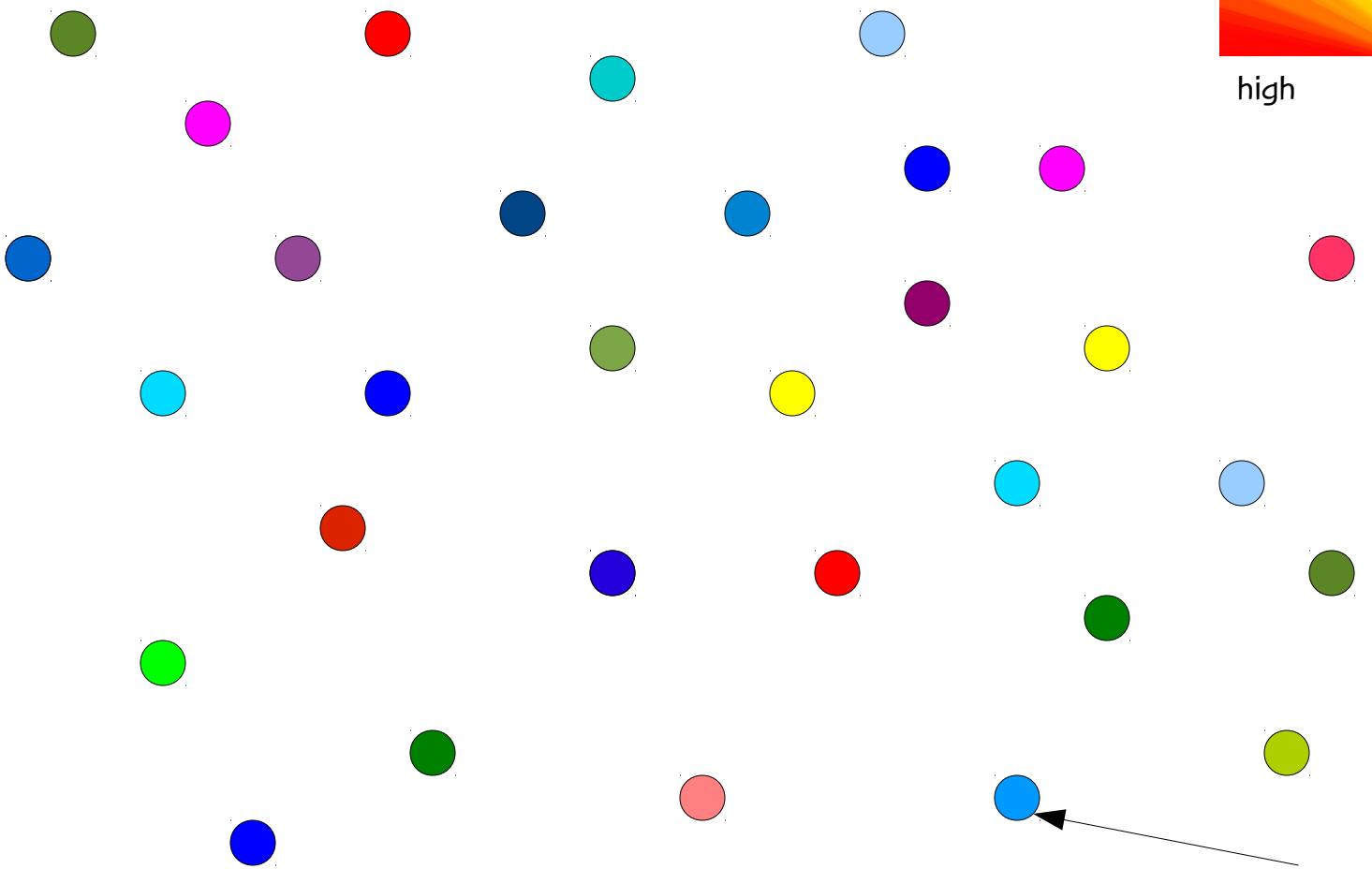

lower
cost
high
low


lower
cost
high
low
solution


lower
cost
high
low
Solution
(color represents cost)

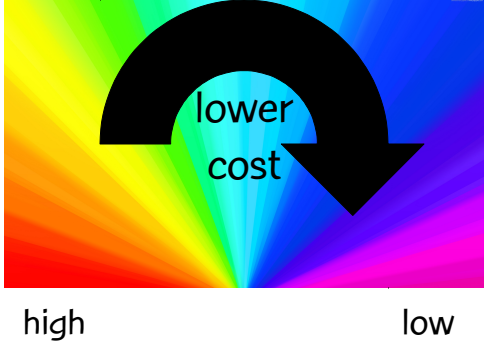
Combinatorial optimization: Find solution with min cost





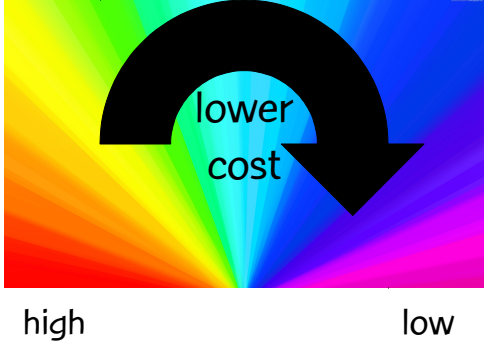
Solution
(color represents cost)

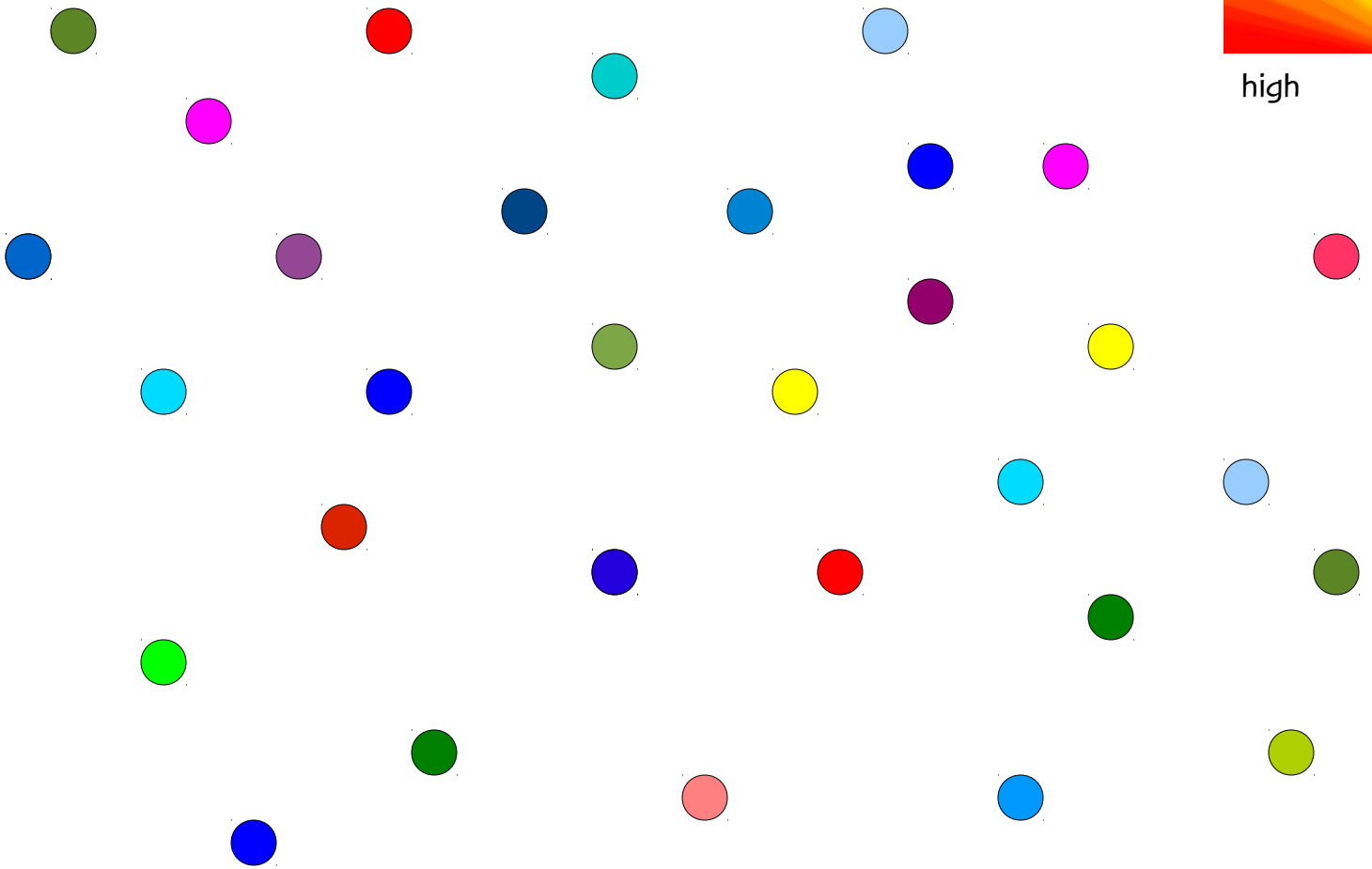
Combinatorial optimization: Find needle in haystack



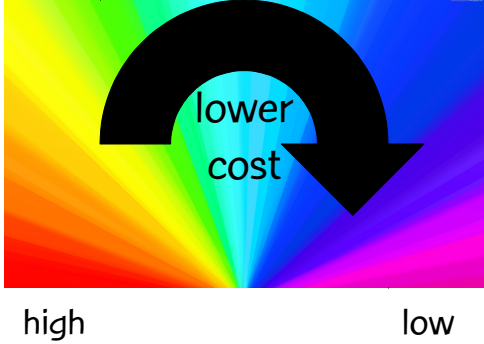
Solution
(color represents cost)

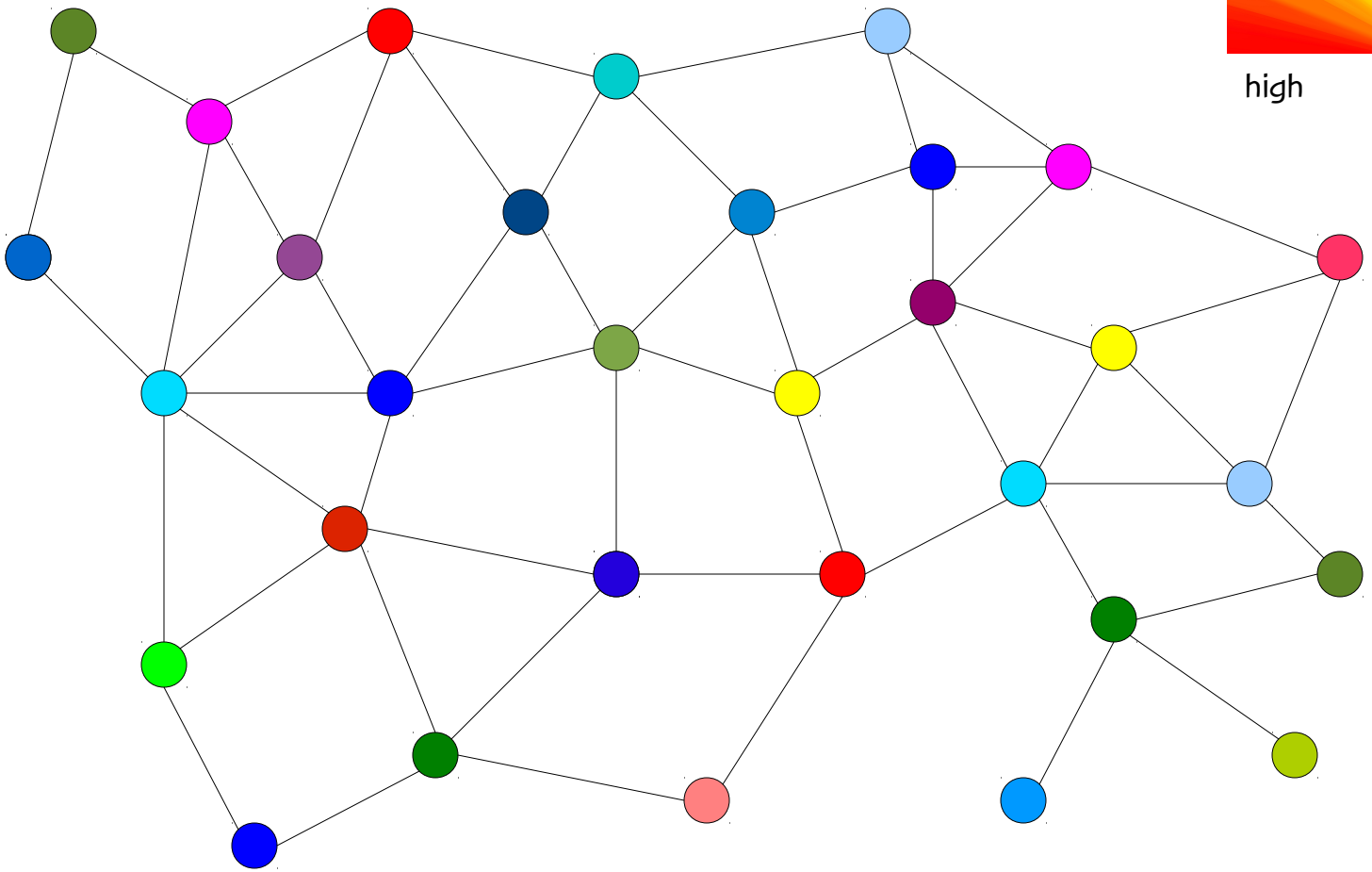
Solutions that differ slightly (in structure) are said to be in each other's neighborhood



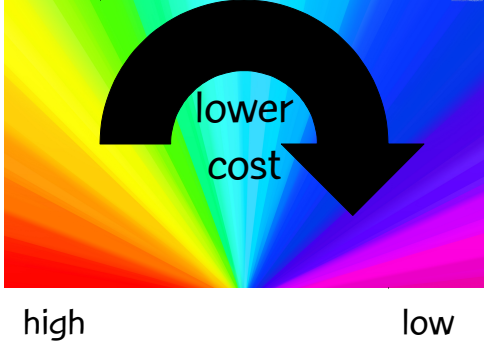


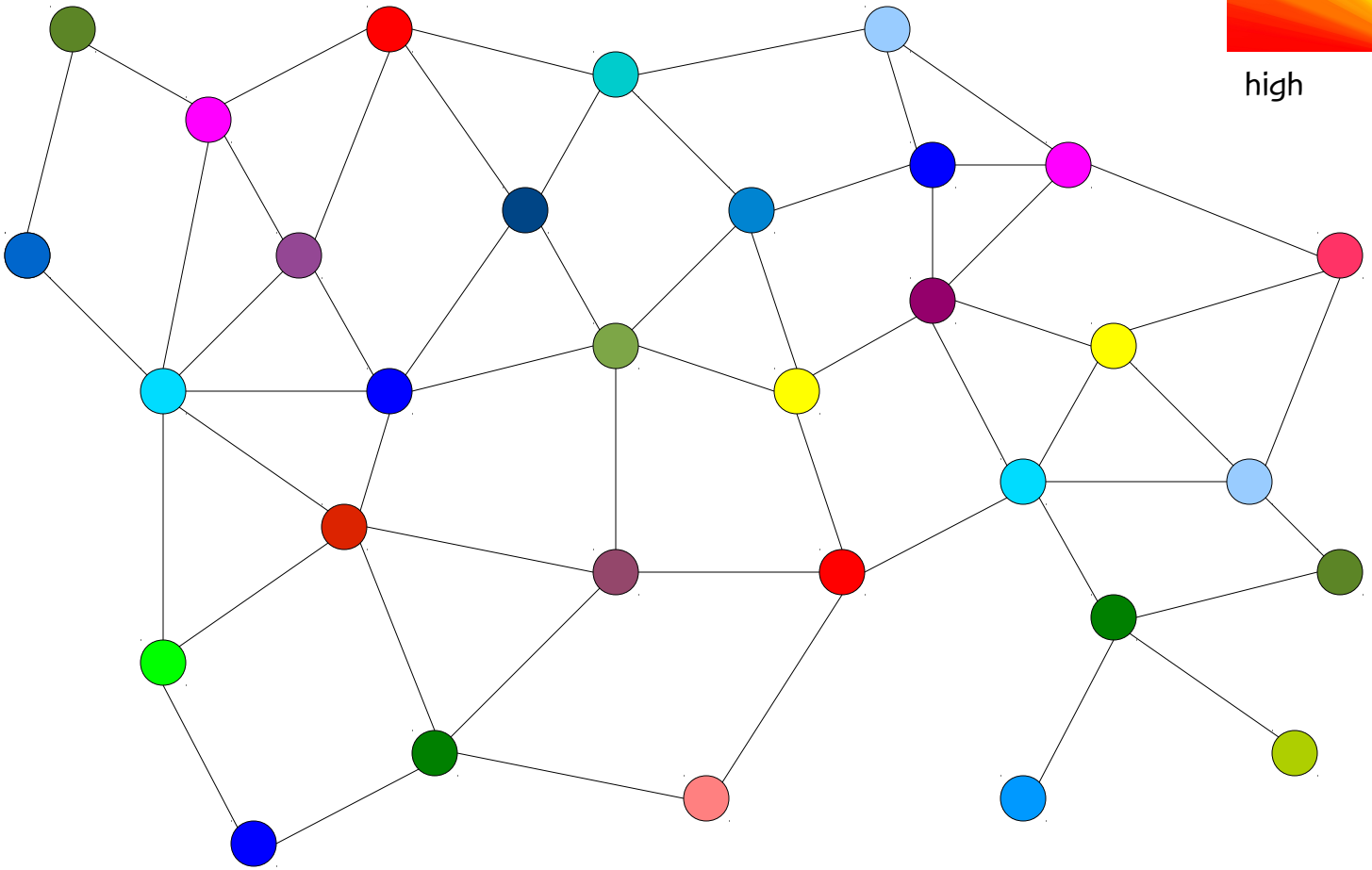
Solutions that differ slightly (in structure) are said to be in each other's neighborhood



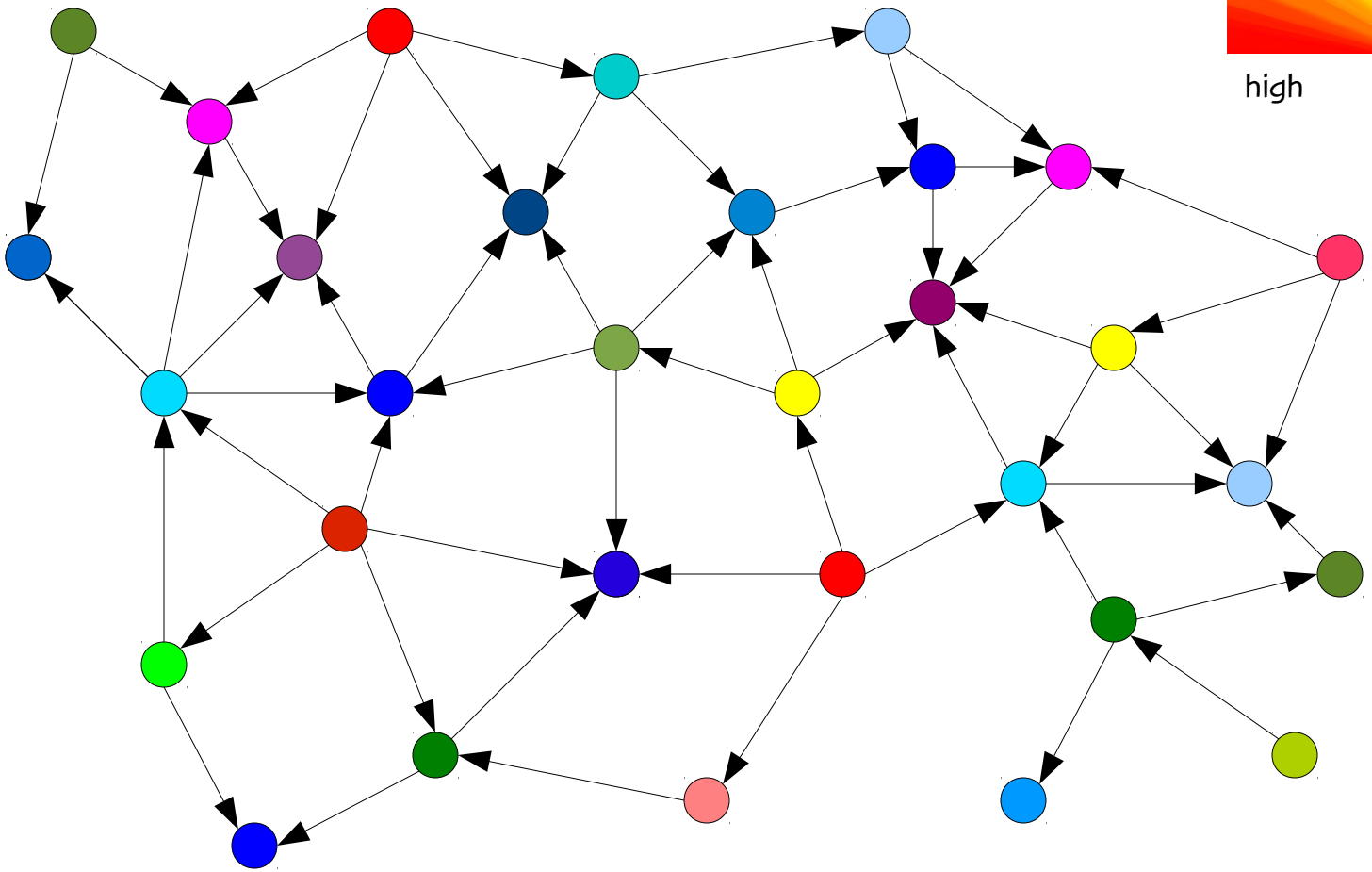


Two solutions is same neighborhood can be reached from one another my means of a move

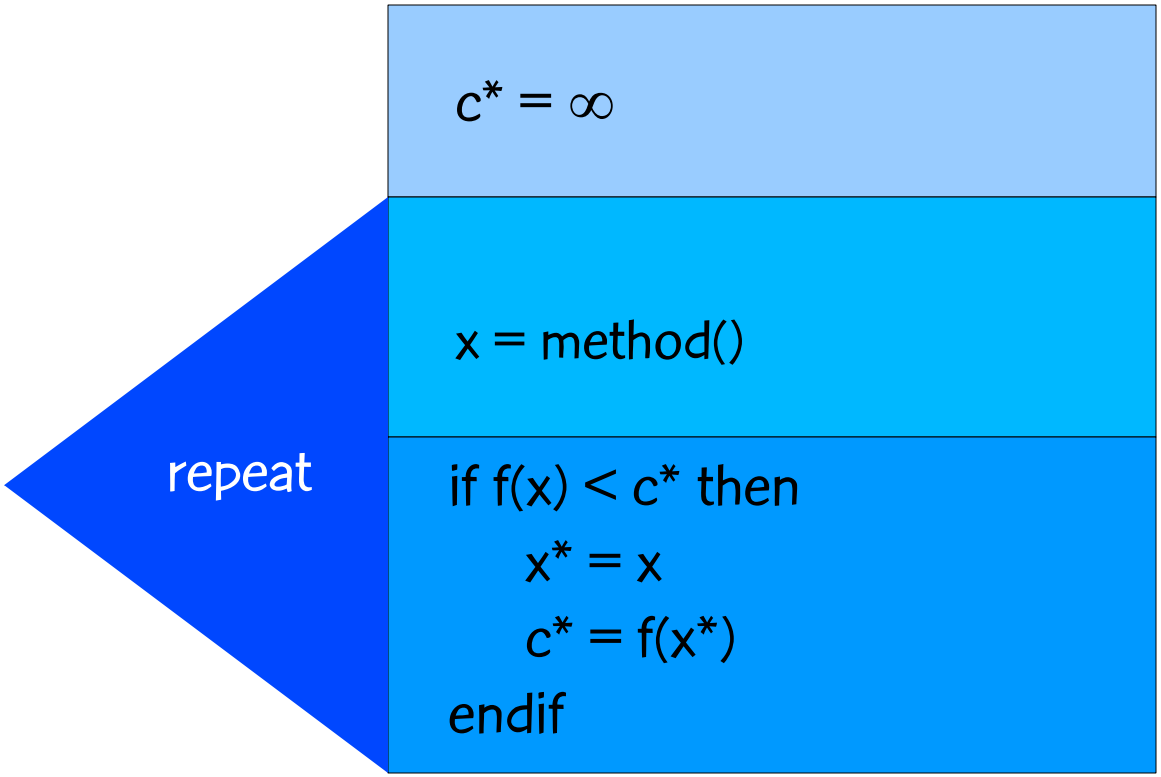




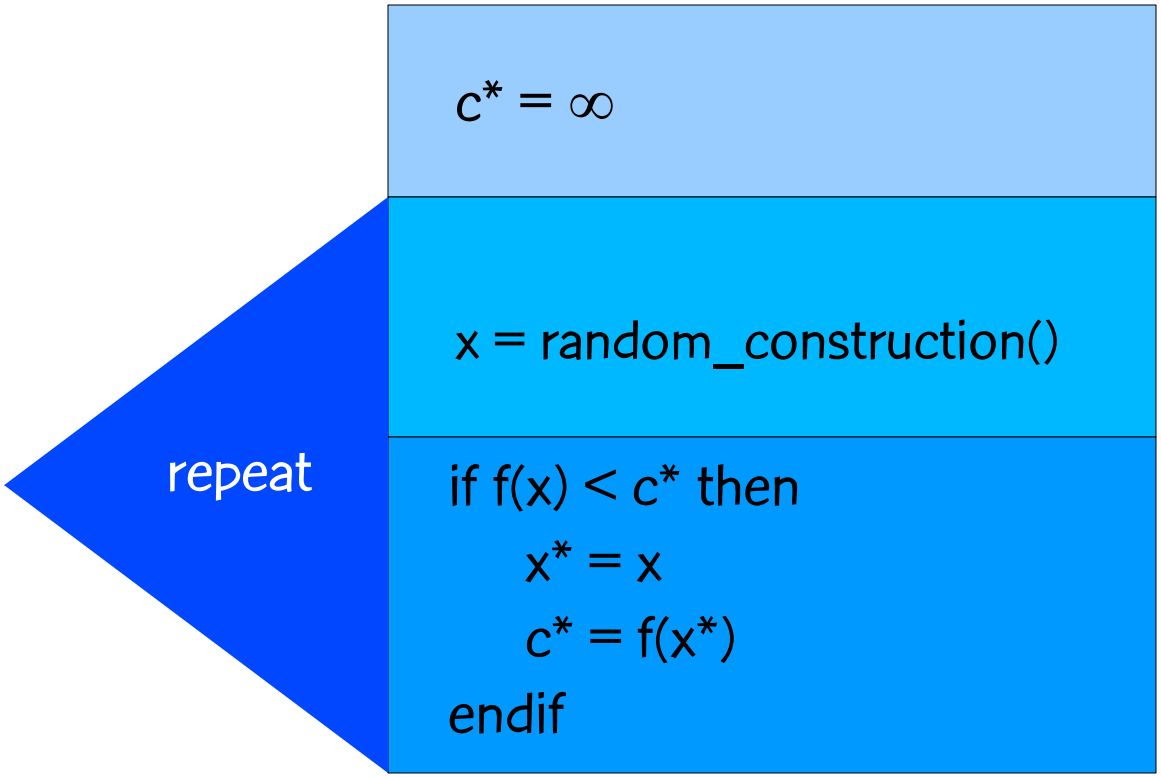
Since solutions have costs, cost-improving moves can be defined



Multi-start method



Random multi-start



Example: probability of finding opt by random selection

Suppose x = (0/1, 0/1, 0/1, 0/1, 0/1) and let the unique optimum be x* = (1,0,0,1,1).

The prob of finding the opt at random is 1/32 = .031 and the prob of not finding it is 31/32.

After k trials, the probability of not finding the opt is $(31/32)^k$ and hence the prob of find it at least once is $1 - (31/32)^k$

For k = 5, p = .146; for k = 10, p = .272; for k = 20, p = .470; for k = 50, p = .796; for k = 100, p = .958; for k = 200, p = .998

Example: Probability of finding opt with K samplings on a 0–1 vector of size N

N:	10	15	20	25	30
K:					
10	.010	.000	.000	.000	.000
100	.093	.003	.000	.000	.000
1000	.624	.030	.000	.000	.000
10000	1.000	.263	.009	.000	.000
100000	1.000	.953	.091	.003	.000

Greedy algorithm

The greedy algorithm

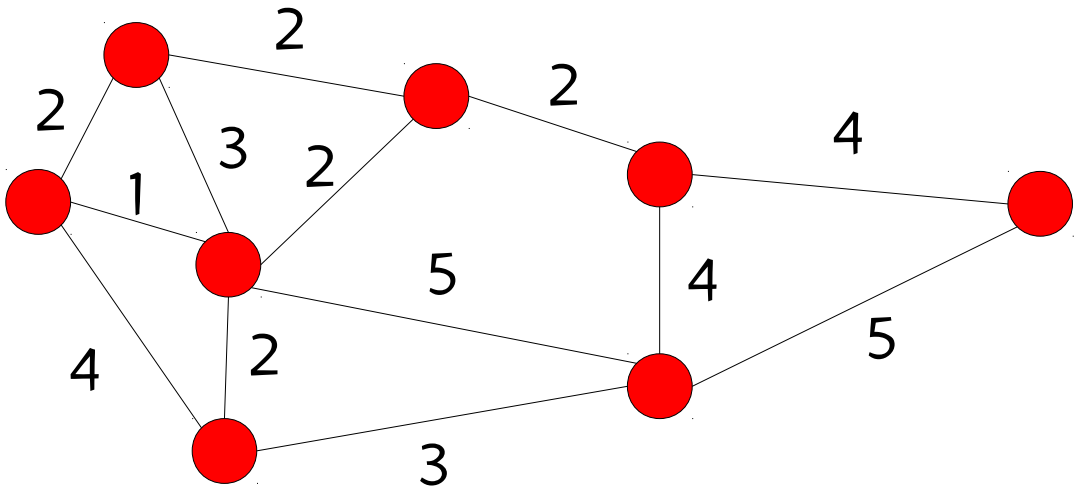
Constructs a solution, one element at a time:

repeat until done

- Defines candidate elements.
- Applies a greedy function to each candidate element.
- Ranks elements according to greedy function value.
- Add best ranked element to solution.

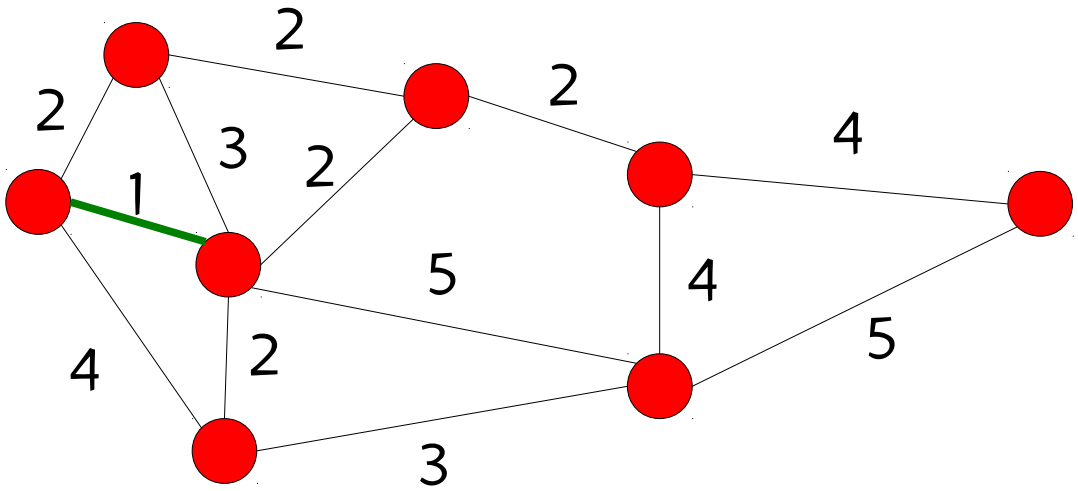
The greedy algorithm

An example: minimum weight spanning tree



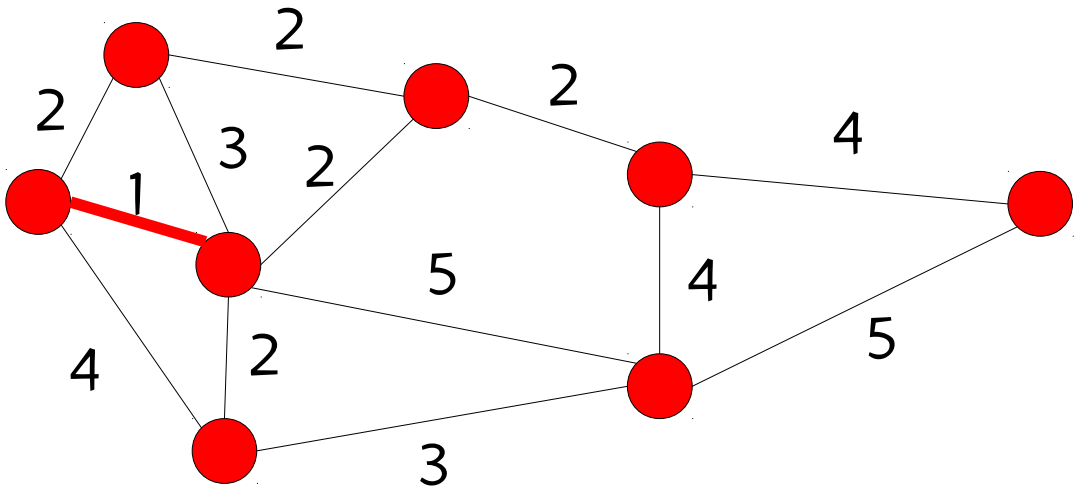
The greedy algorithm

An example: minimum weight spanning tree



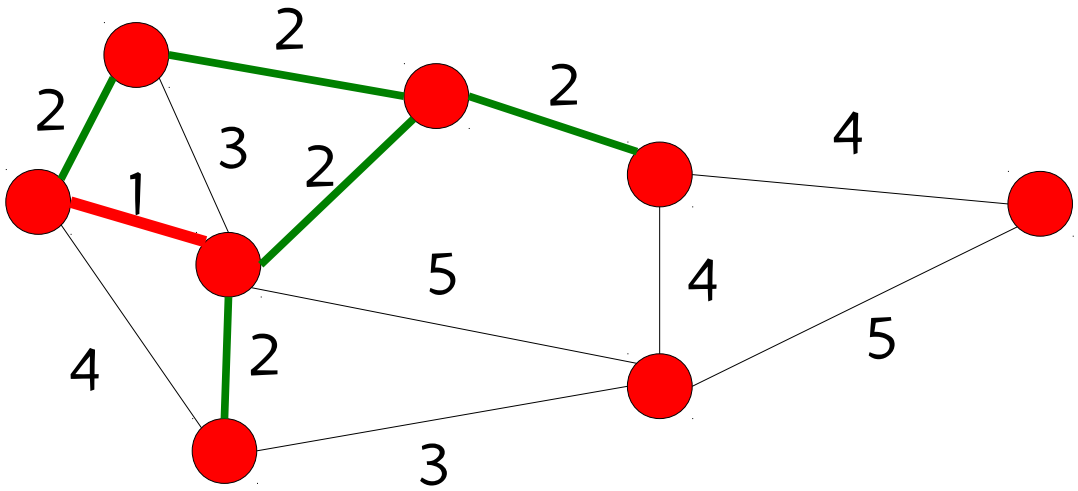
The greedy algorithm

An example: minimum weight spanning tree



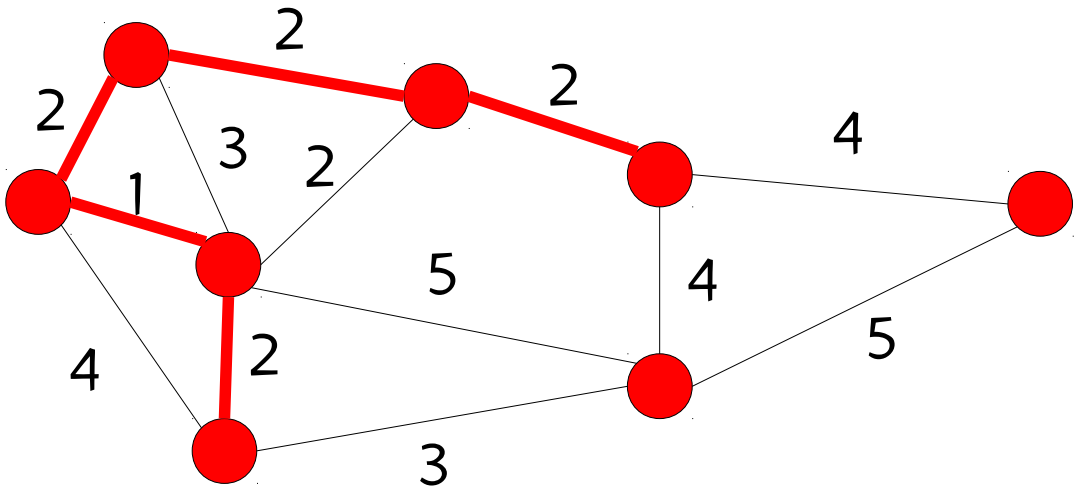
The greedy algorithm

An example: minimum weight spanning tree



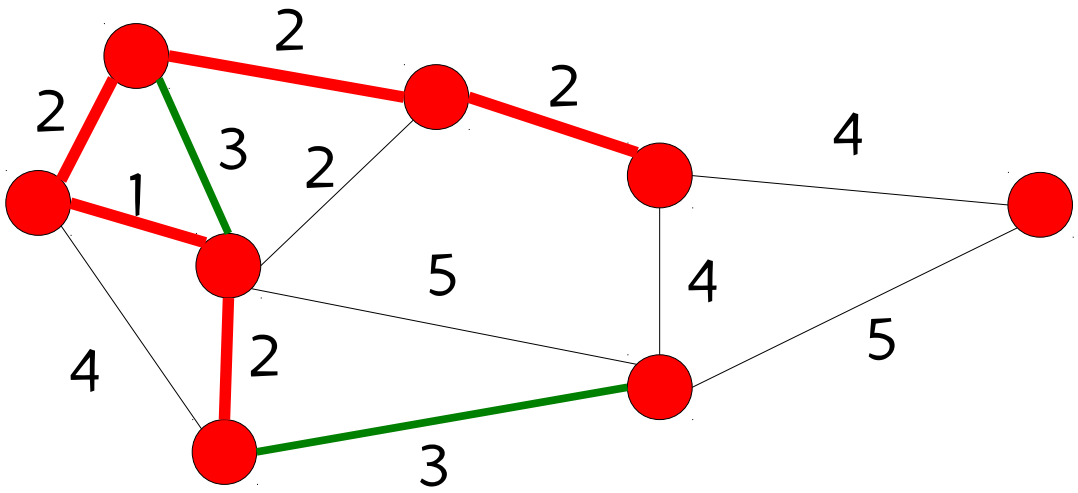
The greedy algorithm

An example: minimum weight spanning tree



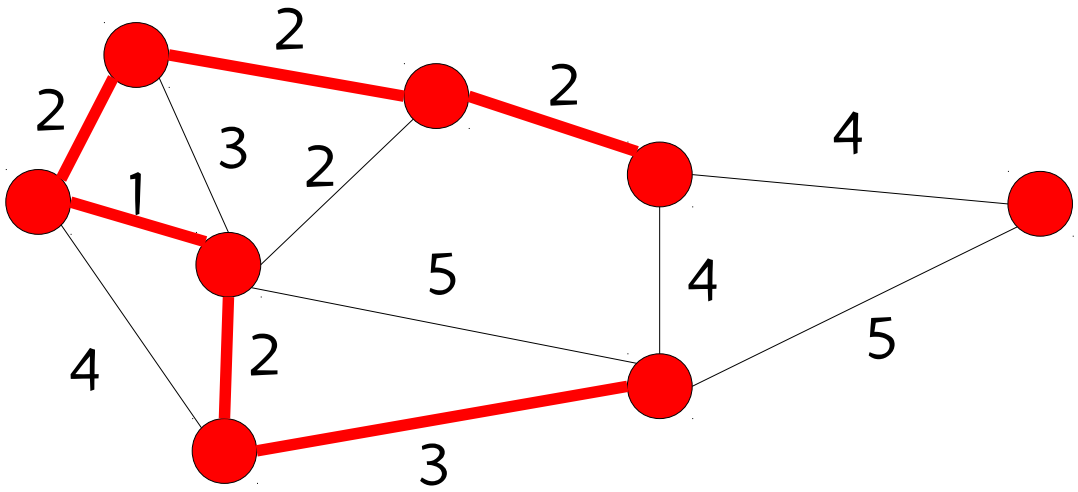
The greedy algorithm

An example: minimum weight spanning tree



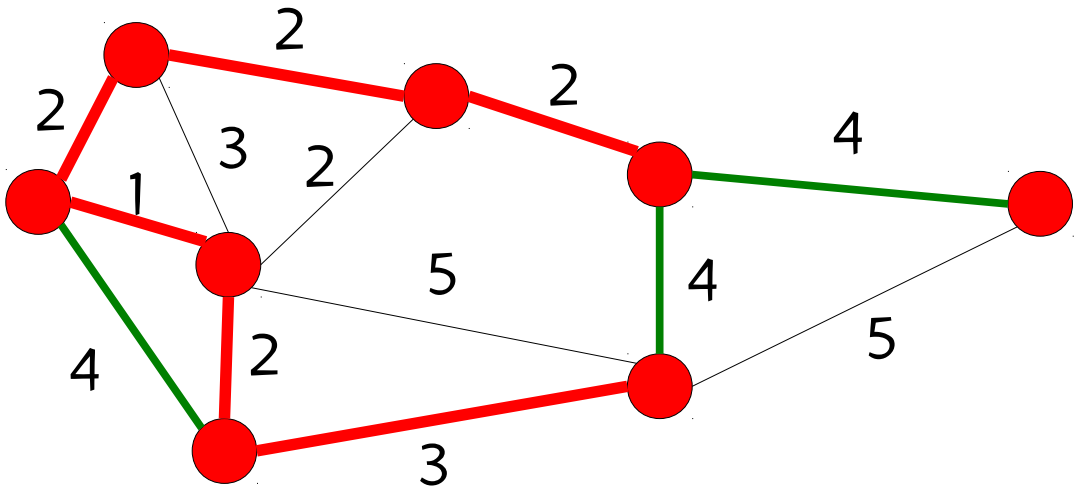
The greedy algorithm

An example: minimum weight spanning tree



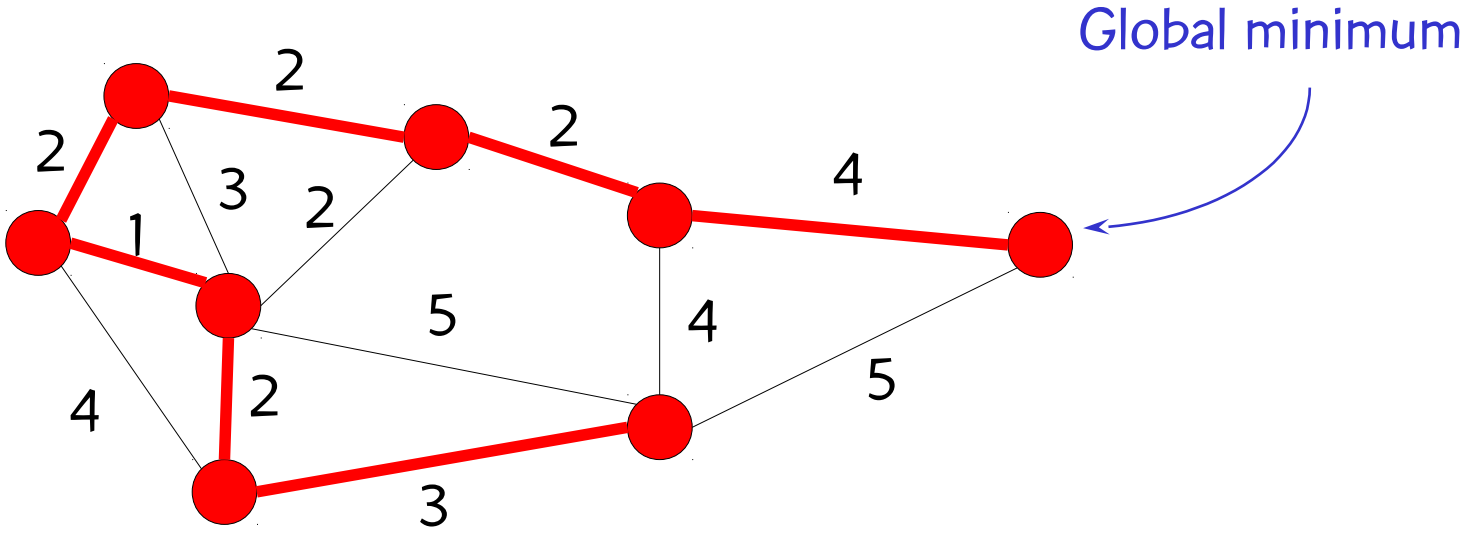
The greedy algorithm

An example: minimum weight spanning tree



The greedy algorithm

An example: minimum weight spanning tree



The greedy algorithm

Another example: Maximum clique

Given graph $G = (V, E)$, find largest subgraph of G such that all vertices are mutually adjacent.

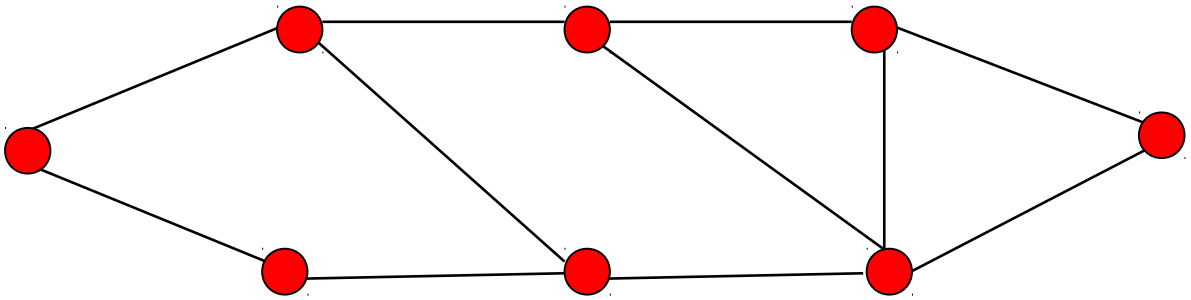
greedy algorithm builds solution, one element (vertex) at a time

candidate set: unselected vertices adjacent to all selected vertices

greedy function: vertex degree with respect to other candidate set vertices.

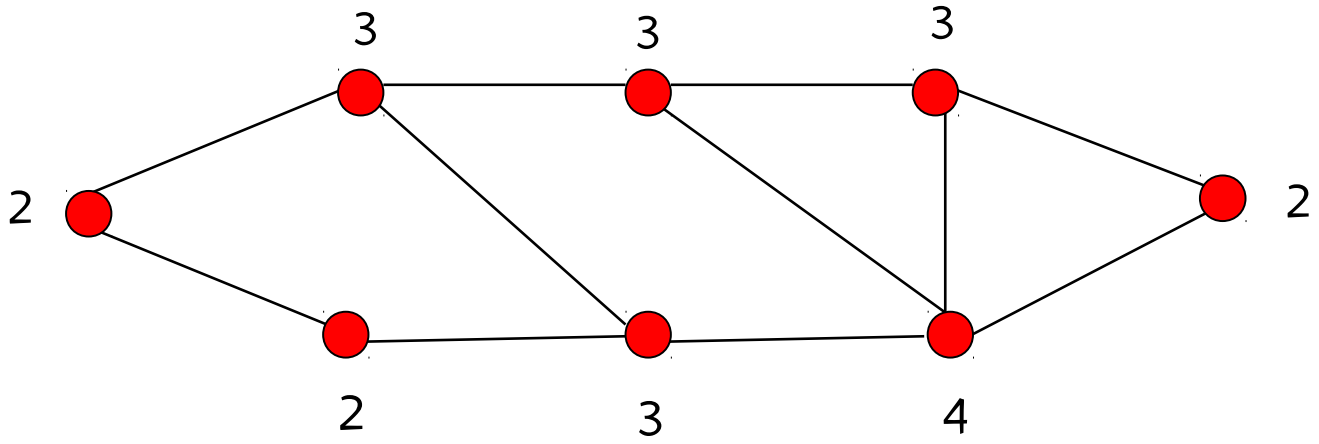
The greedy algorithm

Another example: Maximum clique



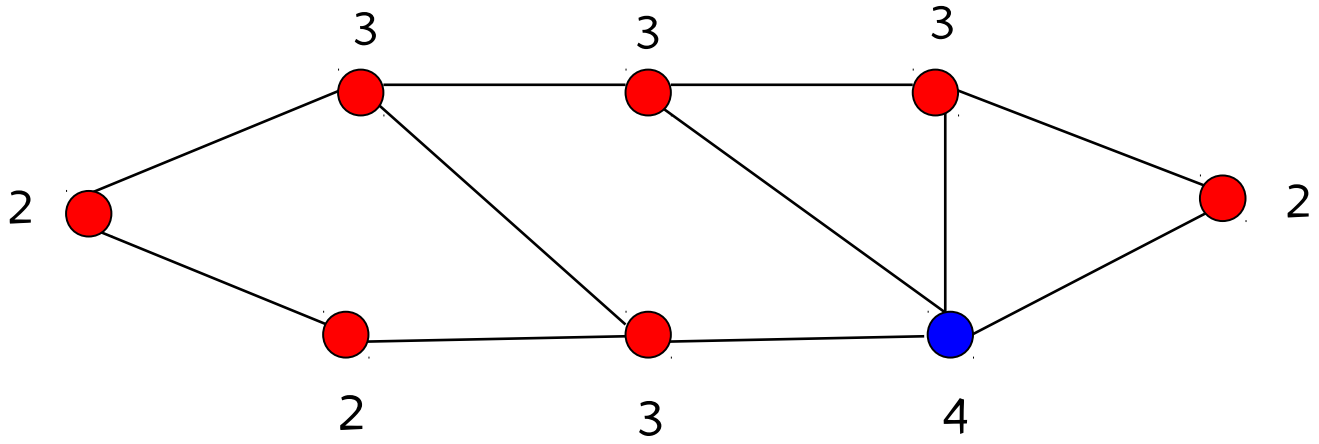
The greedy algorithm

Another example: Maximum clique



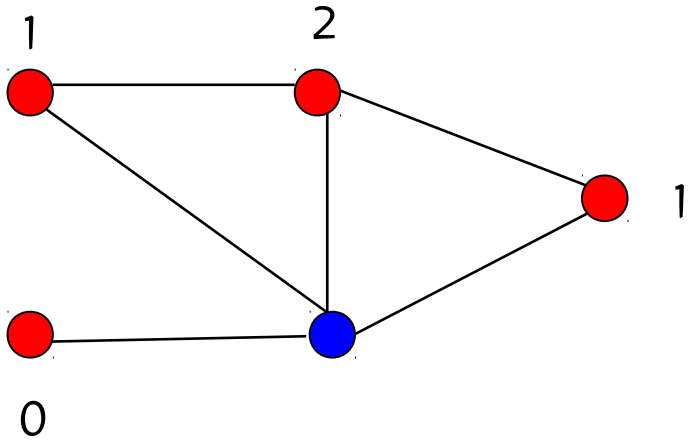
The greedy algorithm

Another example: Maximum clique



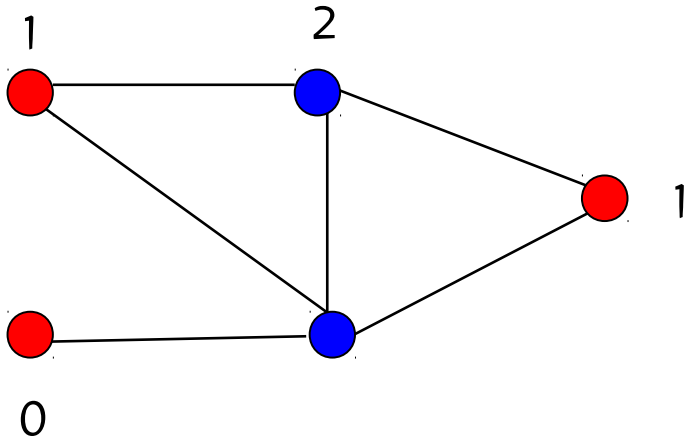
The greedy algorithm

Another example: Maximum clique



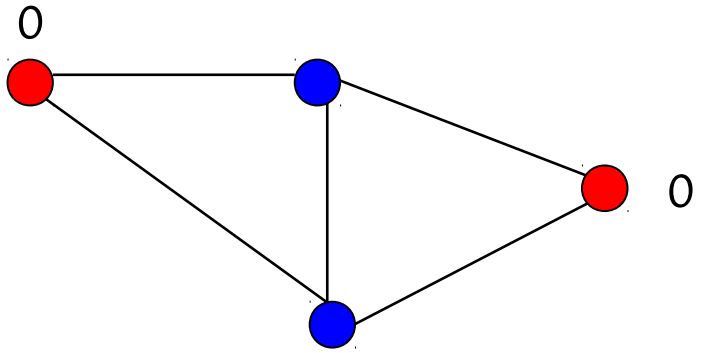
The greedy algorithm

Another example: Maximum clique



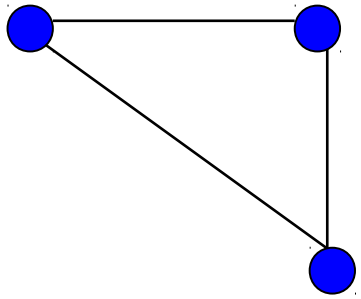
The greedy algorithm

Another example: Maximum clique



The greedy algorithm

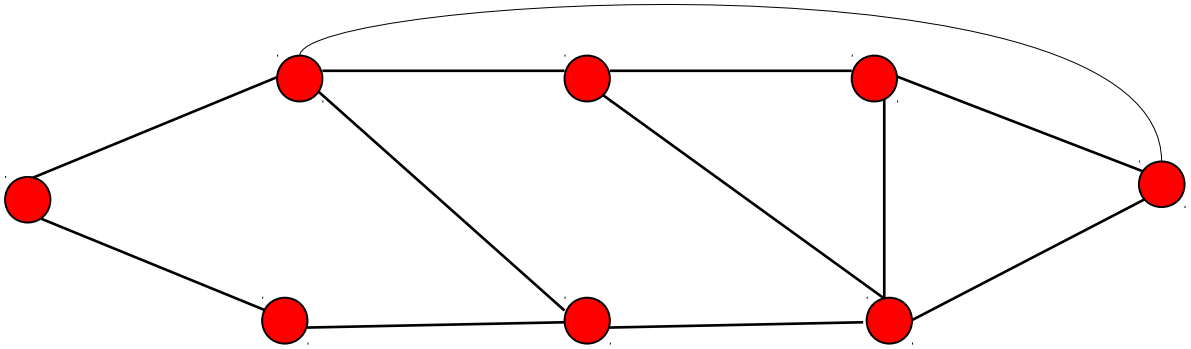
Another example: Maximum clique



global maximum

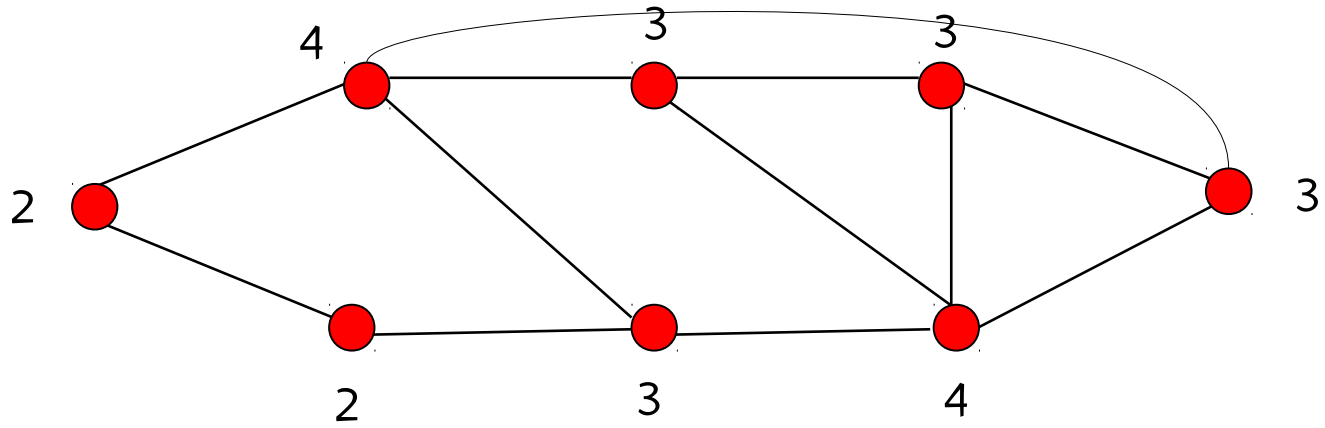
The greedy algorithm

Another example: Maximum clique



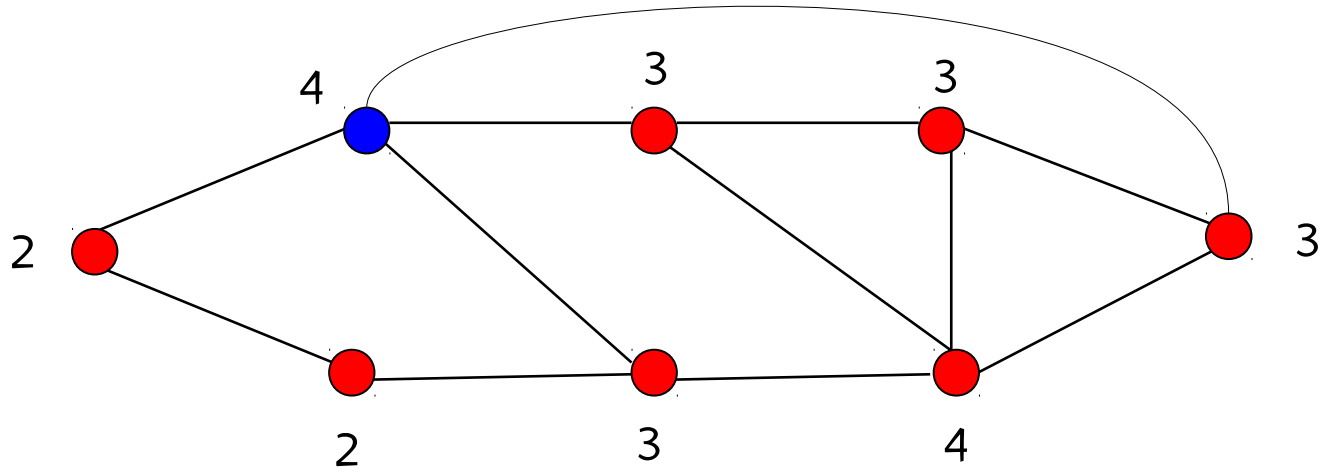
The greedy algorithm

Another example: Maximum clique



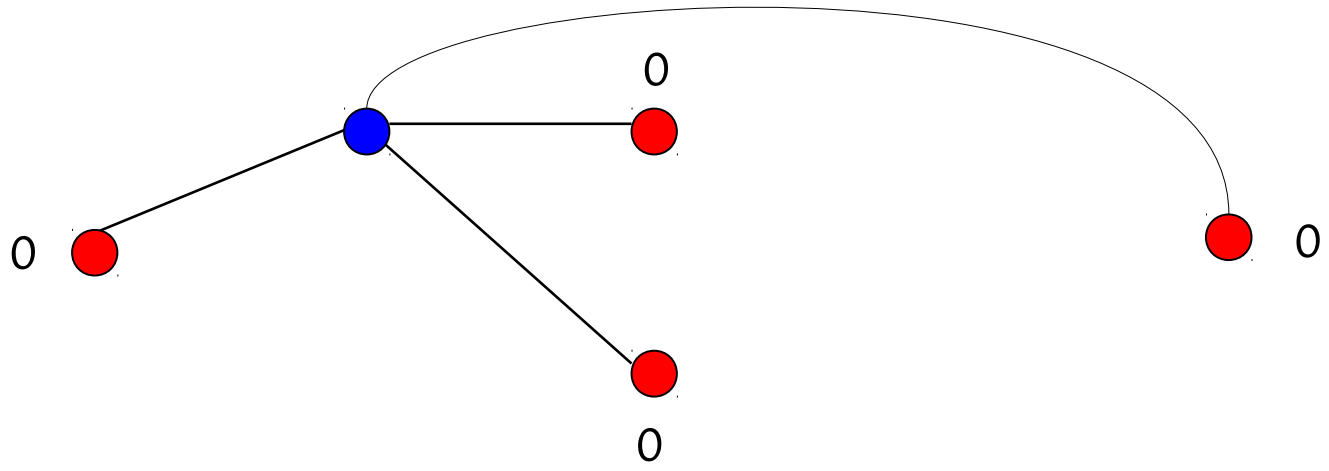
The greedy algorithm

Another example: Maximum clique



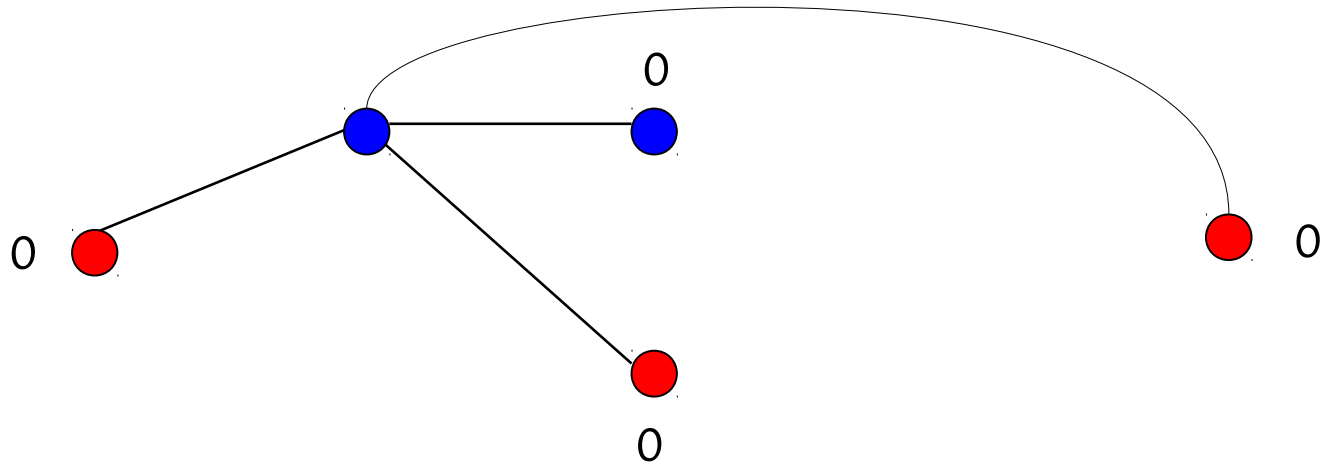
The greedy algorithm

Another example: Maximum clique



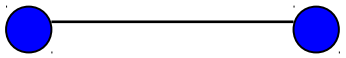
The greedy algorithm

Another example: Maximum clique



The greedy algorithm

Another example: Maximum clique



sub-optimal clique

Semi-greedy heuristic

A semi-greedy heuristic tries to get around convergence to non-global local minima.

repeat until solution is constructed

repeat until done

For each candidate element

apply a greedy function to element

Rank all elements according to their greedy function values

Place well-ranked elements in a restricted candidate list (RCL)

Select an element from the RCL at random & add it to the solution

Semi-greedy heuristic

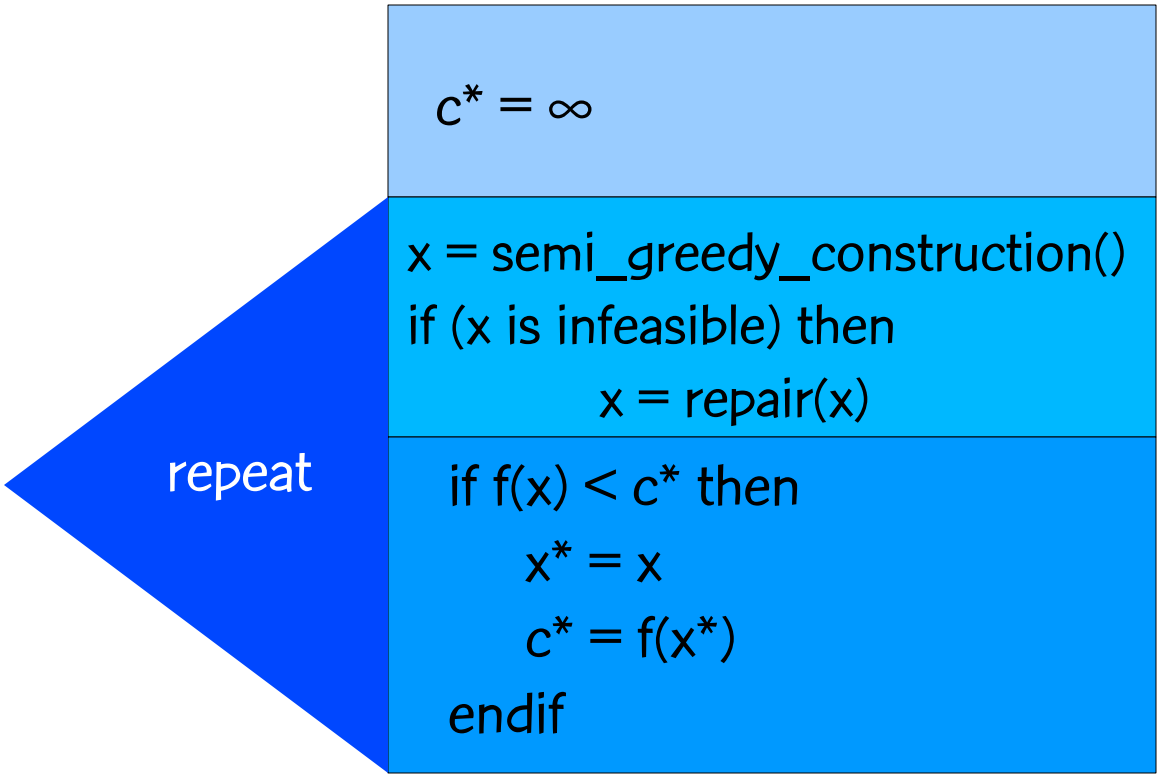
Hart & Shogan (1987) propose two mechanisms for building the RCL:

Cardinality based: place k best candidates in RCL

Value based: place all candidates having greedy values better than $\alpha \cdot$best_value in RCL, where $\alpha \in [0,1]$.

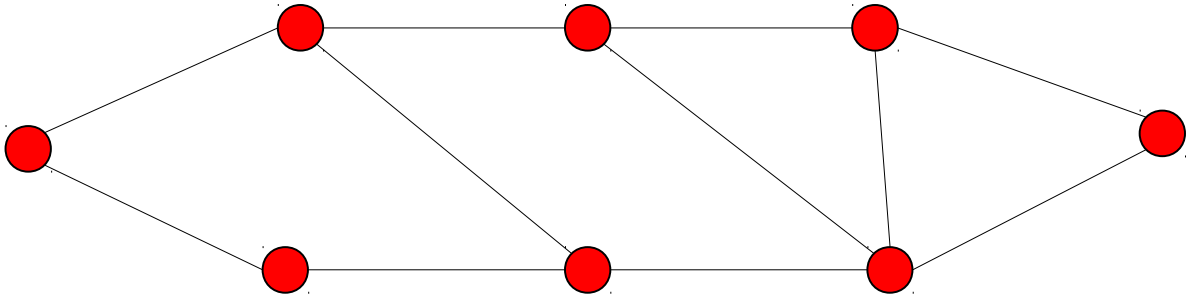
Feo & R. (1989) proposed semi-greedy construction as a basic component of GRASP.

