

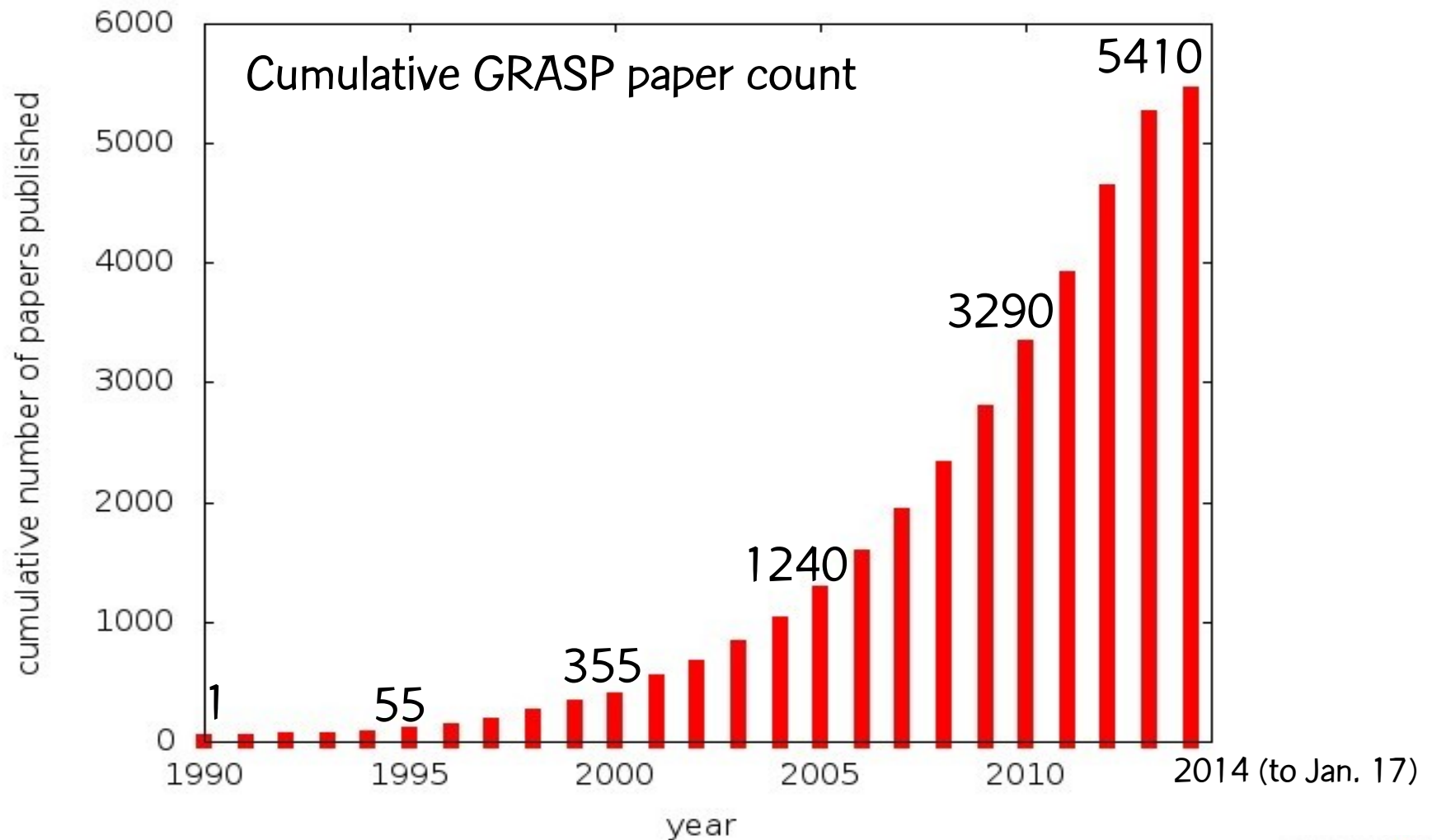
GRASP heuristics for discrete & continuous global optimization



Tutorial given at Learning and Intelligent
Optimization Conference (LION 8)
Gainesville, Florida ♣ February 17, 2014

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

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



Annotated bibliographies of GRASP

- P. Festa and M.G.C. Resende, *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 M.G.C. Resende, *An annotated bibliography of GRASP—Part I: Algorithms*, International Transactions in Operational Research, vol. 16, pp. 1-24, 2009.
- P. Festa and M.G.C. Resende, *An annotated bibliography of GRASP—Part II: Applications*, International Transactions in Operational Research, vol. 16, pp. 131-172, 2009.



Applications Places System Mauricio G. C. Resende Wed Jan 26, 8:37 AM

GRASP Metaheuristic (graspheuristic) on Twitter - Mozilla Firefox

File Edit View History Bookmarks Tools Help

<http://twitter.com/#!/graspheuristic> Google

Most Visited Getting Started Latest BBC Headli...

Google GRASP Search Bookmarks Translate AutoFill GRASP mresende

Gmail - Inbox (3580) - mrese... GRASP Metaheuristic (grasp... Engineer: Home Address: R... How to Take a Screenshot in...

Welcome to #NewTwitter! Read up on what's new. You can still access old Twitter for a limited time. [Close](#)

twitter Search Home Profile Messages Who To Follow graspheuristic

GRASP Metaheuristic
@graspheuristic
GRASP is a metaheuristic for optimization.

[Edit your profile](#)

Timeline Favorites Following Followers Lists

graspheuristic GRASP Metaheuristic
New paper on GRASP for network design for two-stage supply chain: <http://bit.ly/f9jicJ>
9 hours ago

graspheuristic GRASP Metaheuristic
New paper on a GRASP for location of phasor measurement units to find faults in electric power distribution feeders: <http://bit.ly/a5vNj>
18 Jan

graspheuristic GRASP Metaheuristic
New paper on a GRASP for a scheduling problem with periodic maintenance and sequence-dependent setup times: <http://bit.ly/fwNvHU>
18 Jan

graspheuristic GRASP Metaheuristic
New paper on GRASP for workover rig scheduling: <http://bit.ly/gmMc72>
15 Jan

graspheuristic GRASP Metaheuristic
New paper on GRASP for the single-item two-echelon...

About @graspheuristic

172 Tweets 0 Following 29 Followers 1 Listed

Following 0 Followers 29

Find accounts to follow:
[Browse interests](#) · [Find friends](#)

What's Next? · [hide next steps](#)

1. **Get Twitter on your phone**
• [Set up mobile notifications](#)

2. **Set up your profile**
• [Upload a profile picture](#)
• [Write a short bio](#)

[About](#) · [Help](#) · [Blog](#) · [Status](#) · [Jobs](#) · [Terms](#) · [Privacy](#) · [Shortcuts](#)
[Advertisers](#) · [Businesses](#) · [Media](#) · [Developers](#) · [Resources](#) · © 2011 Twitter

Done

GRA... Mozi... Gm... Do... sea... sea2... The ... zotero

Follow GRASP on Twitter:

<http://twitter.com/graspheuristic>

Rethink Possible

LION 8 – February 17, 2014

GRASP & C-GRASP

att.com/rethinkpossible

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

Continuous GRASP for bound constrained global optimization

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 $f(x): x \in X \rightarrow \mathbb{R}$

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

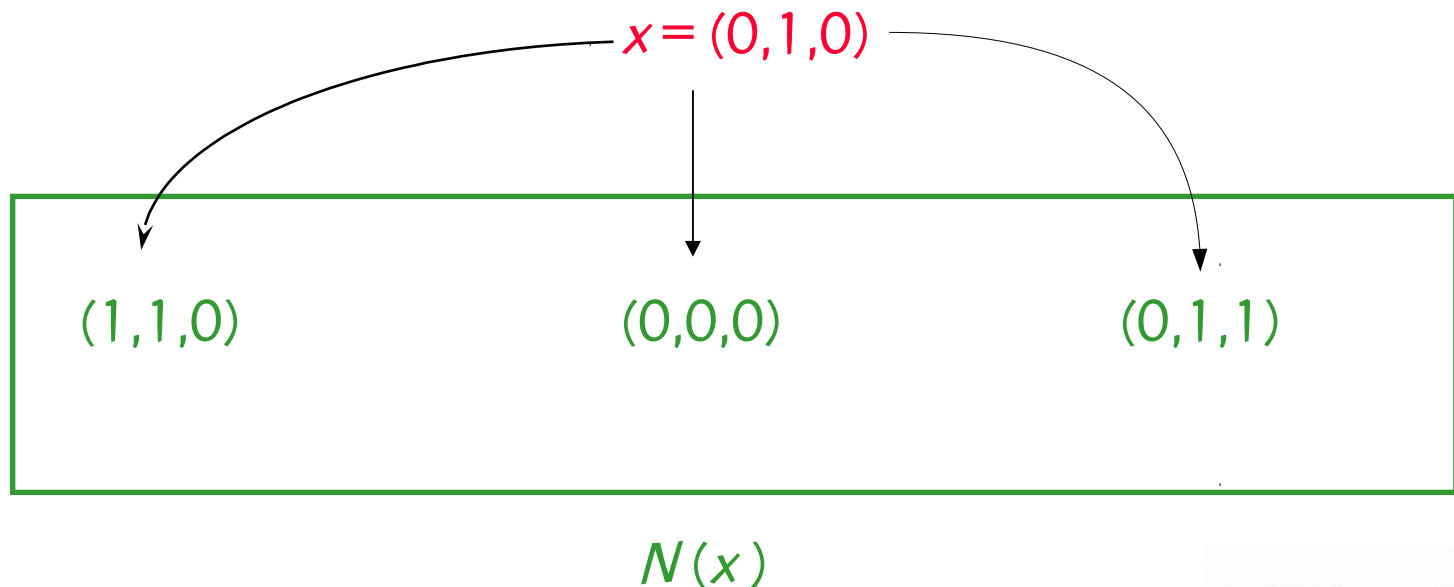
To define local search, one needs to specify a **local neighborhood structure**.

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 .



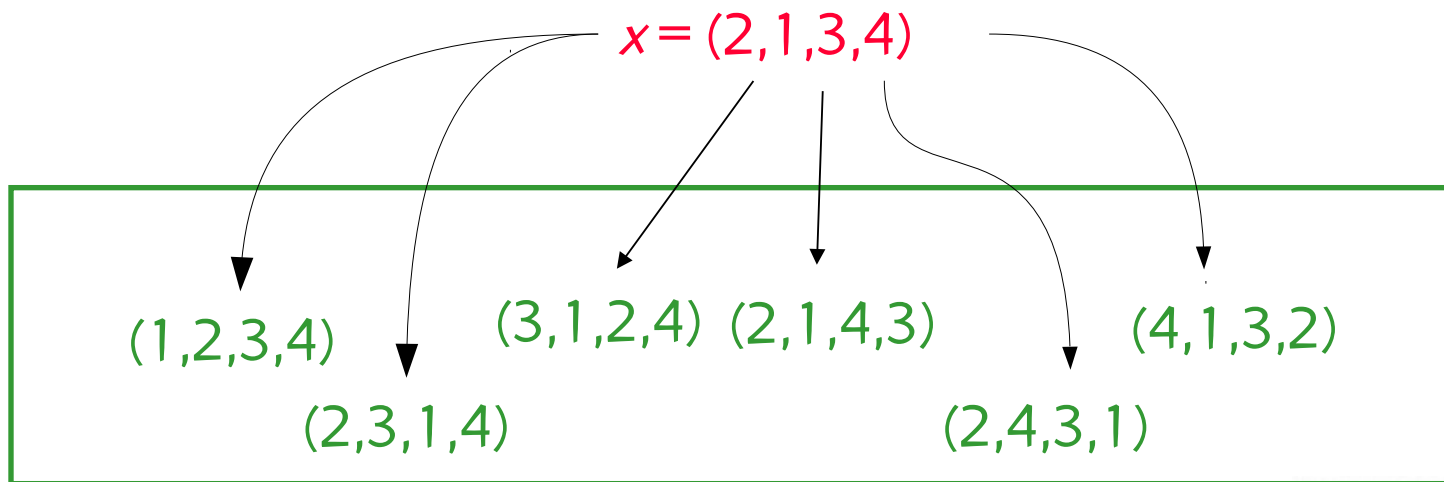
Local Search Neighborhoods

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.



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

Rethink Possible



Local Search

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

$x = x_0 ;$

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

$x = y ;$

}

check for better solution in neighborhood of x

move to better solution y

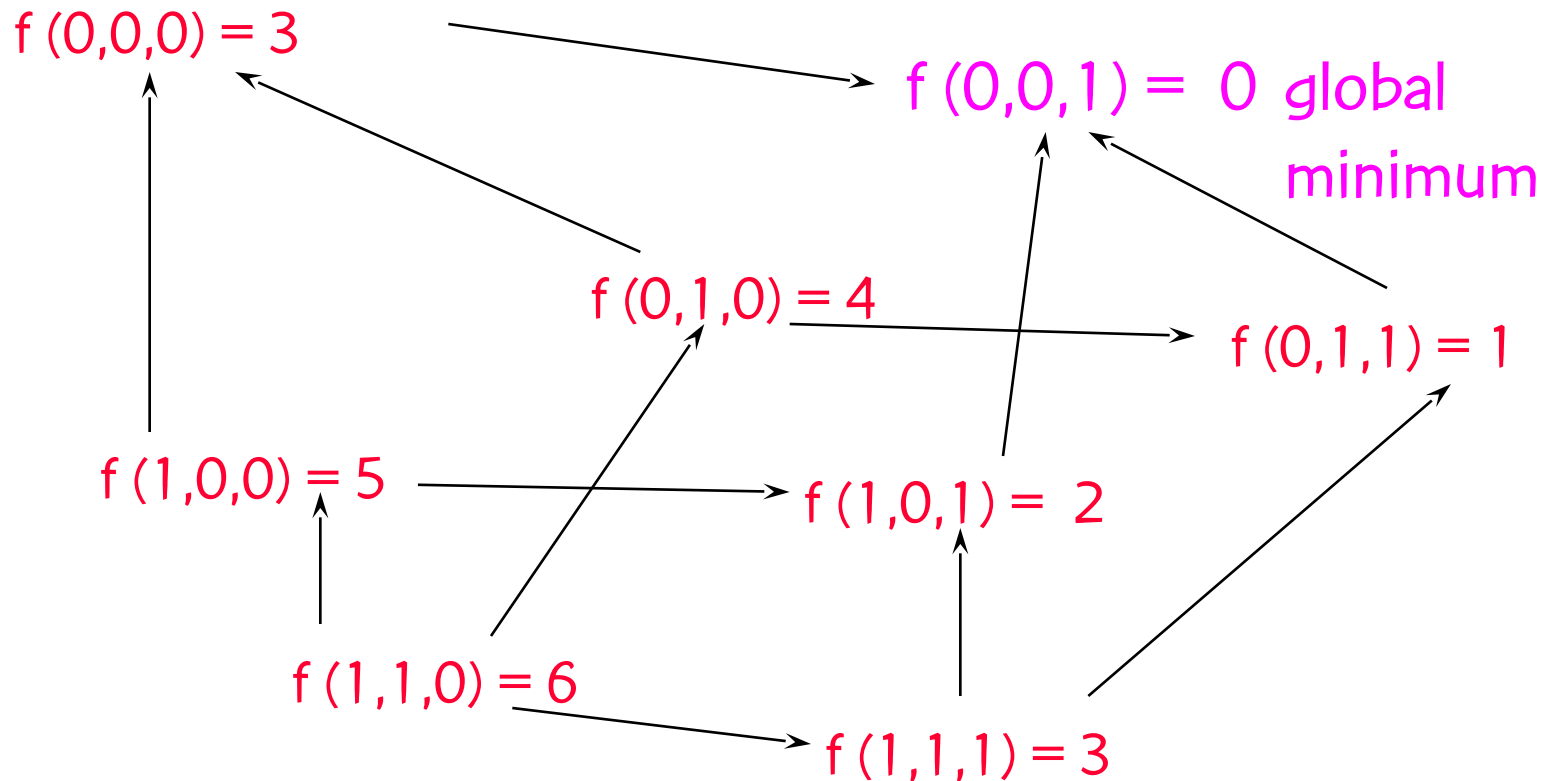
Time complexity of local search can be exponential.

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.

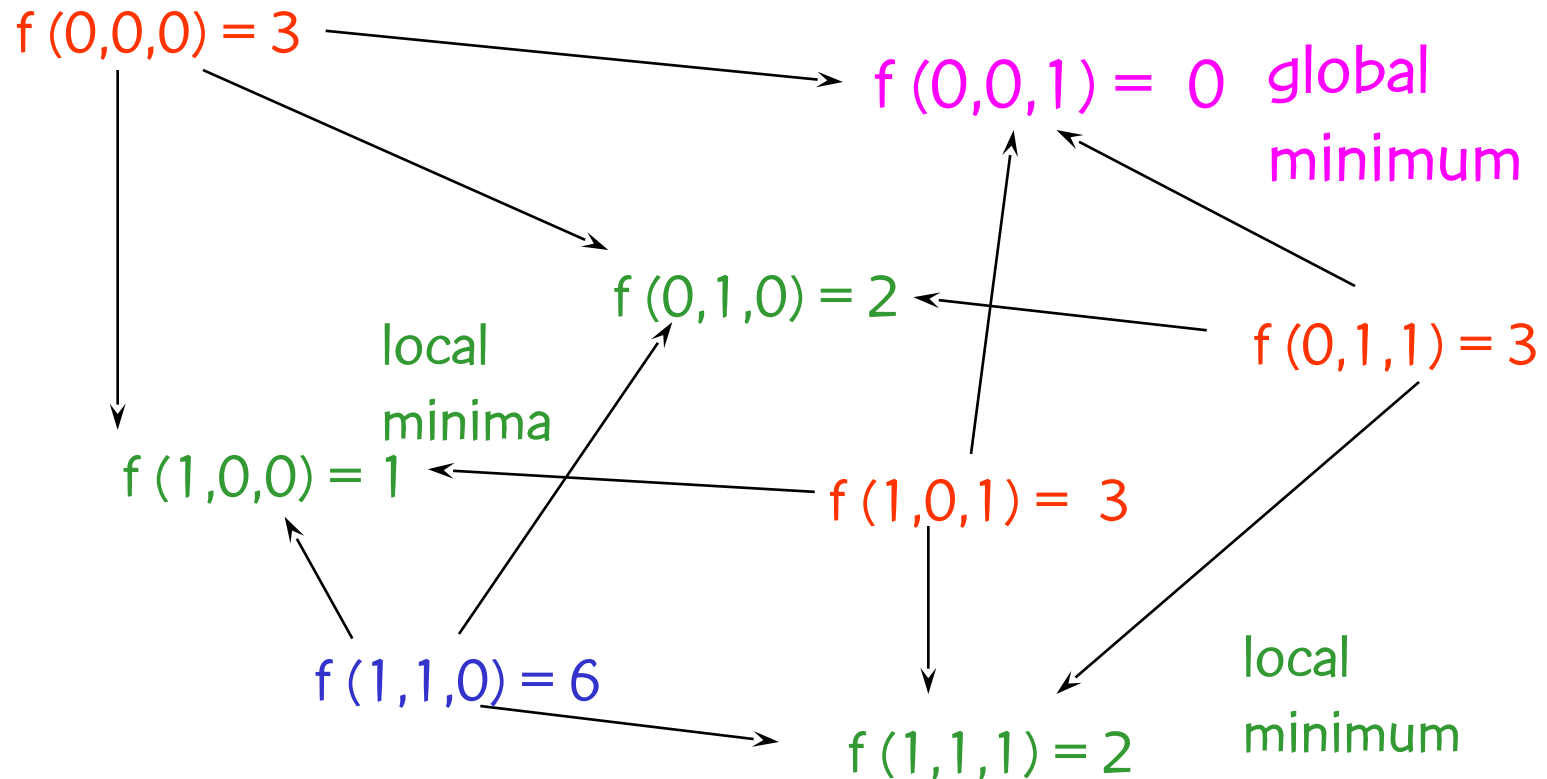
Rethink Possible



att.com/rethinkpossible

Local Search

(more realistic situation)



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

Rethink Possible



Local Search

Effectiveness of local search depends on several factors:

neighborhood structure
function to be minimized

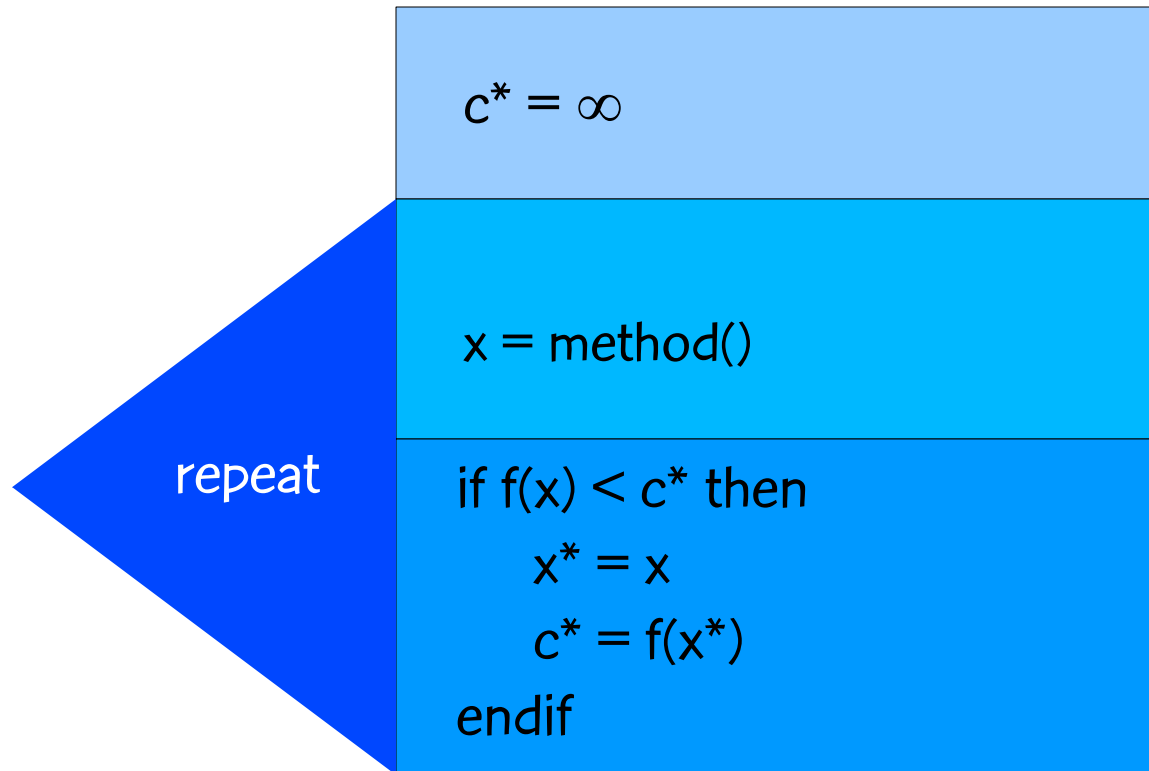
usually pre-determined

starting solution

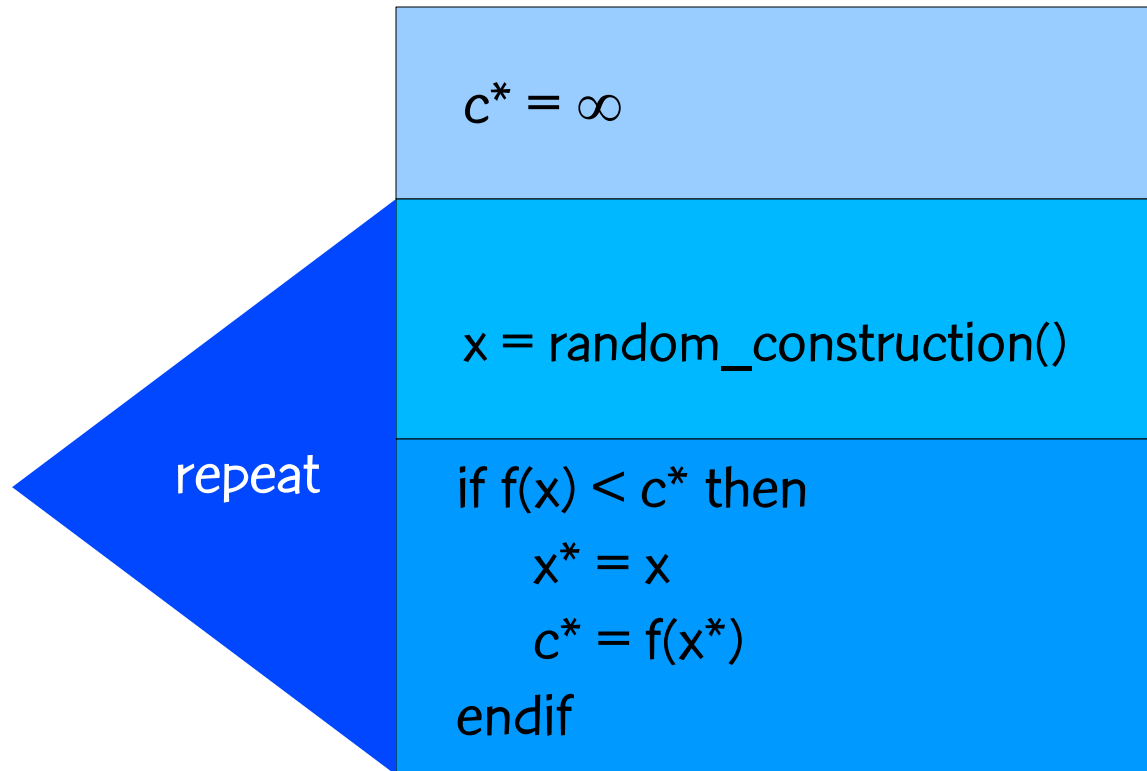
usually easier to control



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.

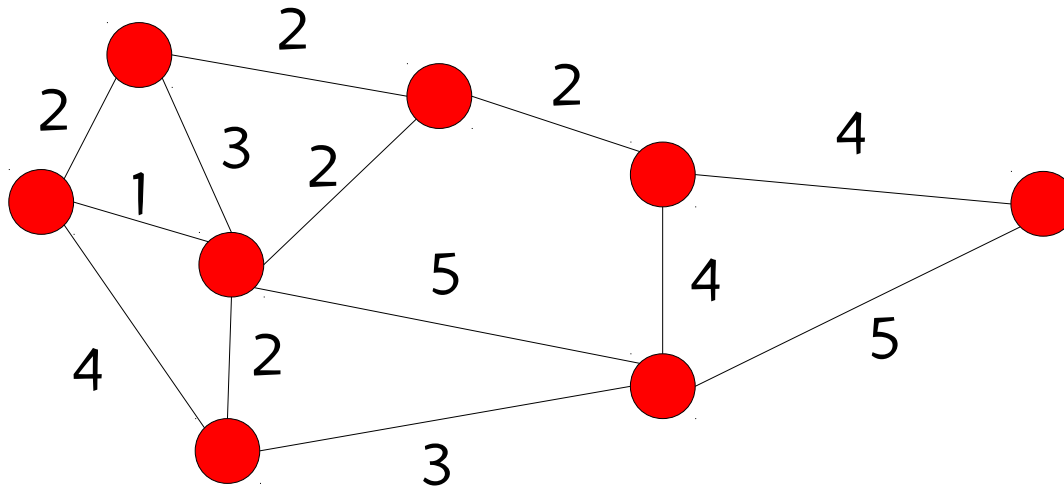
- Ranks elements according to greedy function value.

- Add best ranked element to solution.



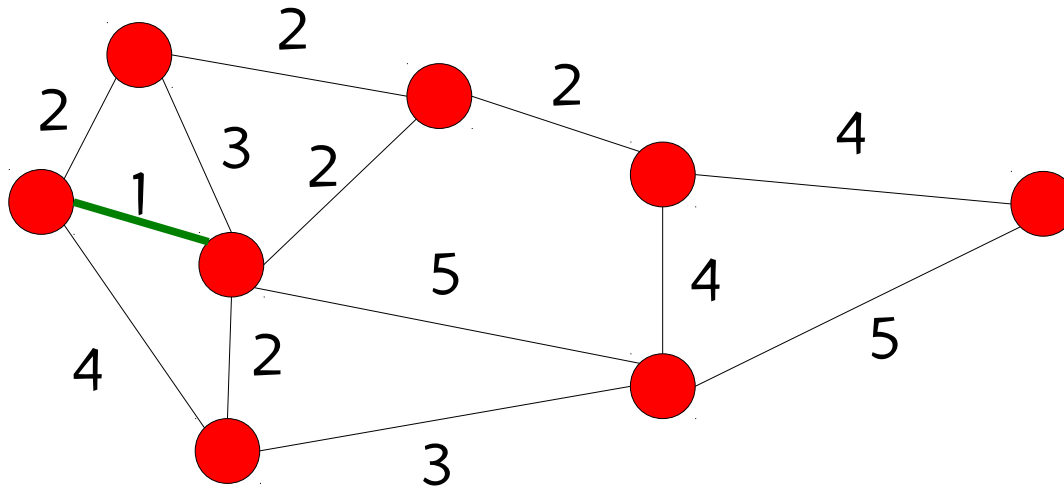
The greedy algorithm

An example: minimum weight spanning tree



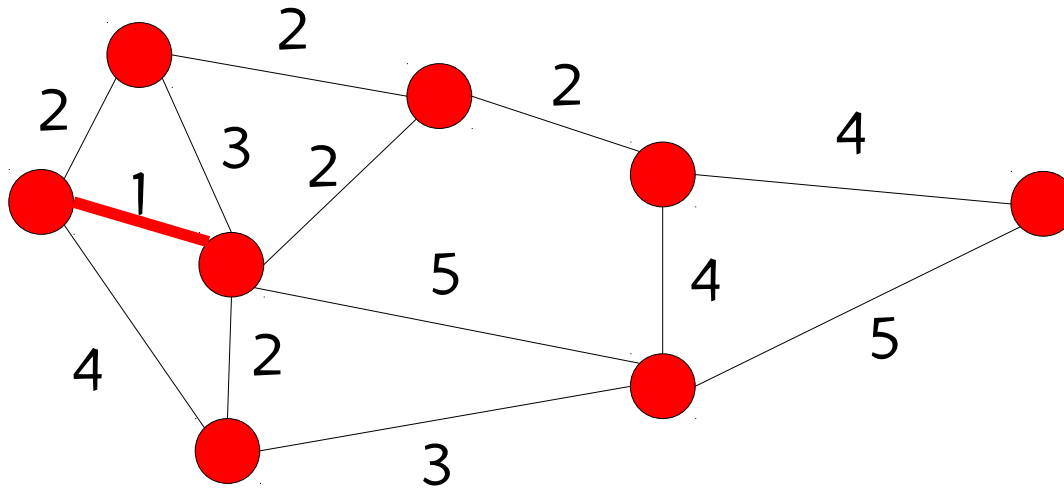
The greedy algorithm

An example: minimum weight spanning tree



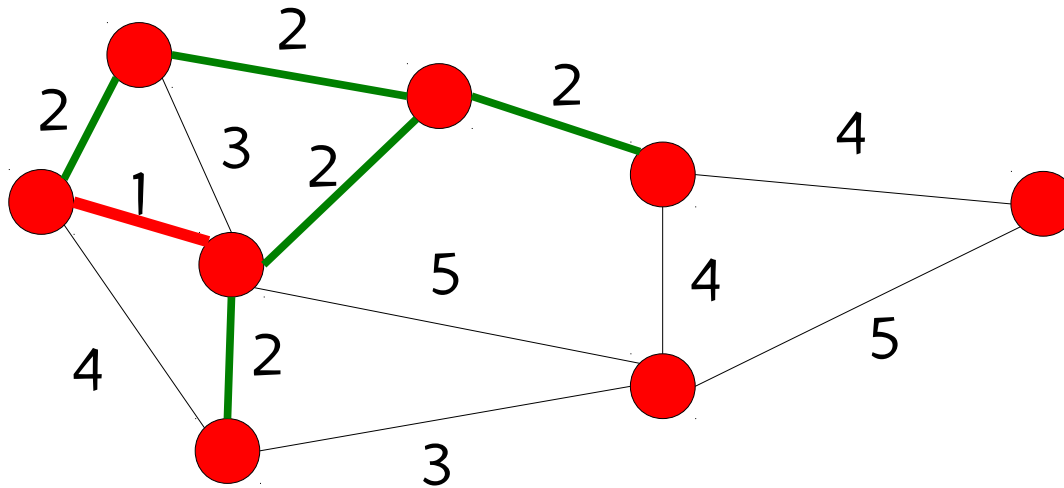
The greedy algorithm

An example: minimum weight spanning tree



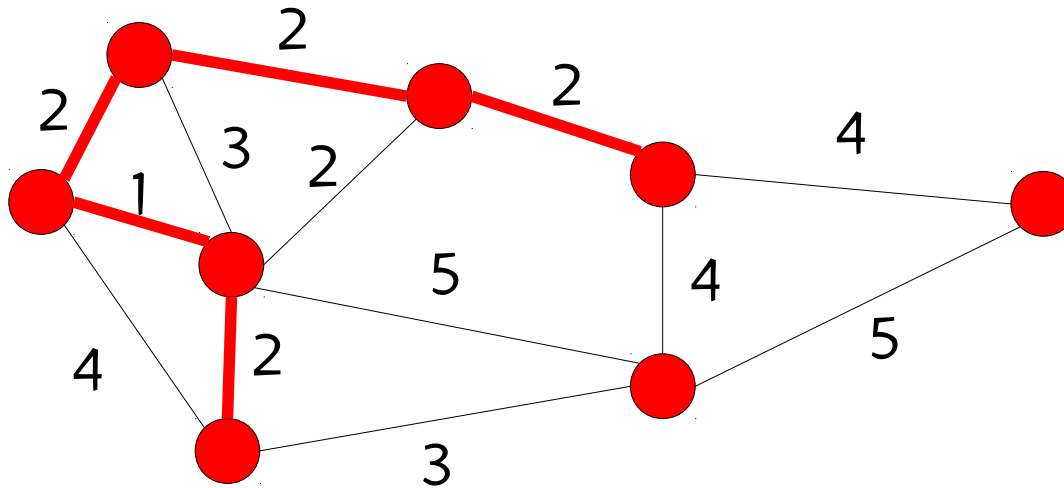
The greedy algorithm

An example: minimum weight spanning tree



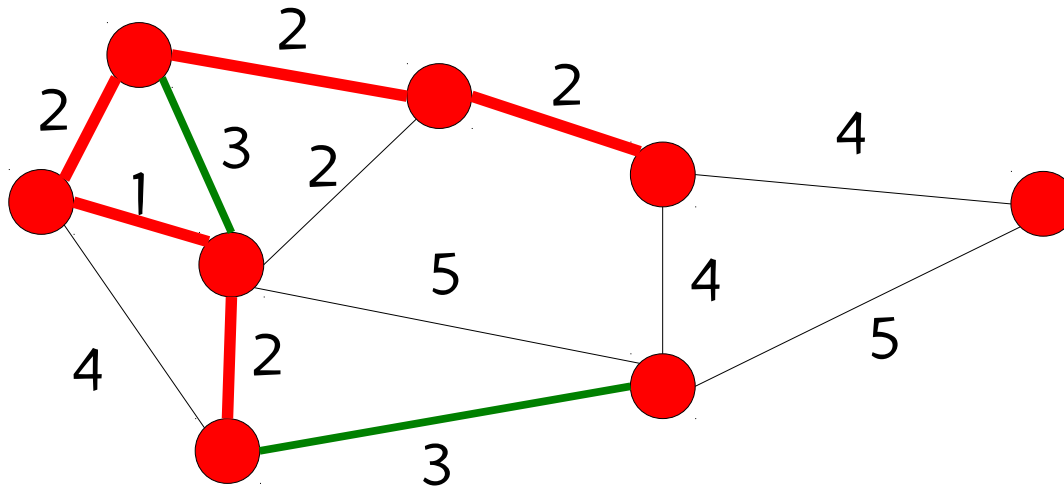
The greedy algorithm

An example: minimum weight spanning tree



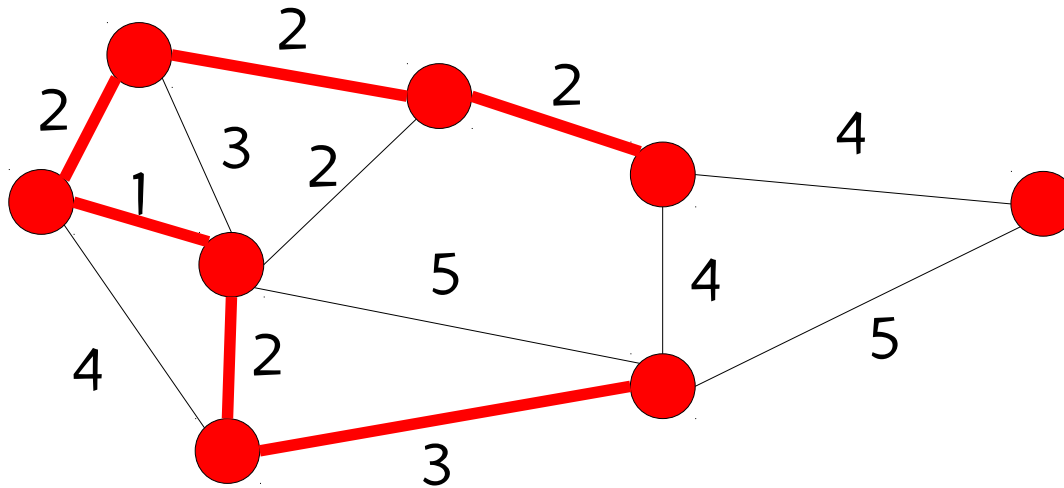
The greedy algorithm

An example: minimum weight spanning tree



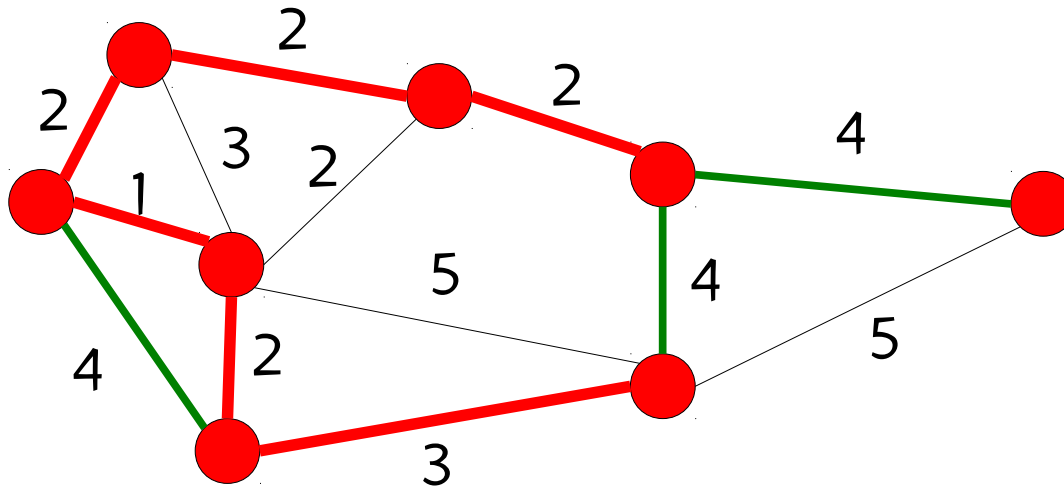
The greedy algorithm

An example: minimum weight spanning tree



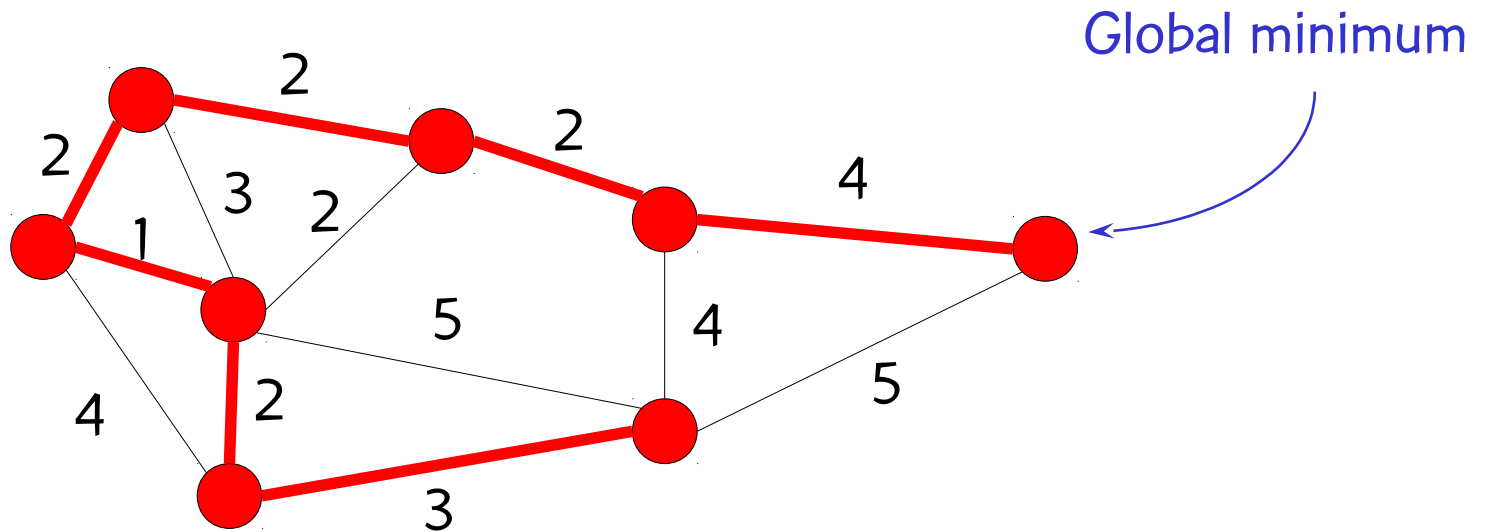
The greedy algorithm

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

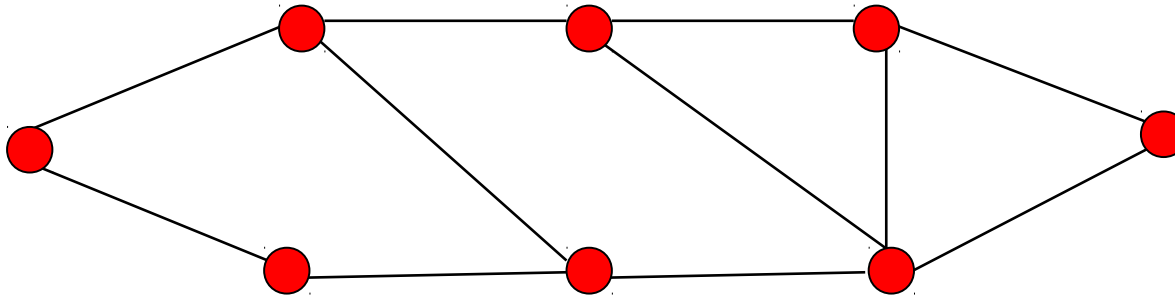
candidate set: unselected vertices adjacent to all selected vertices