Hart-Shogan Algorithm



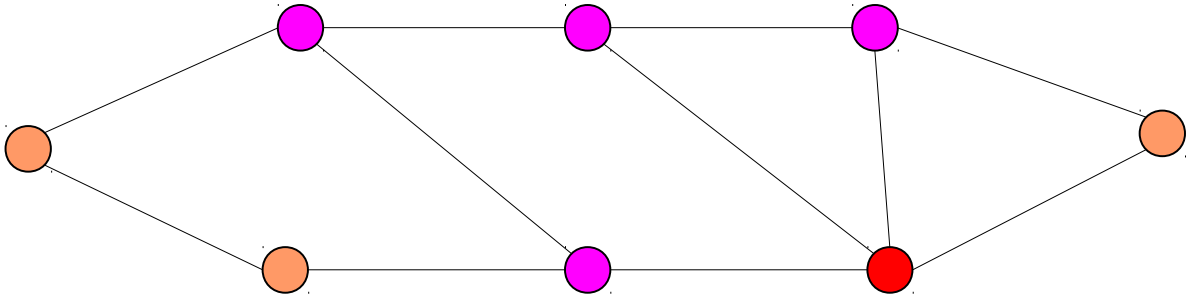
The semi-greedy algorithm

Maximum clique example



The semi-greedy algorithm

Maximum clique example

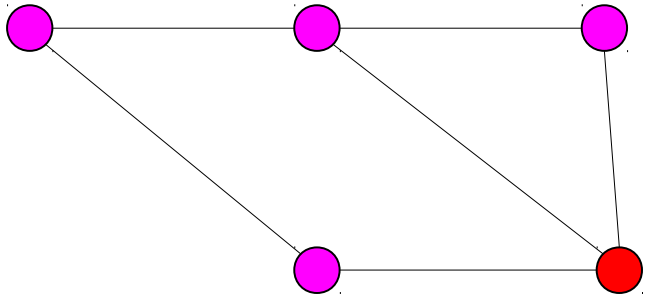


Build clique, one node at a time.

Candidates: nodes adjacent to clique.

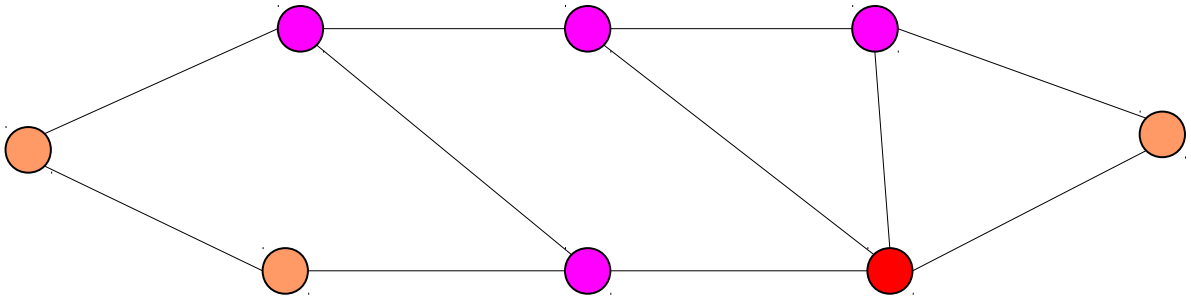
Greedy function: degree with respect to candidate nodes.

RCL =



The semi-greedy algorithm

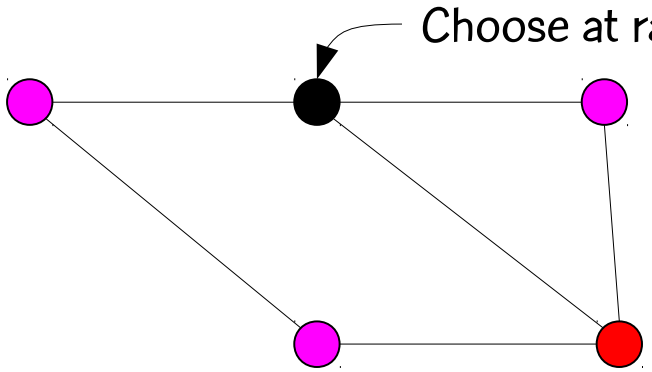
Maximum clique example



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

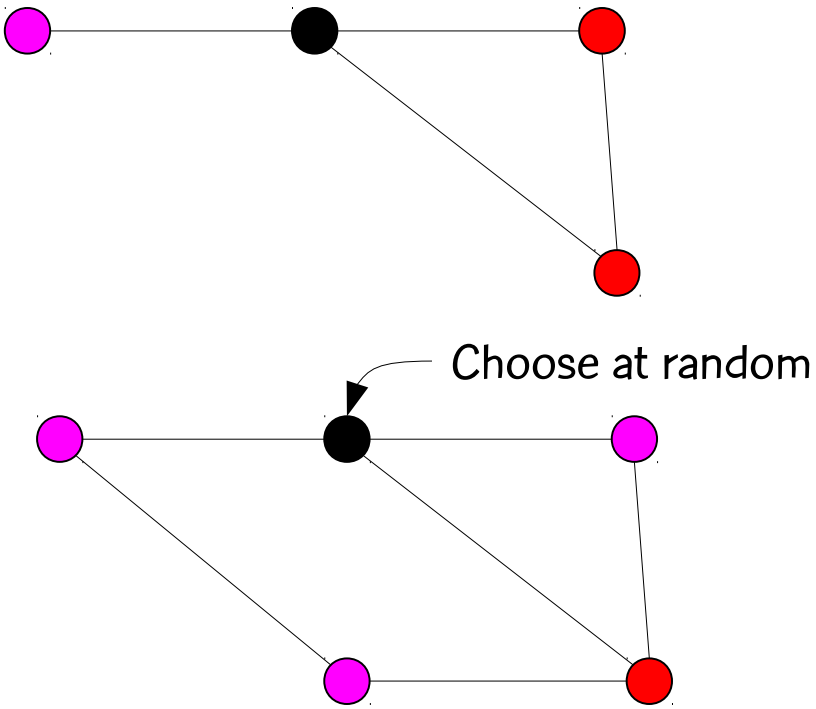


RCL =

Semi-greedy iteration 1

The semi-greedy algorithm

Maximum clique example



RCL =

Build clique, one node at a time.

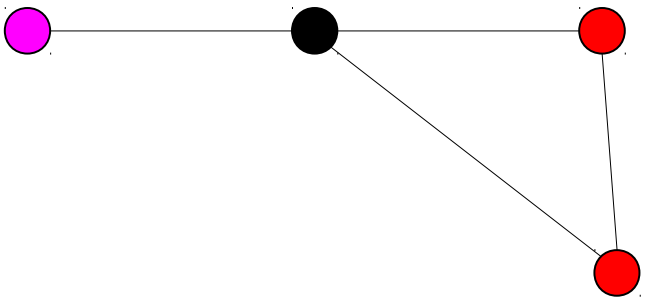
Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

Semi-greedy iteration 1

The semi-greedy algorithm

Maximum clique example



Choose at random

RCL =

Build clique, one node at a time.

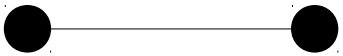
Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

Semi-greedy iteration 1

The semi-greedy algorithm

Maximum clique example



Clique of size 2

Build clique, one node at a time.

Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

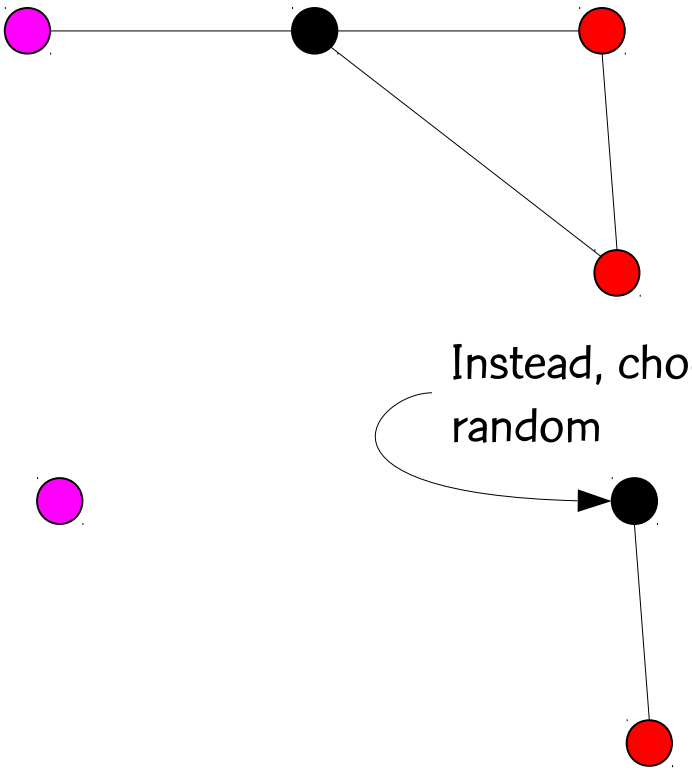
Choose at random

RCL =

Semi-greedy iteration 1

The semi-greedy algorithm

Maximum clique example



RCL =

Build clique, one node at a time.

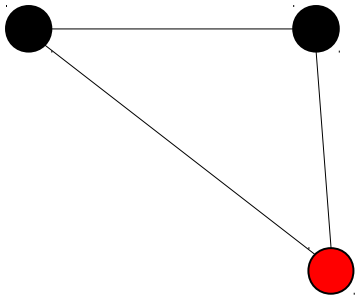
Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

Semi-greedy iteration 2

The semi-greedy algorithm

Maximum clique example



Instead, choose at random

RCL =

Build clique, one node at a time.

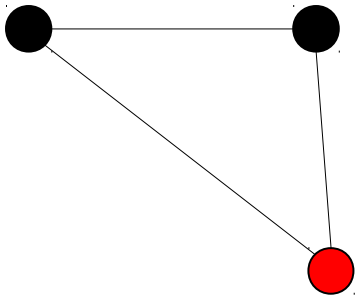
Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

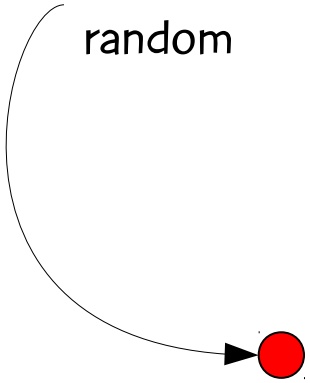
Semi-greedy iteration 2

The semi-greedy algorithm

Maximum clique example



Then, choose at random



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

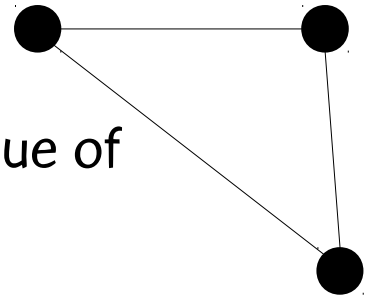
Greedy function: degree with respect to candidate nodes.

RCL =

Semi-greedy iteration 2

The semi-greedy algorithm

Maximum clique example



Optimal clique of size 3

Then, choose at random

RCL =

Build clique, one node at a time.

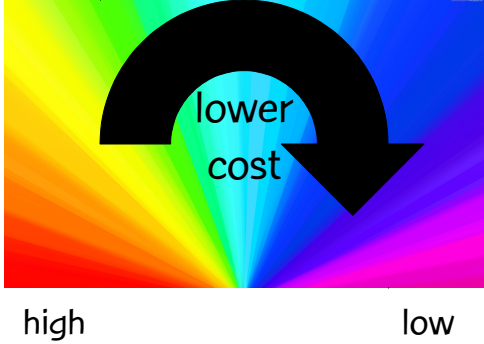
Candidates: nodes adjacent to clique.

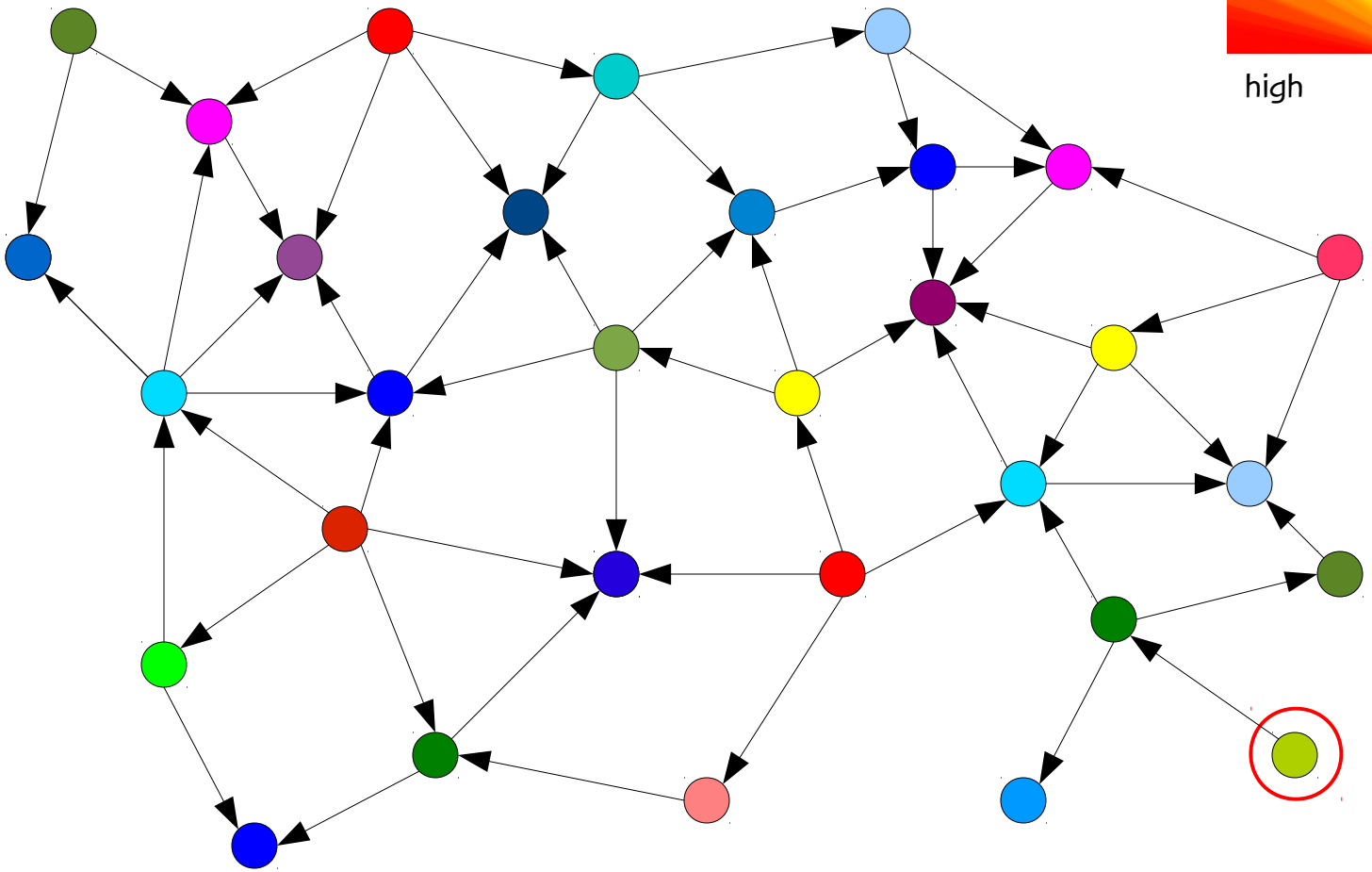
Greedy function: degree with respect to candidate nodes.

Semi-greedy iteration 2

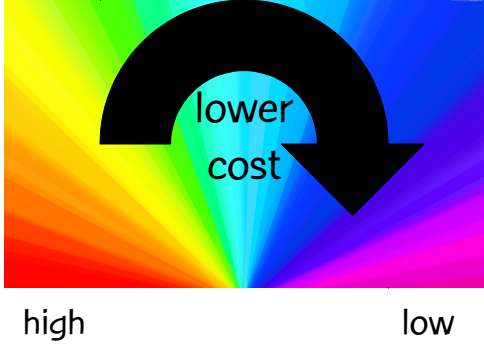
GRASP

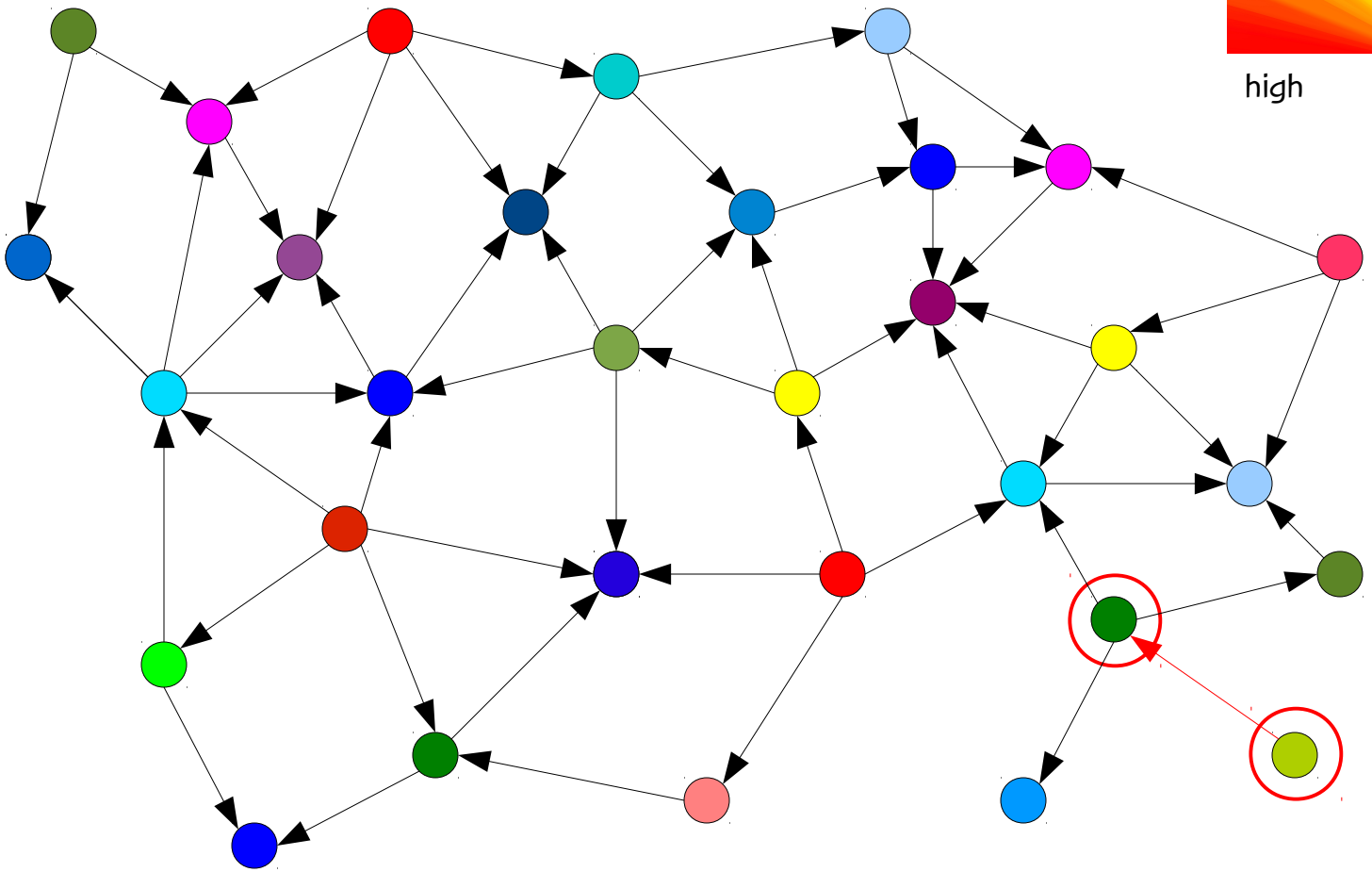
Local search: Start somewhere, improve, ...



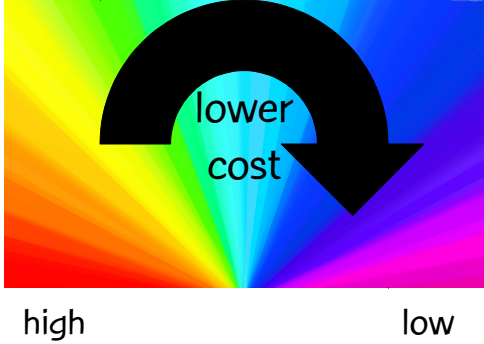


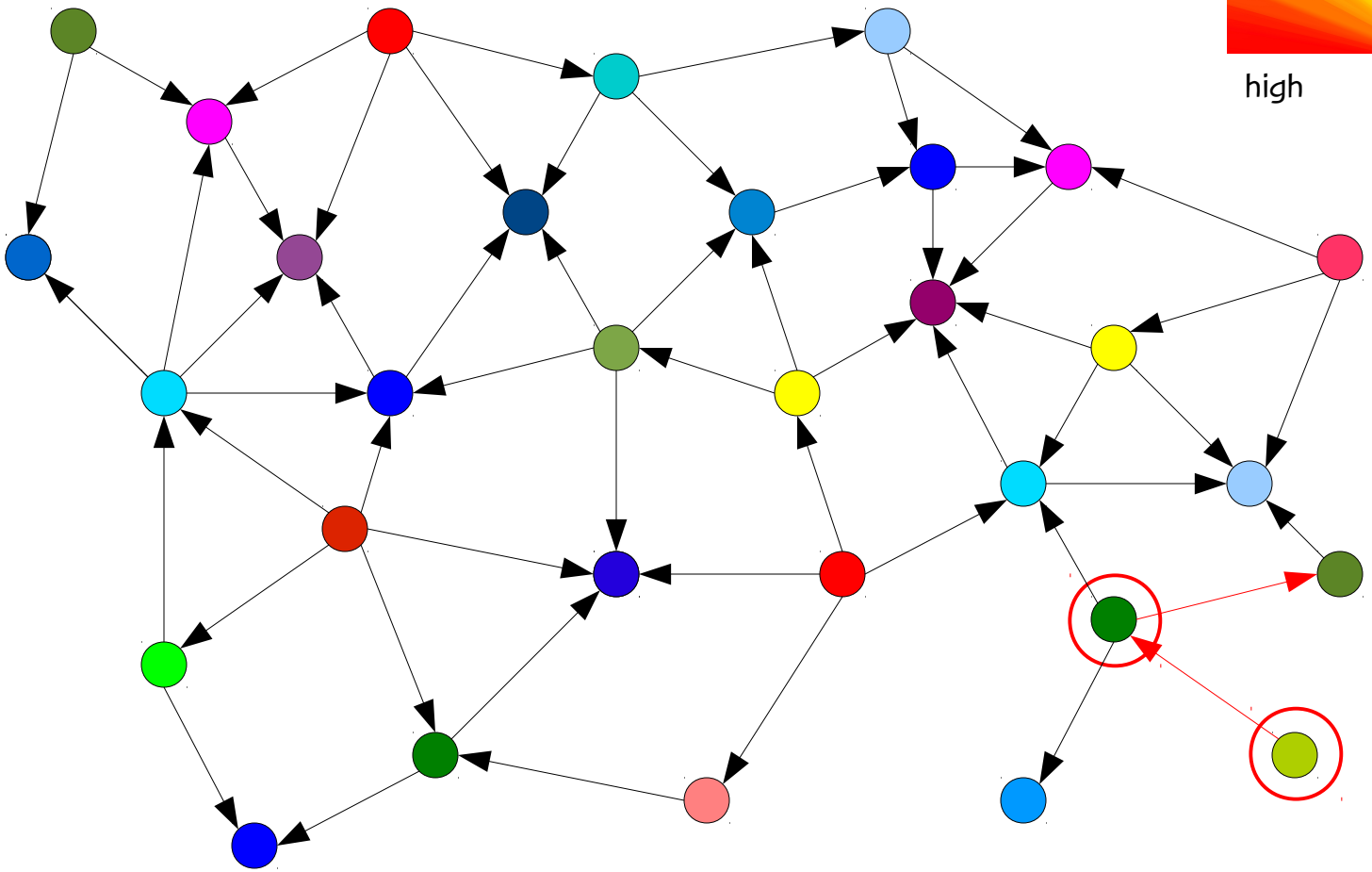
Local search: Start somewhere, improve, ...



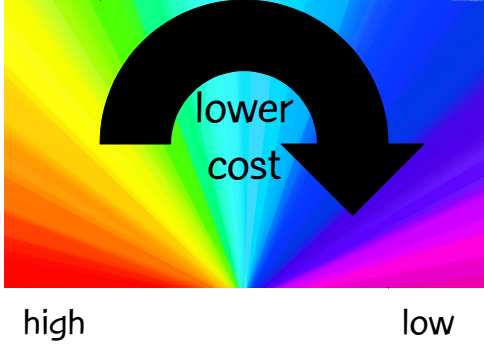


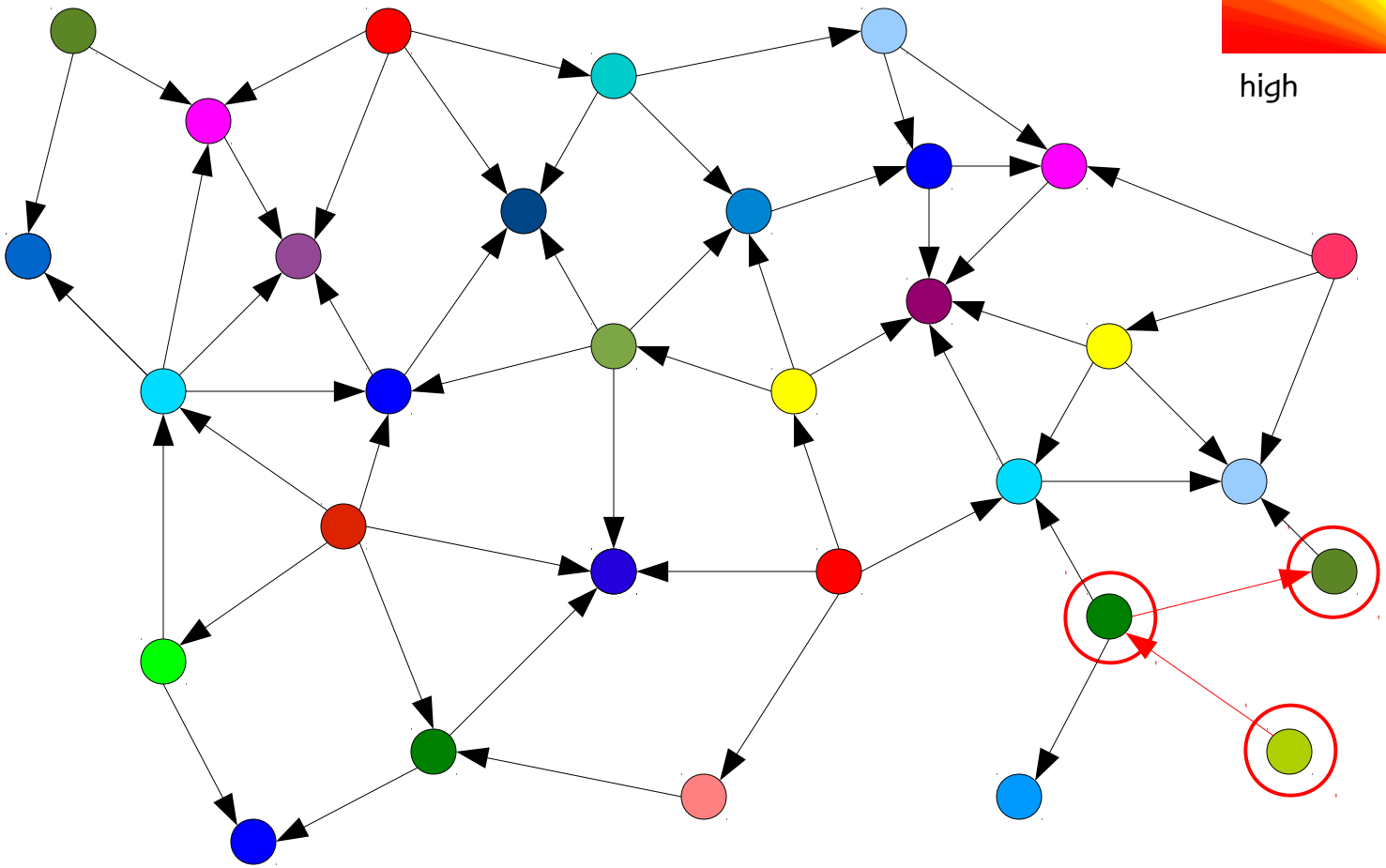
Local search: Start somewhere, improve, ...



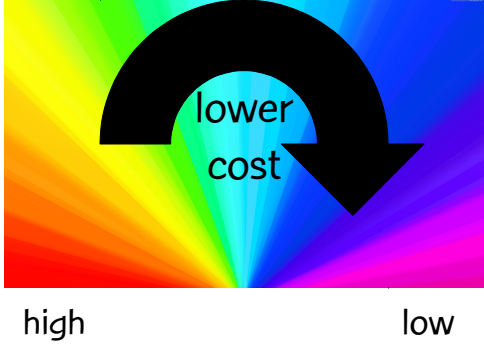


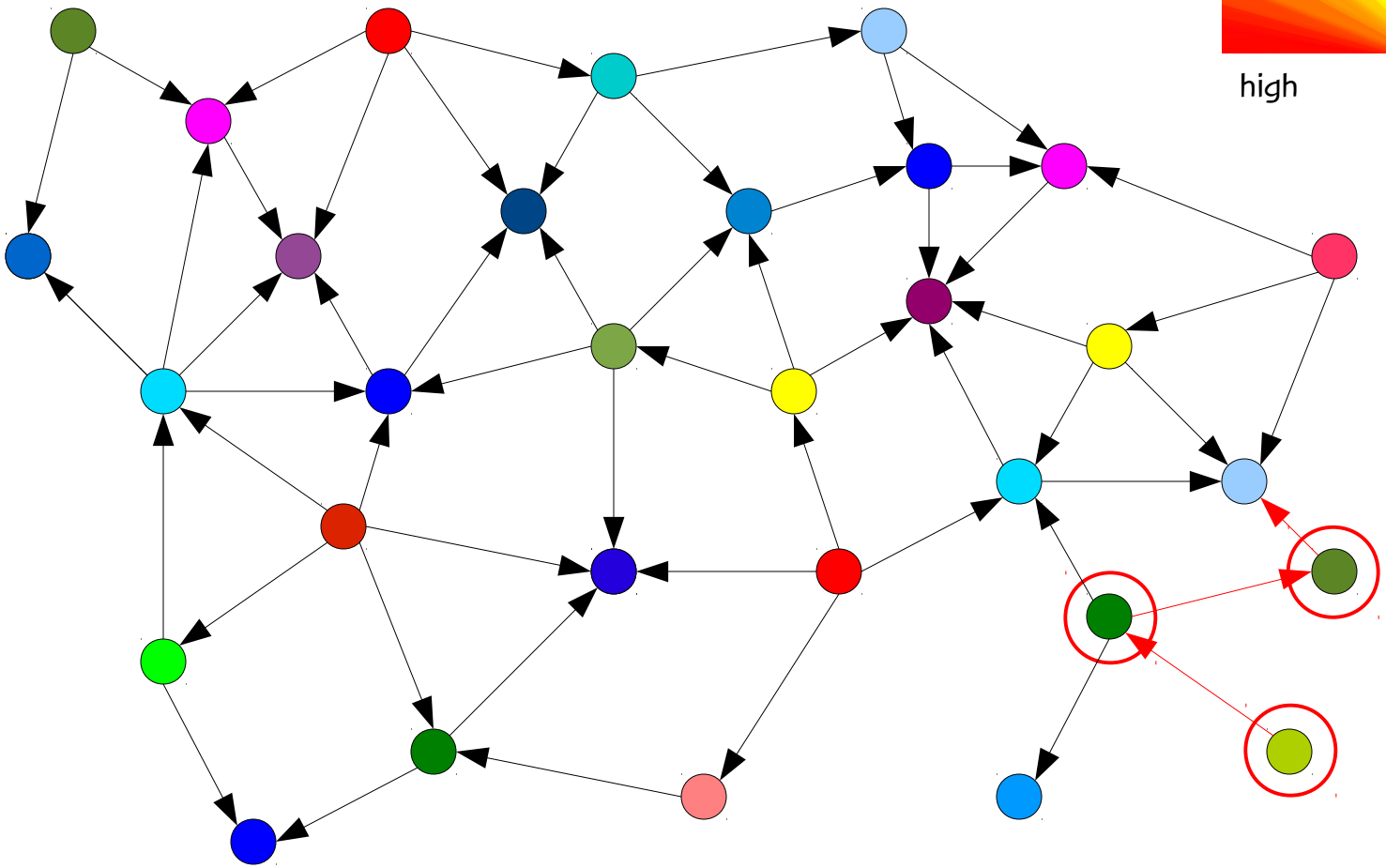
Local search: Start somewhere, improve, ...



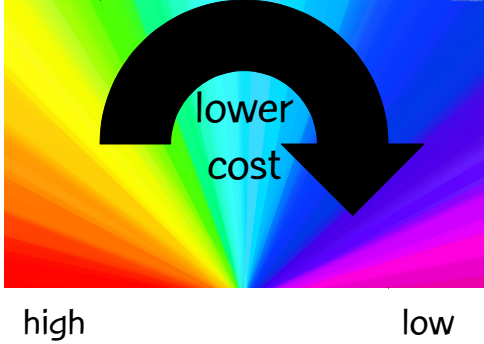


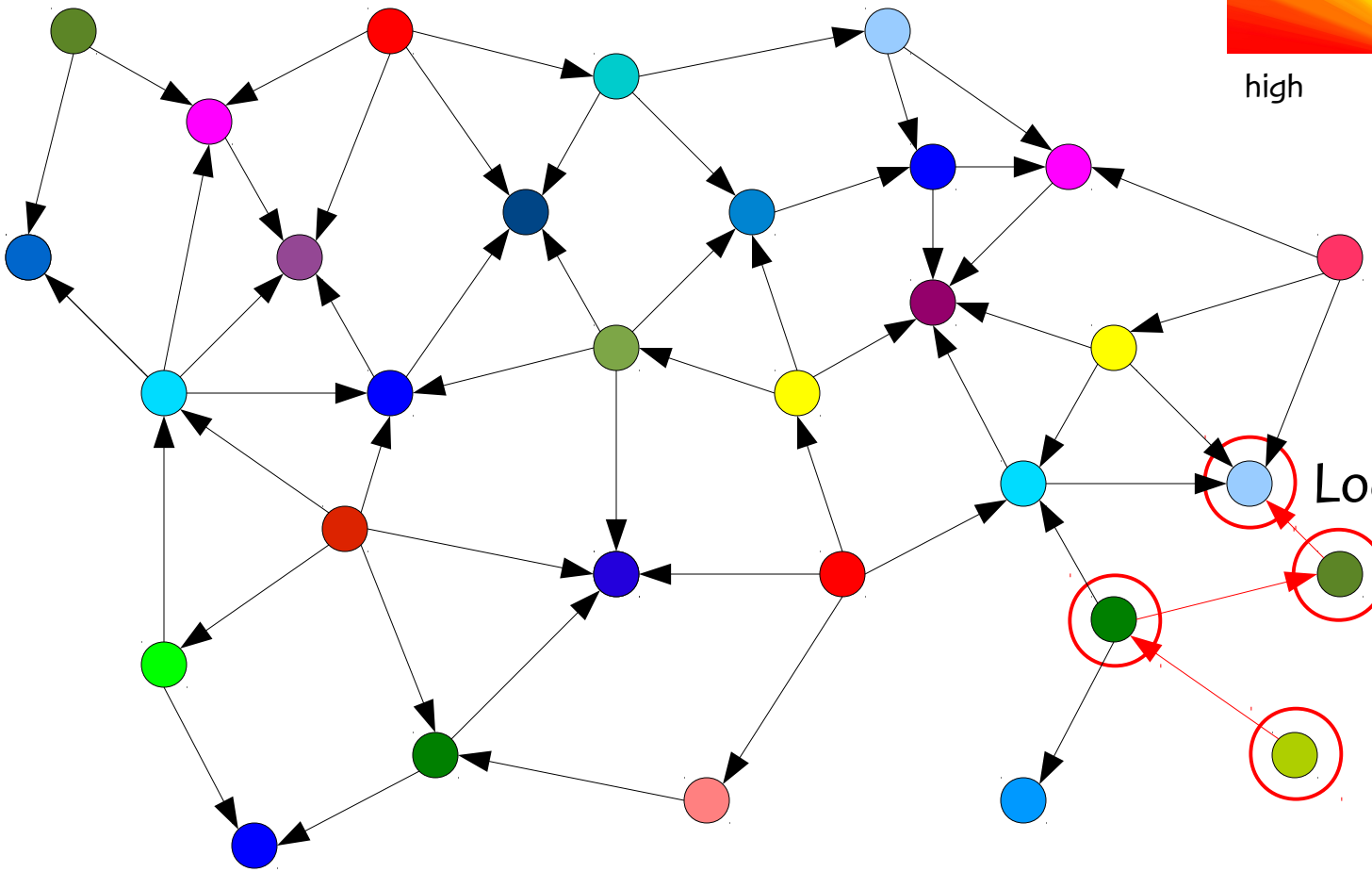
Local search: Start somewhere, improve, ...





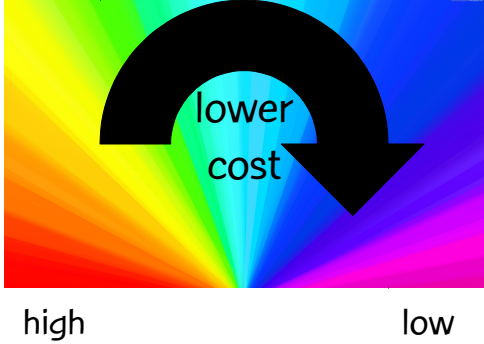
Local search: Start somewhere, improve, ...

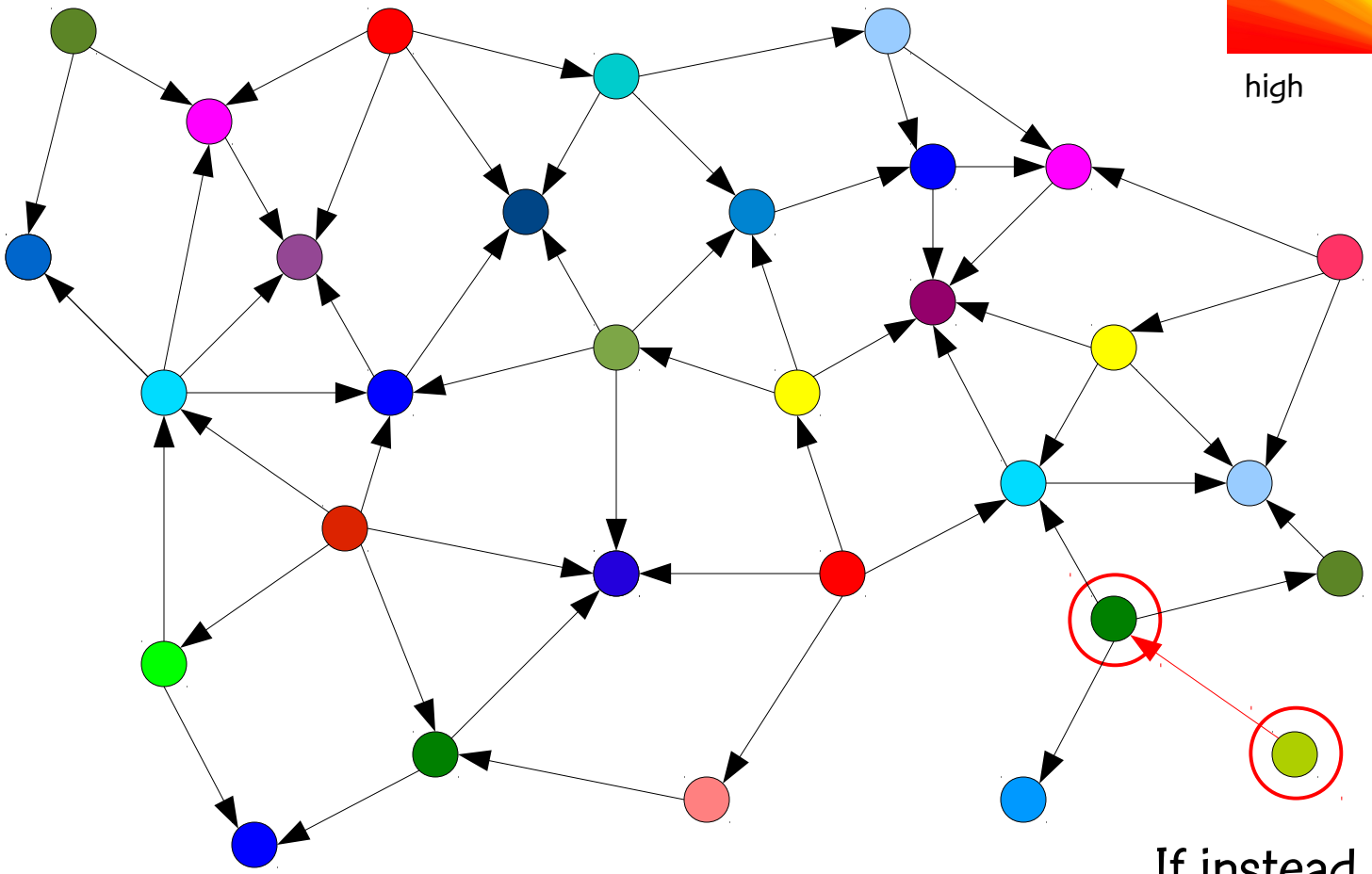




Local minimum
(non-global)

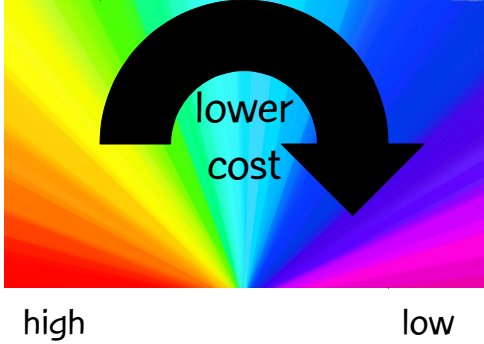
Local search: Start somewhere, improve, ...

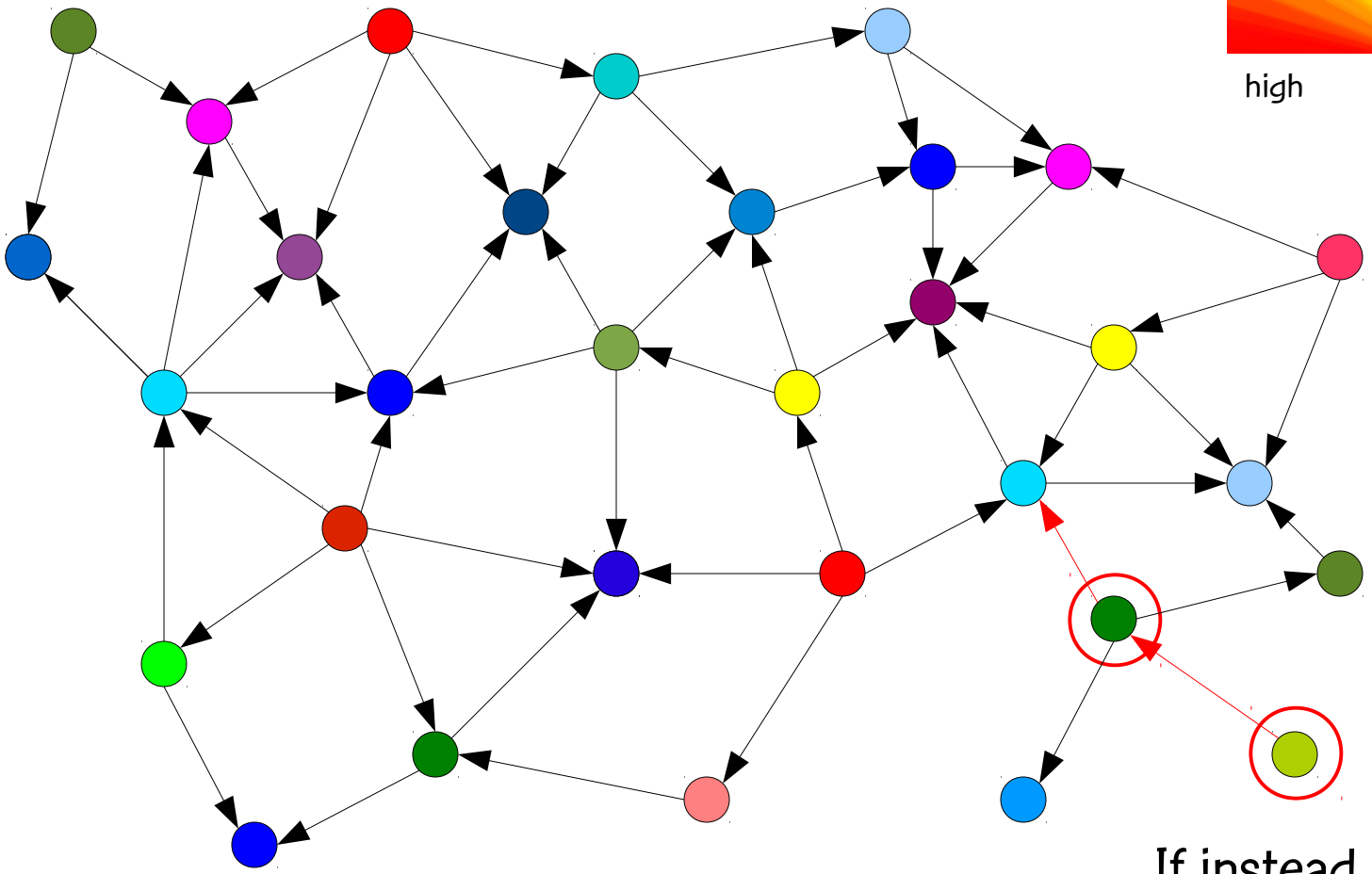




If instead, we took a different route

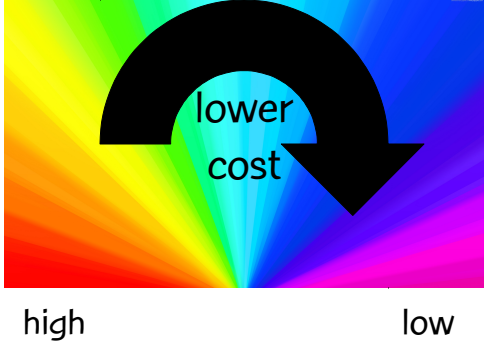
Local search: Start somewhere, improve, ...

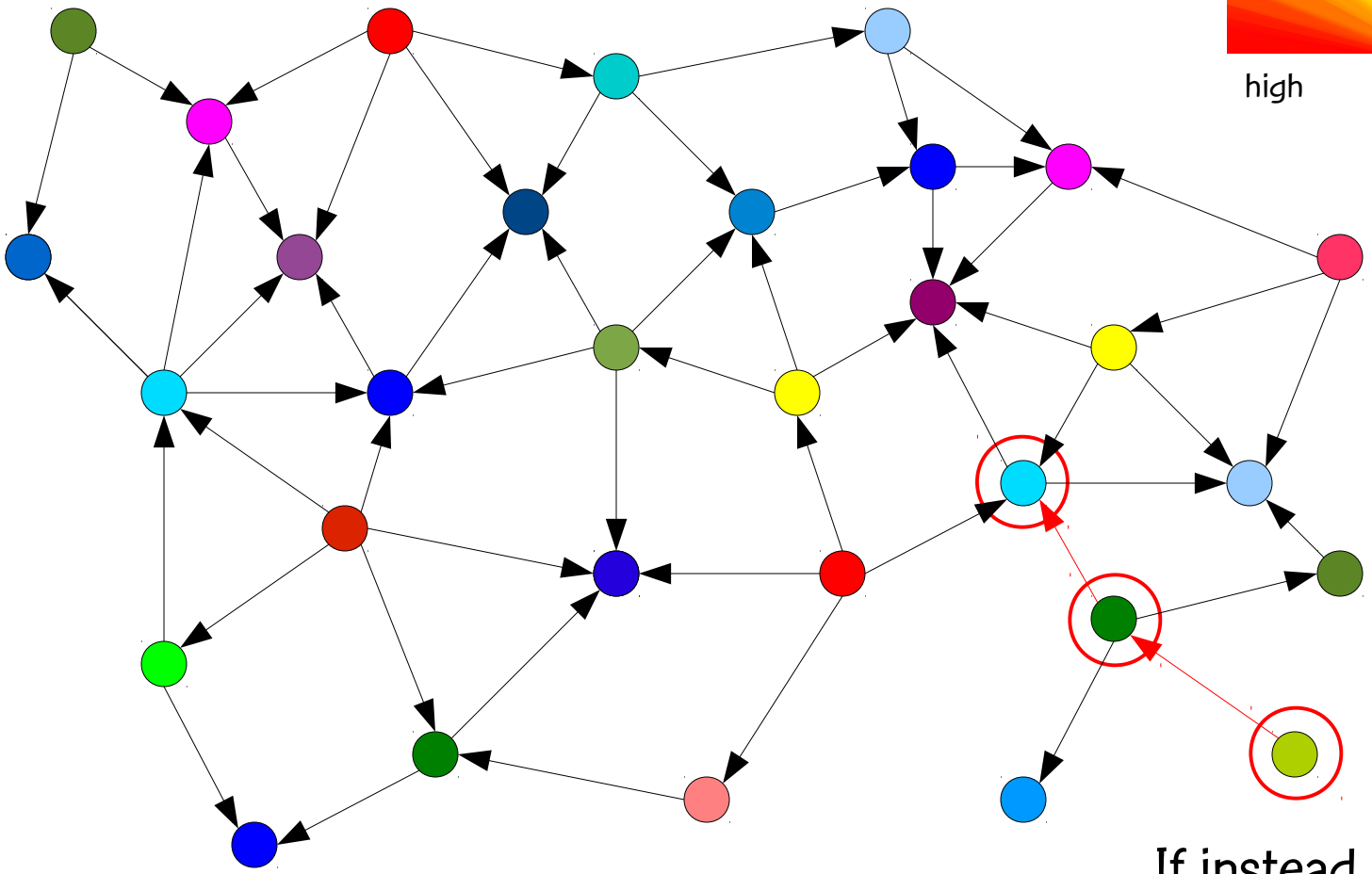




If instead, we took a different route

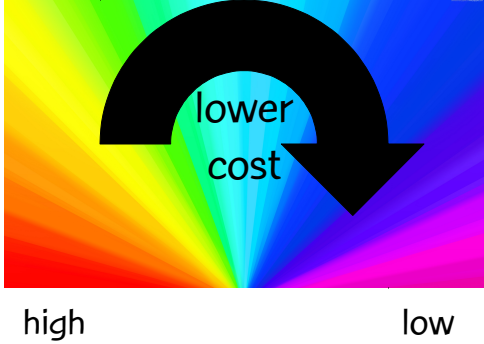
Local search: Start somewhere, improve, ...

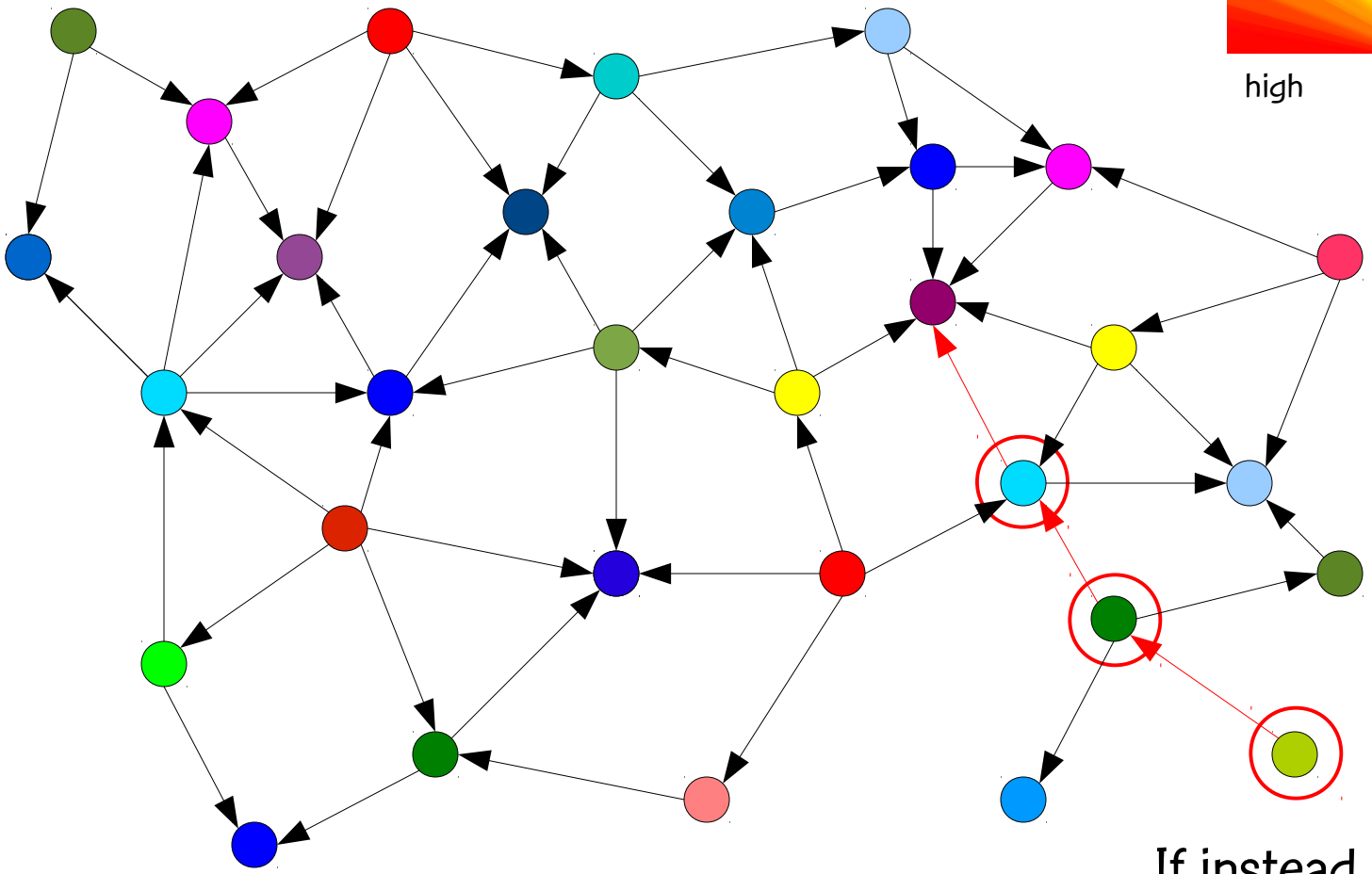




If instead, we took a different route

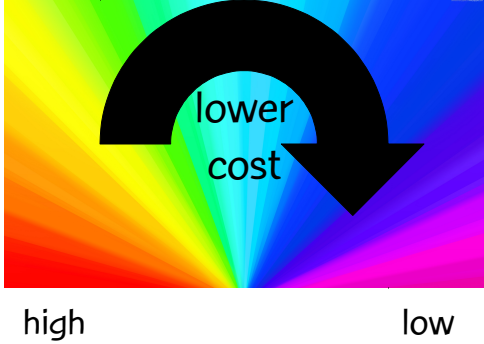
Local search: Start somewhere, improve, ...

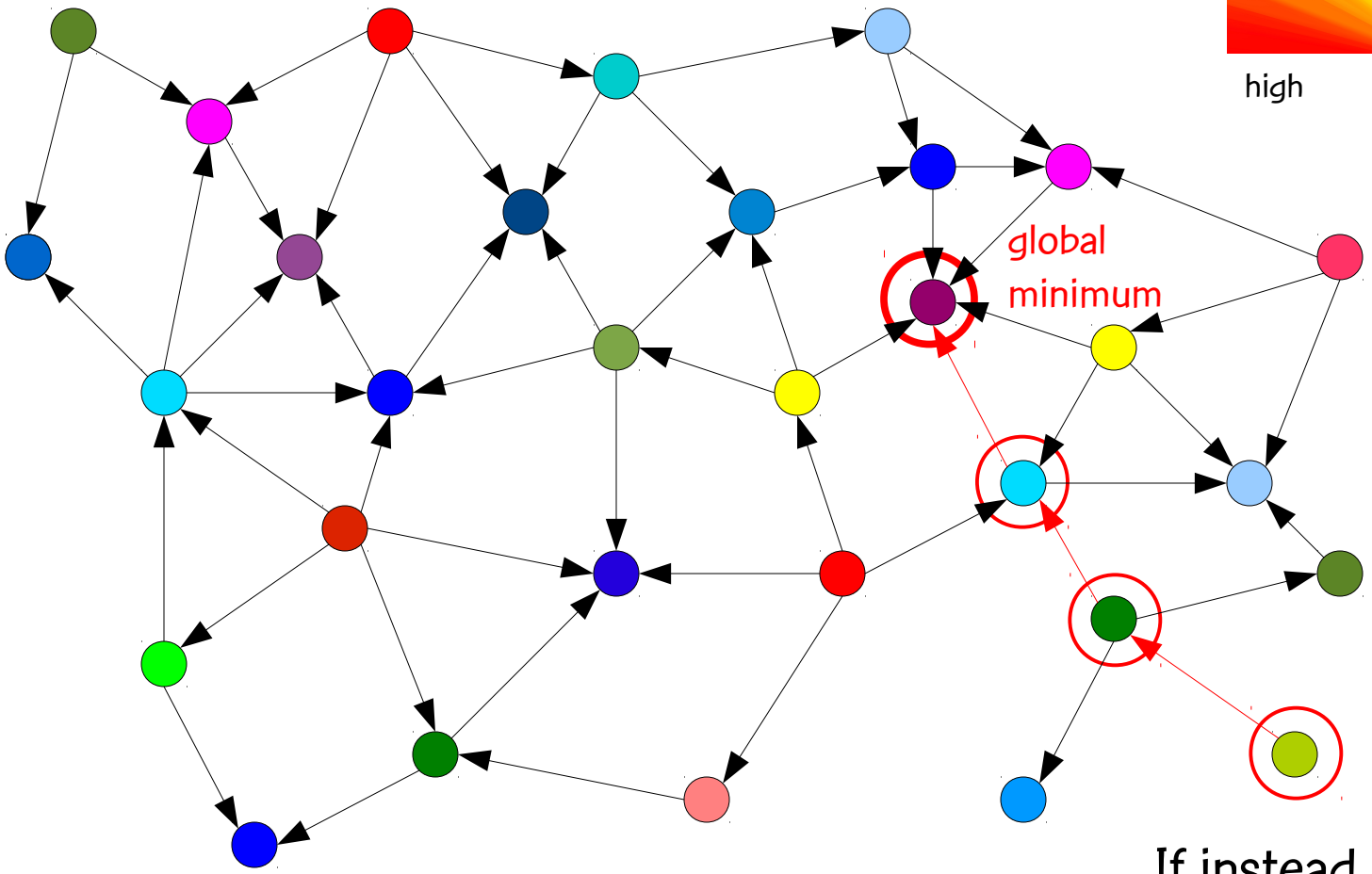




If instead, we took a different route

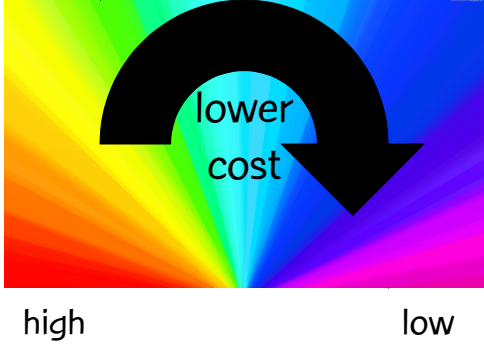
Local search: Start somewhere, improve, ...

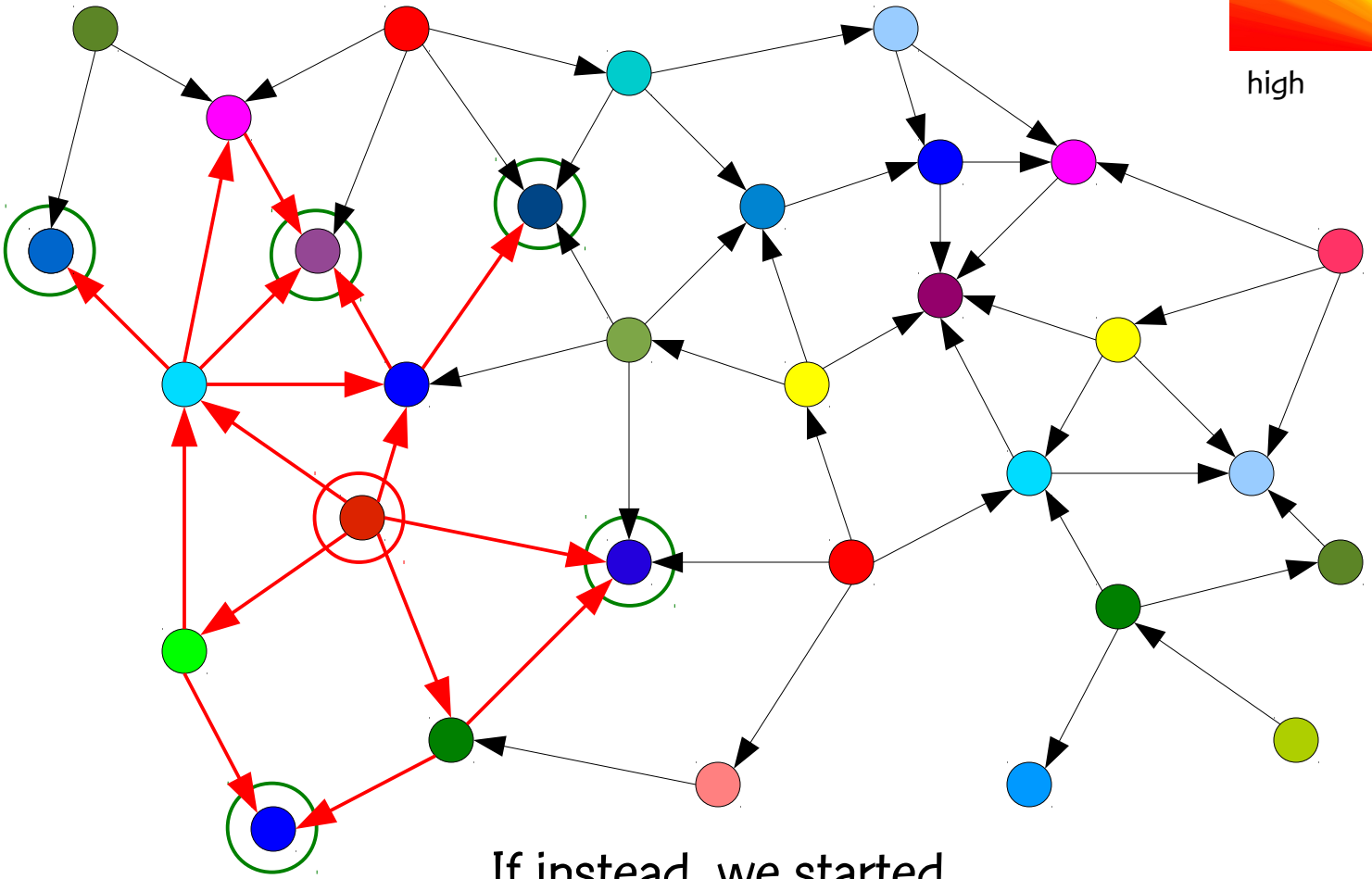




If instead, we took a different route

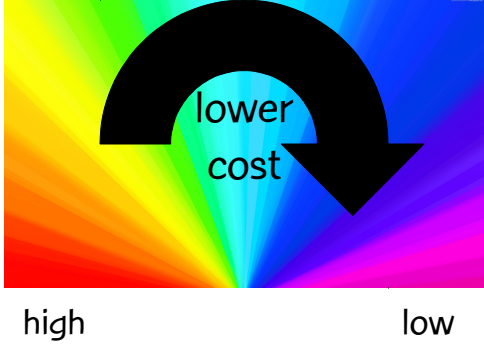
Local search: Start somewhere, improve, ...

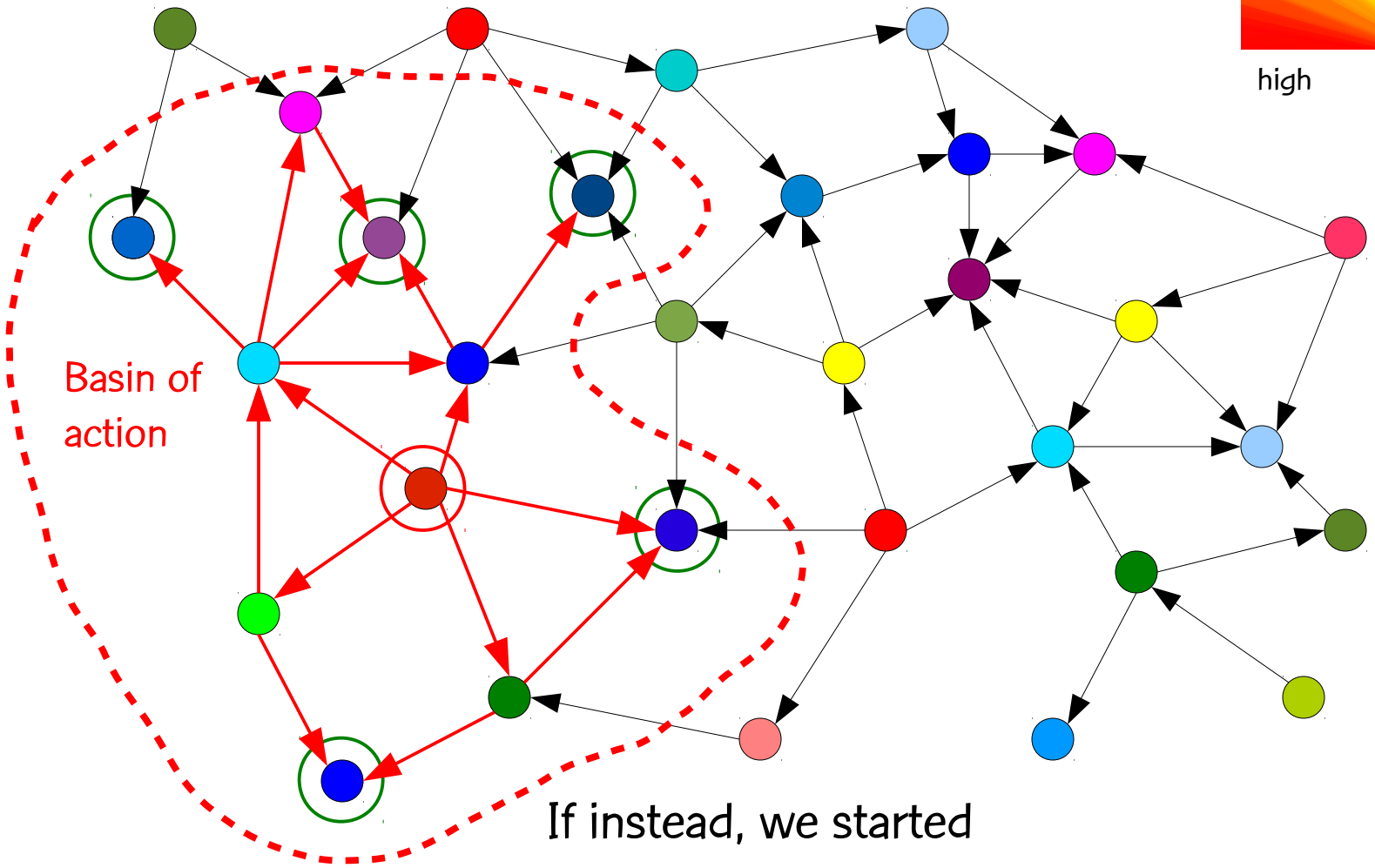




If instead, we started
at a different solution ...

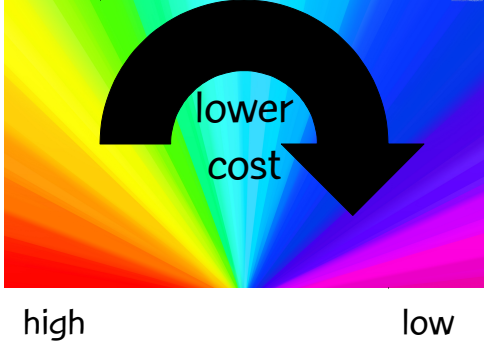
Local search: Start somewhere, improve, ...

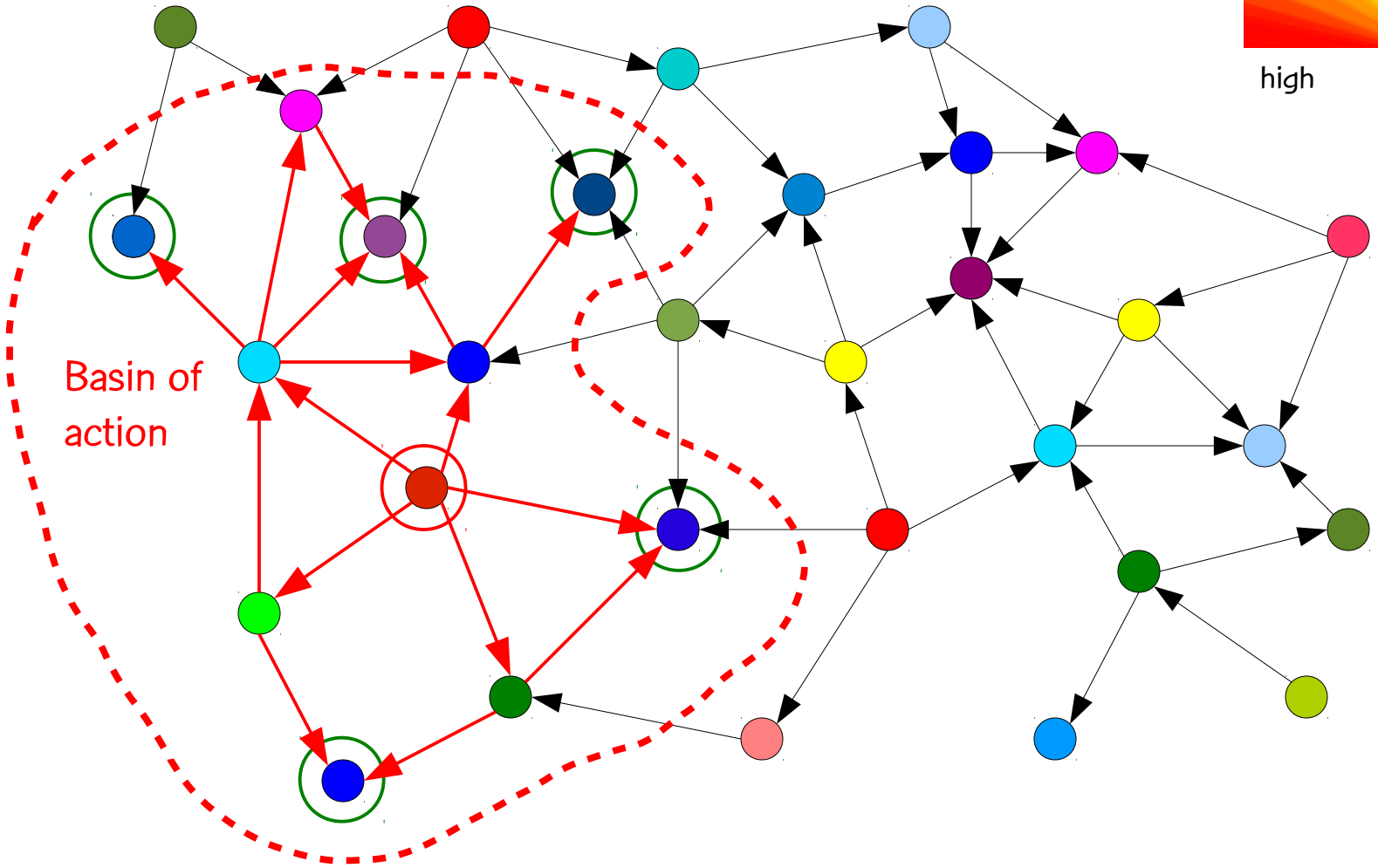




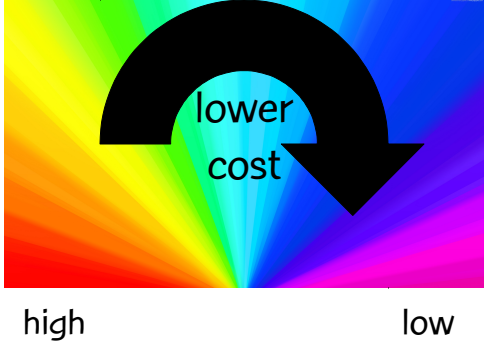
If instead, we started
at a different solution ... no path leads to global min

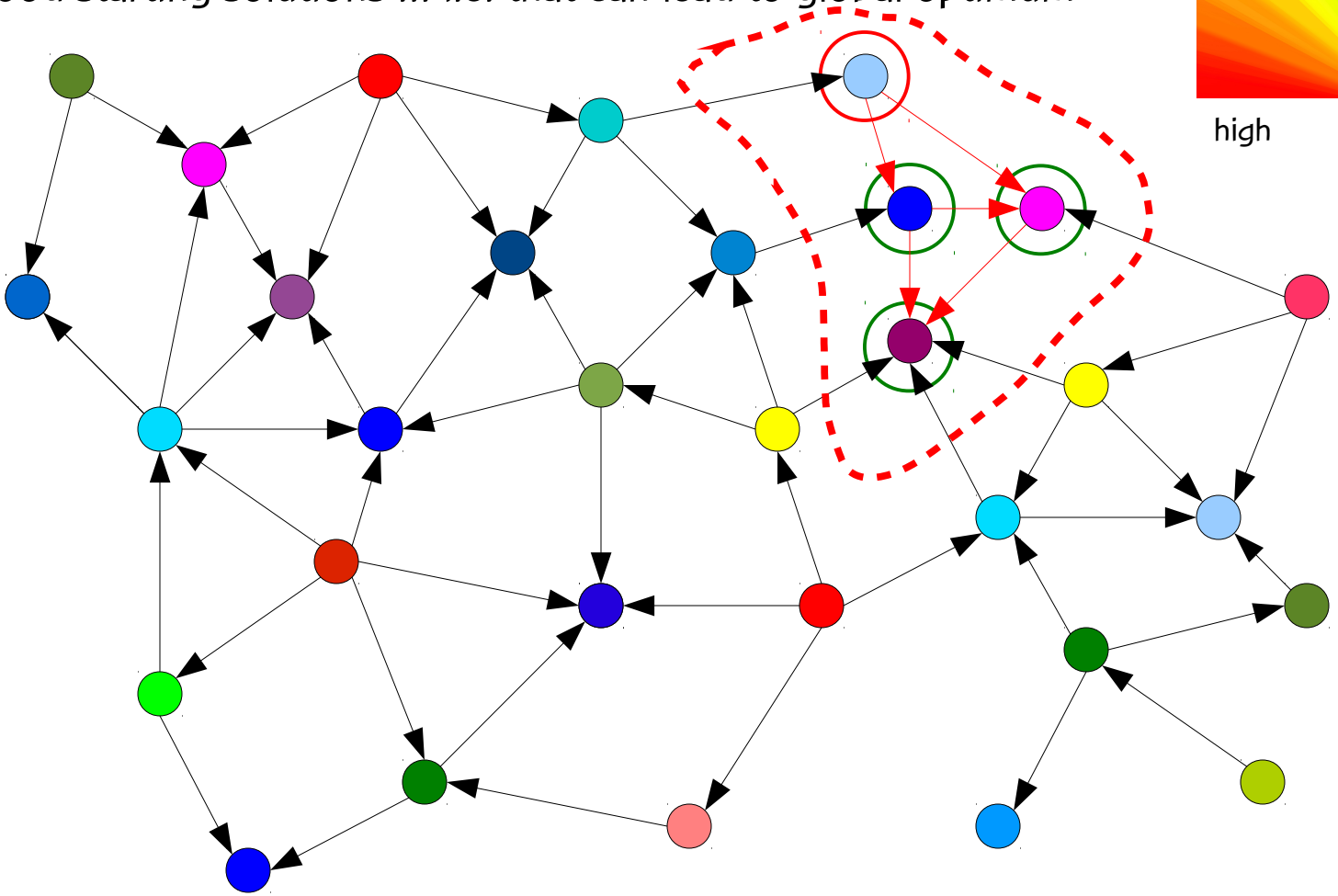
GRASP (Feo & R., 1989): Systematic procedure to generate good starting solutions ... i.e. that can lead to global optimum





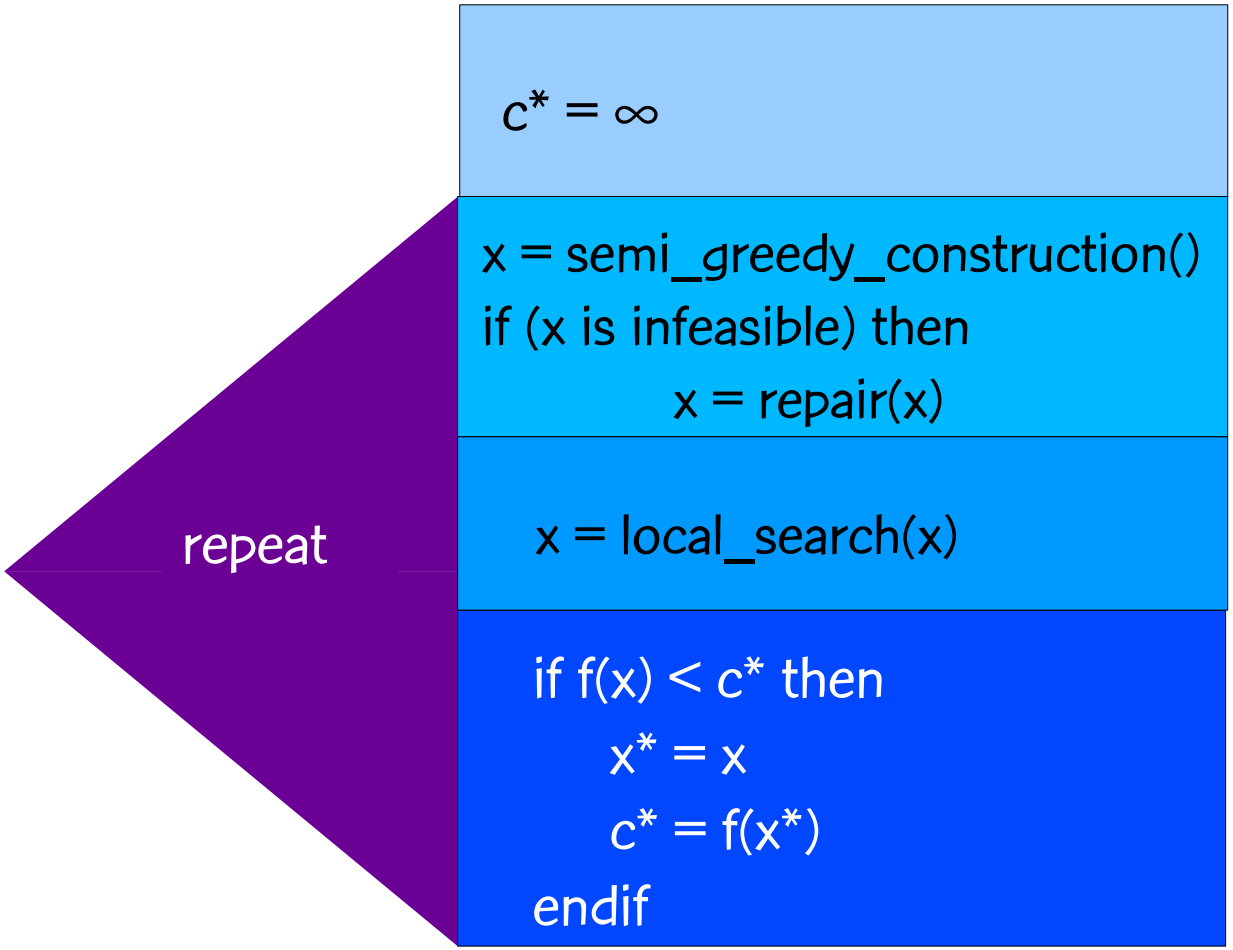
GRASP (Feo & R., 1989): Systematic procedure to generate good starting solutions ... i.e. that can lead to global optimum





GRASP blends greediness with randomness to generate starting solutions for local search

GRASP: Basic algorithm



Semi-greediness is more general in GRASP

GRASP: Basic algorithm

Construction phase: greediness + randomization

Builds a feasible solution combining greediness and randomization

Local search: search in the current neighborhood until a local optimum is found

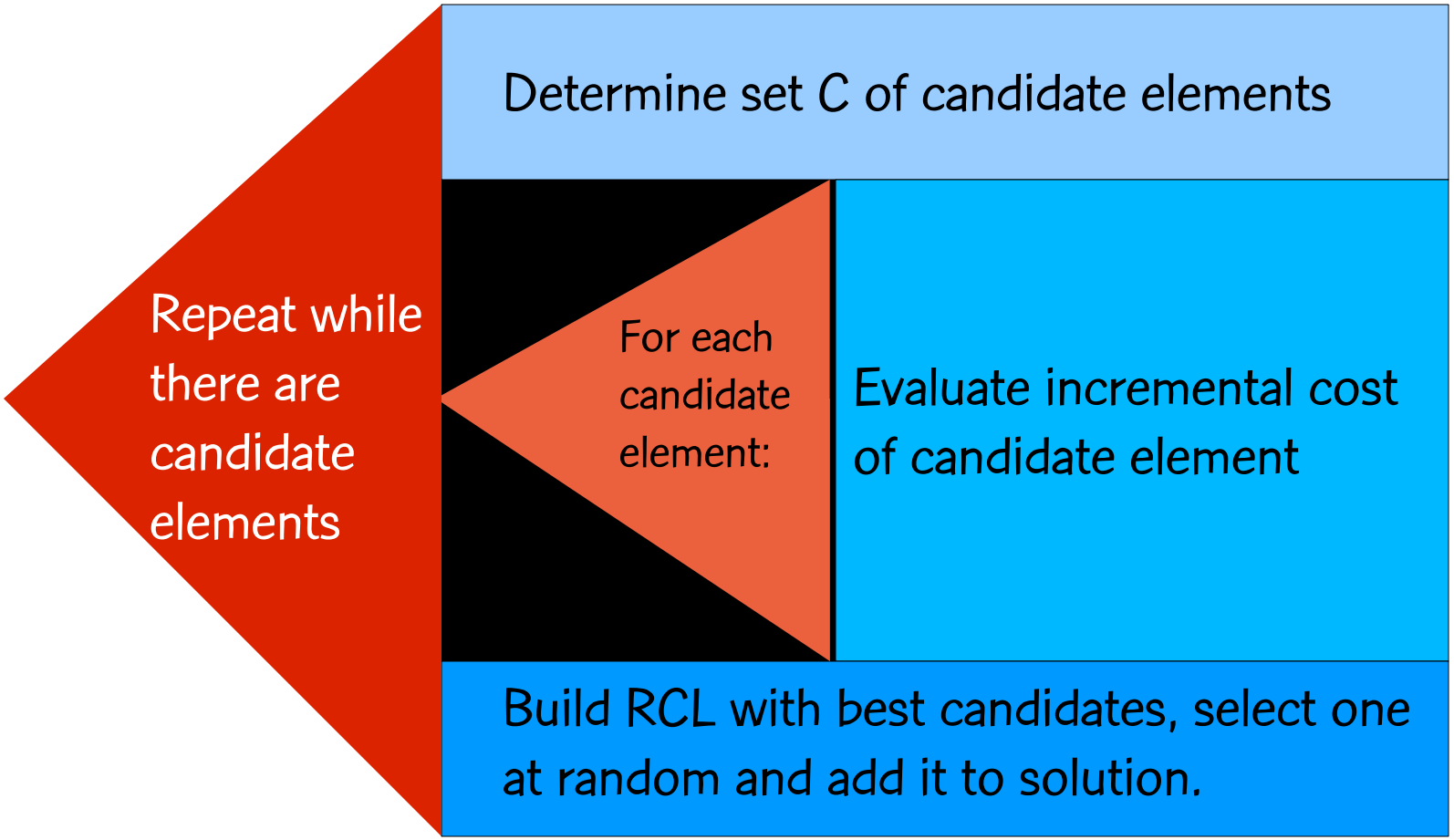
Solutions generated by the construction procedure are not necessarily optimal:

Effectiveness of local search depends on: neighborhood structure, search strategy, and fast evaluation of neighbors, but also on the construction procedure itself.

GRASP Construction

Construction phase: RCL based

restricted candidate list



Construction phase: RCL based

Minimization problem

Basic construction procedure:

Greedy function $c(e)$: incremental cost associated with the incorporation of element e into the current partial solution under construction

c^{min} (resp. c^{max}): smallest (resp. largest) incremental cost

RCL made up by the elements with the smallest incremental costs.

Construction phase

Cardinality-based construction:

p elements with the smallest incremental costs

Quality-based construction:

Parameter α defines the quality of the elements in RCL.

RCL contains elements with incremental cost

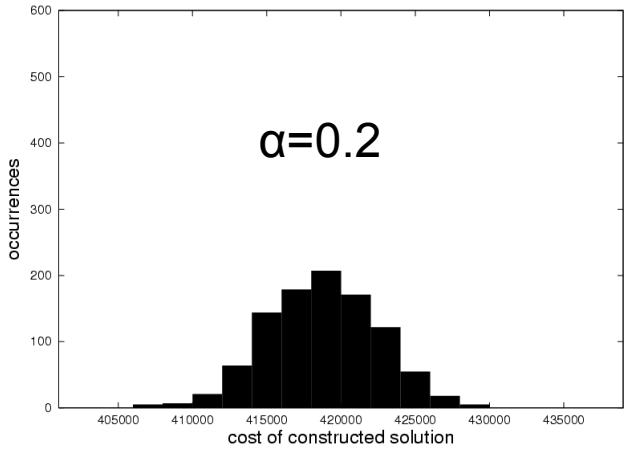
$c^{min} \leq c(e) \leq c^{min} + \alpha\,(c^{max} - c^{min})$

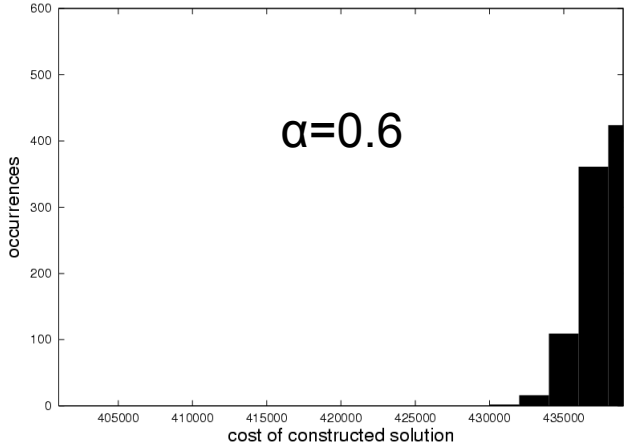
$\alpha = 0$: pure greedy construction

$\alpha = 1$: pure randomized construction

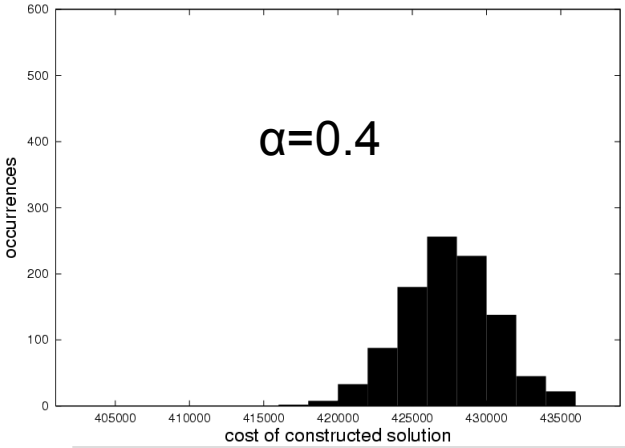
Select at random from RCL using uniform probability distribution

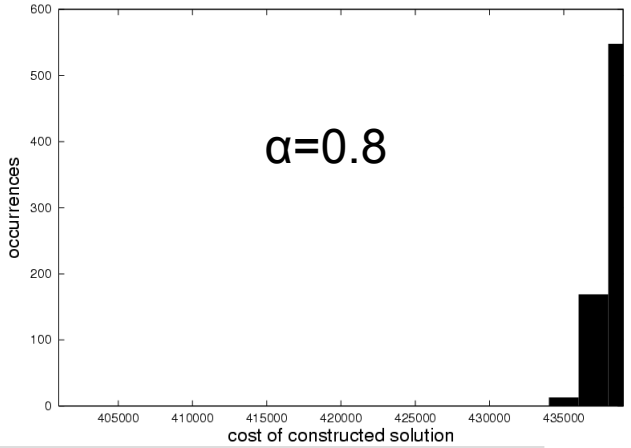
Illustrative results: RCL parameter





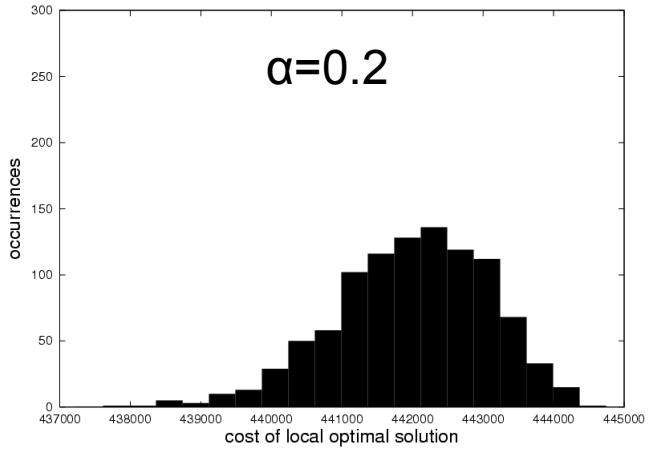
Construction phase only

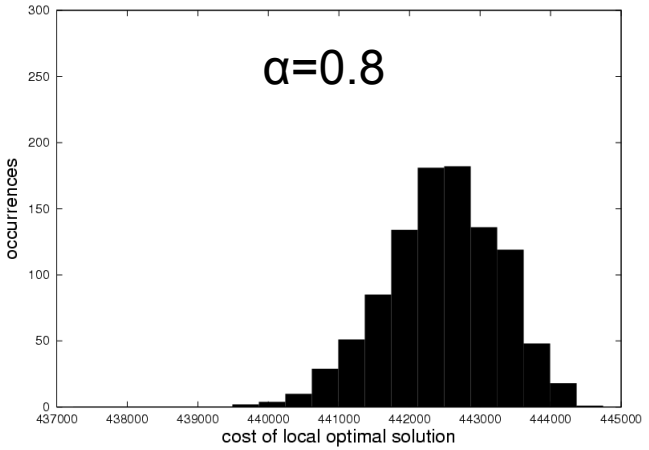




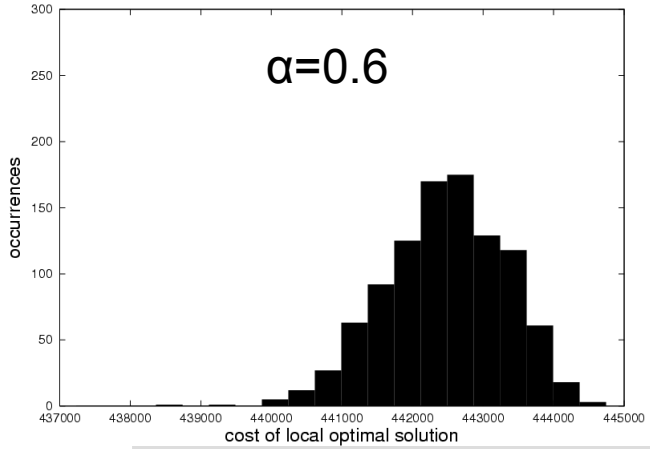
weighted MAX-SAT instance, 1000 GRASP iterations

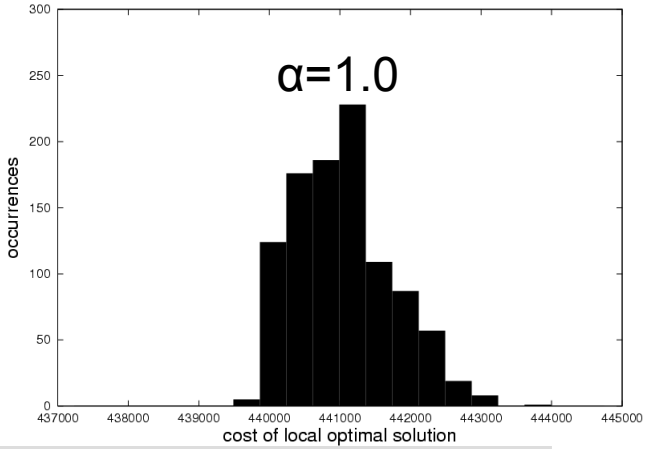
Illustrative results: RCL parameter





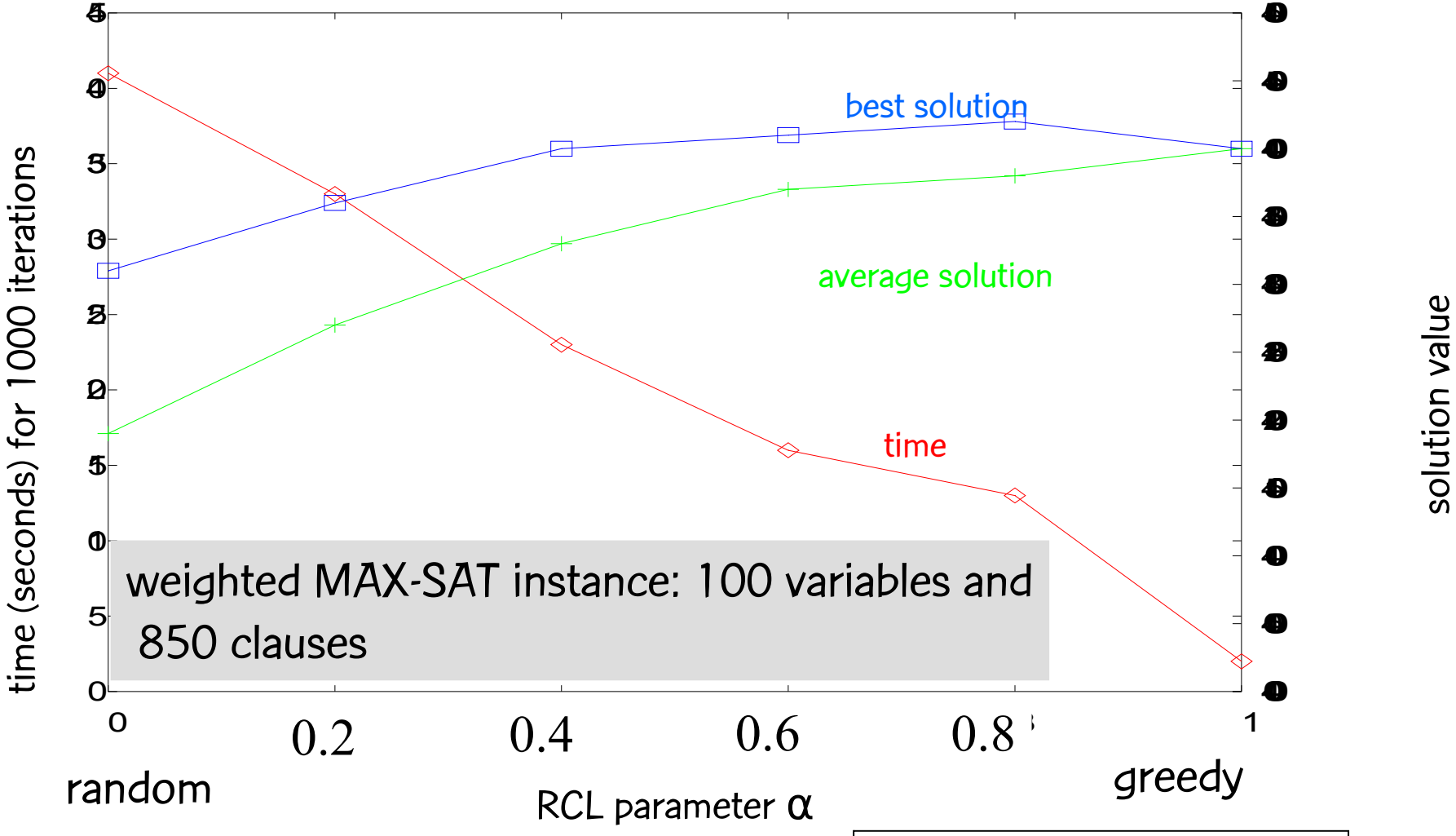
Construction + local search





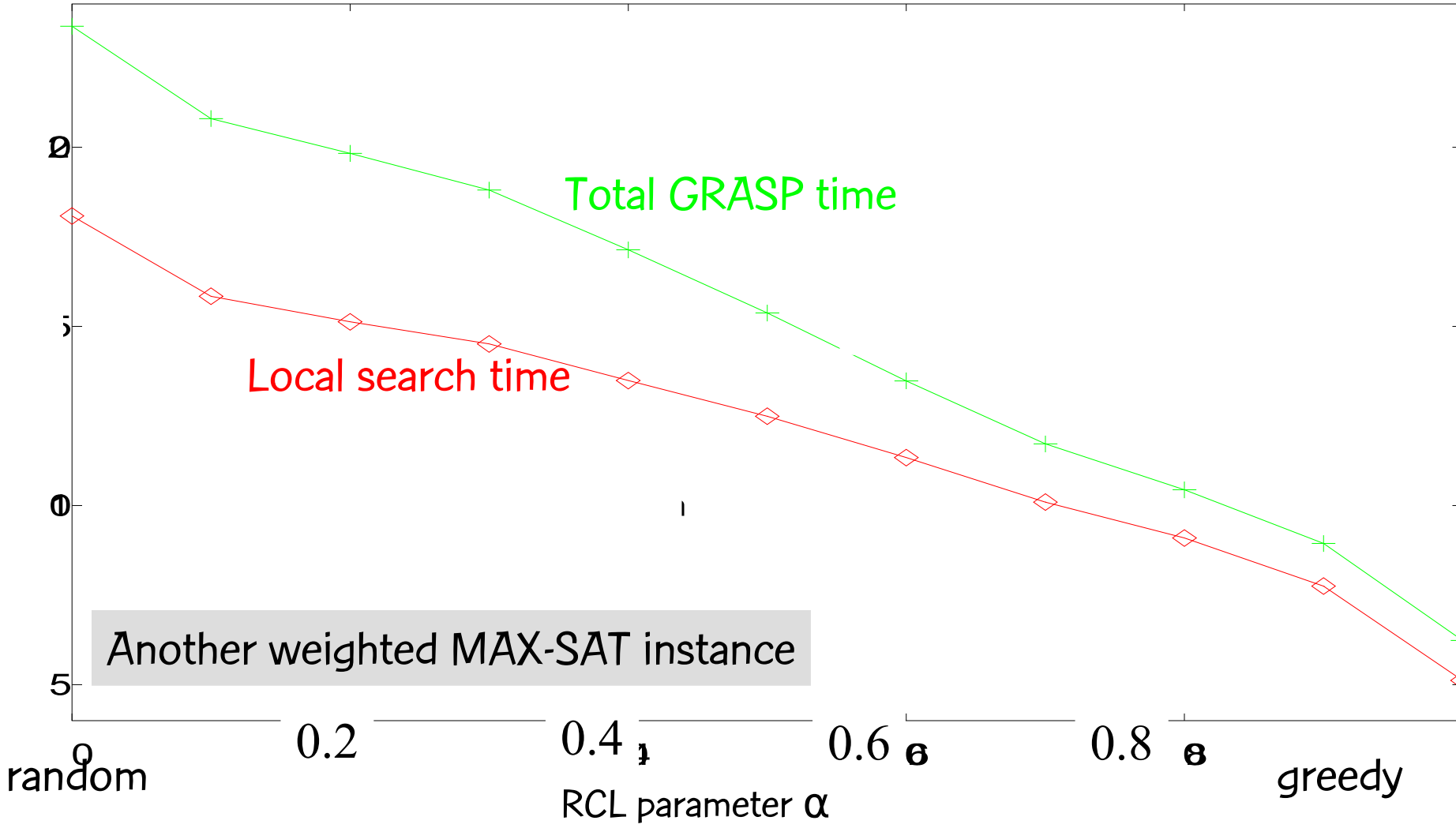
weighted MAX-SAT instance, 1000 GRASP iterations

Illustrative results: RCL parameter

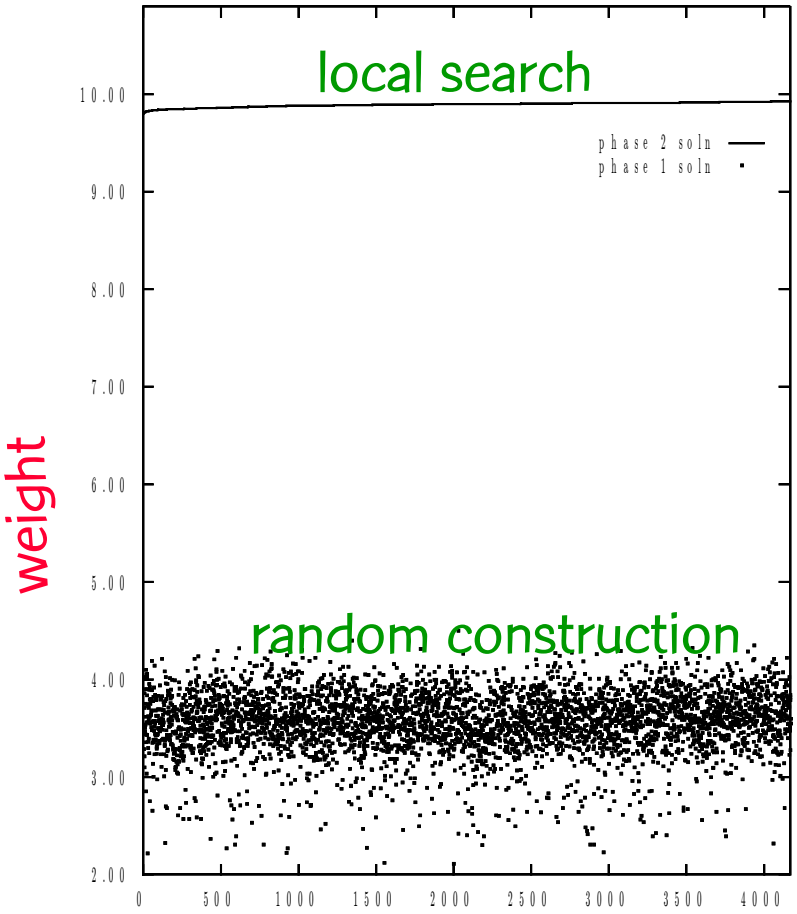


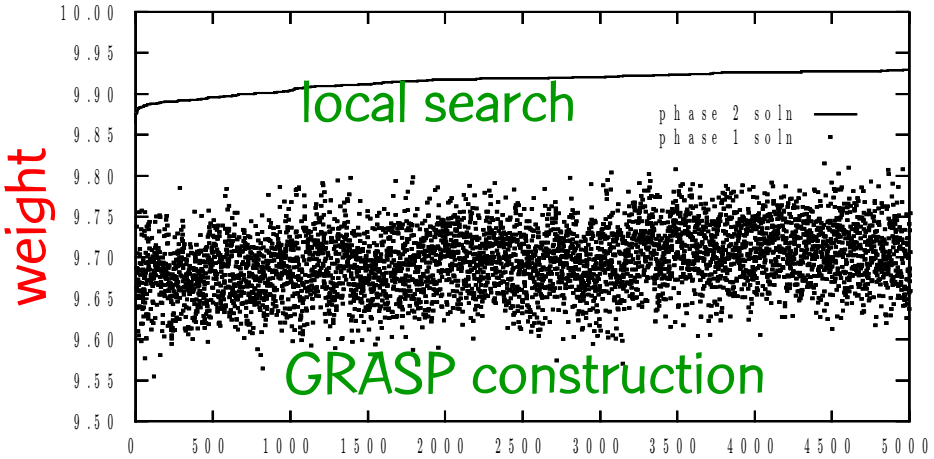
SGI Challenge 196 MHz

Illustrative results: RCL parameter



GRASP: Basic algorithm





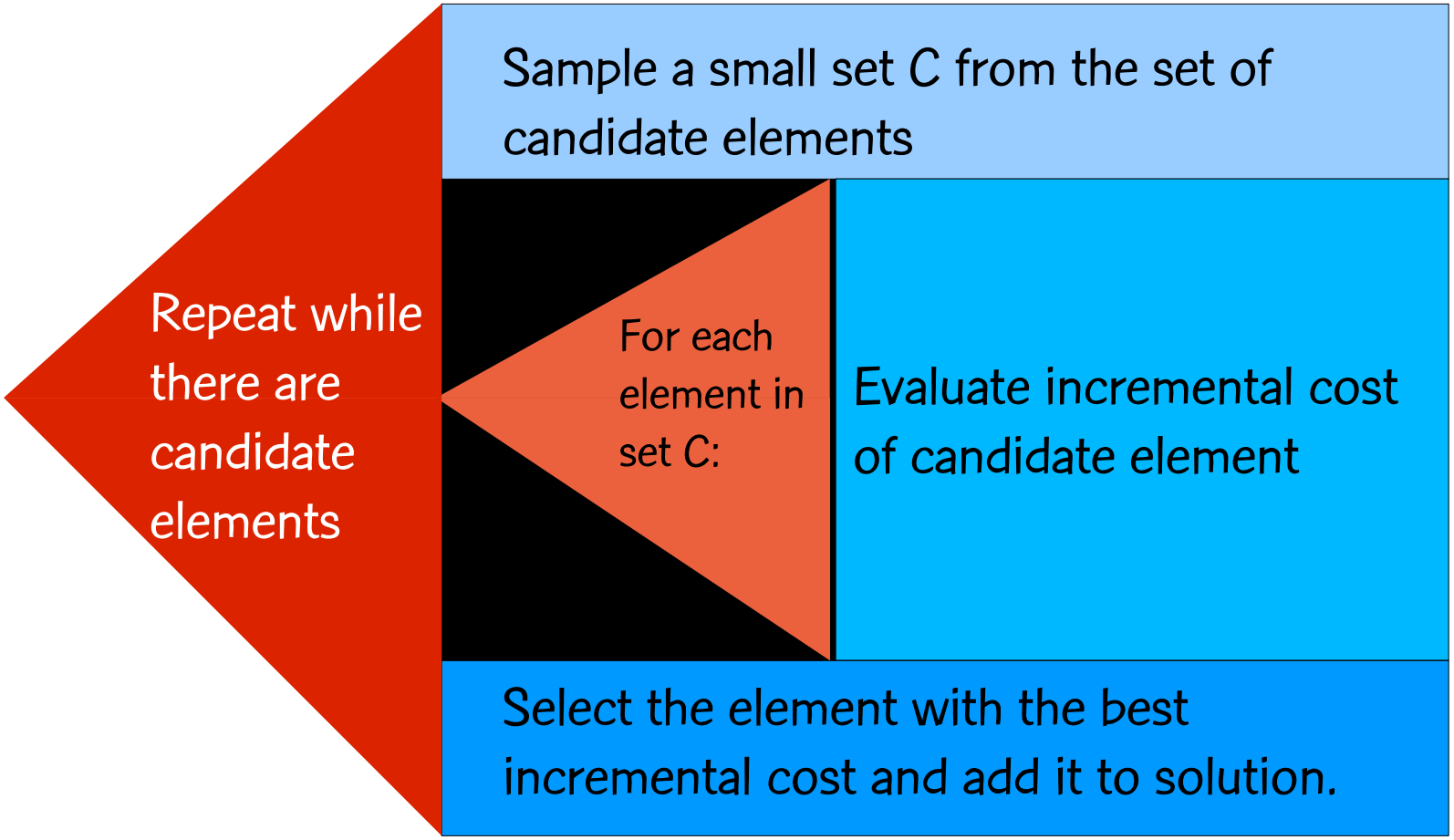
Effectiveness of greedy randomized vs purely randomized construction:

Application: modem placement
max weighted covering problem
maximization problem: $\alpha = 0.85$

Hybrid construction schemes

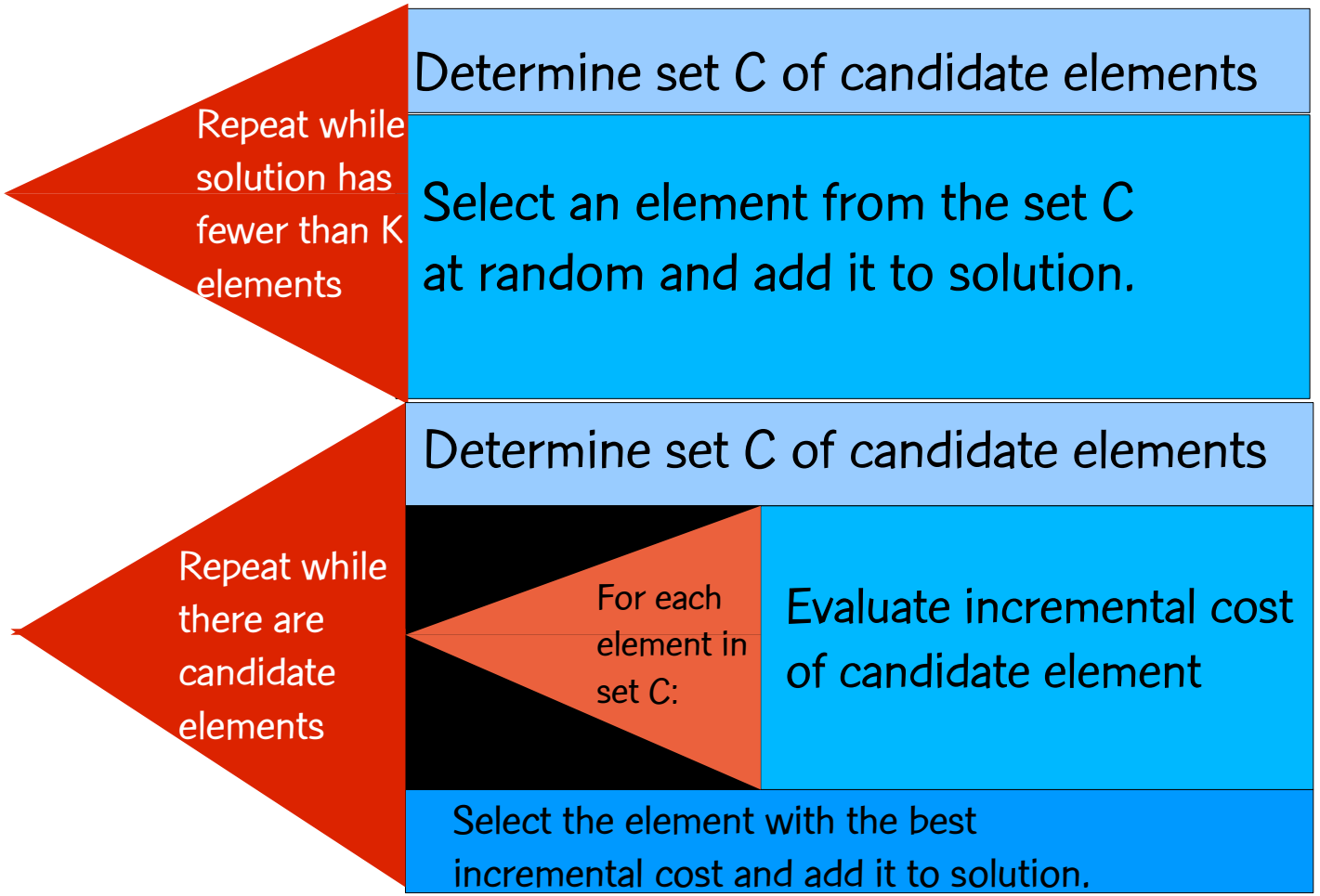
Construction phase: sampled greedy

[R. & Werneck, 2004]



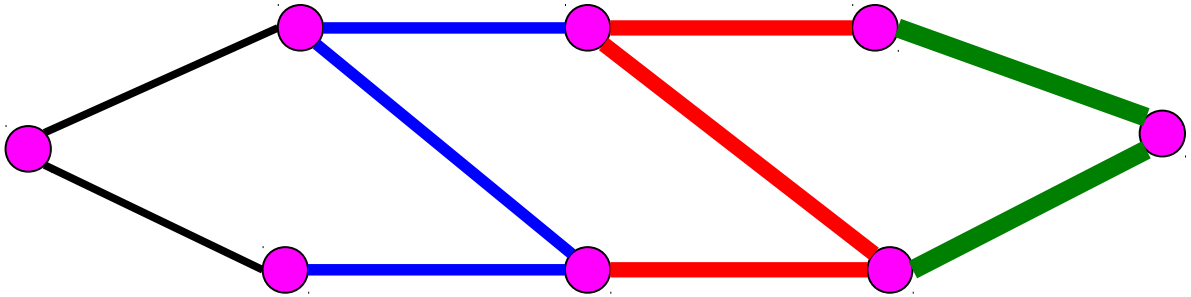
Construction phase: random+greedy

[R. & Werneck, 2004]



Construction with cost perturbation

Canuto, R., & Ribeiro (2001)

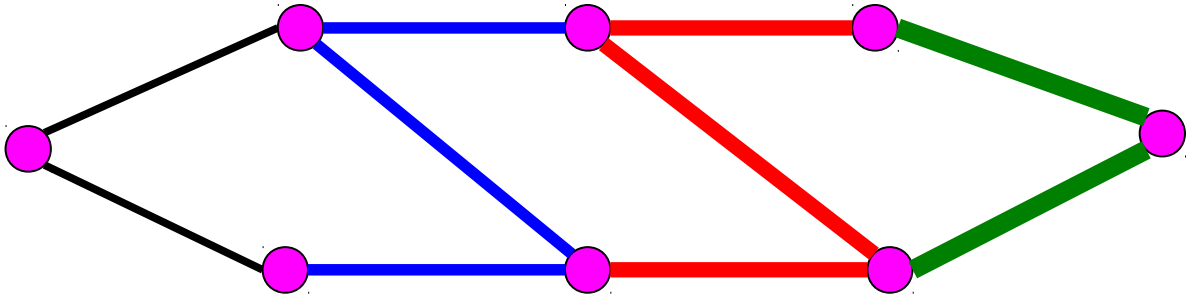


W( ) < W( ) < W( ) < W( )

Perturb with costs increasing from top to bottom.

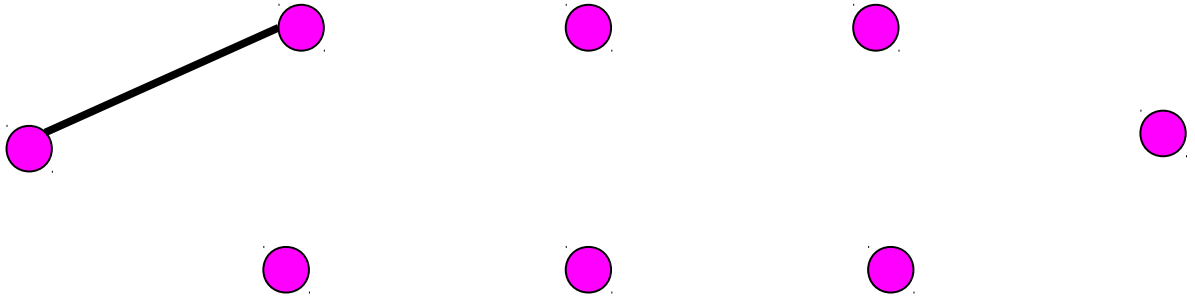
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



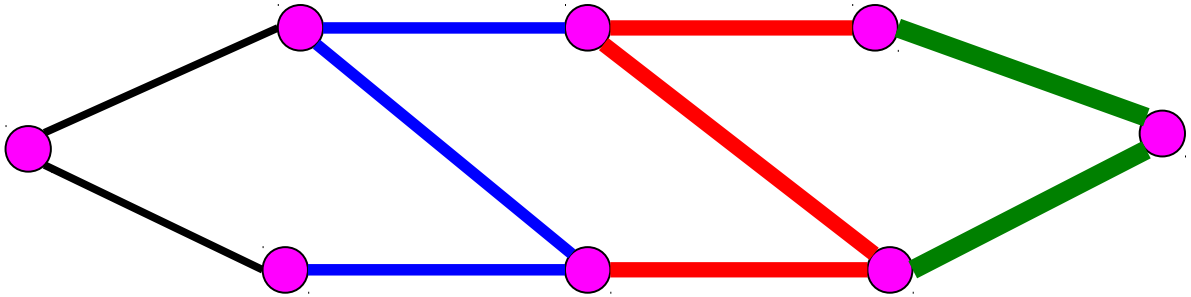
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



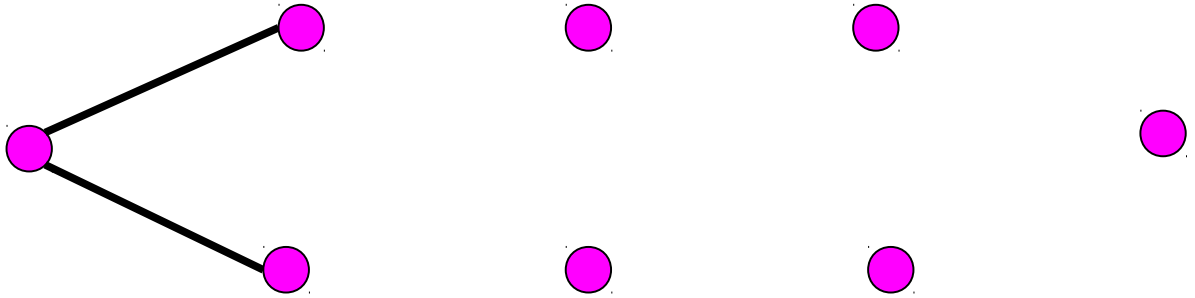
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



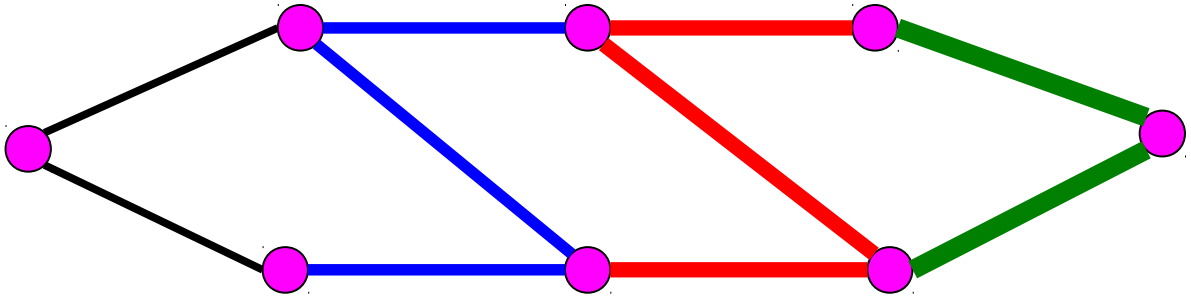
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



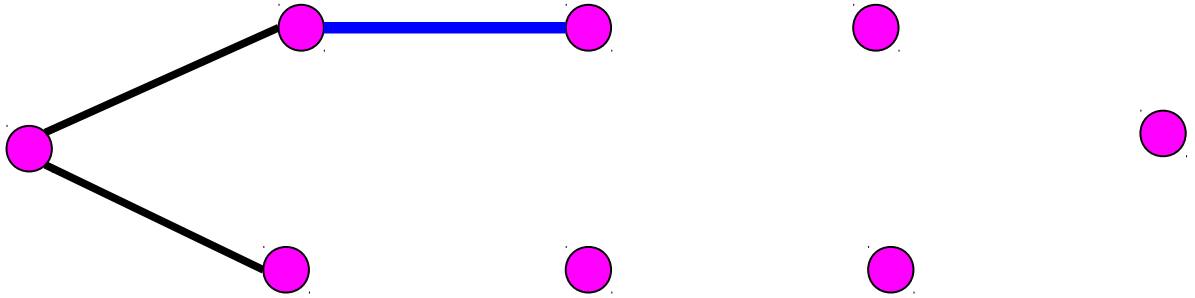
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



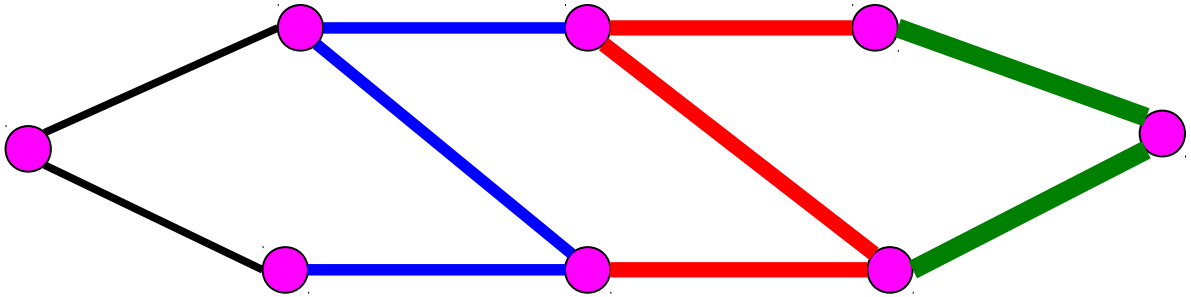
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



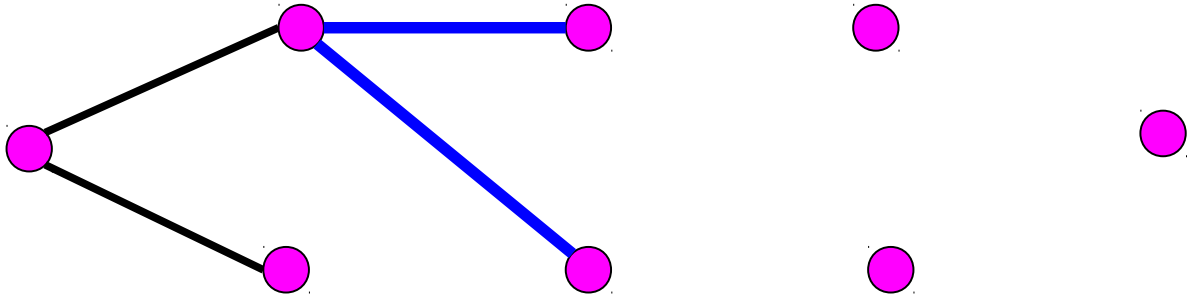
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



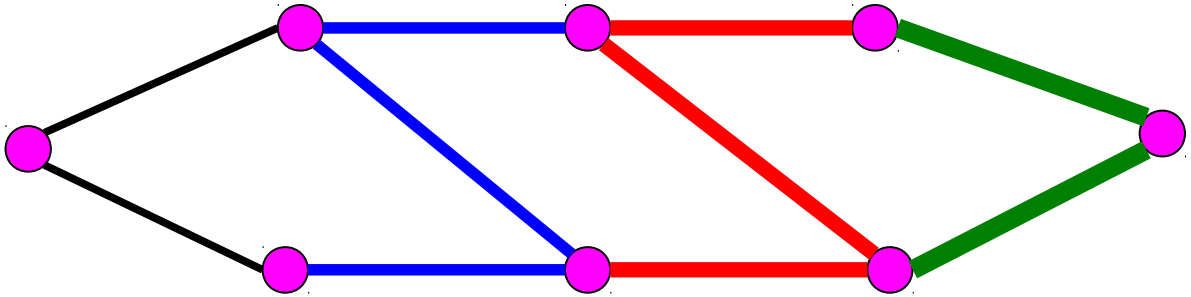
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



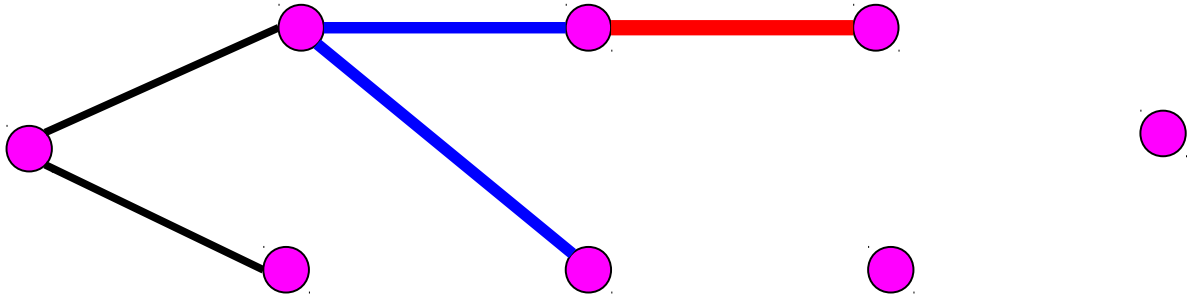
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



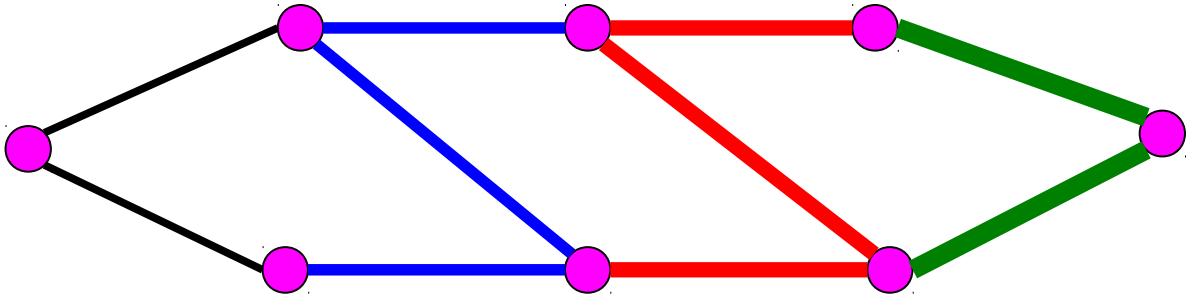
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



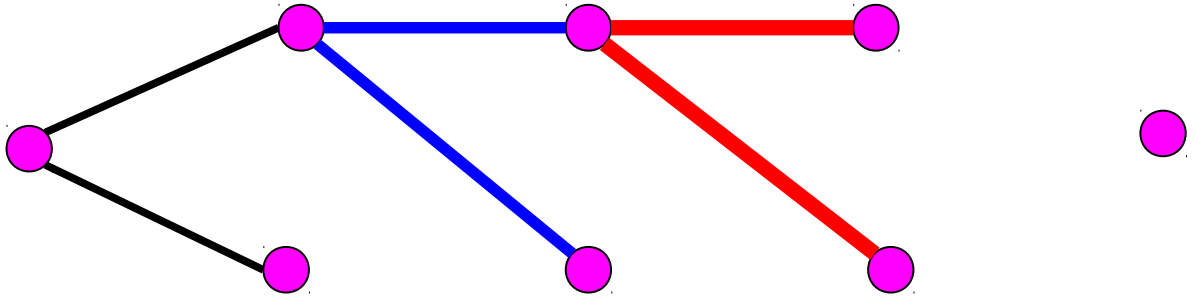
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



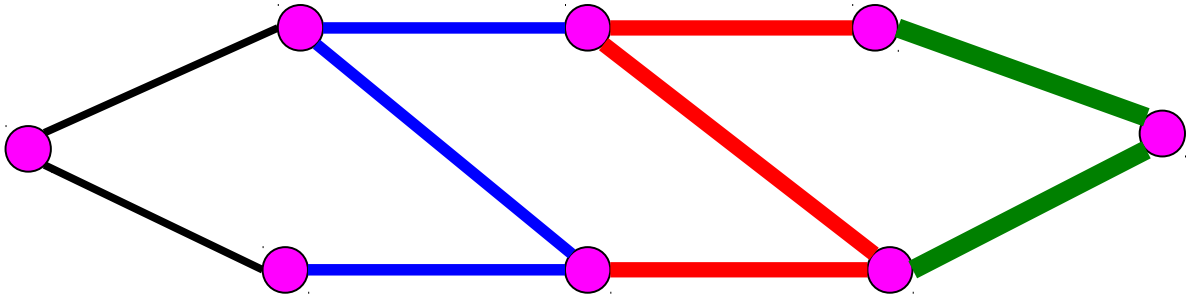
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



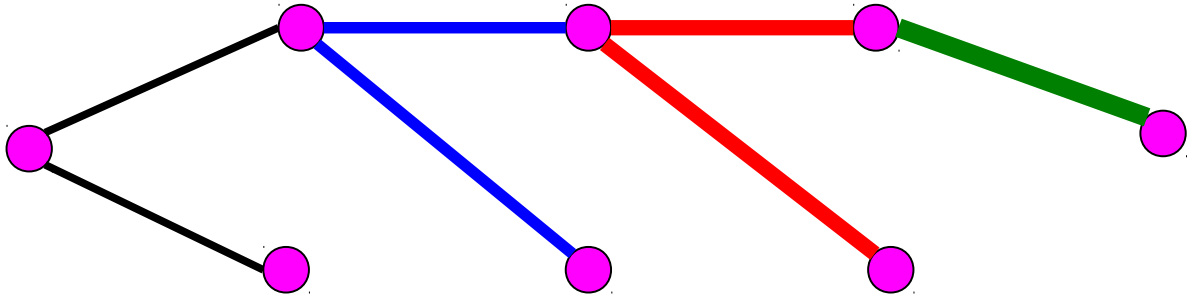
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



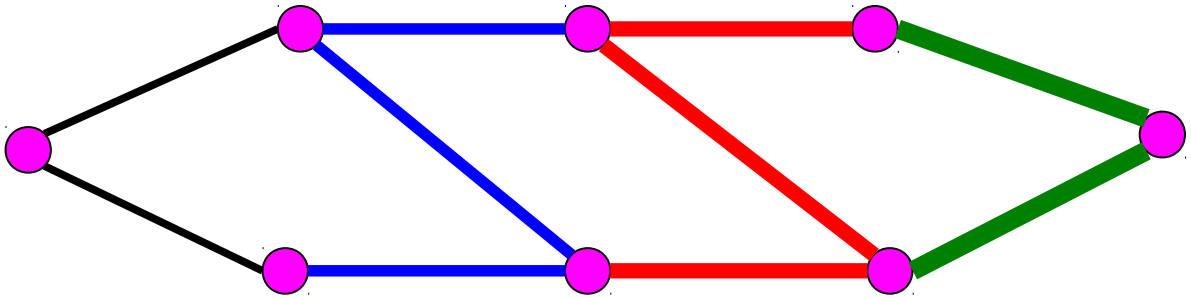
Perturb with costs increasing from top to bottom.

W( ) < W( ) < W( ) < W( )



Construction with cost perturbation

Canuto, R., & Ribeiro (2001)

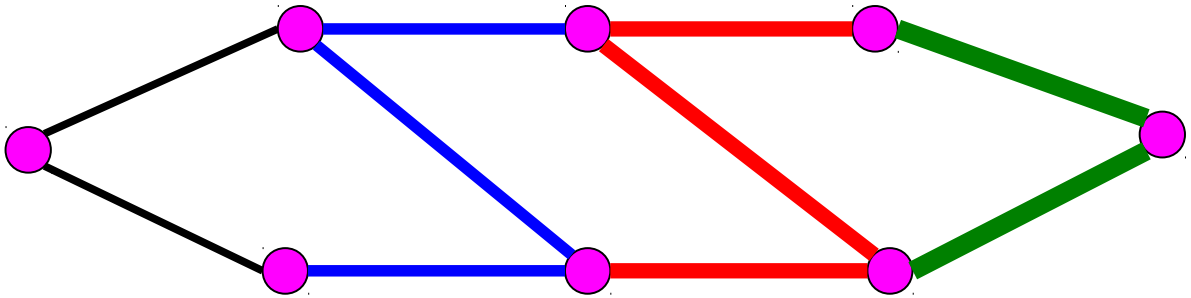


W( ) < W( ) < W( ) < W( )

Perturb with costs increasing from bottom to top.

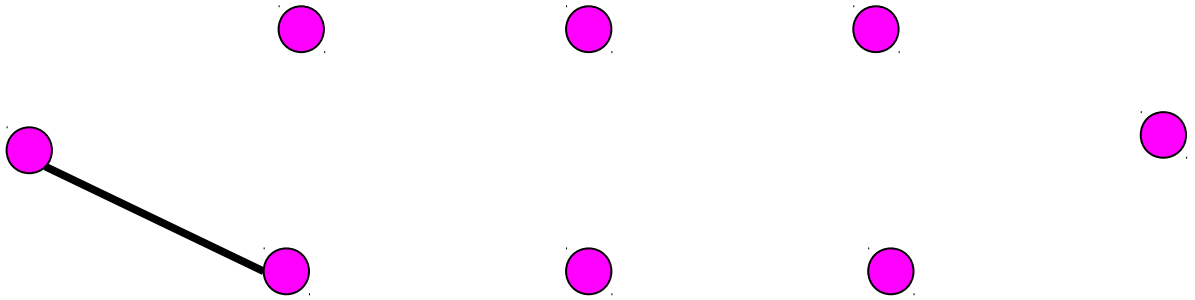
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



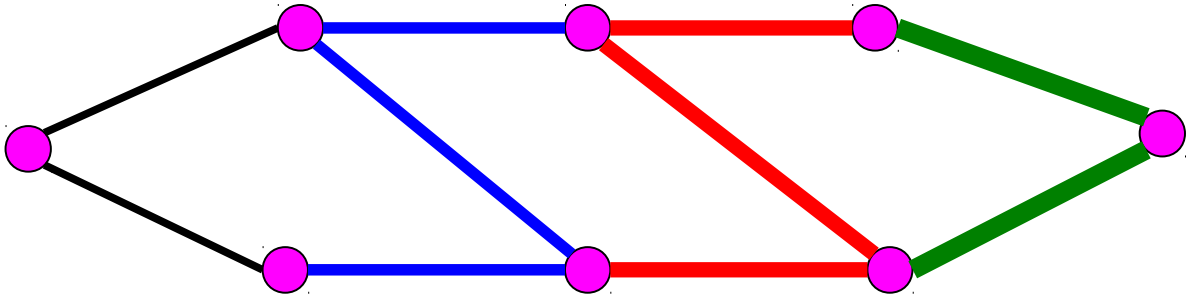
Perturb with costs increasing from bottom to top.

W( ) < W( ) < W( ) < W( )



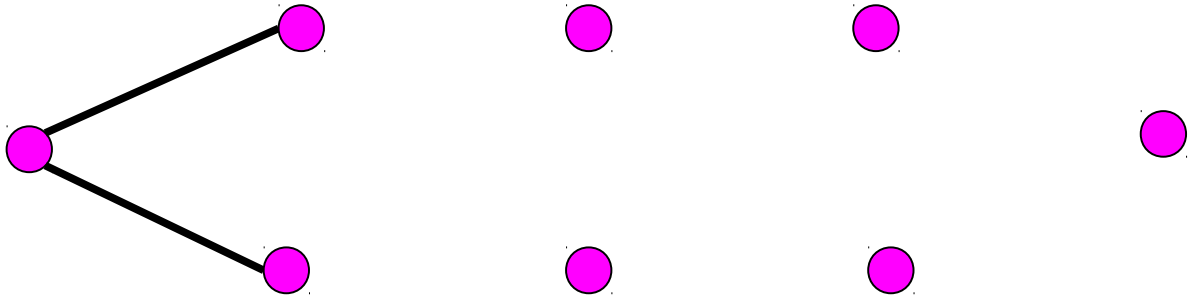
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



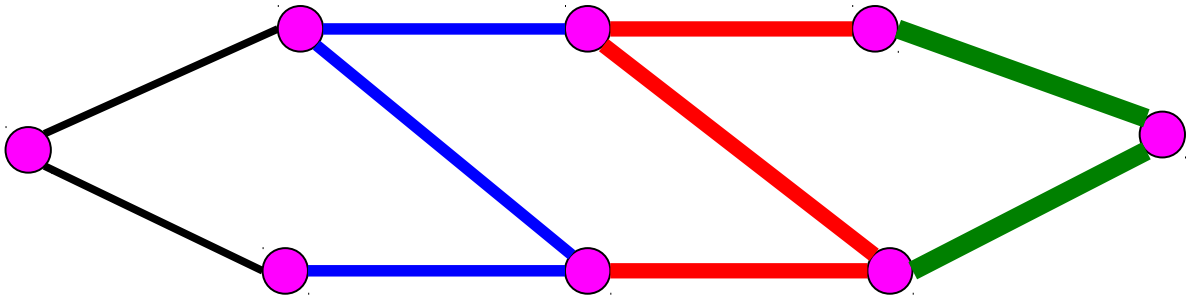
Perturb with costs increasing from bottom to top.