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



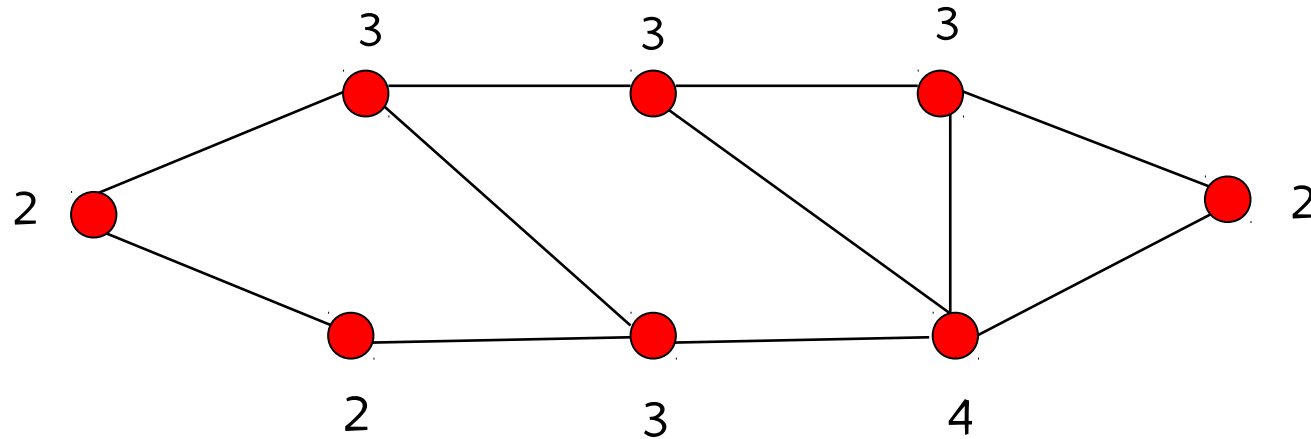
The greedy algorithm

Another example: Maximum clique



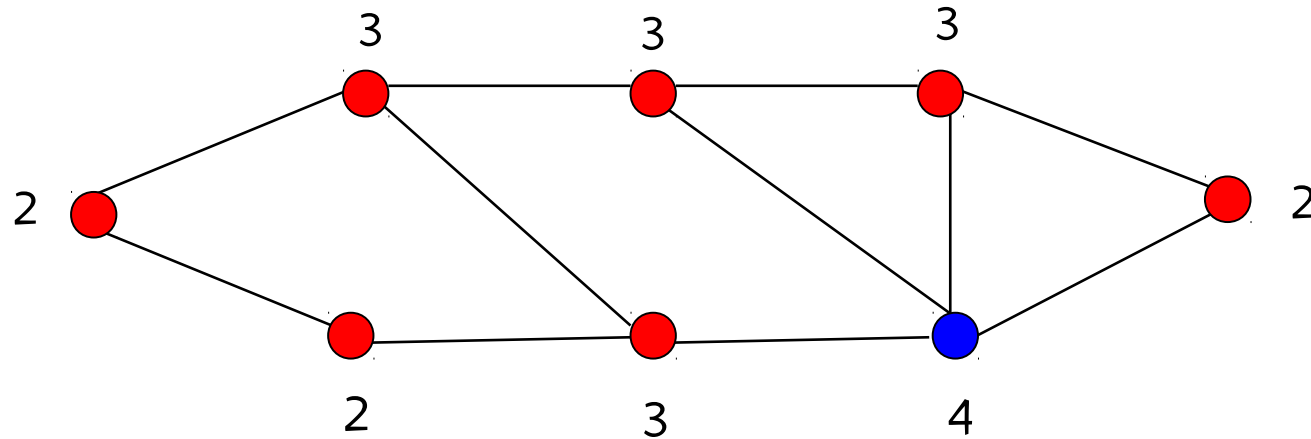
The greedy algorithm

Another example: Maximum clique



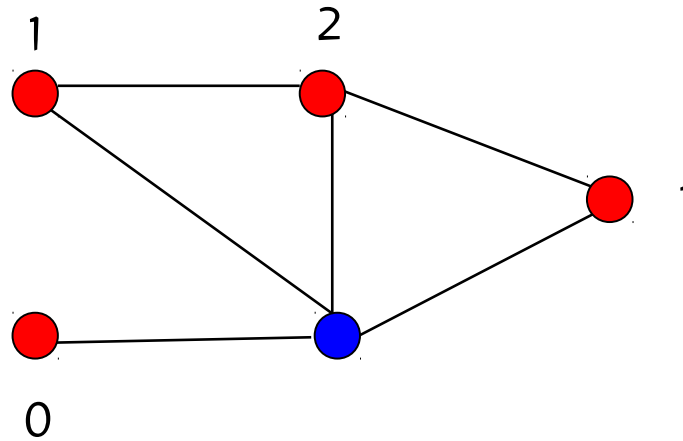
The greedy algorithm

Another example: Maximum clique



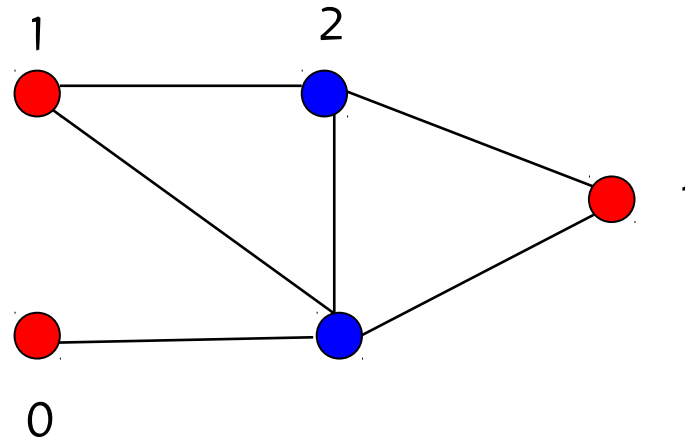
The greedy algorithm

Another example: Maximum clique



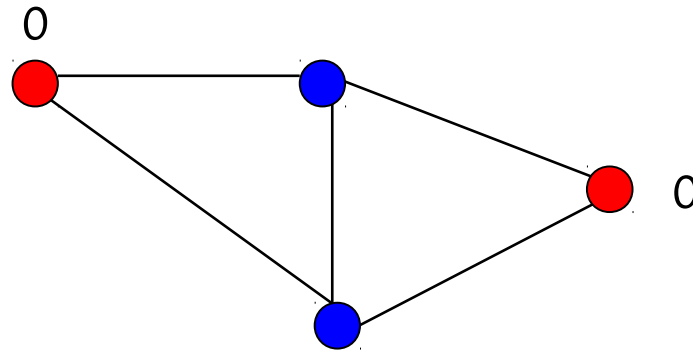
The greedy algorithm

Another example: Maximum clique



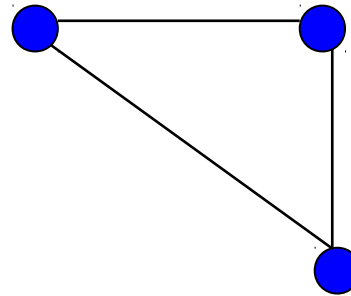
The greedy algorithm

Another example: Maximum clique



The greedy algorithm

Another example: Maximum clique

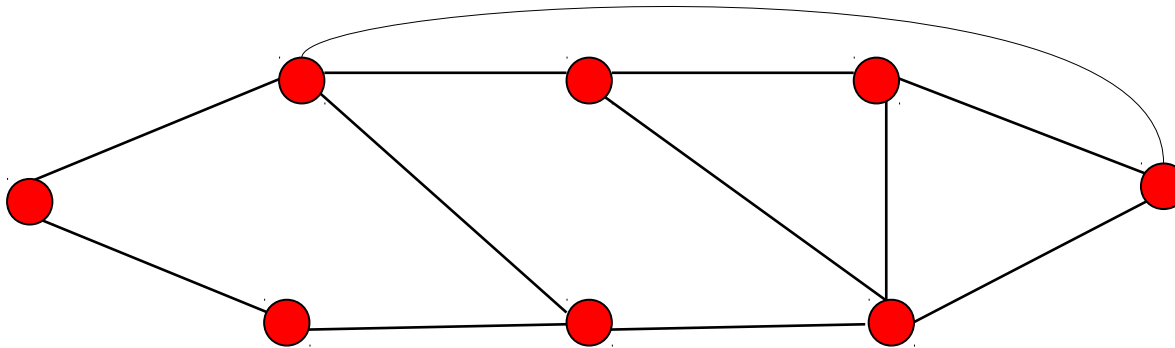


global maximum



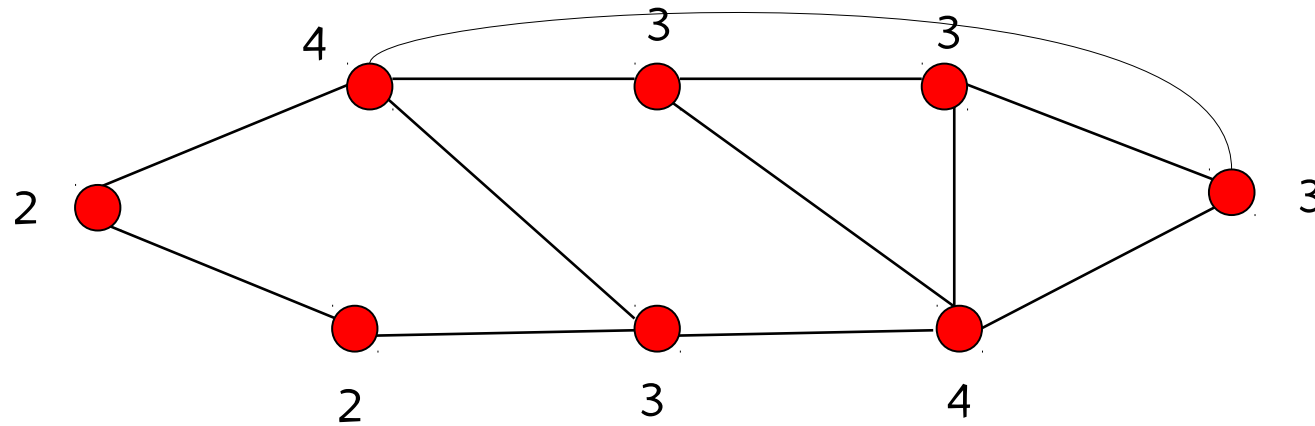
The greedy algorithm

Another example: Maximum clique



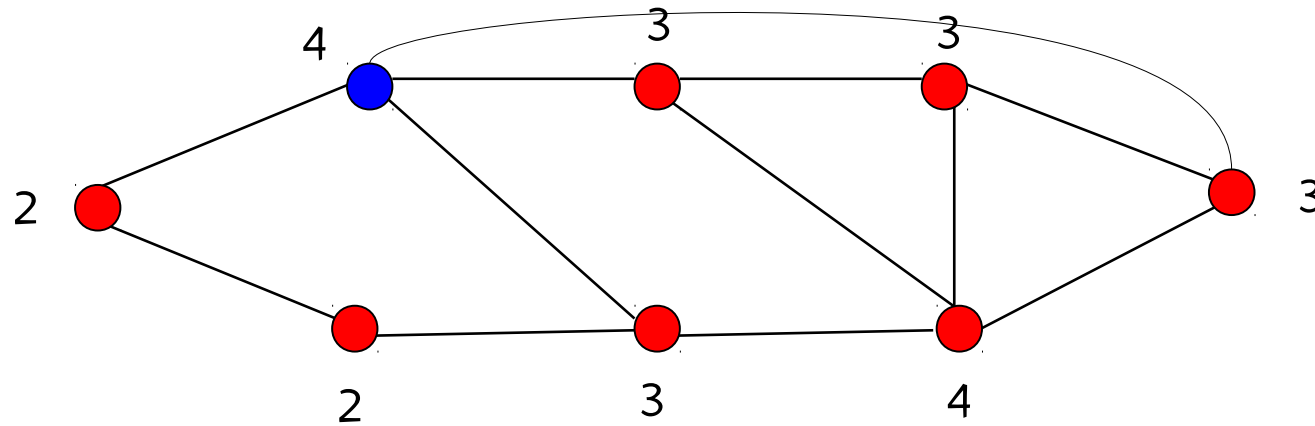
The greedy algorithm

Another example: Maximum clique



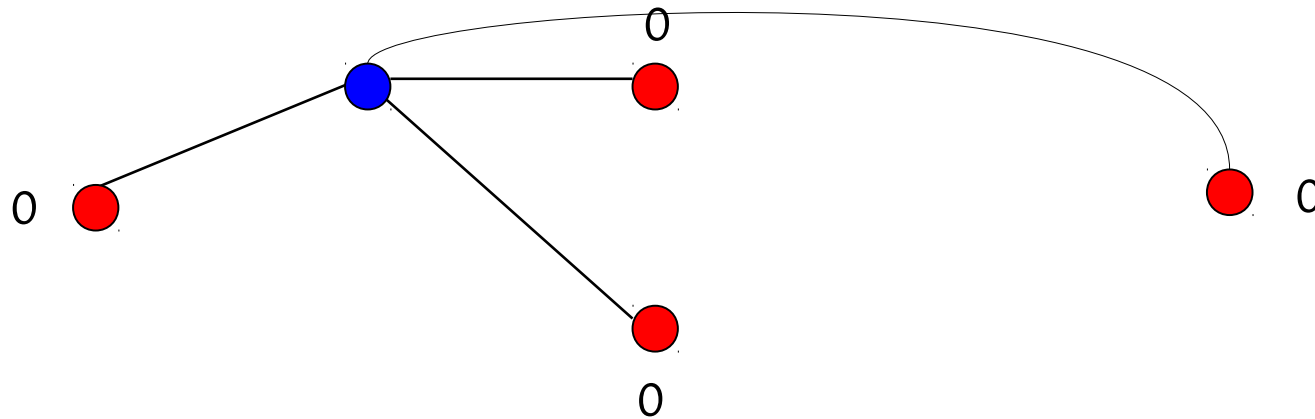
The greedy algorithm

Another example: Maximum clique



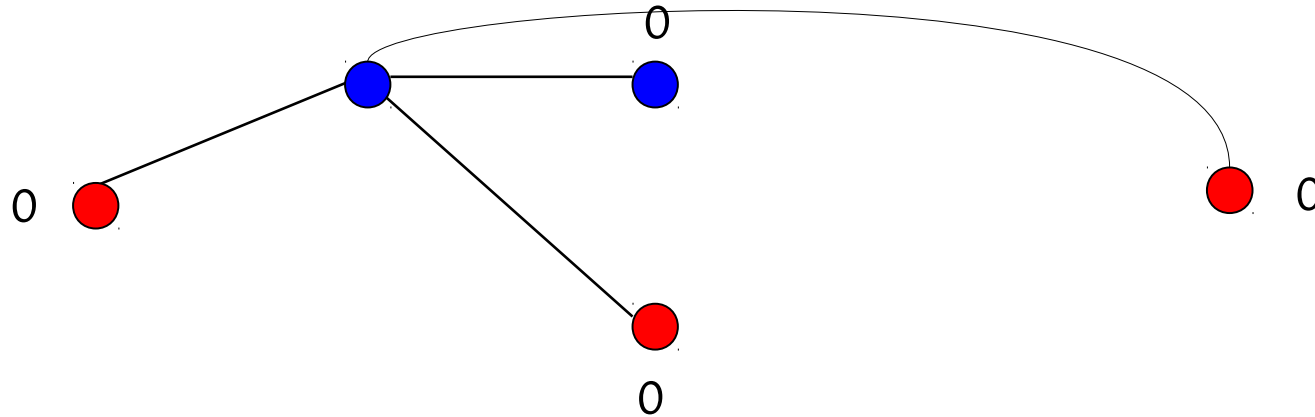
The greedy algorithm

Another example: Maximum clique



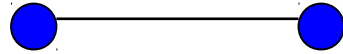
The greedy algorithm

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

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

repeat until done



Semi-greedy heuristic

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

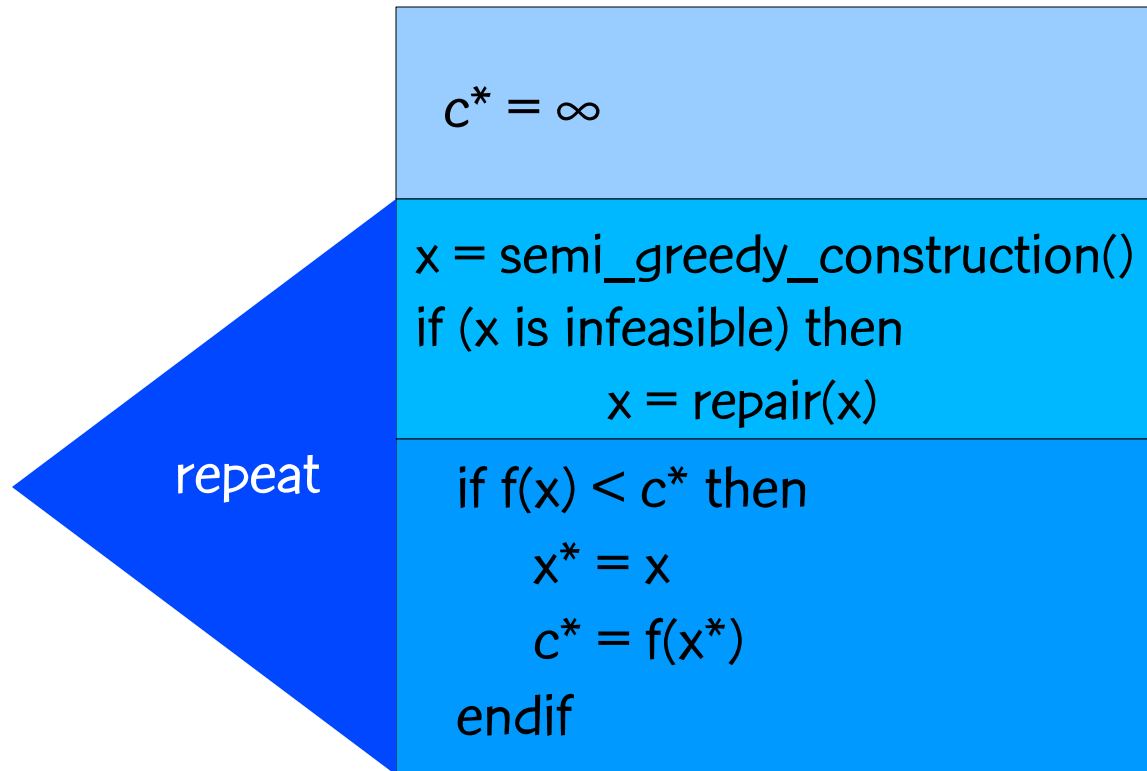
Cardinality based: place k best candidates in RCL

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

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

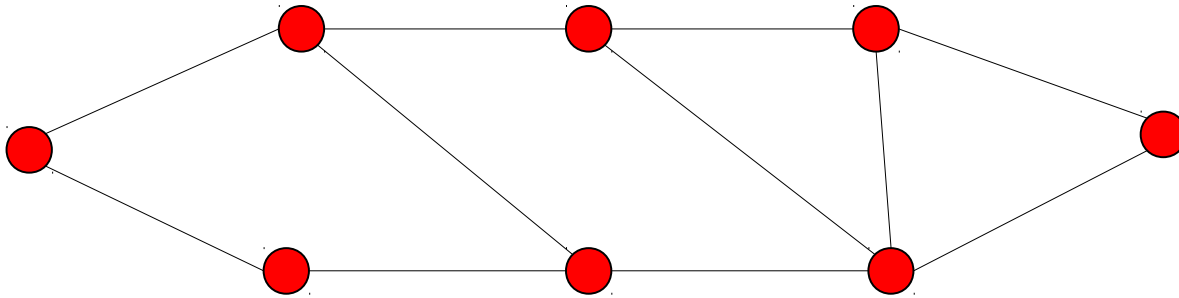


Hart-Shogan Algorithm



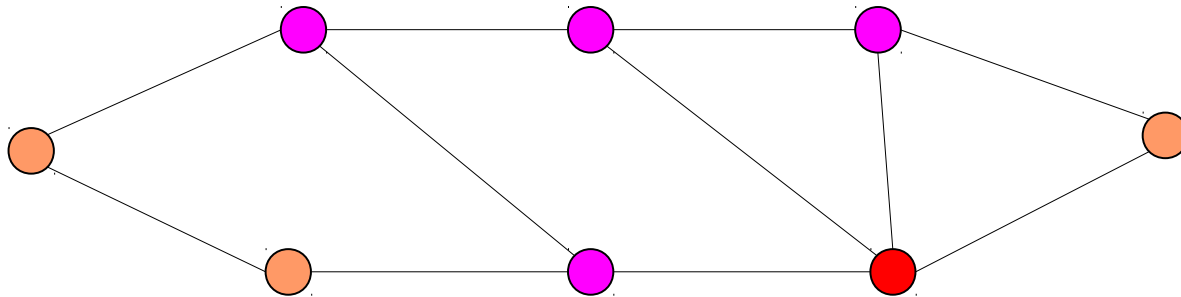
The semi-greedy algorithm

Maximum clique example



The semi-greedy algorithm

Maximum clique example

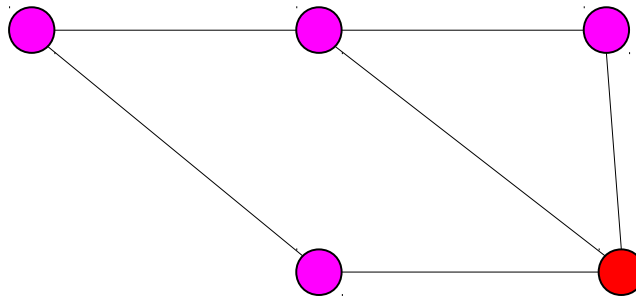


Build clique, one node at a time.

Candidates: nodes adjacent to clique.

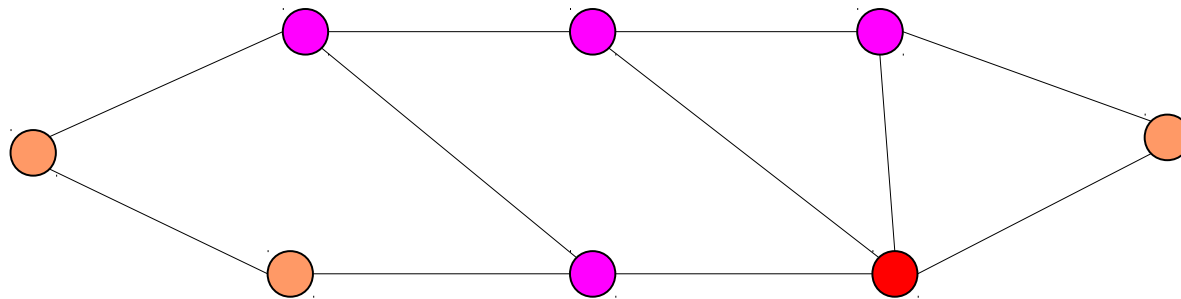
Greedy function: degree with respect to candidate nodes.

RCL =



The semi-greedy algorithm

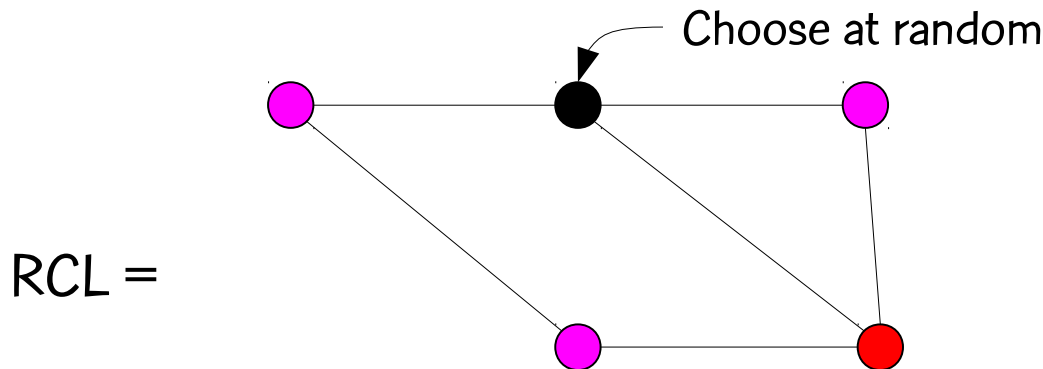
Maximum clique example



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

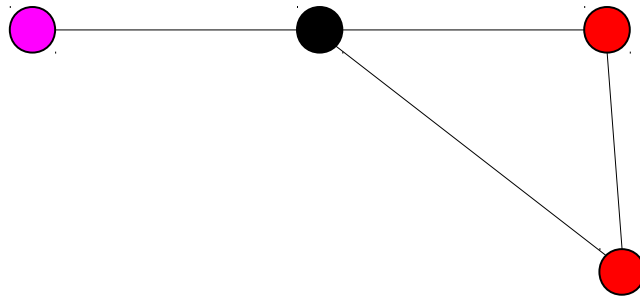


Semi-greedy
iteration 1



The semi-greedy algorithm

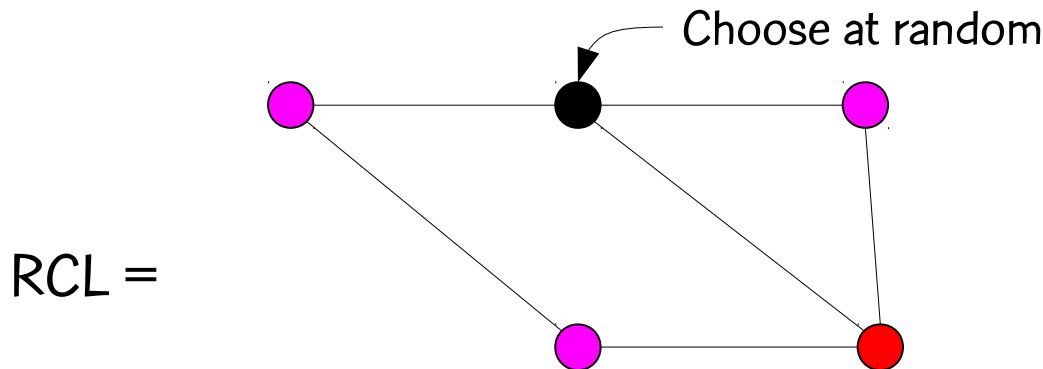
Maximum clique example



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

Greedy function: degree with respect to candidate nodes.

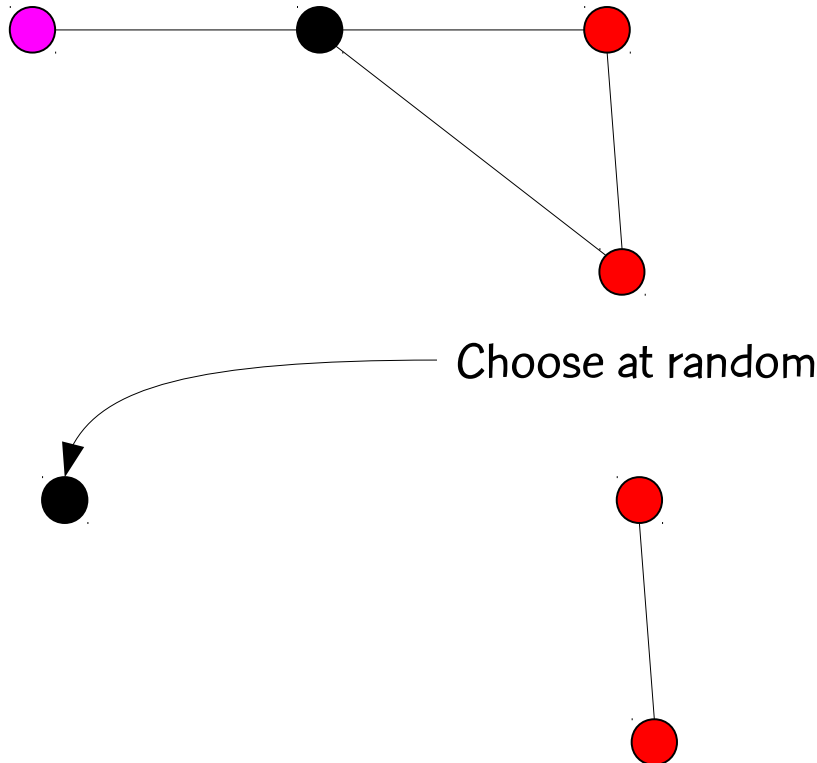


Semi-greedy
iteration 1



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.

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 =

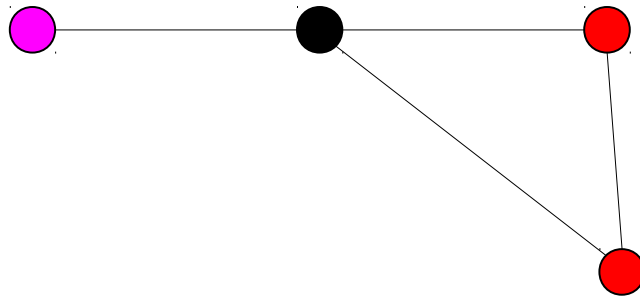


Semi-greedy
iteration 1

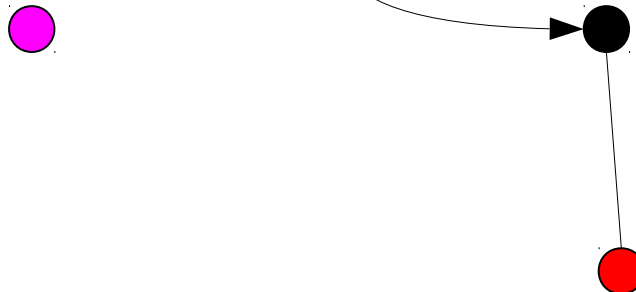


The semi-greedy algorithm

Maximum clique example



Instead, choose at random



RCL =

Build clique, one node at a time.

Candidates: nodes adjacent to clique.

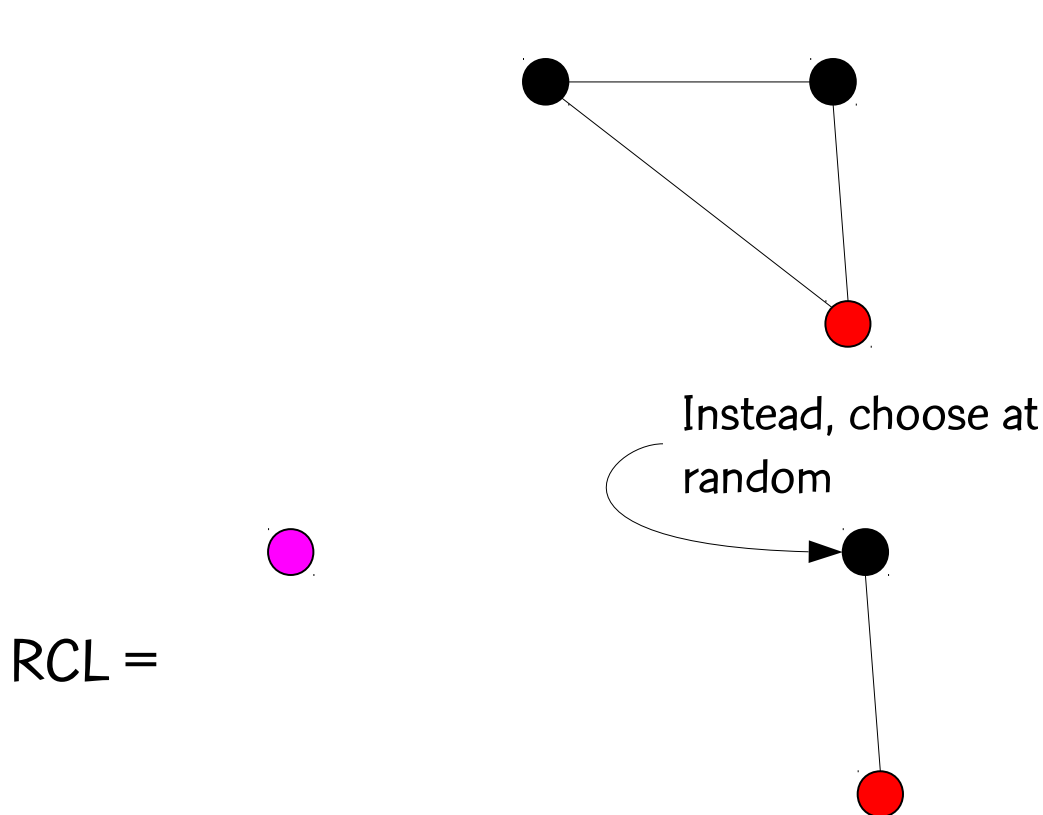
Greedy function: degree with respect to candidate nodes.

Semi-greedy
iteration 2



The semi-greedy algorithm

Maximum clique example



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

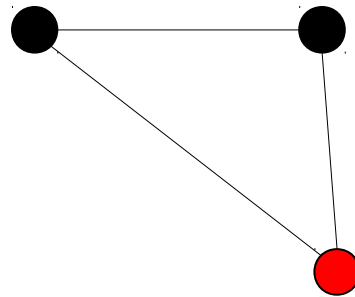
Greedy function: degree with respect to candidate nodes.

Semi-greedy
iteration 2



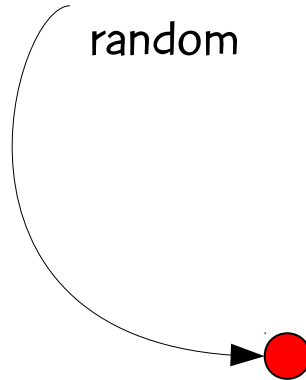
The semi-greedy algorithm

Maximum clique example



Then, choose at
random

RCL =



Build clique, one node at a
time.

Candidates: nodes adjacent
to clique.

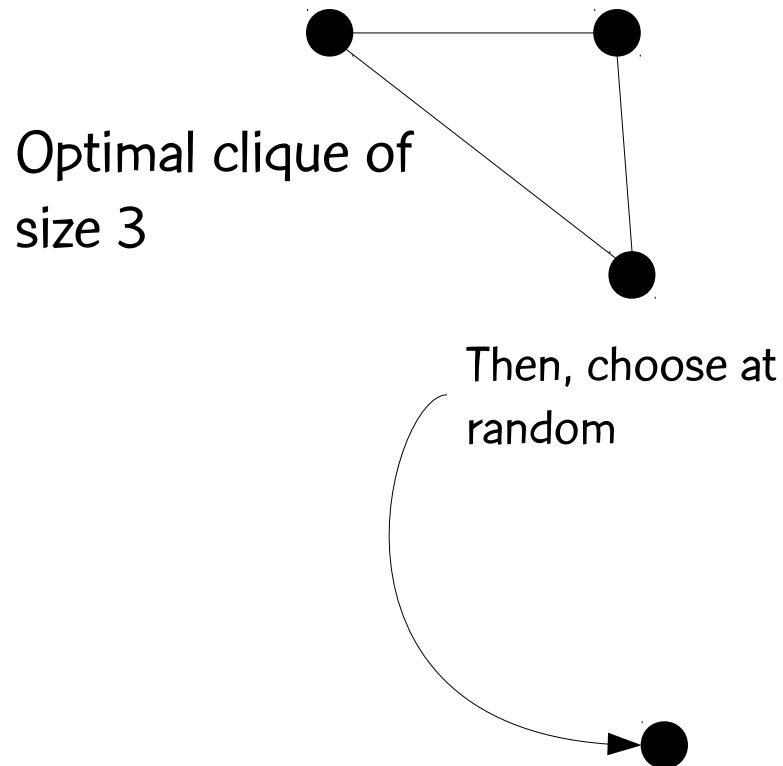
Greedy function: degree
with respect to candidate
nodes.

Semi-greedy
iteration 2



The semi-greedy algorithm

Maximum clique example



Build clique, one node at a time.

Candidates: nodes adjacent to clique.

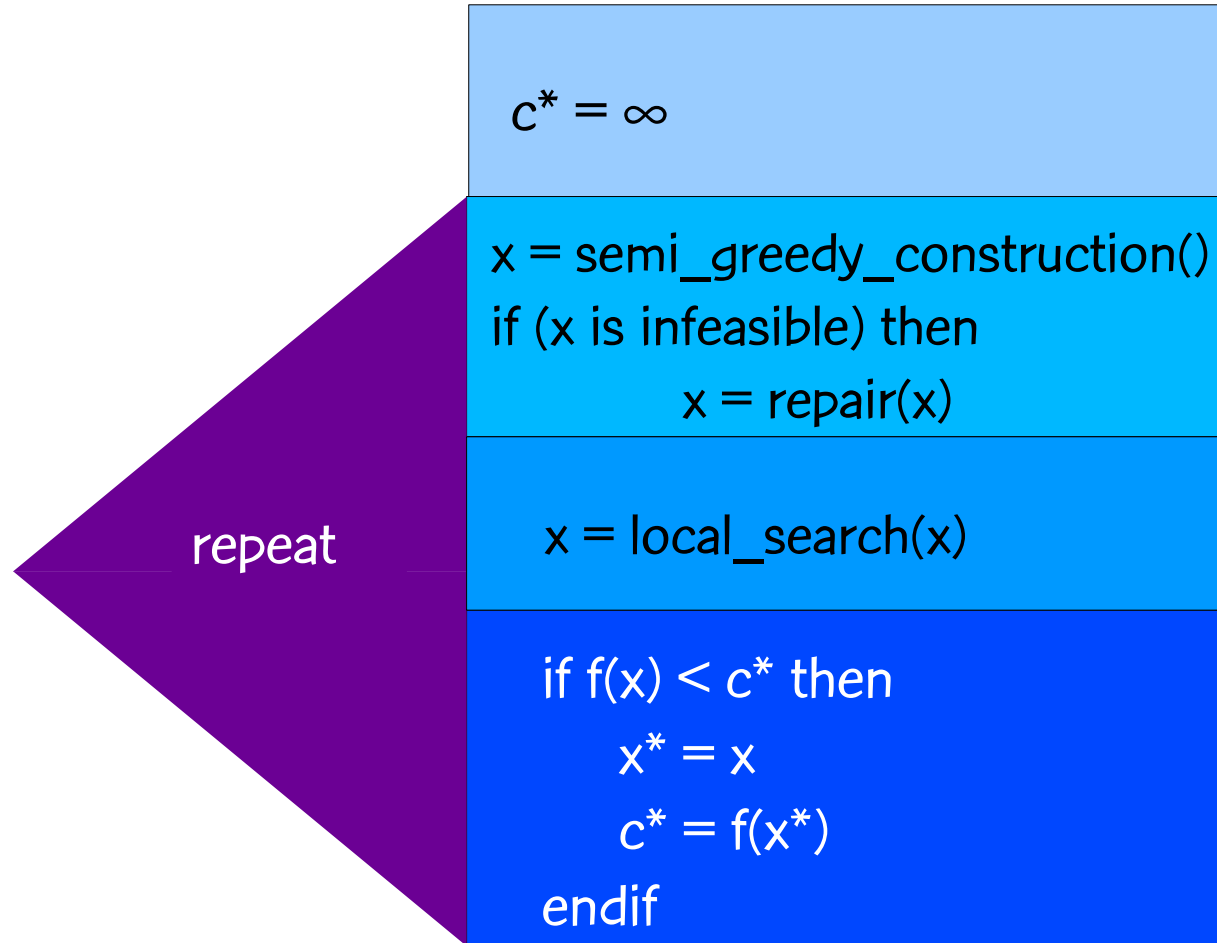
Greedy function: degree with respect to candidate nodes.



GRASP



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.

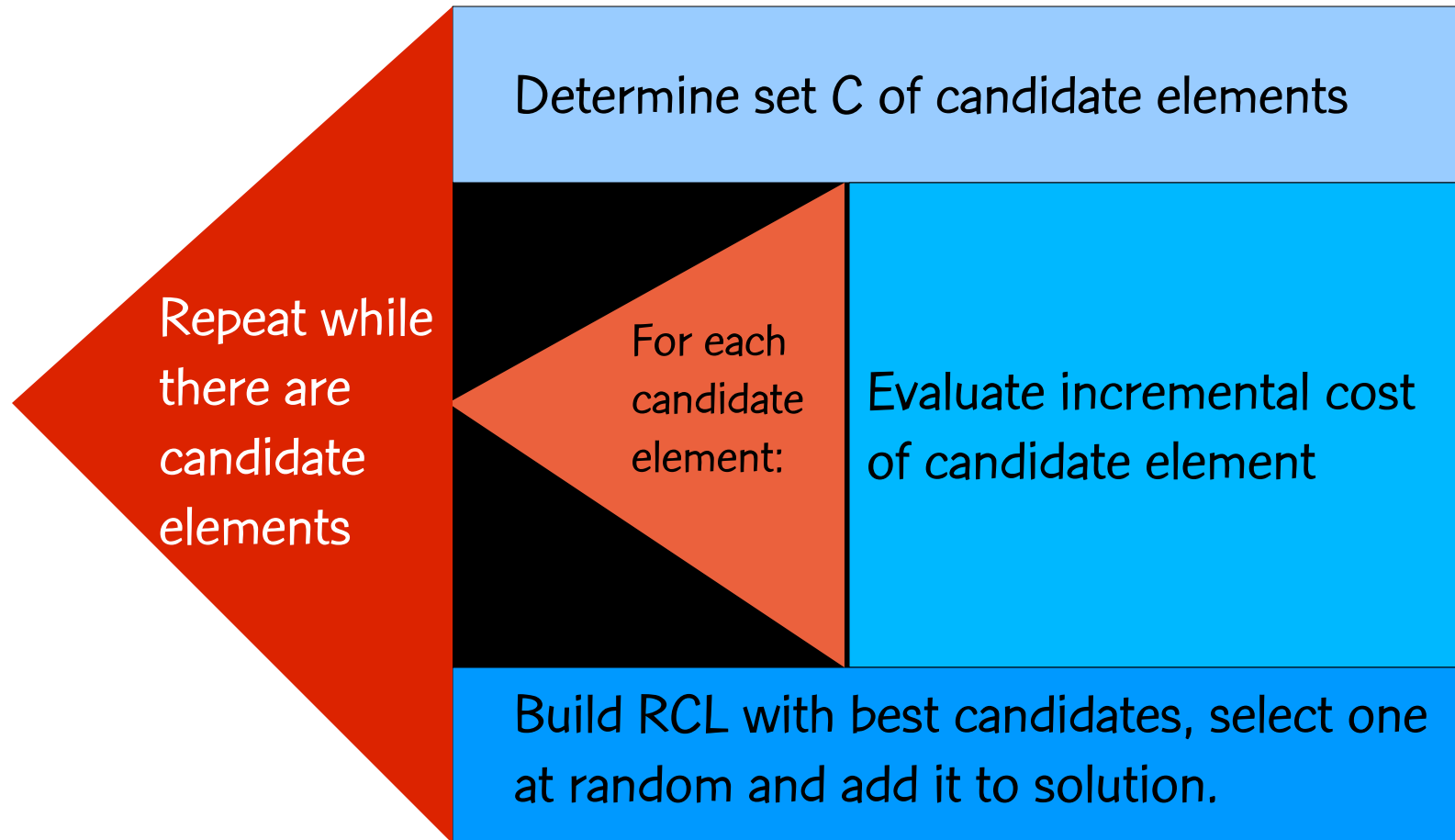


GRASP Construction



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

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

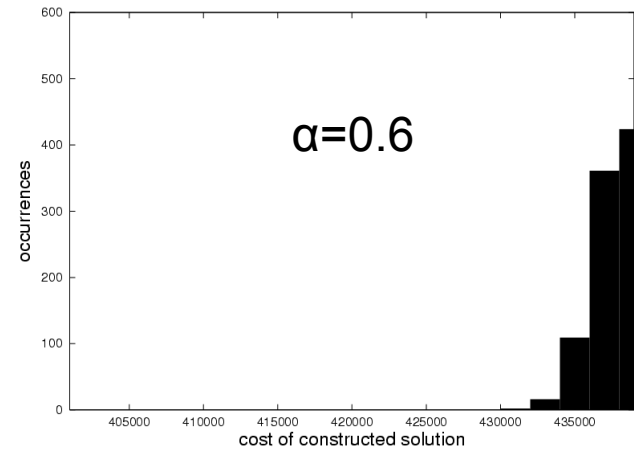
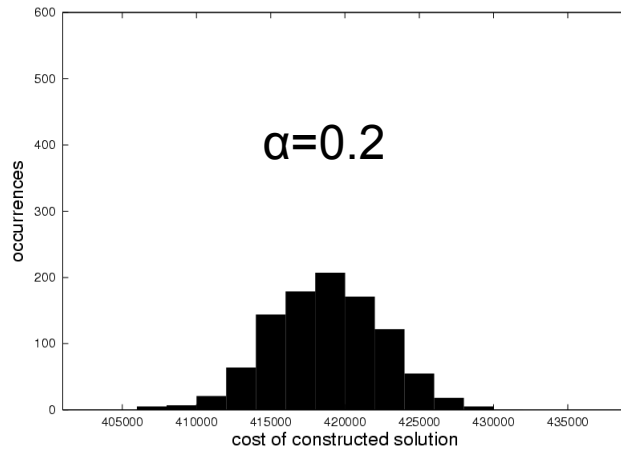
$\alpha = 0$: pure greedy construction

$\alpha = 1$: pure randomized construction

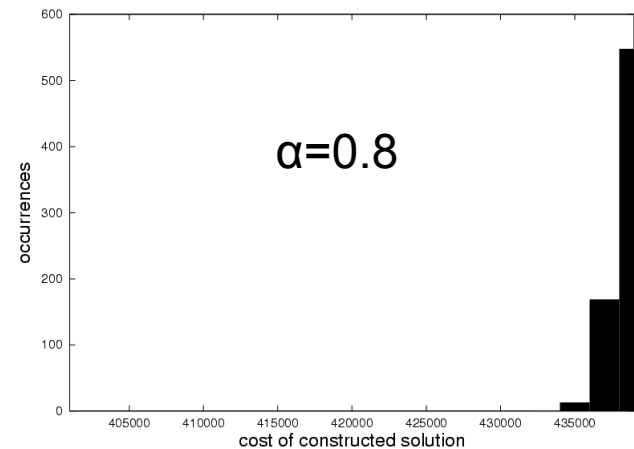
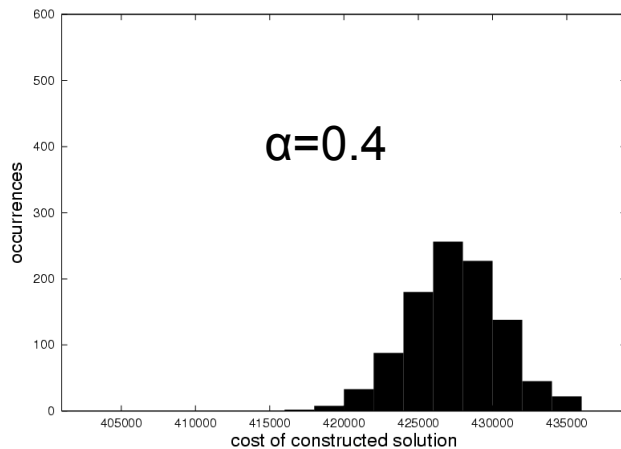
Select at random from RCL using uniform probability distribution



Illustrative results: RCL parameter



Construction phase only

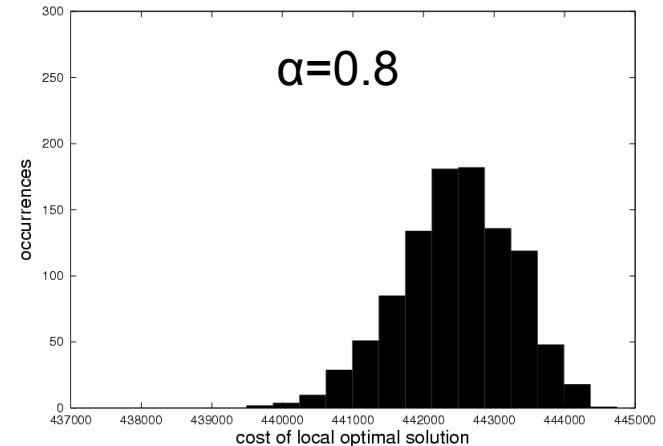
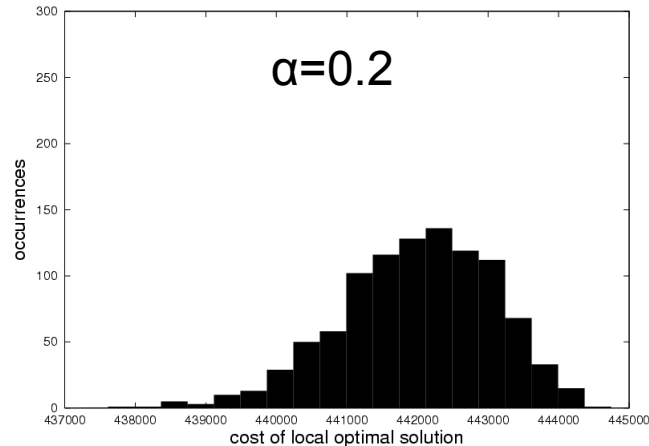


weighted MAX-SAT instance, 1000 GRASP iterations

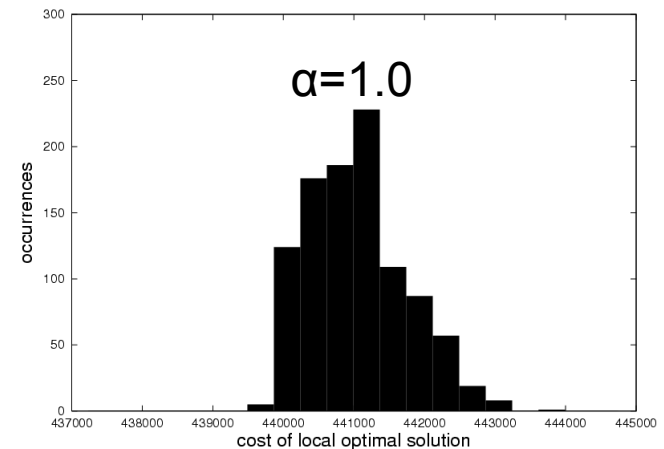
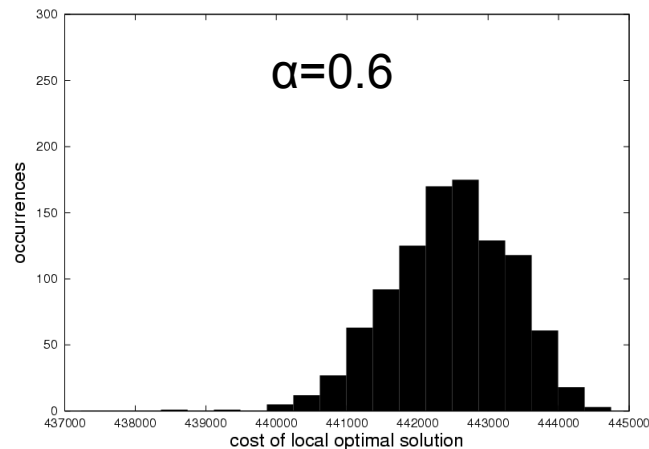
Possible



Illustrative results: RCL parameter



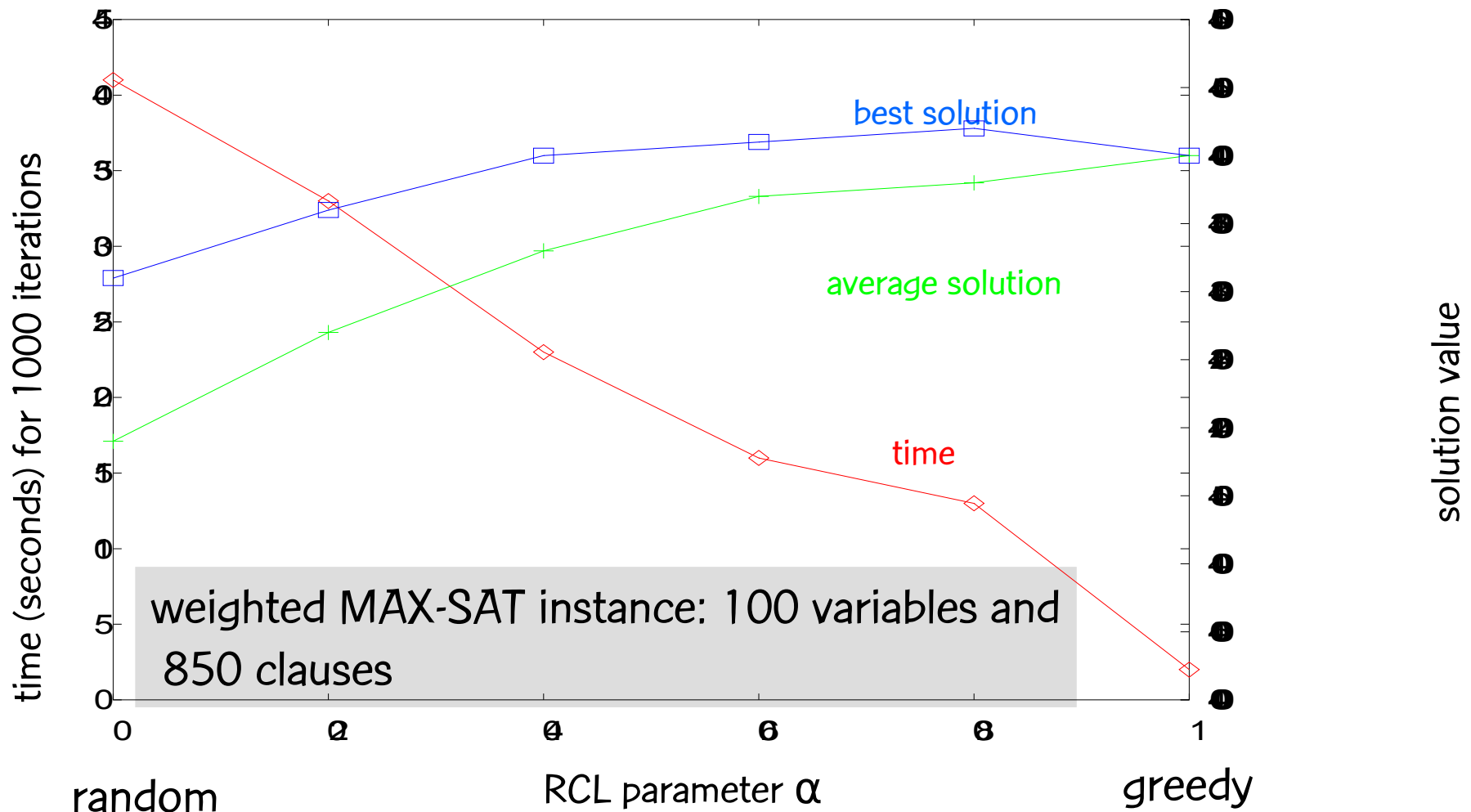
Construction + local search



weighted MAX-SAT instance, 1000 GRASP iterations



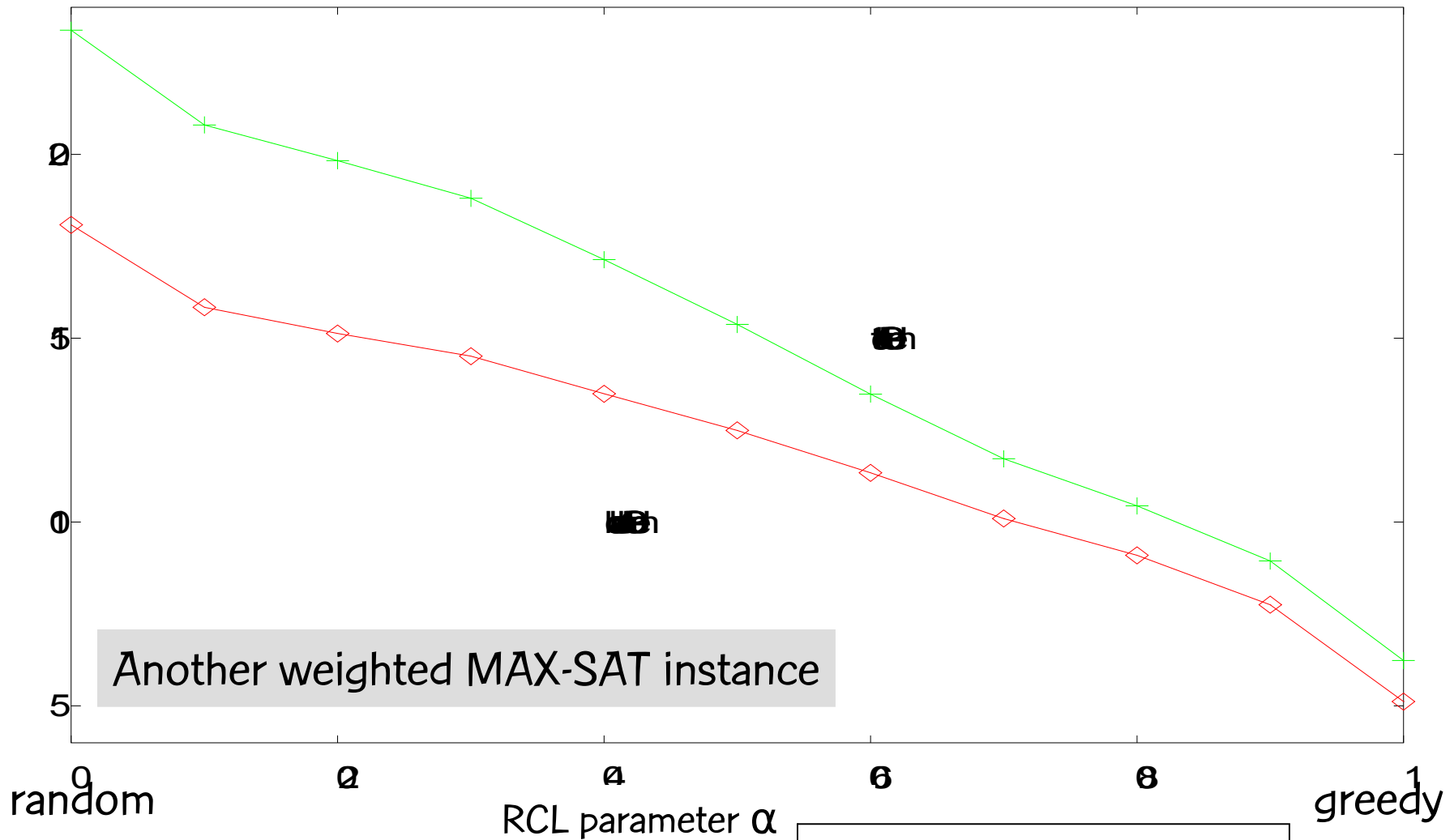
Illustrative results: RCL parameter



SGI Challenge 196 MHz



Illustrative results: RCL parameter

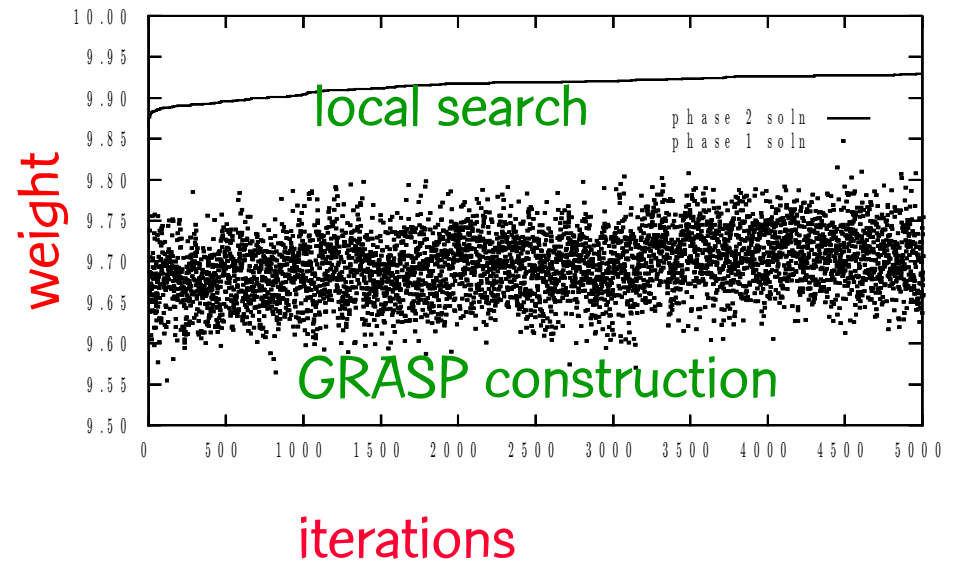
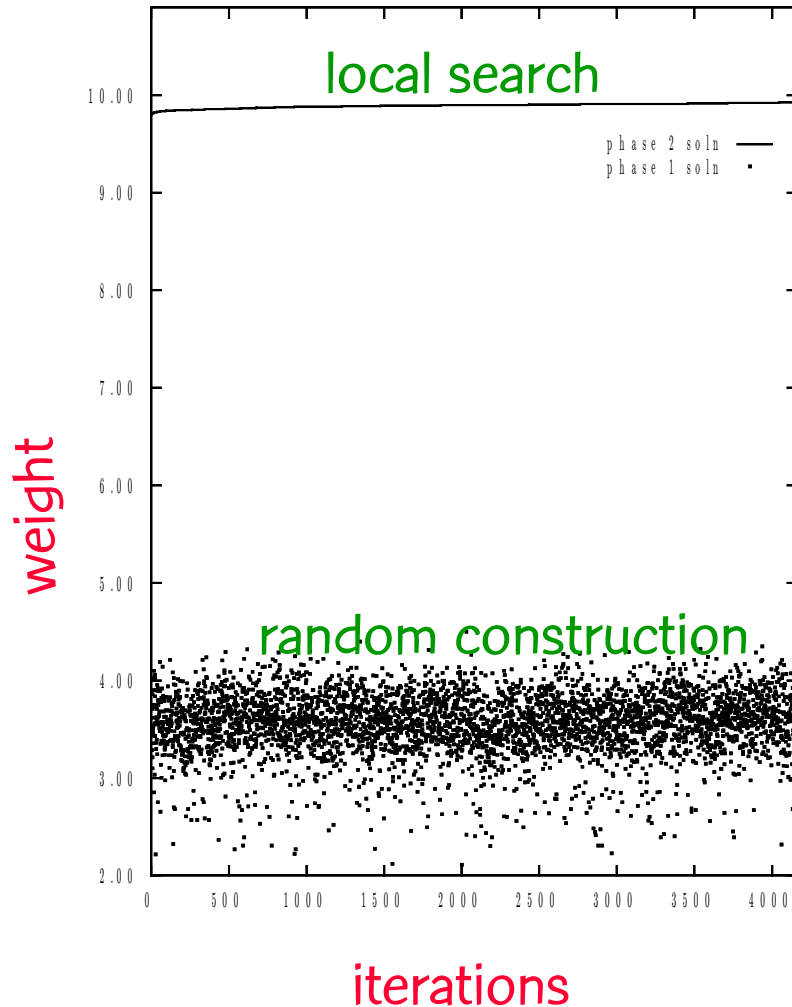


SGI Challenge 196 MHz

Rethink Possible



GRASP: Basic algorithm



Effectiveness of **greedy randomized** vs **purely randomized** construction:

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

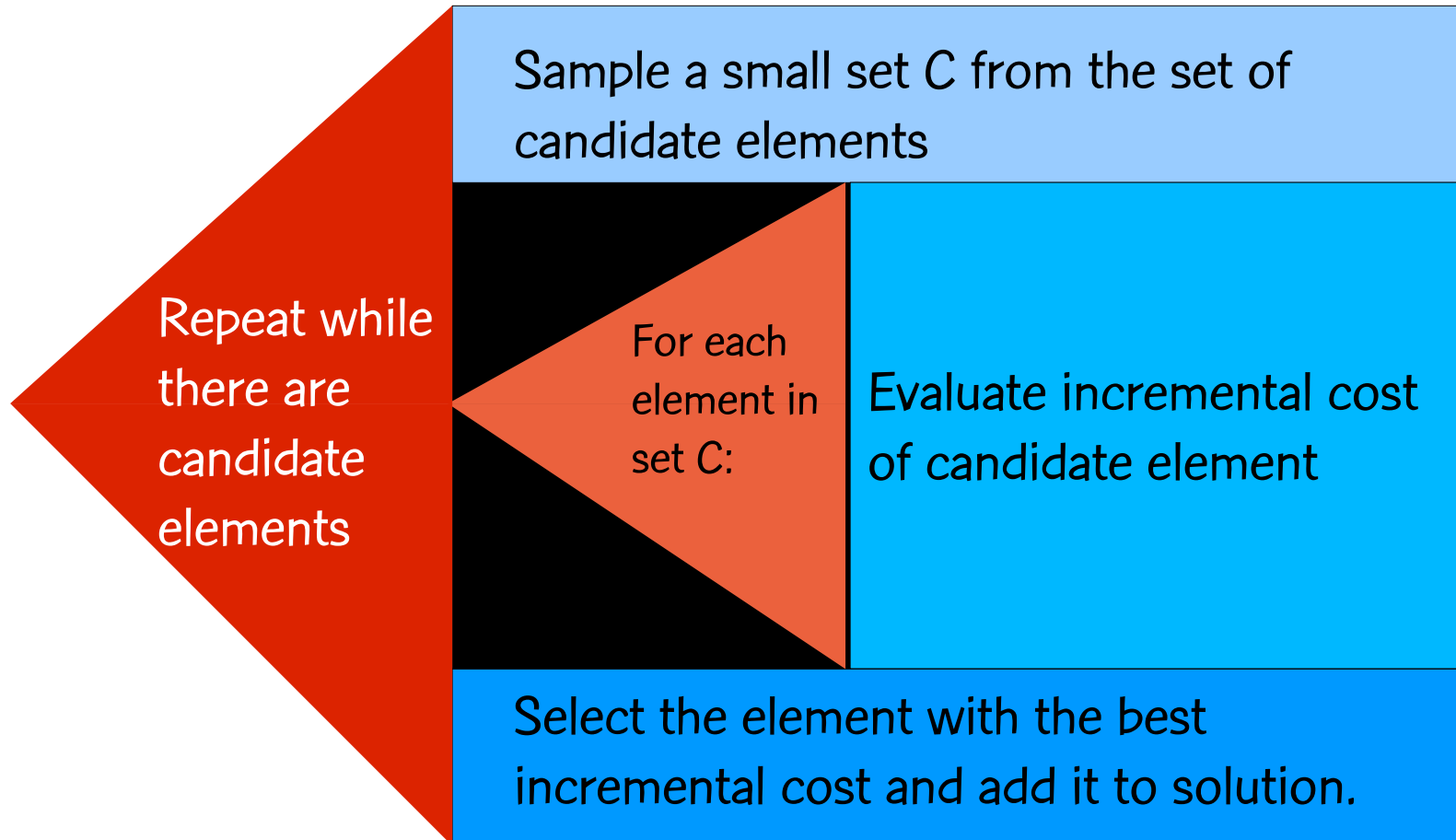


Hybrid construction schemes



Construction phase: sampled greedy

[Resende & Werneck, 2004]

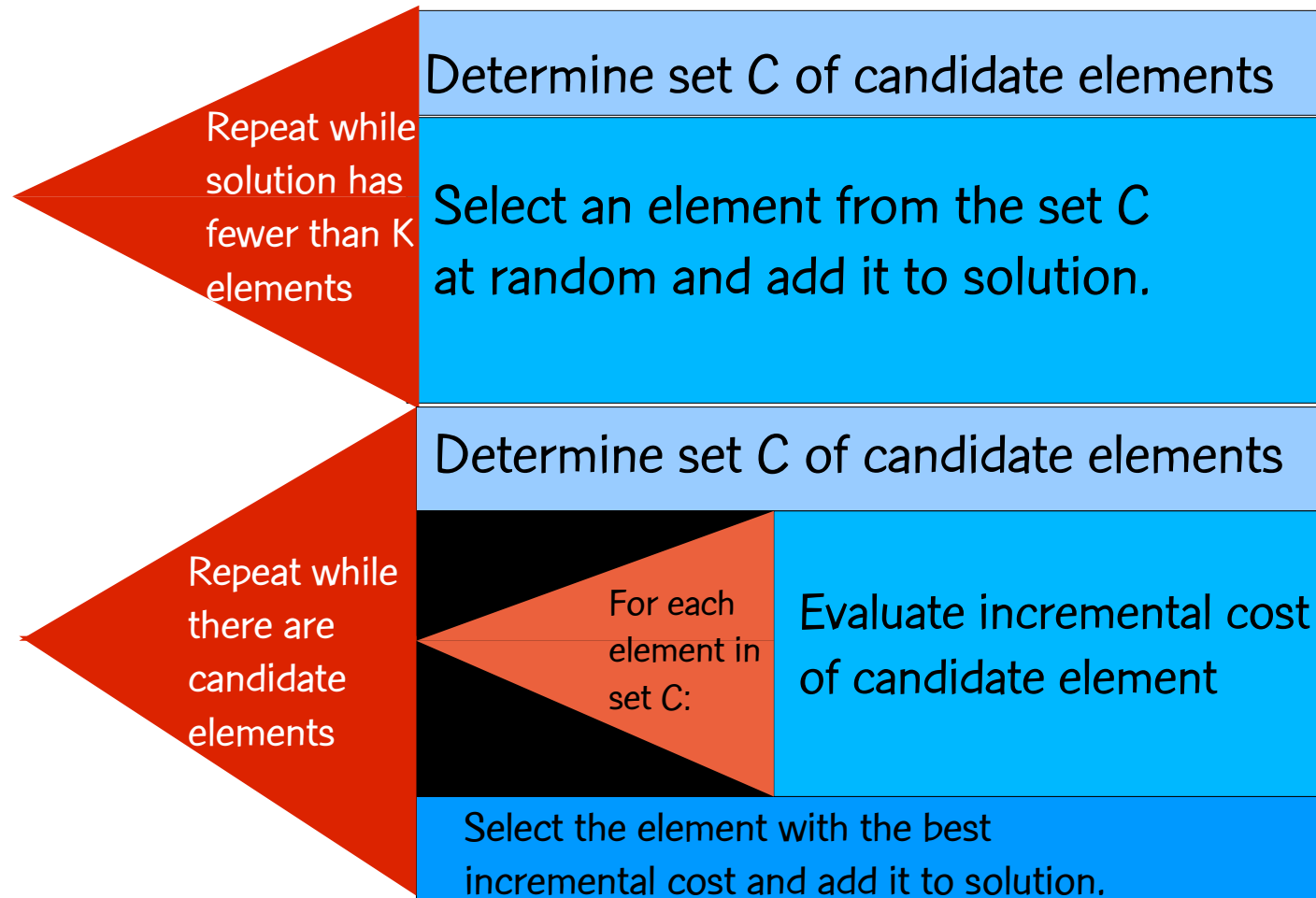


Rethink Possible



Construction phase: random+greedy

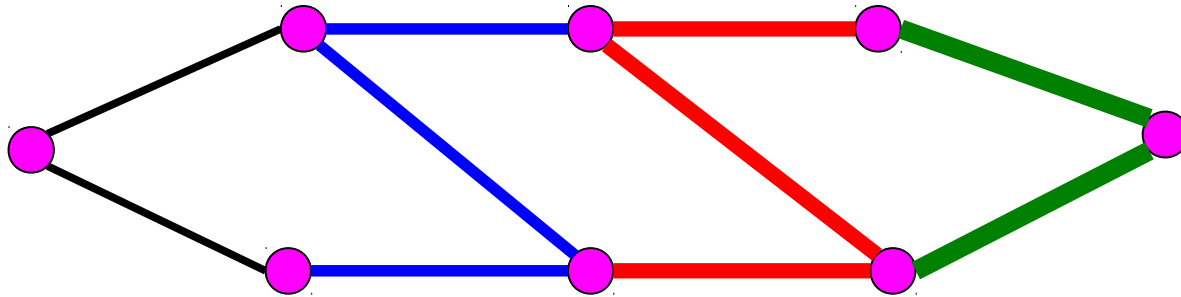
[Resende & Werneck, 2004]



Rethink Possible



Construction with cost perturbation

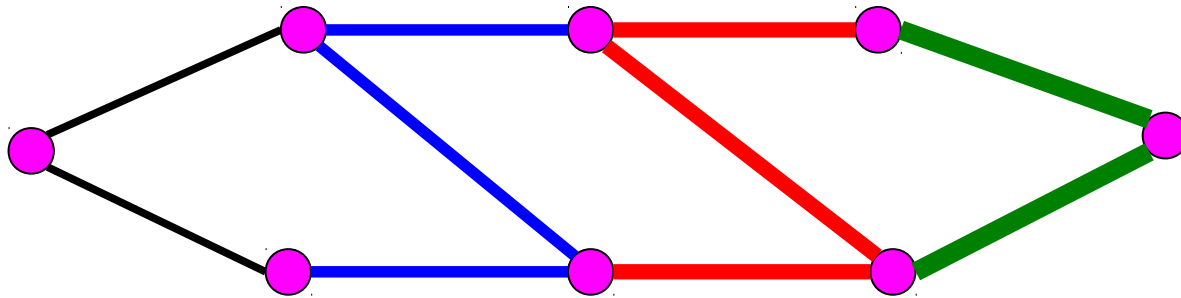


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

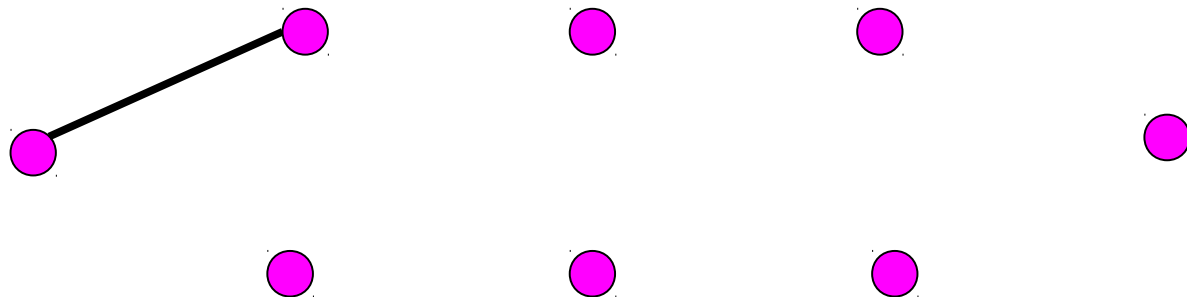


Construction with cost perturbation

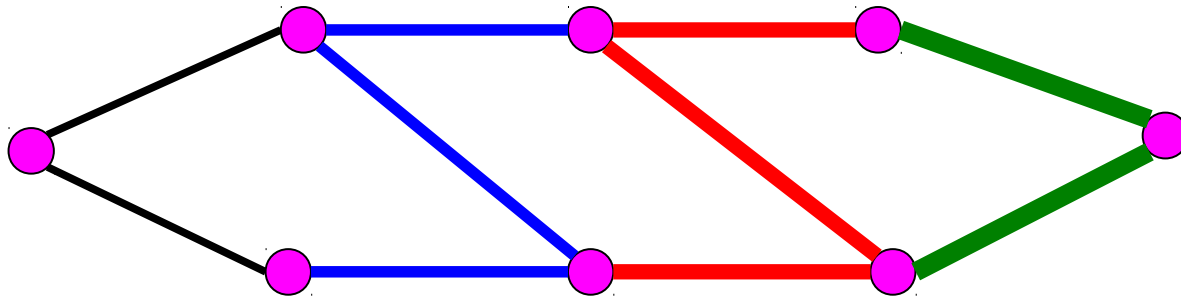


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

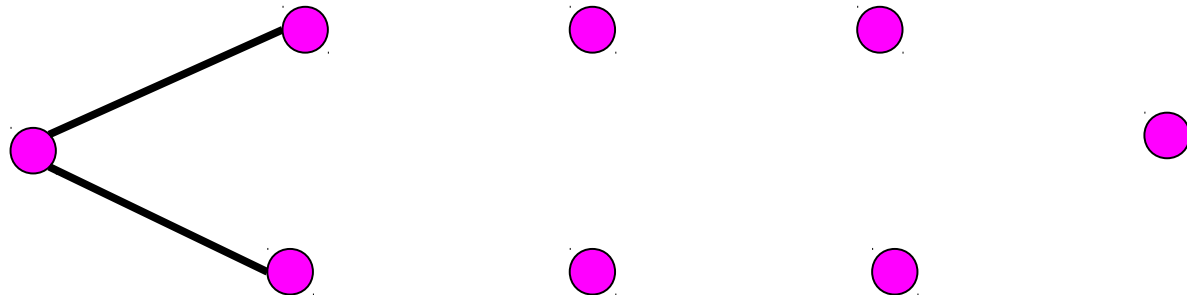


Construction with cost perturbation

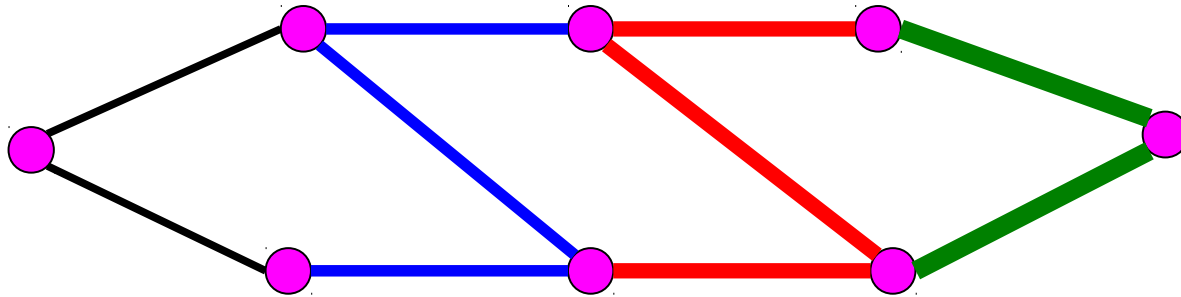


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

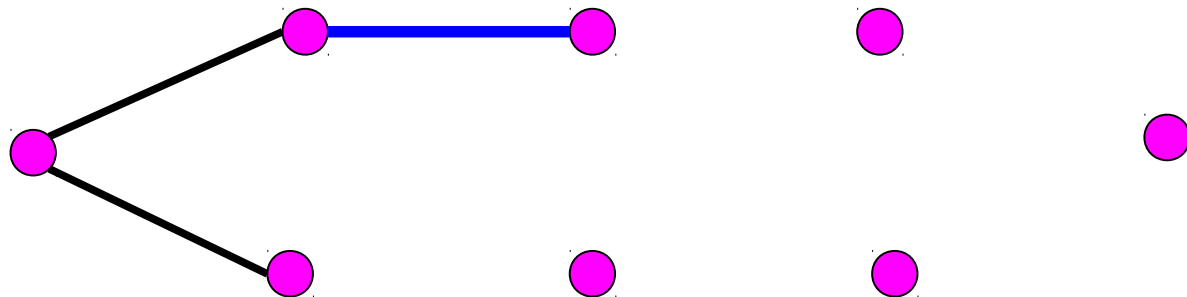


Construction with cost perturbation

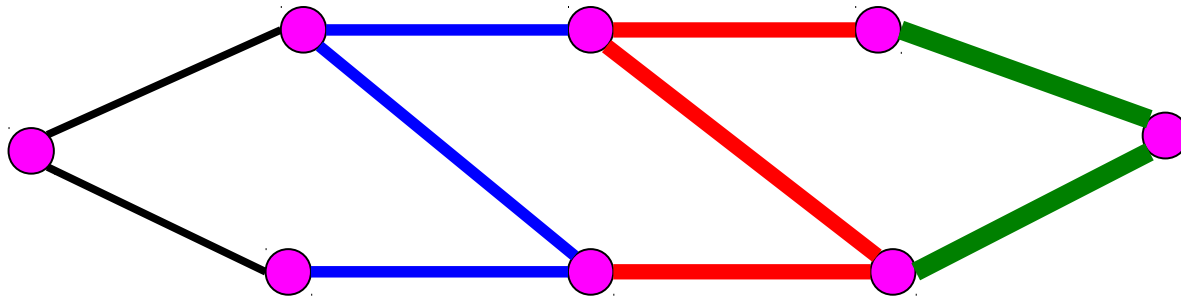


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

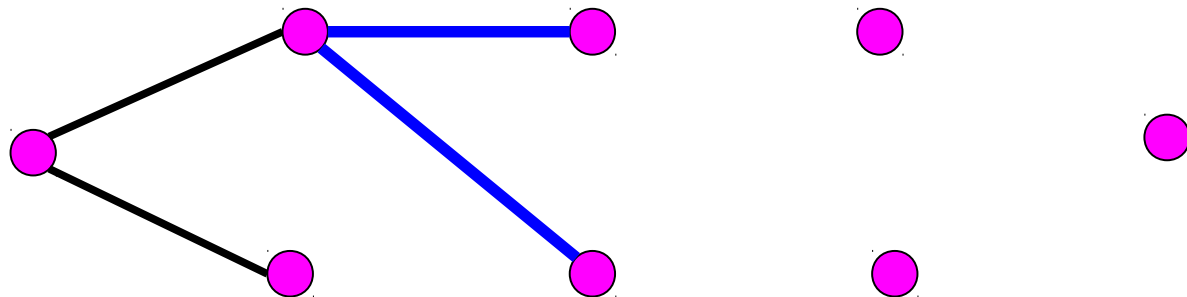


Construction with cost perturbation

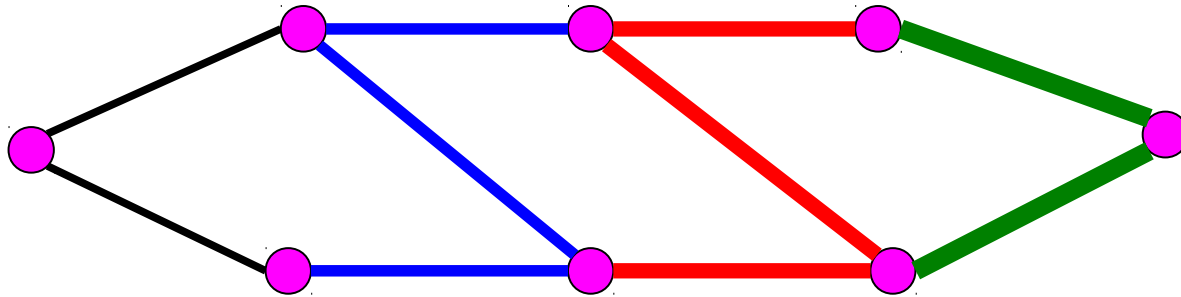


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

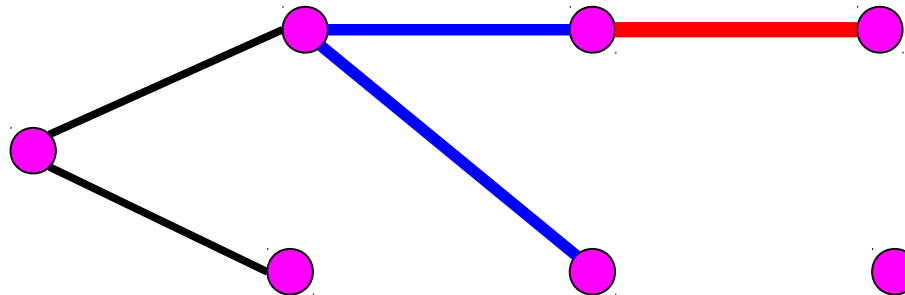


Construction with cost perturbation

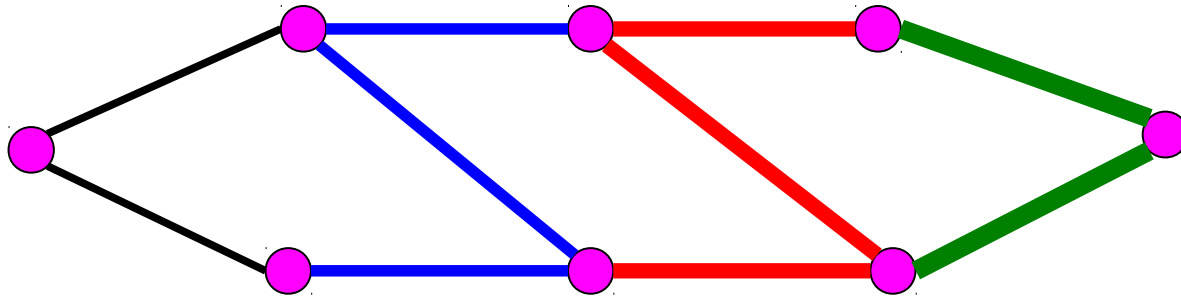


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

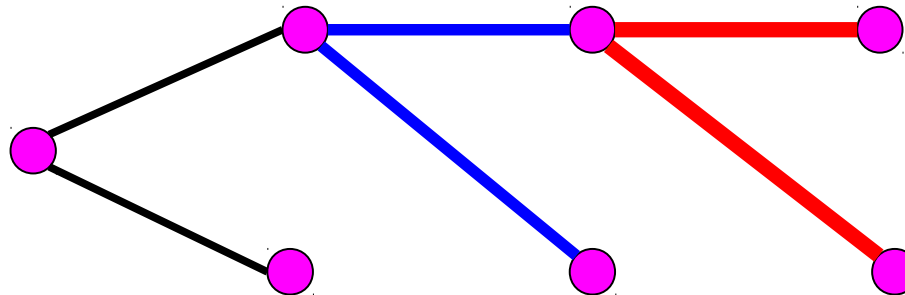


Construction with cost perturbation

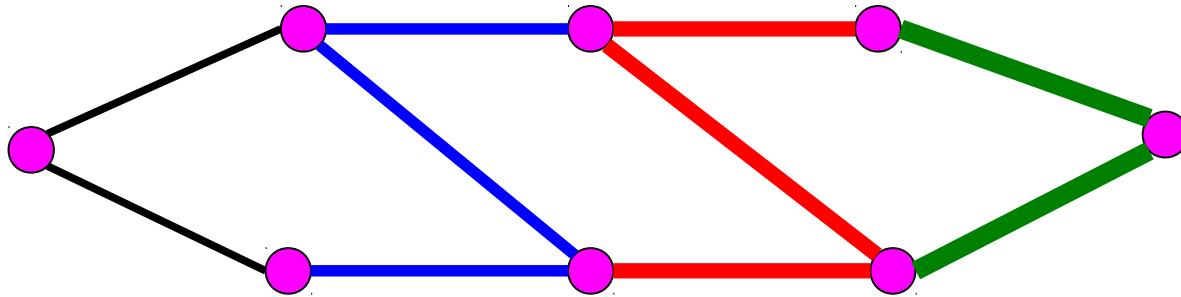


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

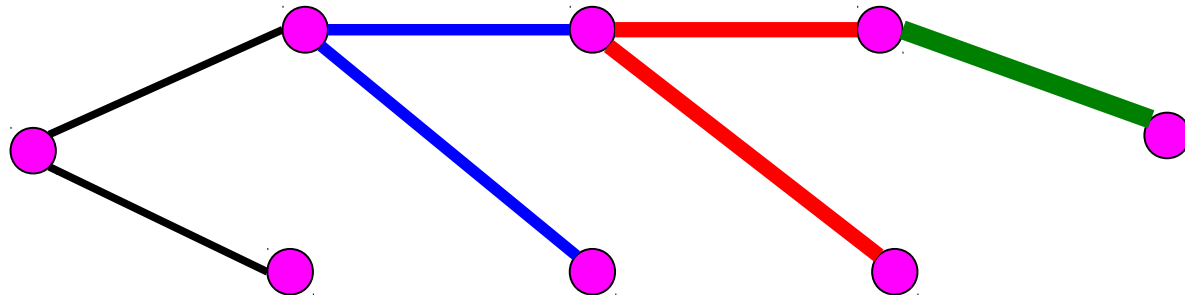


Construction with cost perturbation

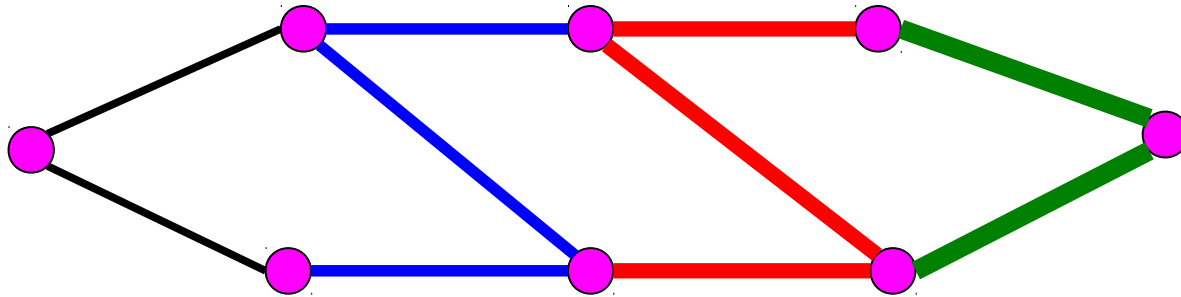


Perturb with costs increasing from top to bottom.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$