W( ) < W( ) < W( ) < W( )



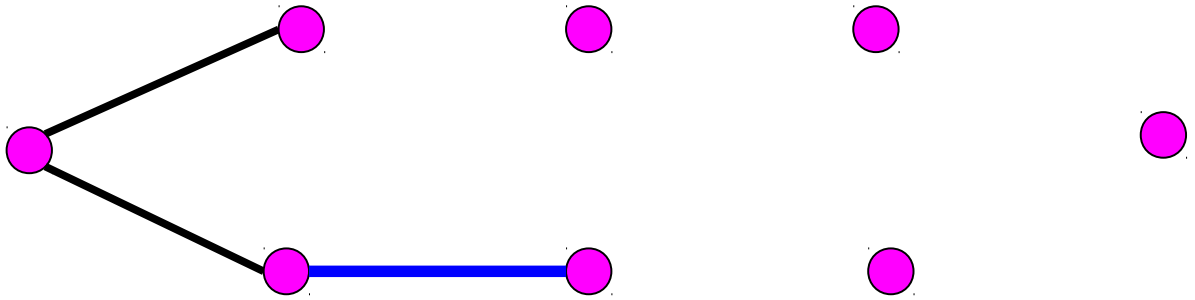
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



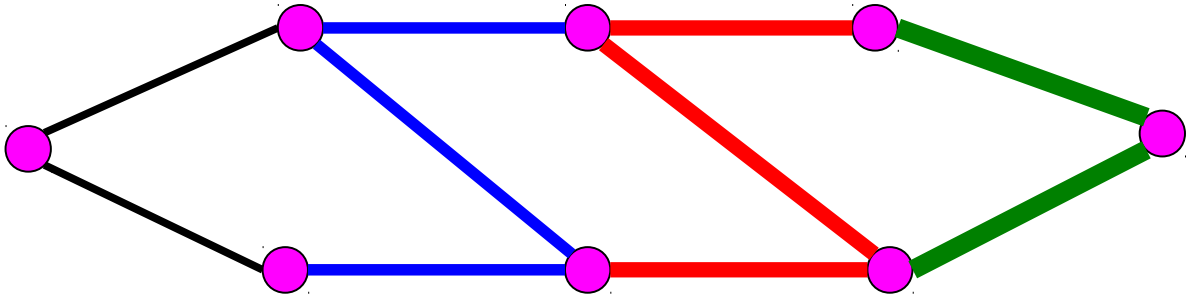
Perturb with costs increasing from bottom to top.

$W(\ ) < W(\ ) < W(\ ) < W(\ )$



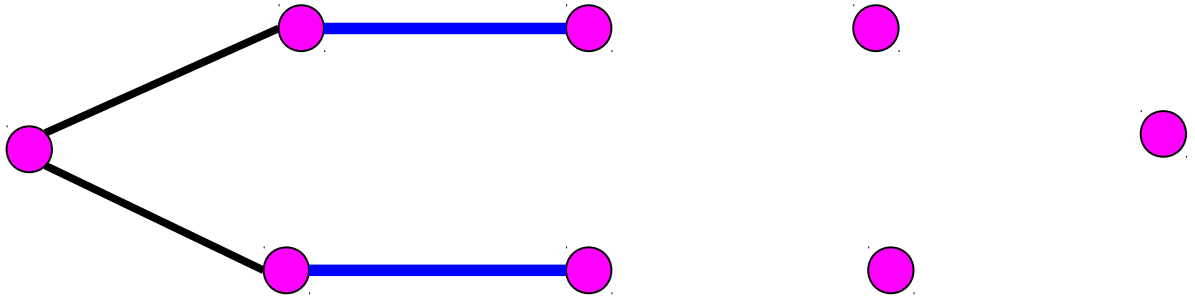
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



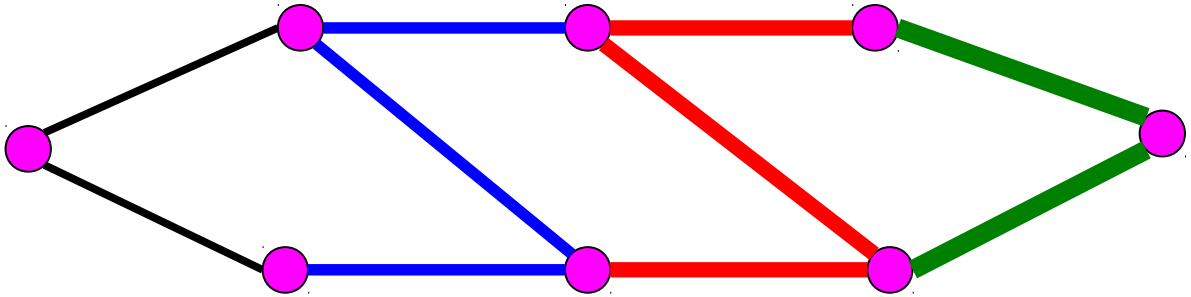
Perturb with costs increasing from bottom to top.

W( ) < W( ) < W( ) < W( )



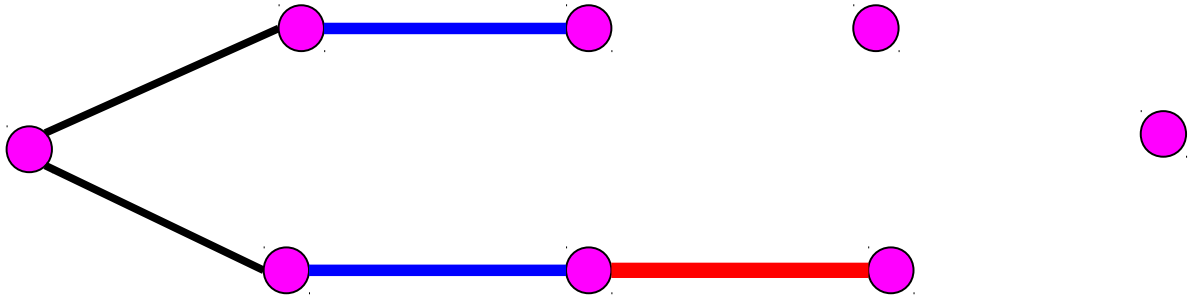
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



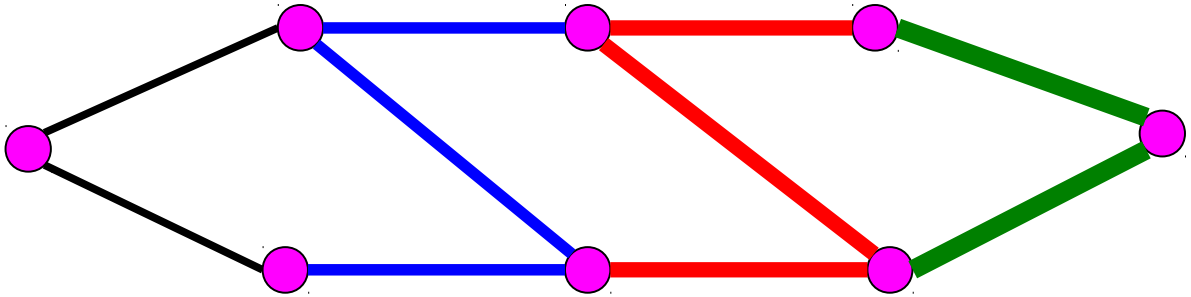
Perturb with costs increasing from bottom to top.

$W(\ ) < W(\ ) < W(\ ) < W(\ )$



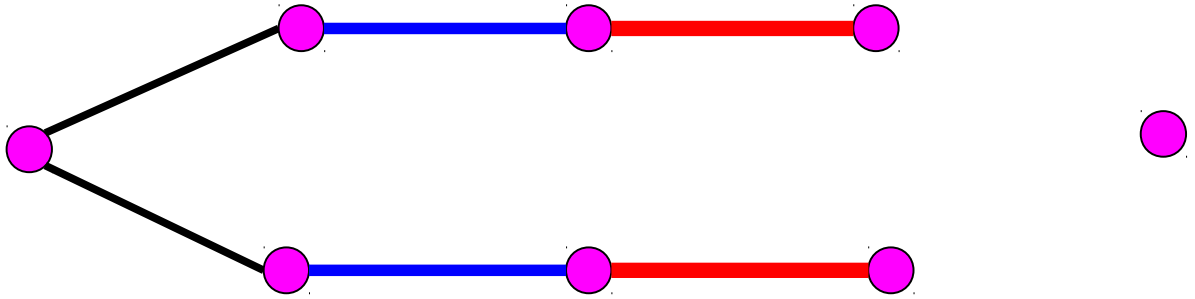
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



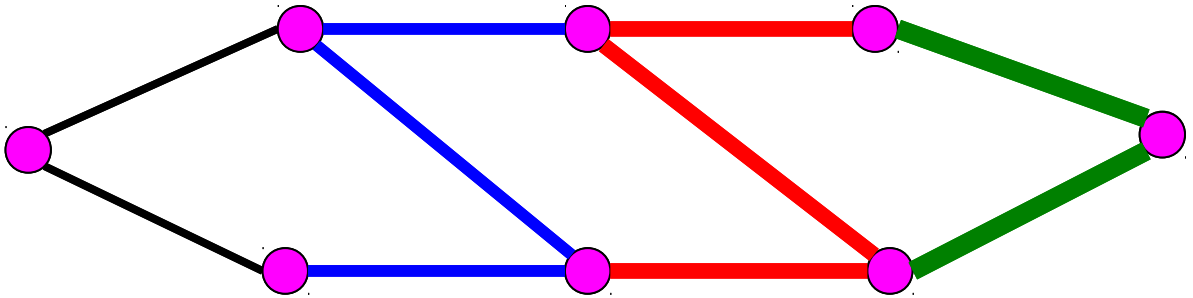
Perturb with costs increasing from bottom to top.

W( ) < W( ) < W( ) < W( )



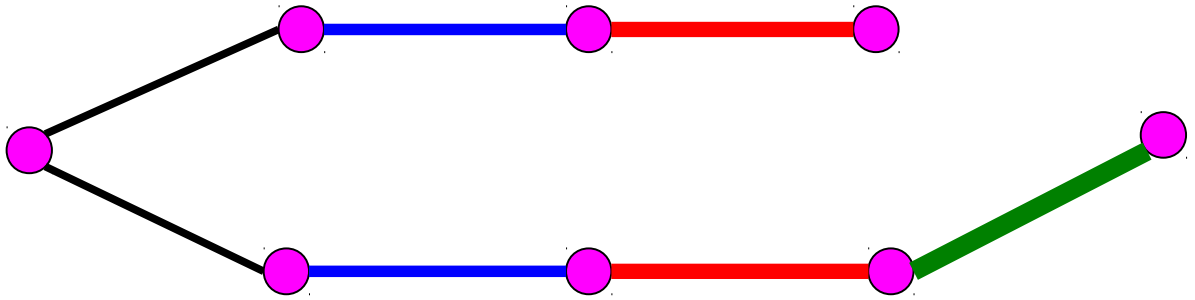
Construction with cost perturbation

Canuto, R., & Ribeiro (2001)



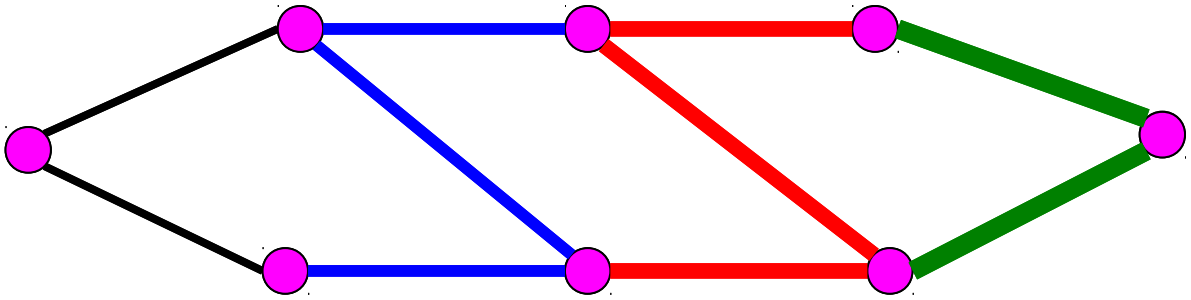
Perturb with costs increasing from bottom to top.

W( ) < W( ) < W( ) < W( )



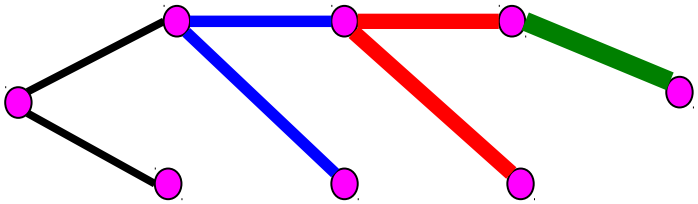
Construction with cost perturbation

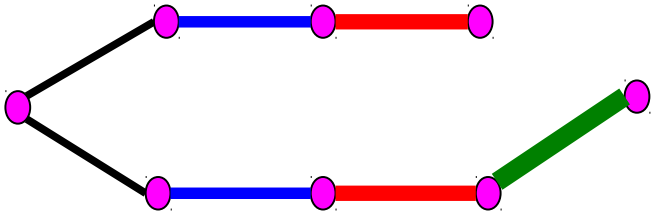
Canuto, R., & Ribeiro (2001)



W( ) < W( ) < W( ) < W( )

Greedy heuristic generates two different spanning trees.





Reactive GRASP

Prais & Ribeiro (2000)

When building RCL, what α to use?

Fix a some value $0 \leq \alpha \leq 1$

Choose α at random (uniformly) at each GRASP iteration.

Another approach reacts to search ...

At each GRASP iteration, a value of the RCL parameter α is chosen from a discrete set of values $[\alpha_1, \alpha_2, \ldots, \alpha_m]$.

The probability that α_k is selected is p_k.

Reactive GRASP: adaptively changes the probabilities $[p_1, p_2, \ldots, p_m]$ to favor values of α that produce good solutions.

Reactive GRASP

Prais & Ribeiro (2000)

Reactive GRASP for minimization ...

Initially $p_k = 1/m$, for $k = 1,...,m$. (α's are selected uniformly at random)

Define

$F(S^*)$ be the best solution so far

A_k be the average value of the solutions obtained with α_k

Every N_α GRASP iterations, compute

$q_k = F(S^*) / A_k$, for $k = 1,...,m$

$p_k = q_k / \text{sum}(q_i \mid i = 1,...,m)$

Reactive GRASP

Prais & Ribeiro (2000)

Reactive GRASP for minimization ...

Initially $p_k = 1/m$, for $k = 1,...,m$. (α's are selected uniformly at random)

Define

$F(S^*)$ be the best solution so far

A_k be the average value of the solutions obtained with α_k

Every N_α GRASP iterations, compute

$q_k = F(S^*) / A_k$, for $k = 1,...,m$

$p_k = q_k / \text{sum}(q_i \mid i = 1,...,m)$

The more suitable is α_k, the larger is q_k, and consequently p_k, making α_k more likely to chosen.

Path-relinking (PR)

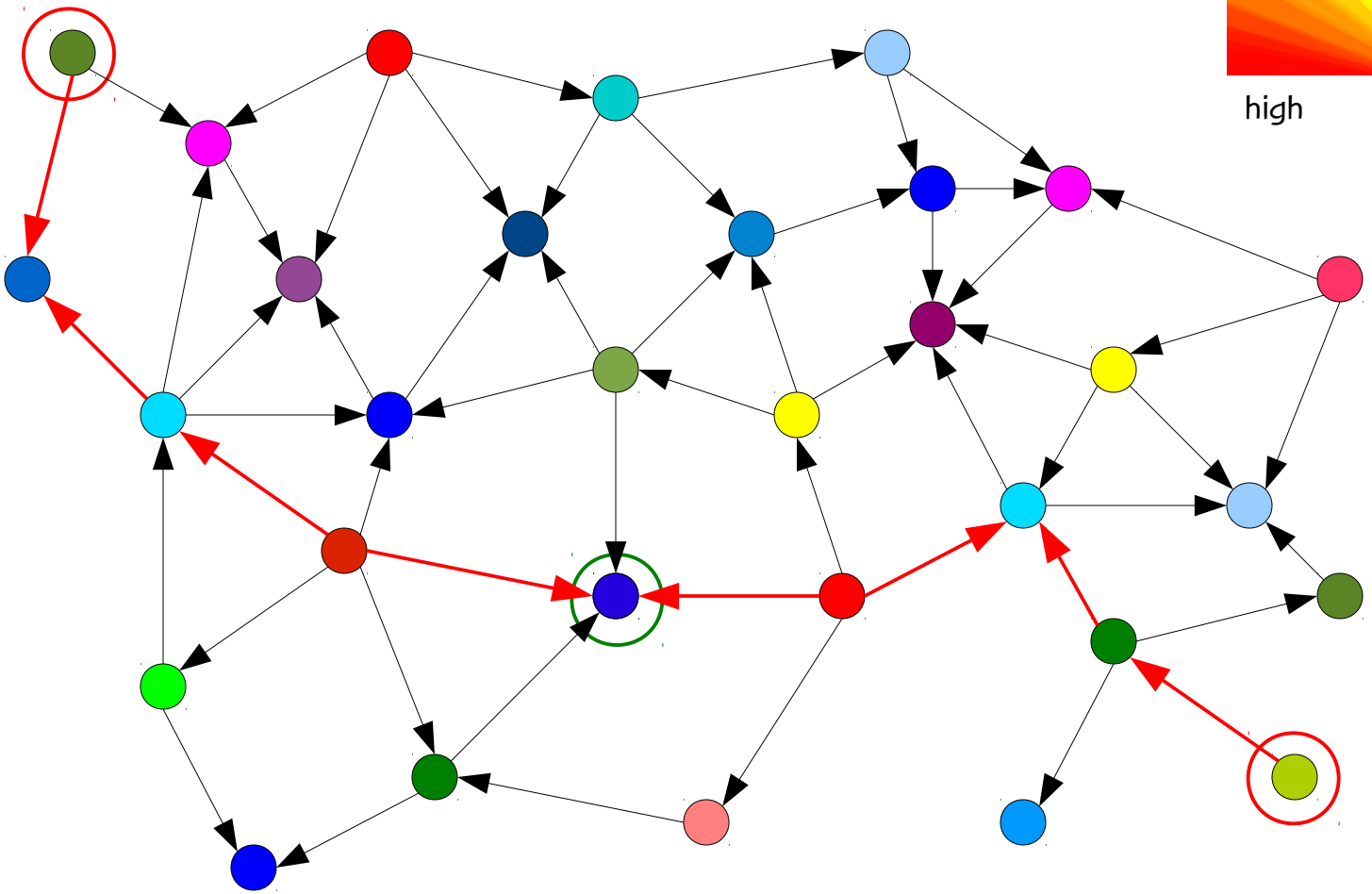
Path-relinking

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

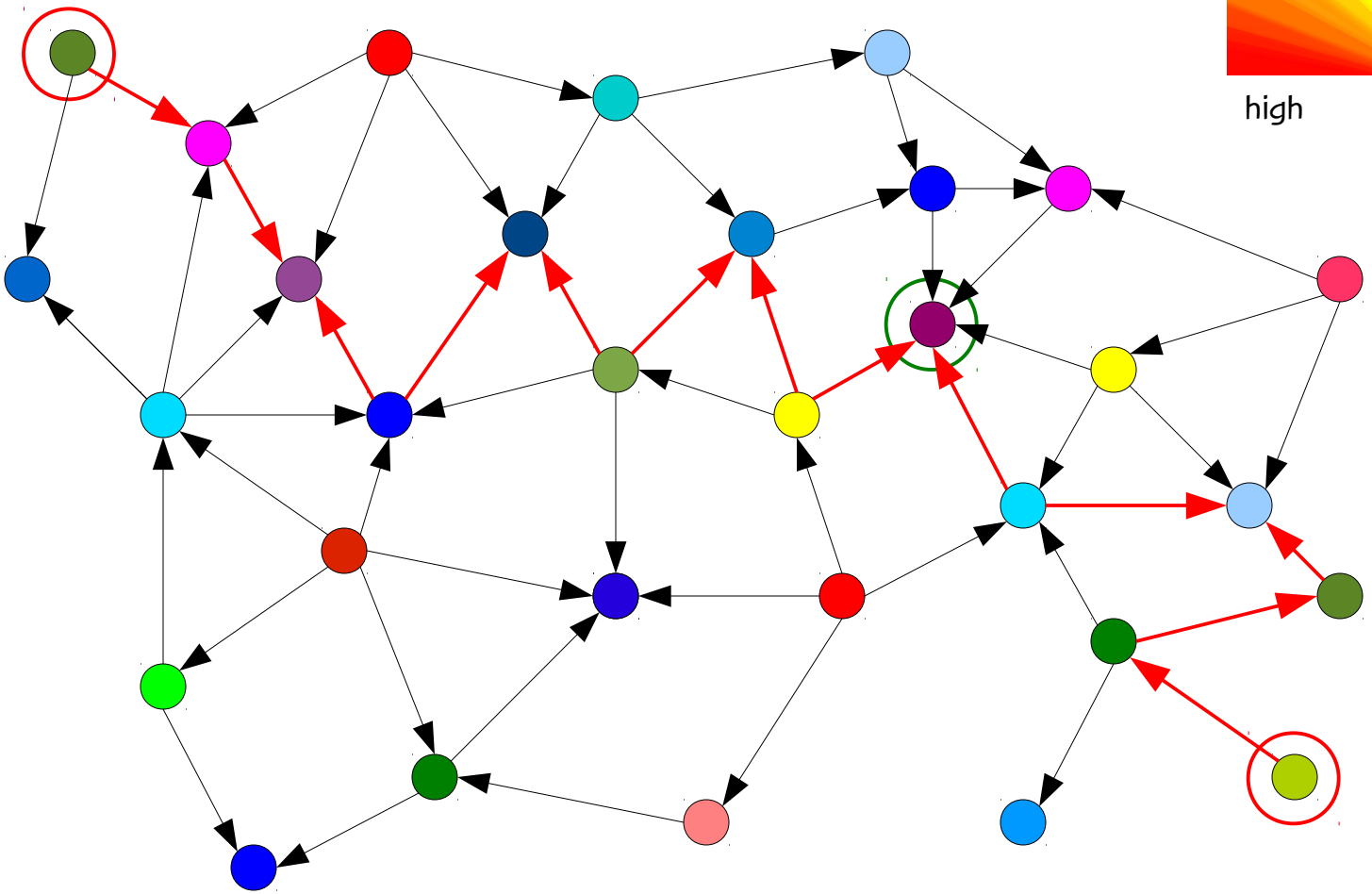
Originally proposed in the context of tabu search and scatter search.

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

Path relinking: Explores path connecting two good-quality solutions

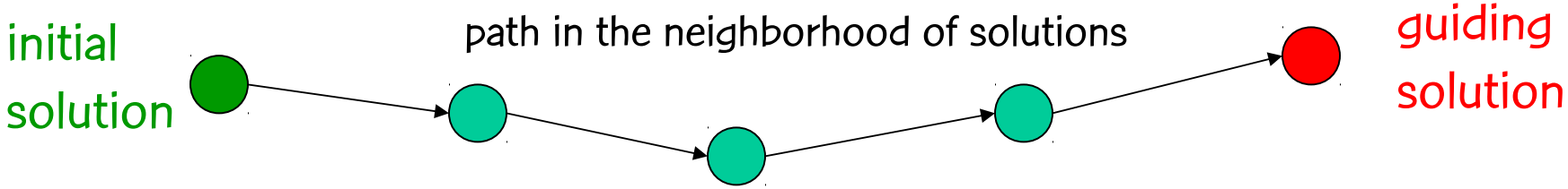


Path relinking: Explores path connecting two good-quality solutions



Path-relinking

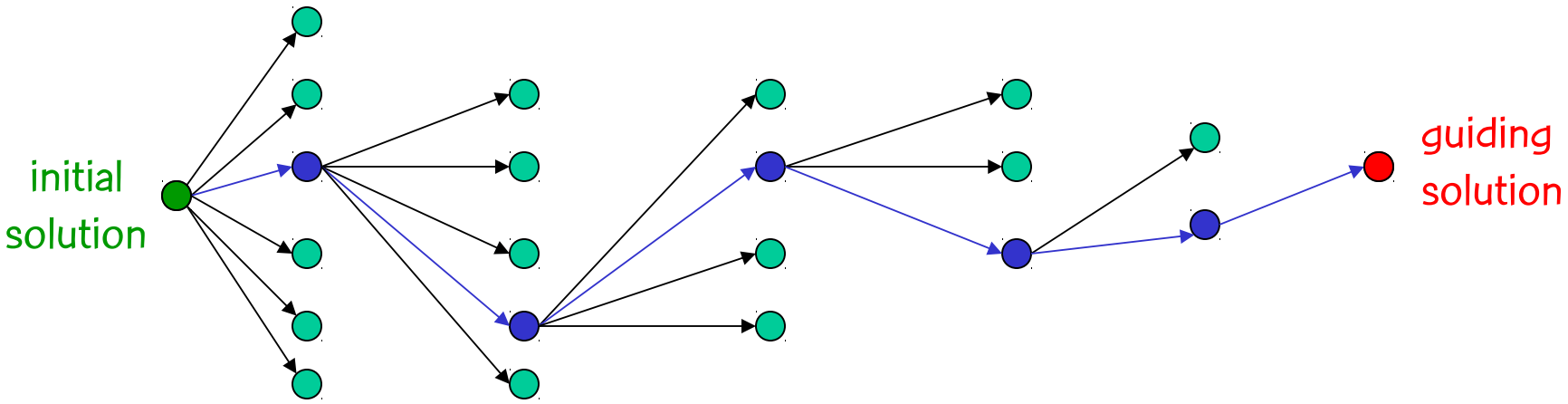
Exploration of trajectories that connect high quality (elite) solutions:



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:



starting solution

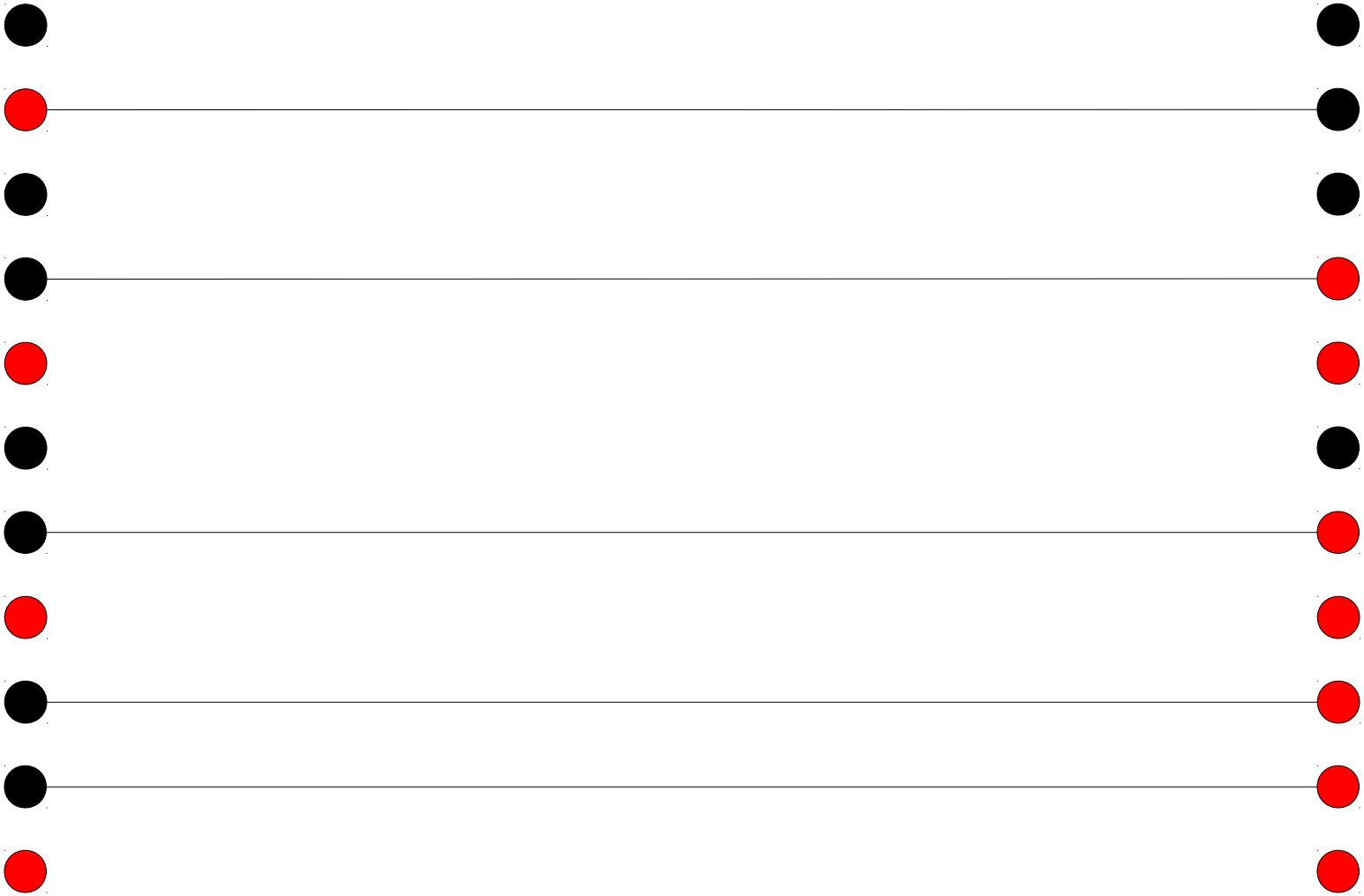
PR example

guiding solution

starting solution x

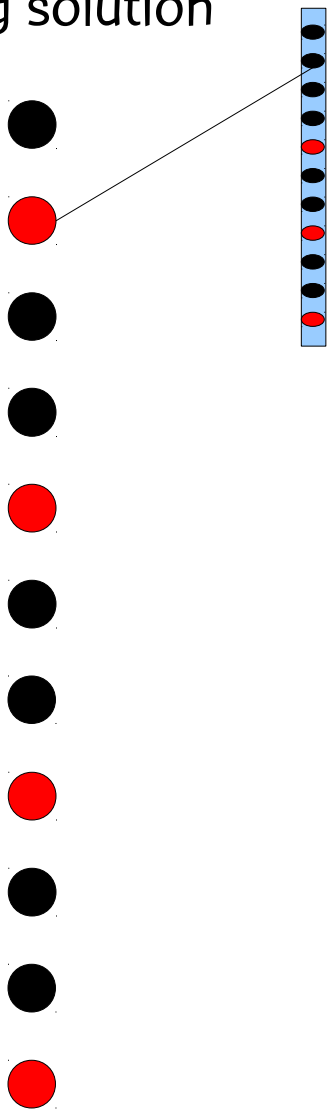
PR example

guiding solution y



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

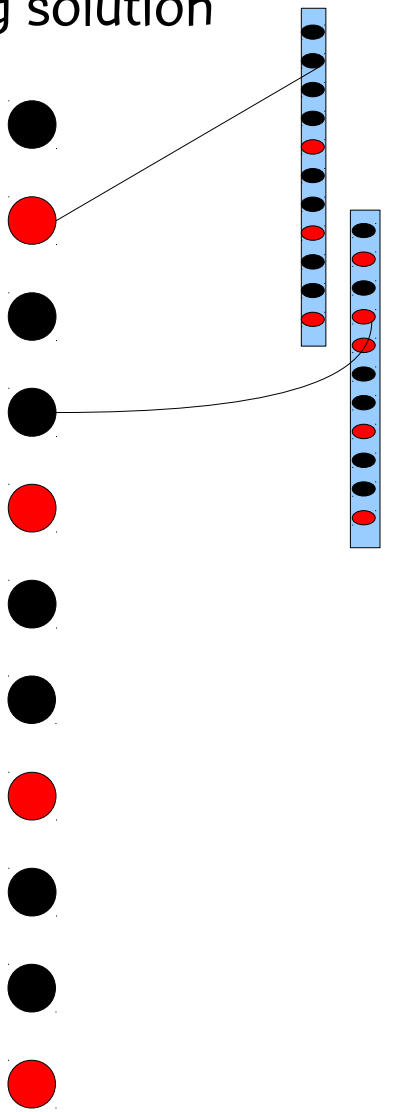
starting solution



PR example

guiding solution

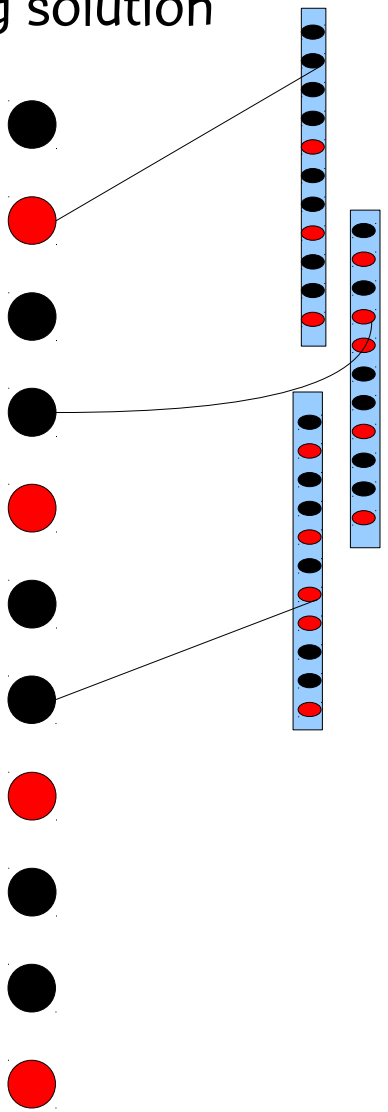
starting solution



PR example

guiding solution

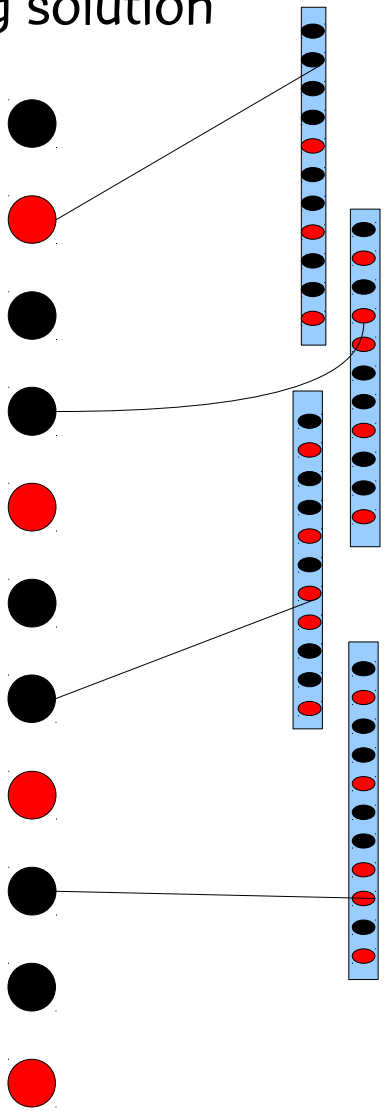
starting solution



PR example

guiding solution

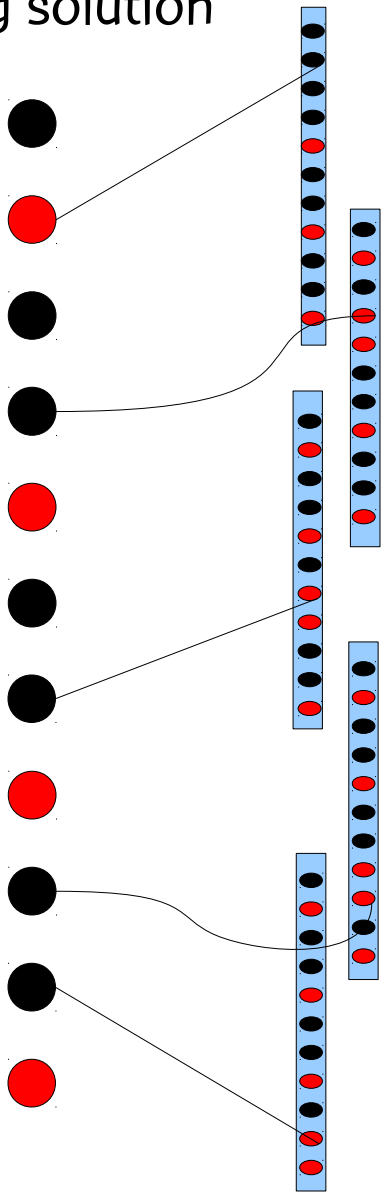
starting solution



PR example

guiding solution

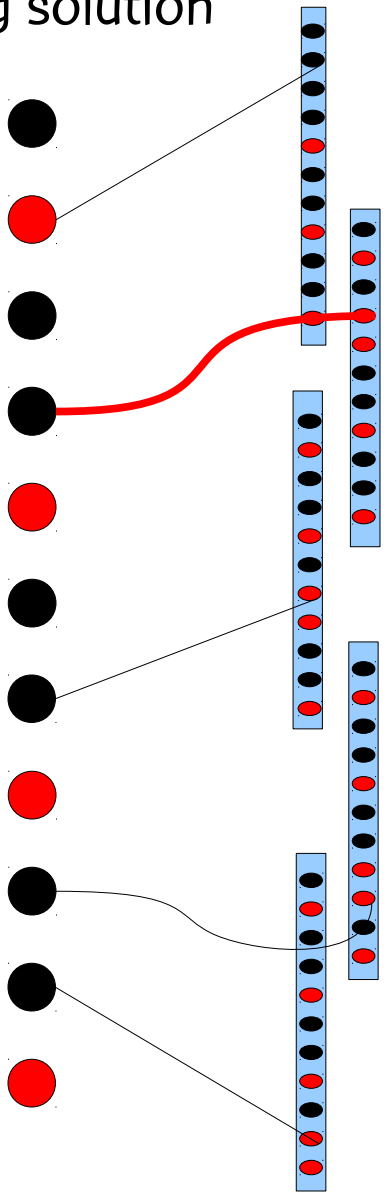
starting solution



PR example

guiding solution

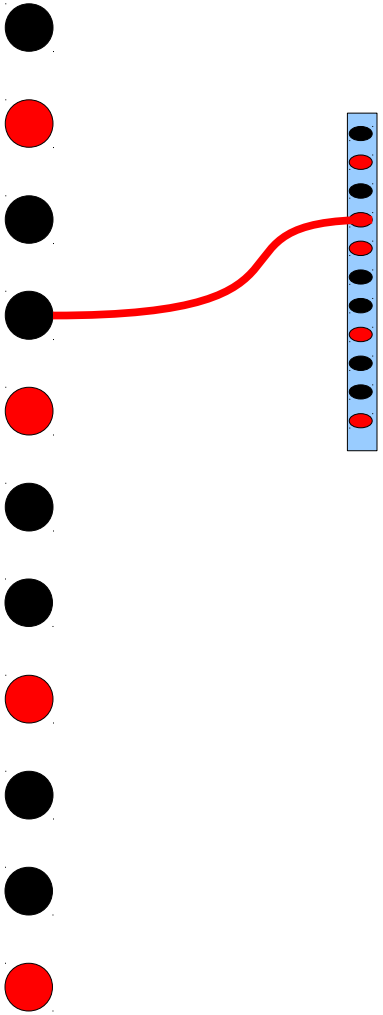
starting solution



PR example

guiding solution

starting solution



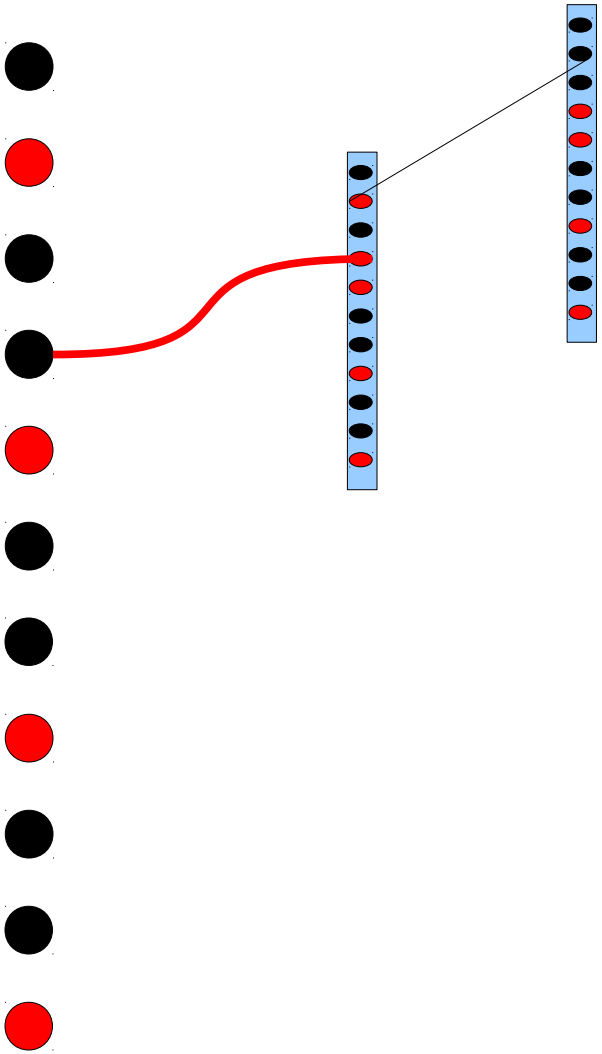
PR example

guiding solution

starting solution

PR example

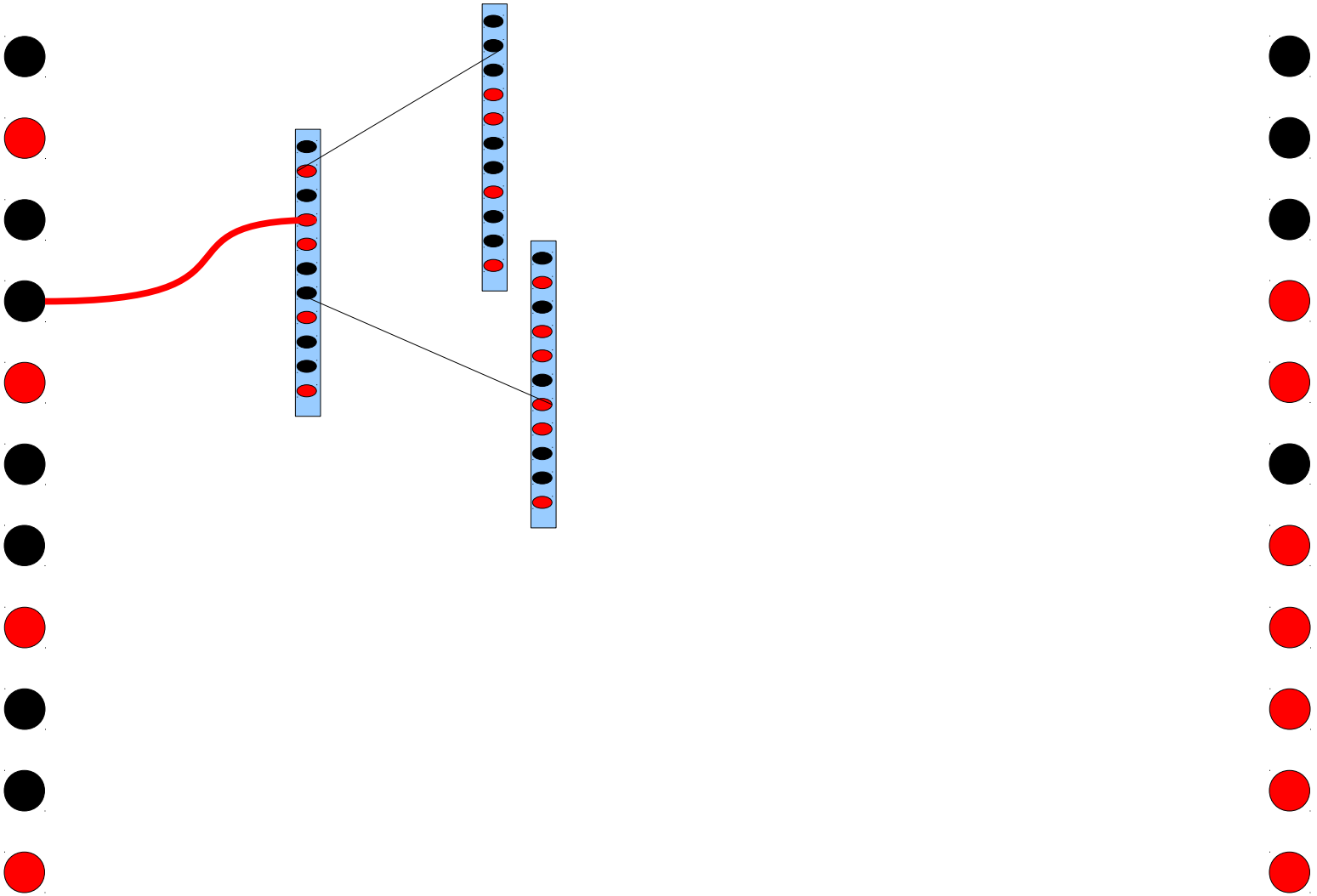
guiding solution



starting solution

PR example

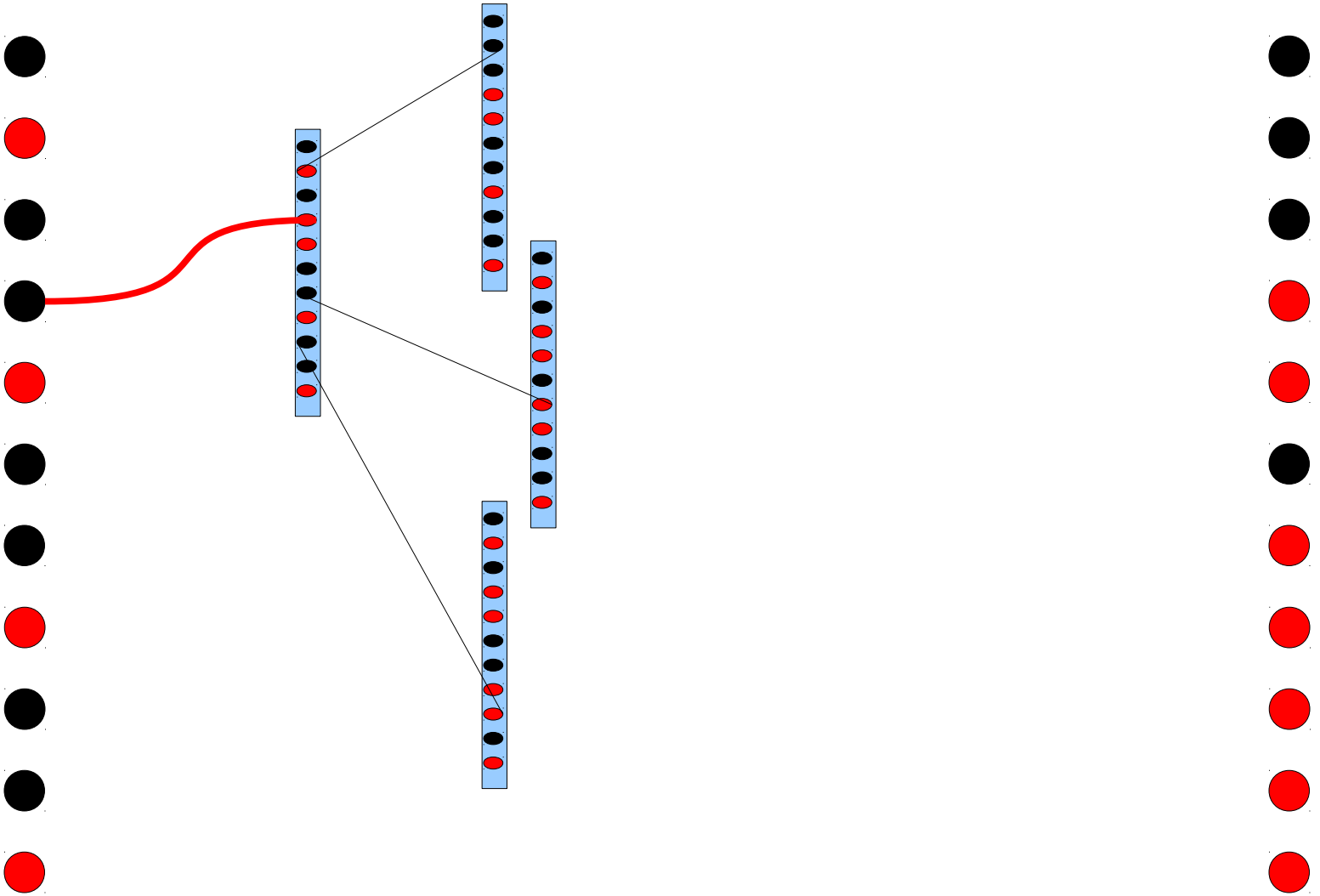
guiding solution



starting solution

PR example

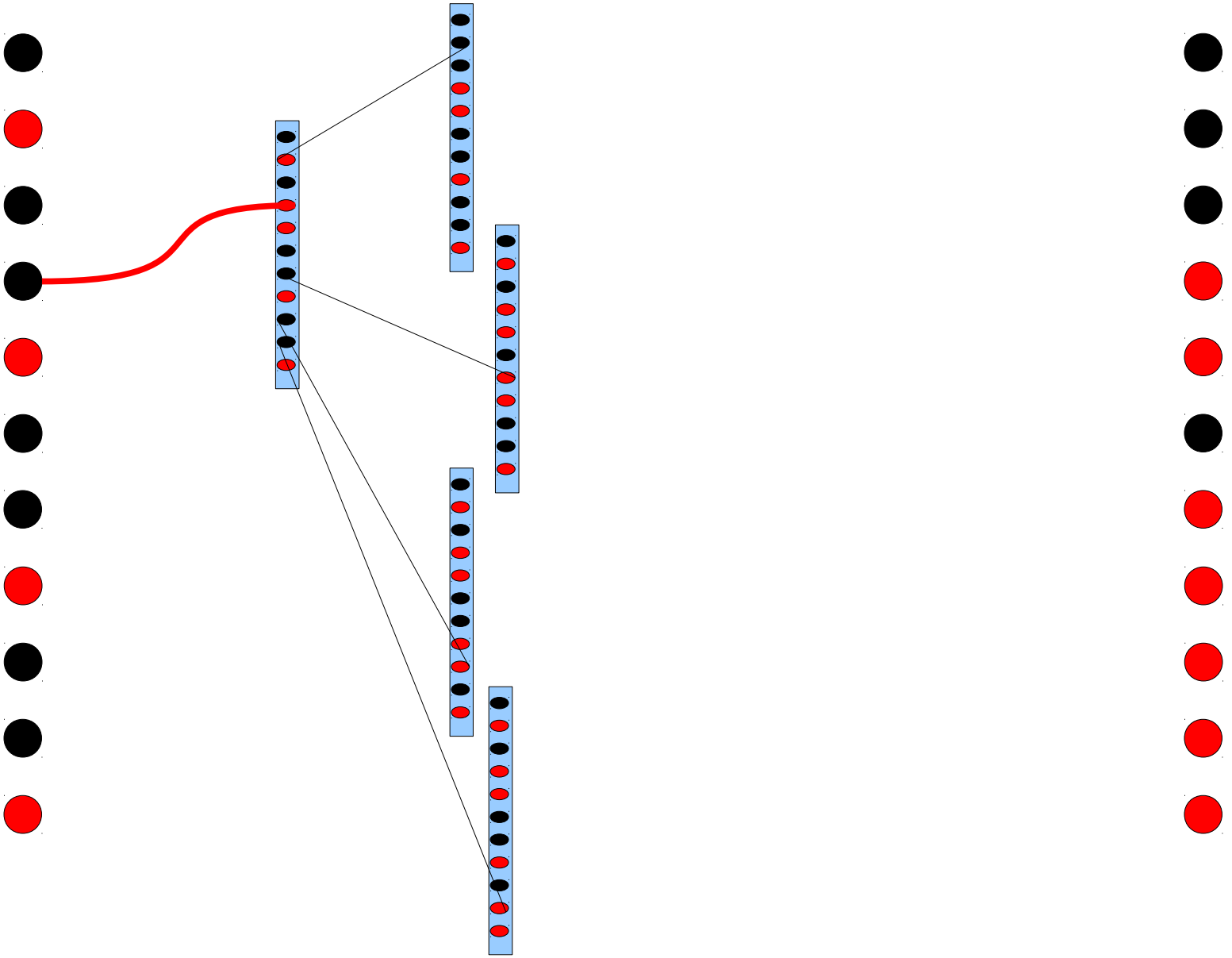
guiding solution



starting solution

PR example

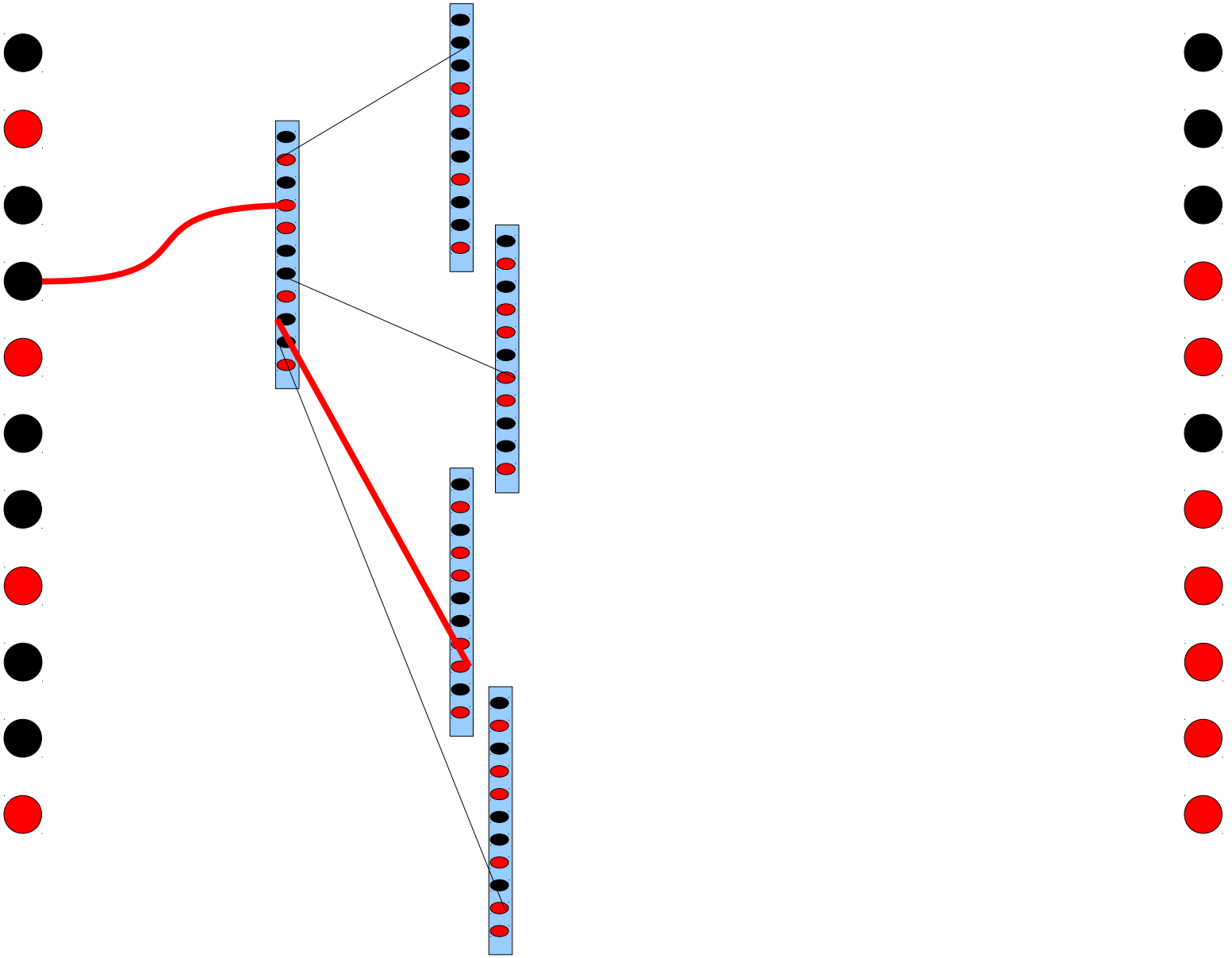
guiding solution



starting solution

PR example

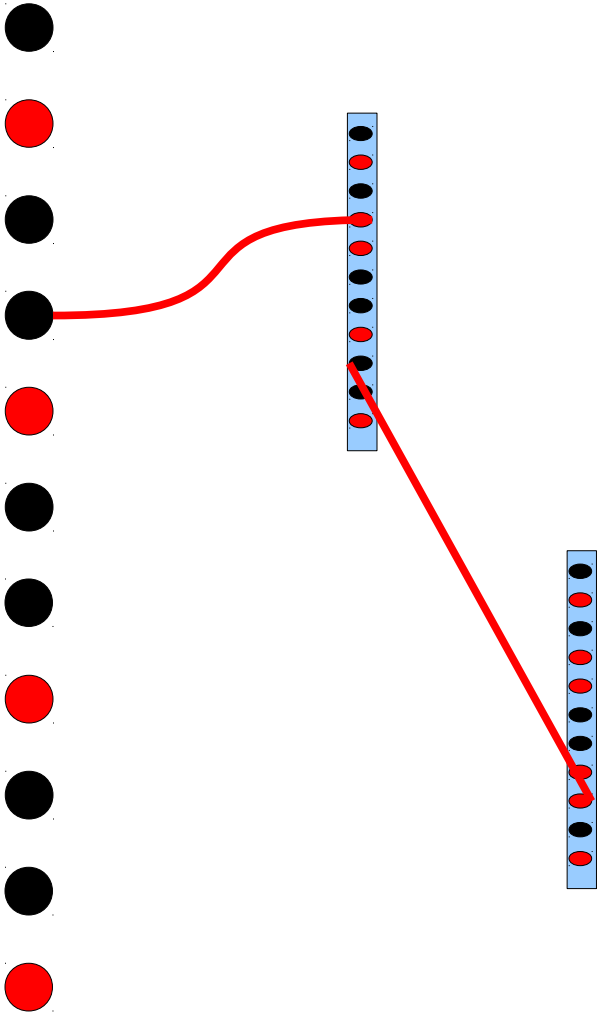
guiding solution



starting solution

PR example

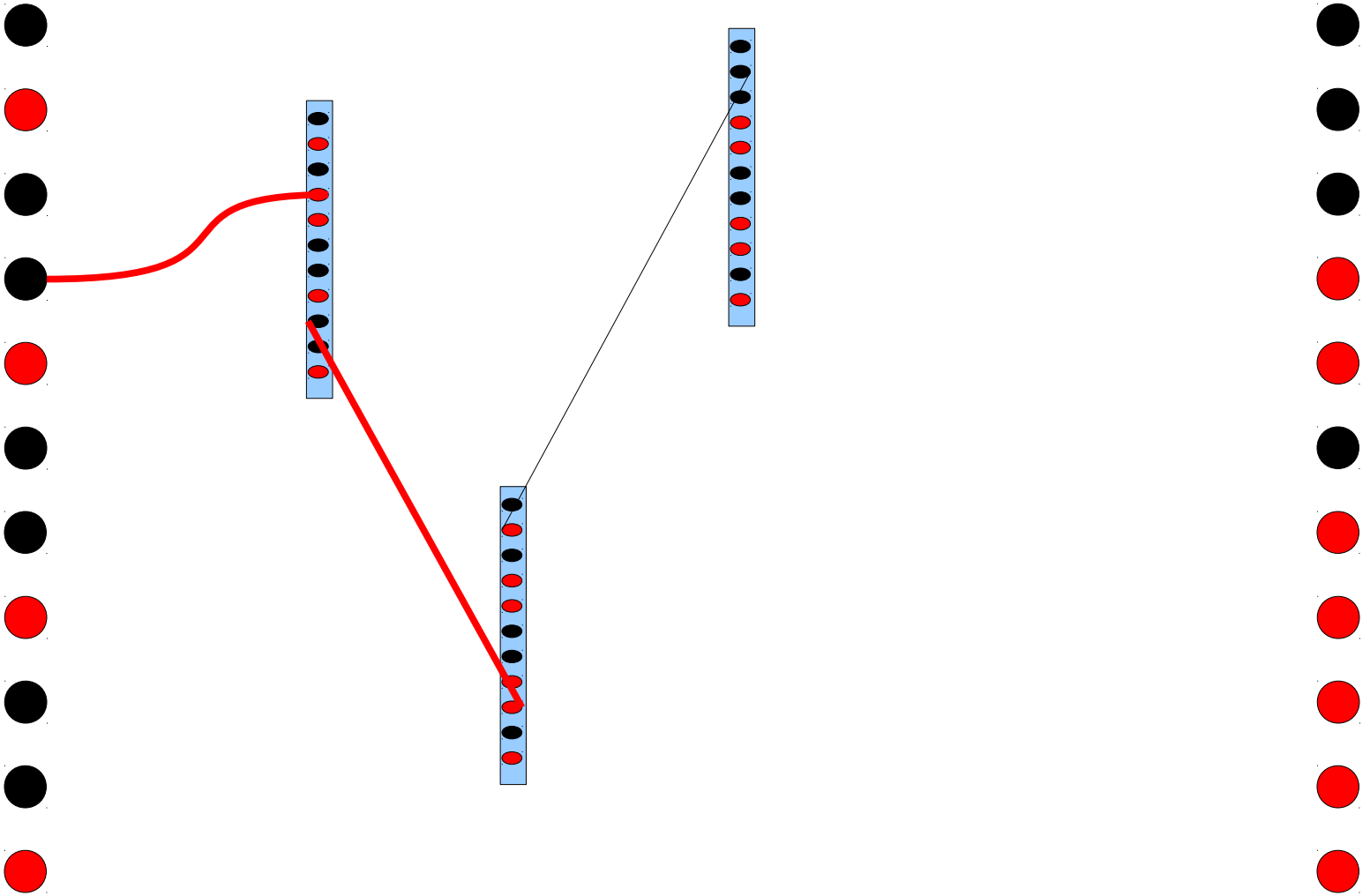
guiding solution



starting solution

PR example

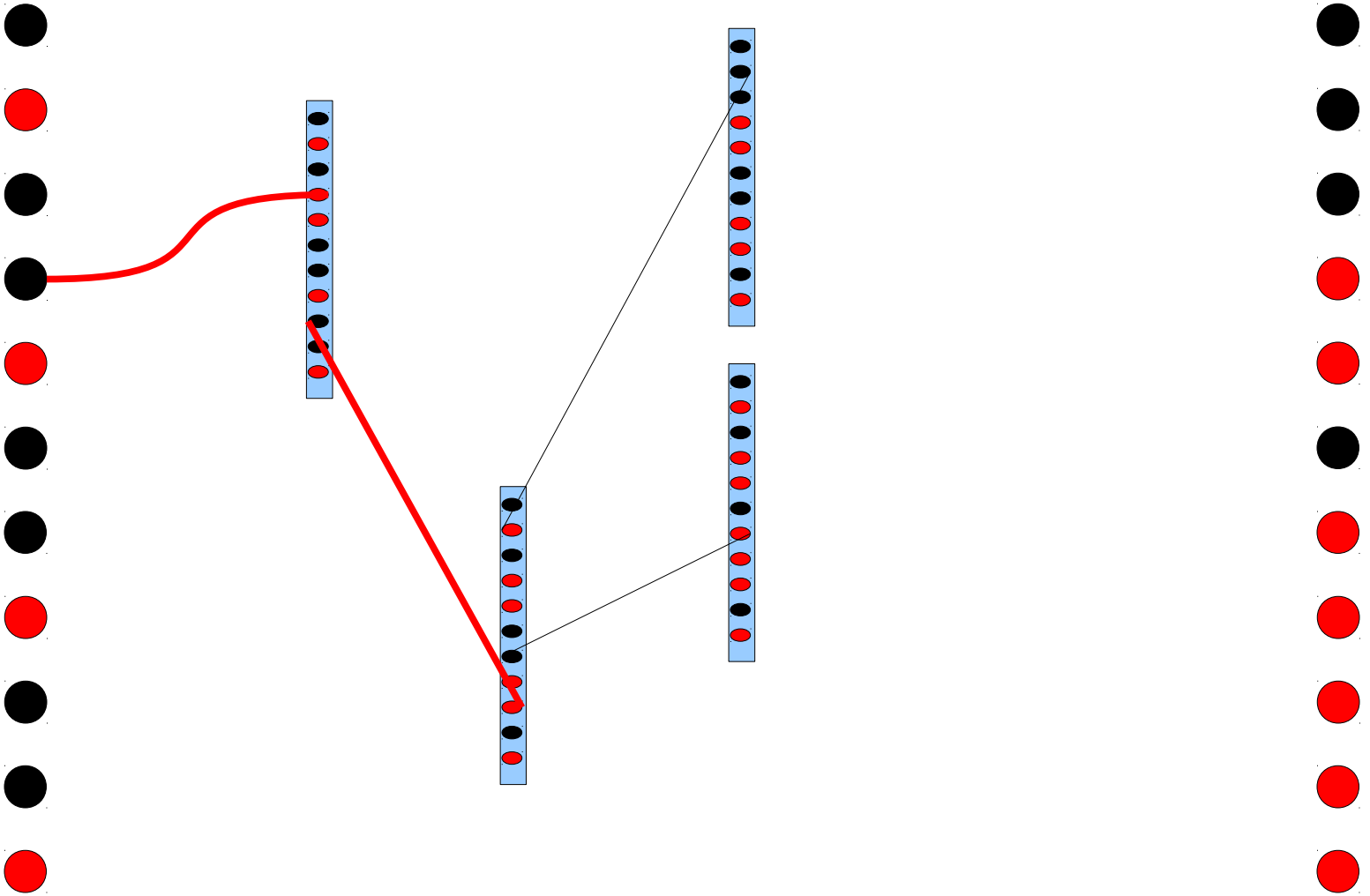
guiding solution



starting solution

PR example

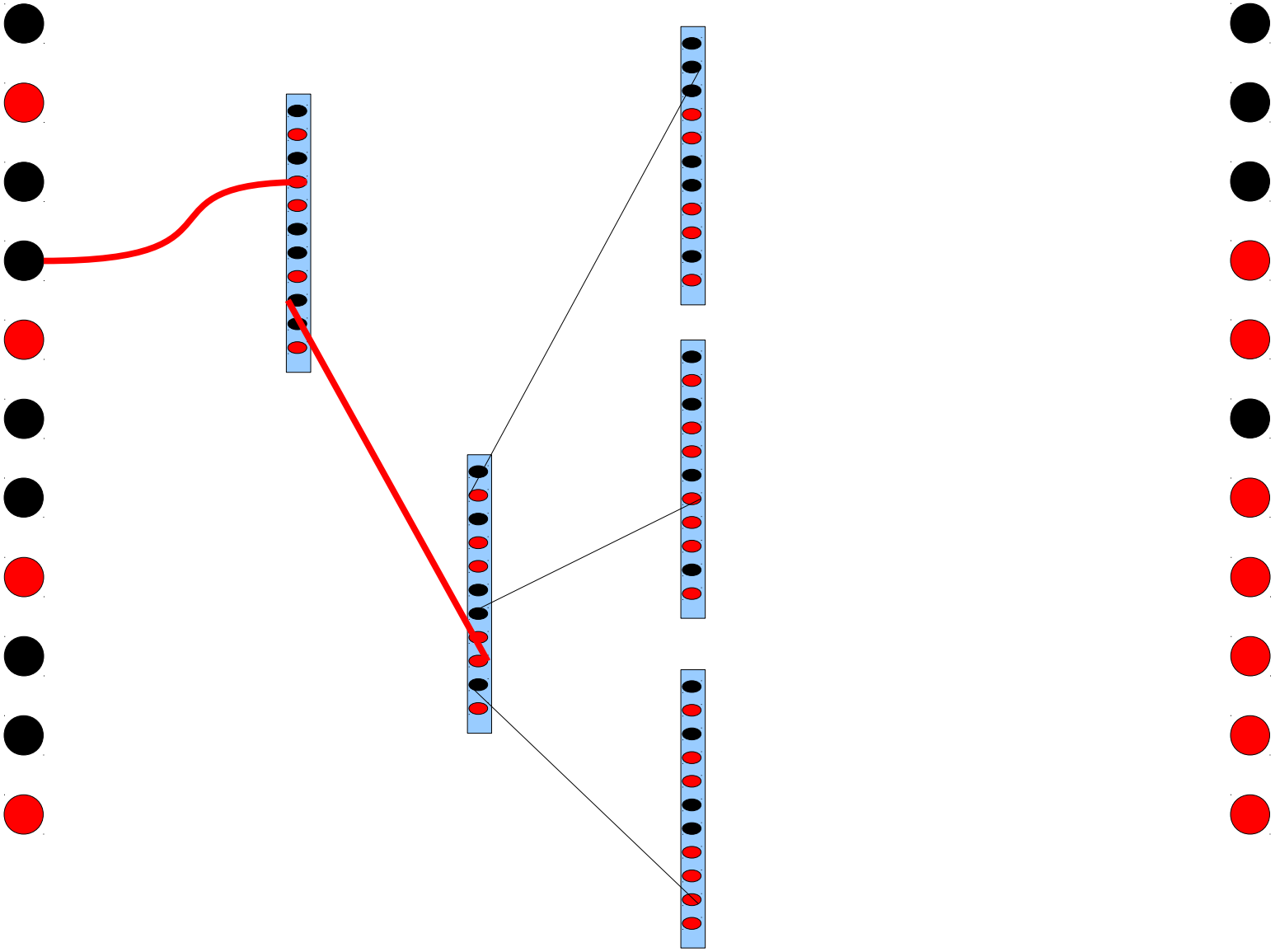
guiding solution



starting solution

PR example

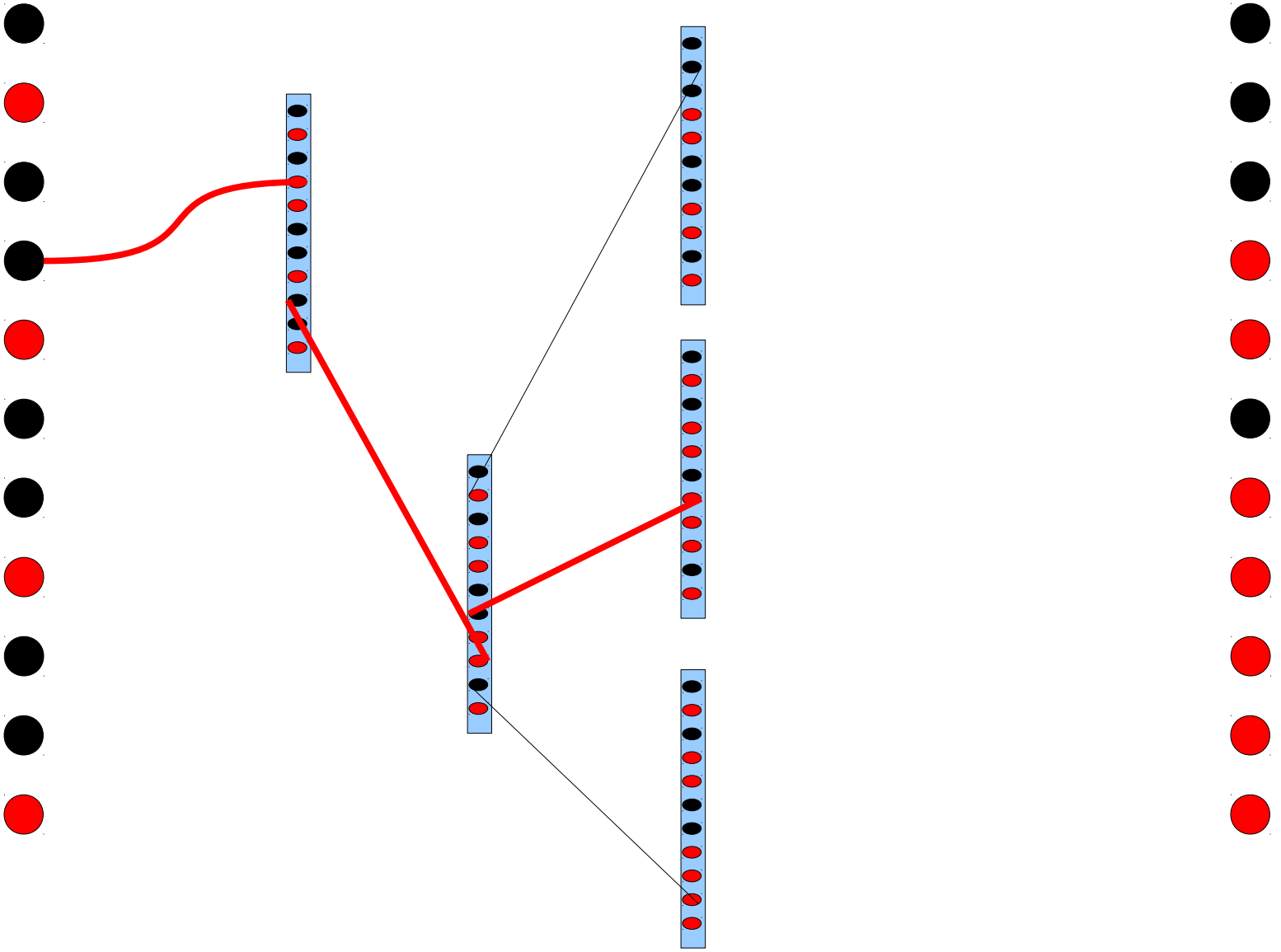
guiding solution



starting solution

PR example

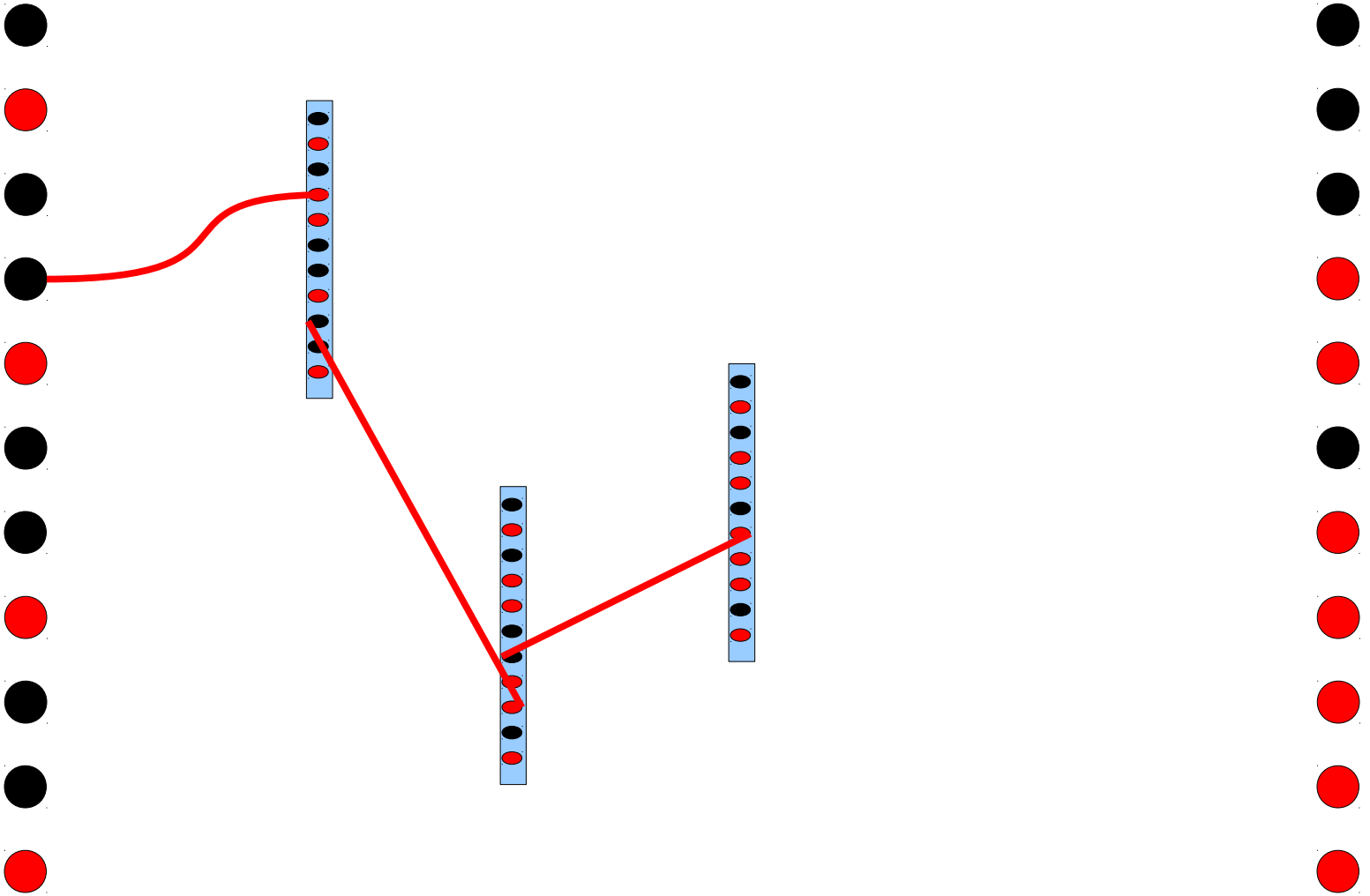
guiding solution



starting solution

PR example

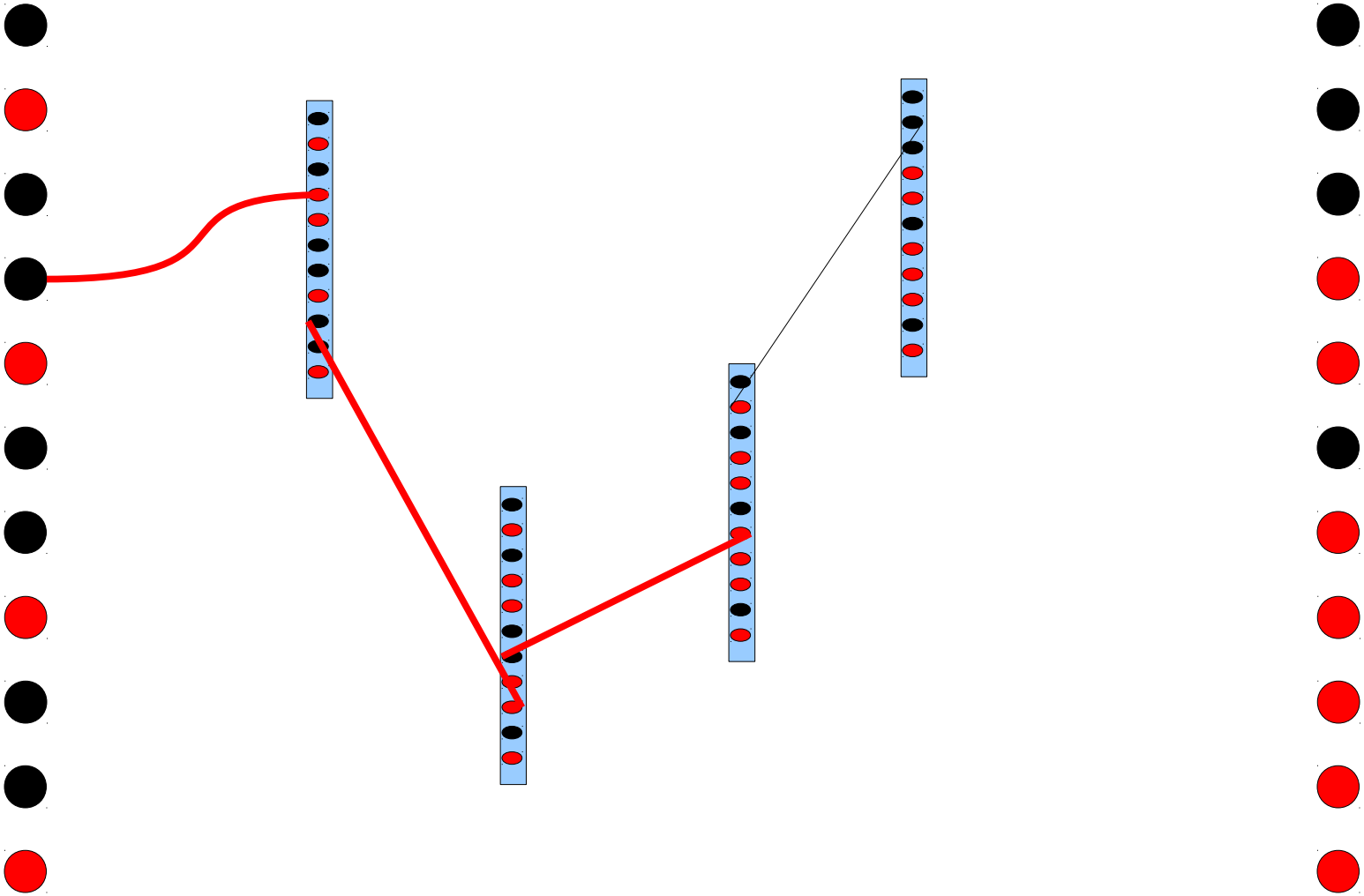
guiding solution



starting solution

PR example

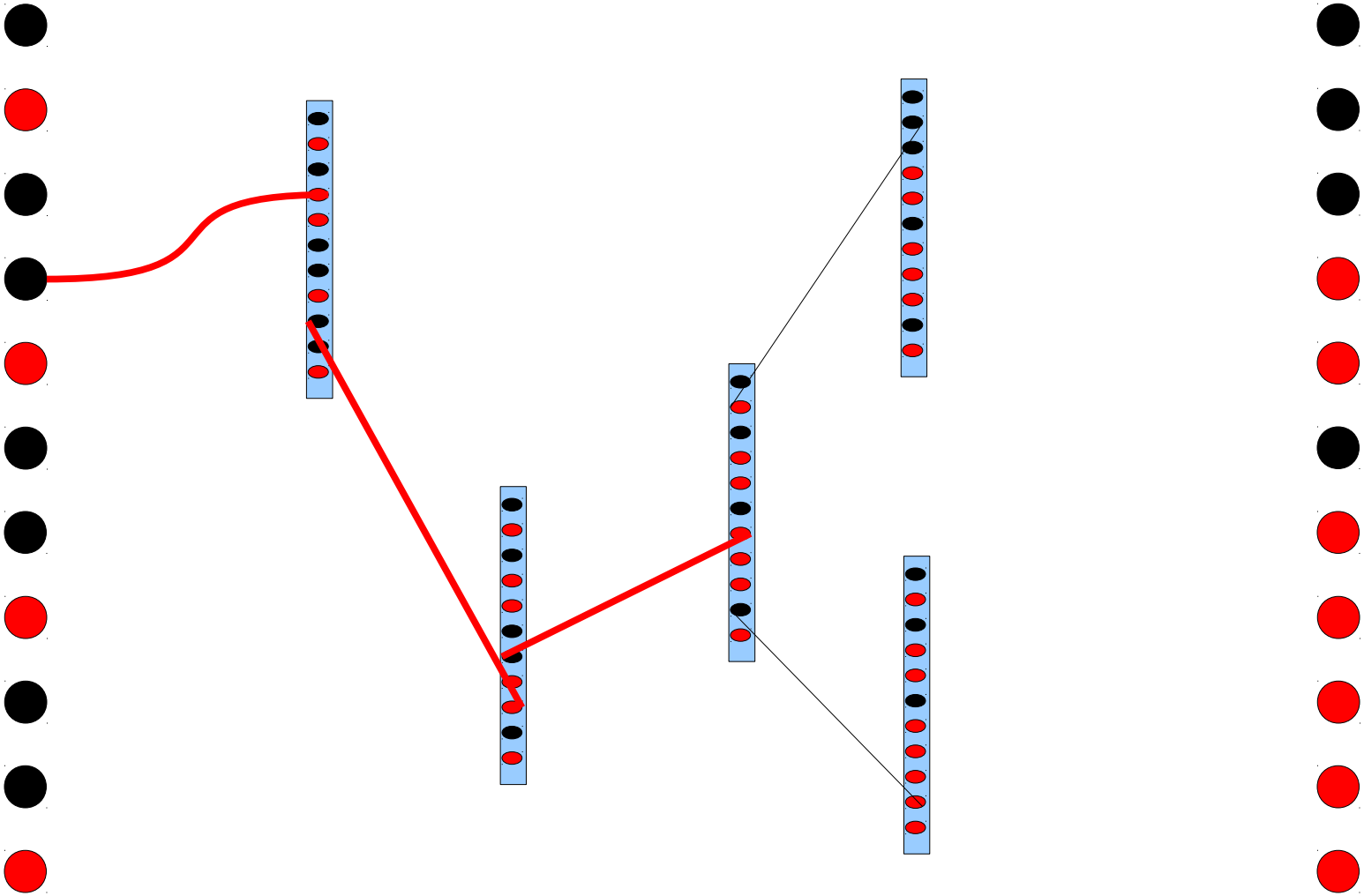
guiding solution



starting solution

PR example

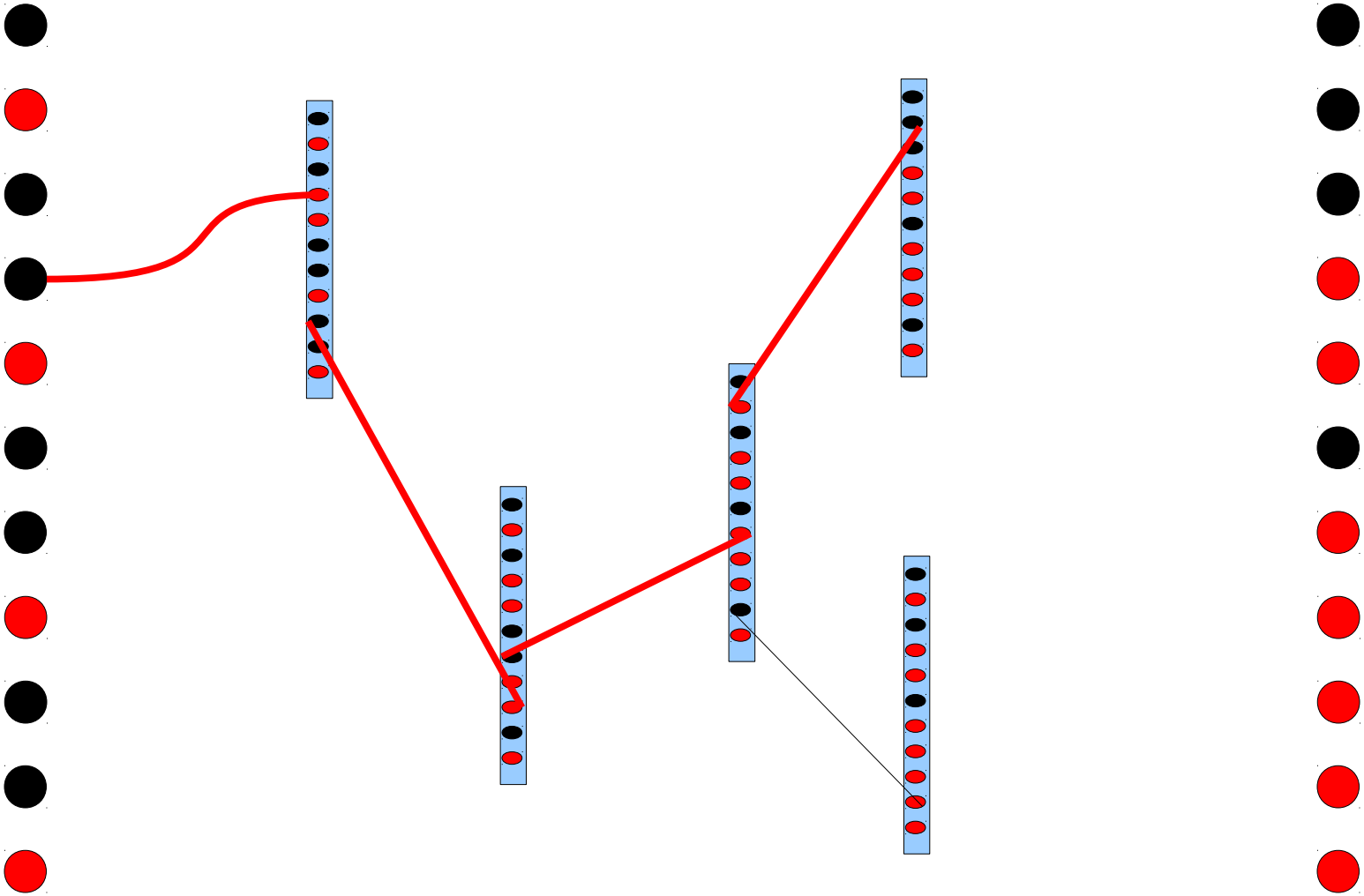
guiding solution



starting solution

PR example

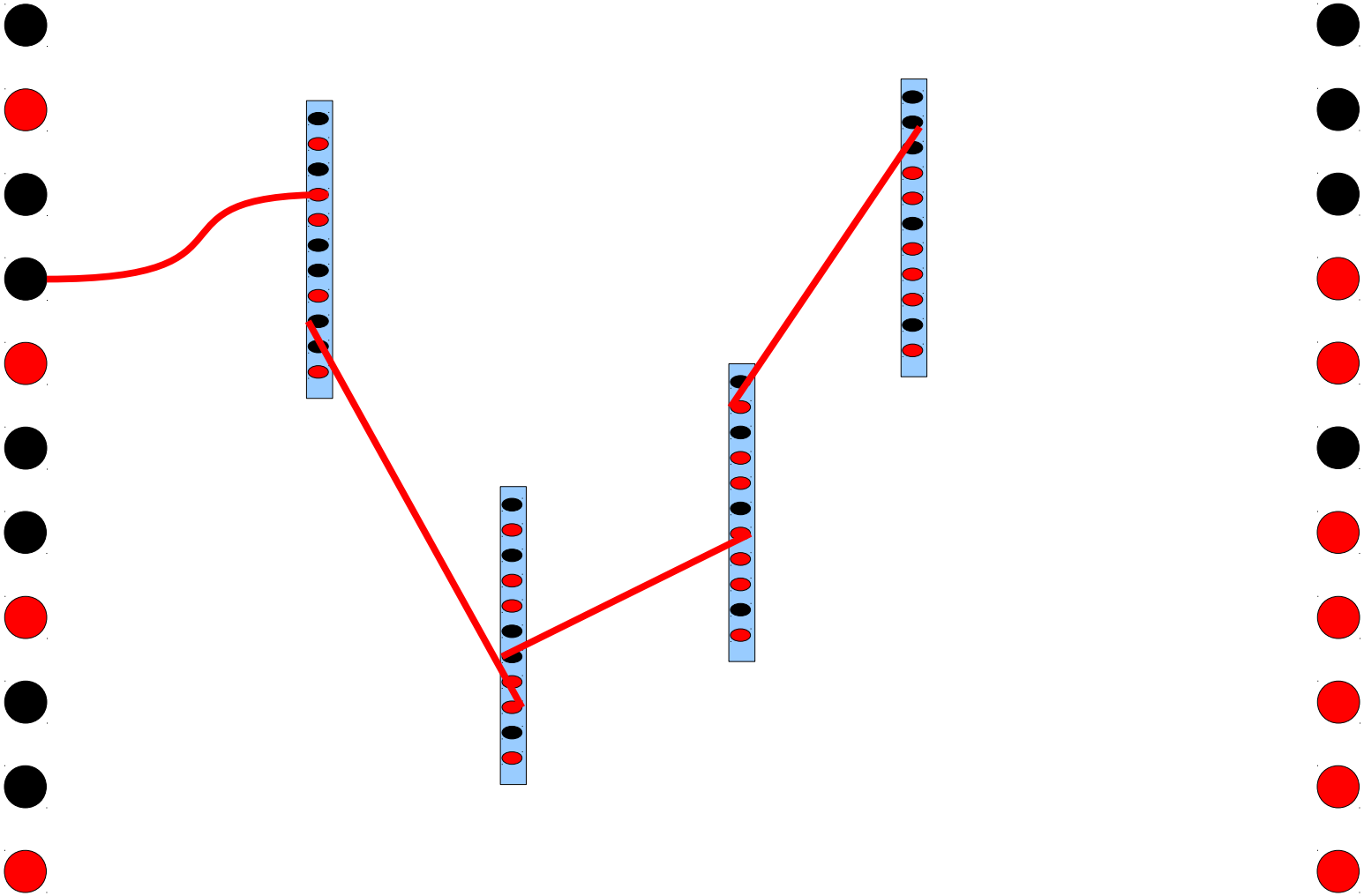
guiding solution

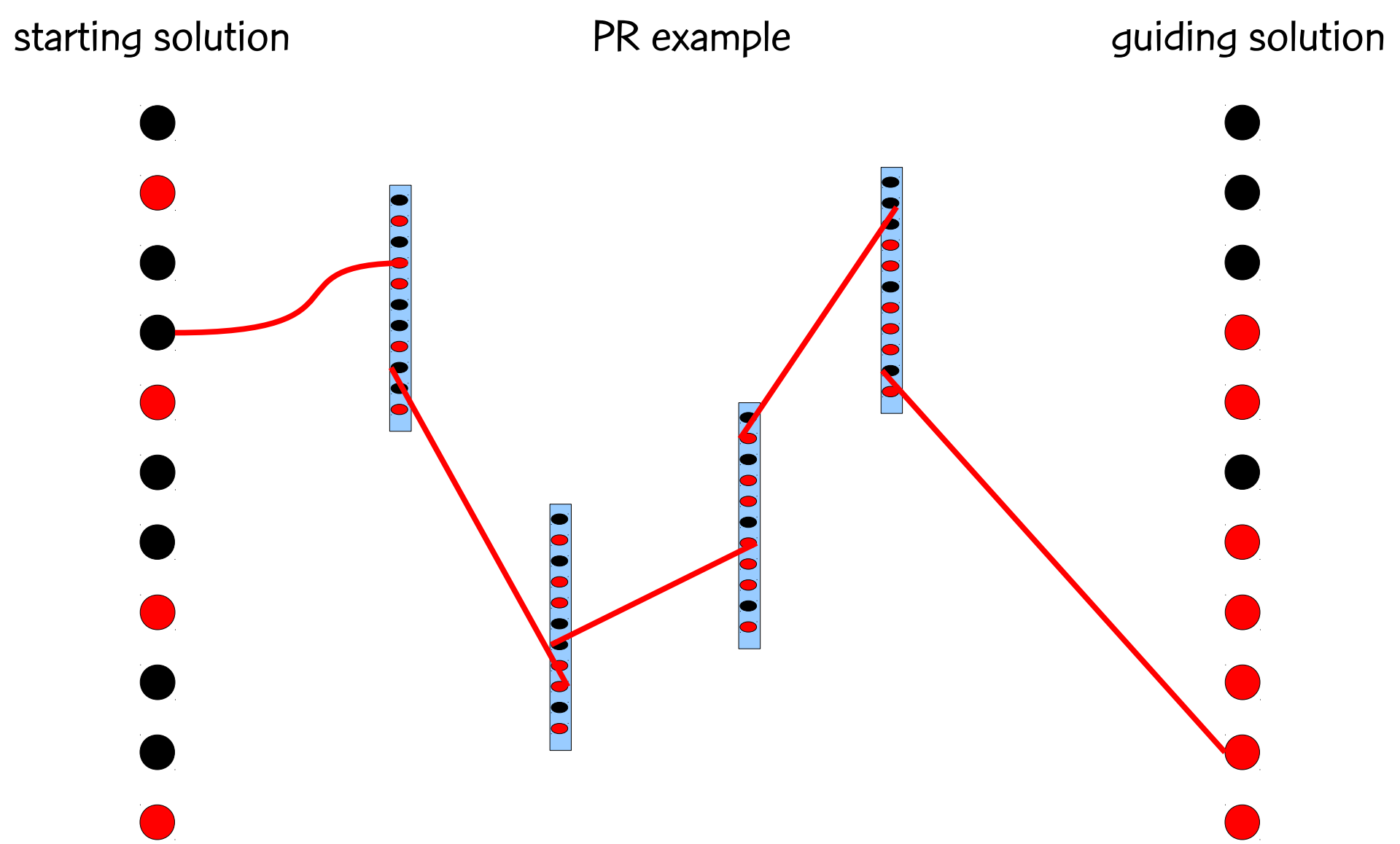


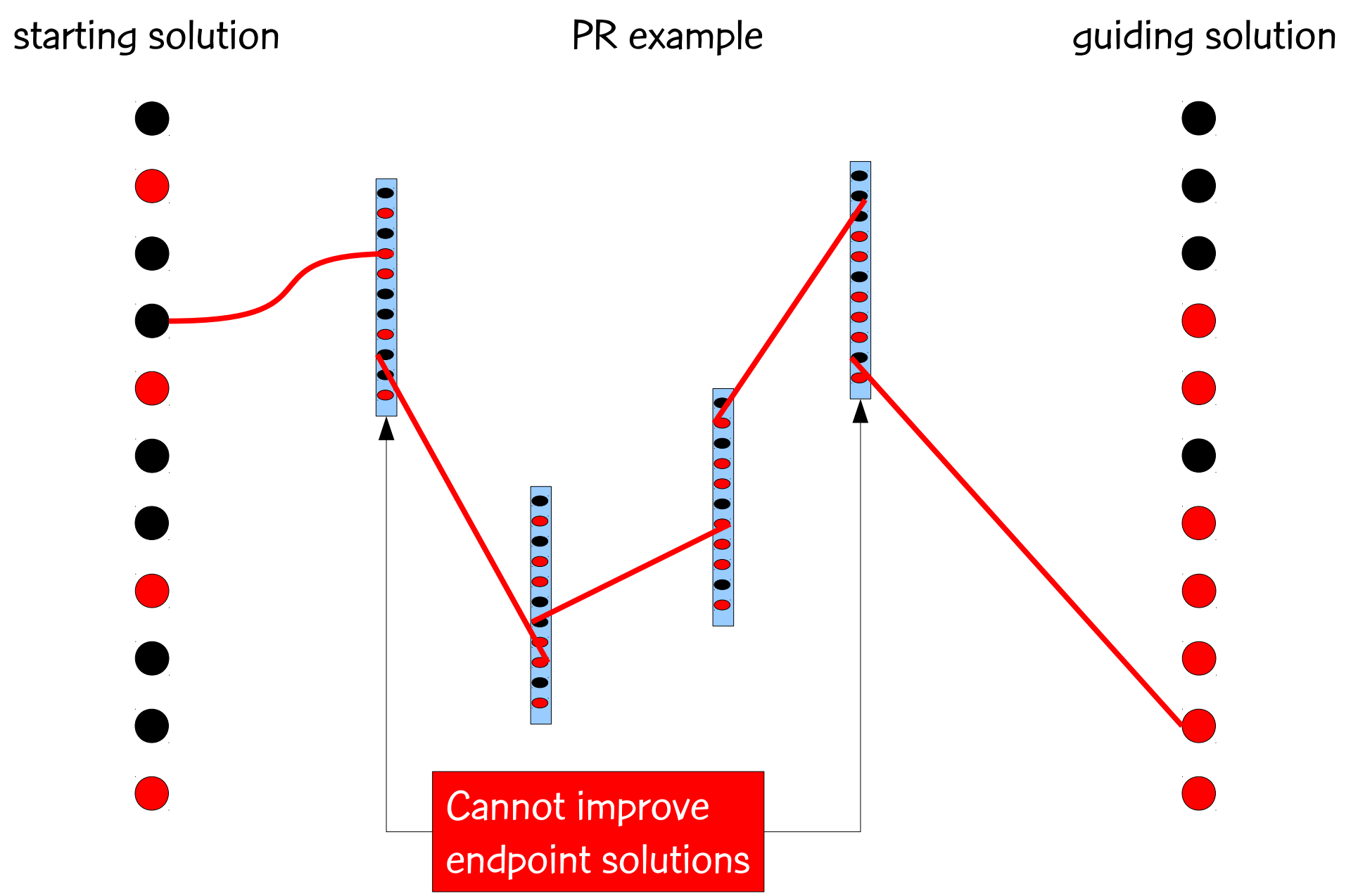
starting solution

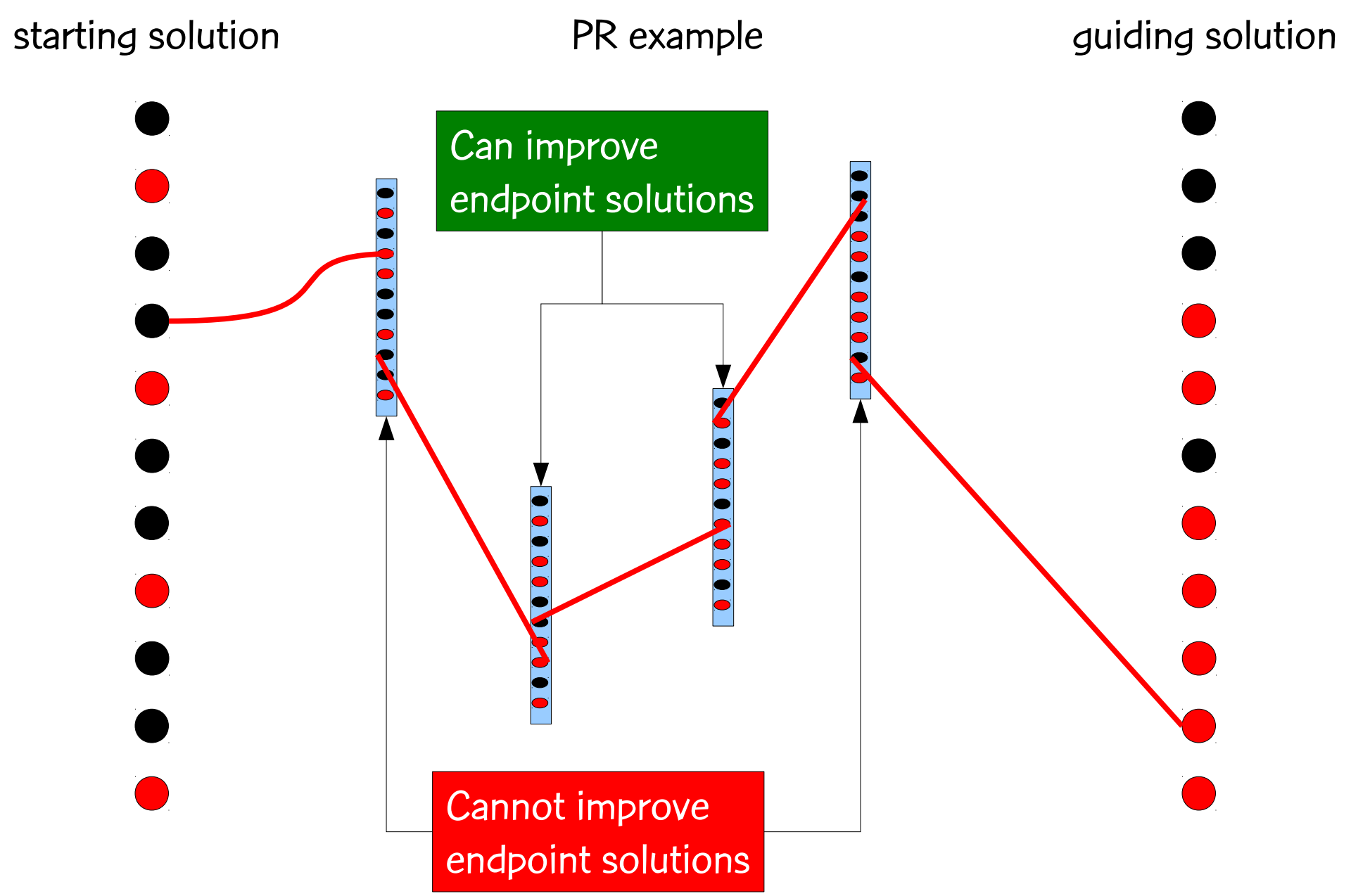
PR example

guiding solution




starting solution
PR example
guiding solution

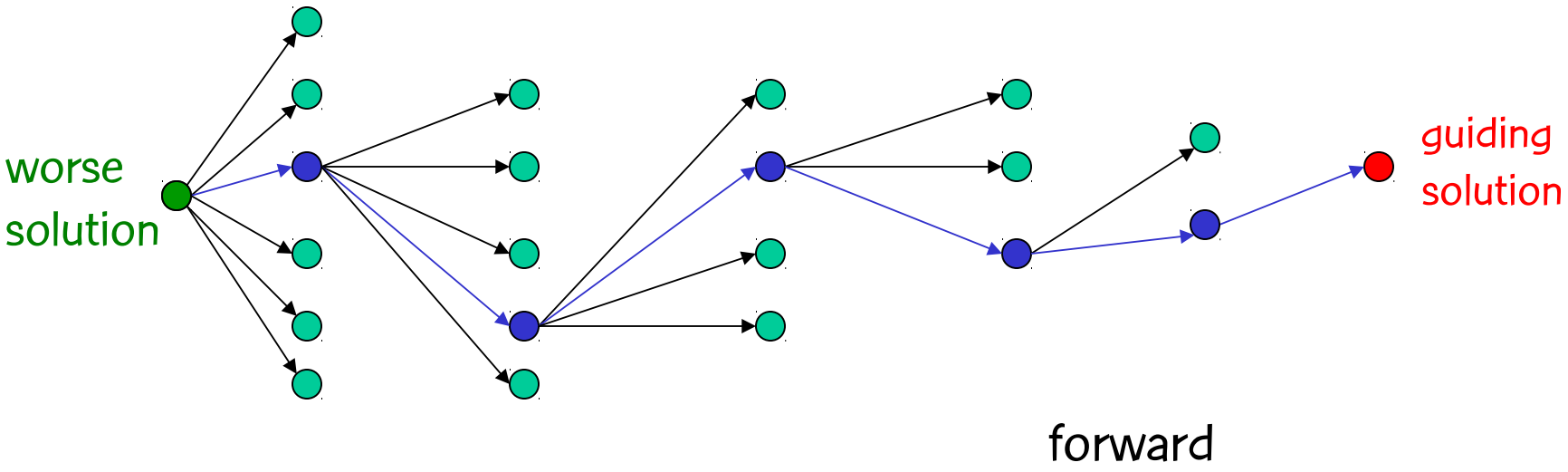

starting solution
PR example
guiding solution
Cannot improve
endpoint solutions


starting solution
PR example
guiding solution
Can improve endpoint solutions
Cannot improve endpoint solutions

Forward path-relinking

Variants: trade-offs between computation time and solution quality

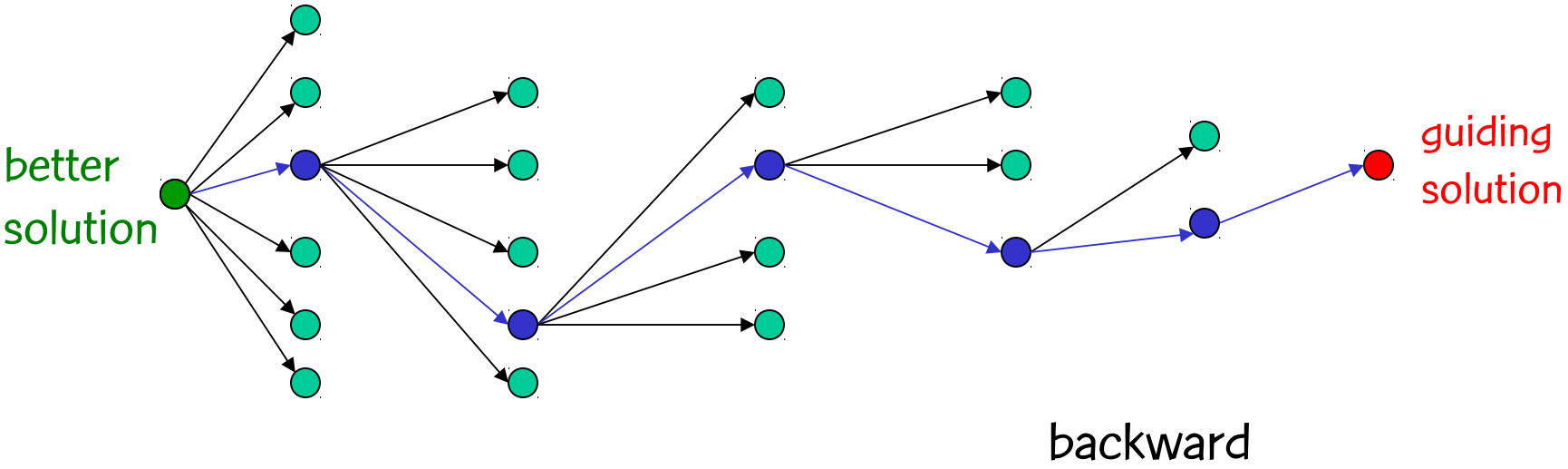
Forward PR adopts as initial solution the worse of the two input solutions and uses the better solution as the guide.



Backward path-relinking

Variants: trade-offs between computation time and solution quality

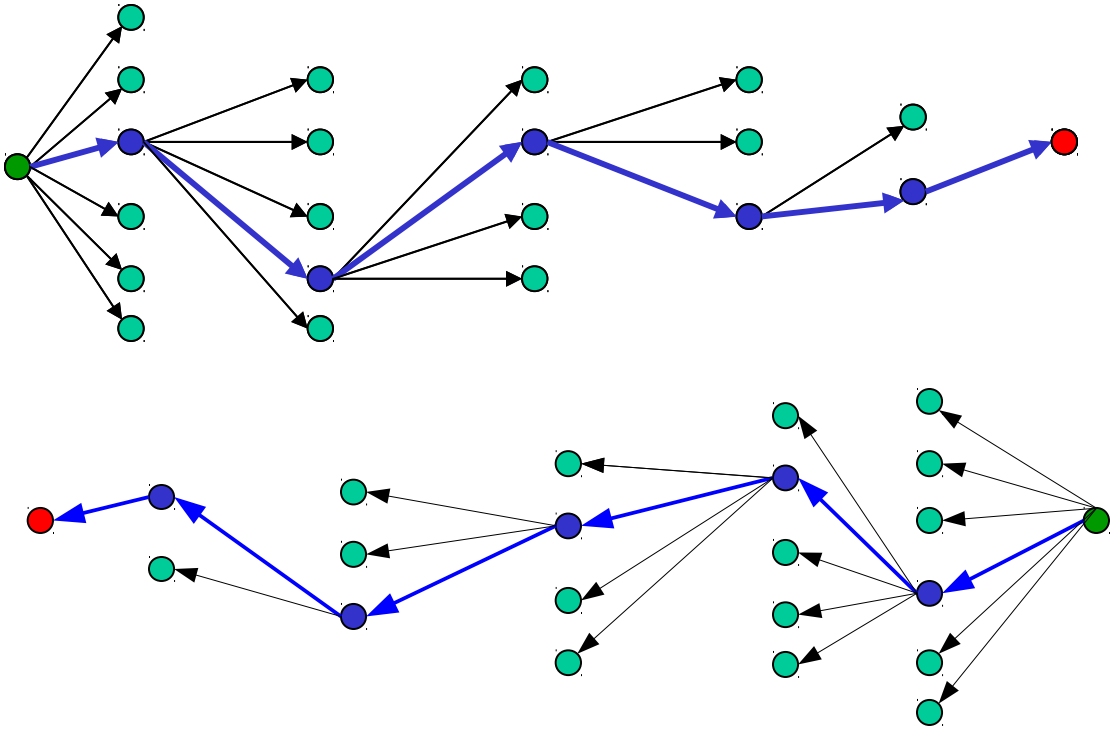
Backward PR usually does better: Better to start from the best of the two input solutions, neighborhood of the initial solution is explored more than of the guide!



Back and forth path-relinking

Variants: trade-offs between computation time and solution quality

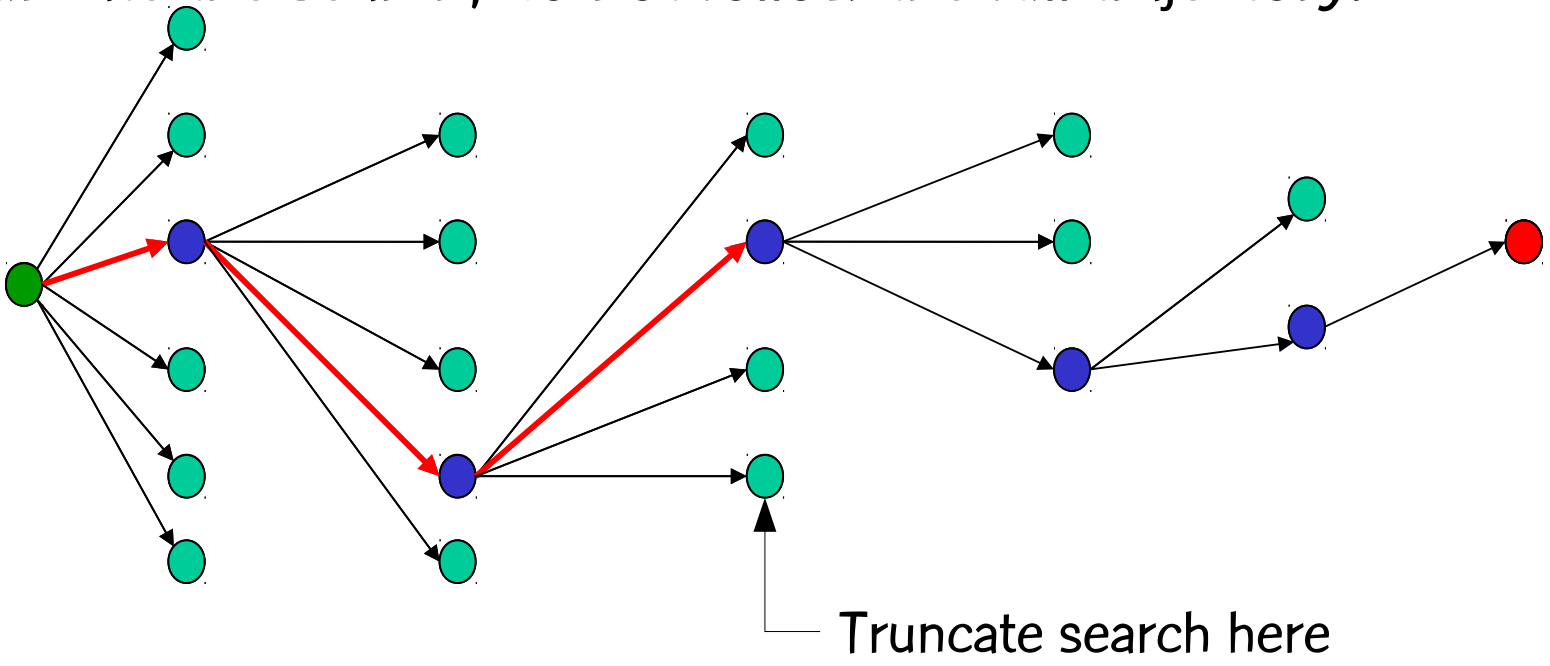
Explore both trajectories: twice as much time, often with only marginal improvements!



Truncated path-relinking

Variants: trade-offs between computation time and solution quality

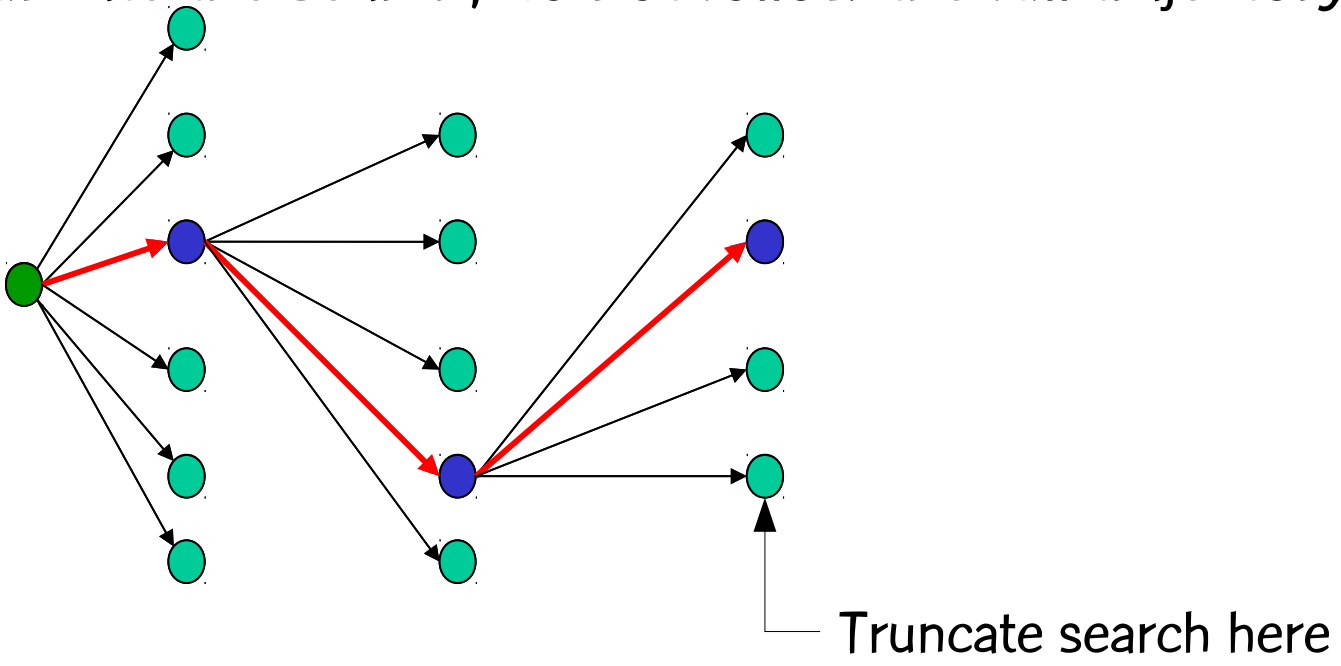
Truncate the search, do not follow the full trajectory.



Truncated path-relinking

Variants: trade-offs between computation time and solution quality

Truncate the search, do not follow the full trajectory.



Mixed path-relinking

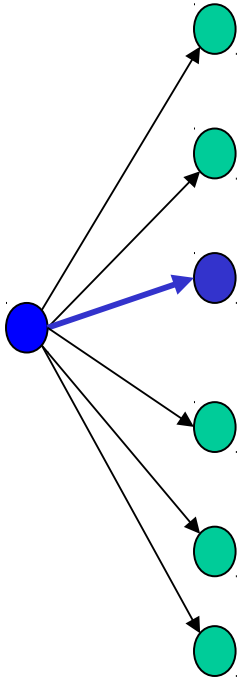
Variants: trade-offs between computation time and solution quality

Mixed path-relinking (Glover, 1997; Rosseti, 2003)

Mixed path-relinking

Variants: trade-offs between computation time and solution quality

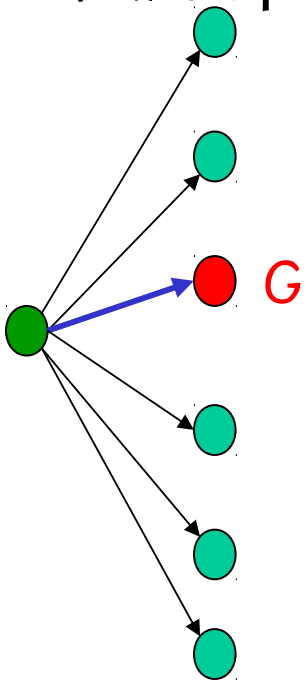
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

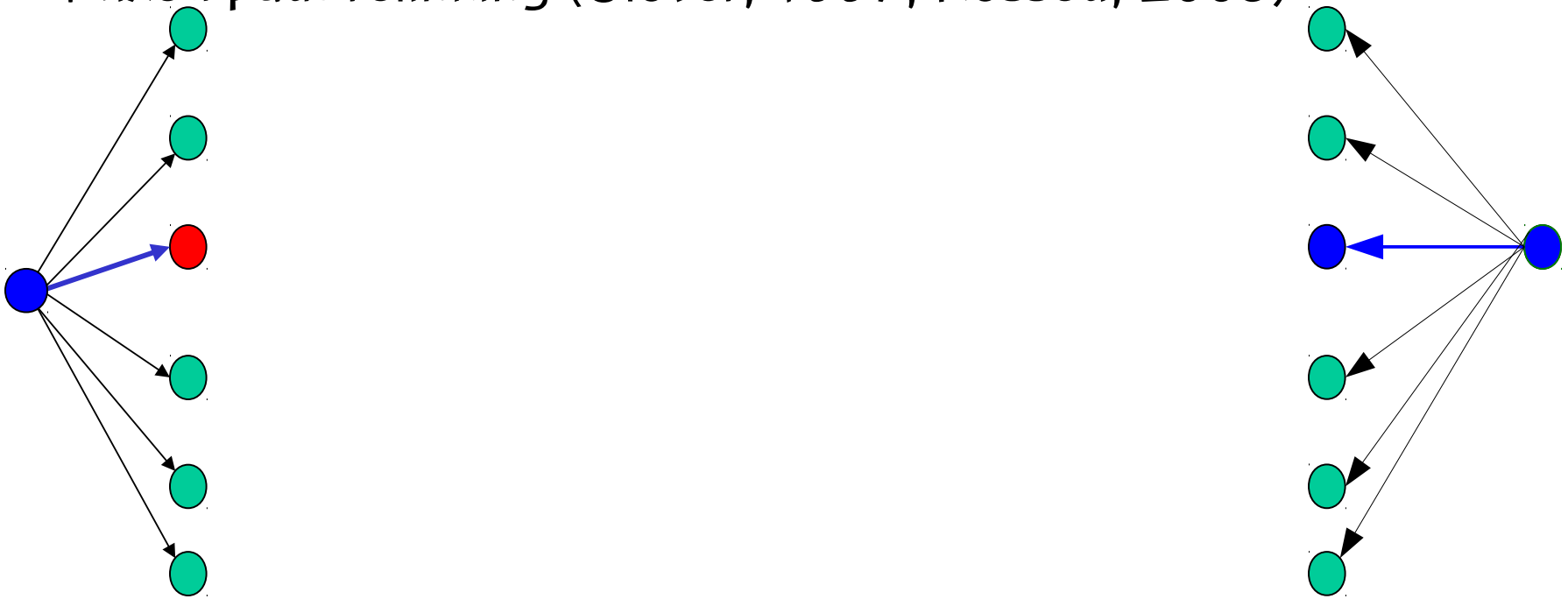
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

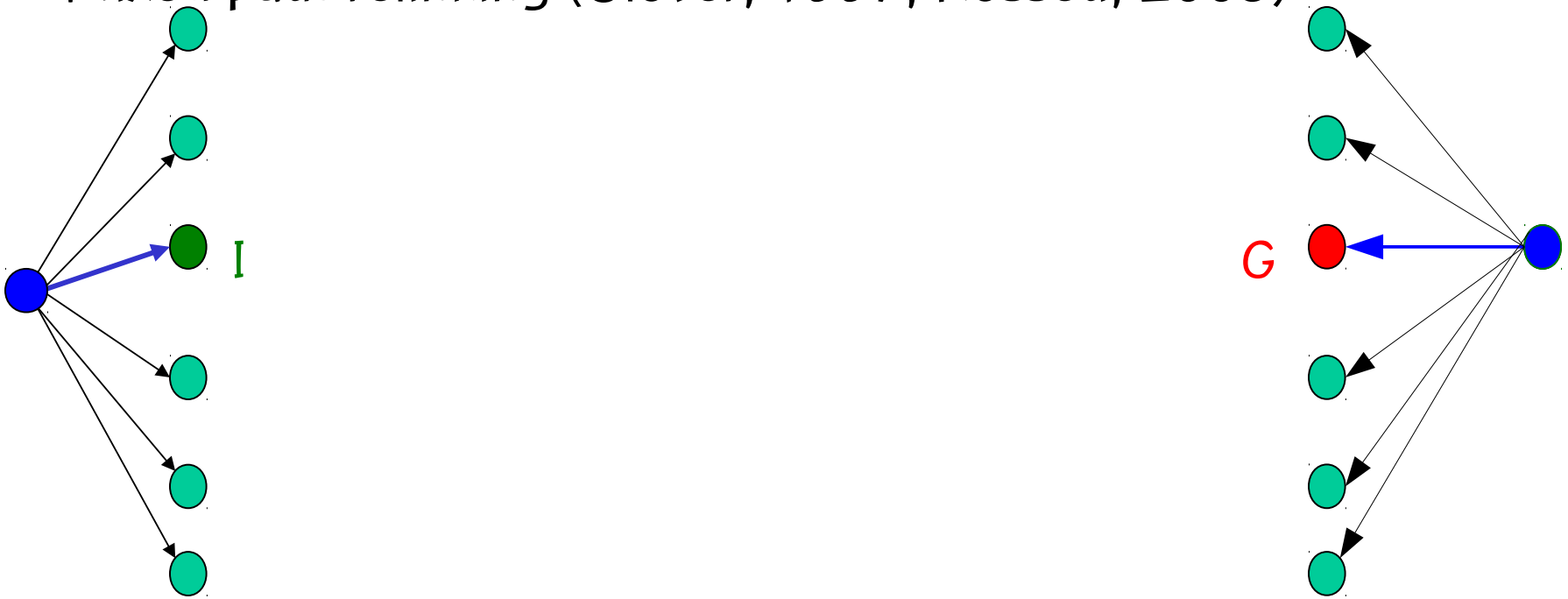
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

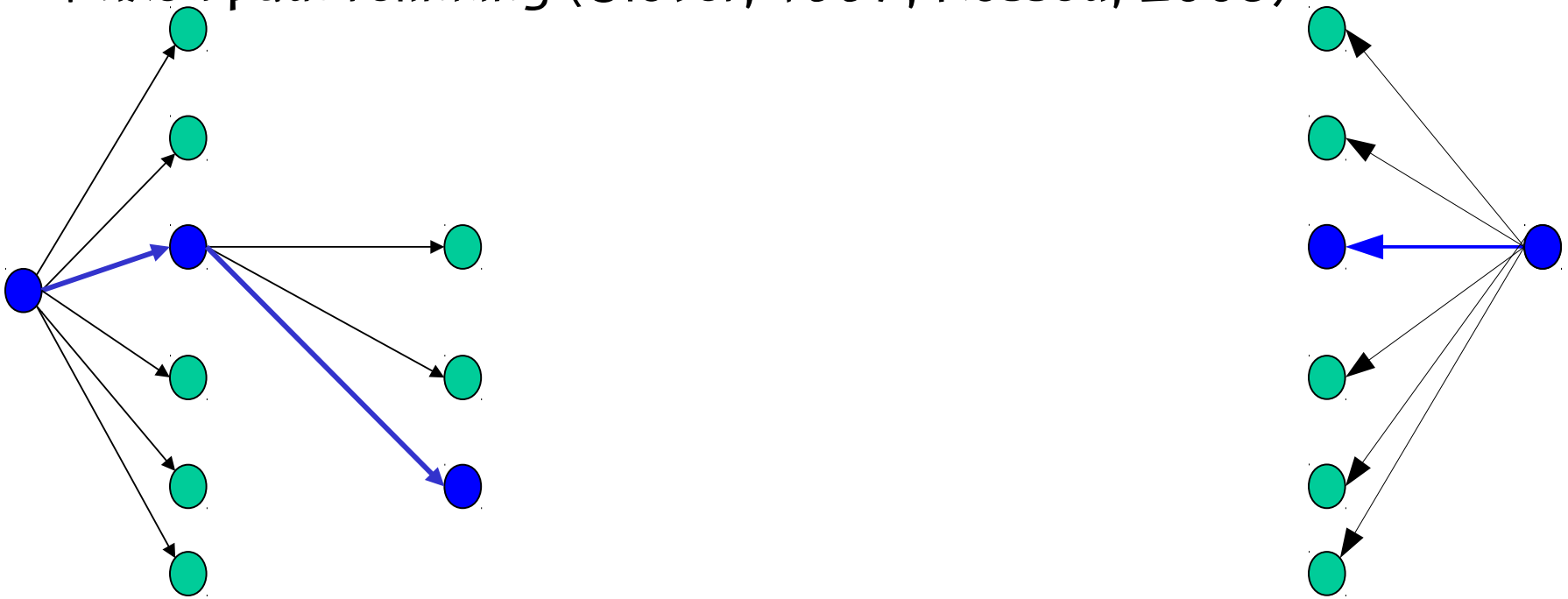
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

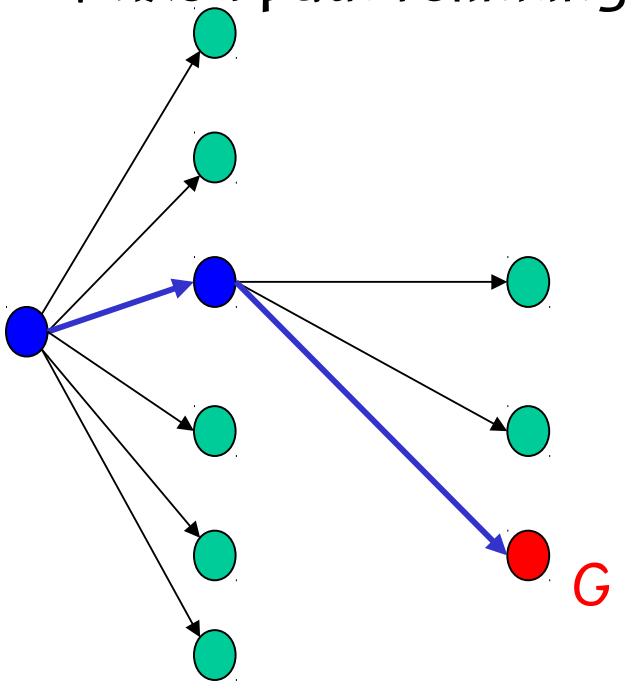
Mixed path-relinking (Glover, 1997; Rosseti, 2003)

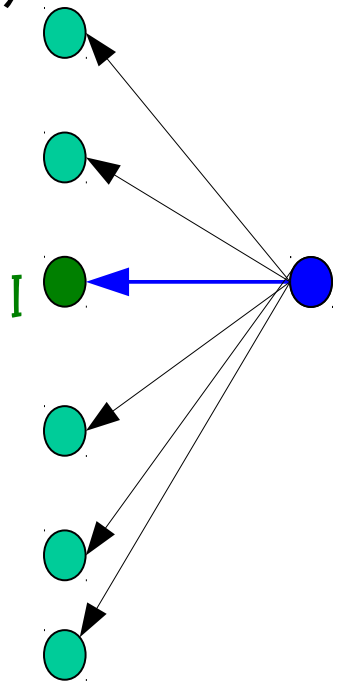


Mixed path-relinking

Variants: trade-offs between computation time and solution quality

Mixed path-relinking (Glover, 1997; Rosseti, 2003)

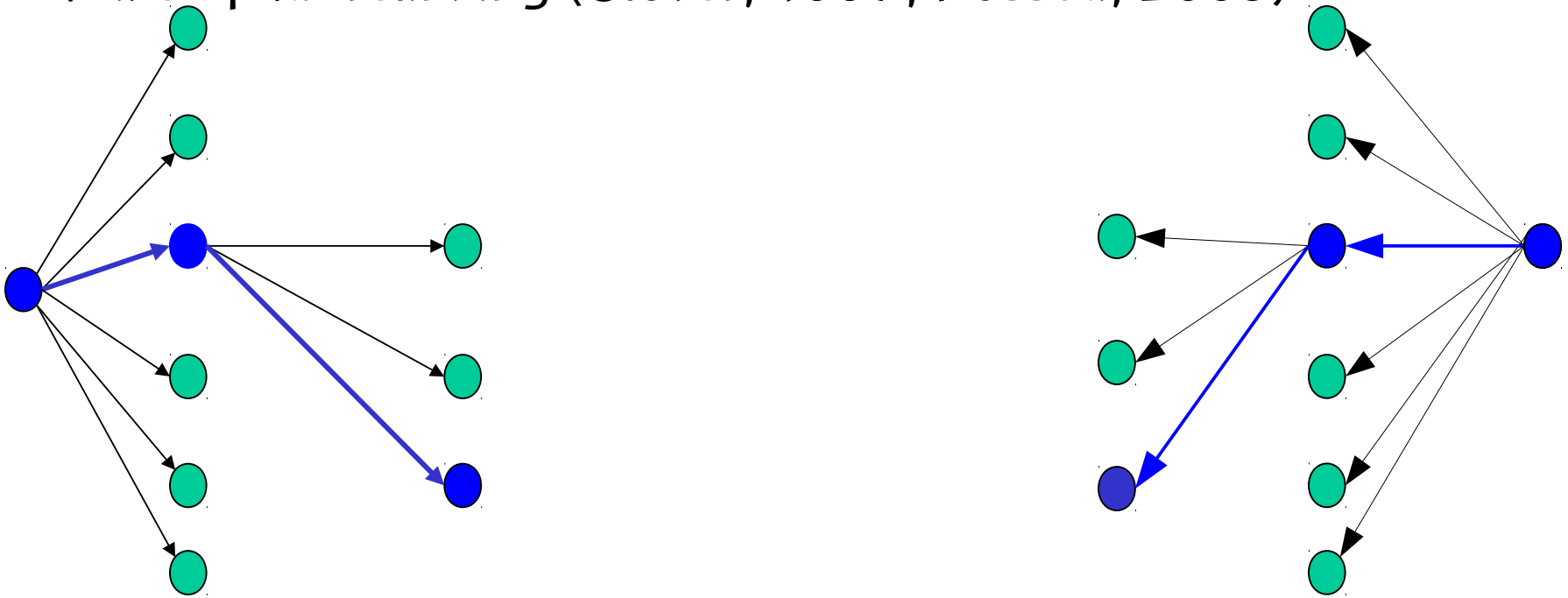




Mixed path-relinking

Variants: trade-offs between computation time and solution quality

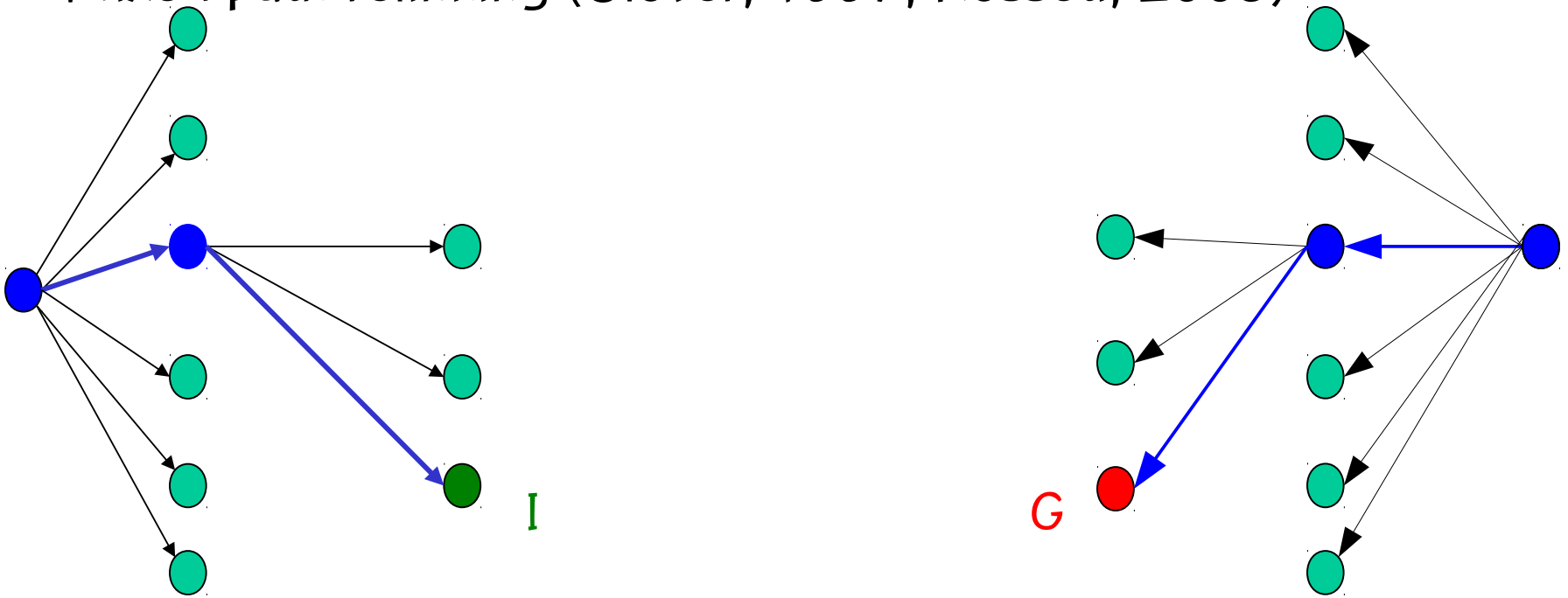
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