Construction with cost perturbation

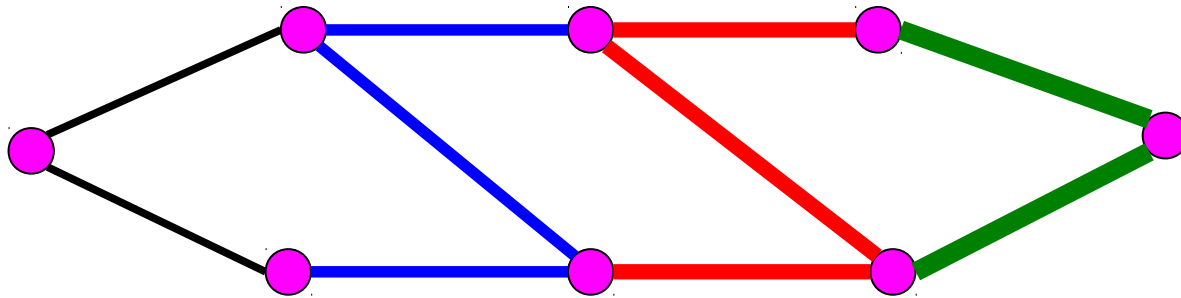


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

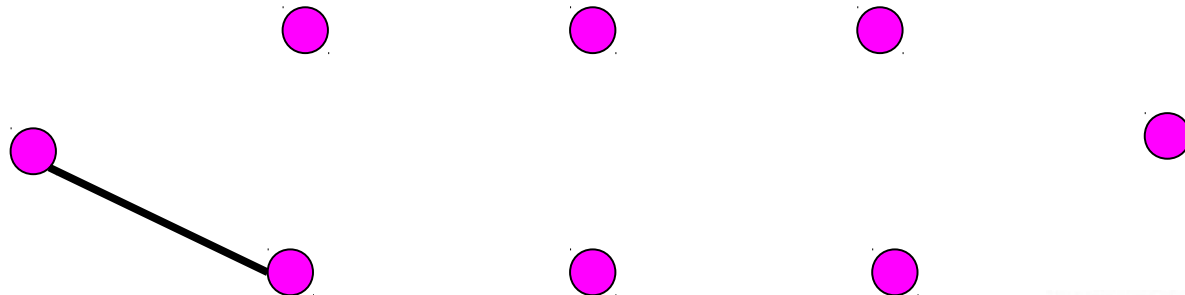


Construction with cost perturbation

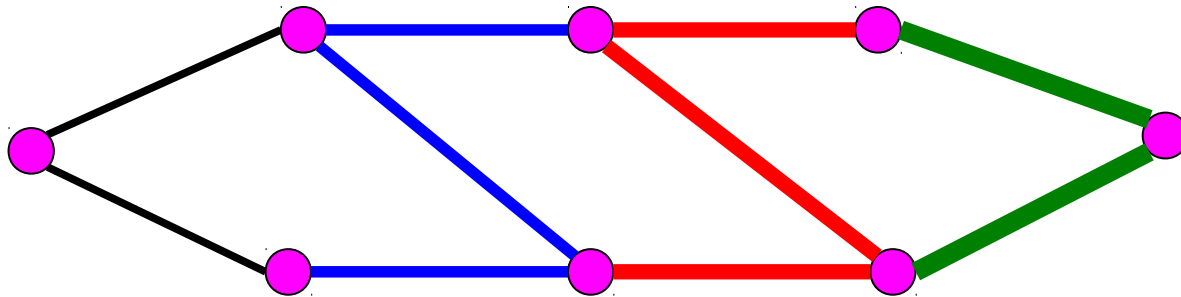


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

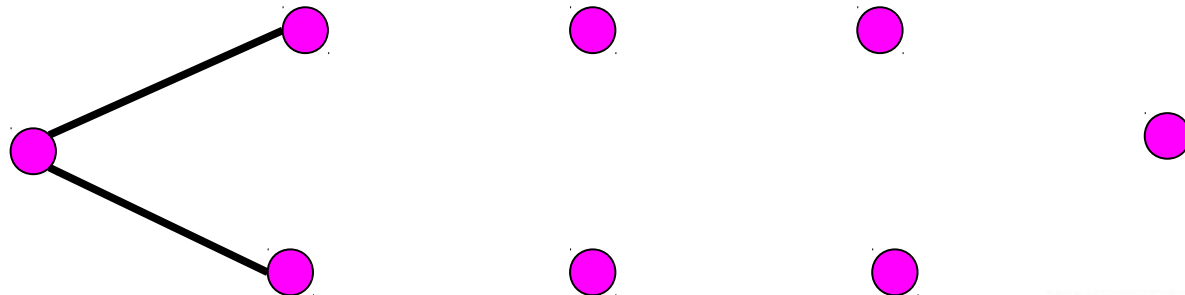


Construction with cost perturbation

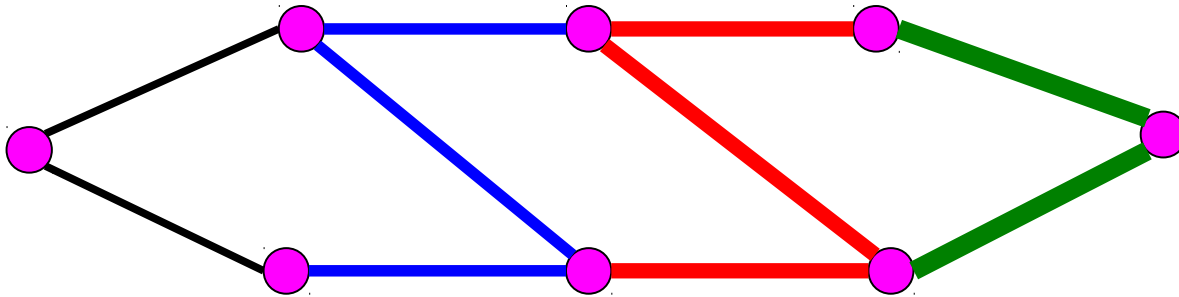


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

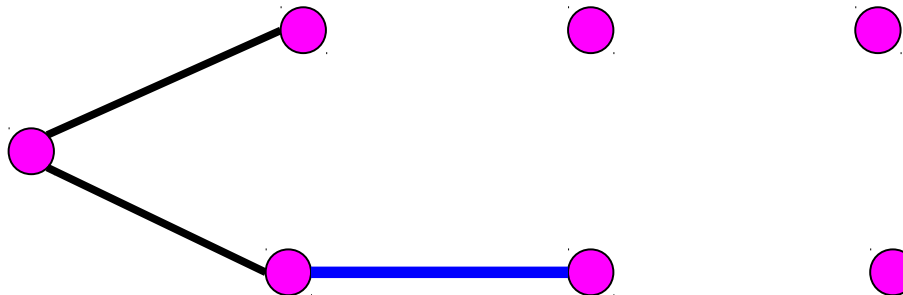


Construction with cost perturbation

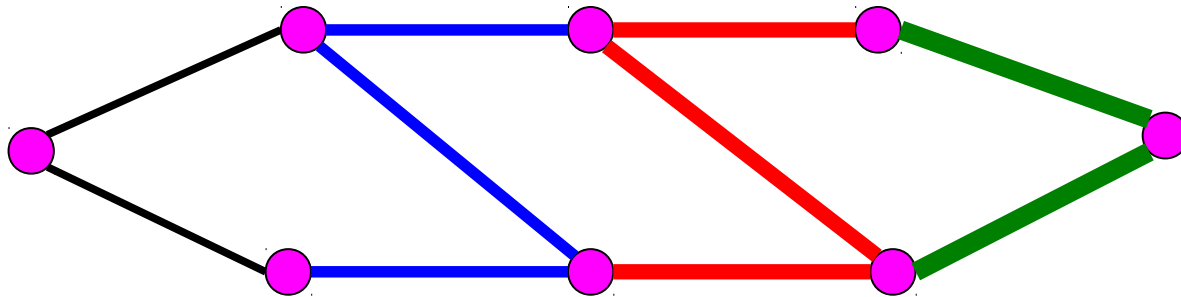


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

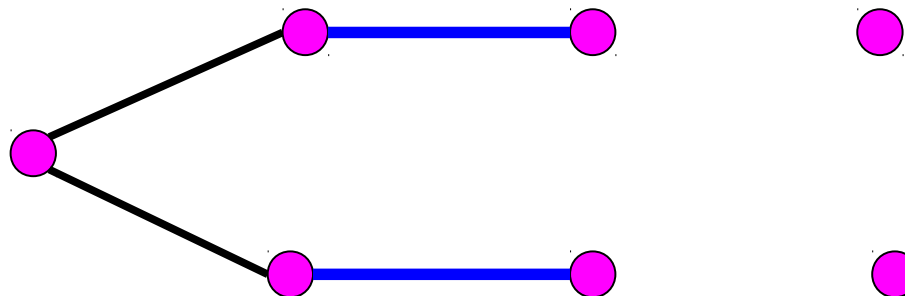


Construction with cost perturbation

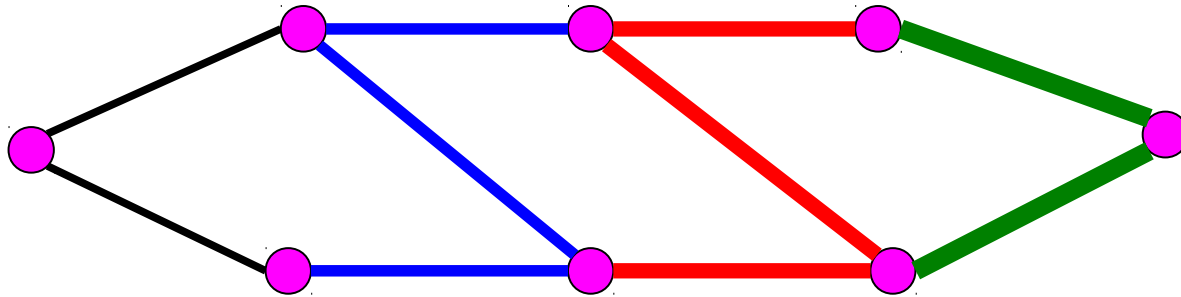


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

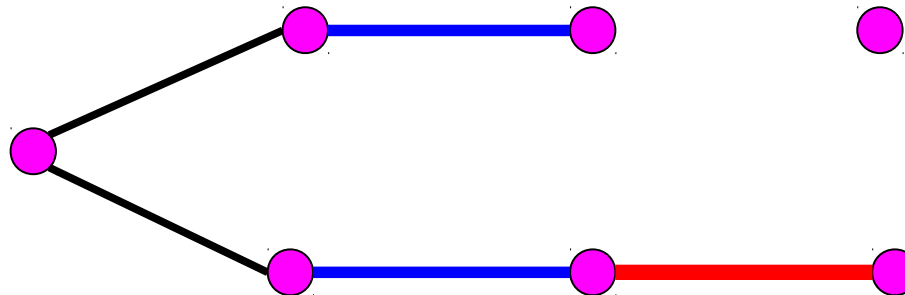


Construction with cost perturbation

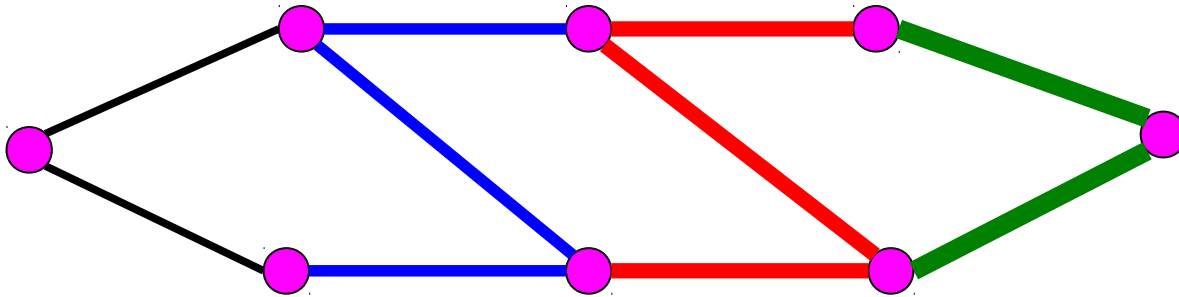


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

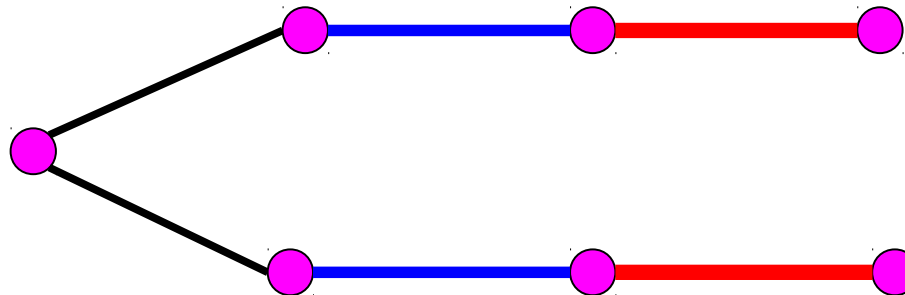


Construction with cost perturbation

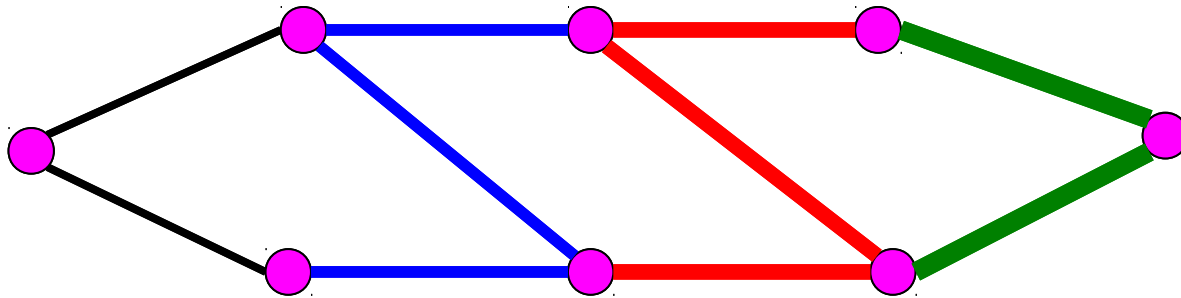


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

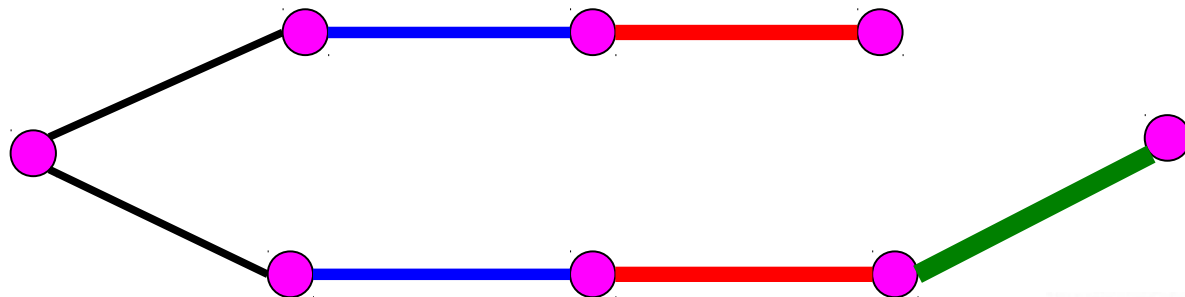


Construction with cost perturbation

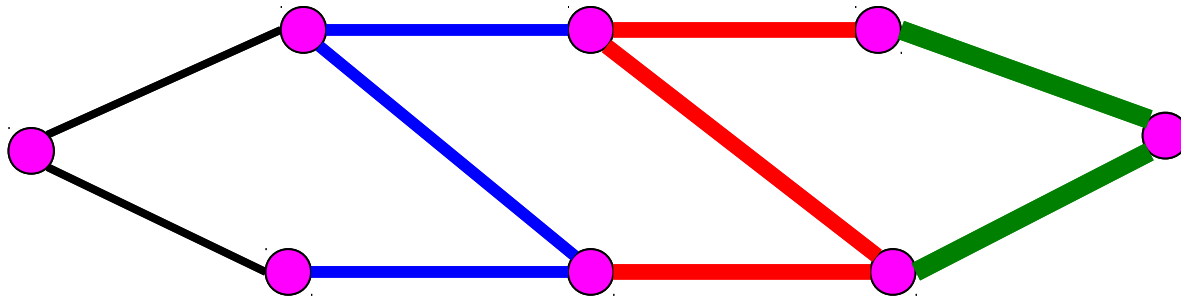


Perturb with costs increasing from bottom to top.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$

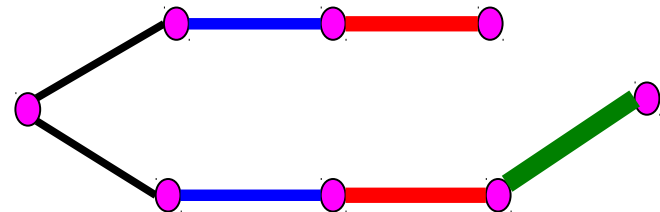
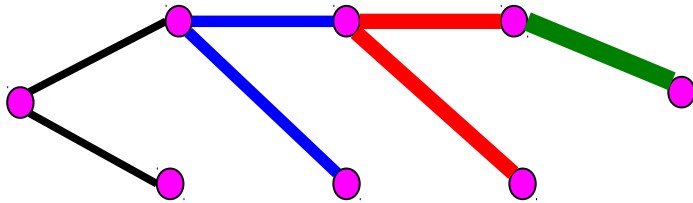


Construction with cost perturbation



Greedy heuristic generates two different spanning trees.

$$W(\text{I}) < W(\text{II}) < W(\text{III}) < W(\text{IV})$$



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 $[\alpha_1, \alpha_2, \dots, \alpha_m]$.

The probability that α_k is selected is p_k .

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



Reactive GRASP

Prais & Ribeiro (2000)

Reactive GRASP for minimization ...

Initially $p_k = 1/m$, for $k = 1, \dots, m$. (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 $k = 1, \dots, m$

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



Reactive GRASP

Prais & Ribeiro (2000)

Reactive GRASP for minimization ...

Initially $p_k = 1/m$, for $k = 1, \dots, m$. (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

$$q_k = F(S^*) / A_k, \text{ for } k = 1, \dots, m$$

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

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

Path-relinking (PR)



Path-relinking

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

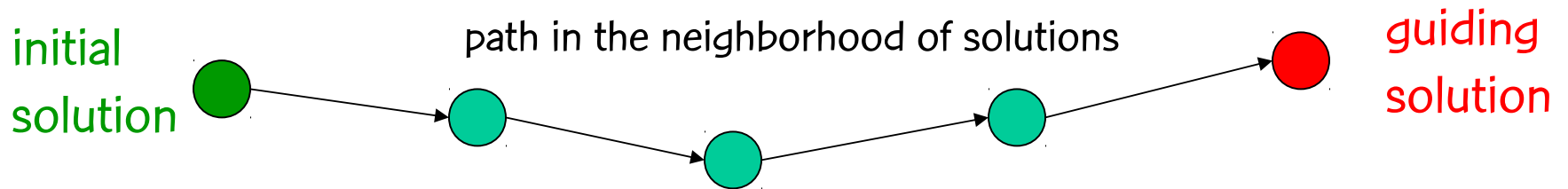
Originally proposed in the context of tabu search and scatter search.

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



Path-relinking

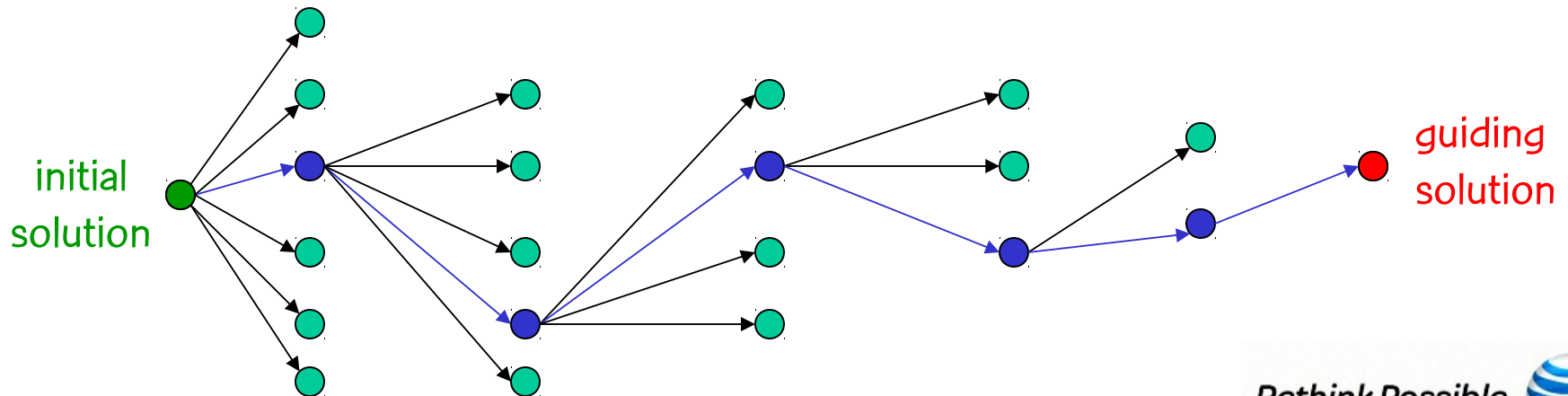
Exploration of trajectories that connect high quality (elite) solutions:



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

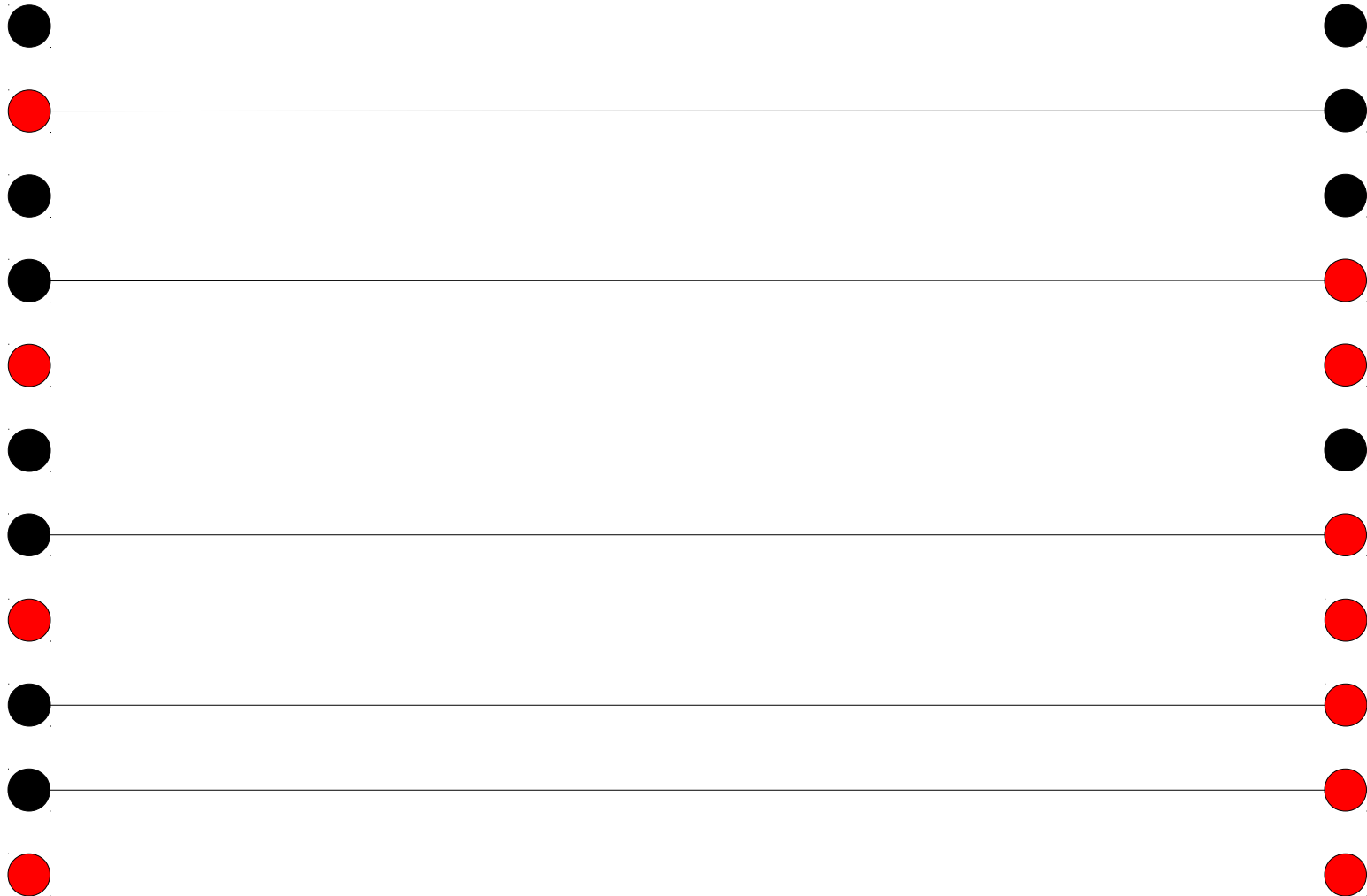
guiding solution



starting solution x

PR example

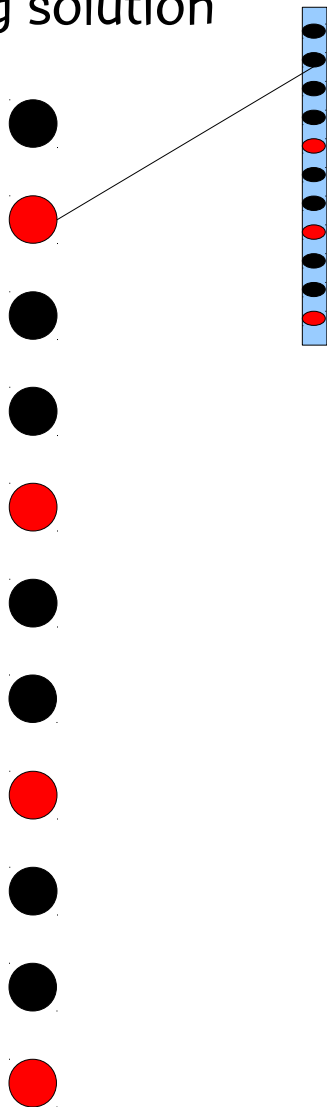
guiding solution y



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



starting solution

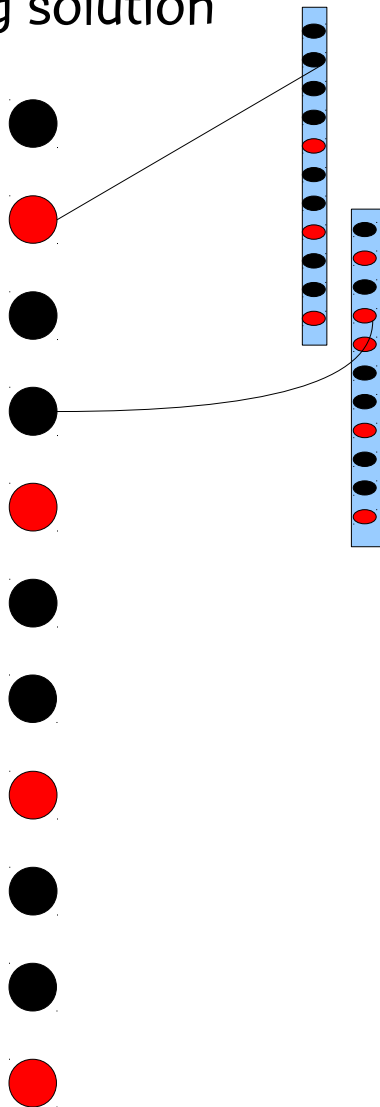


PR example

guiding solution



starting solution

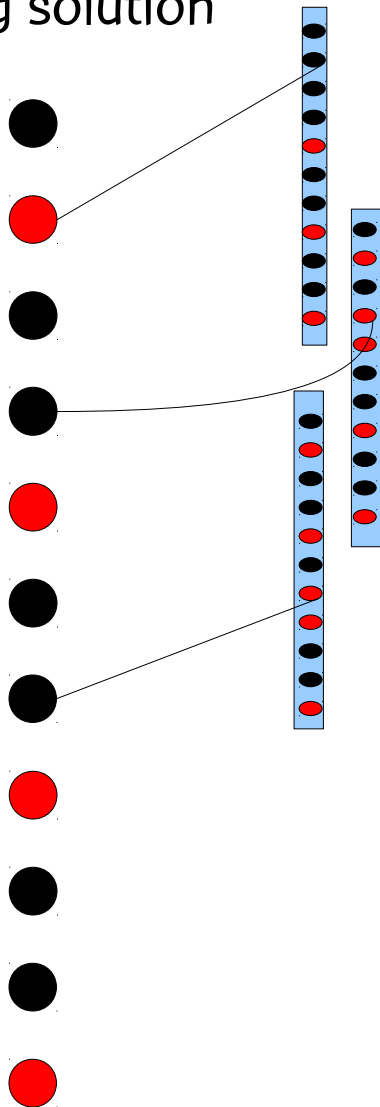


PR example

guiding solution



starting solution

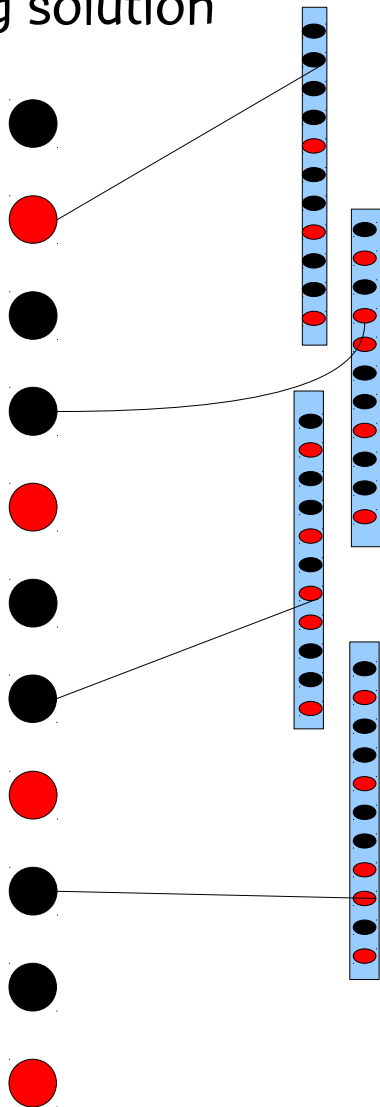


PR example

guiding solution



starting solution

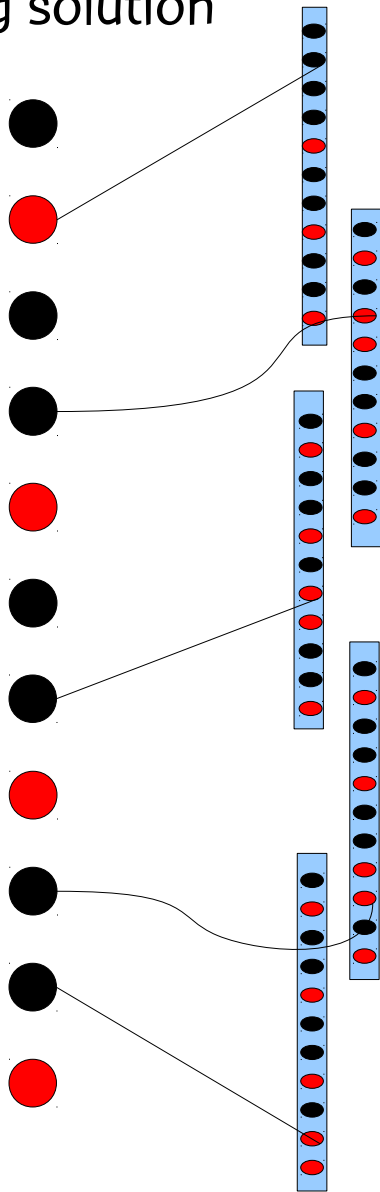


PR example

guiding solution



starting solution

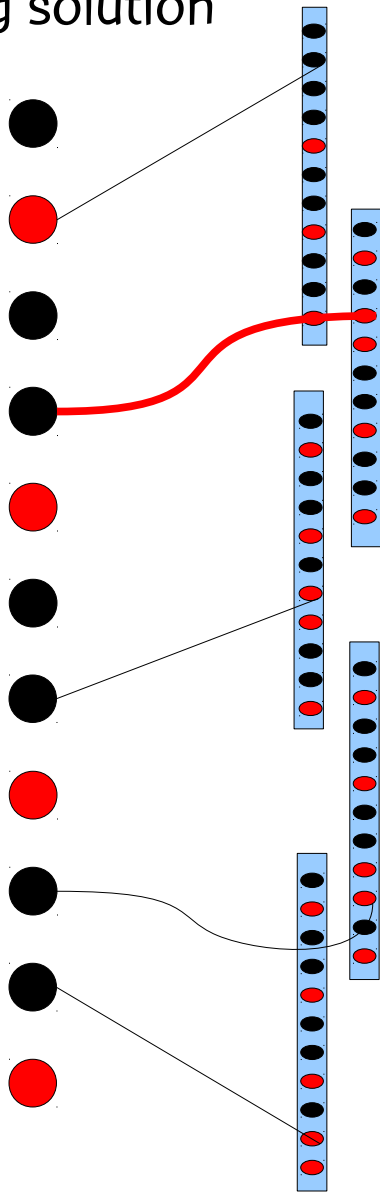


PR example

guiding solution



starting solution

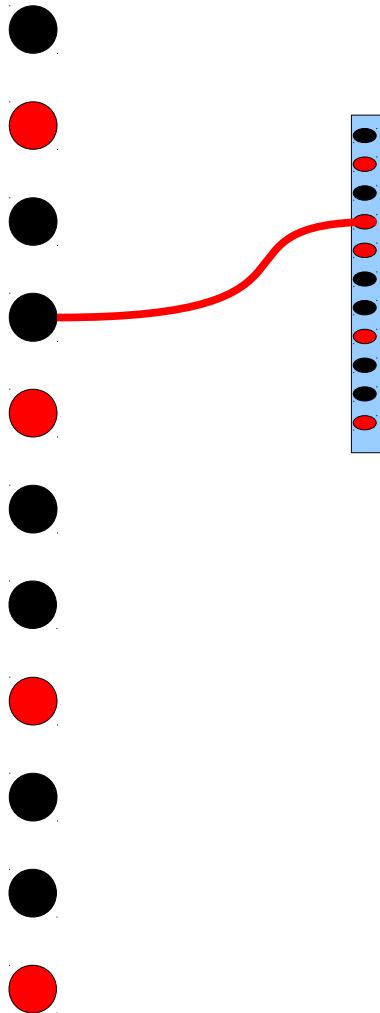


PR example

guiding solution



starting solution

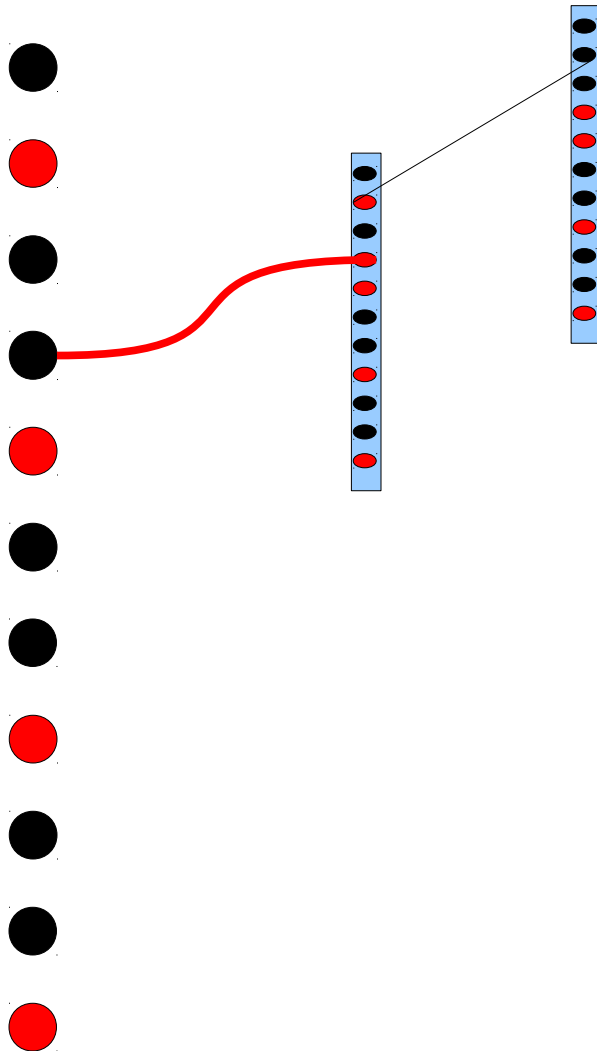


PR example

guiding solution



starting solution

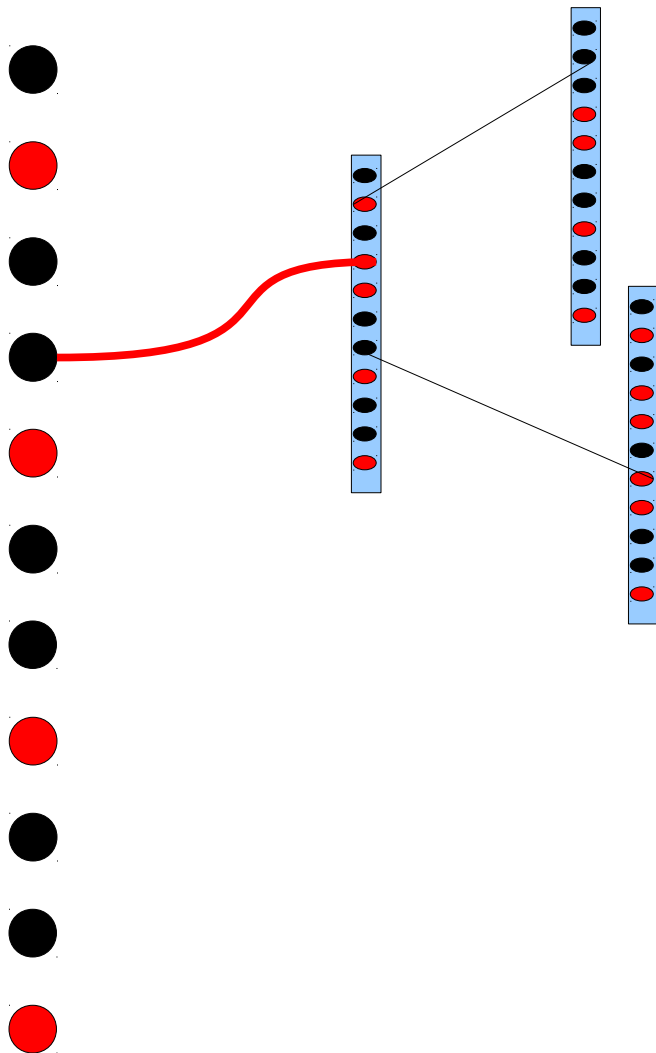


PR example

guiding solution



starting solution



PR example

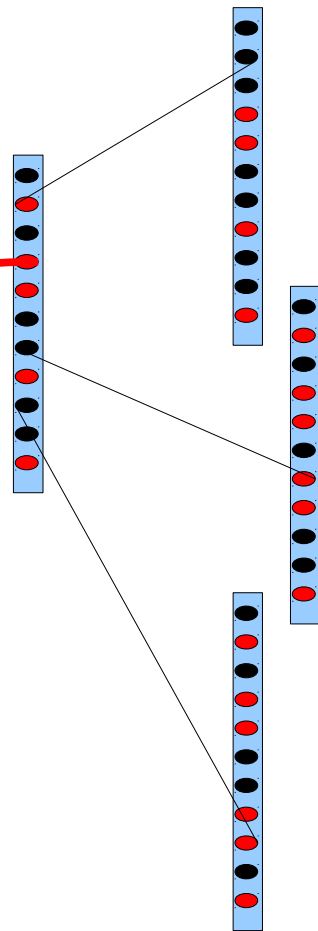
guiding solution



starting solution



PR example



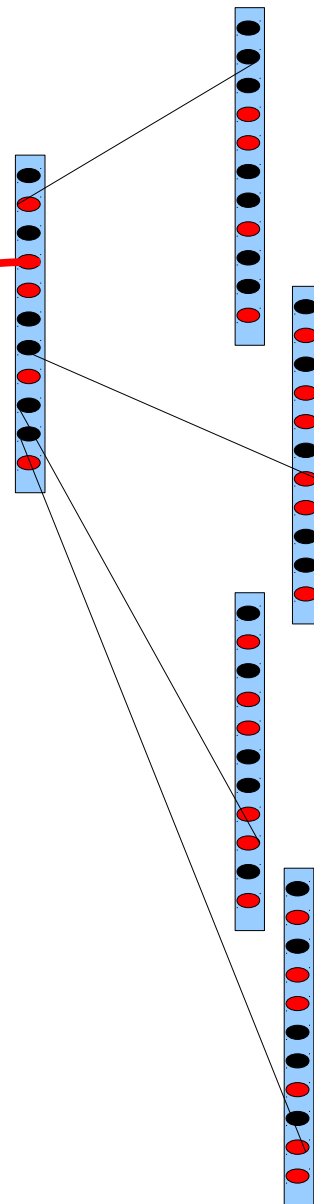
guiding solution



starting solution



PR example



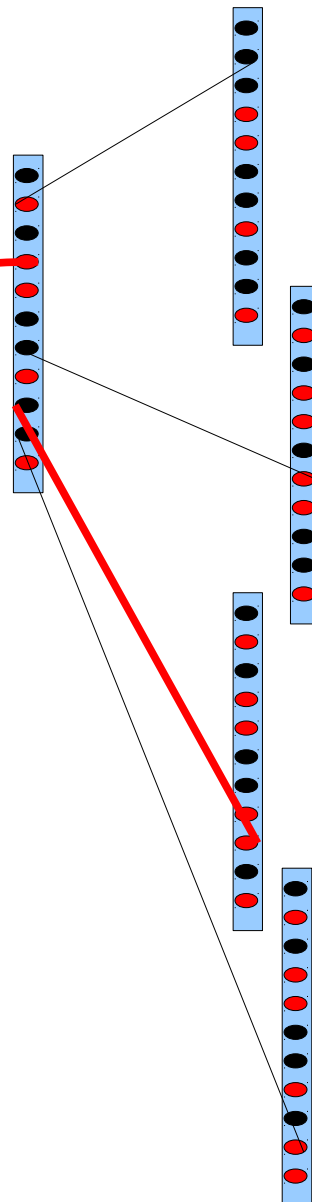
guiding solution



starting solution



PR example



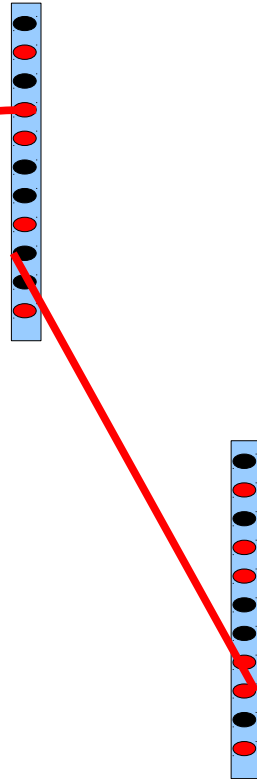
guiding solution



starting solution



PR example



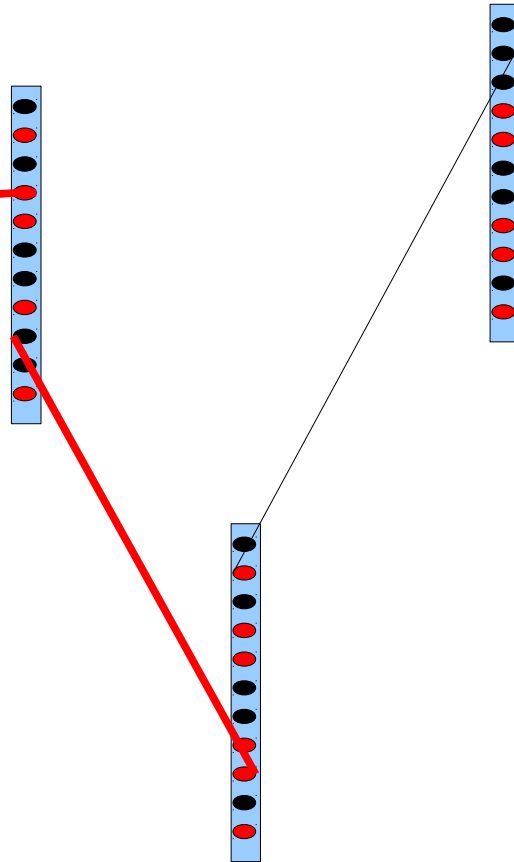
guiding solution



starting solution



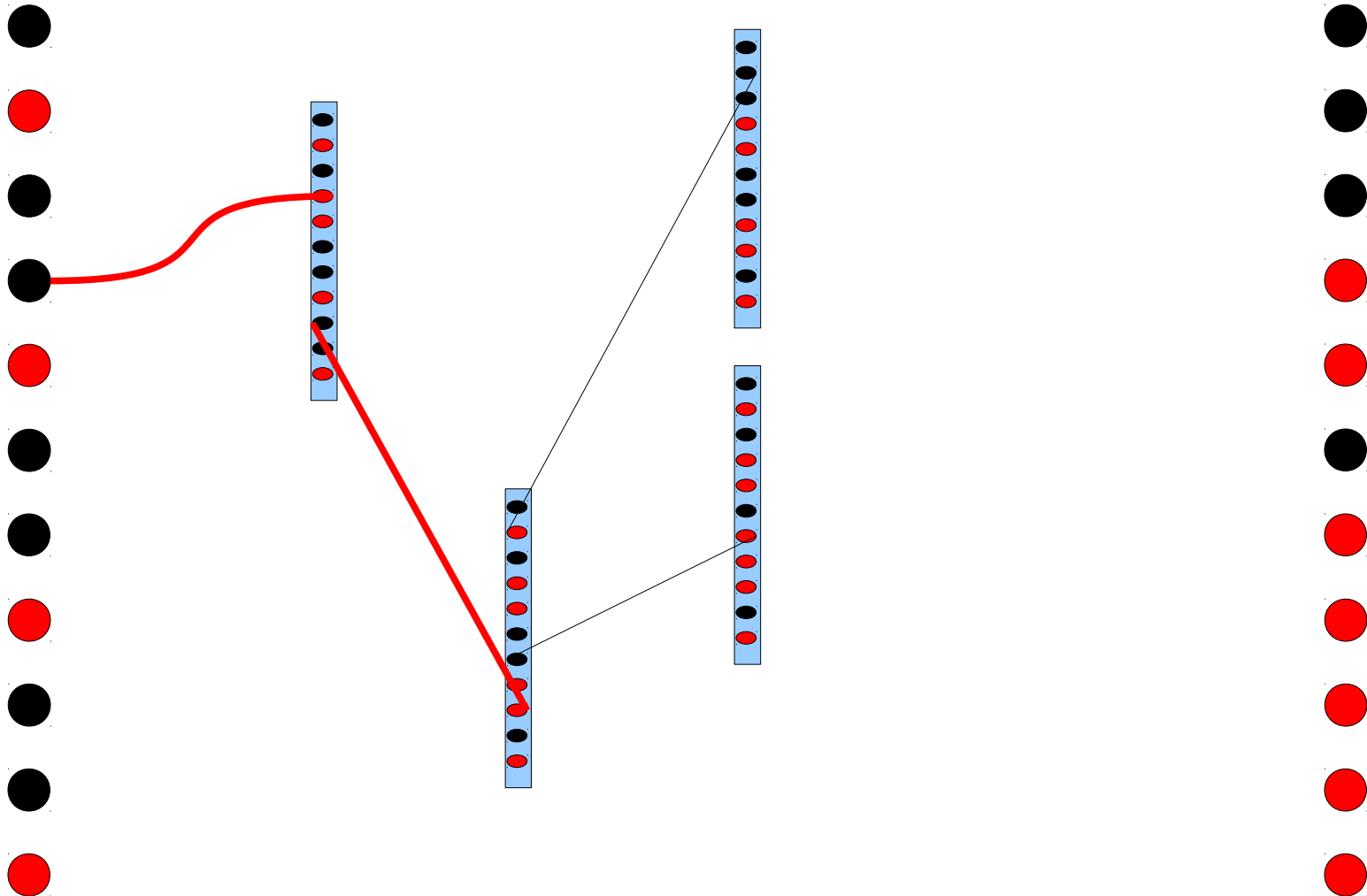
PR example



guiding solution



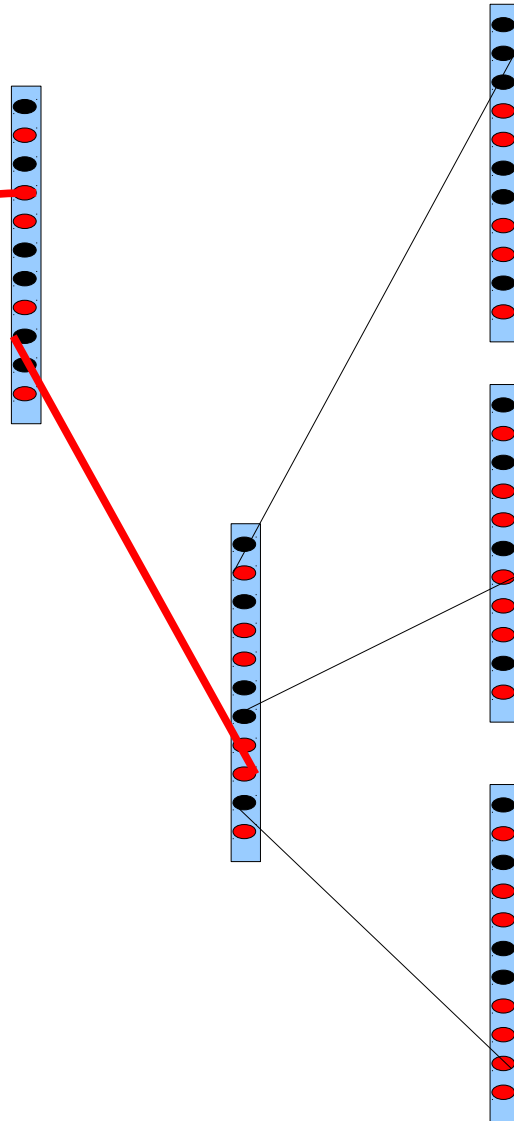
guiding solution



starting solution



PR example



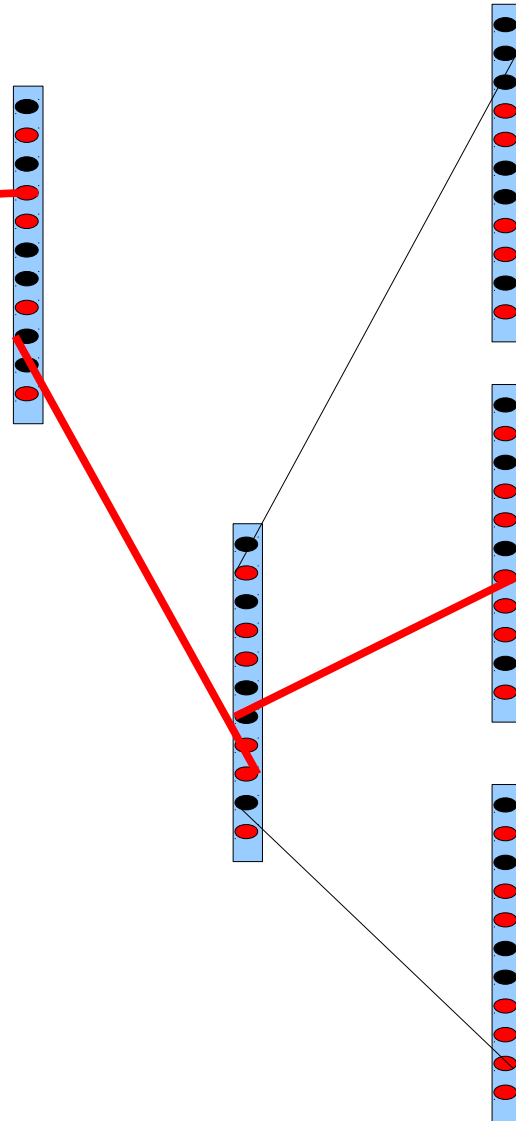
guiding solution



starting solution



PR example



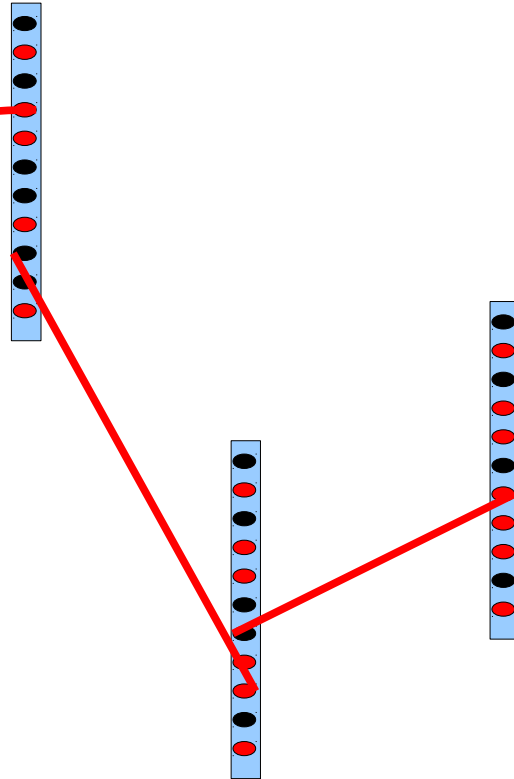
guiding solution



starting solution



PR example



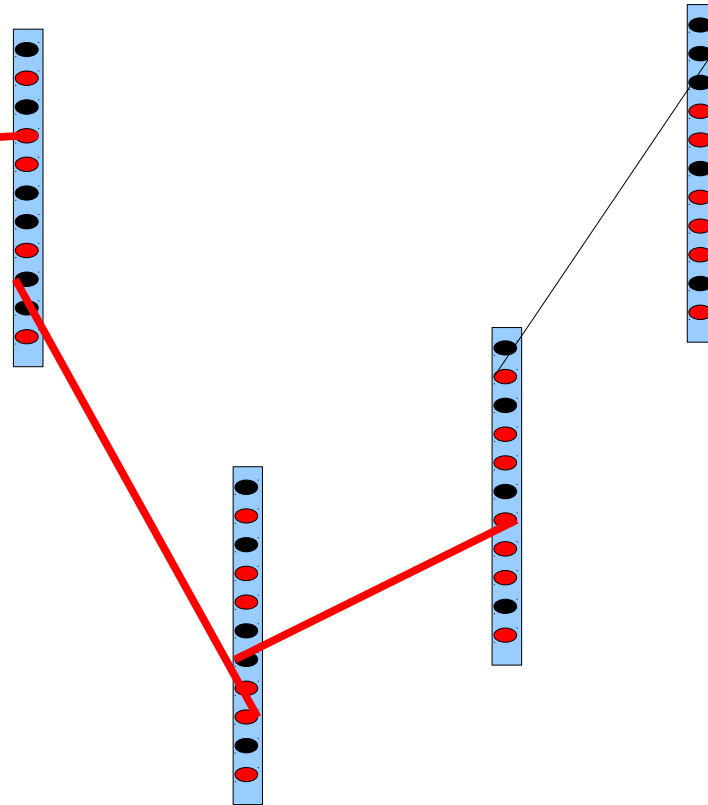
guiding solution



starting solution



PR example



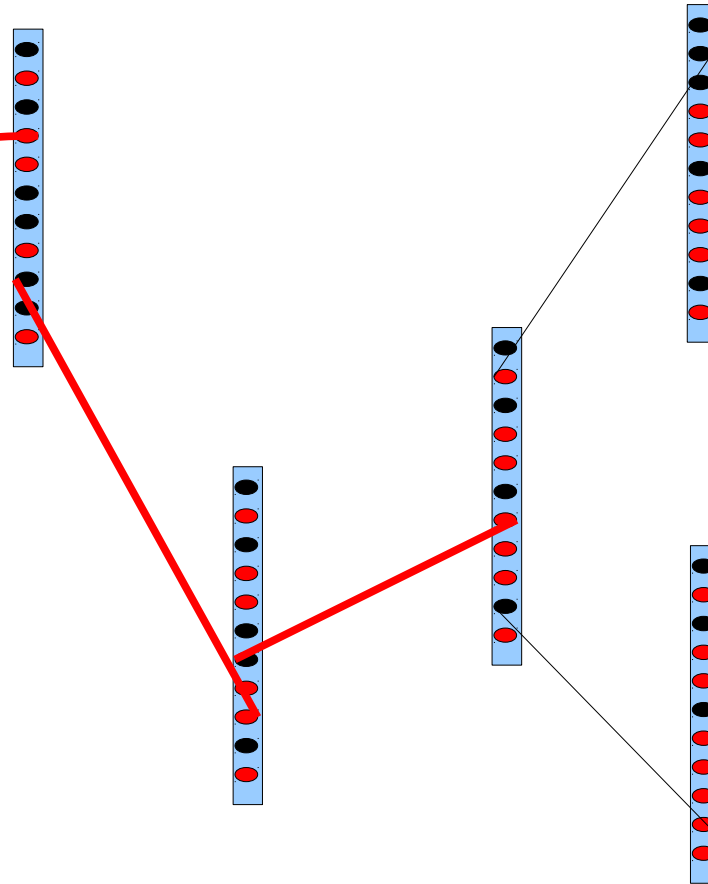
guiding solution



starting solution



PR example



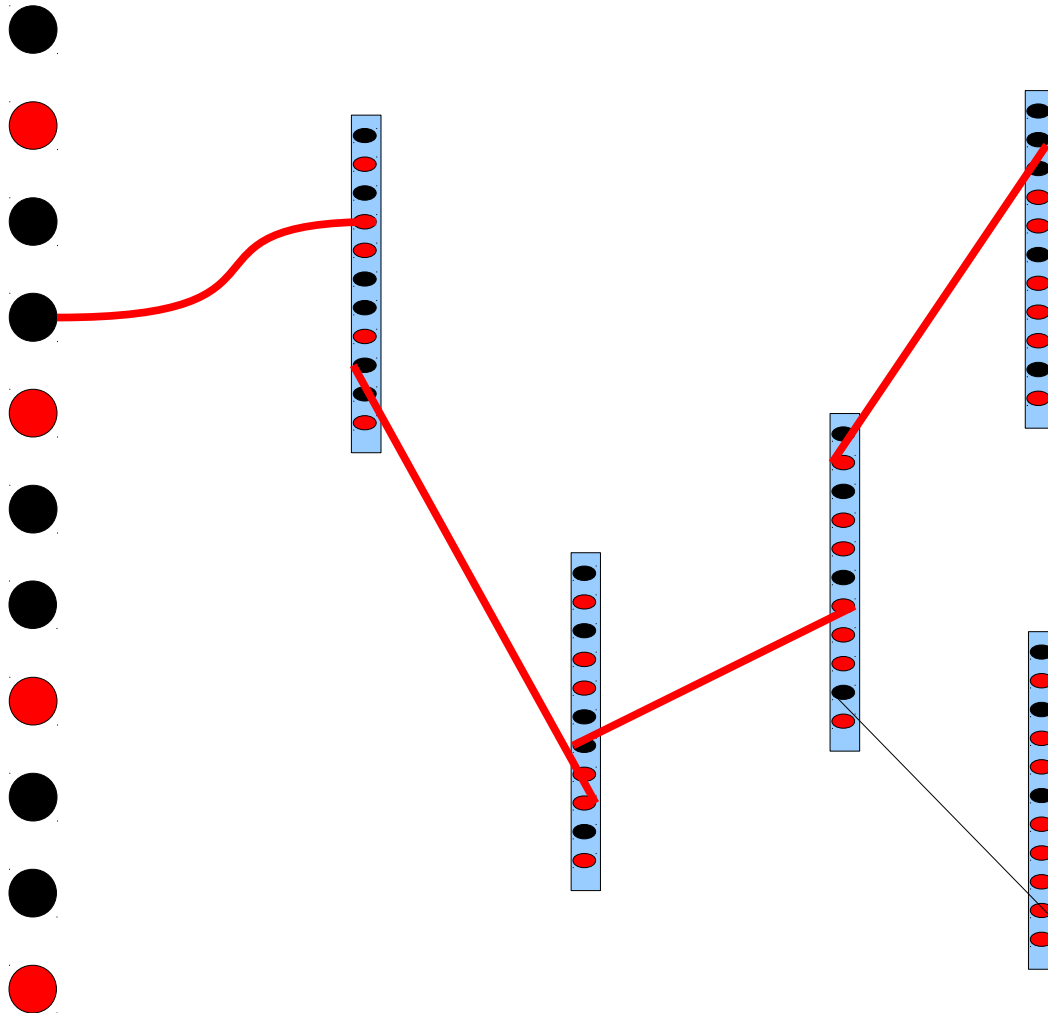
guiding solution



starting solution

PR example

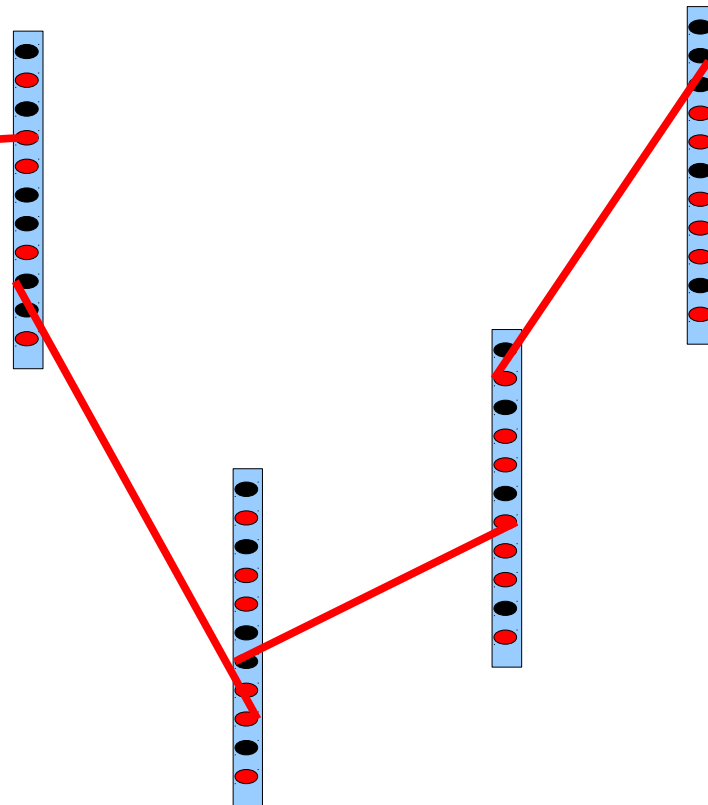
guiding solution



starting solution



PR example



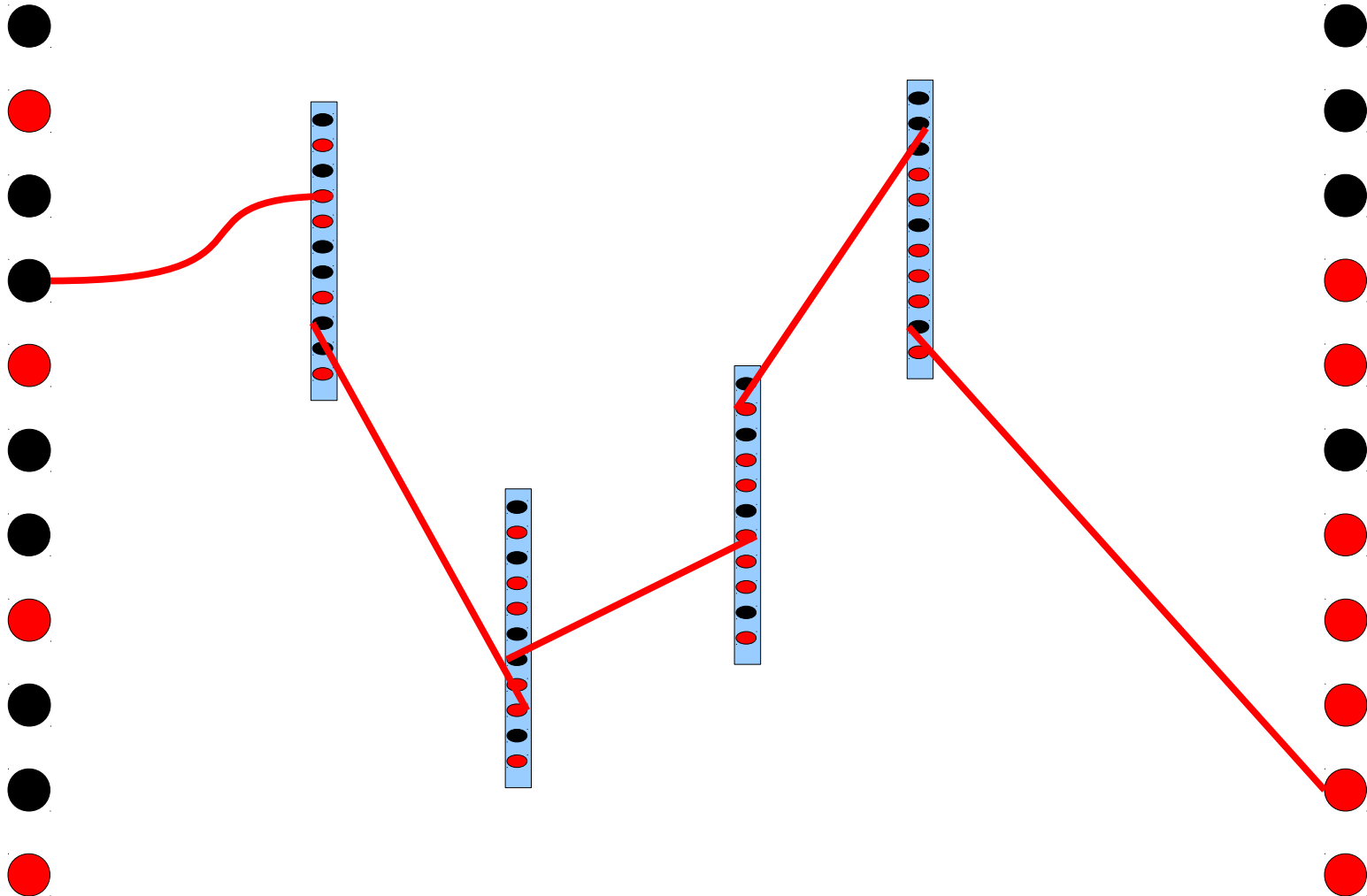
guiding solution



starting solution

PR example

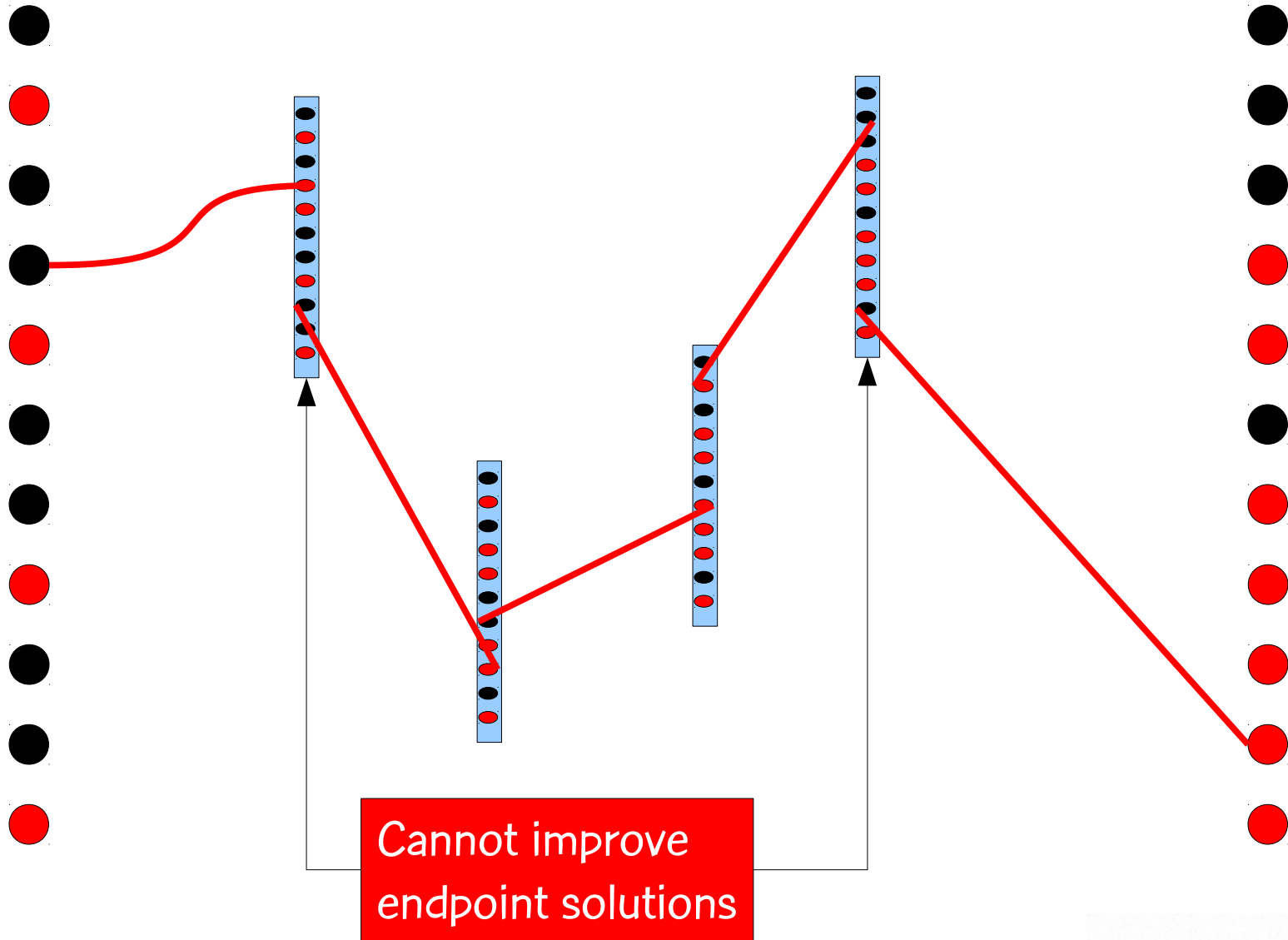
guiding solution



starting solution

PR example

guiding solution



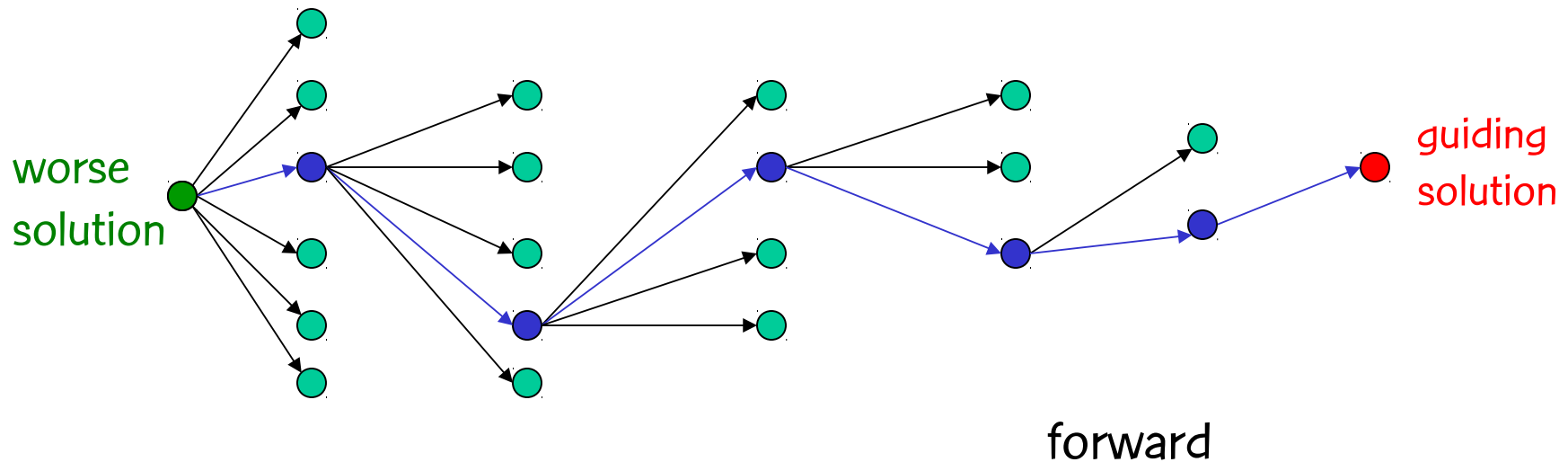
guiding solution



Forward path-relinking

Variants: trade-offs between computation time and solution quality

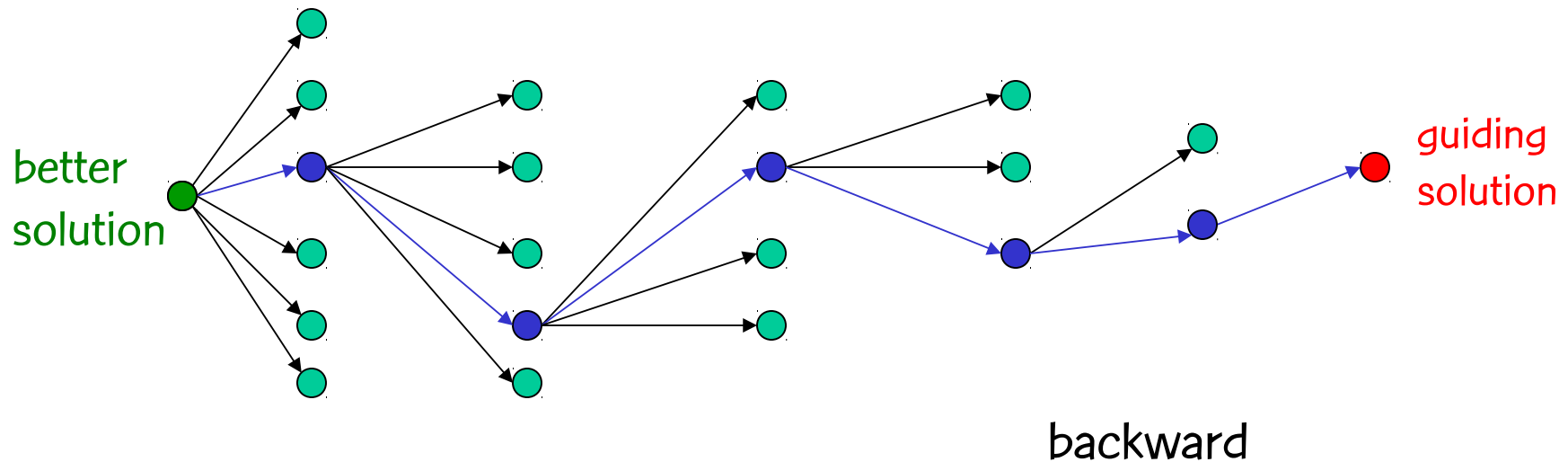
Forward PR adopts as initial solution the worse of the two input solutions and uses the better solution as the guide.



Backward path-relinking

Variants: trade-offs between computation time and solution quality

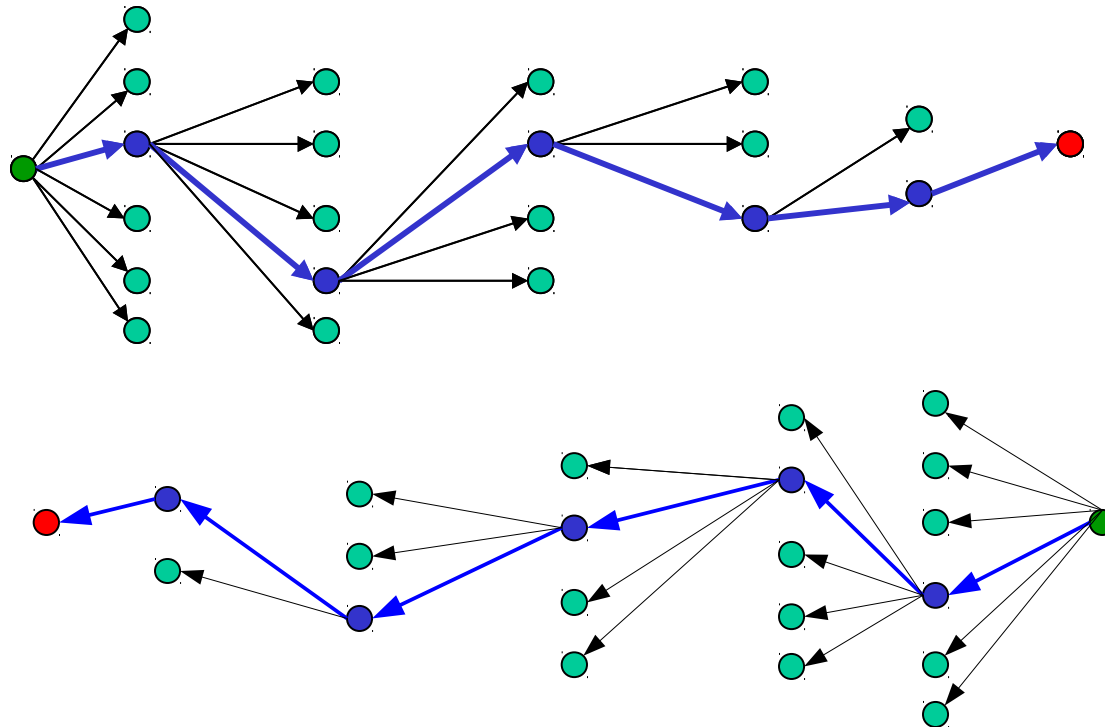
Backward PR usually does better: **Better to start from the best of the two input solutions**, neighborhood of the initial solution is explored more than of the guide!



Back and forth path-relinking

Variants: trade-offs between computation time and solution quality

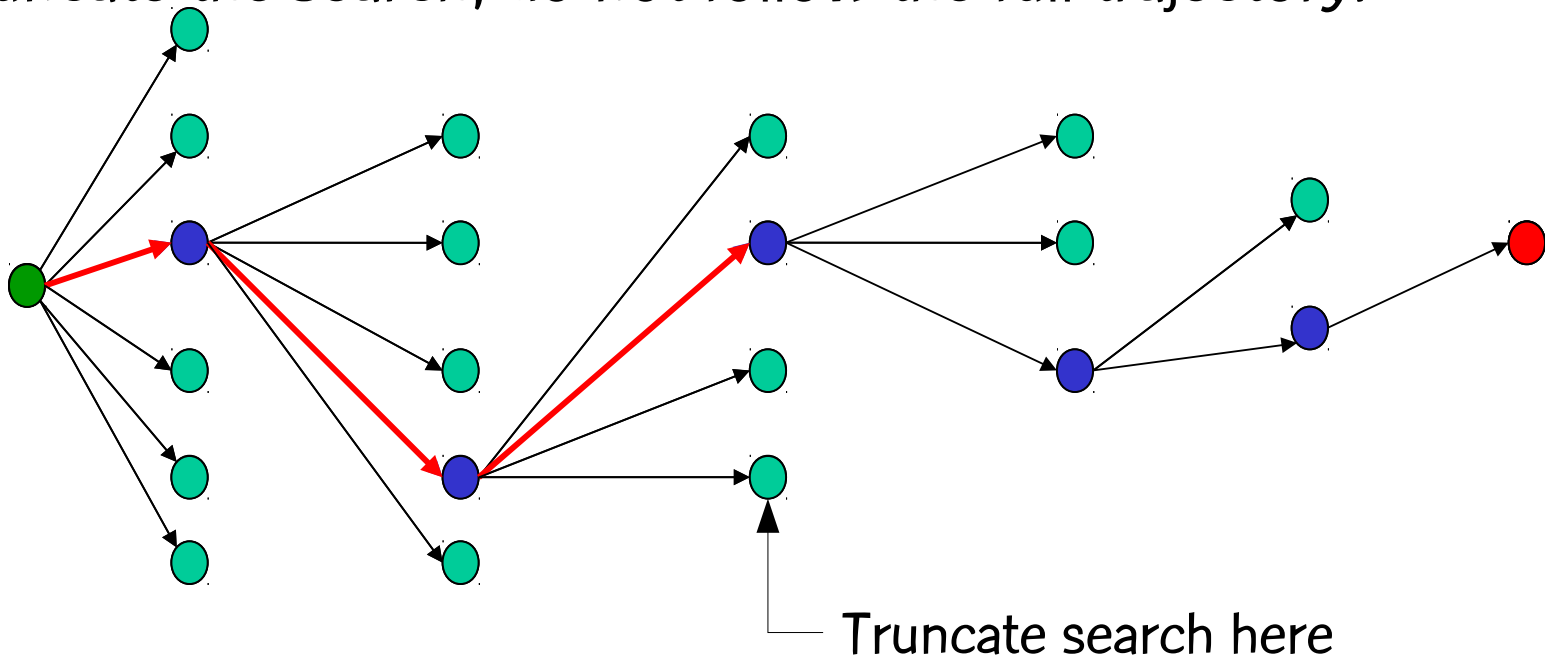
Explore both trajectories: **twice as much time**, often with only marginal improvements!



Truncated path-relinking

Variants: trade-offs between computation time and solution quality

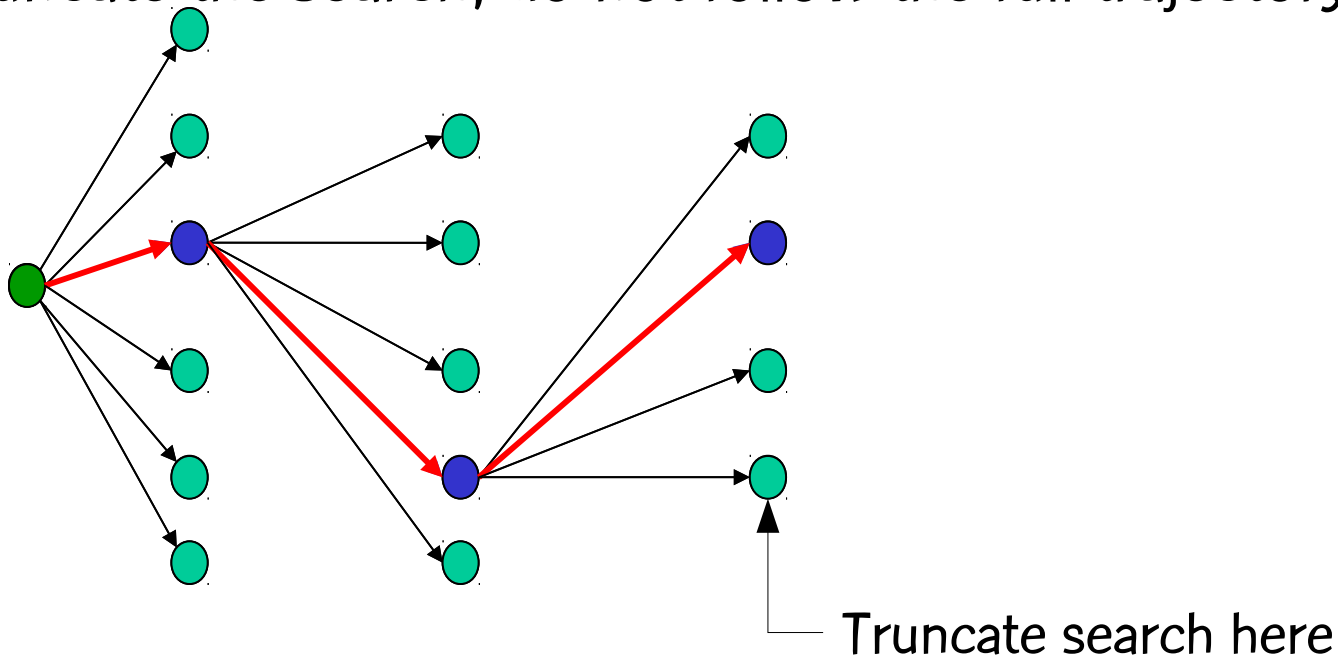
Truncate the search, do not follow the full trajectory.



Truncated path-relinking

Variants: trade-offs between computation time and solution quality

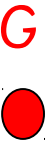
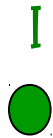
Truncate the search, do not follow the full trajectory.



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

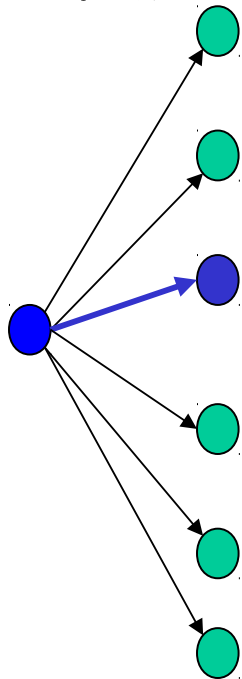
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

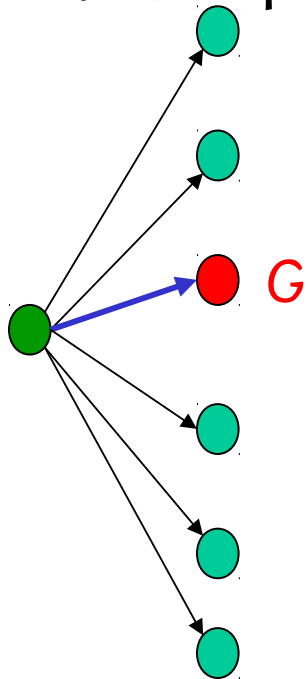
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

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



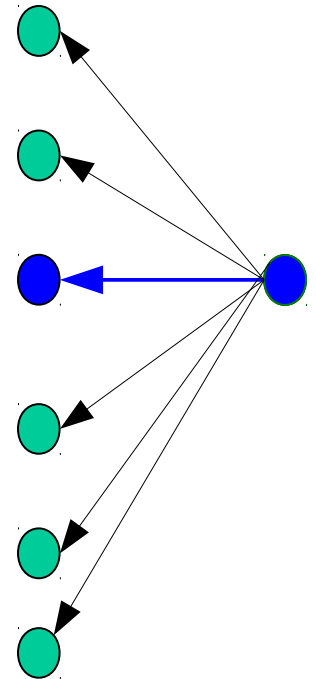
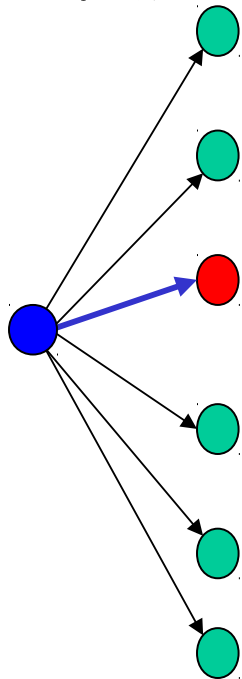
I ●



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

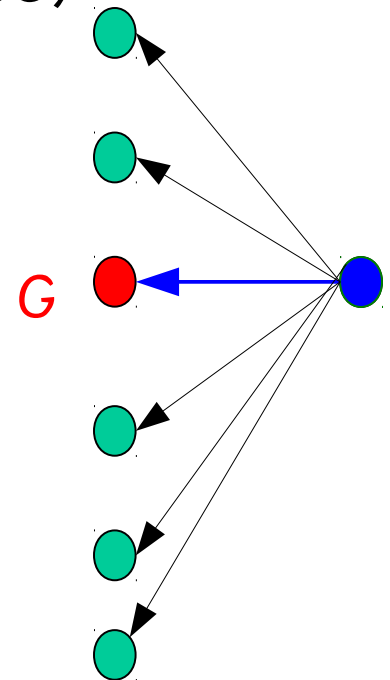
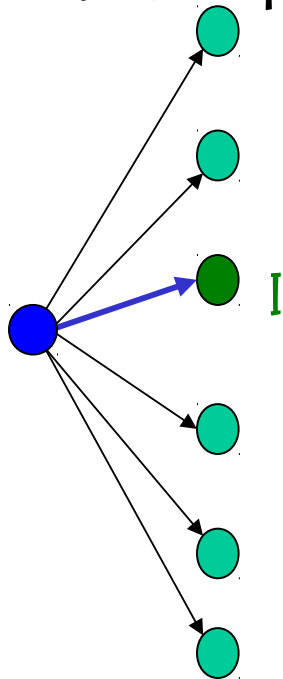
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

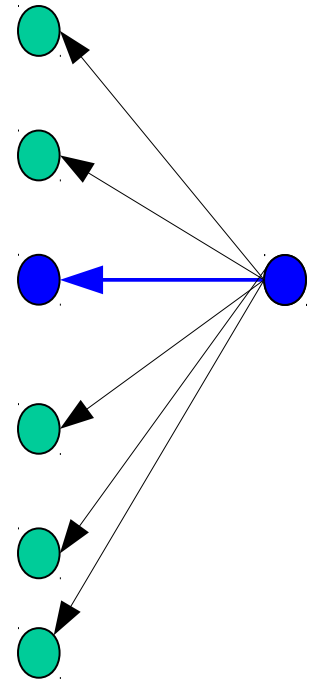
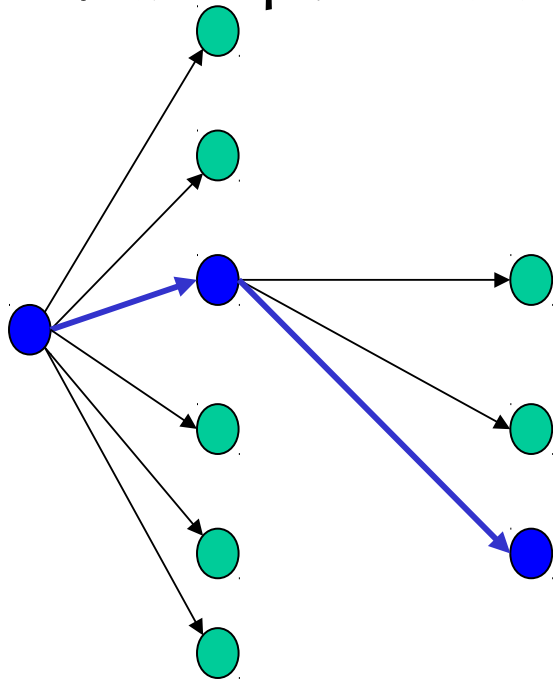
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

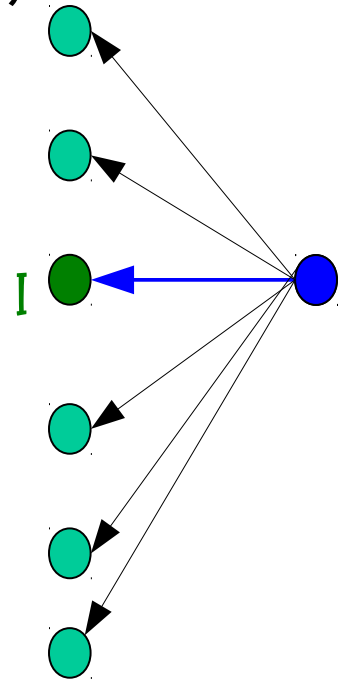
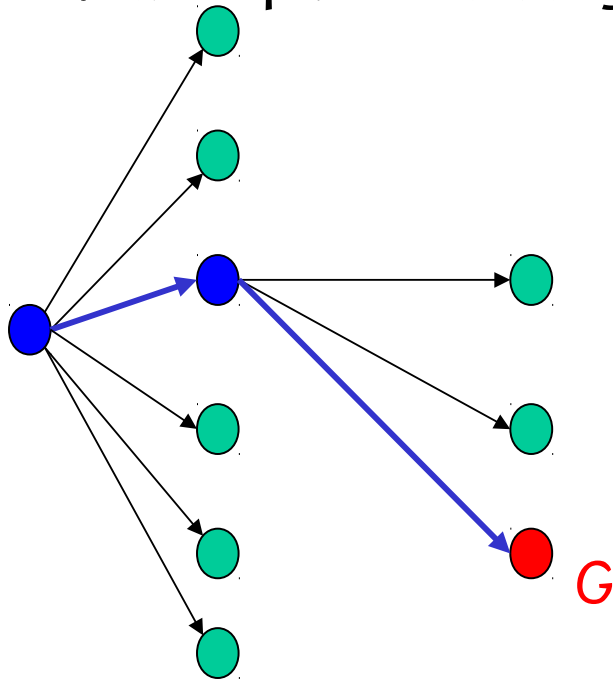
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

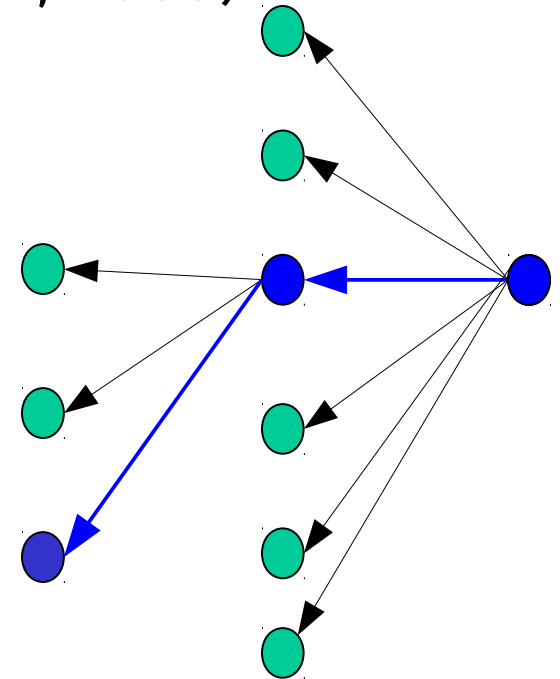
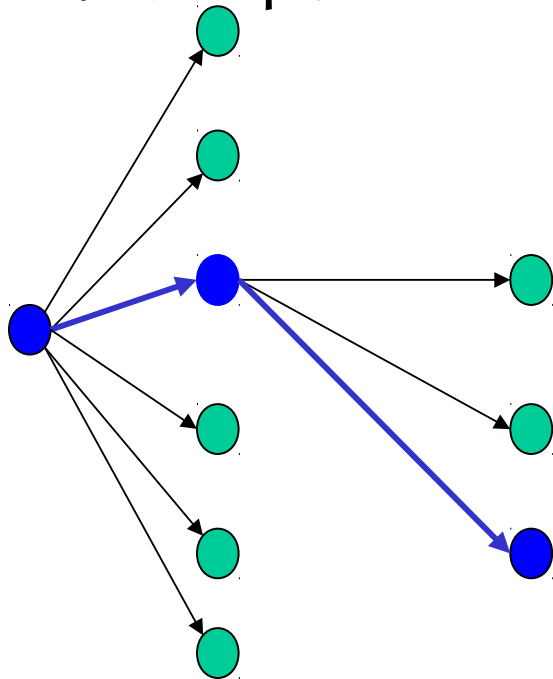
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

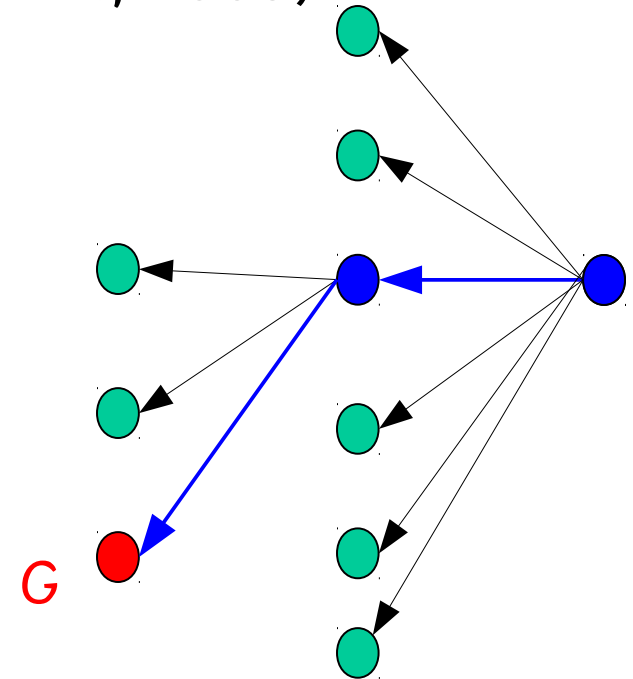
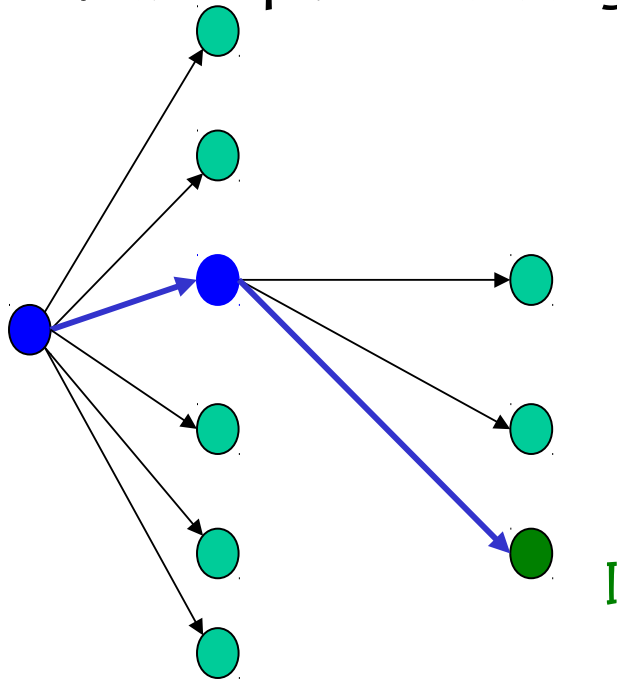
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

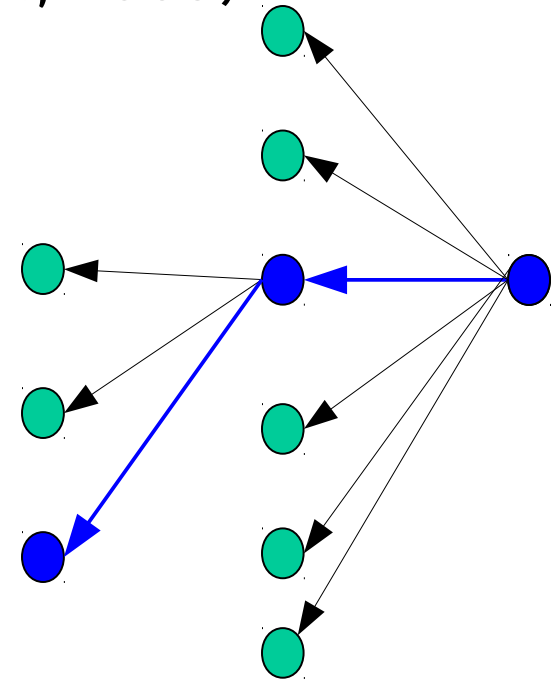
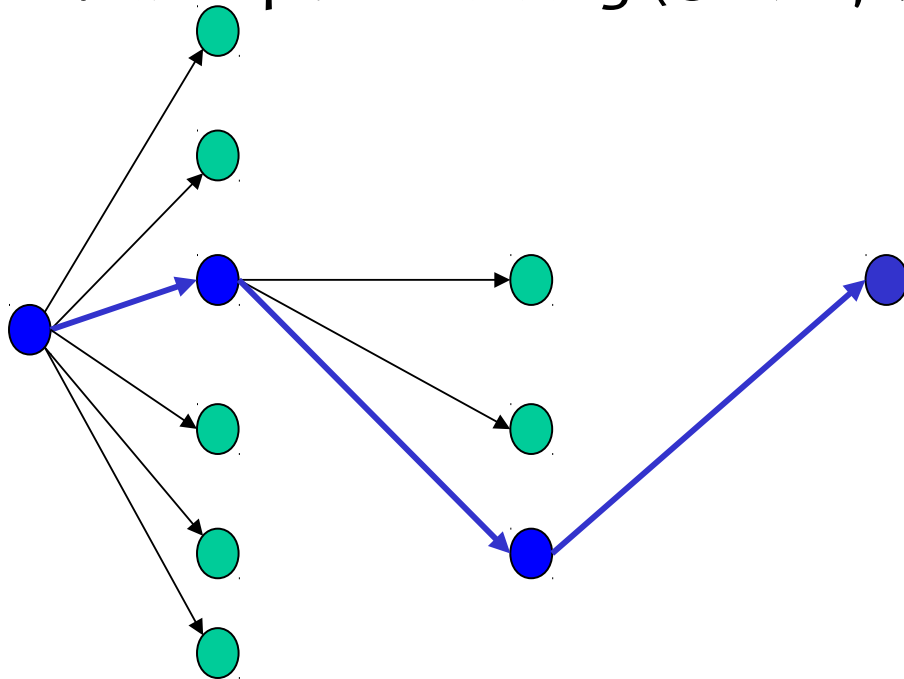
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

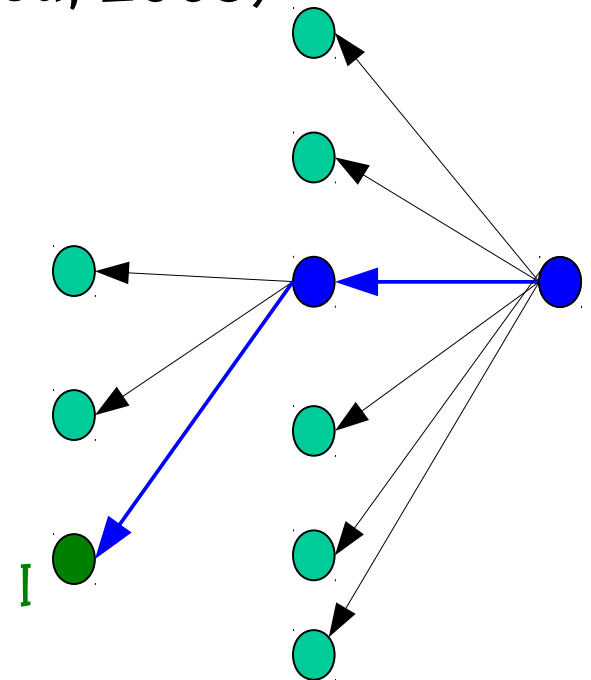
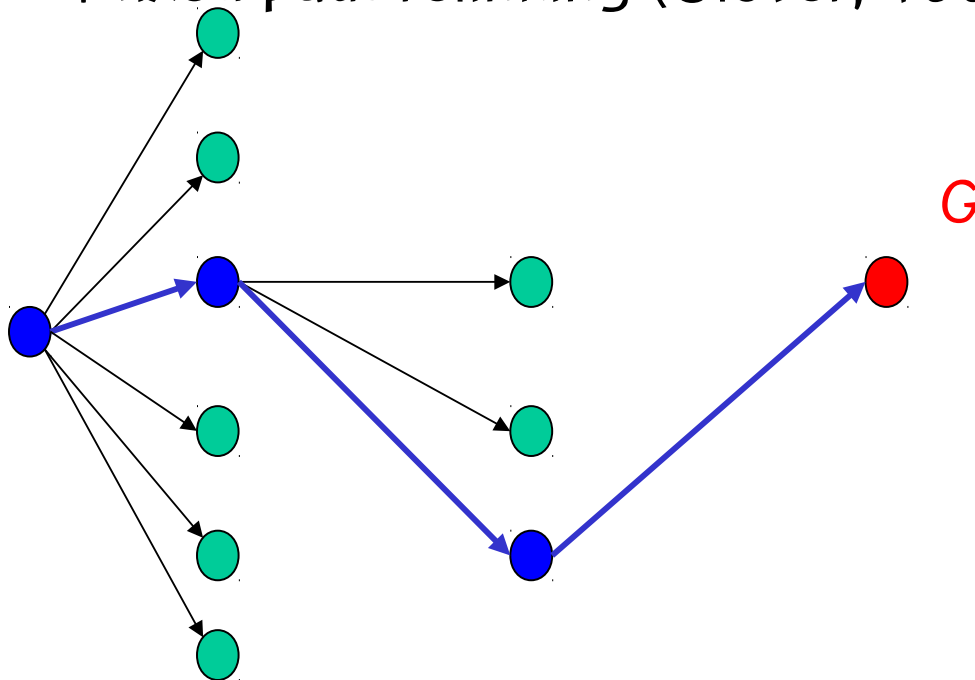
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

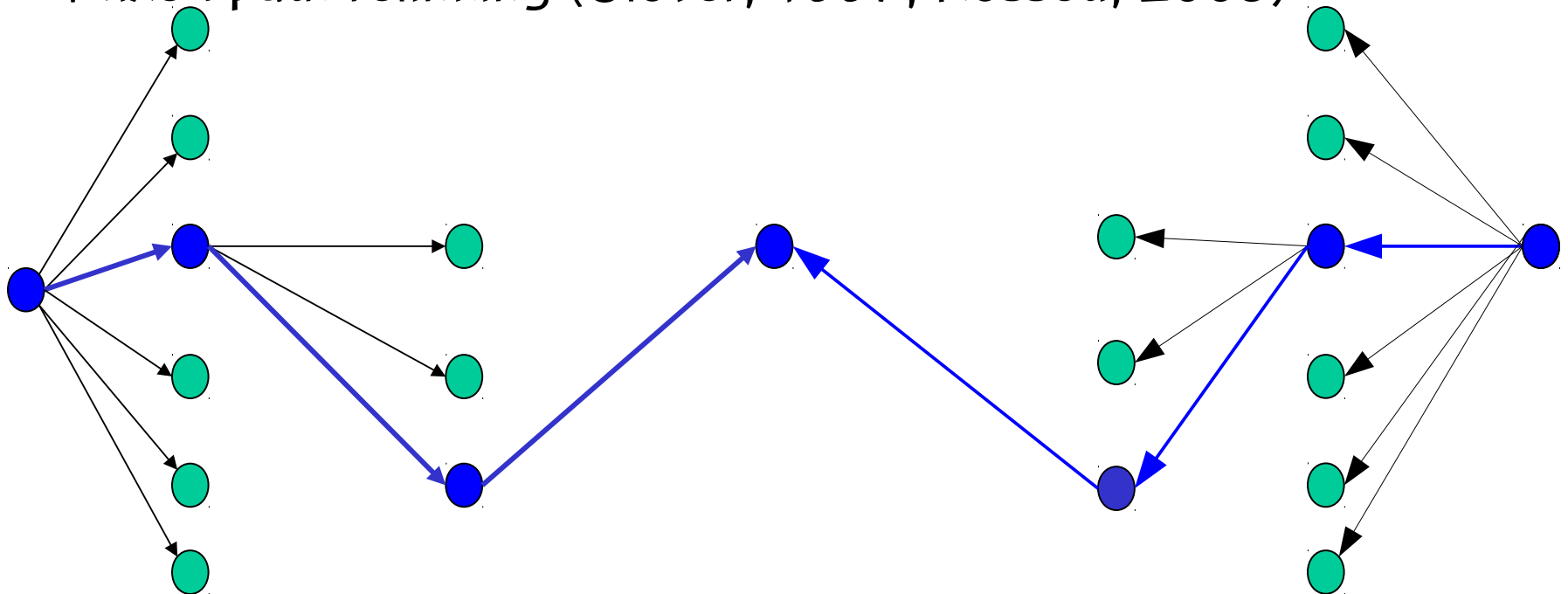
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



Mixed path-relinking

Variants: trade-offs between computation time and solution quality

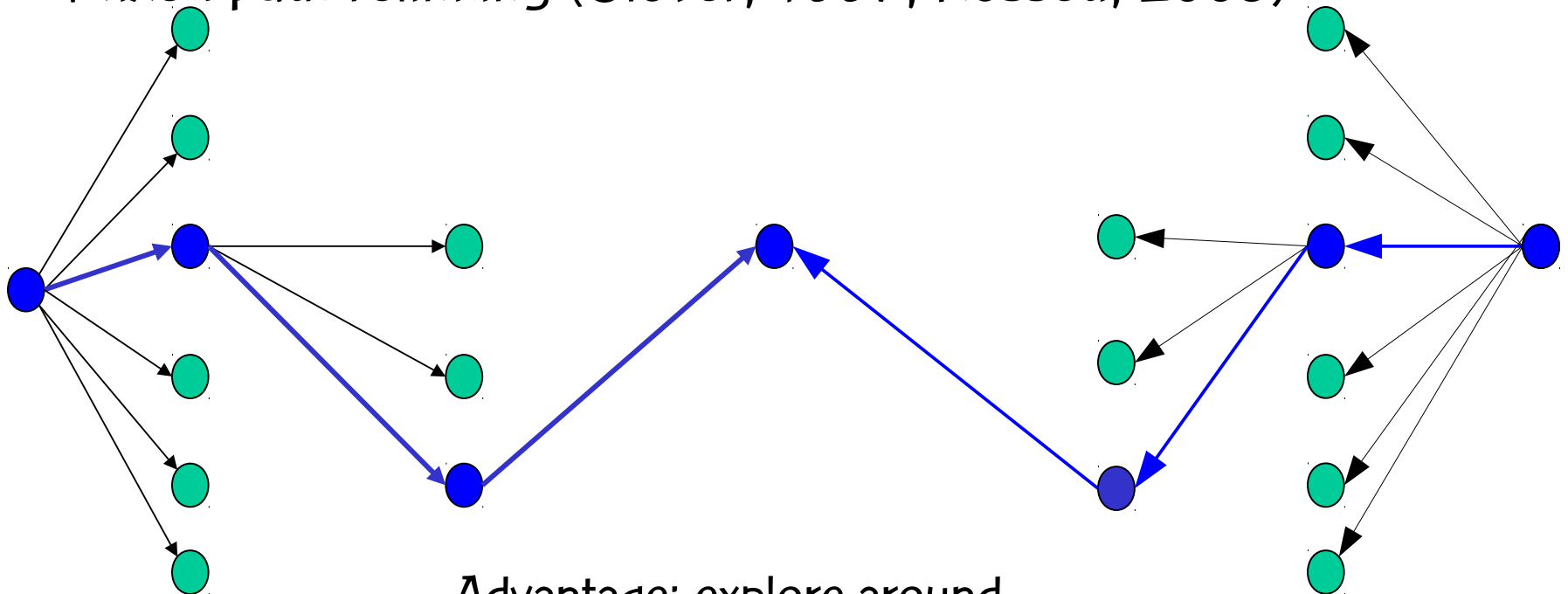
Mixed path-relinking (Glover, 1997; Rosseti, 2003)



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.

Rethink Possible

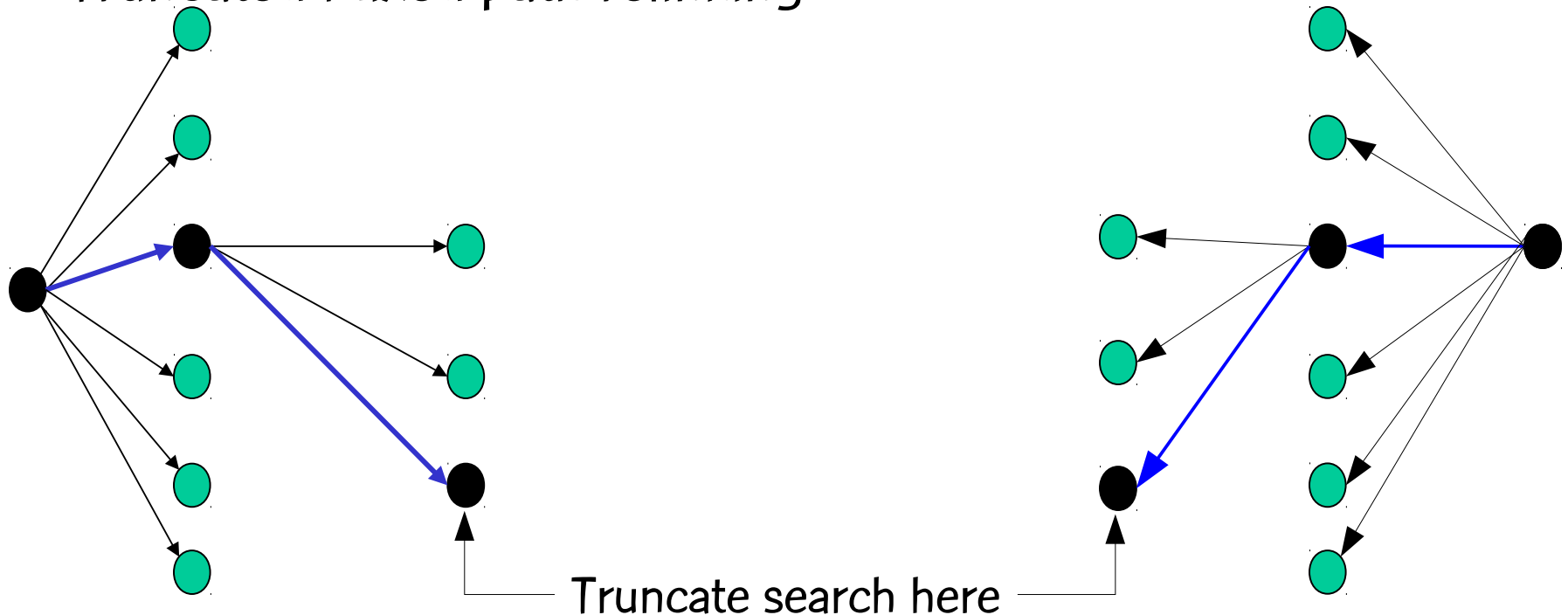


att.com/rethinkpossible

Truncated mixed path-relinking

Variants: trade-offs between computation time and solution quality

Truncated mixed path-relinking

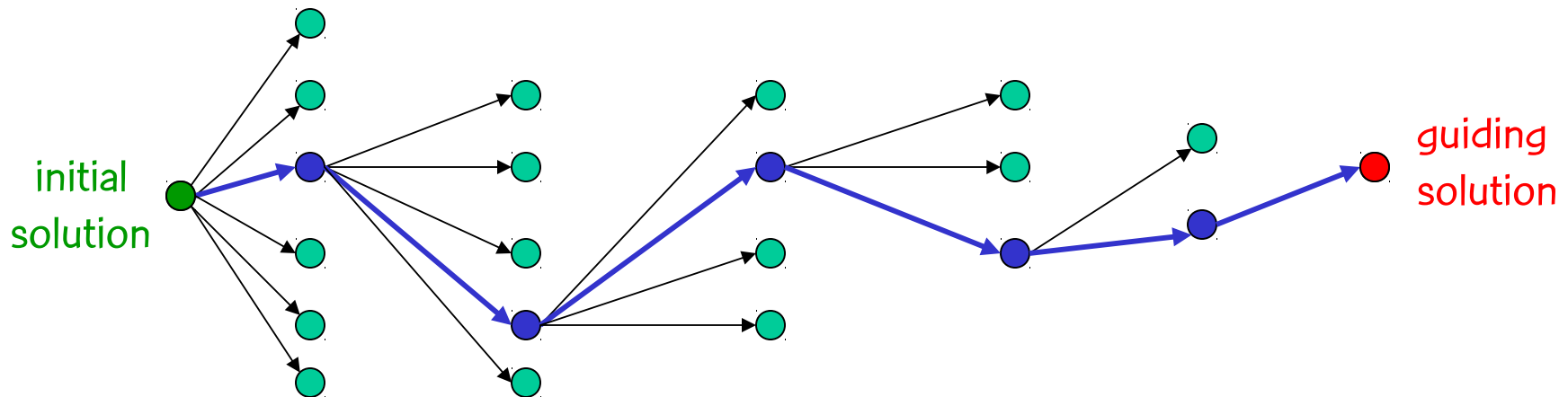


Greedy randomized adaptive path-relinking

Faria, Binato, Resende, & Falcão (2001, 2005)

Incorporates semi-greediness into PR.

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

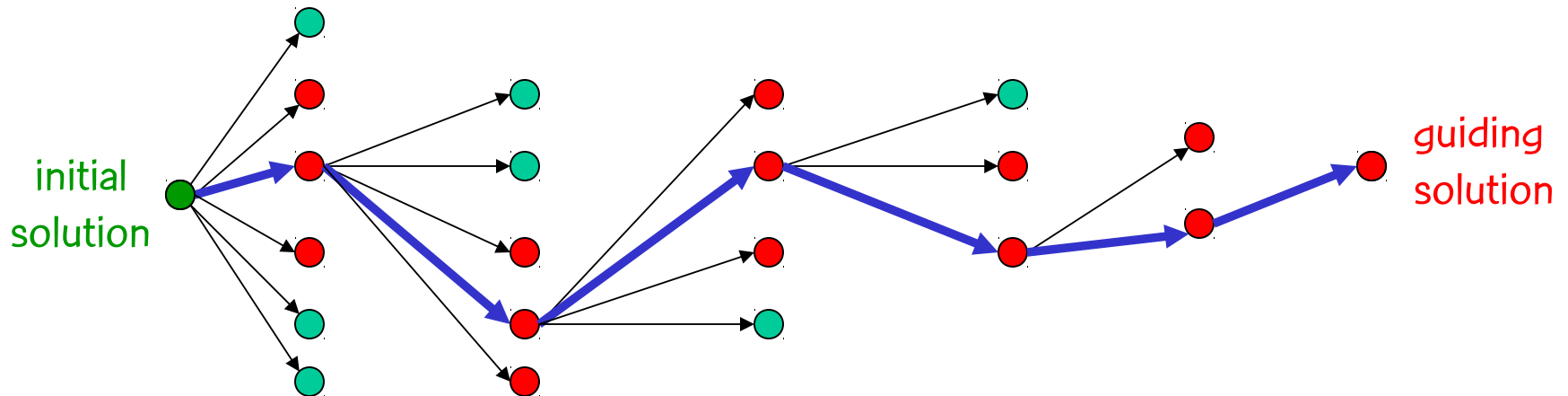


Greedy randomized adaptive path-relinking

Faria, Binato, Resende, & Falcão (2001, 2005)

Incorporates semi-greediness into PR.

graPR creates RCL with best moves: samples several paths

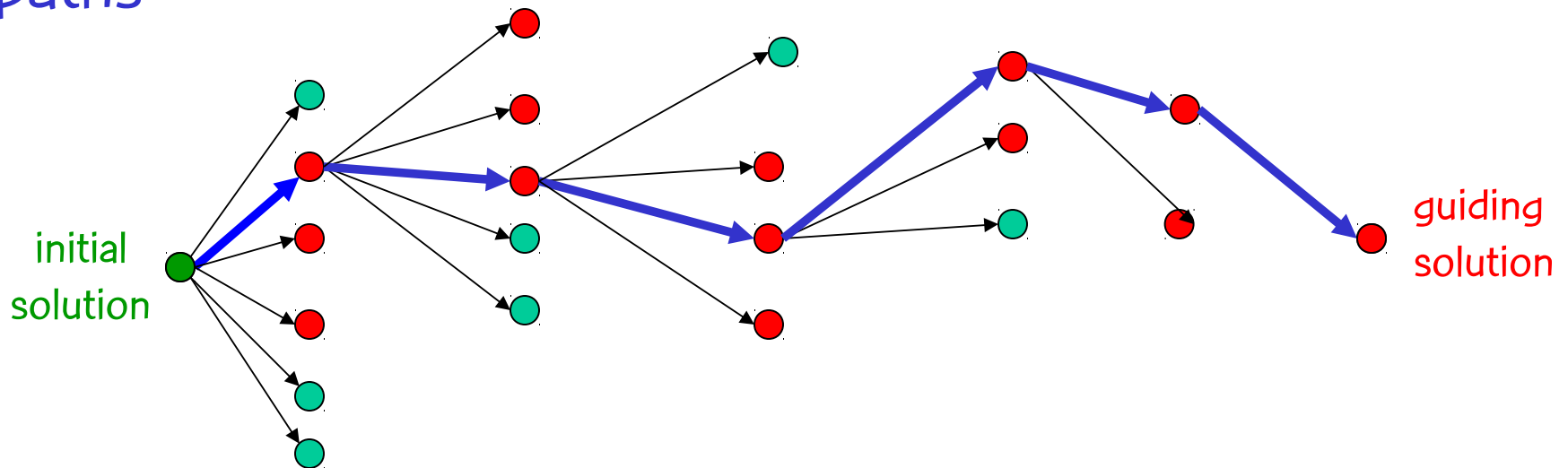


Greedy randomized adaptive path-relinking

Faria, Binato, Resende, & Falcão (2001, 2005)

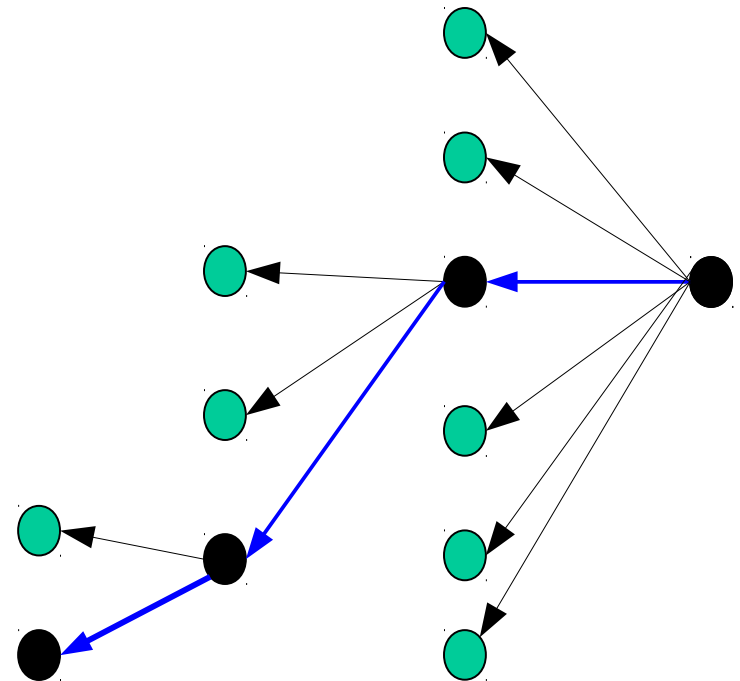
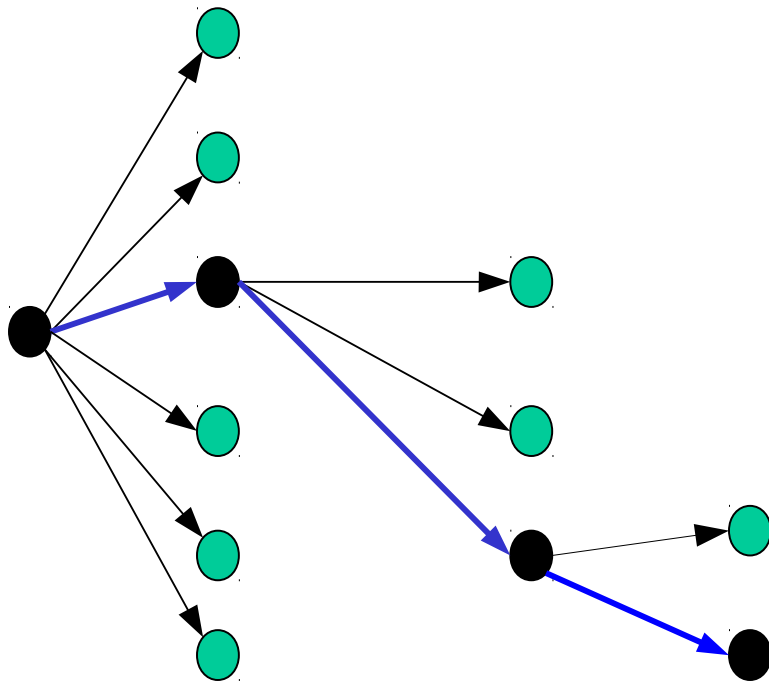
Incorporates semi-greediness into PR.

graPR creates RCL with best moves: samples several paths



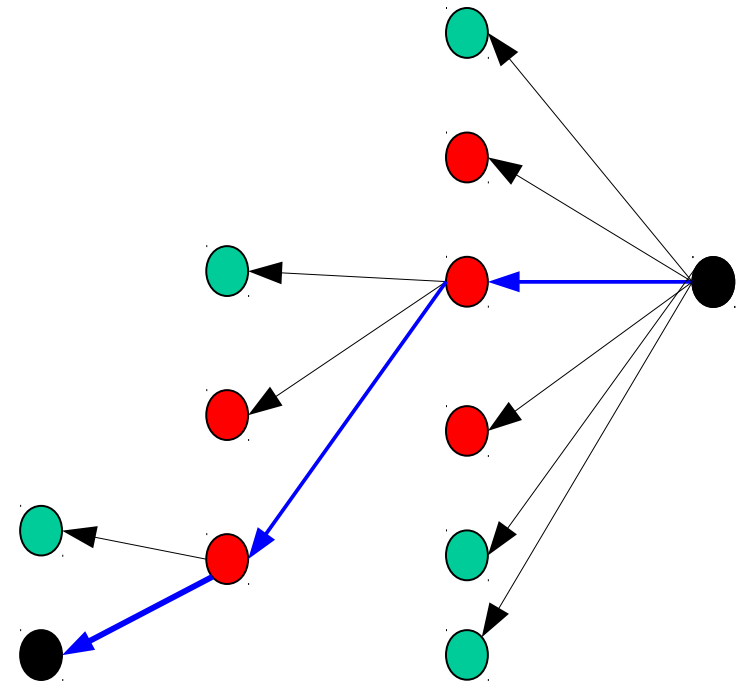
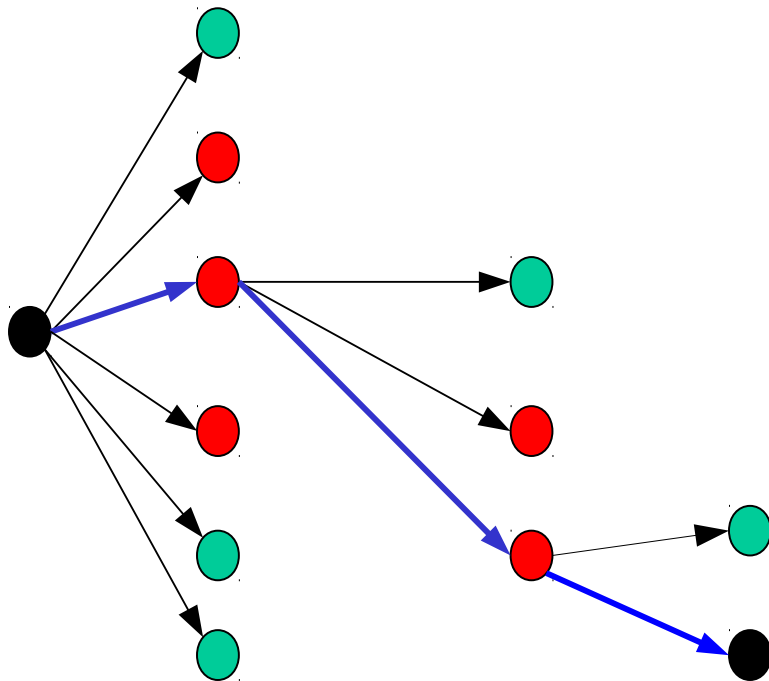
Truncated mixed graPR

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

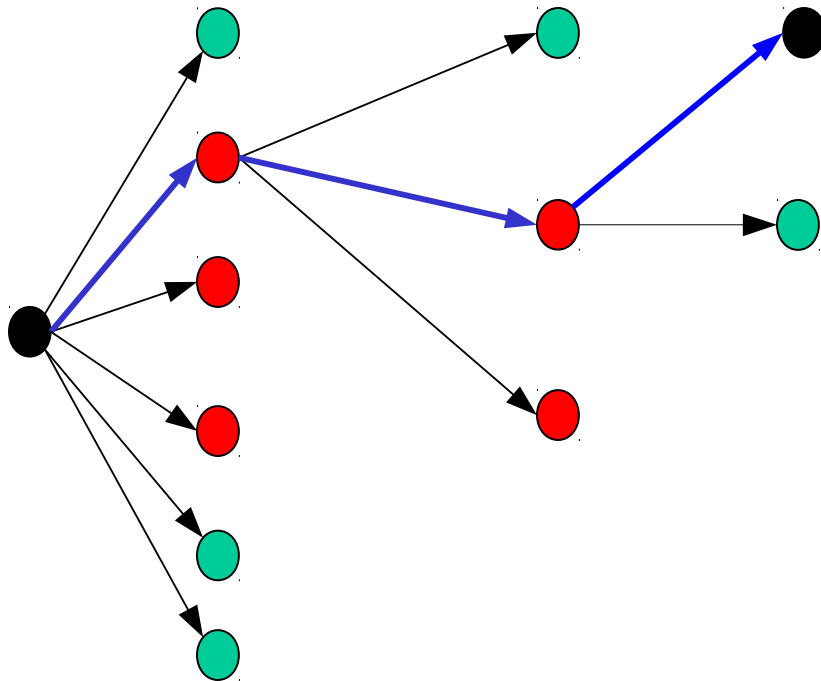


Truncated mixed graPR

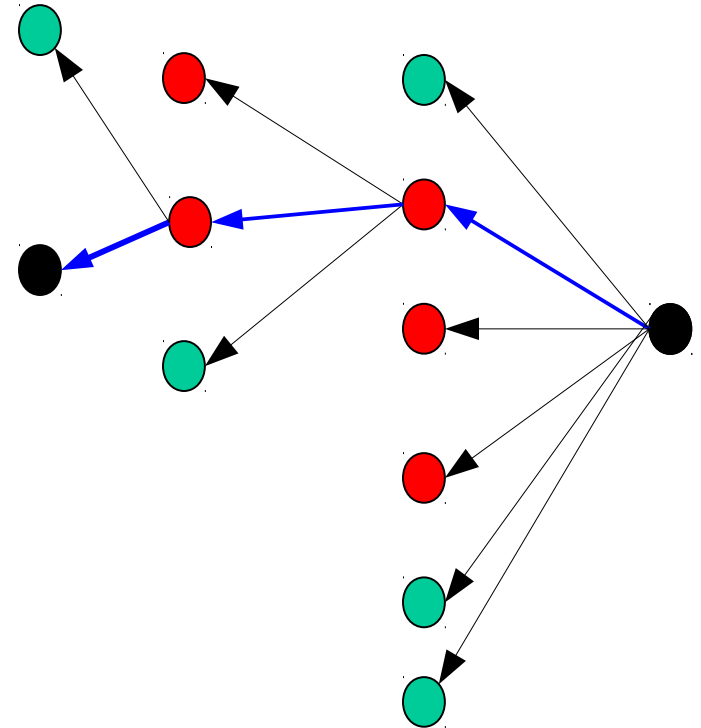
With high probability, truncated mixed graPR explores different path segments each time it is executed between the same pair of solutions.



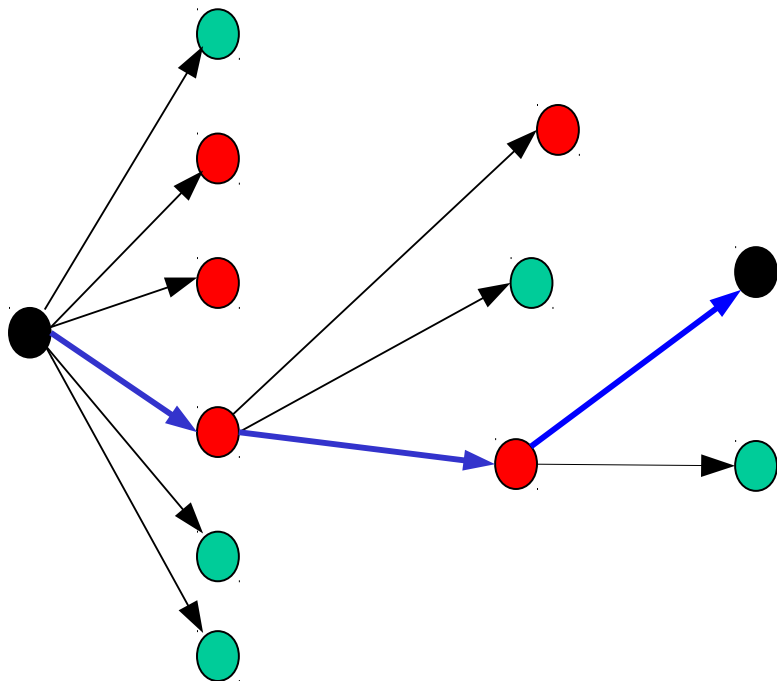
Truncated mixed graPR



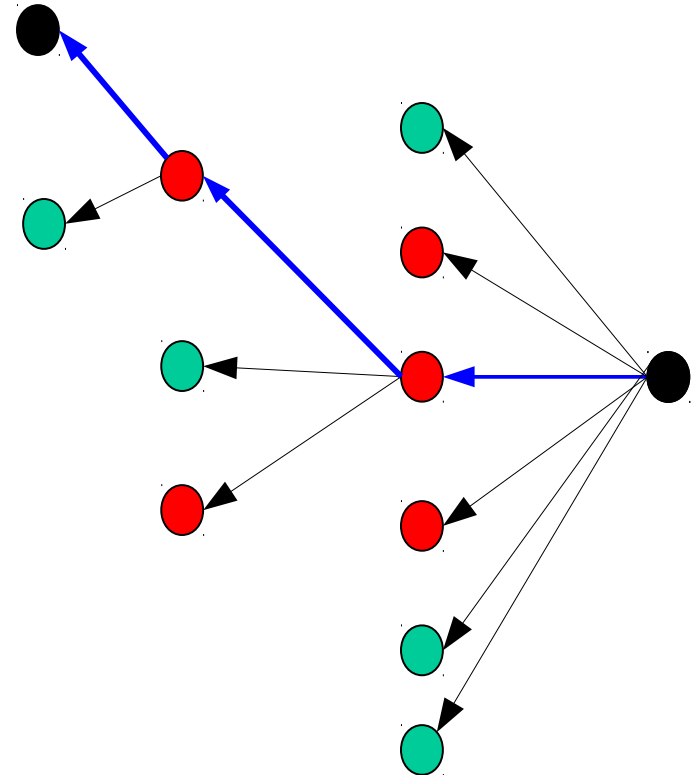
With high probability, truncated mixed graPR explores different path segments each time it is executed between the same pair of solutions.