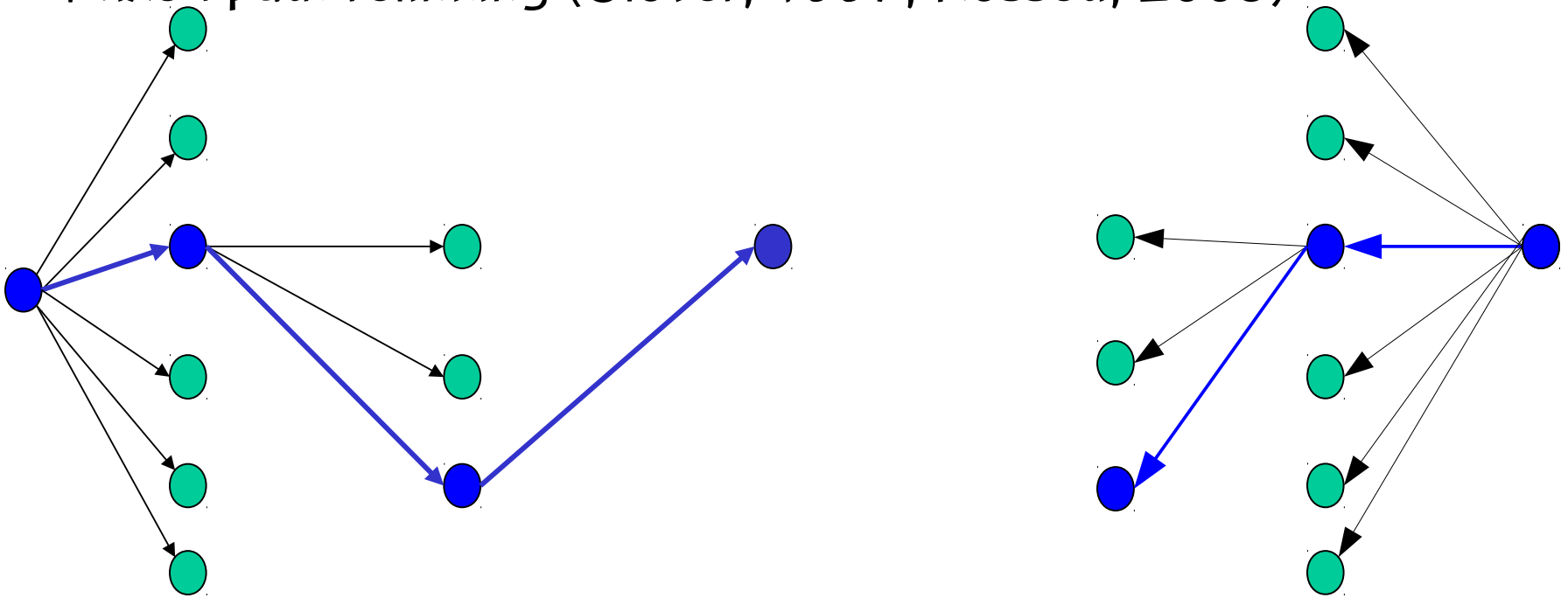
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

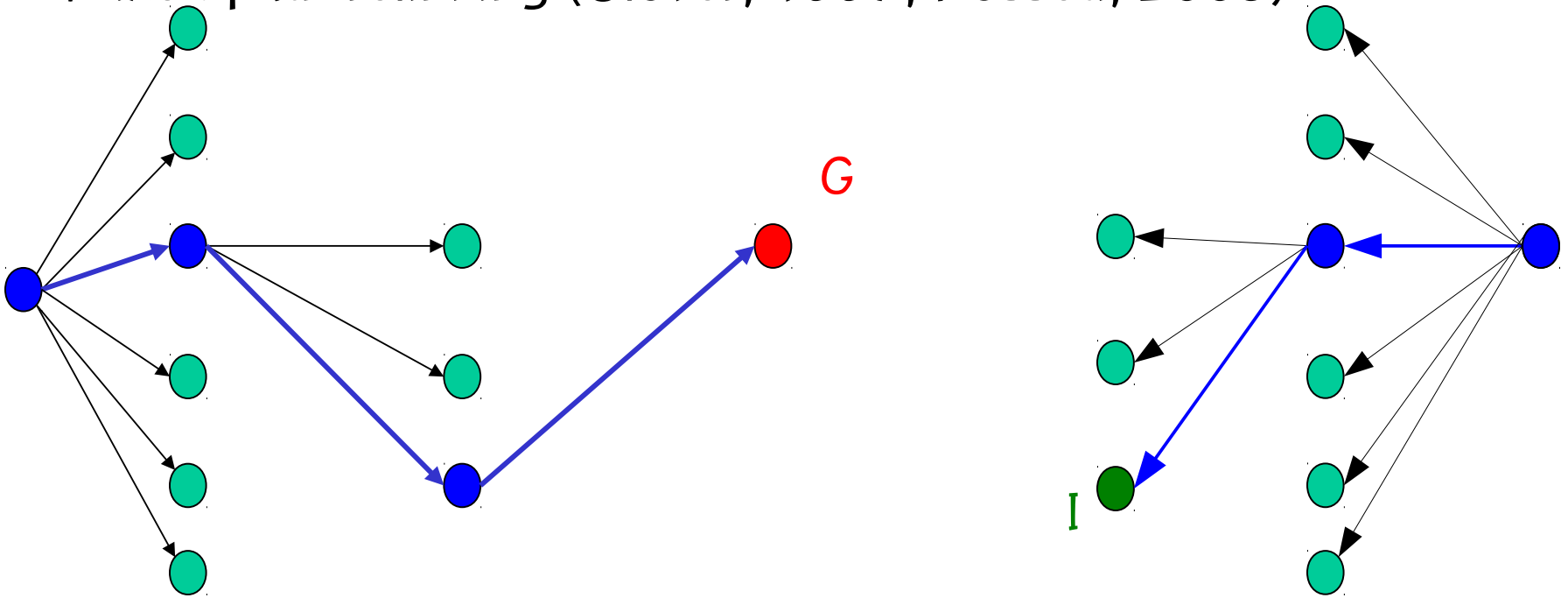
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

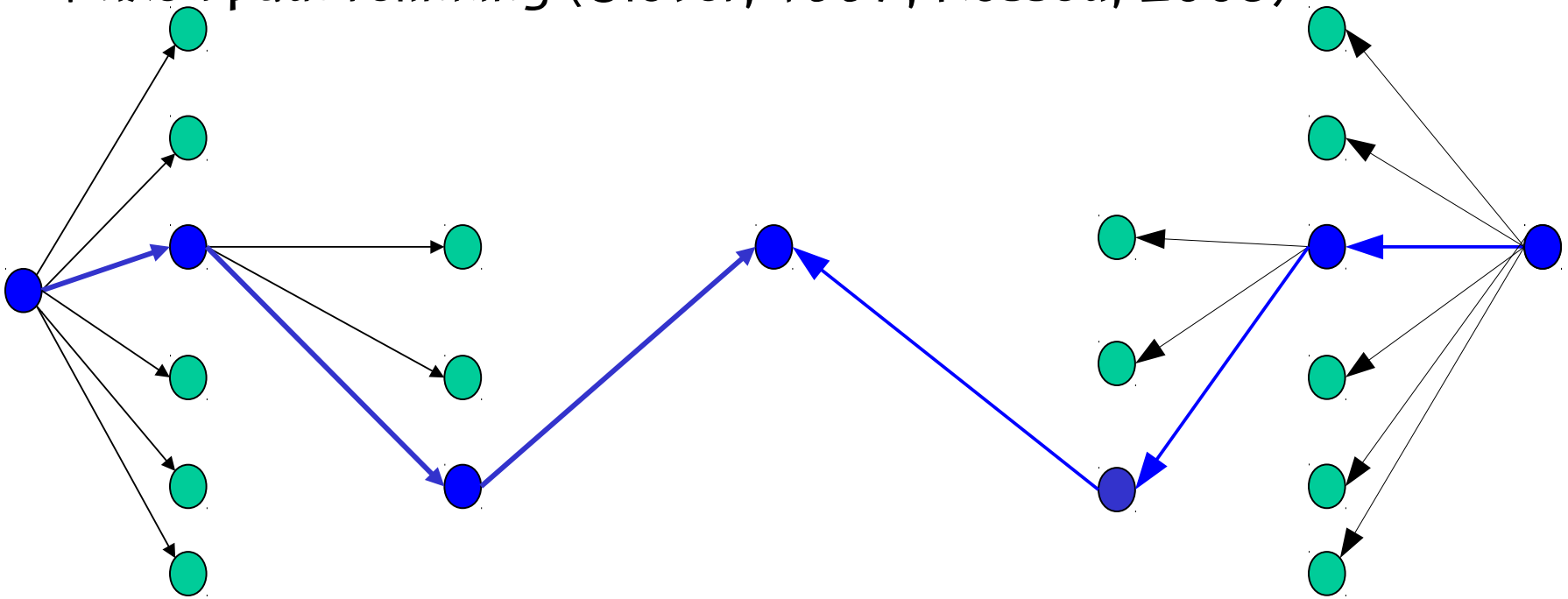
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

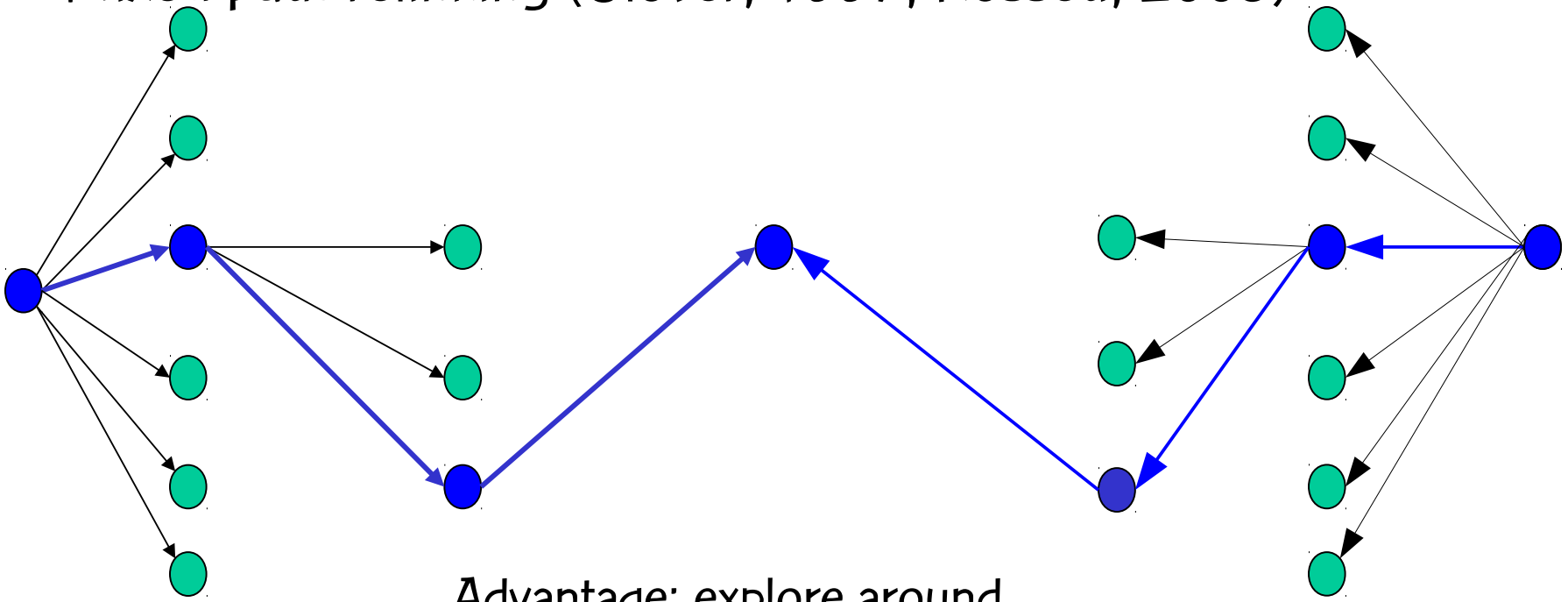
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

Mixed path-relinking (Glover, 1997; Rosseti, 2003)

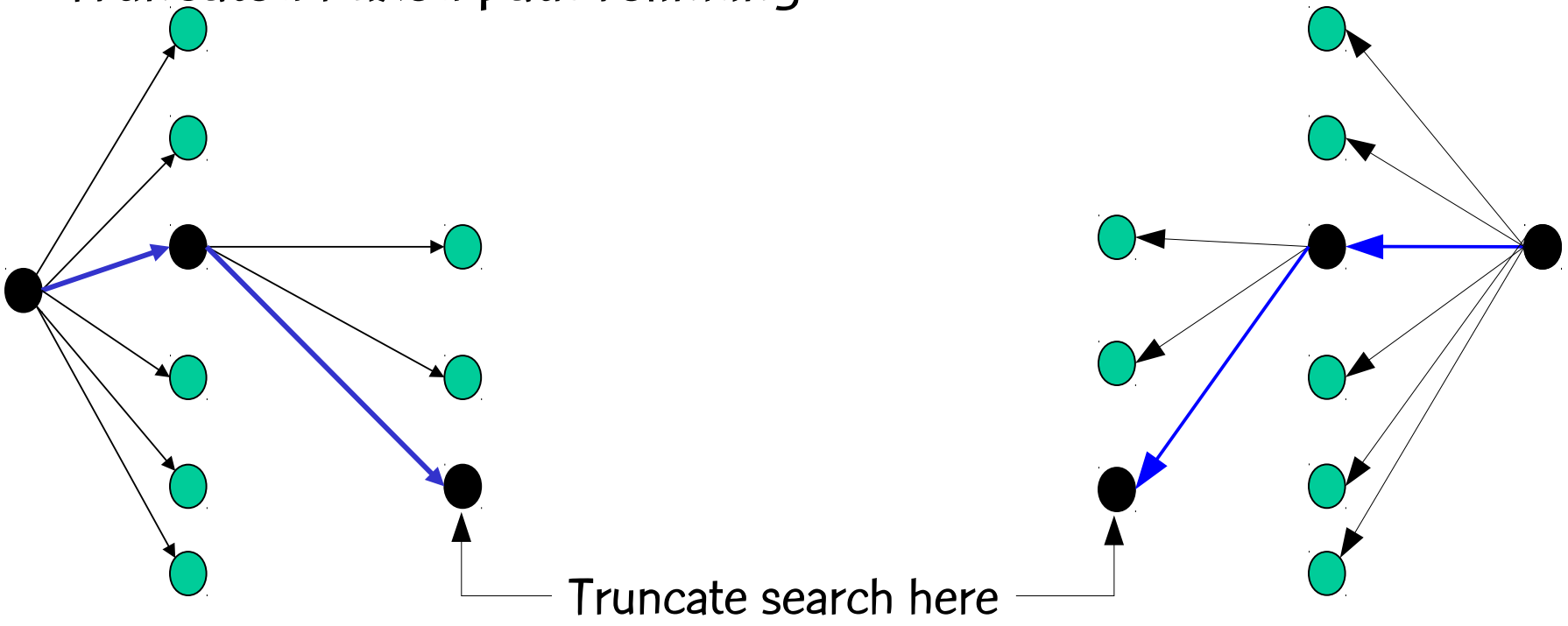


Advantage: explore around neighborhoods of both input solutions.

Truncated mixed path-relinking

Variants: trade-offs between computation time and solution quality

Truncated mixed path-relinking

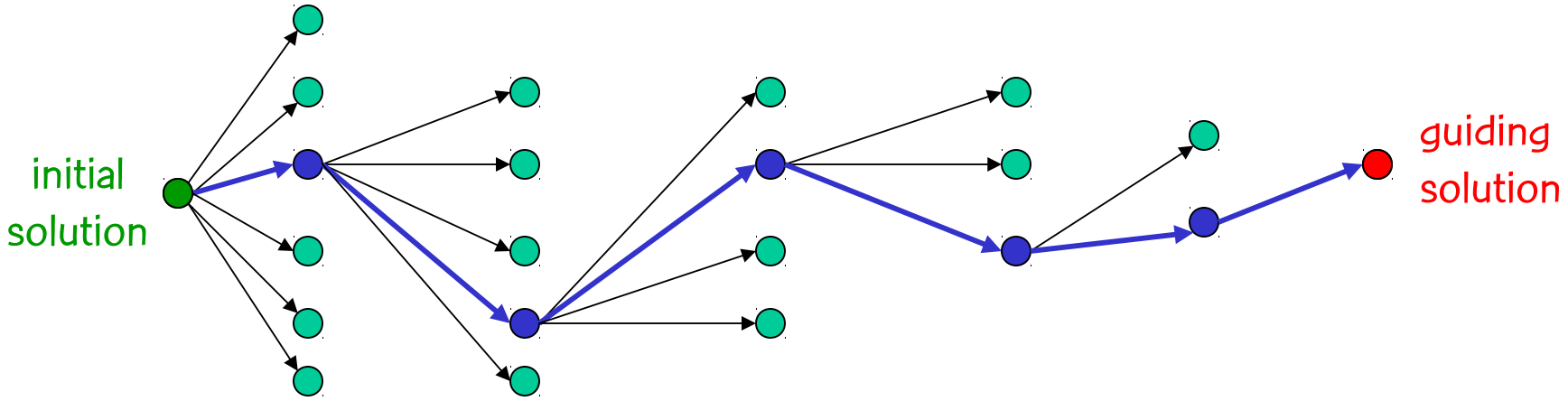


Greedy randomized adaptive path-relinking

Faria, Binato, R., & Falcão (2001, 2005)

Incorporates semi-greediness into PR.

Standard PR selects moves greedily: samples one of exponentially many paths

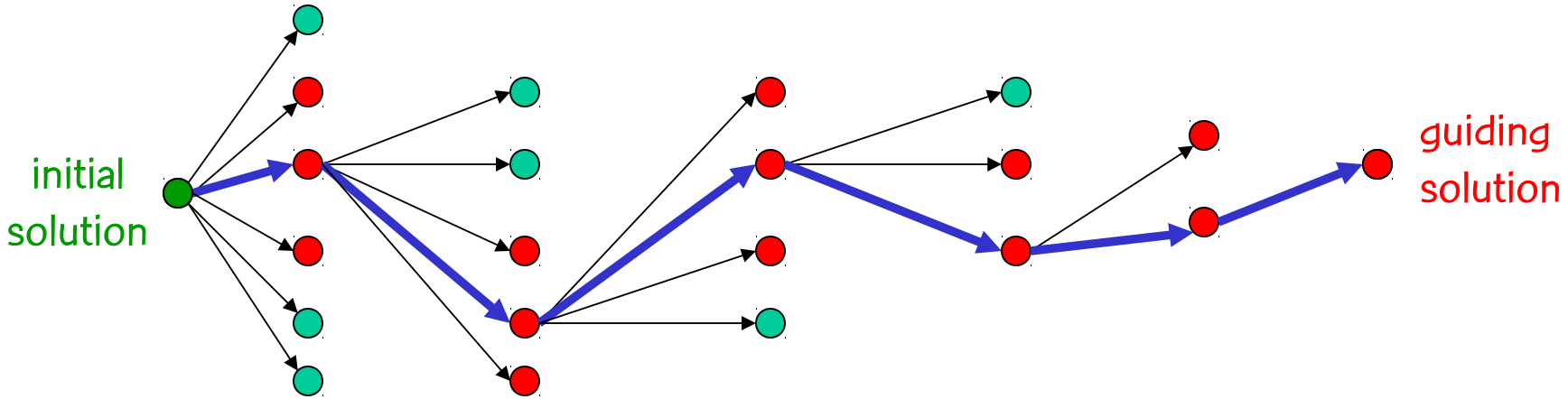


Greedy randomized adaptive path-relinking

Faria, Binato, R., & Falcão (2001, 2005)

Incorporates semi-greediness into PR.

graPR creates RCL with best moves: samples several paths

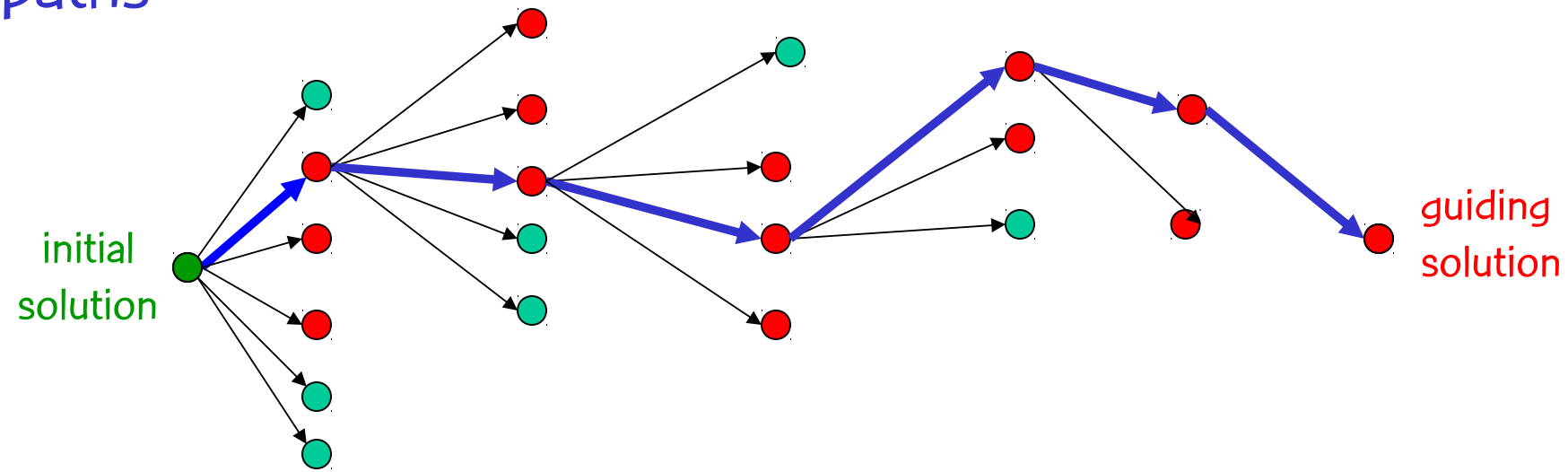


Greedy randomized adaptive path-relinking

Faria, Binato, R., & Falcão (2001, 2005)

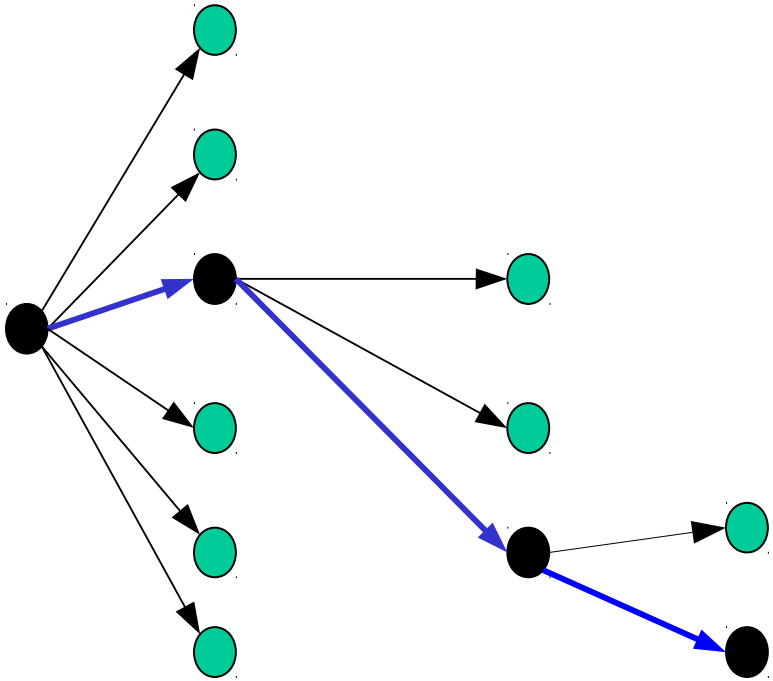
Incorporates semi-greediness into PR.

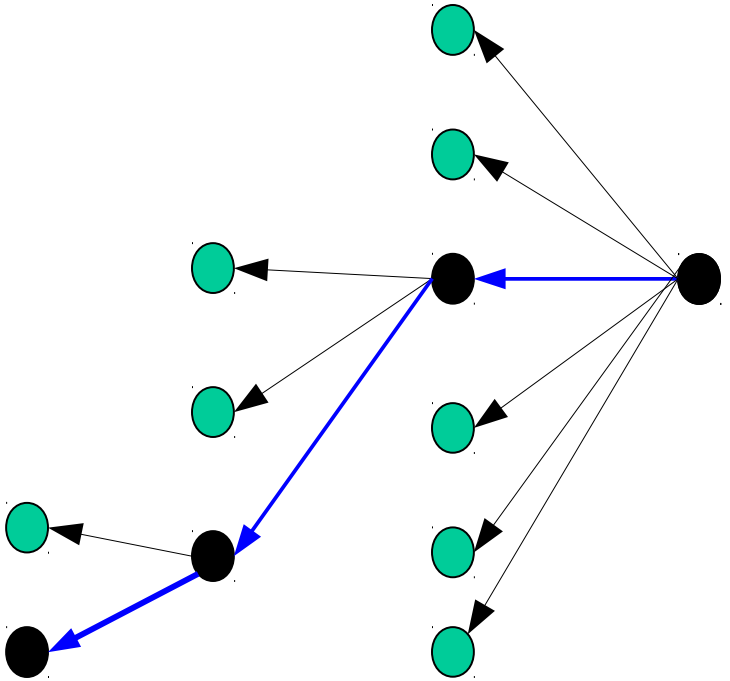
graPR creates RCL with best moves: samples several paths



Truncated mixed graPR

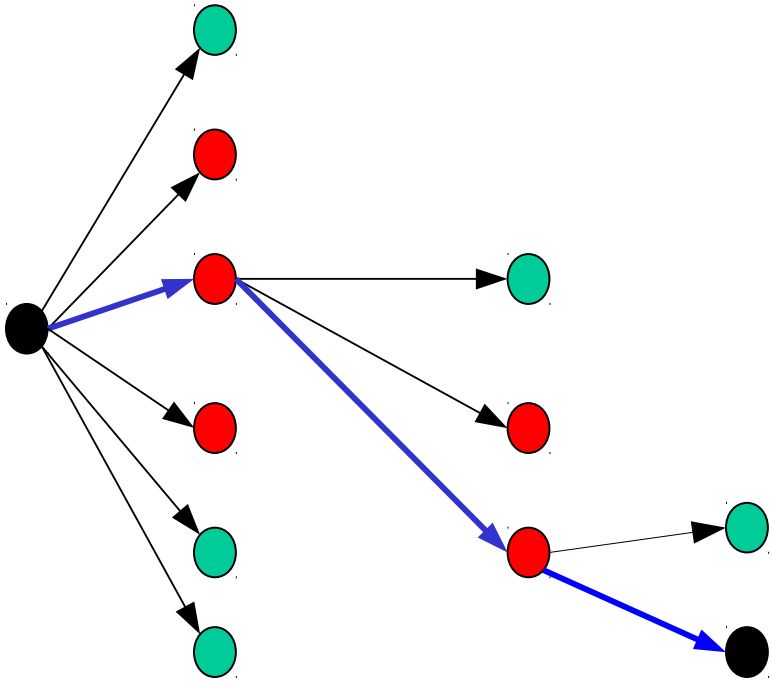
When applied to a given pair of solutions truncated mixed PR explores one of exponentially many path segments each time it is executed.

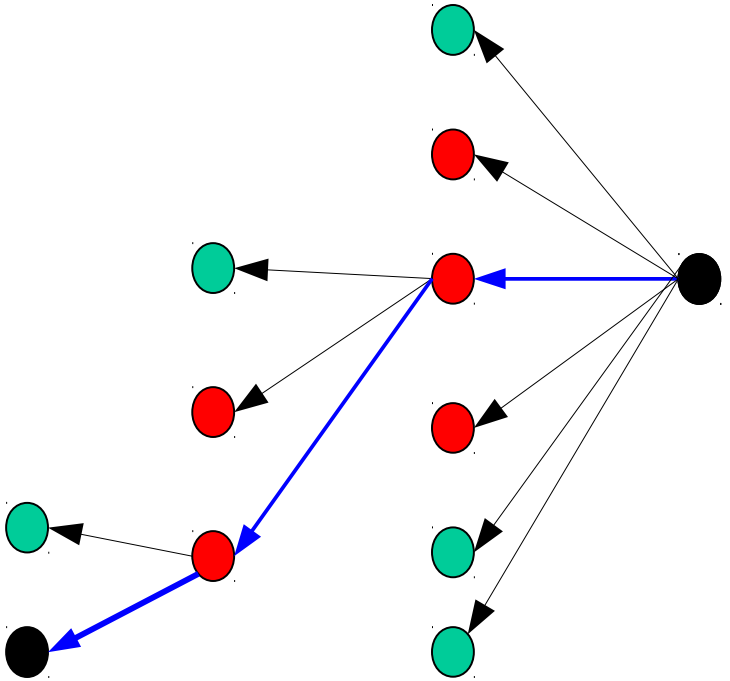




Truncated mixed graPR

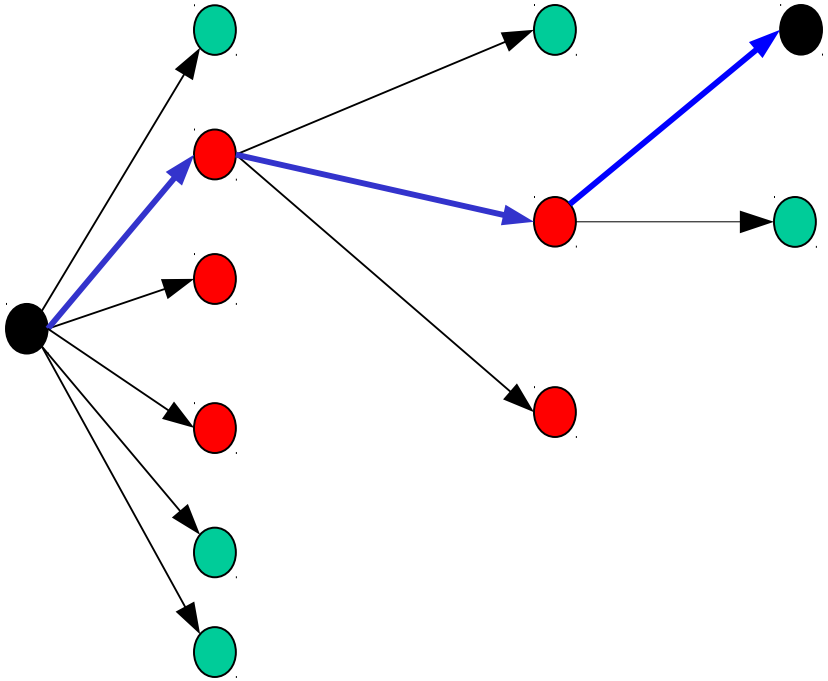
With high probability, truncated mixed graPR explores different path segments each time it is executed between the same pair of solutions.

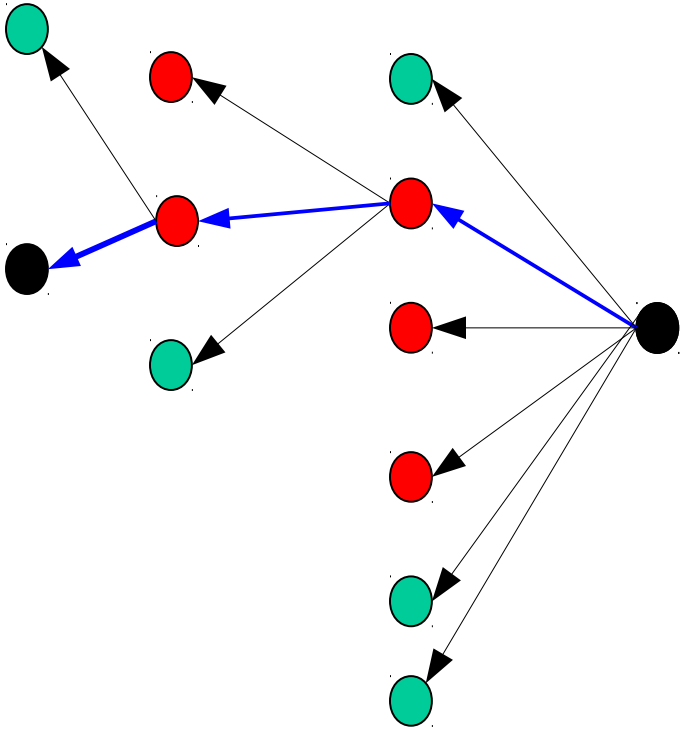




Truncated mixed graPR

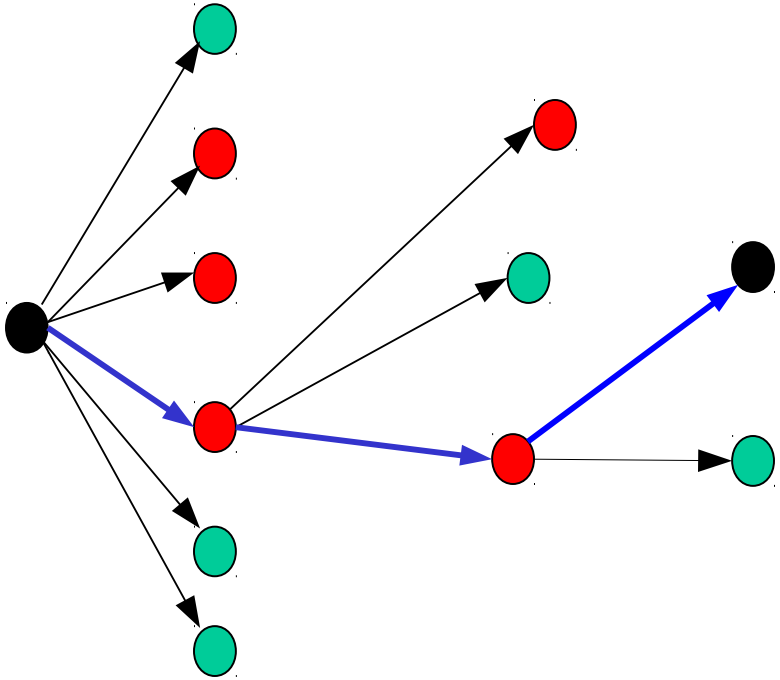
With high probability, truncated mixed graPR explores different path segments each time it is executed between the same pair of solutions.

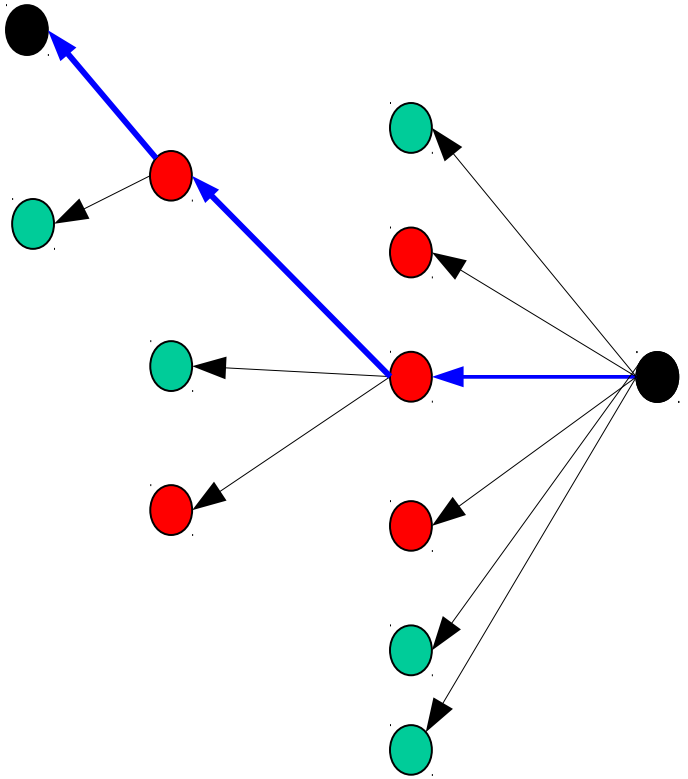




Truncated mixed graPR

With high probability, truncated mixed graPR explores different path segments each time it is executed between the same pair of solutions.





GRASP with path-relinking

GRASP with path-relinking

First proposed by Laguna and Martí (1999).

Maintains a set of elite solutions found during GRASP iterations.

After each GRASP iteration (construction and local search):

Use GRASP solution as initial solution.

Select an elite solution uniformly at random: guiding solution.

Perform path-relinking between these two solutions.

GRASP with path-relinking

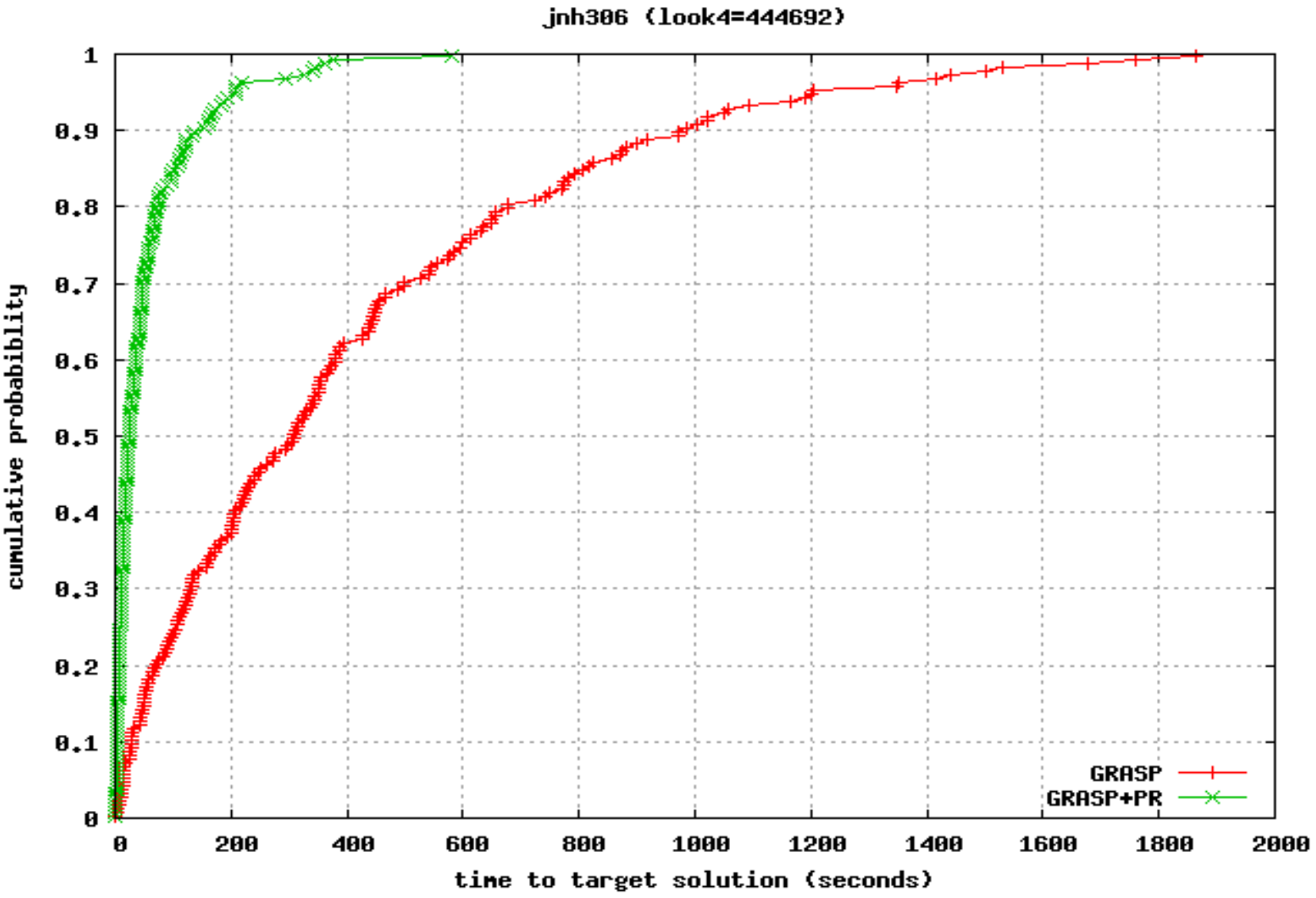
Since 1999, there has been a lot of activity in hybridizing GRASP with path-relinking.

Surveys by R. & Ribeiro (2005), R., Ribeiro, Glover & Martí (2010) & Ribeiro & R. (2012).

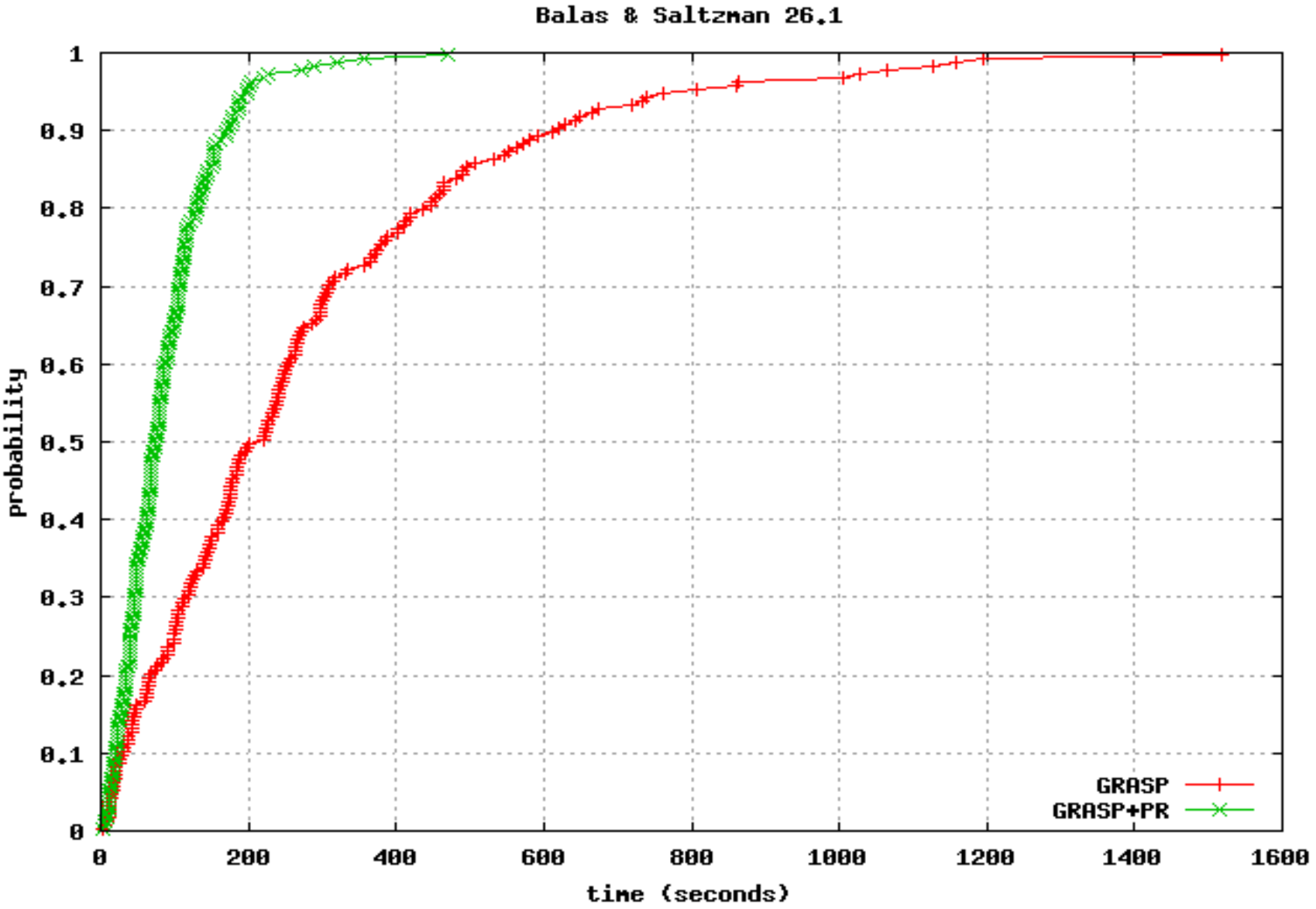
Main observation from experimental studies:

GRASP with path-relinking outperforms pure GRASP.

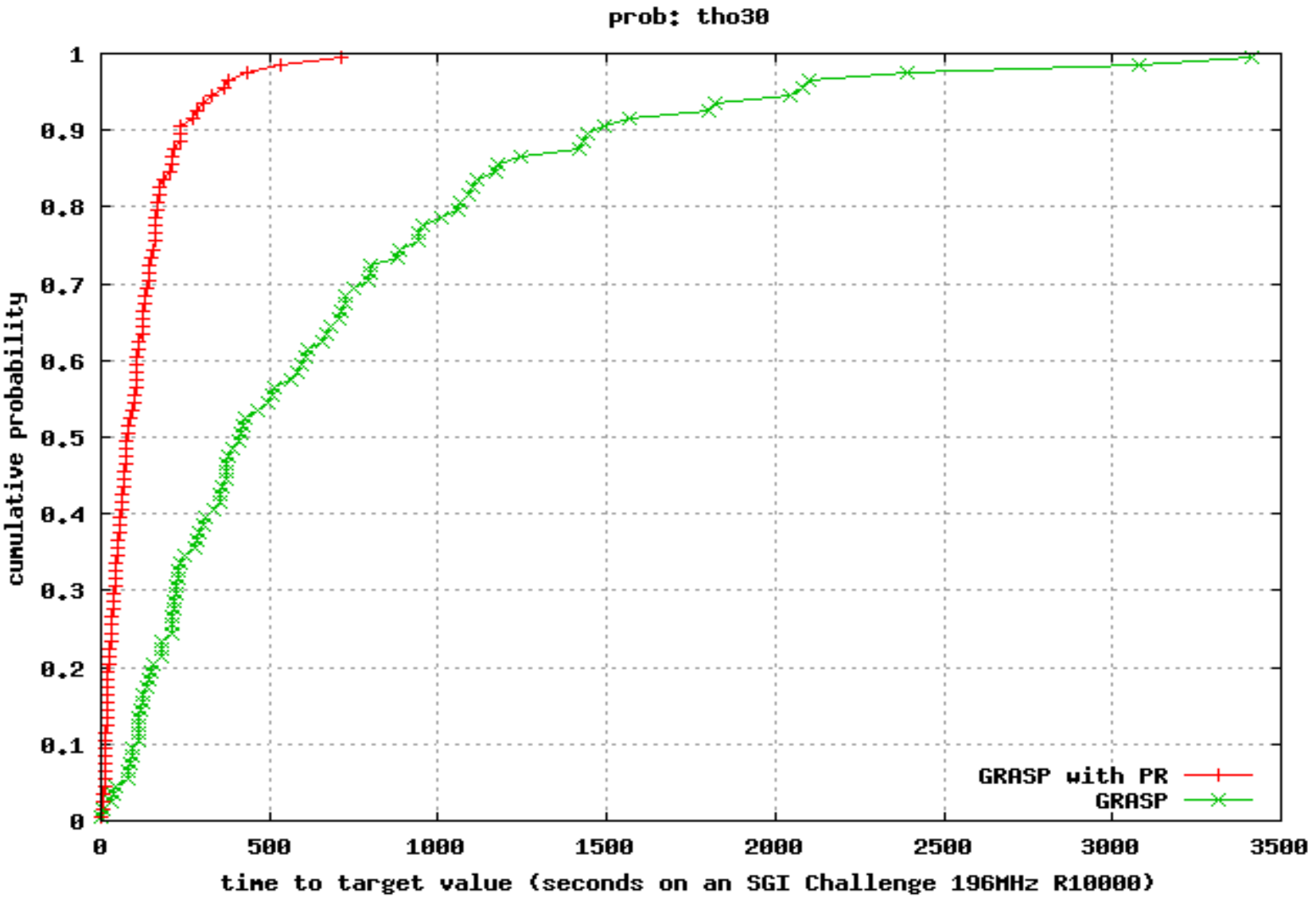
MAX-SAT (Festa, Pardalos, Pitsoulis, and R., 2006)



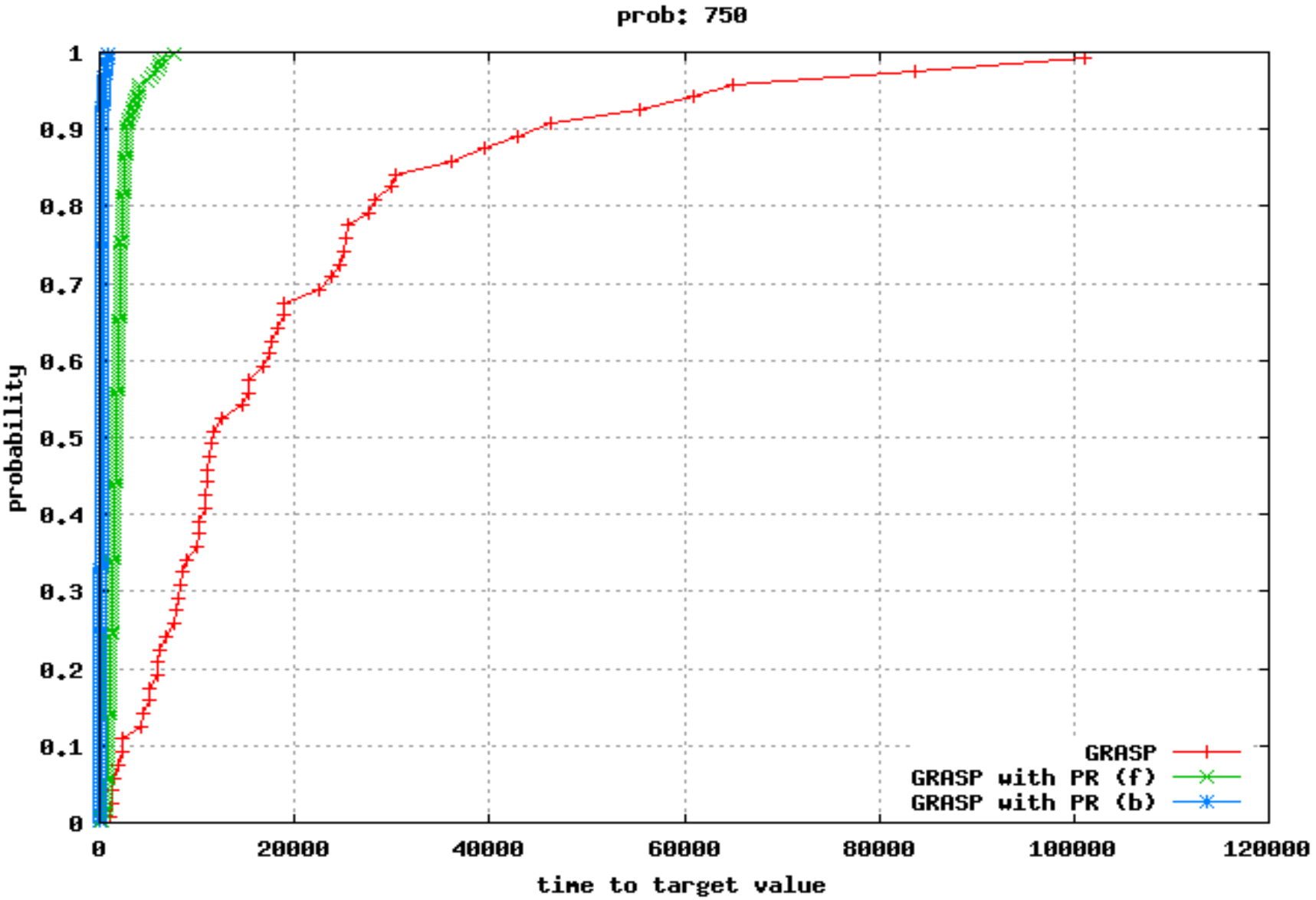
3-index assignment (Aiex, R., Pardalos, & Toraldo, 2005)



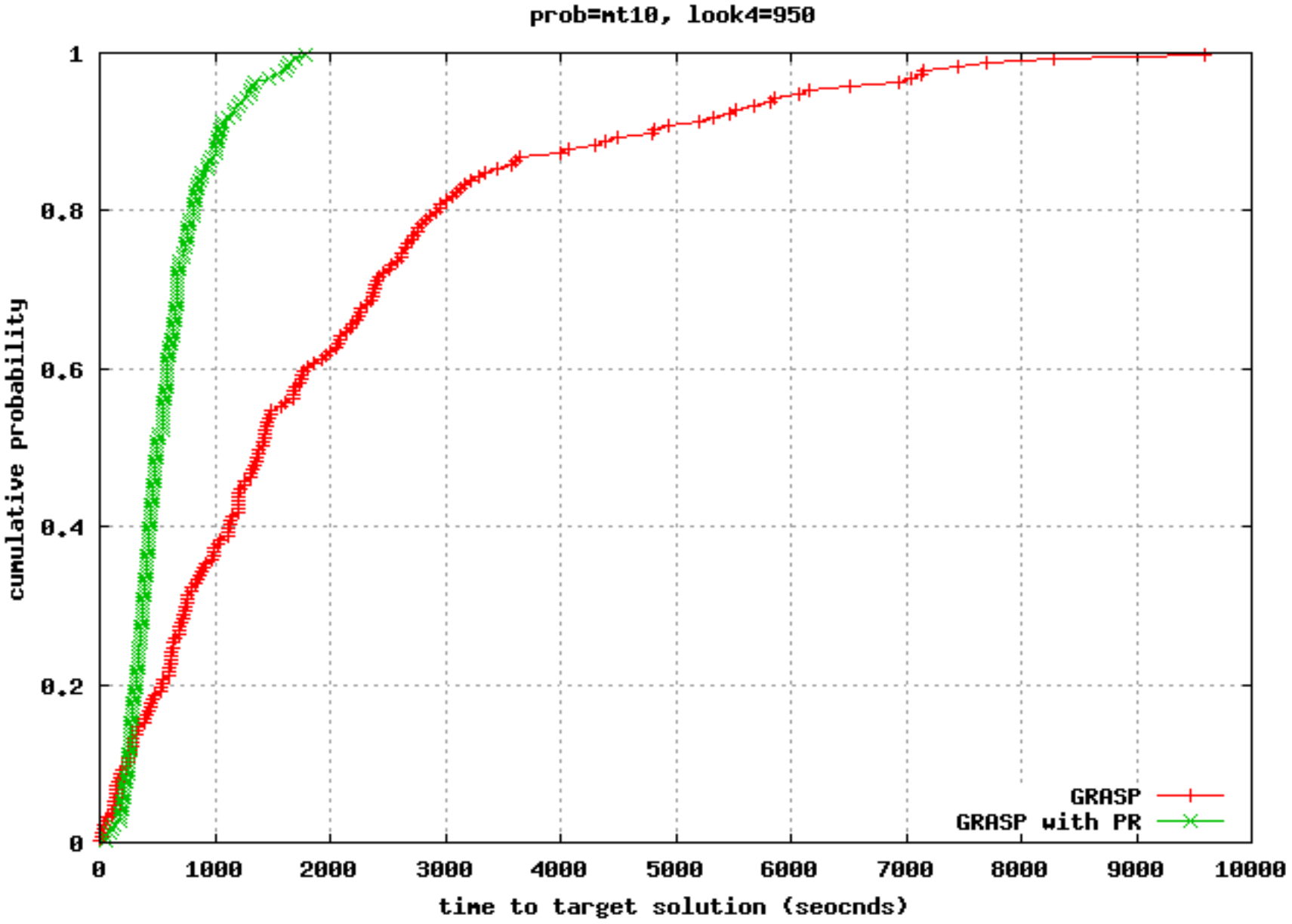
QAP (Oliveira, Pardalos, and R., 2004)



Bandwidth packing (R. and Ribeiro, 2003)



Job shop scheduling (Aiex, Binato, & R., 2003)



GRASP with path-relinking:
Pool management

P is a set (pool) 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

(R. & Werneck, 2004, 2006)

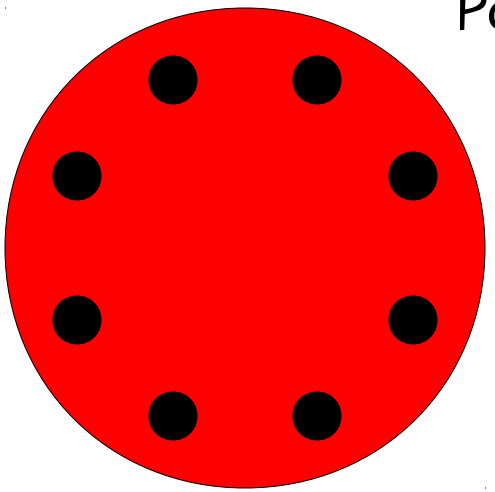
Evolutionary path-relinking "evolves" the pool, i.e. transforms it into a pool of diverse elements whose solution values are better than those of the original pool.

Evolutionary path-relinking can be used

as an intensification procedure at certain points of the solution process;

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

Evolutionary path-relinking (EvPR)

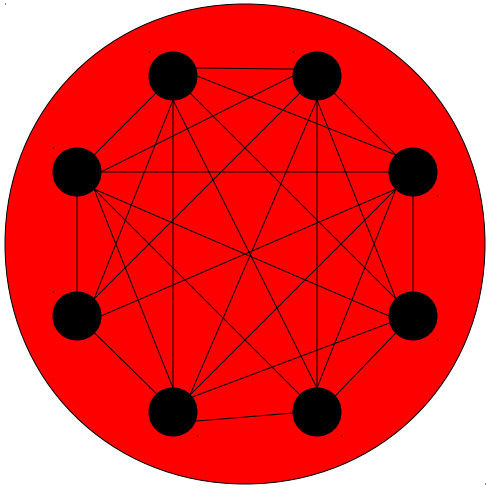


Population P(0)

Each "population" of EvPR starts with a pool of elite solutions of size $|P|$.

Population P(0) is the current elite set.

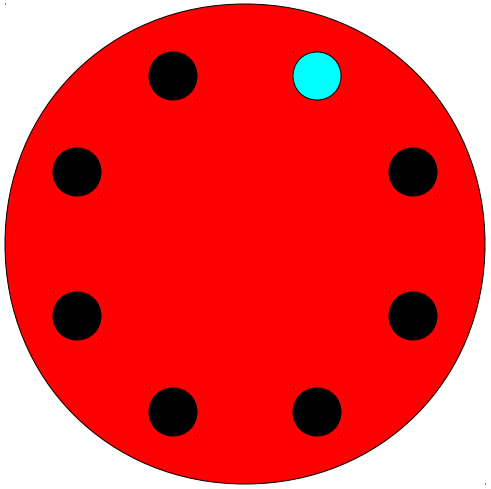
Evolutionary path-relinking (EvPR)



All pairs of elite solutions (x,y) in K-th population P(K) are path-relinked and the resulting z = PR(x,y) is a candidate for inclusion in population P(K+1).

Rules for inclusion into P(K+1) are the same used for inclusion into any pool.

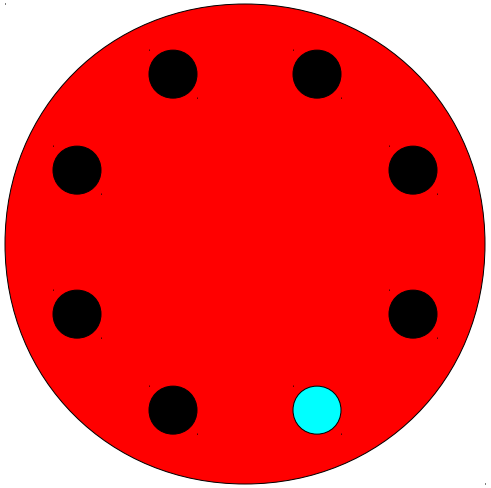
Evolutionary path-relinking (EvPR)



Population P(K)

If best solution in population P(K+1) has same objective function value as best solution in population P(K), process stops.

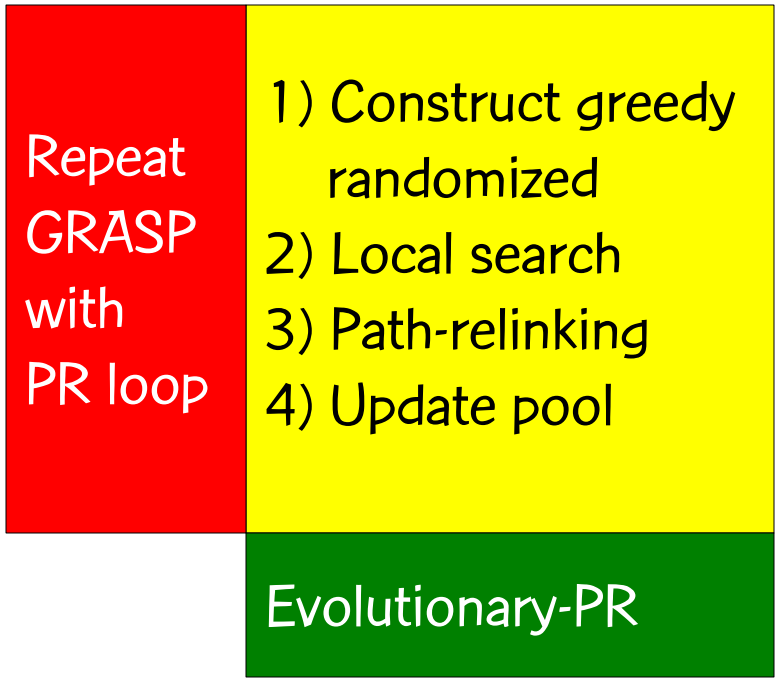
Else K=K+1 and repeat.