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

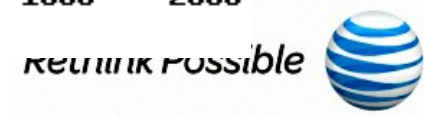
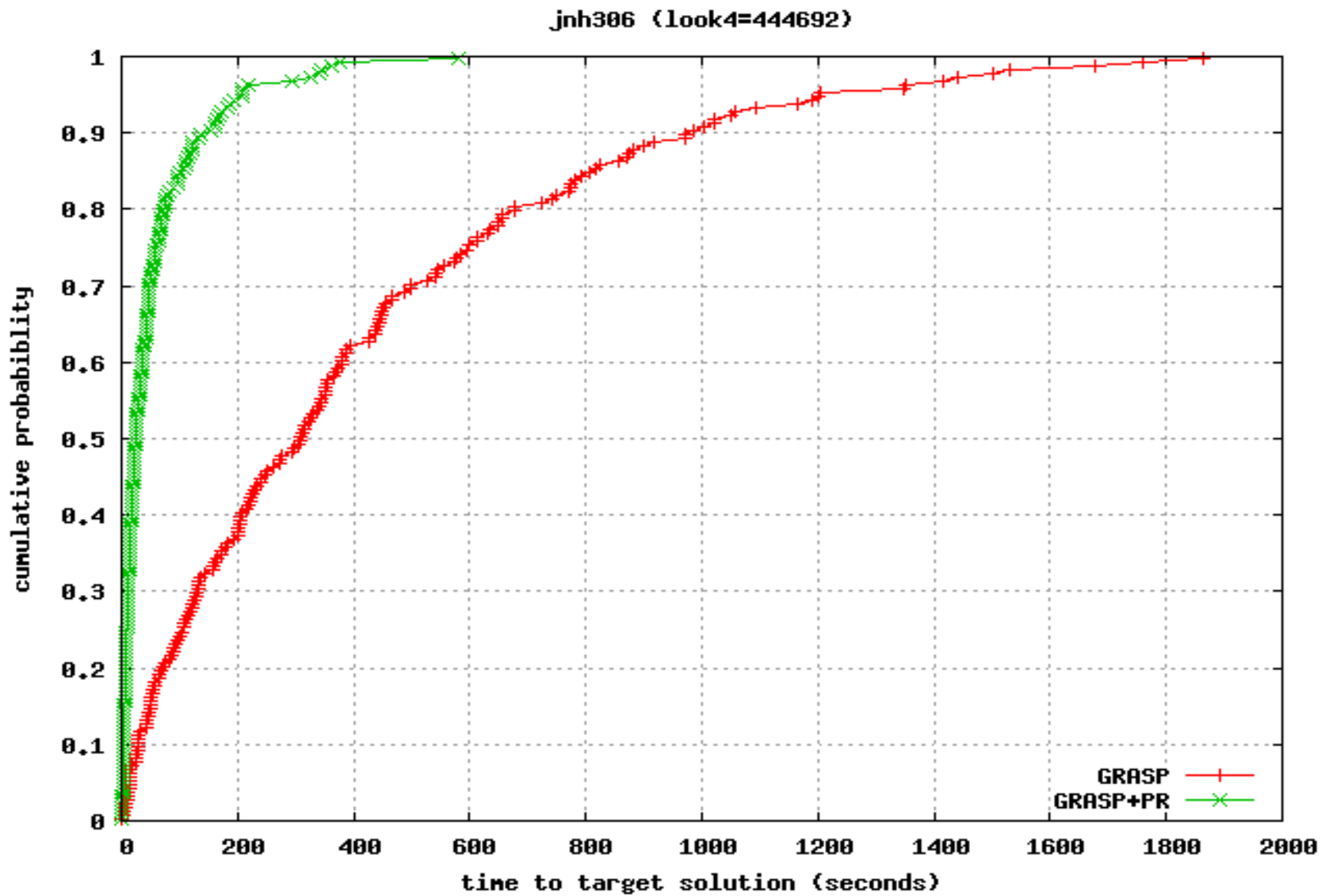
Since 1999, there has been a lot of activity in hybridizing GRASP with path-relinking.

Survey by Resende & Ribeiro in MIC 2003 book of Ibaraki, Nonobe, and Yagiura (2005).

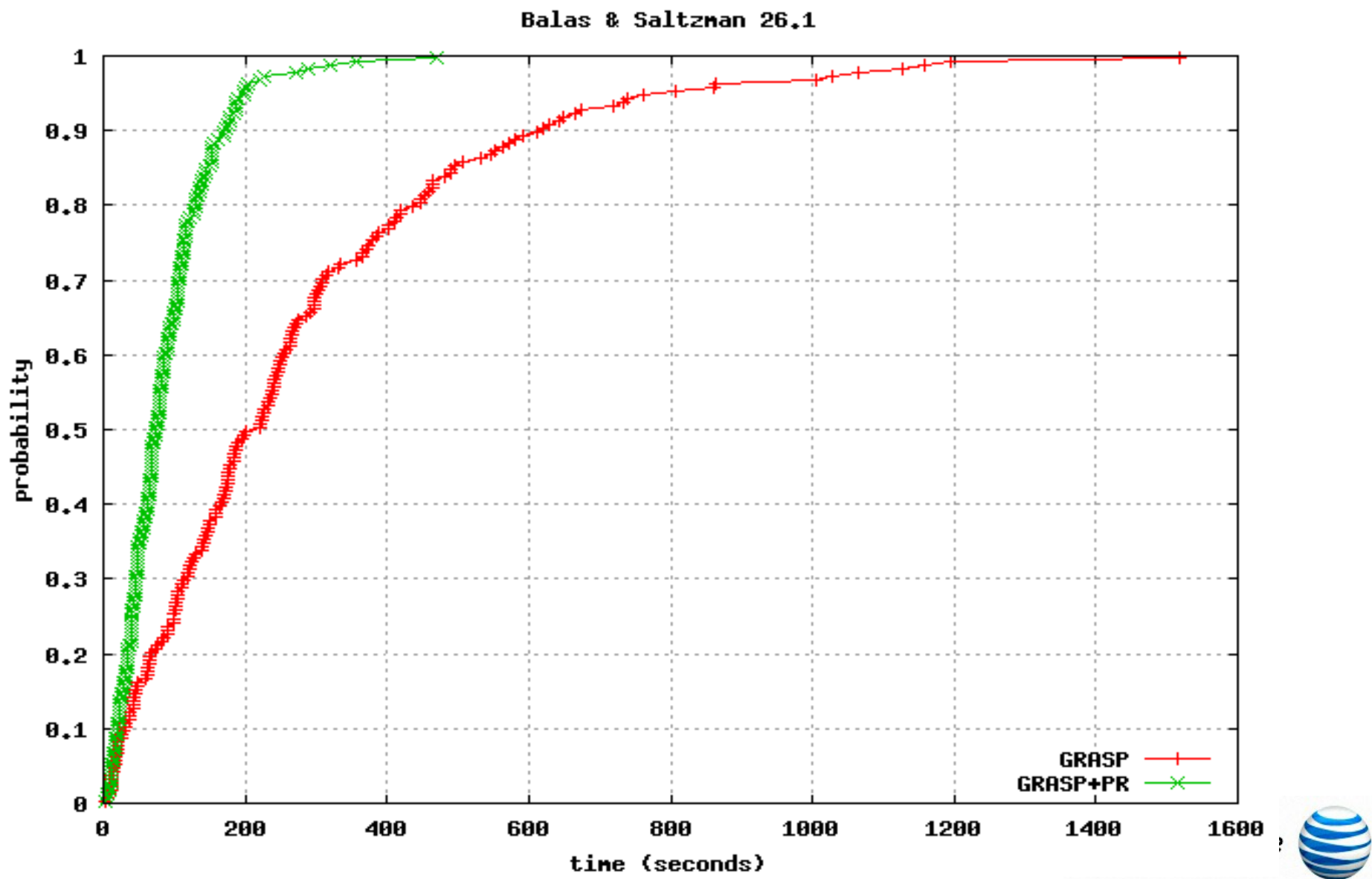
Main observation from experimental studies:
GRASP with path-relinking outperforms pure GRASP.



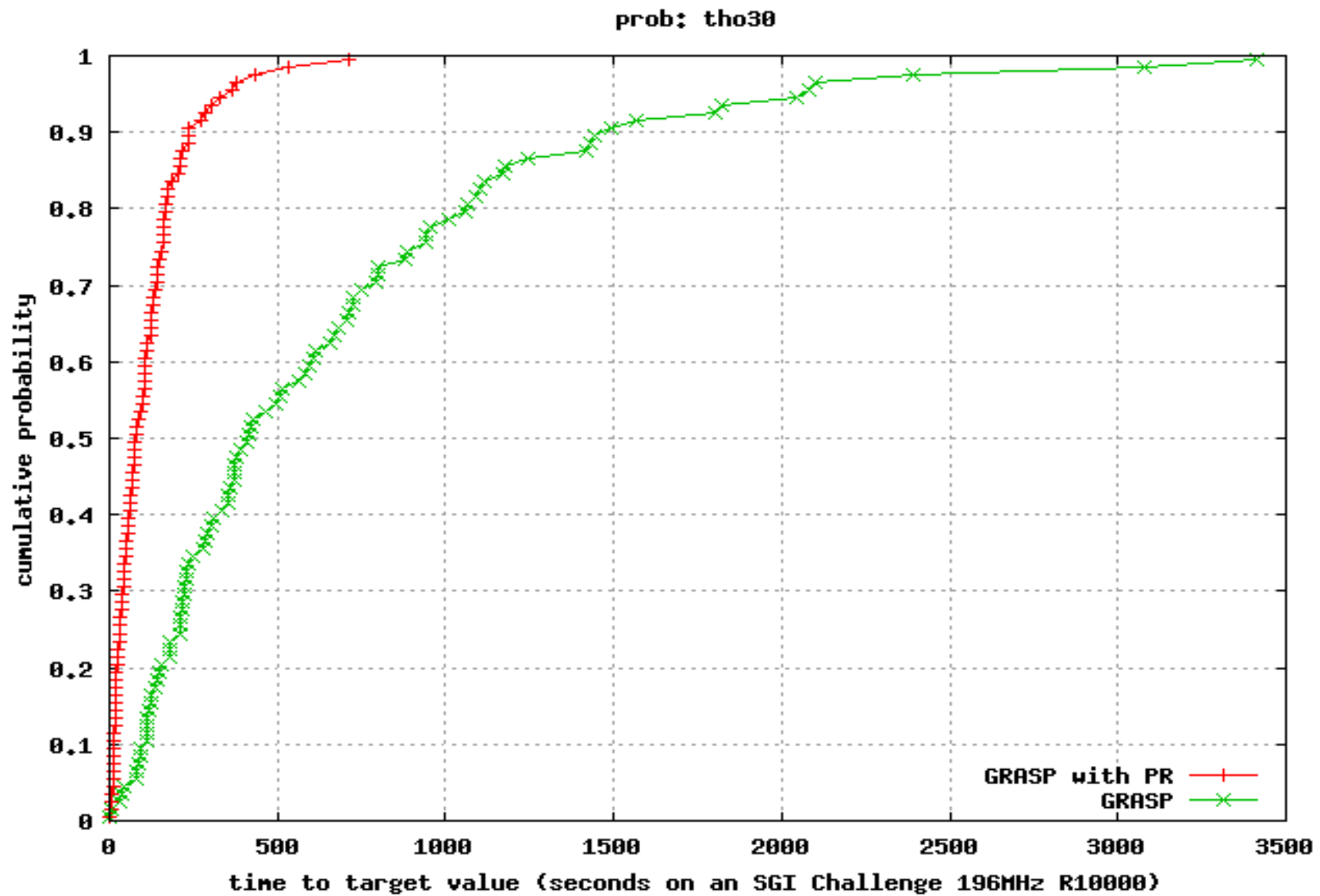
MAX-SAT (Festa, Pardalos, Pitsoulis, and Resende, 2006)



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



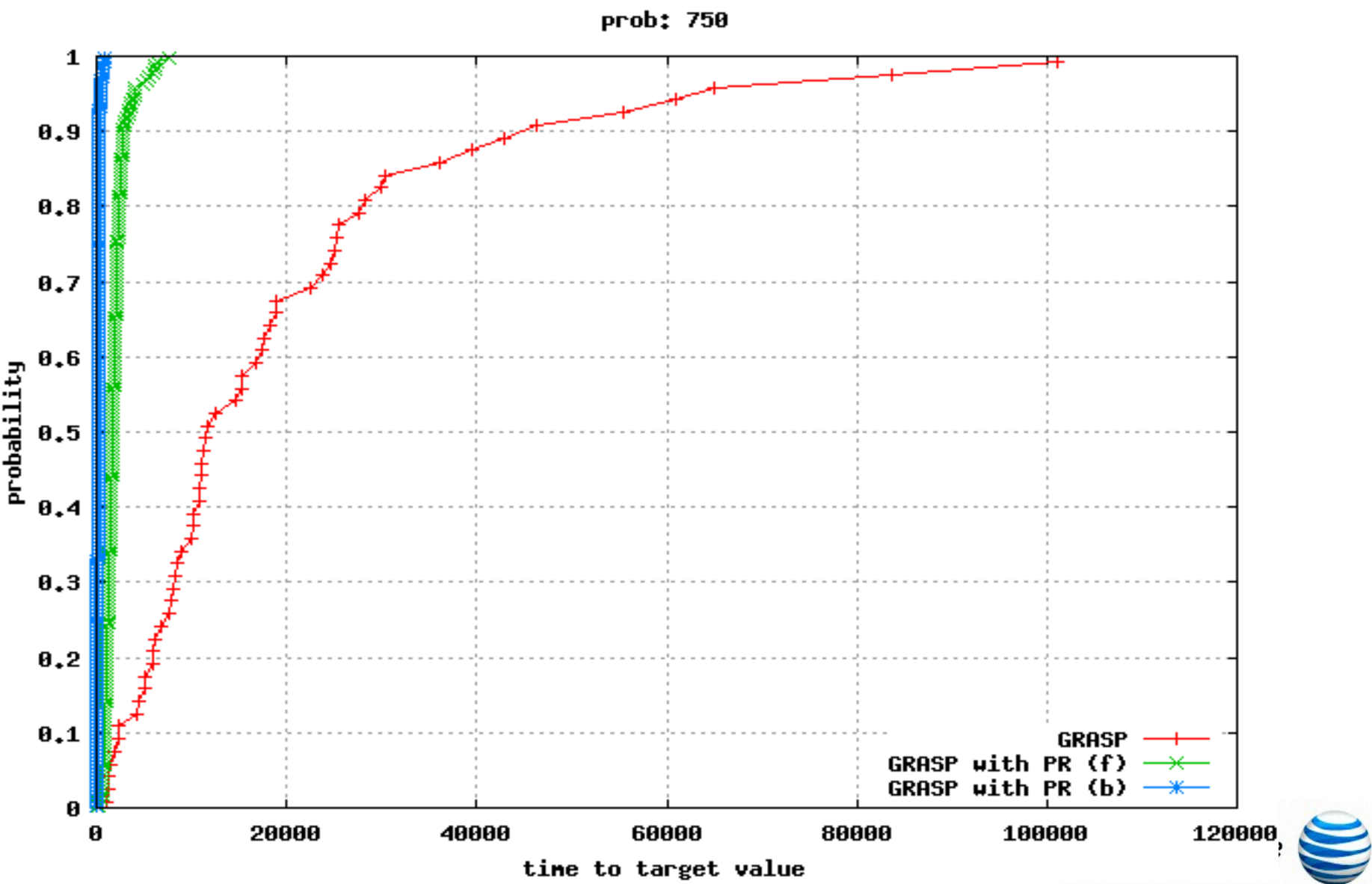
QAP (Oliveira, Pardalos, and Resende, 2004)



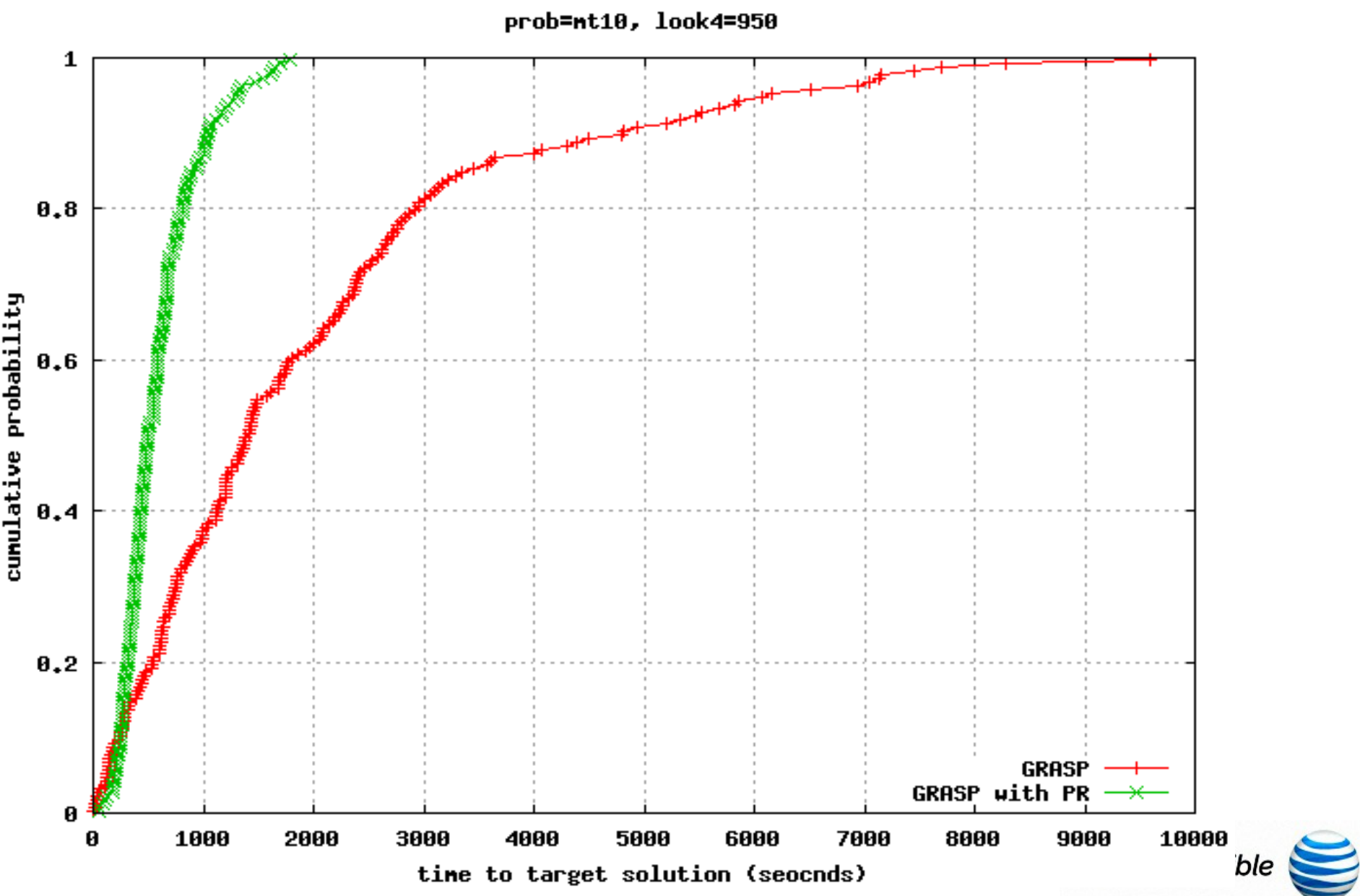
rethink possible



Bandwidth packing (Resende and Ribeiro, 2003)



Job shop scheduling (Aiex, Binato, & Resende, 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

(Resende & 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.

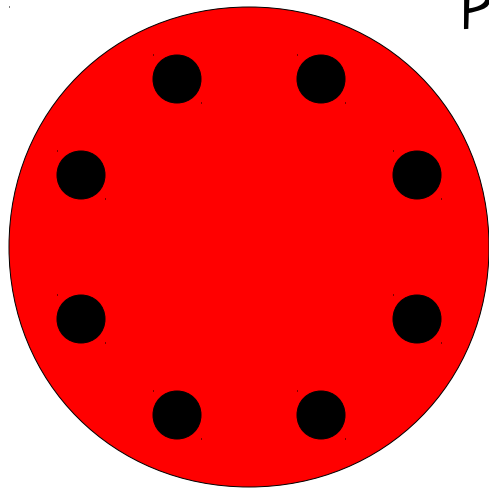
Evolutionary path-relinking can be used

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)



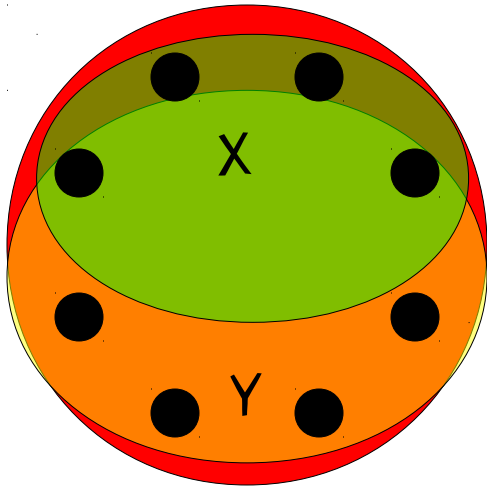
Population $P(0)$

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)

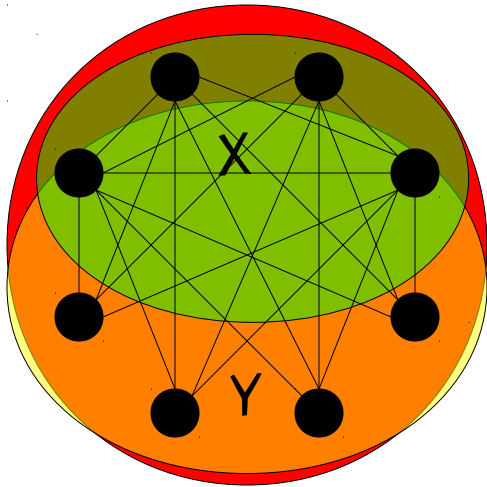


All pairs of elite solutions (x,y) in K -th population $P(K)$, such that $x \in X \subseteq P(K)$ and $y \in Y \subseteq P(K)$, are path-relinked and the resulting $z = PR(x,y)$ is a candidate for inclusion in population $P(K+1)$.

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



Evolutionary path-relinking (EvPR)

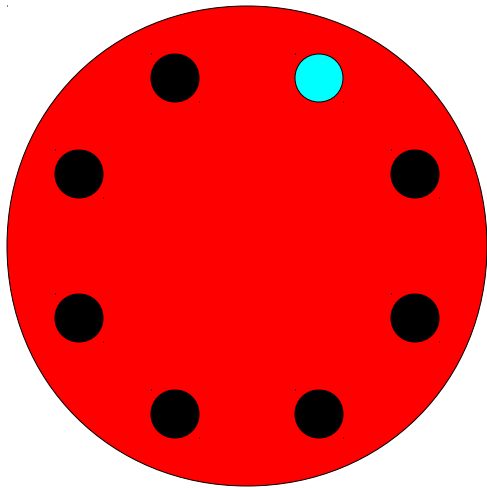


All pairs of elite solutions (x,y) in K -th population $P(K)$, such that $x \in X \subseteq P(K)$ and $y \in Y \subseteq P(K)$, are path-relinked and the resulting $z = PR(x,y)$ is a candidate for inclusion in population $P(K+1)$.

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



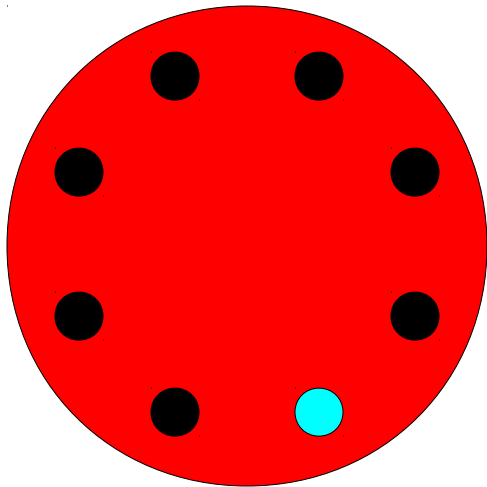
Evolutionary path-relinking (EvPR)



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.

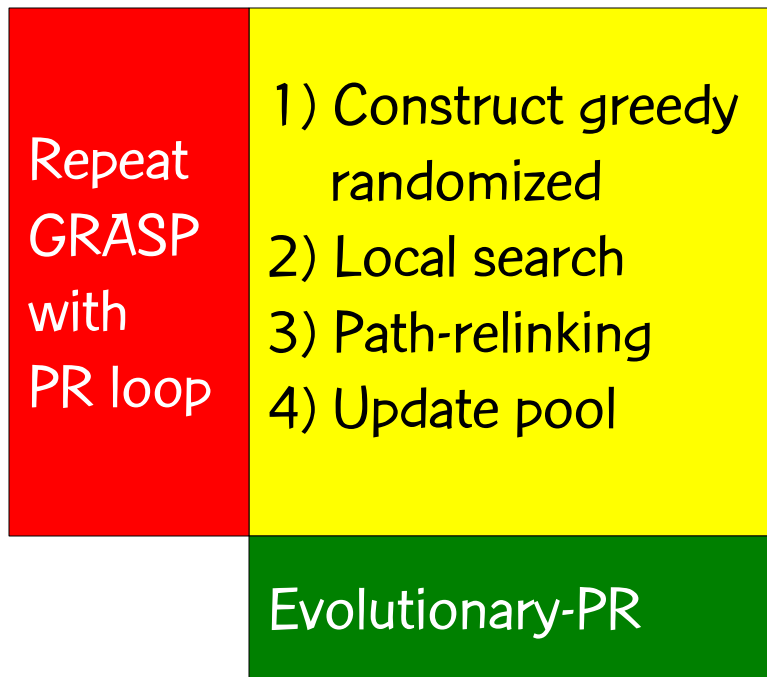


Population $P(K+1)$

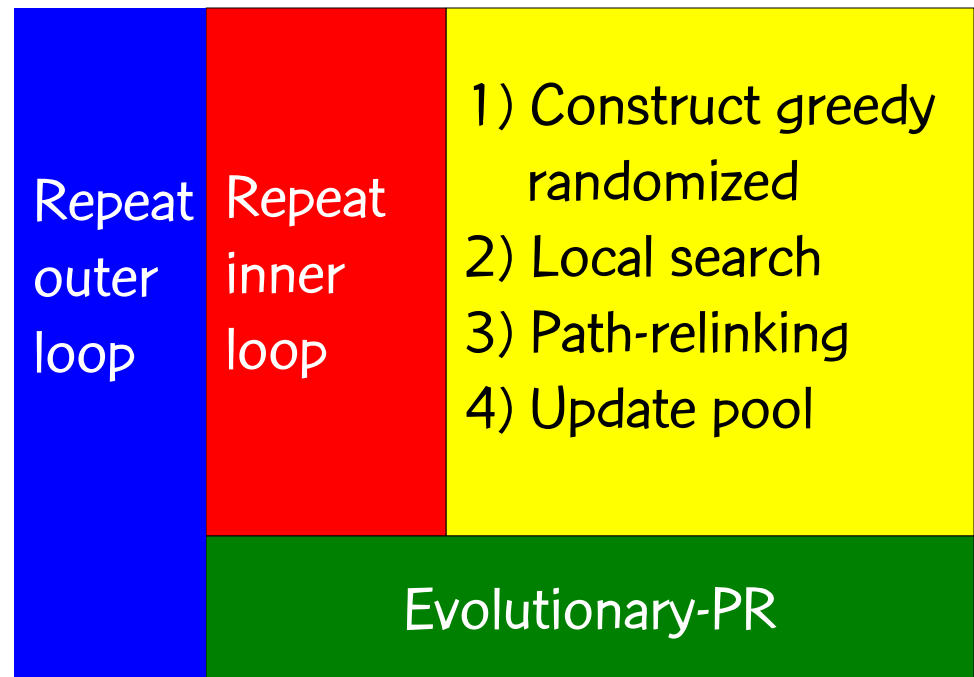


GRASP with evolutionary path-relinking

As post-optimization

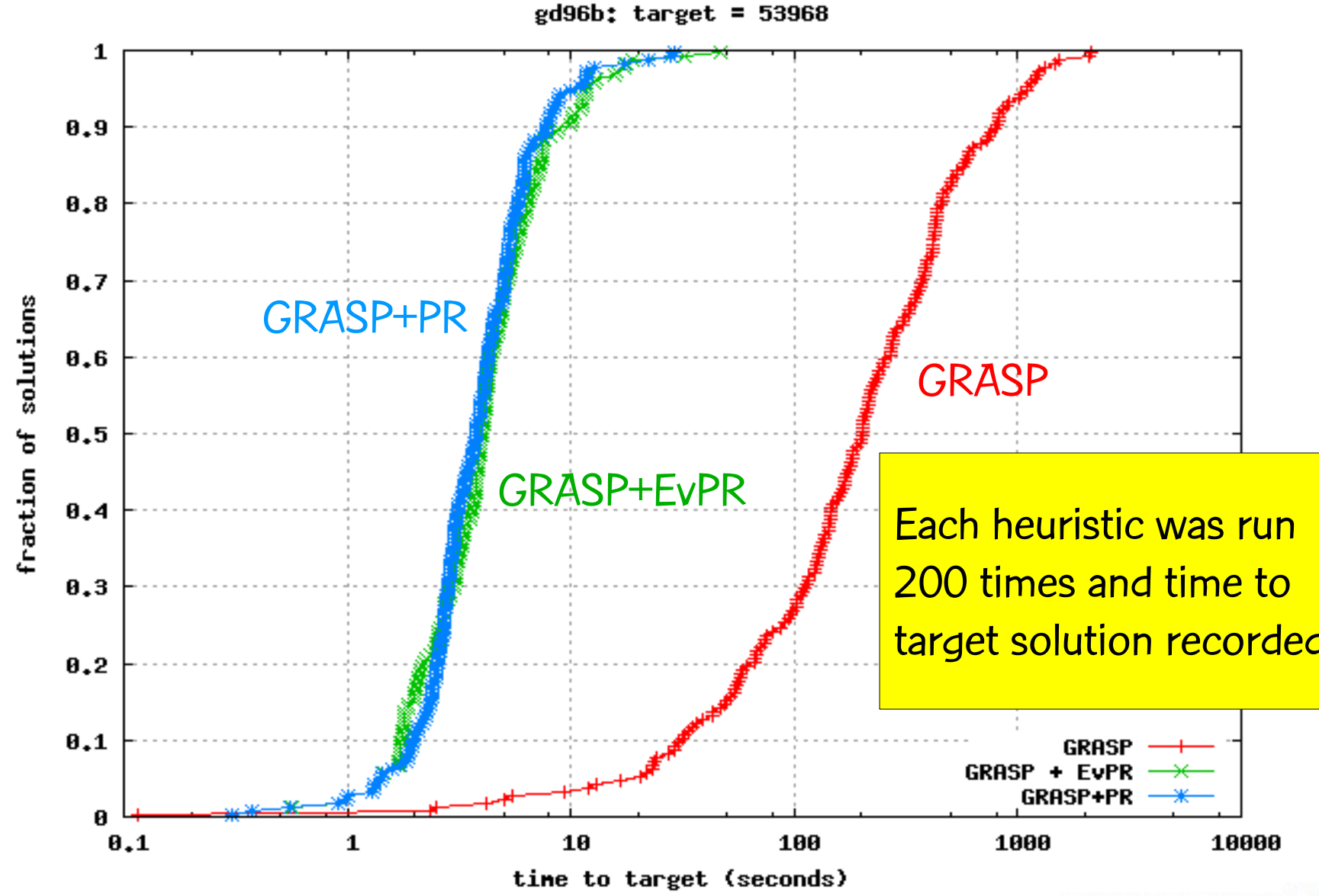


During GRASP + PR

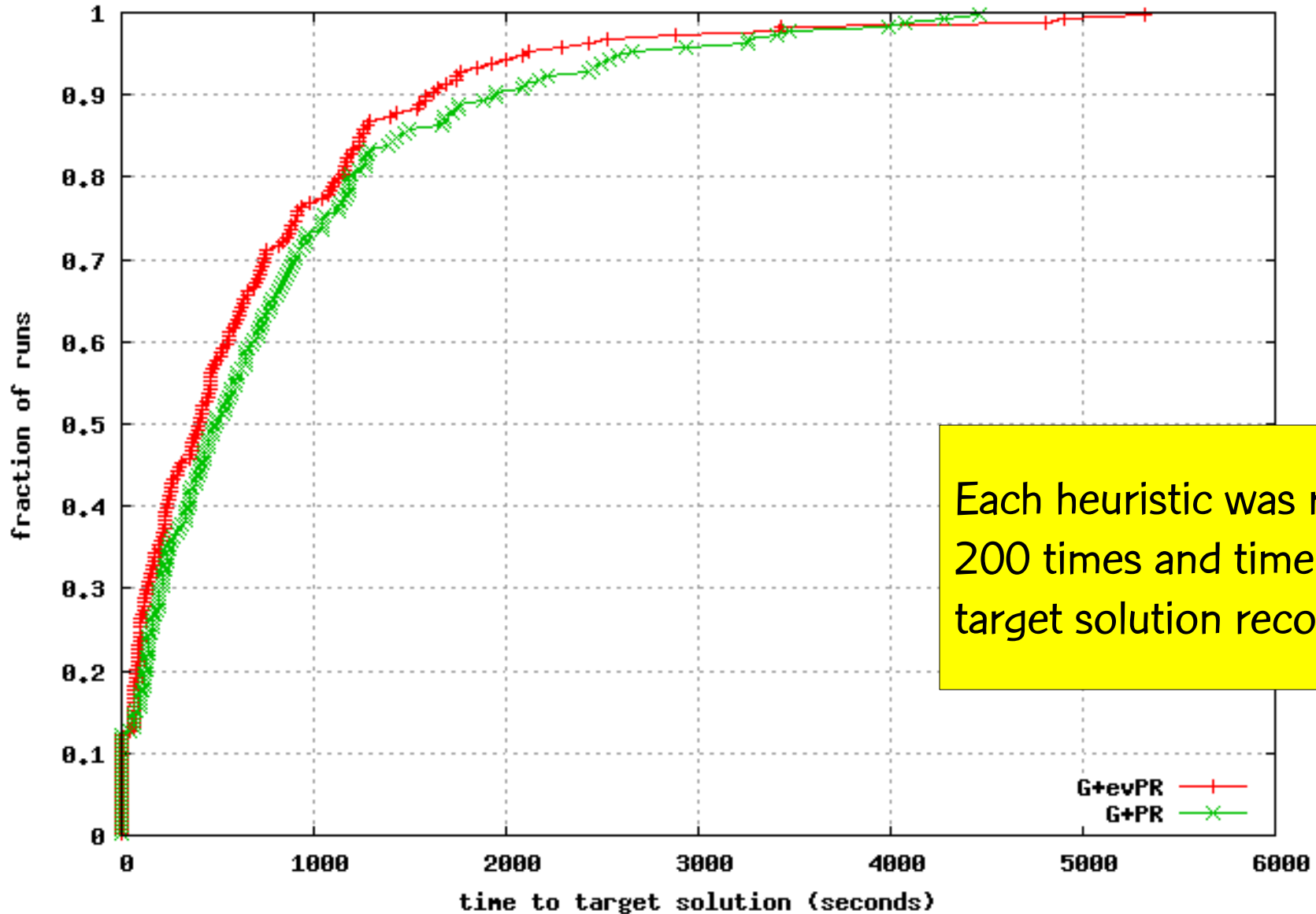


(Resende & Werneck, 2004, 2006)





gd96a minmax lf=1118: G+PR vs G+evPR

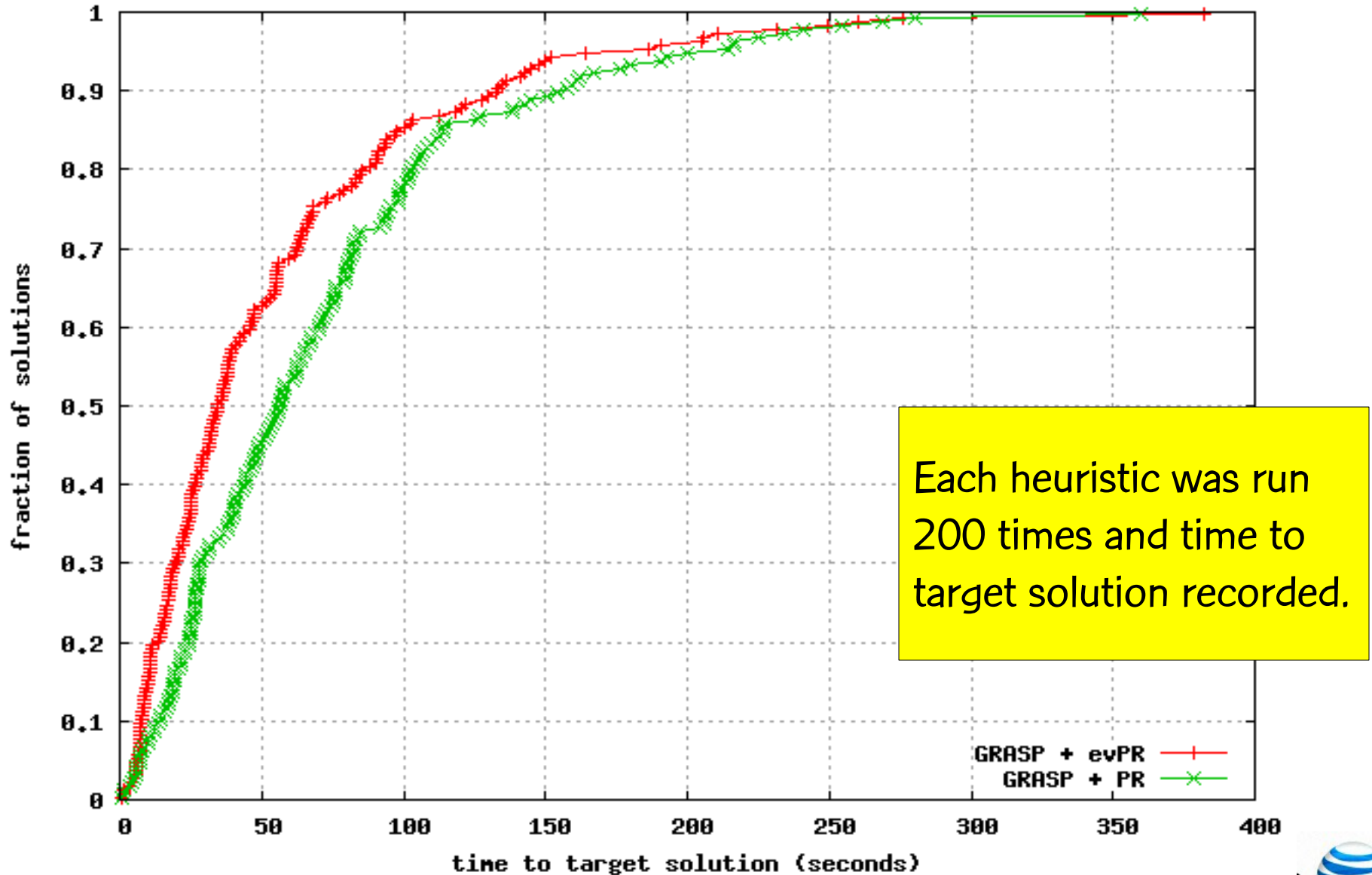


Each heuristic was run 200 times and time to target solution recorded.

rethink possible

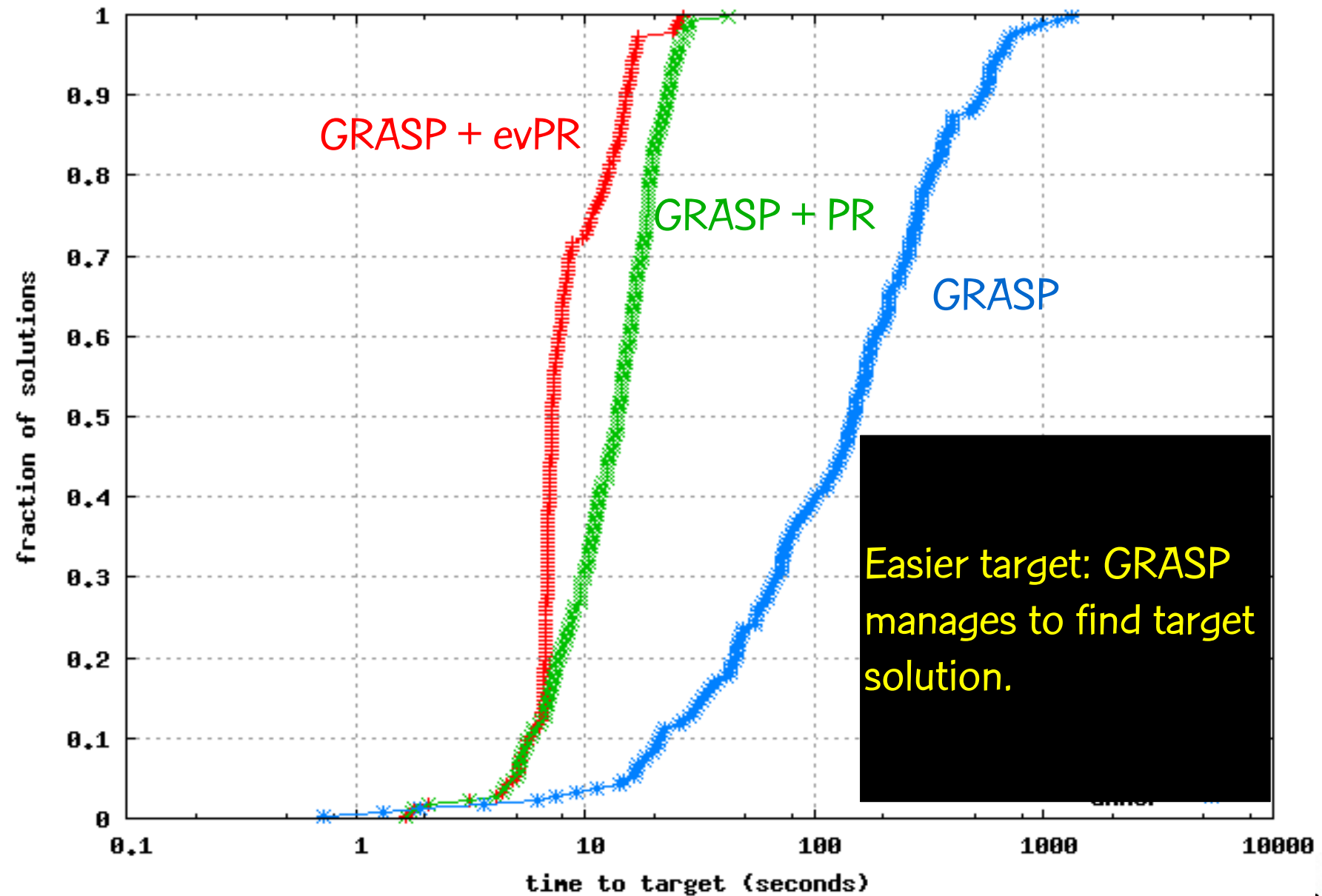


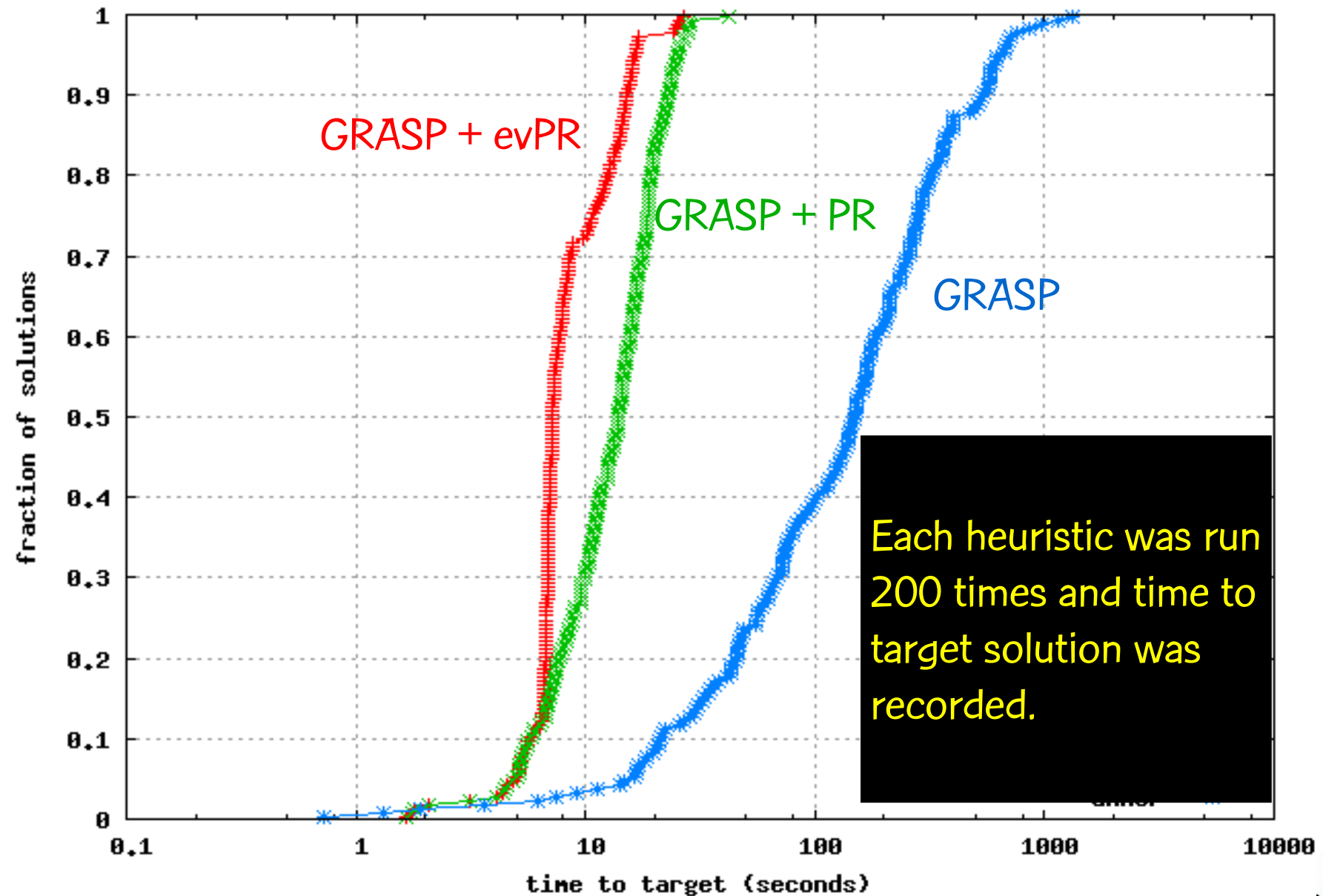
gd96d: look4 = 112 min maxcut

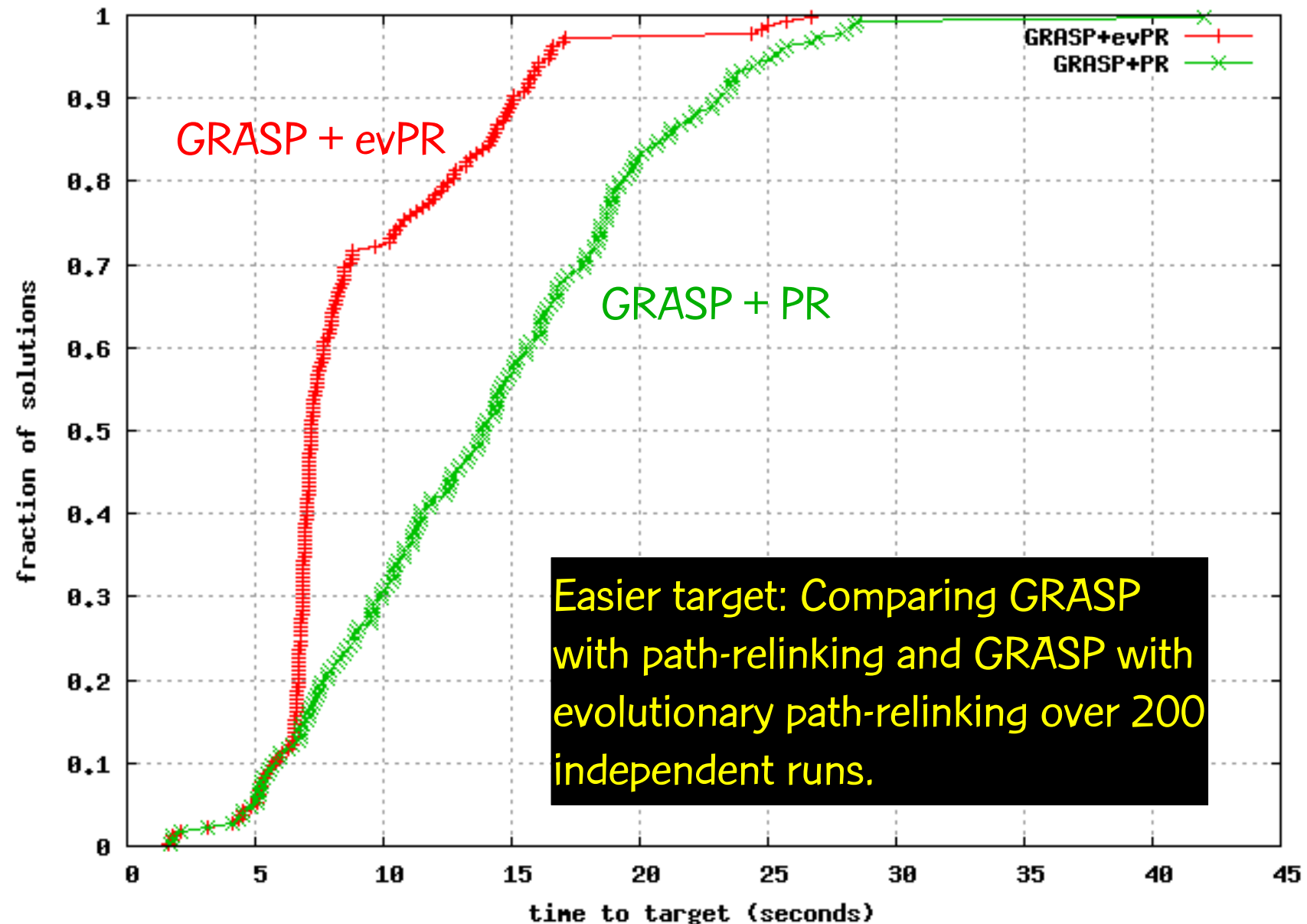


RETHINK POSSIBLE



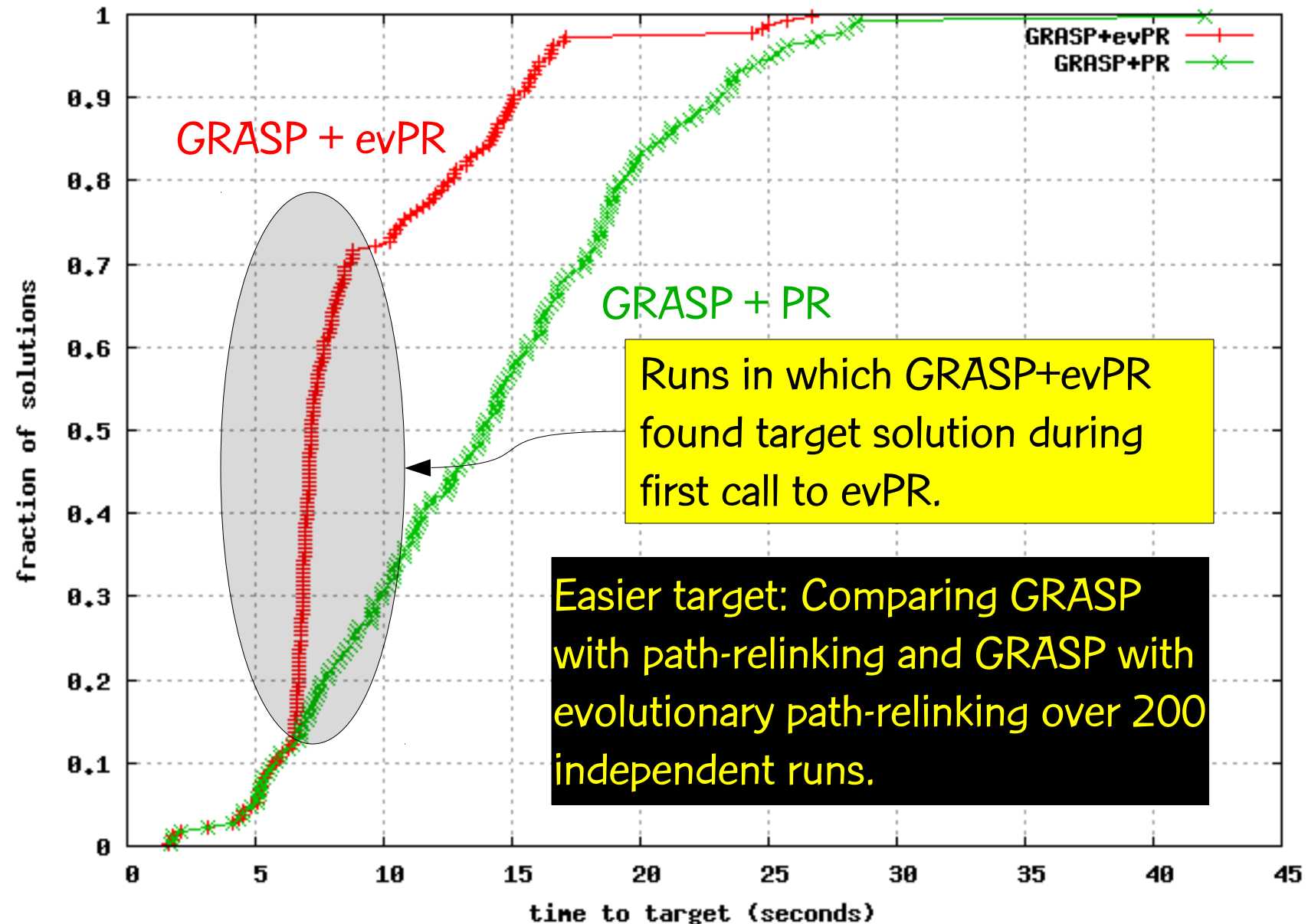


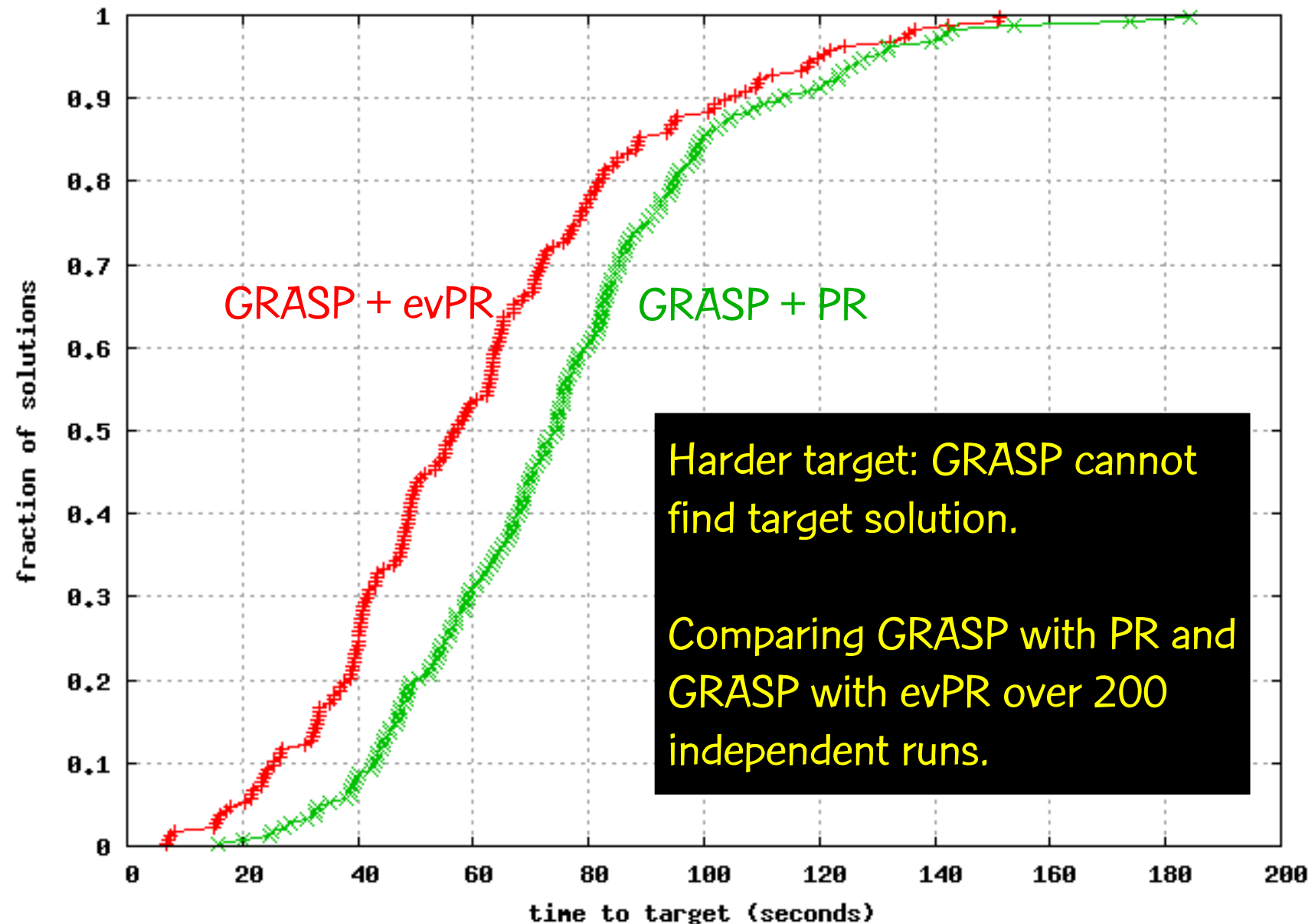




Easier target: Comparing GRASP with path-relinking and GRASP with evolutionary path-relinking over 200 independent runs.







Continuous GRASP (C-GRASP)



C-GRASP

- C-GRASP is a metaheuristic to finding optimal or near-optimal solutions to
 - Min $f(x)$ subject to: $L \leq x \leq U$
 - where $x, L, U \in \mathbb{R}^n$
 - and $f(x)$ can have discontinuities, be non-differentiable, be the output of a simulation, etc.



C-GRASP

- C-GRASP is based on the discrete optimization metaheuristic GRASP
- It was proposed in 2006 by U. of FL ISE PhD students Michael Hirsch and Claudio Meneses with Mauricio Resende and Panos Pardalos.
- M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende, "Global optimization by continuous GRASP," *Optimization Letters*, vol. 1, pp. 201-212, 2007.
- M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende, "Speeding up continuous GRASP," *European J. of Operational Research*, vol. 205, pp. 507-521, 2010.

Rethink Possible



C-GRASP

- C-GRASP is a multi-start procedure, i.e. a major loop is repeated until some stopping criterion is satisfied.
- In each major iteration
 - x is initialized with a solution randomly selected from the box defined by vectors L and U .
 - a number of minor iterations are carried out, where each minor iterations consists of a construction phase and a local improvement phase.
 - Minor iterations are done on a dynamic grid and stops when the grid is too dense.

C-GRASP

$f^* = \infty$

while (stopping criterion not satisfied) **do**

$x = \text{random}[L,U]; h = h(\text{start});$

while ($h \geq h(\text{end})$) **do**

$x = \text{ConstructGreedyRandomized}(x)$

$x = \text{LocalImprovement}(x)$

if ($f(x) < f^*$) **then** { $x^* = x; f^* = f(x)$ }

if (x did not improve this iteration) **then** { $h = h/2$ }

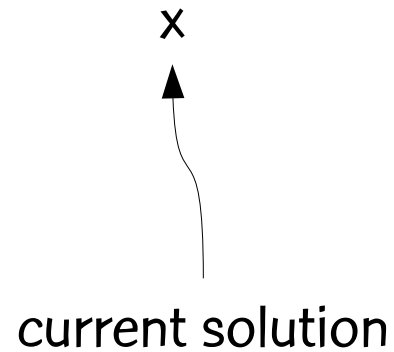
end while

end while

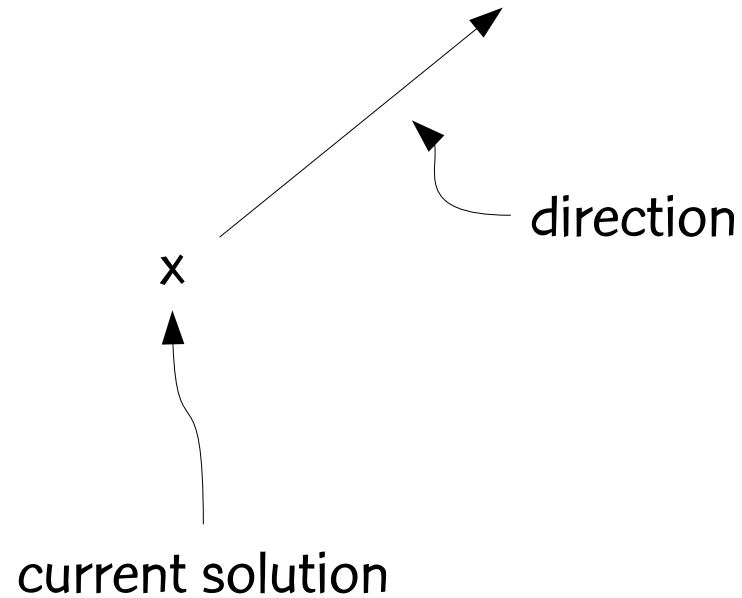
return x^*



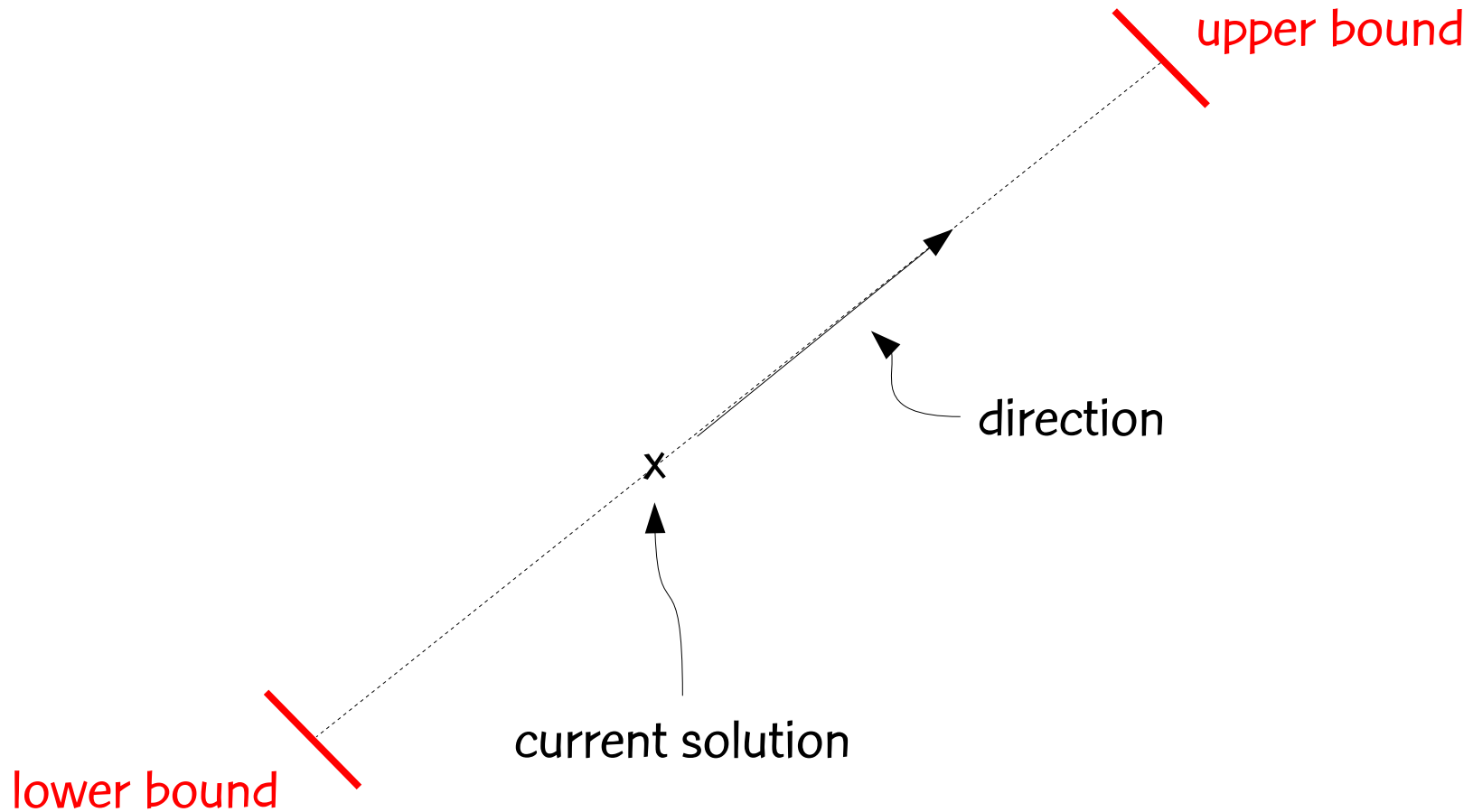
C-GRASP line search



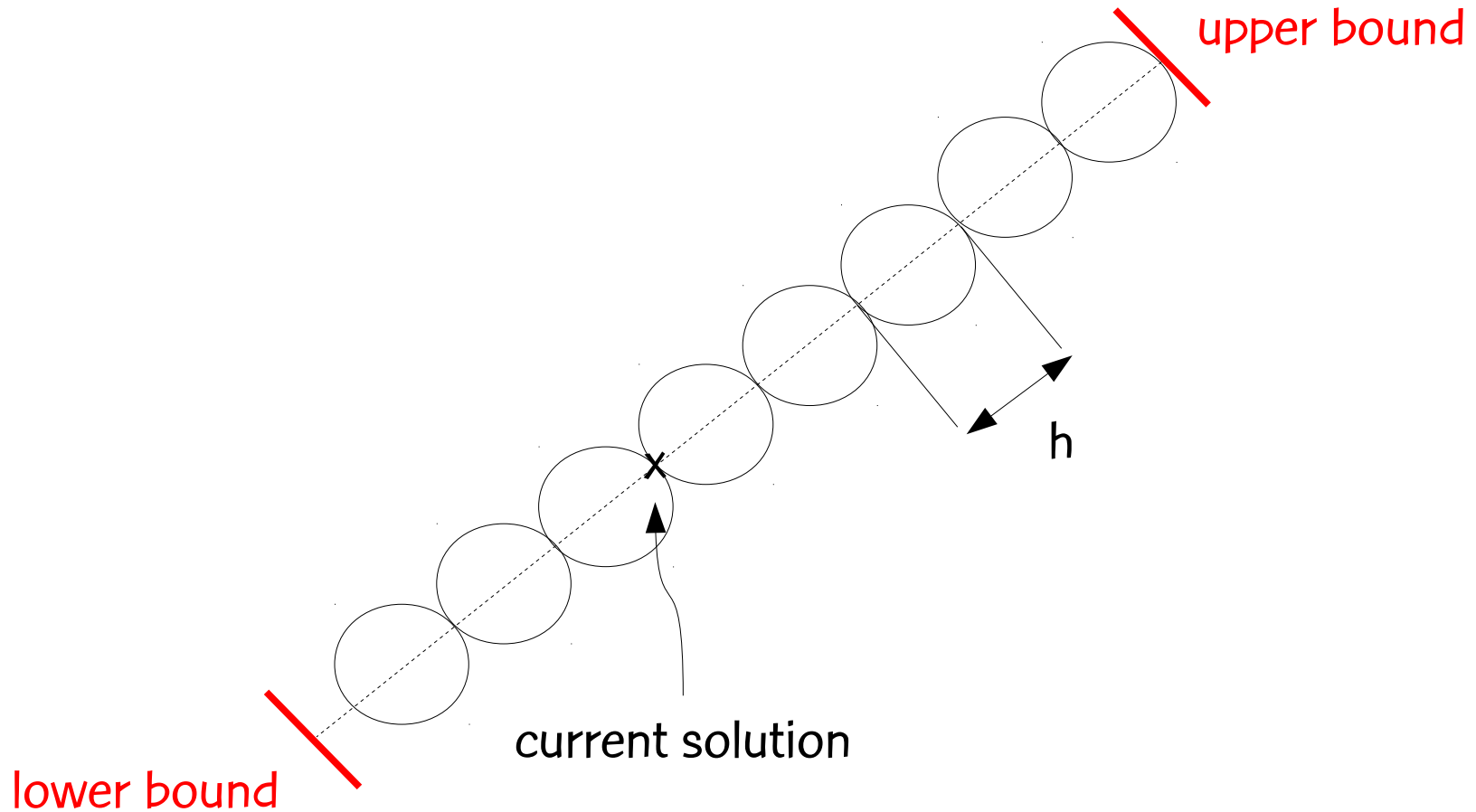
C-GRASP line search



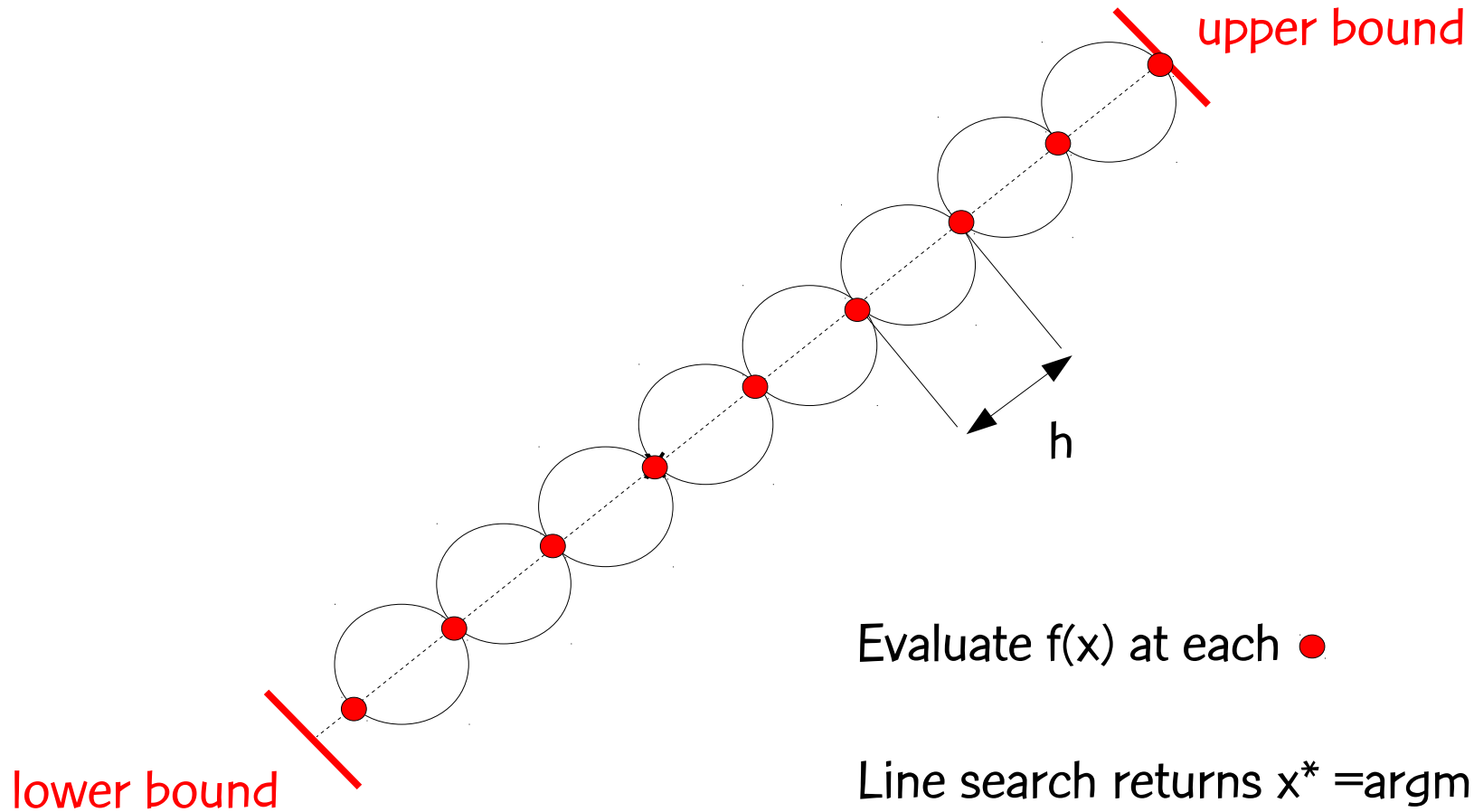
C-GRASP line search



C-GRASP line search



C-GRASP line search



C-GRASP greedy randomized construction

$\text{unset} = \{1, 2, 3, \dots, n\}; x = x^0$

for ($k = 1, 2, \dots, n$) **do**

for (all $i \in \text{unset}$) **do**

$z_i = \text{line search in direction } e_i = (0, 0, \dots, 1, \dots, 0)$

i-th component

end for

$\text{RCL} = \{ i \in \text{unset} \mid f(z_i) < \text{CUTOFF} \}$

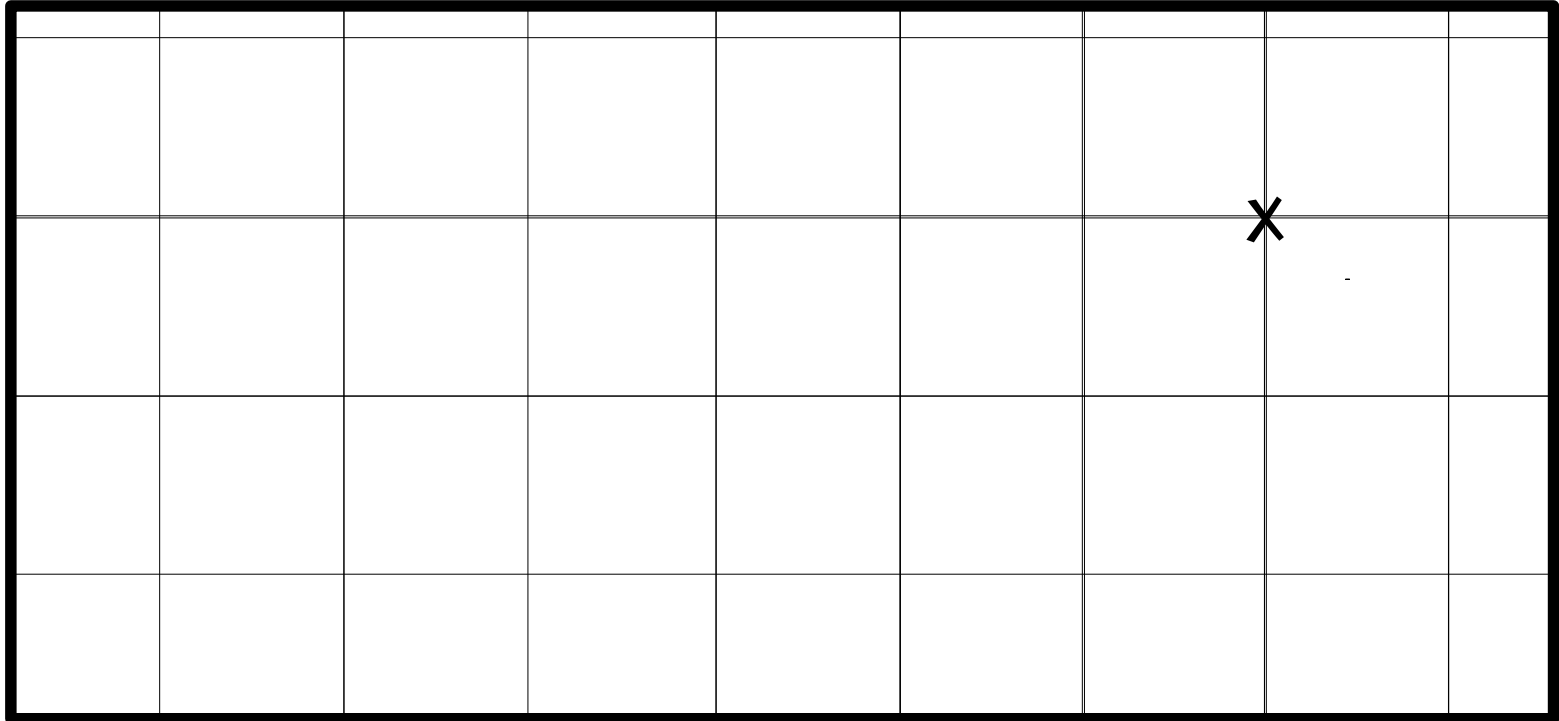
Select at random $i^* \in \text{RCL}$

Set $x_{i^*} = z_{i^*}; \text{unset} = \text{unset} \setminus \{i^*\}$

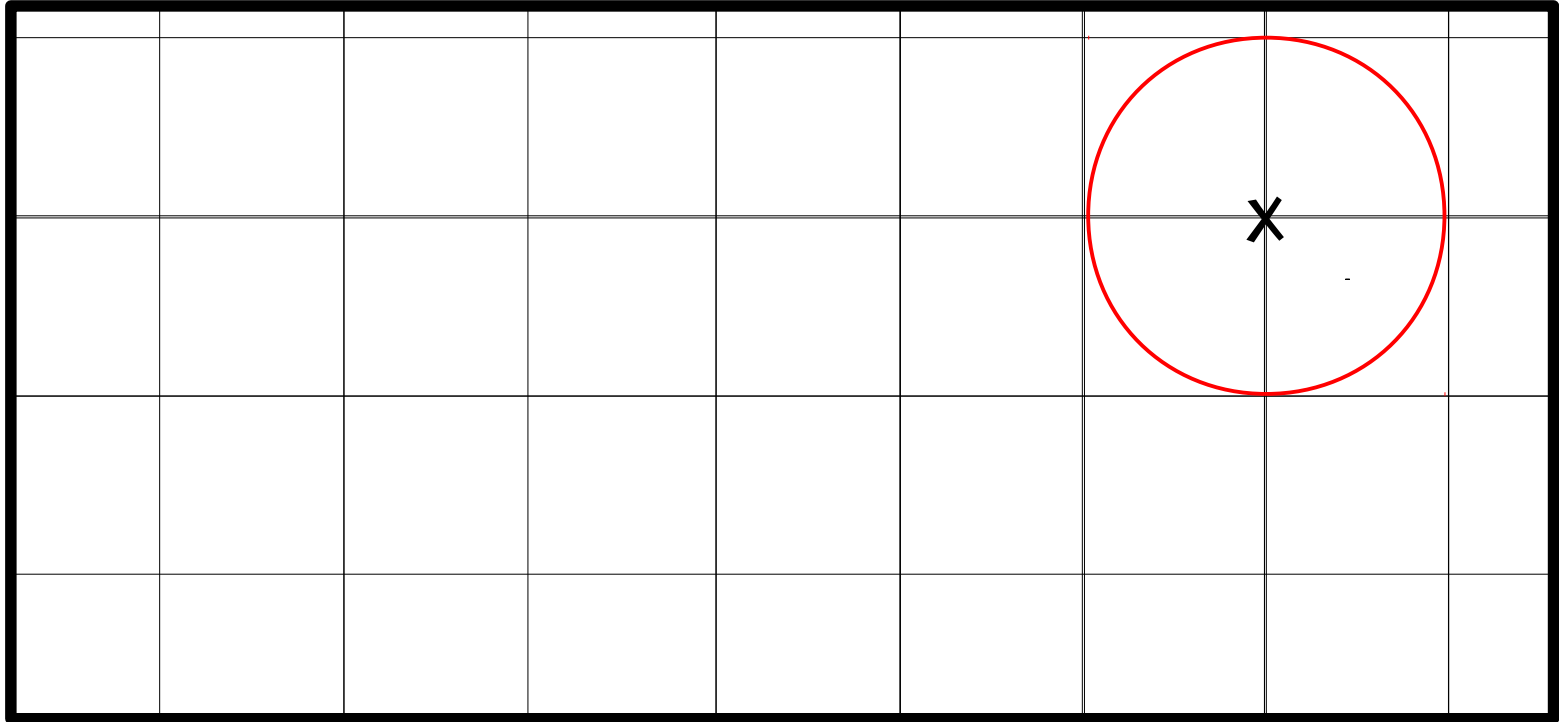
end for



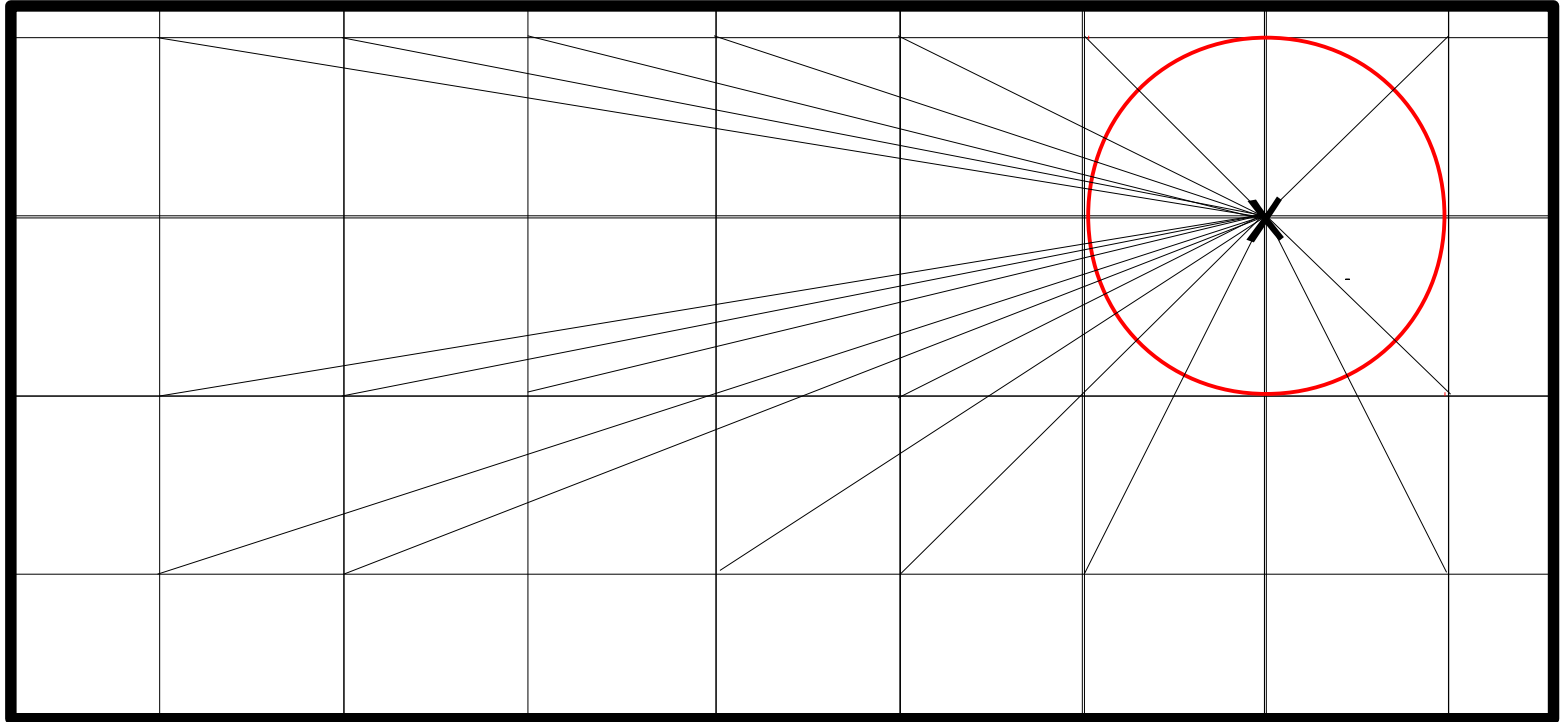
C-GRASP local improvement



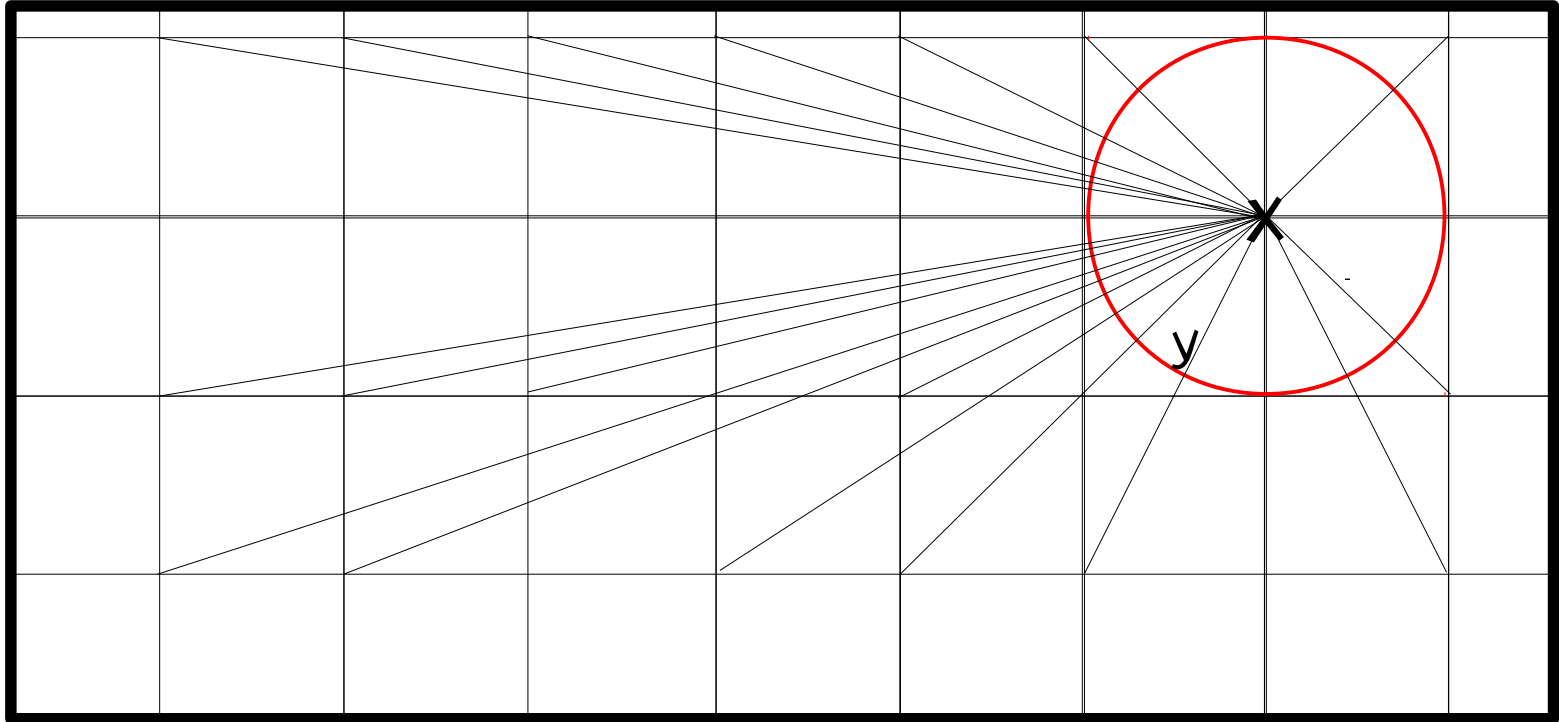
C-GRASP local improvement



C-GRASP local improvement



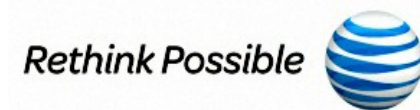
C-GRASP local improvement



Sample projected point y on circle and evaluate $f(y)$

If $f(y) < f(x)$ then set $x = y$, translate grid to intersect x
and restart local search from x

If max-points are examined without improvement: x is h-local min



C-GRASP

- M.J. Hirsch, “GRASP-based heuristics for continuous global optimization problems,” Dept. of Industrial & Systems Engineering, University of Florida, Gainesville, Florida, 2006.
 - Michael Hirsch's Ph.D. thesis.



C-GRASP

- M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende, "Sensor registration in a sensor network by continuous GRASP," IEEE Military Communications Conference (MILCOM), 2006.
 - Sensor registration is the process of removing (accounting for) non-random errors, or biases, in sensor data.
 - We solve the sensor registration problem when some data is not seen by all sensors, and the correspondence of data seen by the different sensors is not known.
 - We outperform previous methods in the literature and have been issued a U.S. Patent.



C-GRASP

- M.J. Hirsch, C.N. Meneses, P.M. Pardalos, M.A. Ragle, and M.G.C. Resende, "A continuous GRASP to determine the relationship between drugs and adverse reactions," in "Data Mining, Systems Analysis and Optimization in Biomedicine," O. Seref, O.Erhun Kundakcioglu, and P.M. Pardalos (eds.), AIP Conference Proceedings, vol. 953, pp. 106-121, Springer, 2008.
 - We formulate the drug-reaction relationship problem as a continuous global optimization problem



C-GRASP

- M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende, "Solving systems of nonlinear equations with continuous GRASP," *Nonlinear Analysis: Real World Applications*, vol. 10, pp. 2000-2006, 2009.
 - We formulate a system of nonlinear equations as nonlinear function which has min value zero. After finding a root, we add a barrier around the root and resolve to find the next root.



C-GRASP

- E.G. Birgin, E.M. Gozzi, M.G.C. Resende, and R.M.A. Silva, “Continuous GRASP with a local active-set method for bound-constrained global optimization,” *J. of Global Optimization*, vol. 48, pp. 289-310, 2010.
 - We adapt C-GRASP for global optimization of functions for which gradients can be computed. To to this, we use GENCAN (Birgin and Martínez, 2002), an active-set method for bound-constrained local minimization as the local improvement procedure.



C-GRASP

- R.M.A. Silva, M.G.C. Resende, and P.M. Pardalos, “A C-GRASP Python/C library for bound-constrained global optimization,” *Optimization Letters*, vol. 7, pp. 967-984, 2013.
 - We describe **libcgrpp**, a GNU-style dynamic shared Python/C library.
 - The function to be minimized is encoded in Python and read by the library.
 - Solver can be standalone or called from a C program.



C-GRASP

- M.J. Hirsch, P.M. Pardalos, and M.G.C. Resende, “Correspondence of projected 3D points and lines using a continuous GRASP,” *International Transactions in Operational Research*, vol. 18, pp. 493-511, 2011.
 - Computer vision application



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 describing C-GRASP, an adaptation of GRASP for bound-constrained global optimization.



The End

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