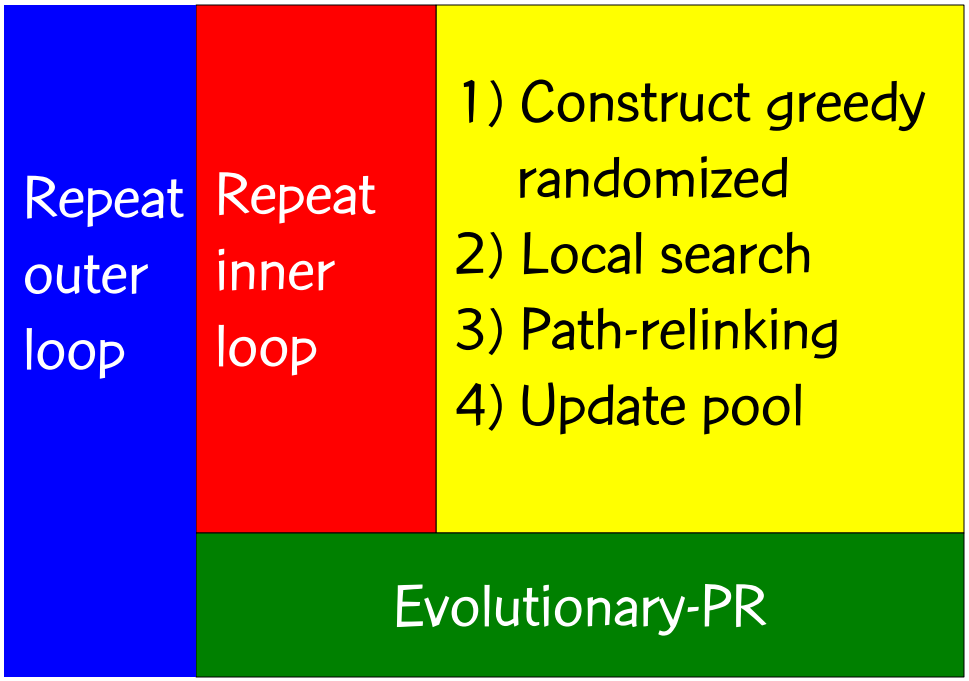
Population P(K+1)

GRASP with evolutionary path-relinking

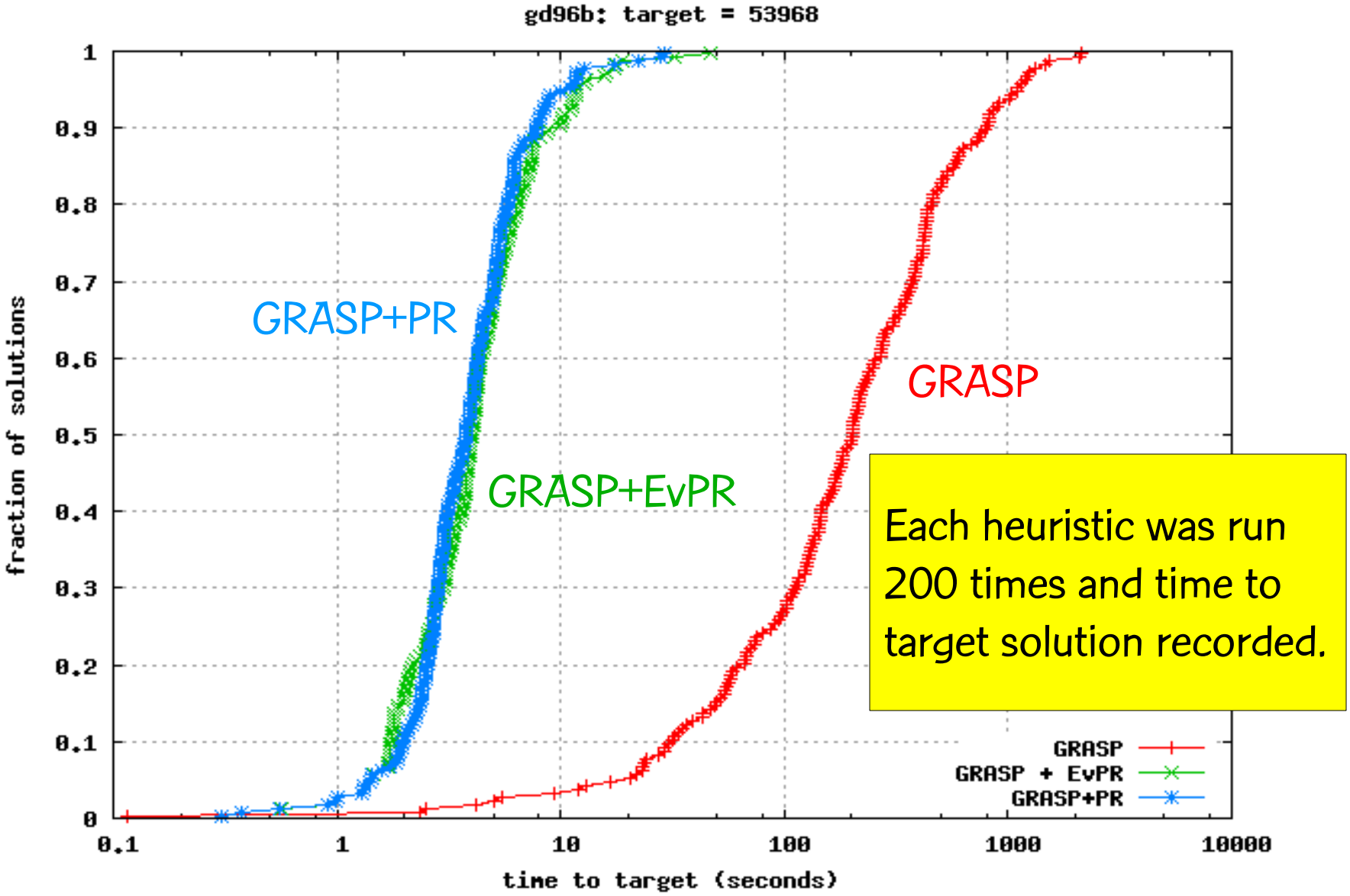
As post-optimization



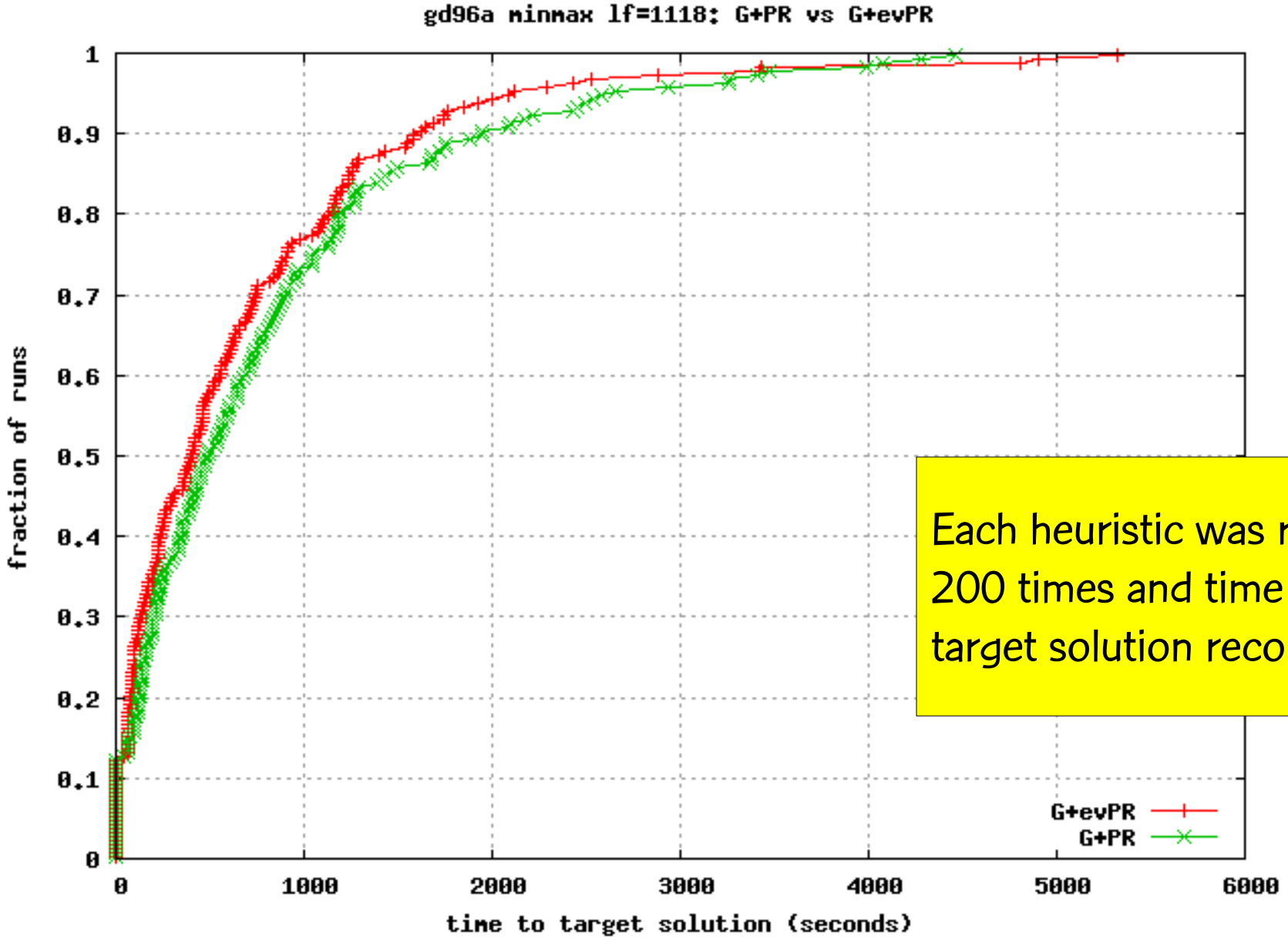
During GRASP + PR



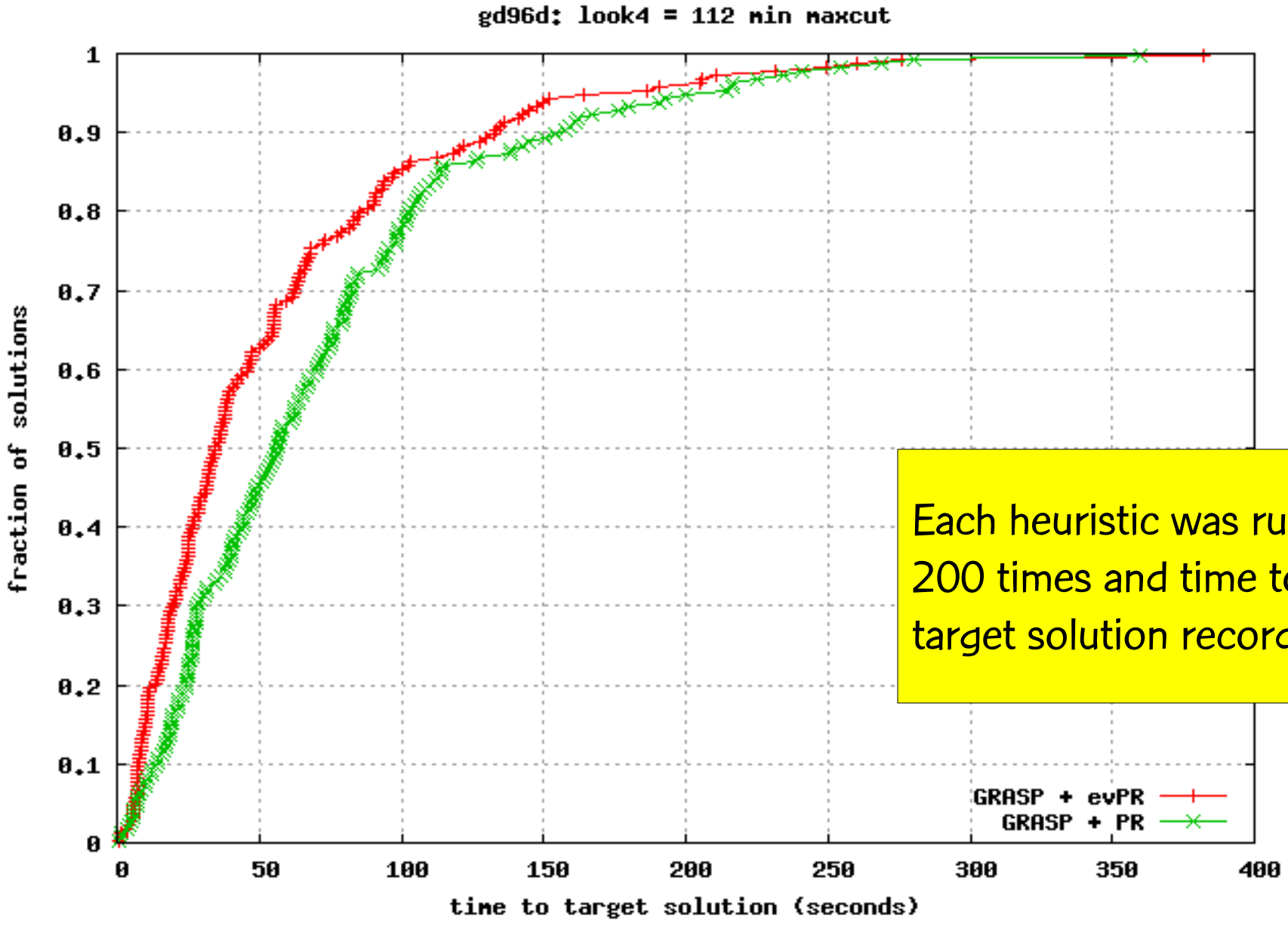
(Resende & Werneck, 2004, 2006)



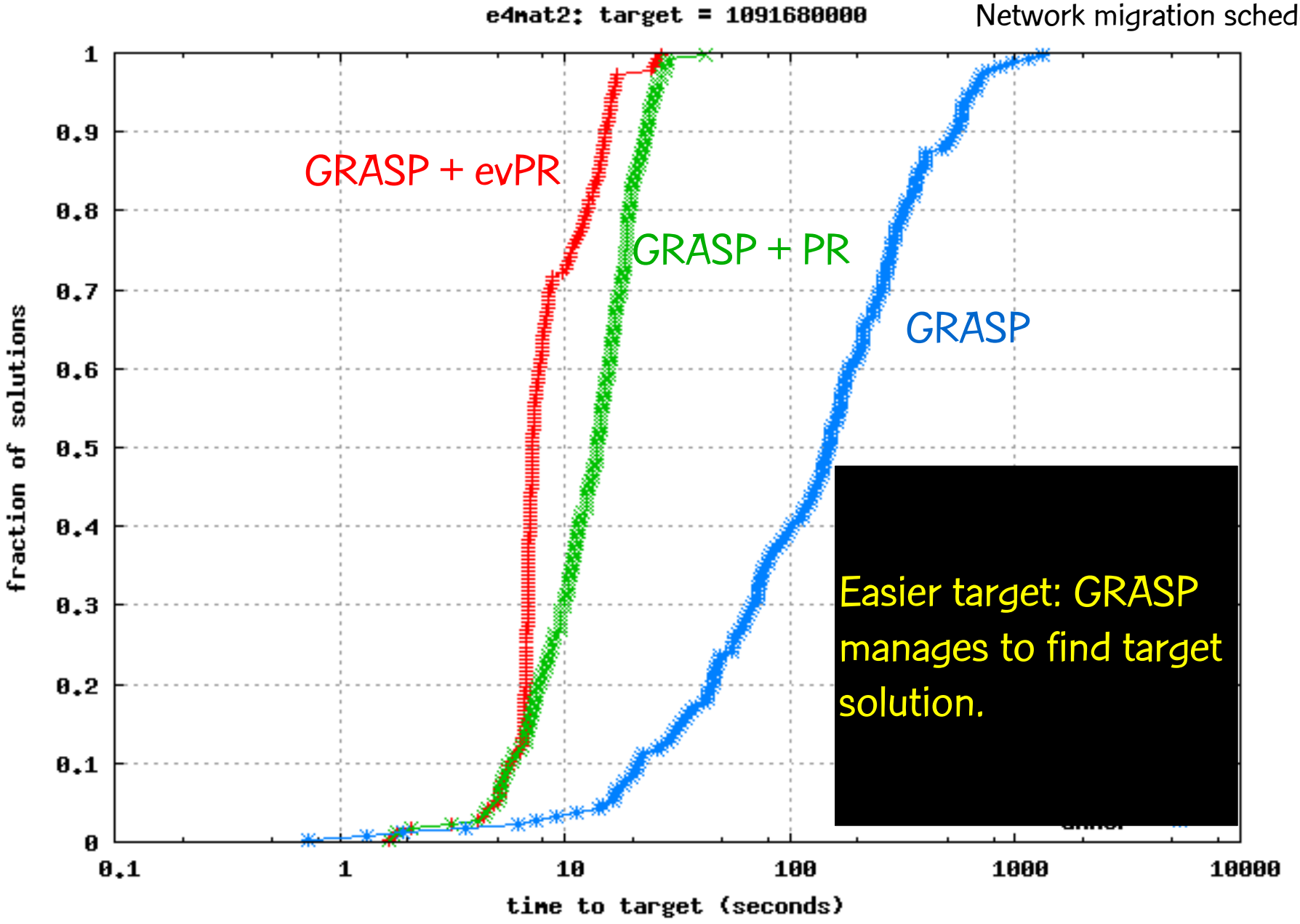
Each heuristic was run 200 times and time to target solution recorded.

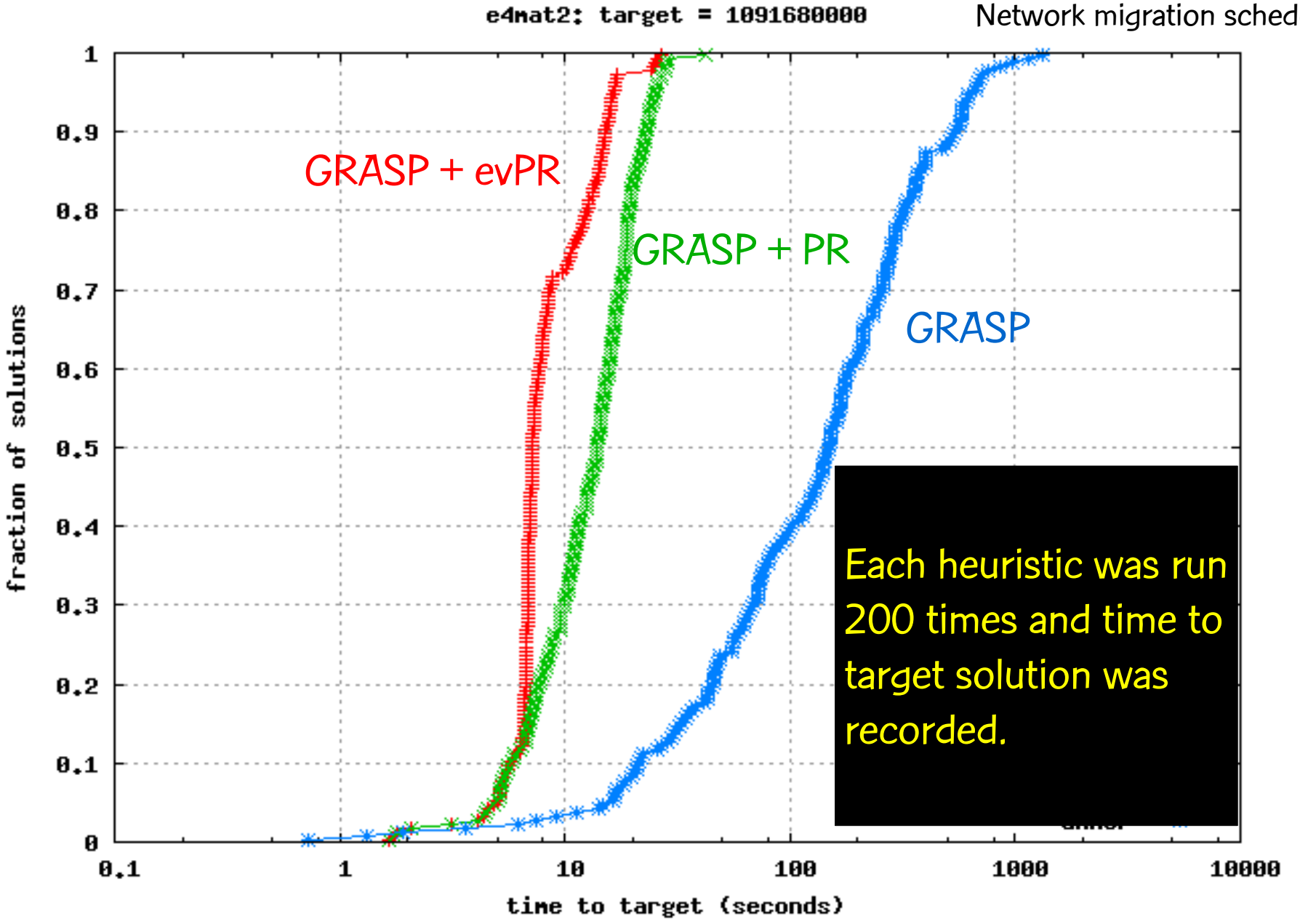


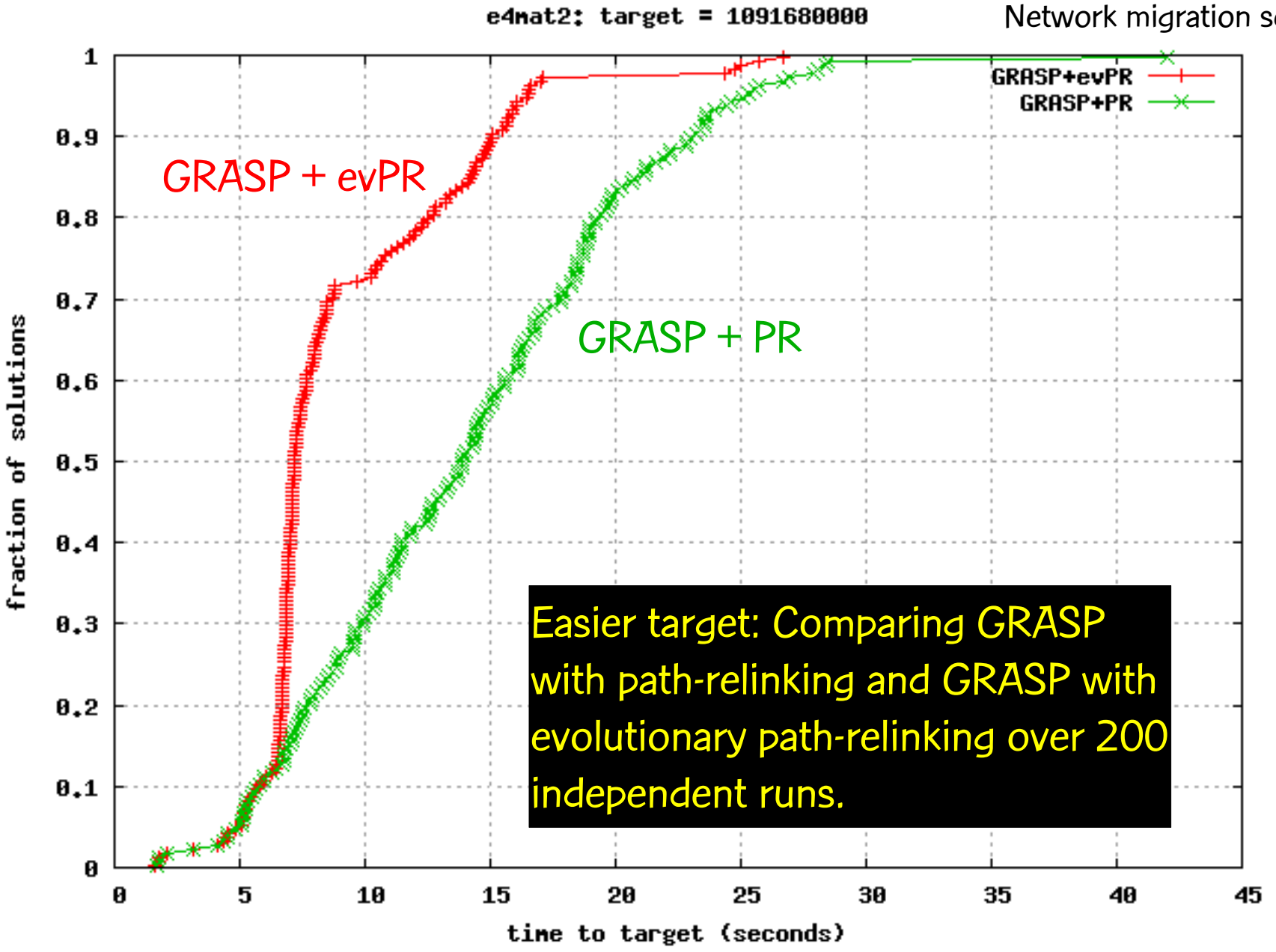
Each heuristic was run 200 times and time to target solution recorded.

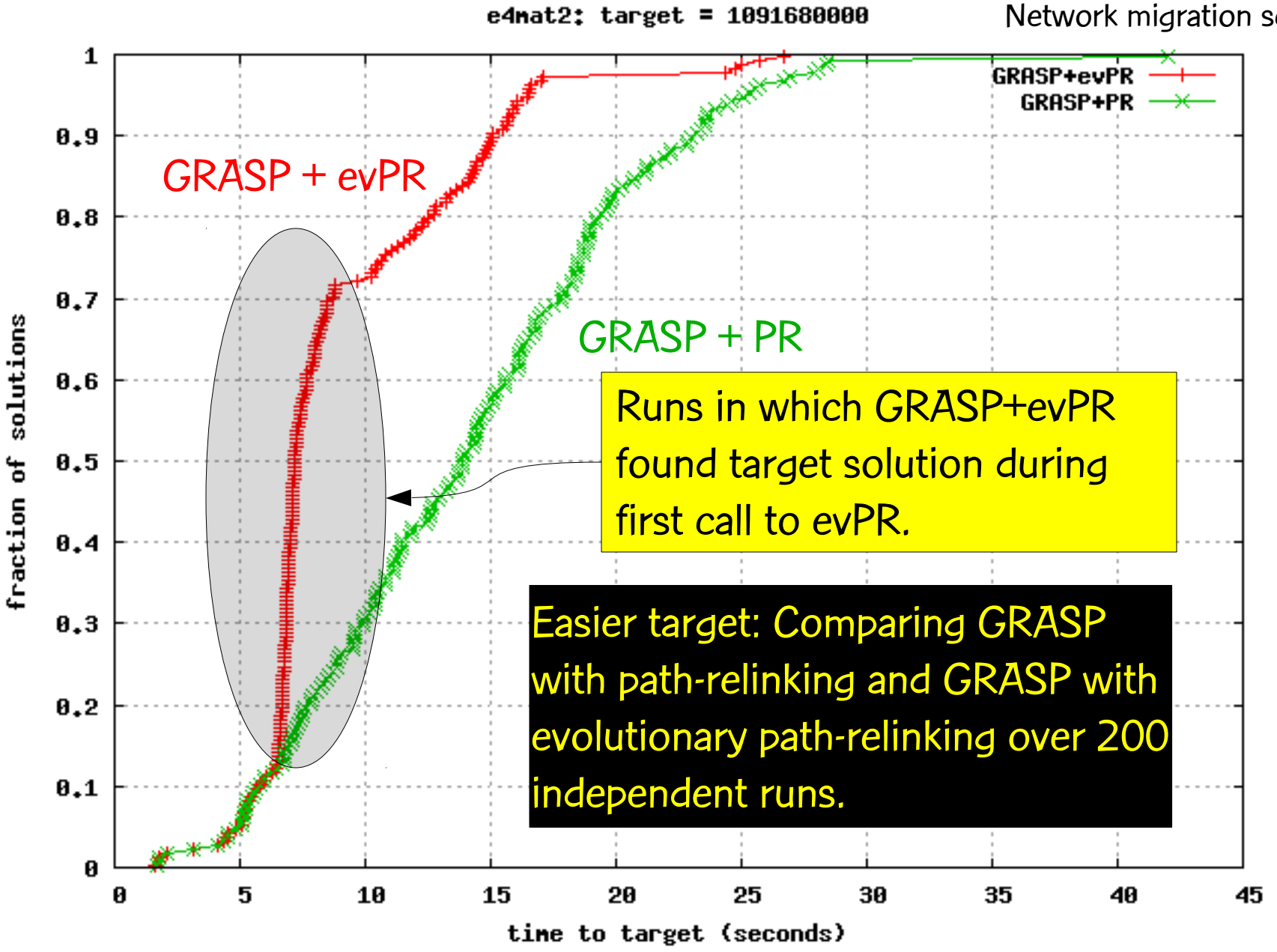


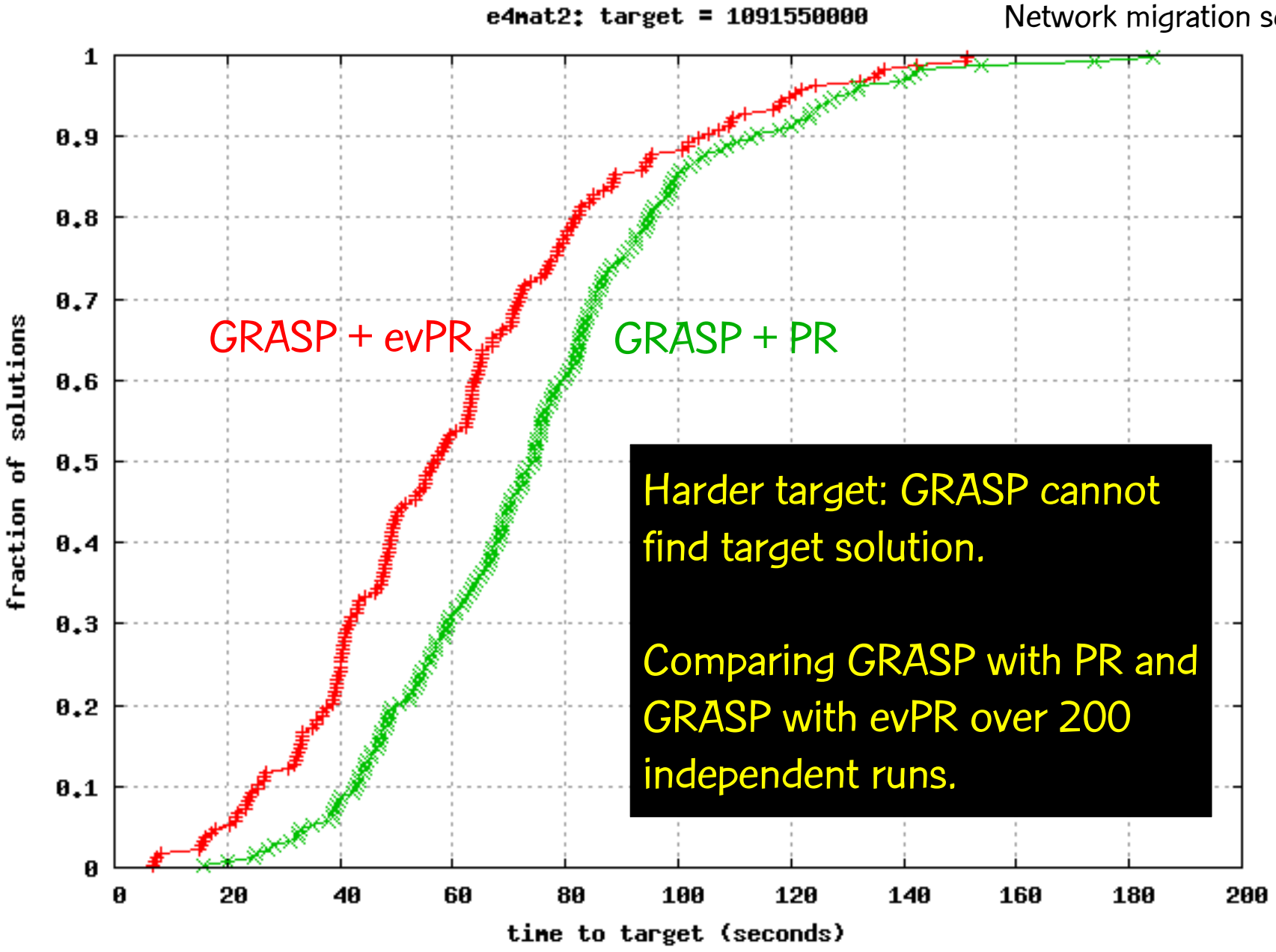
Each heuristic was run 200 times and time to target solution recorded.


e4mat2: target = 1091680000
GRASP + evPR
GRASP + PR
GRASP
Easier target: GRASP manages to find target solution.
fraction of solutions
time to target (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.1
1
10
100
1000
10000


e4mat2: target = 1091680000
GRASP + evPR
GRASP + PR
GRASP
Each heuristic was run 200 times and time to target solution was recorded.
fraction of solutions
time to target (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.1
1
10
100
1000
10000


e4mat2: target = 1091680000
GRASP+evPR
GRASP+PR
GRASP + evPR
GRASP + PR
Easier target: Comparing GRASP with path-relinking and GRASP with evolutionary path-relinking over 200 independent runs.
fraction of solutions
time to target (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30
35
40
45

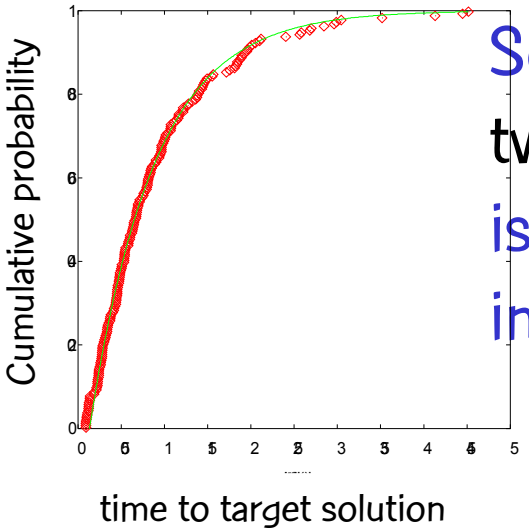

e4mat2: target = 1091680000
GRASP+evPR
GRASP+PR
GRASP + evPR
GRASP + PR
Runs in which GRASP+evPR found target solution during first call to evPR.
Easier target: Comparing GRASP with path-relinking and GRASP with evolutionary path-relinking over 200 independent runs.
fraction of solutions
time to target (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30
35
40
45


e4mat2: target = 1091550000
fraction of solutions
time to target (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120
140
160
180
200
GRASP + evPR
GRASP + PR
Harder target: GRASP cannot find target solution.
Comparing GRASP with PR and GRASP with evPR over 200 independent runs.

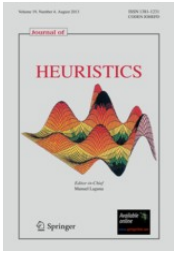
Other topics not covered today

- Runtime distribution of GRASP
- Parallel GRASP & parallel GRASP with path-relinking
- Restart strategies for GRASP with path-relinking
- Continuous GRASP
- Automatic configuration of algorithm components and tuning of parameters
- LaGRASP: Lagrangian GRASP

Runtime distribution of GRASP



Solution time to a sub-optimal target value fits a two-parameter exponential distribution. Hence, it is possible to achieve linear speed-up by implementing GRASP in parallel.



R.M. Aiex, R., and C.C. Ribeiro, "Probability distribution of solution time in GRASP: An experimental investigation," J. of Heuristics, vol. 8, pp. 343-373, 2002.

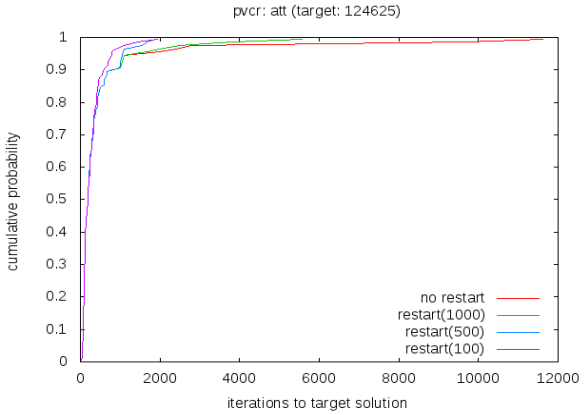
Parallel GRASP & GRASP with PR

Possible to achieve linear speed-up by implementing GRASP in parallel and super-linear speed-up with GRASP with PR.

T.A. Feo, R., and S.H. Smith, "A greedy randomized adaptive search procedure for maximum independent set," Operations Research, vol. 42, pp. 860-878, 1994.

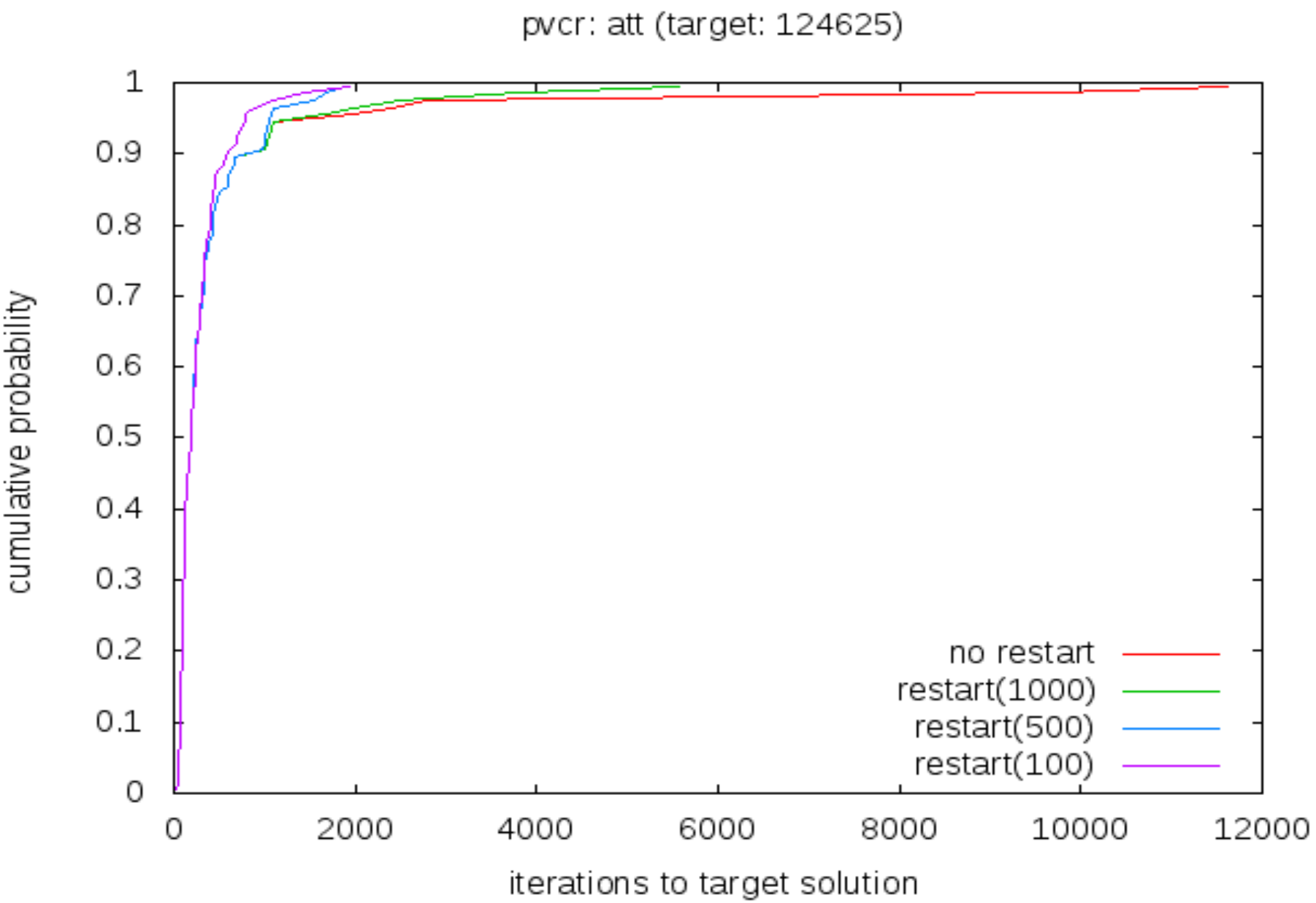
R.M. Aiex, S. Binato, and R., "Parallel GRASP with path-relinking for job shop scheduling," Parallel Computing, vol. 29, pp. 393-430, 2003

Restart strategies for GRASP with PR



With restart, it is possible to reduce maximum, average, and standard deviation of iteration count (running time) when compared with no restart

R. and C.C. Ribeiro, "Restart strategies for GRASP with path-relinking heuristics," Optimization Letters, vol. 5, pp. 467-478, 2011.


pvcr: att (target: 124625)
cumulative probability
iterations to target solution
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2000
4000
6000
8000
10000
12000
no restart
restart(1000)
restart(500)
restart(100)

Continuous GRASP

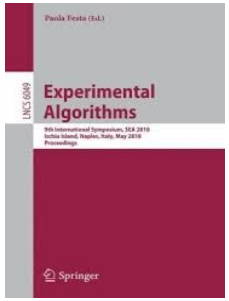
C-GRASP is an extension of GRASP for multi-modal box-constrained continuous global optimization

M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and R., "Global optimization by continuous GRASP," Optimization Letters, vol. 1, pp. 201-212, 2007.

M.J. Hirsch, P.M. Pardalos, and R., "Speeding up continuous GRASP," European J. of Operational Research, vol. 205, pp. 507-521, 2010.

Automatic configuration of algorithm components and tuning of parameters

Components of GRASP can be automatically configured, parameters automatically tuned, resulting in significant speedups when compared to manually configured and tuned GRASP.



P. Festa, J.F. Gonçalves, R., and R.M.A. Silva, "Automatic tuning of GRASP with path-relinking heuristics with a biased random-key genetic algorithm," in "Experimental Algorithms," P. Festa (Ed.), Lecture Notes in Computer Science, vol. 6049, pp. 338-349, 2010.

L.F. Morán-Mirabal, J.L. González-Velarde, and R., "Automatic tuning of GRASP with evolutionary path-relinking," in "Hybrid Metaheuristics 2013 (HM 2013)," M.J. Blesa et al., (Eds.), Lecture Notes in Computer Science, vol. 7919, pp. 62-77, 2013.

LaGRASP: Lagrangian GRASP

LaGRASP makes use of dual information, using reduced costs in place of original costs, leading to faster convergence and improved solutions.

L.S. Pessoa, R., and C.C. Ribeiro, "A hybrid Lagrangean heuristic with GRASP and path relinking for set k-covering," Computers and Operations Research, vol. 40, pp. 3132-3146, 2013

Some applications of GRASP

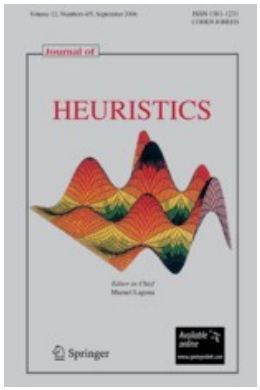
Some applications of GRASP and GRASP+PR at AT&T

- Worldnet PoP placement
- Caller cluster detection in call detail graph
- Unsplittable multi-commodity flow
- PBX telephone migration scheduling
- Handover minimization

Some applications of GRASP and GRASP+PR at AT&T

- Worldnet PoP placement
- Caller cluster detection in call detail graph
- Unsplittable multi-commodity flow
- PBX telephone migration scheduling
- Handover minimization

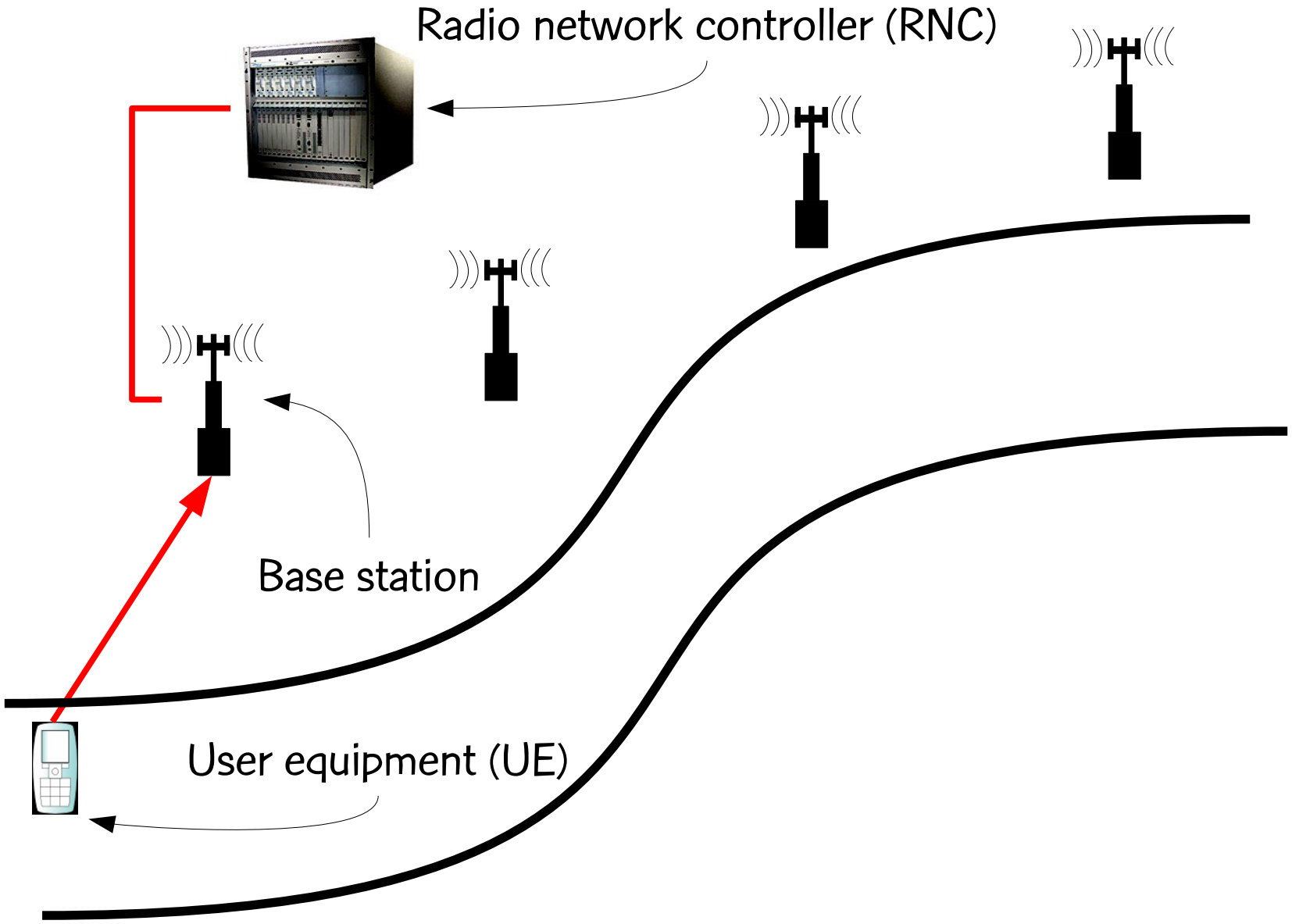
Handover minimization

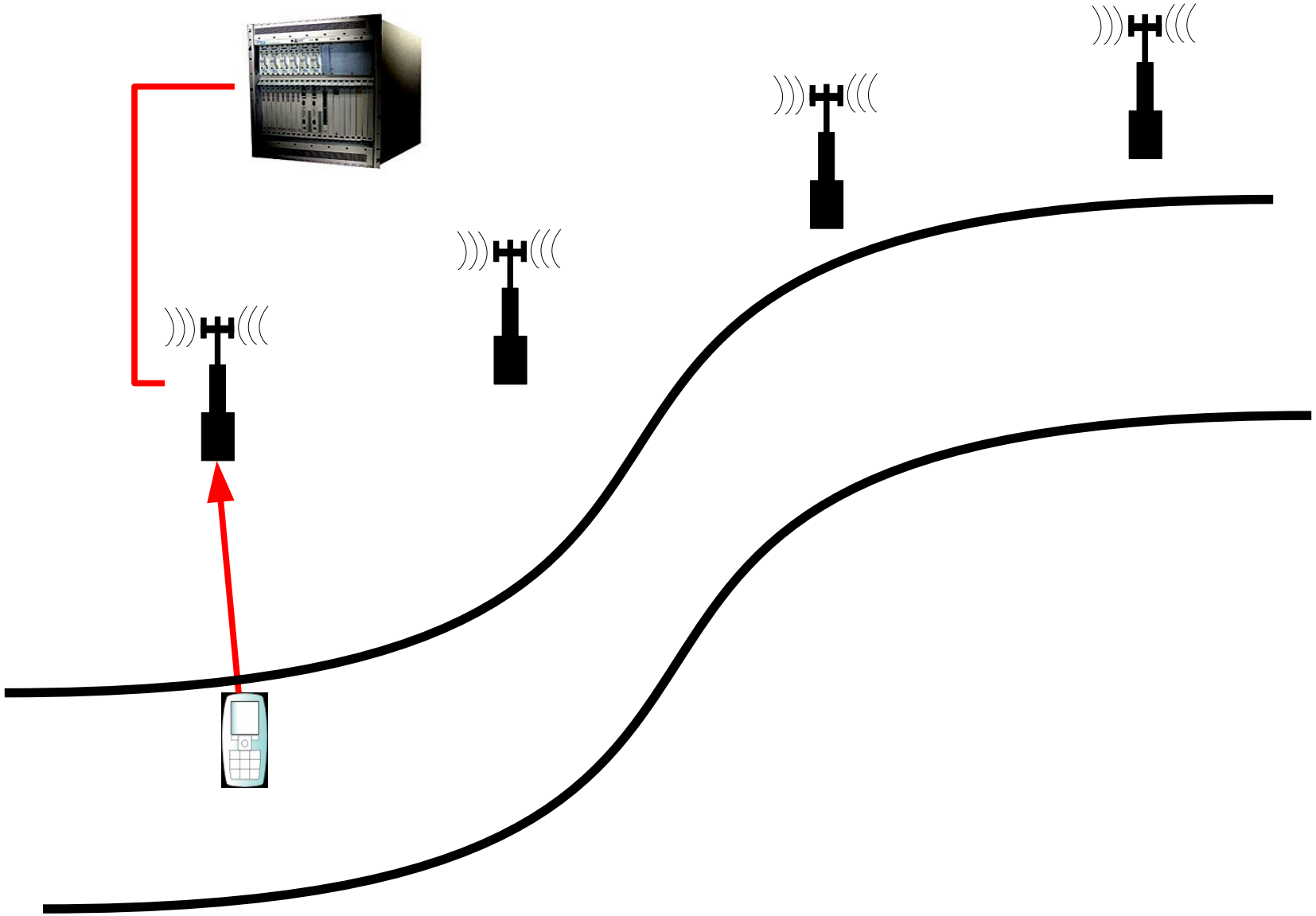


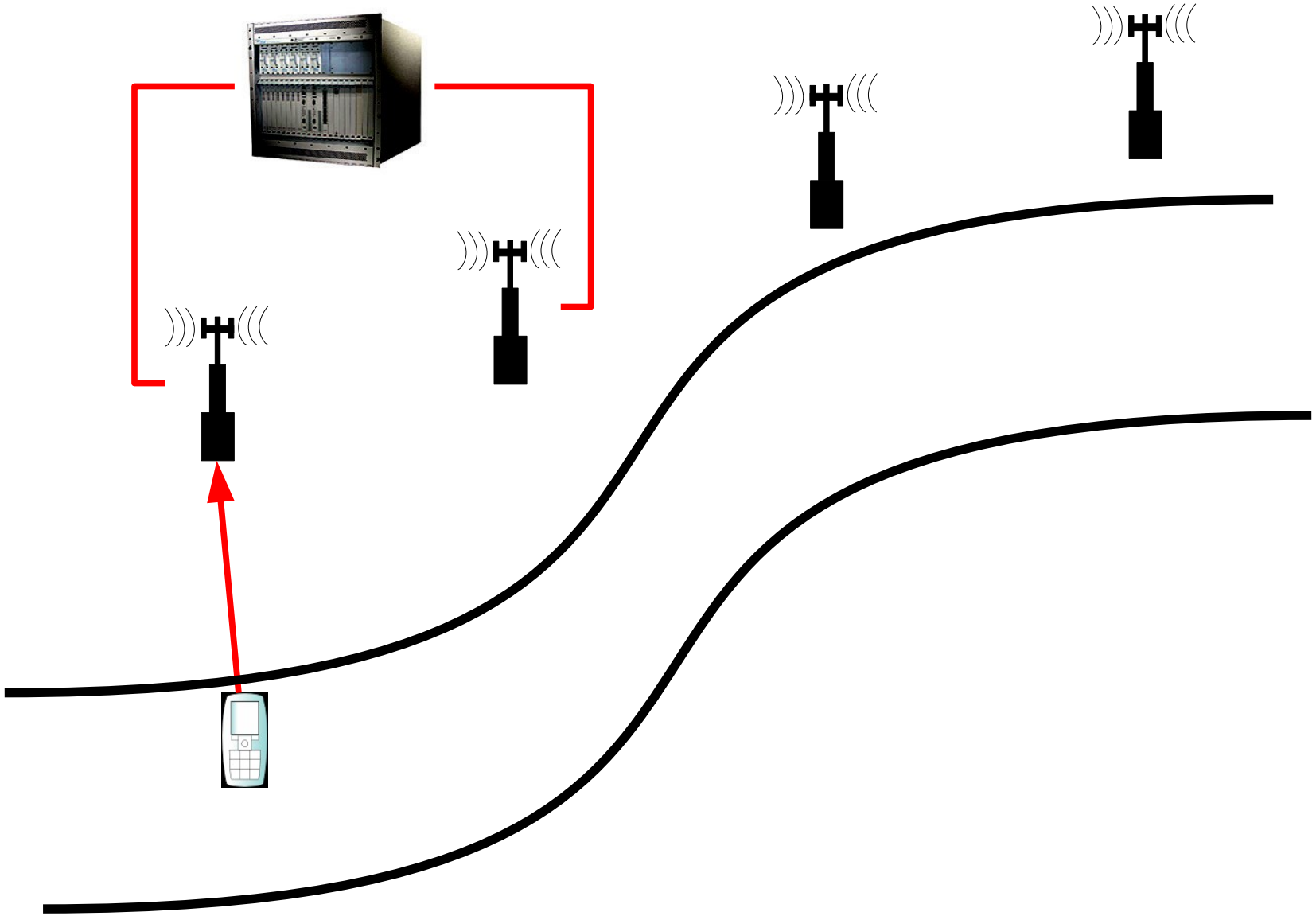
L.F. Morán-Mirabal, J.L. González-Velarde, R., and R.M.A. Silva, "Randomized heuristics for handover minimization in mobility networks", J. of Heuristics, vol. 19, pp. 845-880, 2013

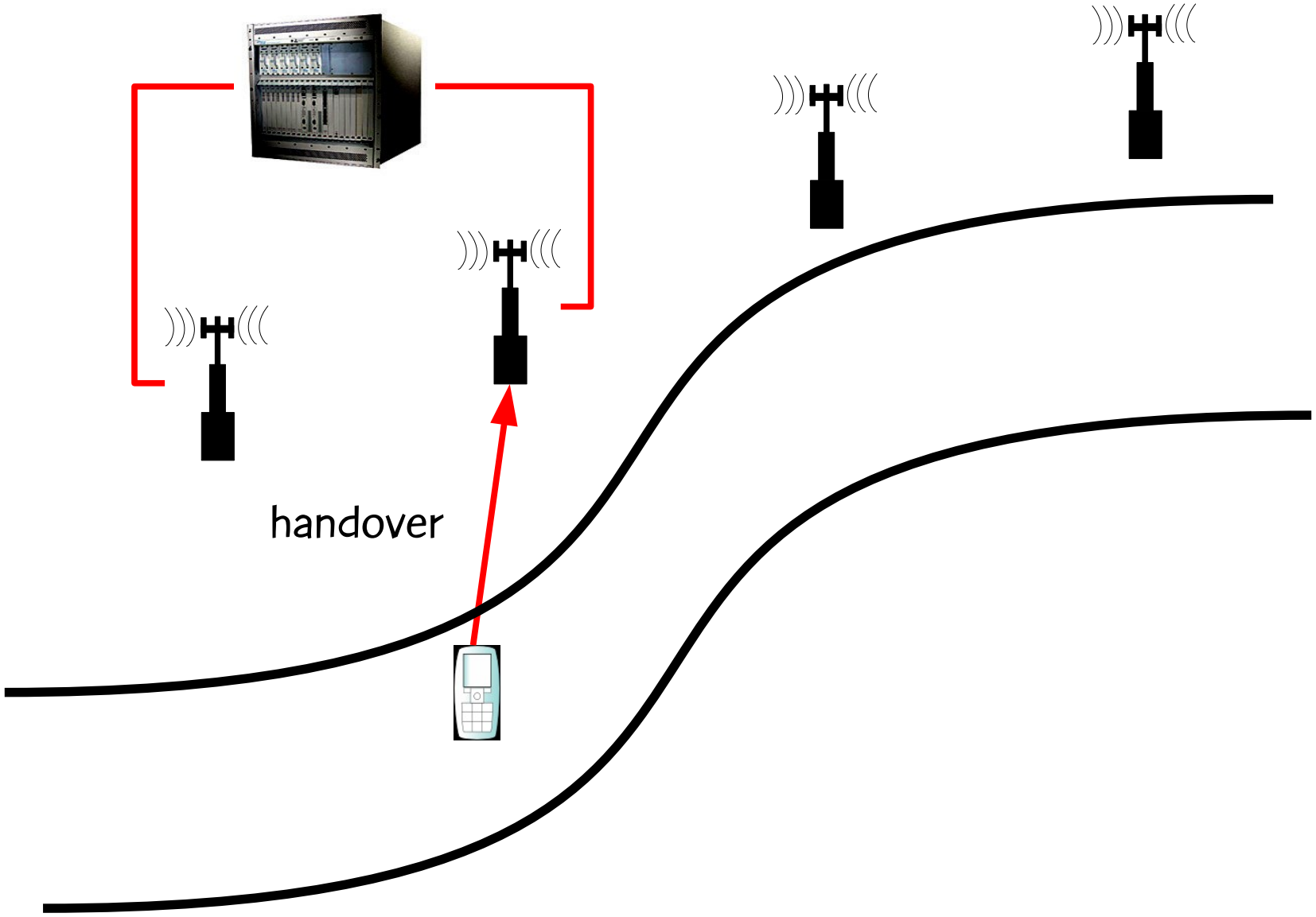
Tech report available here:

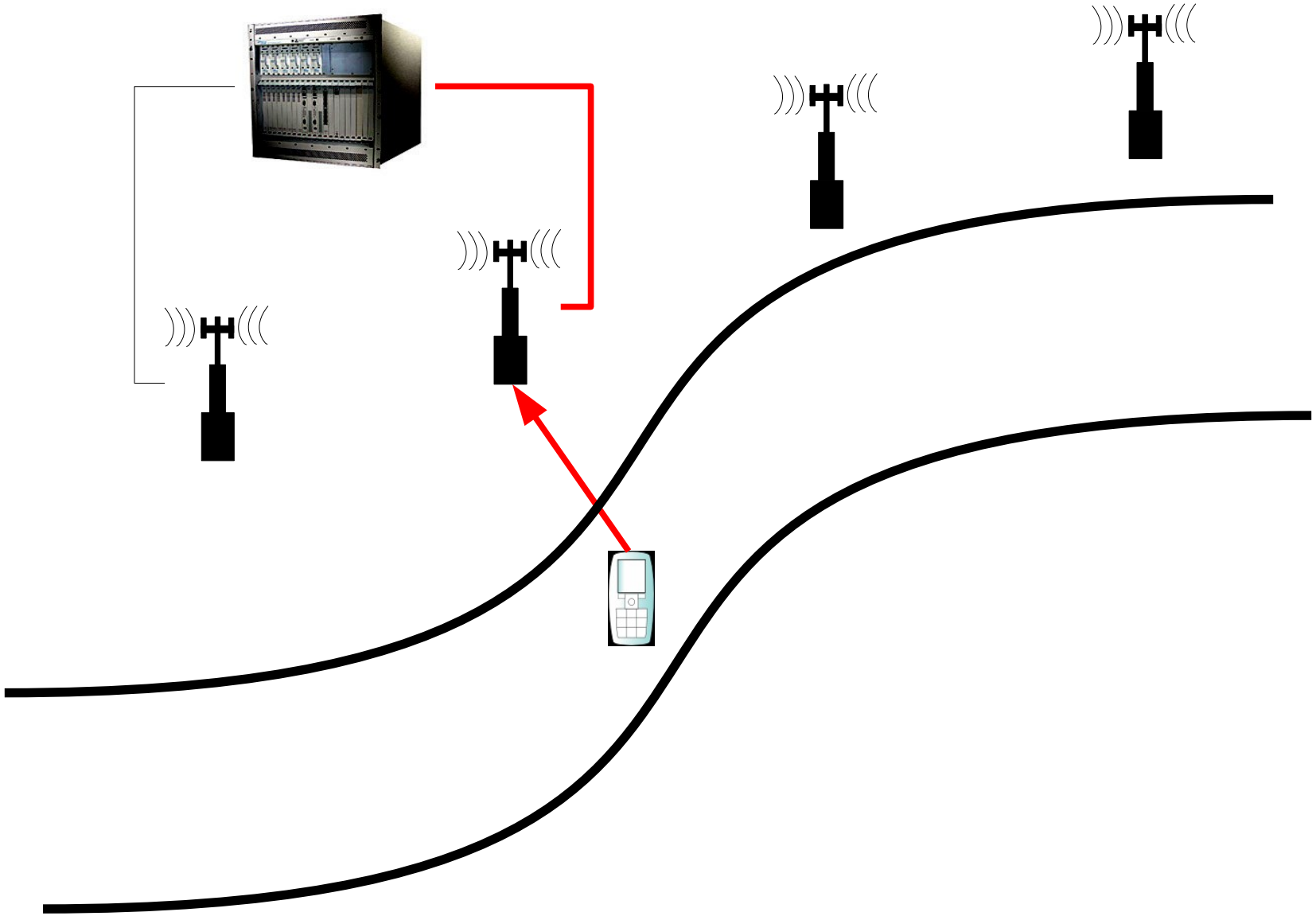
http://www.research.att.com/~mgcr/doc/randh-mhp.pdf

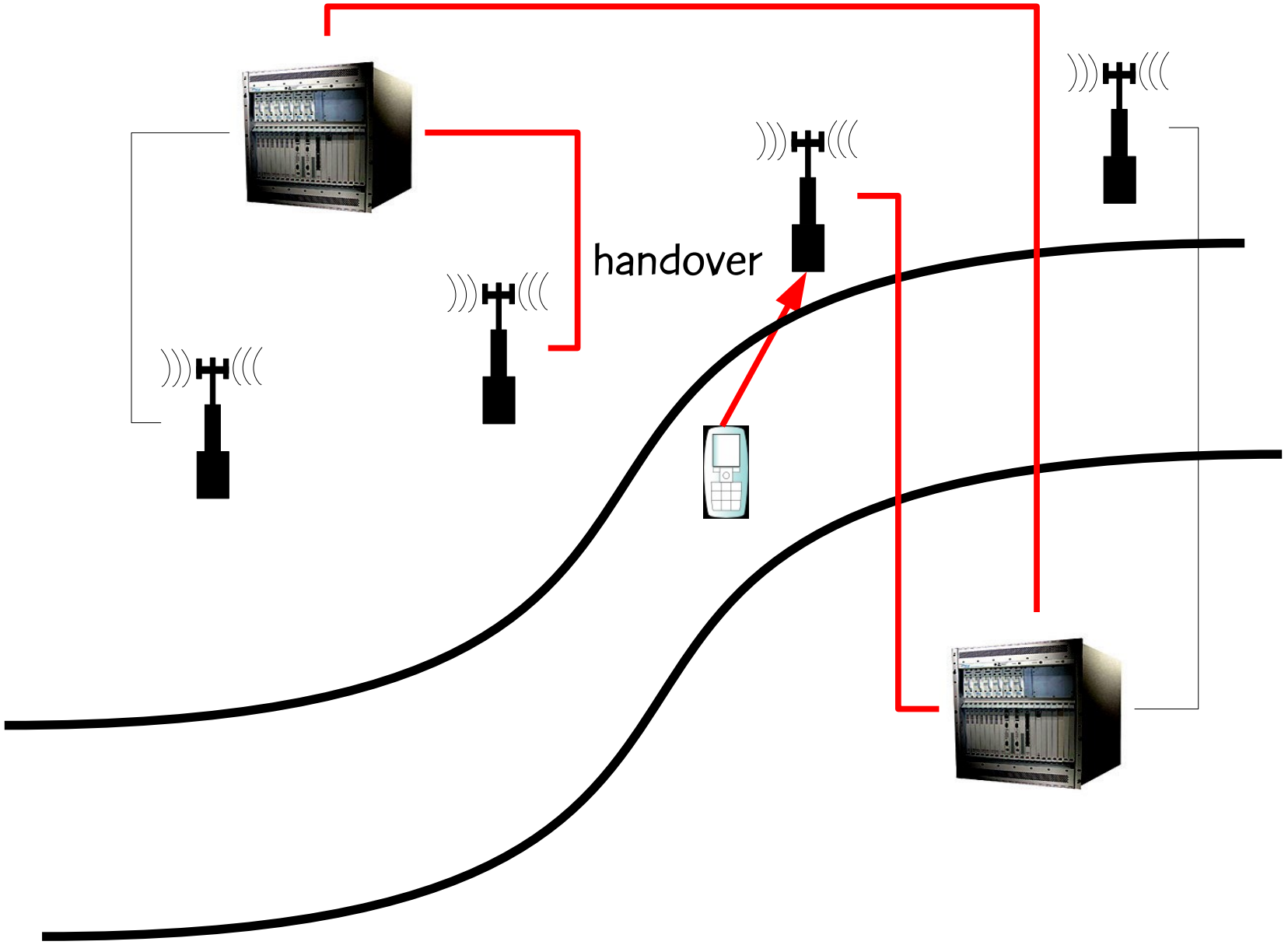

Radio network controller (RNC)
Base station
User equipment (UE)

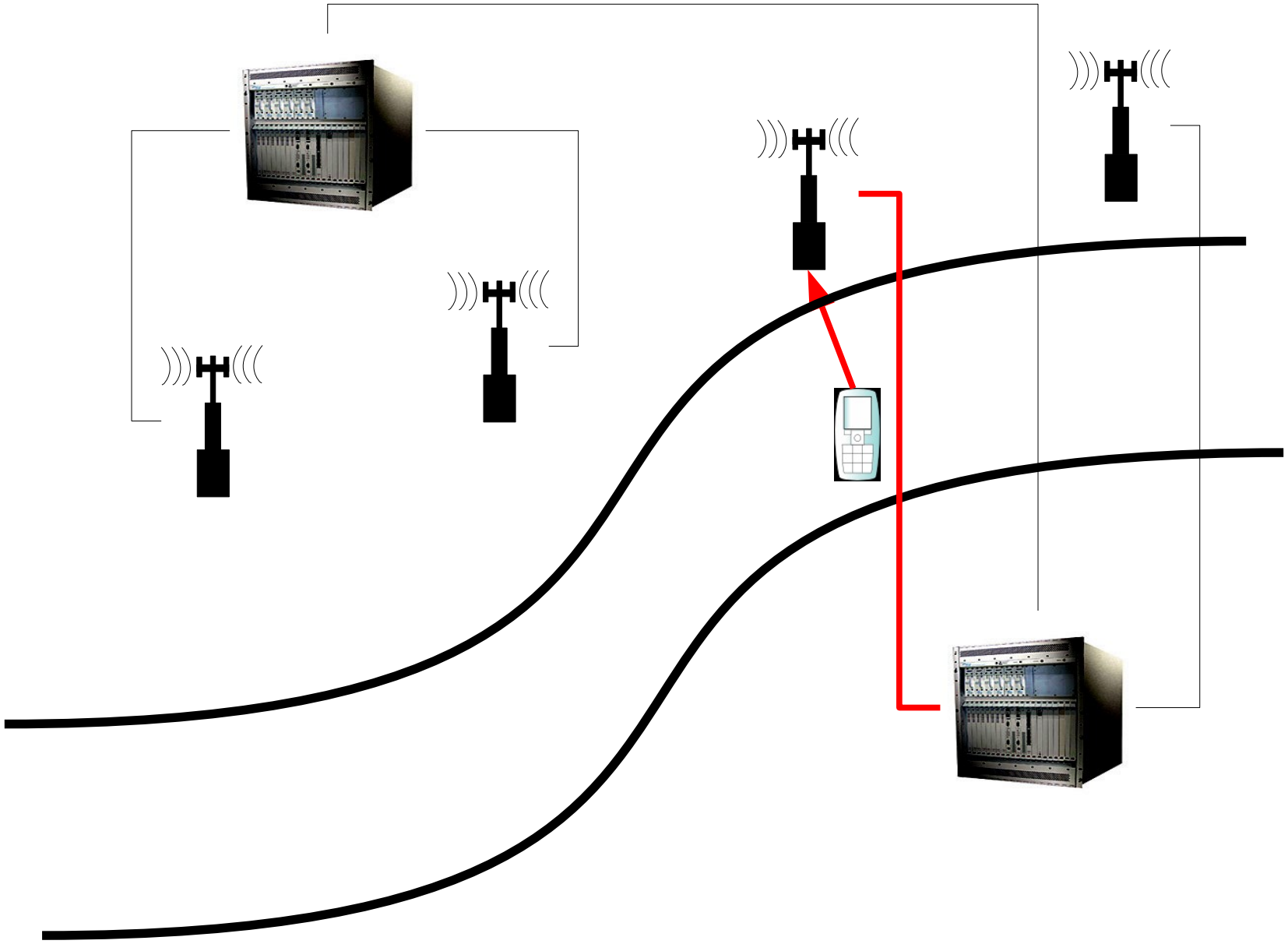


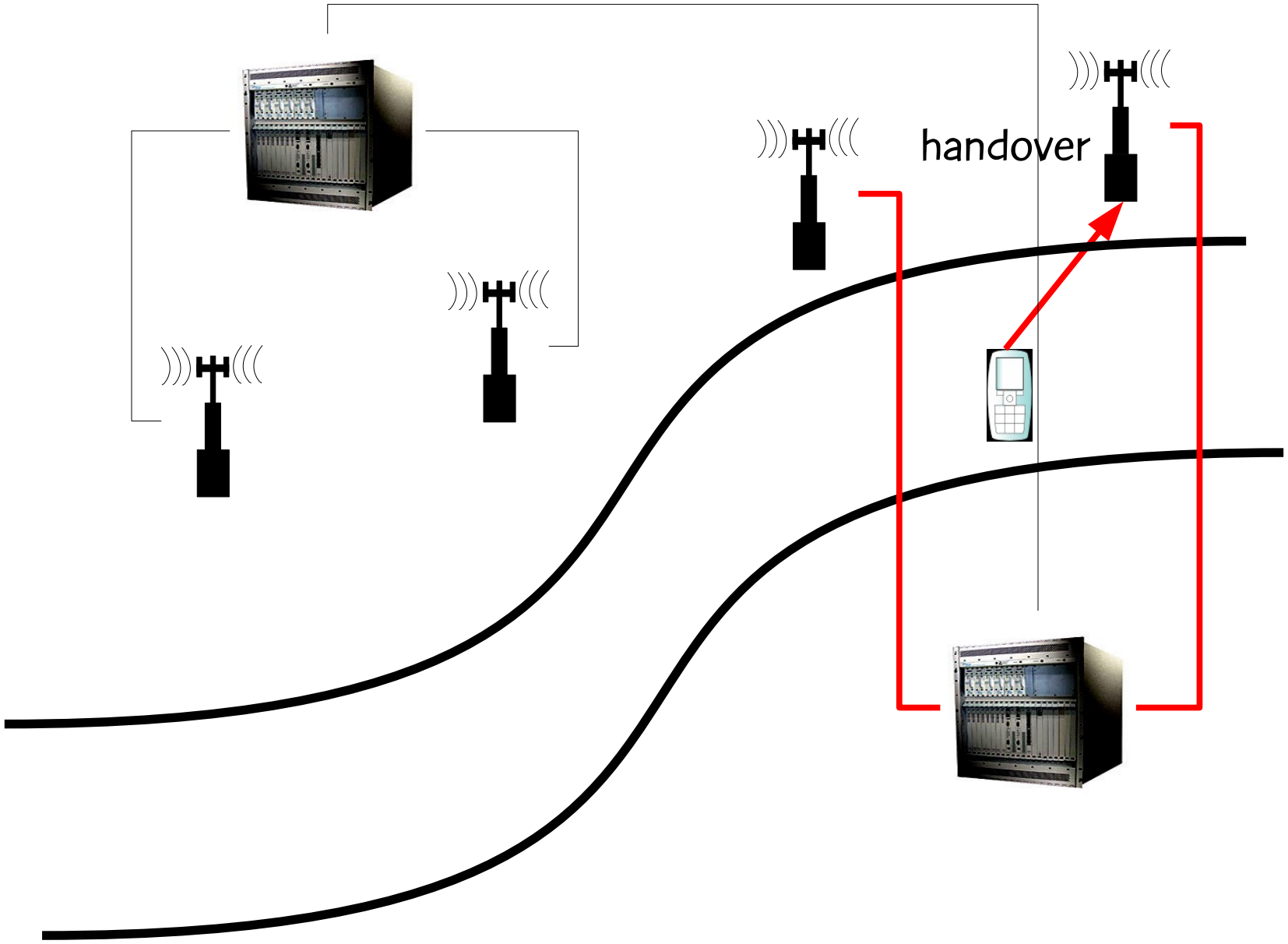


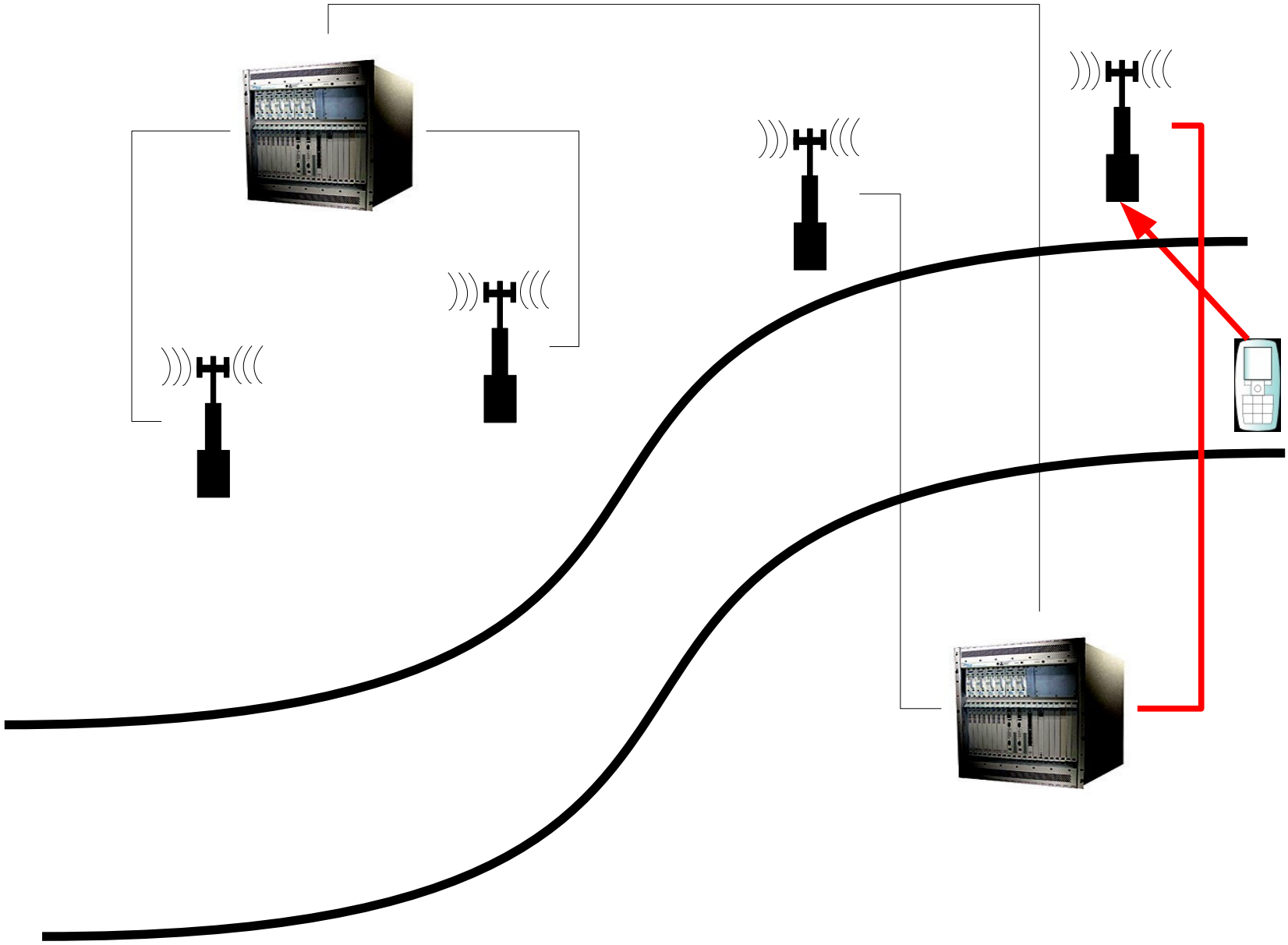

handover

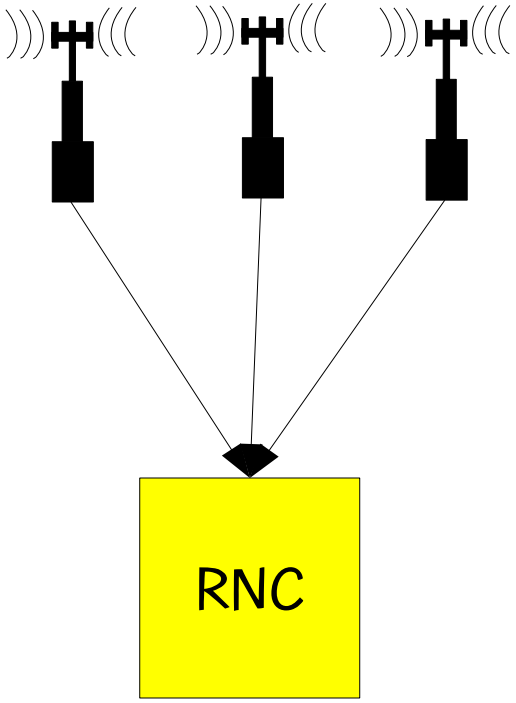



handover

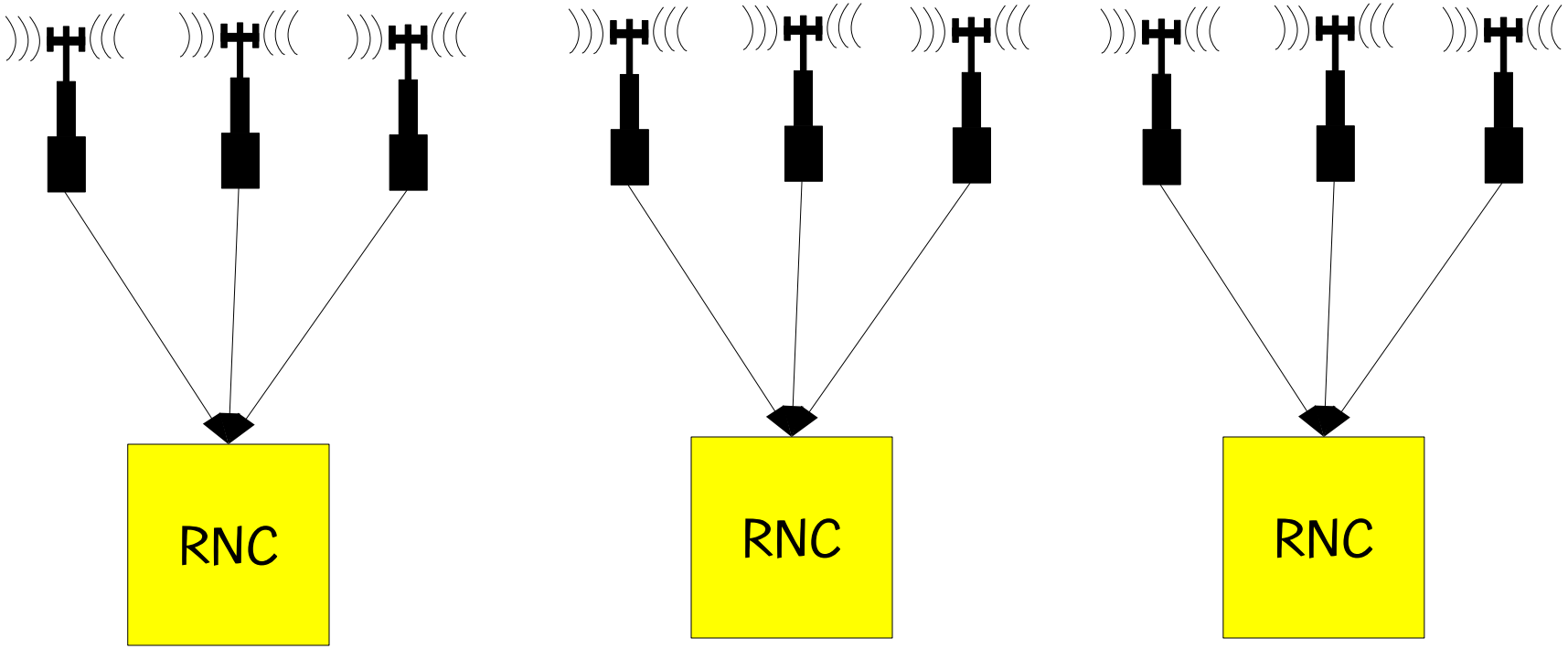



handover

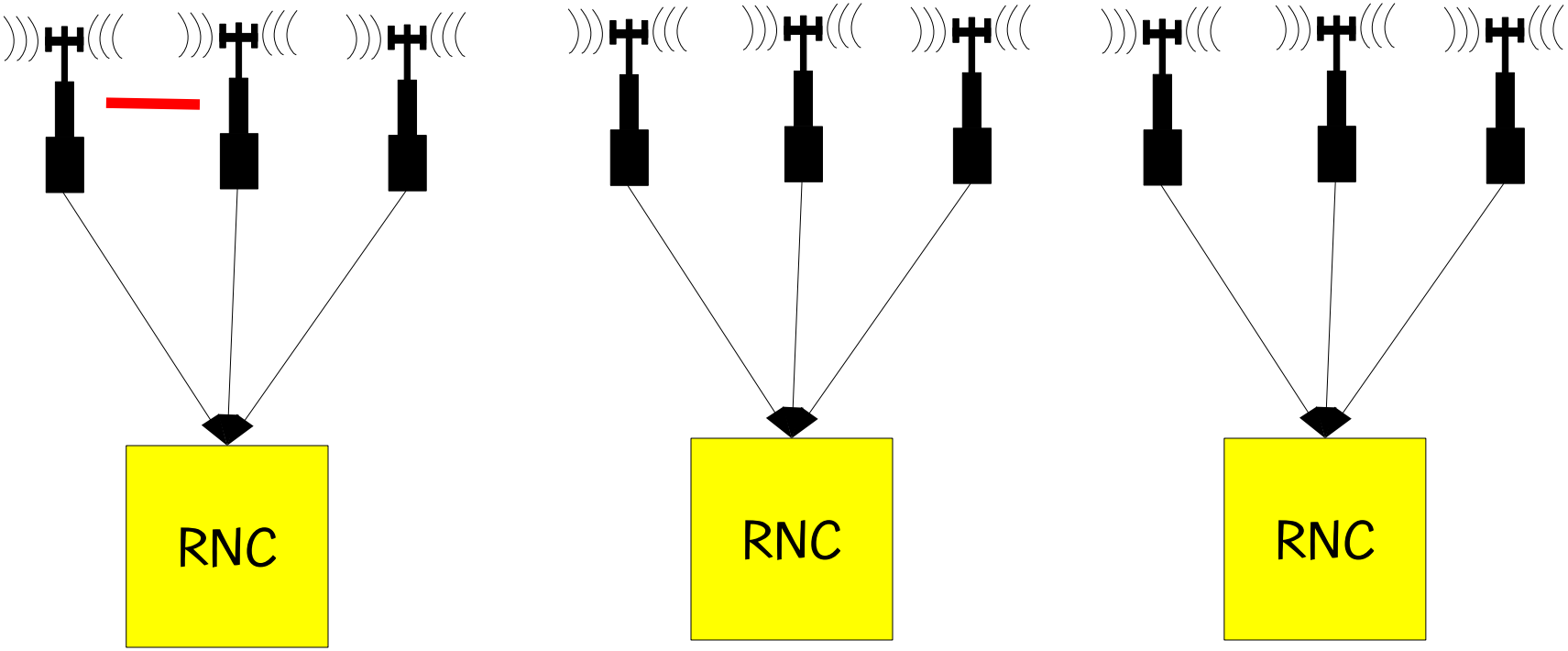




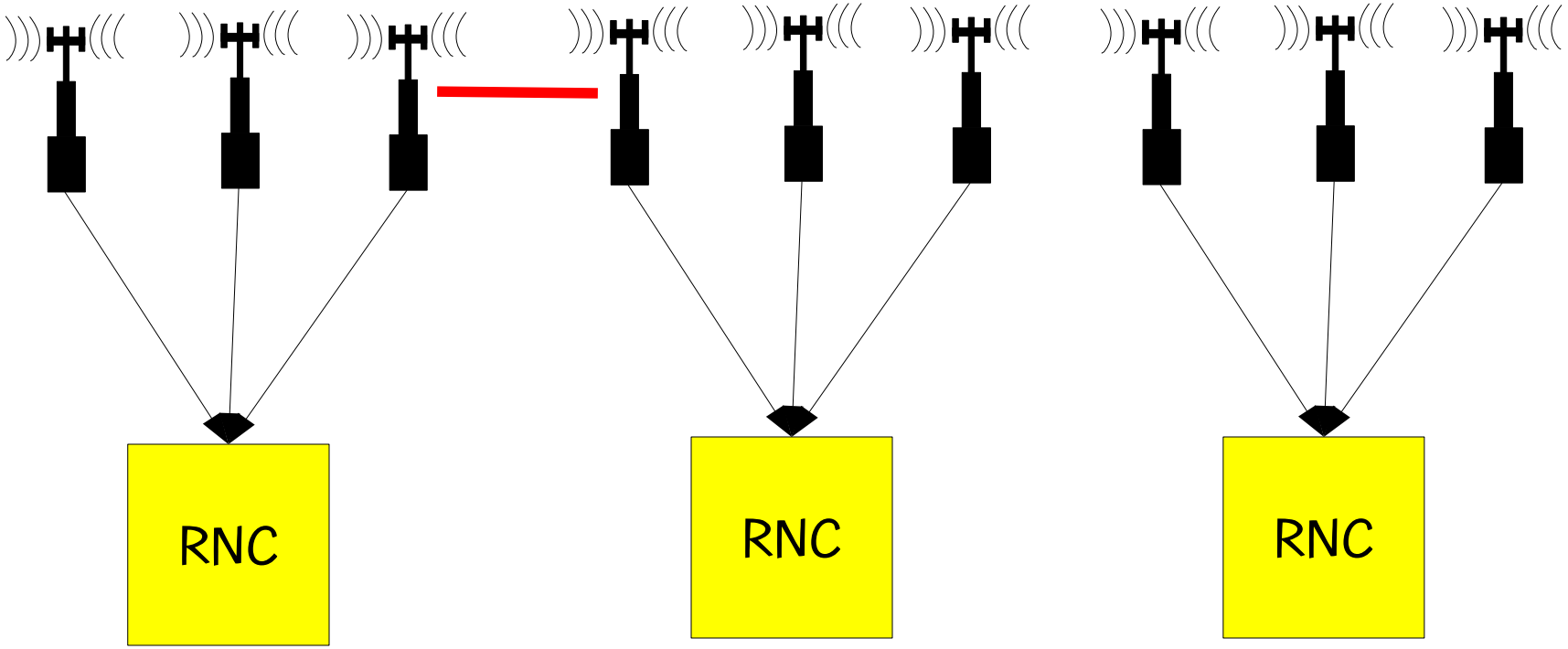
- Each base station has associated with it an amount of traffic.
- Each base station is connected to a Radio Network Controller (RNC).
- Each RNC can have one or more base stations connected to it.
- Each RNC can handle a given amount of traffic … this limits the subsets of base stations that can be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



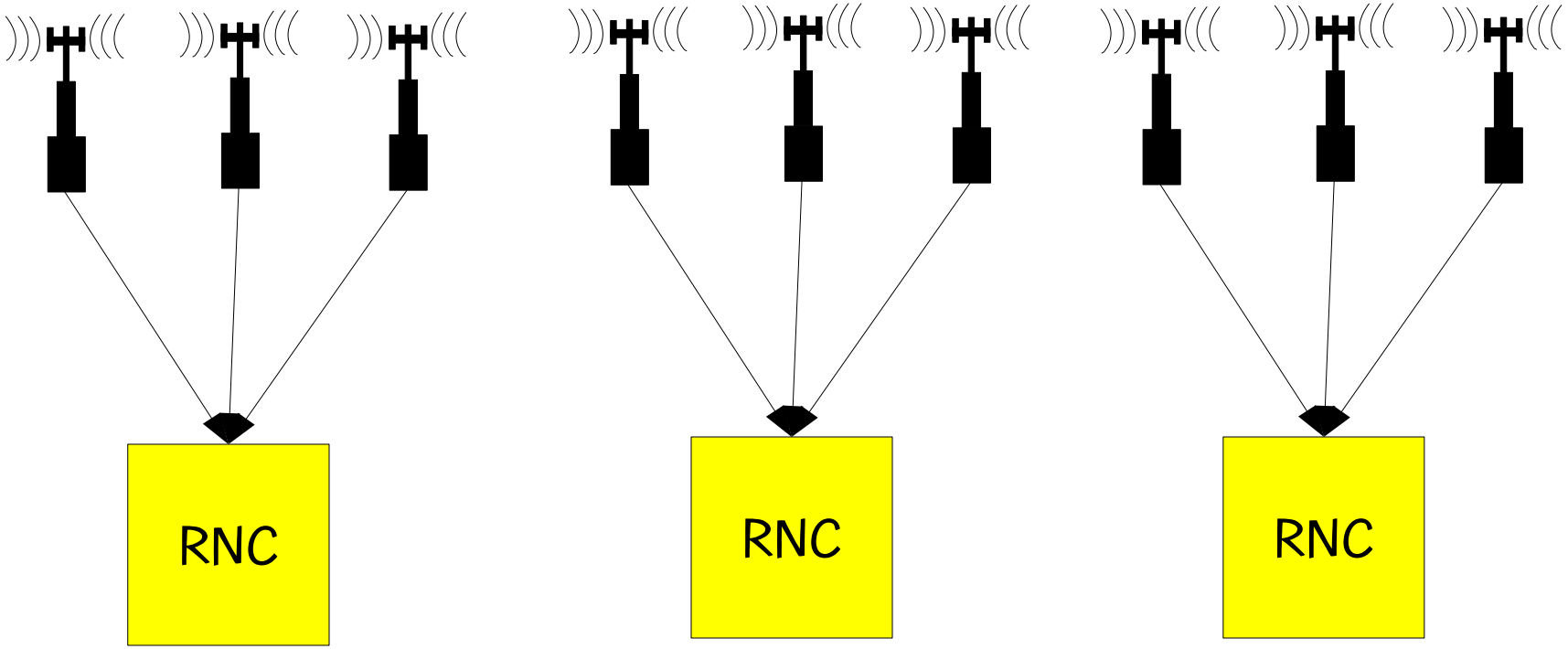
- Handovers can occur between base stations



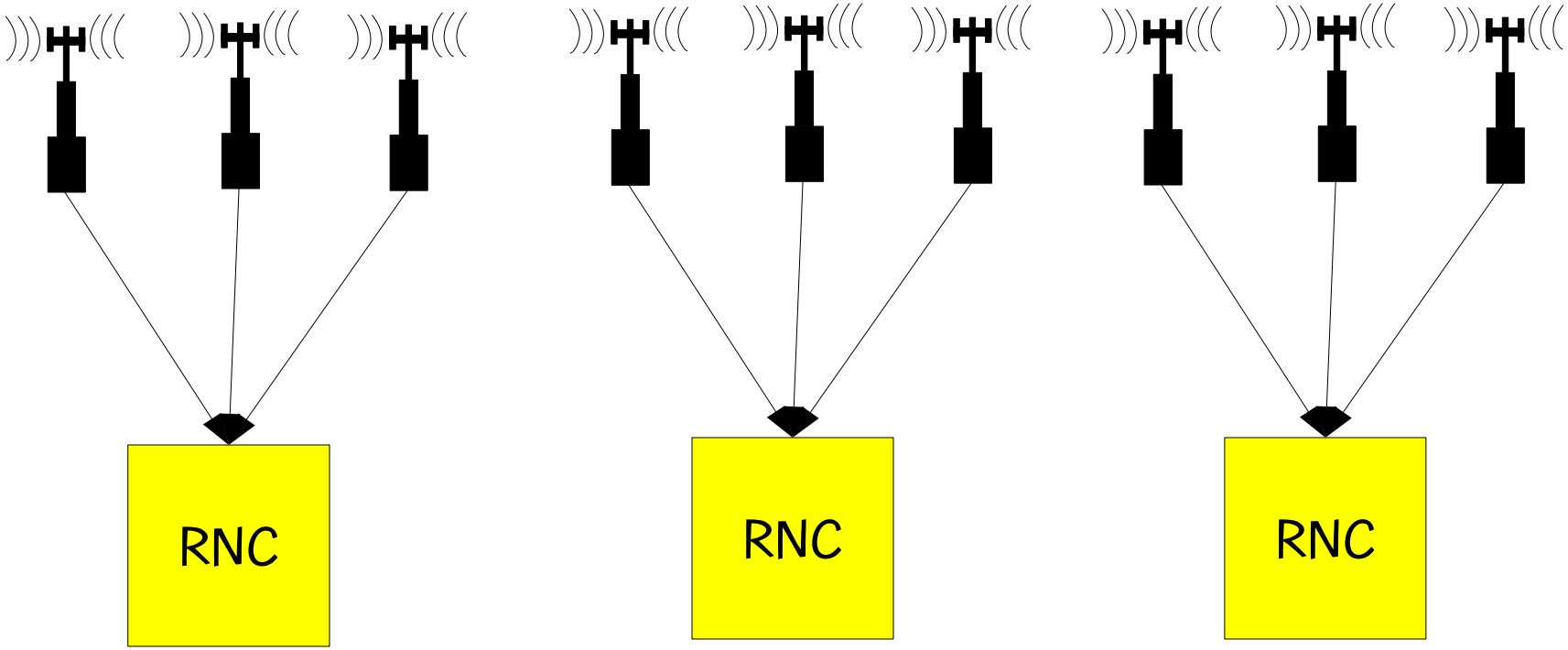
- Handovers can occur between base stations
  - connected to the same RNC



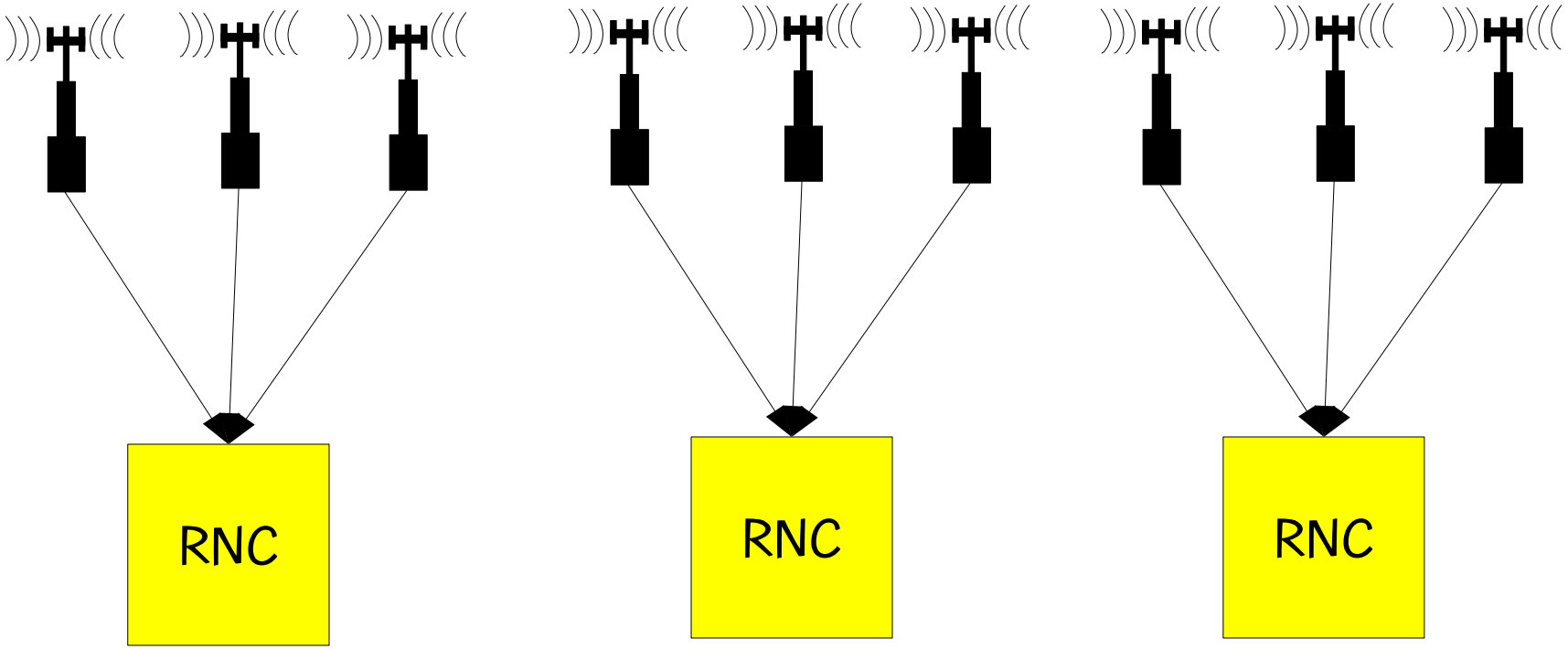
- Handovers can occur between base stations
  - connected to the same RNC
  - connected to different RNCs



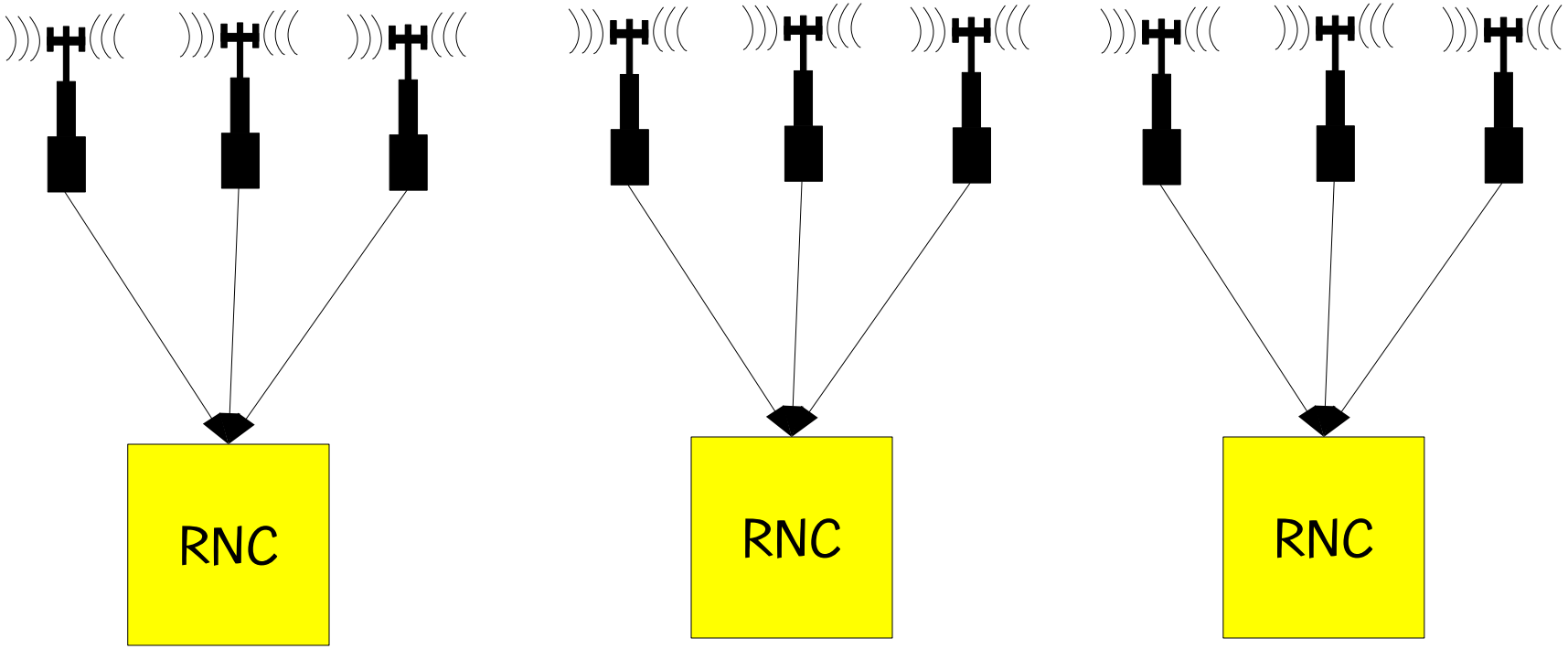
- Handovers between base stations connected to different RNCs tend to fail more often than handovers between base stations connected to the same RNC.
- Handover failure results in dropped call!



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

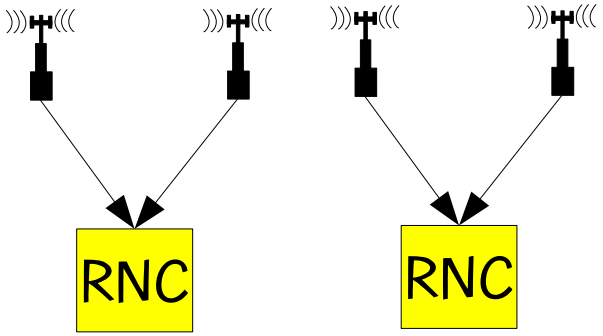


- HANDOVER MINIMIZATION: Assign base stations to RNCs such that RNC capacity is not violated and number of handovers between base stations assigned to different RNCs is minimized.



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

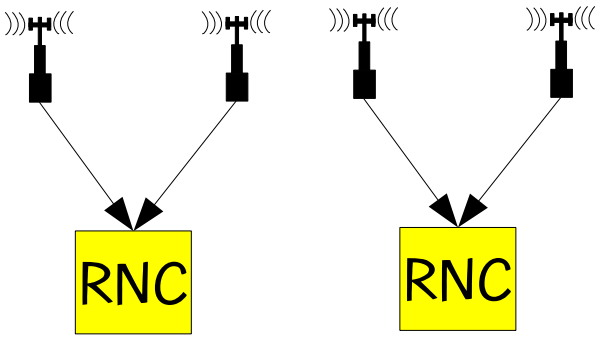
Node-capacitated graph partitioning problem



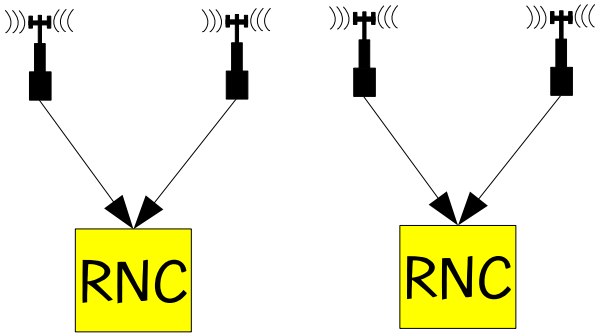
Example

- 4 BSs: 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

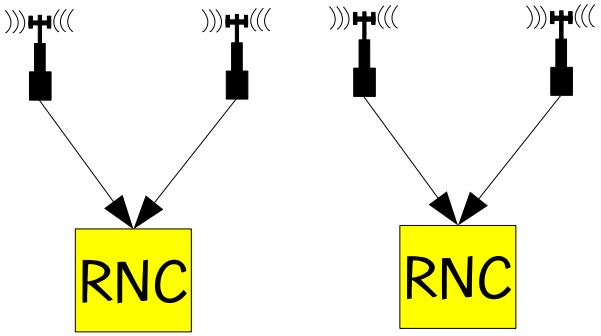


- 4 BSs: t(1) = 25; t(2) = 15; t(3) = 35; t(4) = 25
- 2 RNCs: c(1) = 50; c(2) = 60
- Given this traffic profile and RNC capacities the feasible configurations are:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }
  - RNC(1): { 1, 4 };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



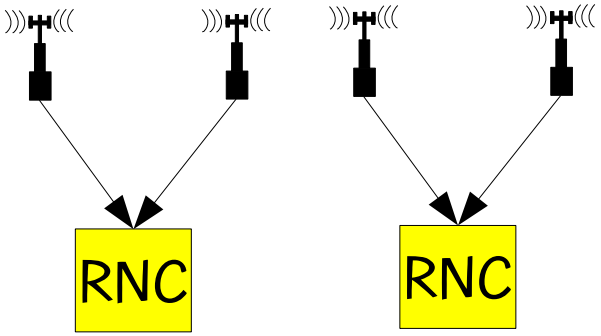
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- Total handover for each configuration:



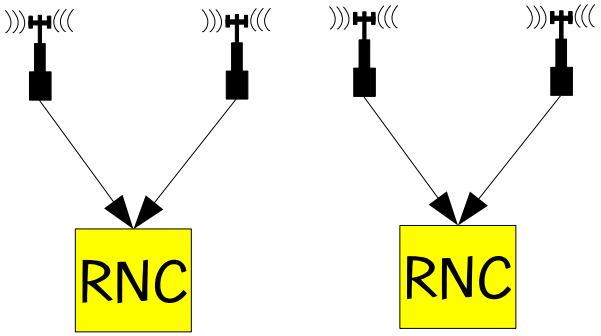
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- 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



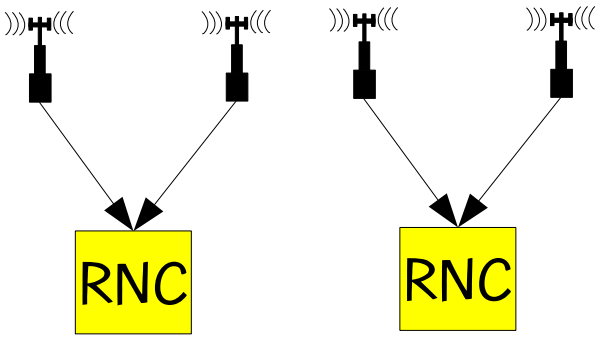
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- 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



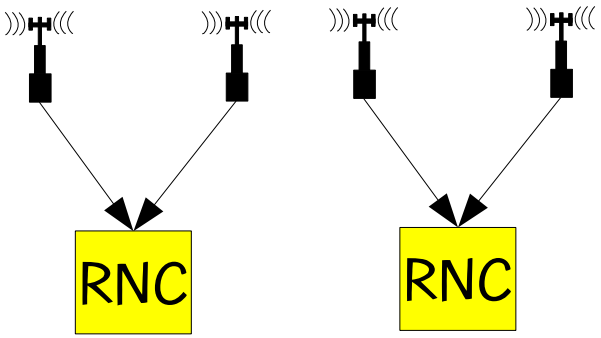
	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

- 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

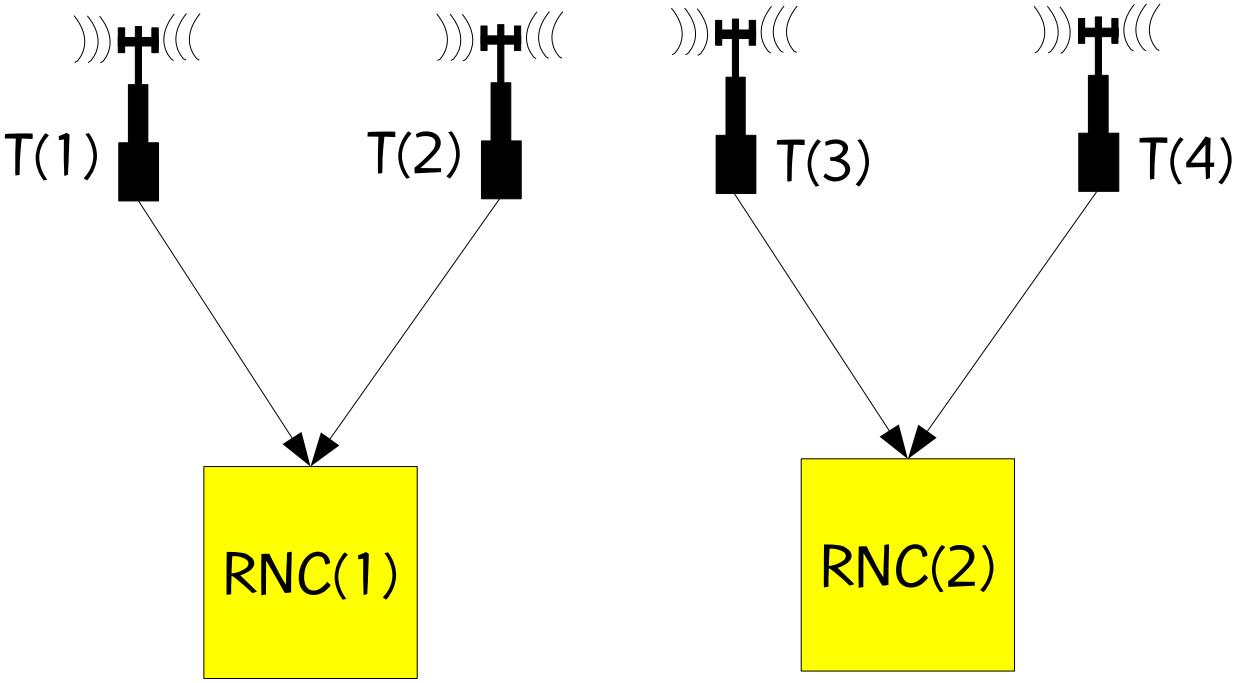


	1	2	3	4
1	0	100	10	0
2	100	0	200	50
3	10	200	0	500
4	0	50	500	0

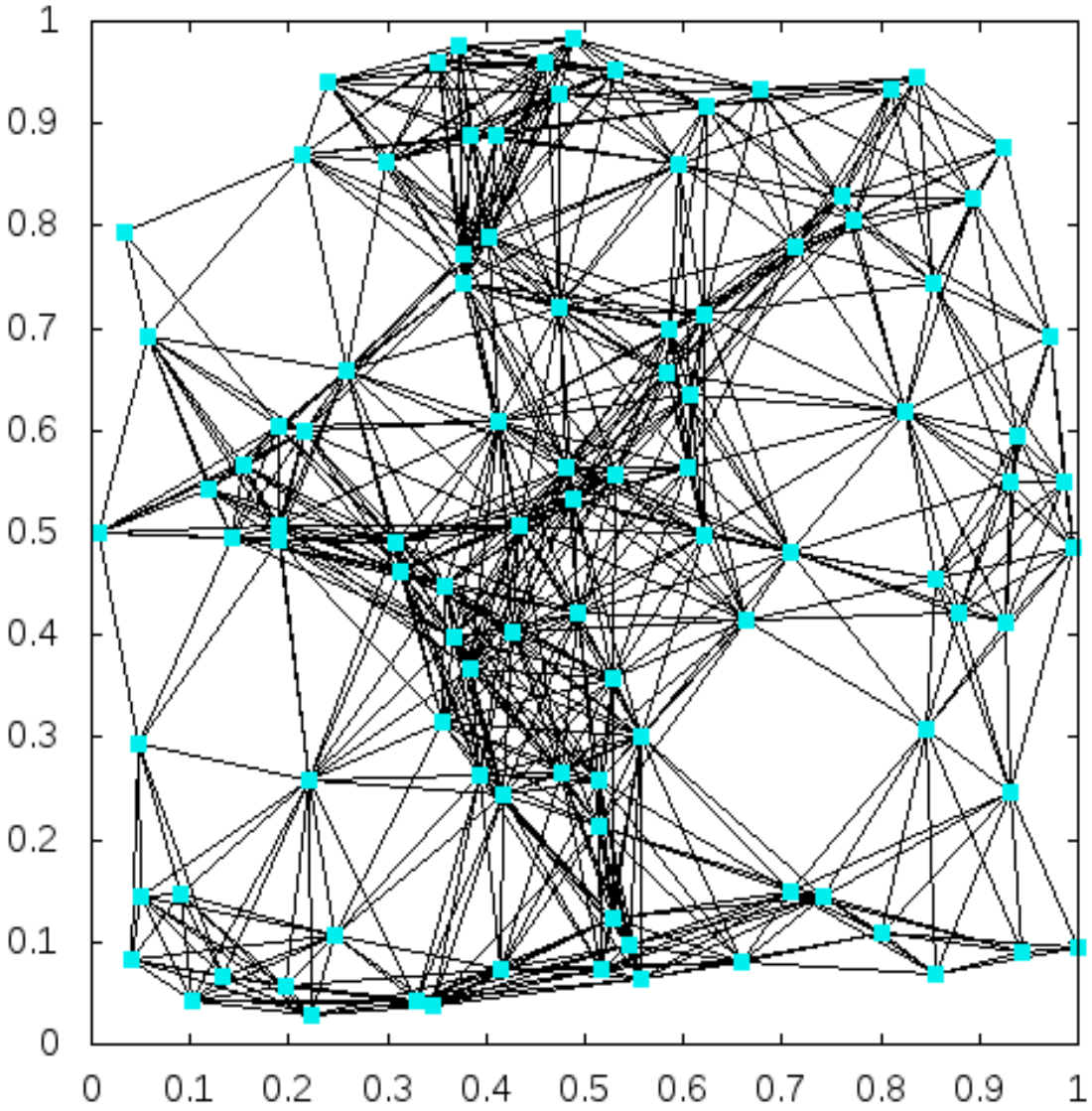
- Total handover for each configuration:
  - RNC(1): { 1, 2 }; RNC(2): { 3, 4 }: h(1,3) + h(1,4) + h(2,3) + h(2,4) = 10 + 0 + 200 + 50 = **260**
  - RNC(1): { 2, 3 }; RNC(2): { 1, 4 }: h(2,1) + h(2,4) + h(3,1) + h(3,4) = 100 + 50 + 10 + 500 = 660
  - RNC(1): { 2, 4 }; RNC(2): { 1, 3 }: h(2,1) + h(2,3)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

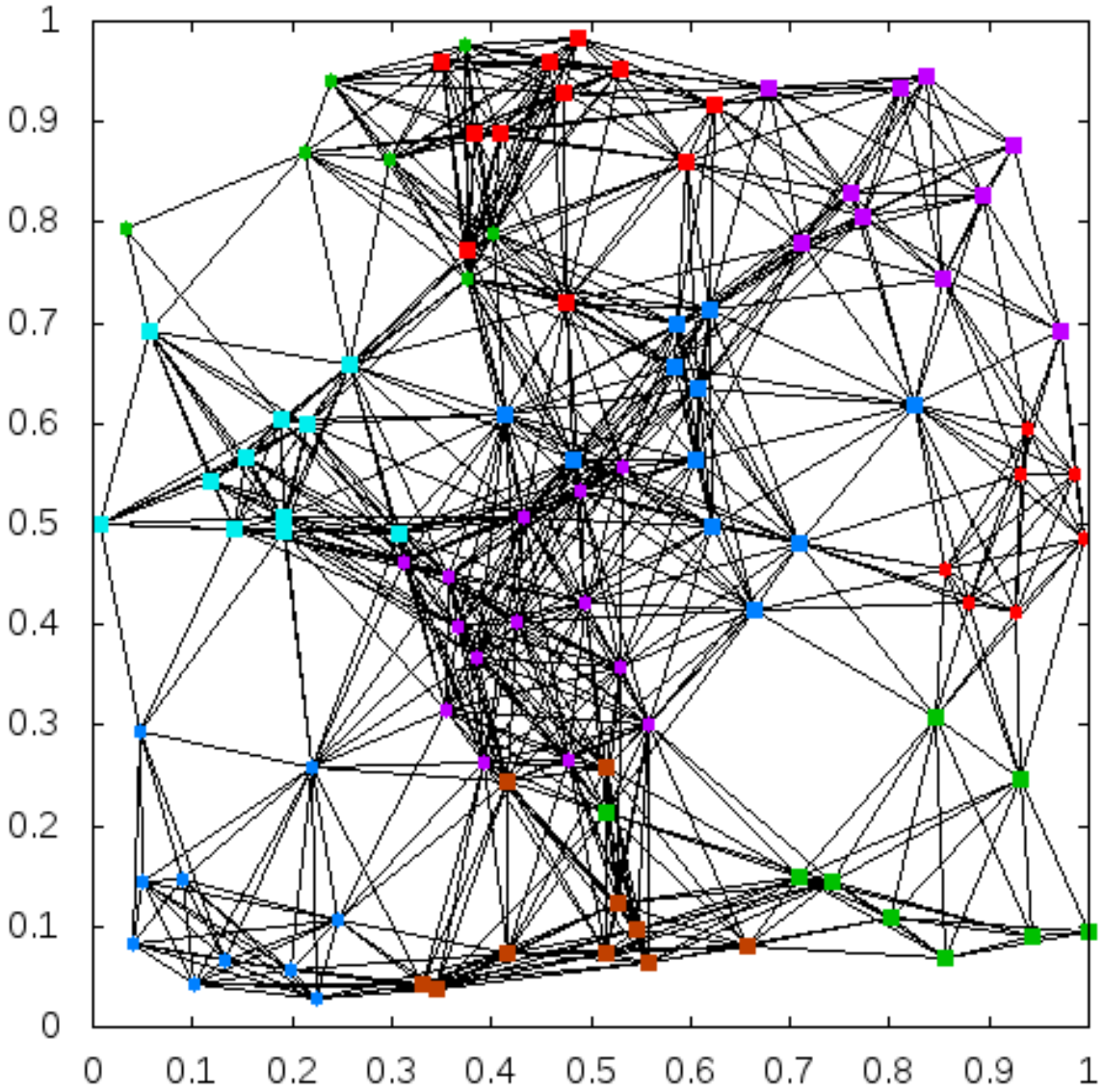
Optimal configuration:



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



Tower are assigned to RNCs indicated by distinct colors/shapes



CPLEX MIP solver

Towers	RNCs	BKS	CPLEX	time (s)
20	10	7602	7602	18.8
30	15	18266	18266	25911.0
40	15	29700	29700	101259.9
100	15	19000	49270	1 day
100	25	36412	58637	1 day
100	50	60922	70740	1 day

CPLEX MIP solver

Towers	RNCs	BKS	CPLEX	time (s)
20	10	7602	7602	18.8
30	15	18266	18266	25911.0
40	15	29700	29700	101259.9
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	18.8
30	15	18266	18266	25911.0
40	15	29700	29700	101259.9
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!

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 BS-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 BS to RNC
  - while RNC can accommodate an unassigned BS
    - use heuristic B to assign next BS to RNC
- If all available RNCs have been opened and some BS 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 BS to RNC

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

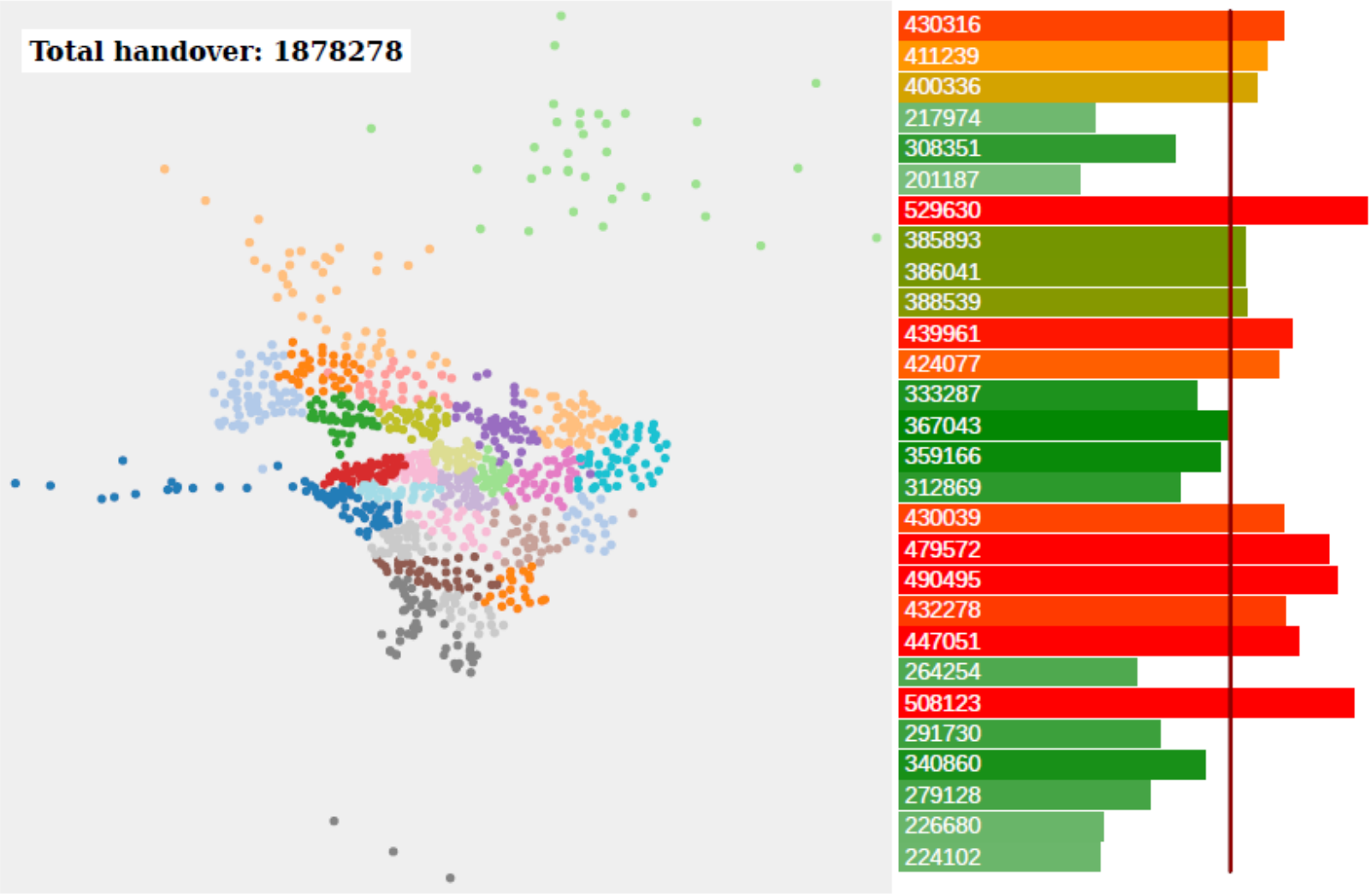
Randomized greedy construction:
Heuristic B to assign remaining BSs to RNC

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

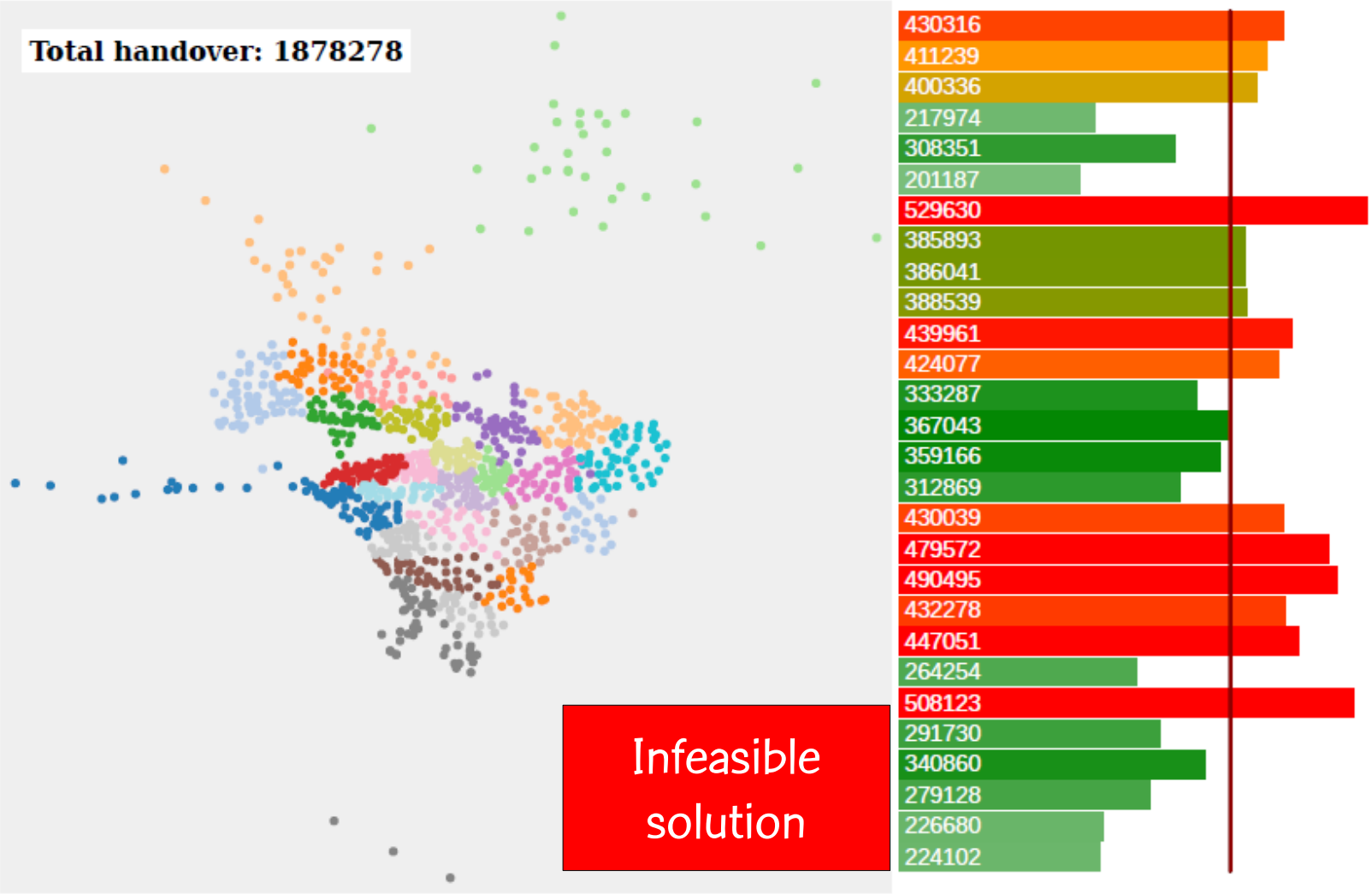
Local search

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

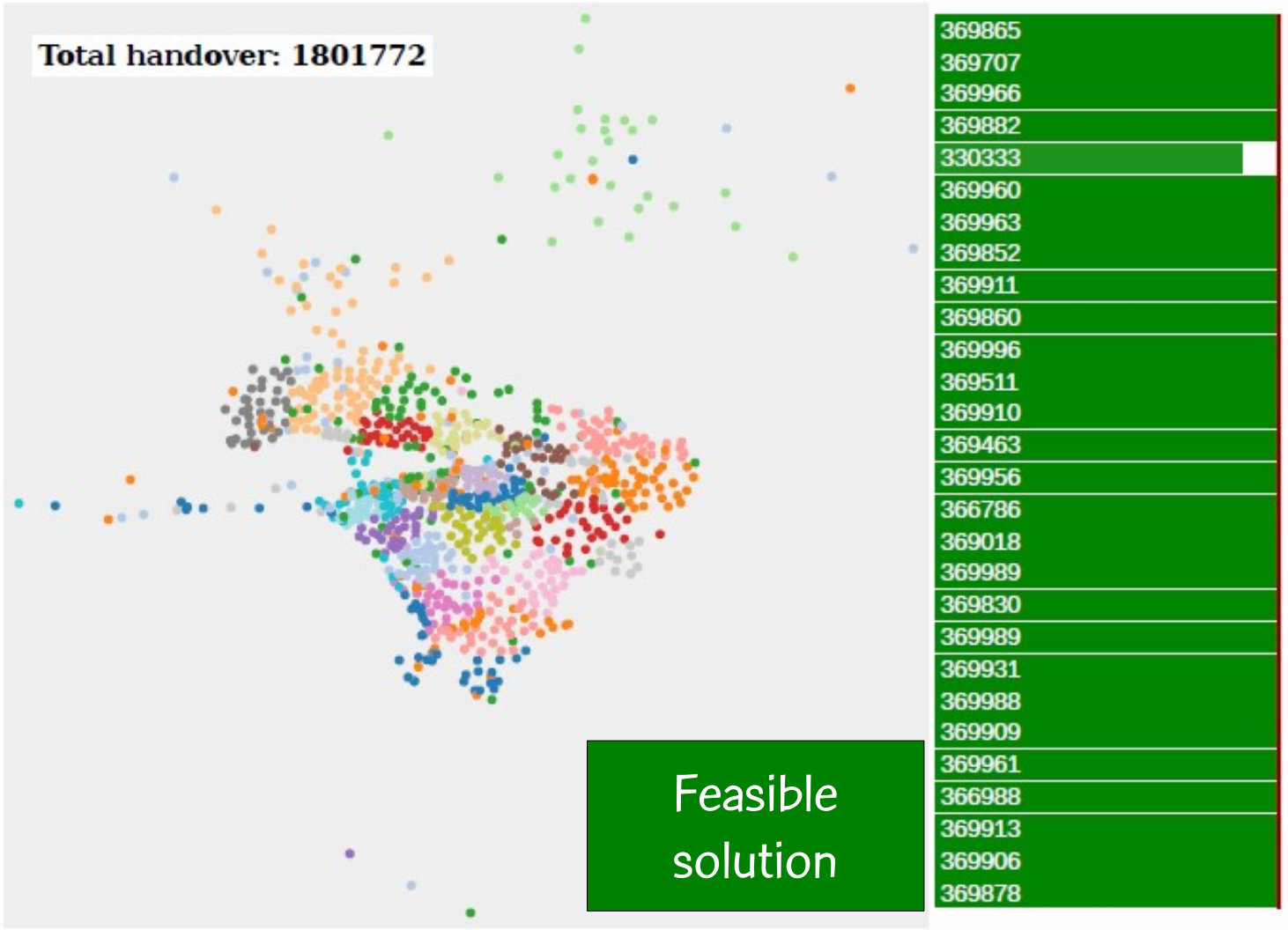
Real instance with about 1000 towers and 30 RNC: Manually produced solution



Real instance with about 1000 towers and 30 RNC: Manually produced solution

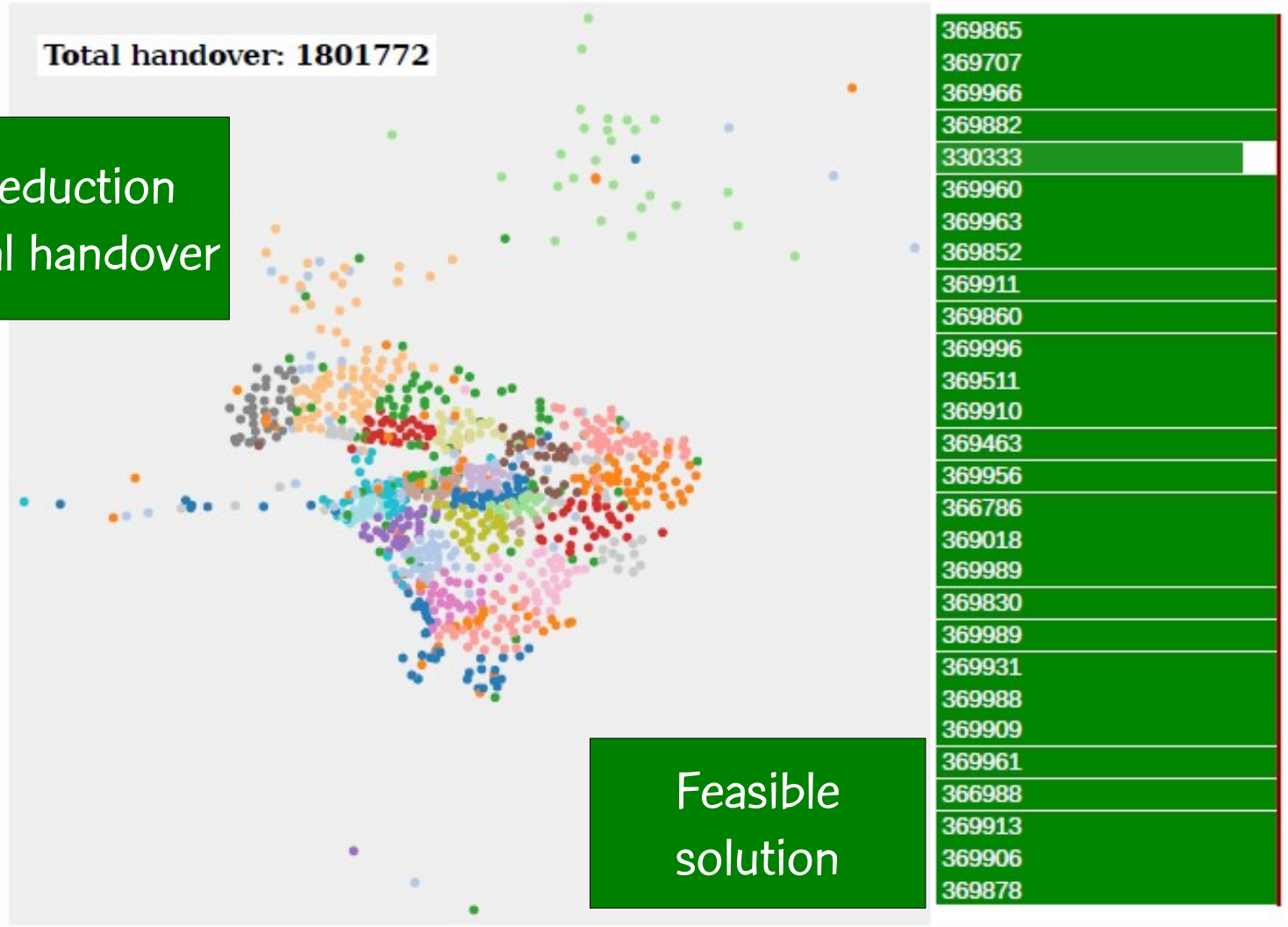


Real instance with about 1000 towers and 30 RNC: GRASP+EvPR solution



Real instance with about 1000 towers and 30 RNC: GRASP+EvPR solution

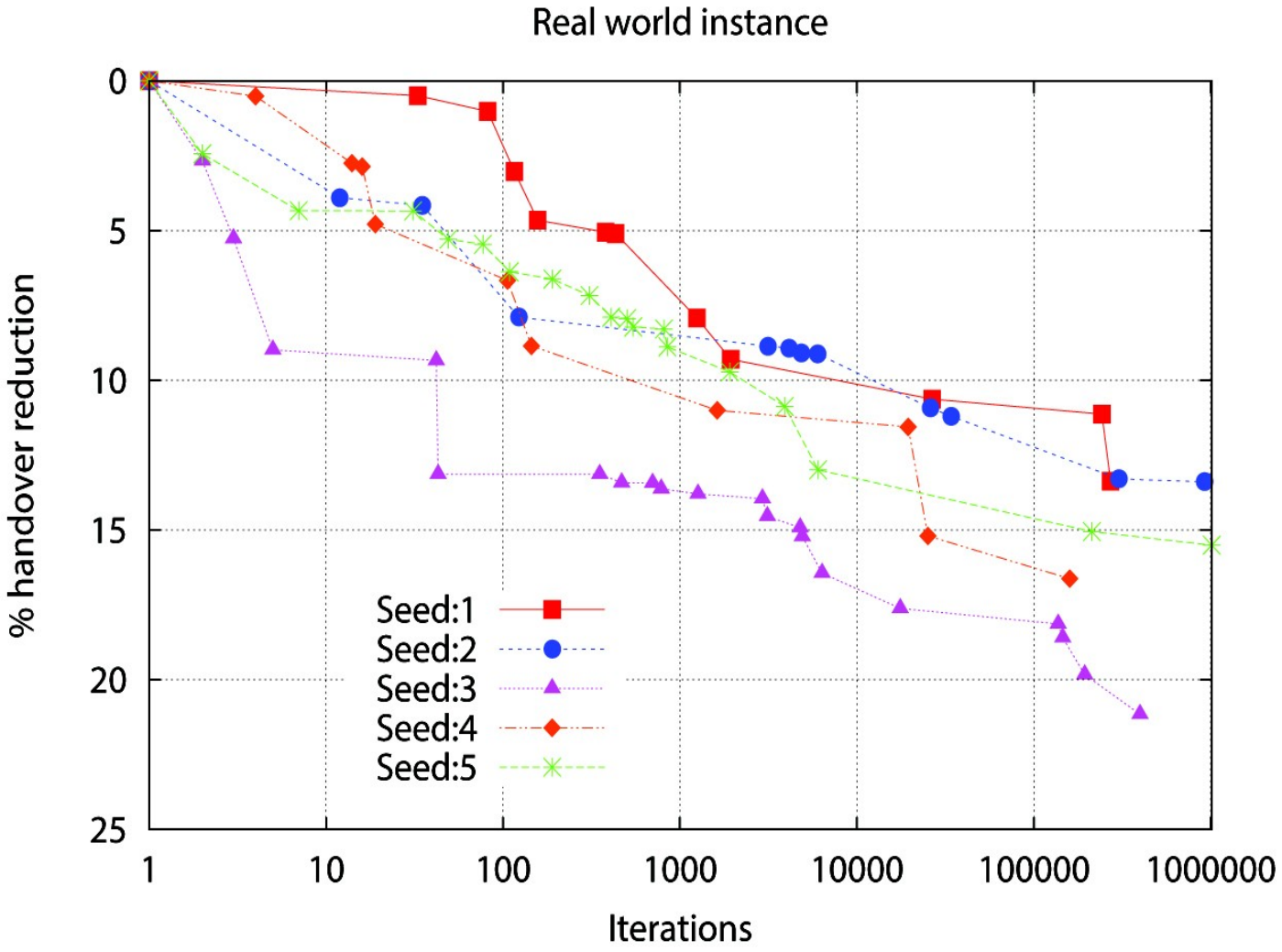
RNC capacity



4% reduction in total handover

Feasible solution

Progress of best feasible solution for five independent runs of GevPR-HMP on a real instance with about 1000 towers and 30 RNCs.



Concluding remarks

Concluding remarks

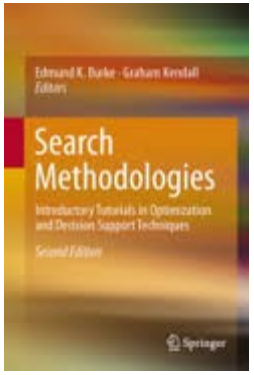
We have given a review of classical GRASP

We then showed how the main components of GRASP (randomized construction and local search) can be replaced

We showed how hybridization with path-relinking and elite sets can add memory mechanisms to GRASP

We concluded with a recent application of GRASP.

Two recent surveys of GRASP



R. and C.C. Ribeiro, "GRASP: Greedy Randomized Adaptive Search Procedures, in "Search Methodologies," 2nd edition, E.K. Burke and G. Kendall (Eds.), Chapter 11, pp. 287-310, Springer, 2014.

R. and C.C. Ribeiro, "Greedy randomized adaptive search procedures: Advances and applications," in "Handbook of Metaheuristics," 2nd edition, M. Gendreau and J.-Y. Potvin (Eds.), pp. 281-317, Springer, 2010.

Forthcoming book on GRASP

R. and C.C. Ribeiro, "Problem solving with GRASP: Greedy Randomized Adaptive Search Procedures," Springer, 2014.

The End

These slides and all papers cited in this talk
can be downloaded from my homepage:
http://mauricioresende.com