MIC'2003
Kyoto, August 25-28, 2003

GRASP and Path-Relinking: Advances and Applications

Maurício G.C. RESENDE
AT&T Labs Research
USA

Celso C. RIBEIRO
Catholic University of Rio de Janeiro
Brazil

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

GRASP: Basic algorithm

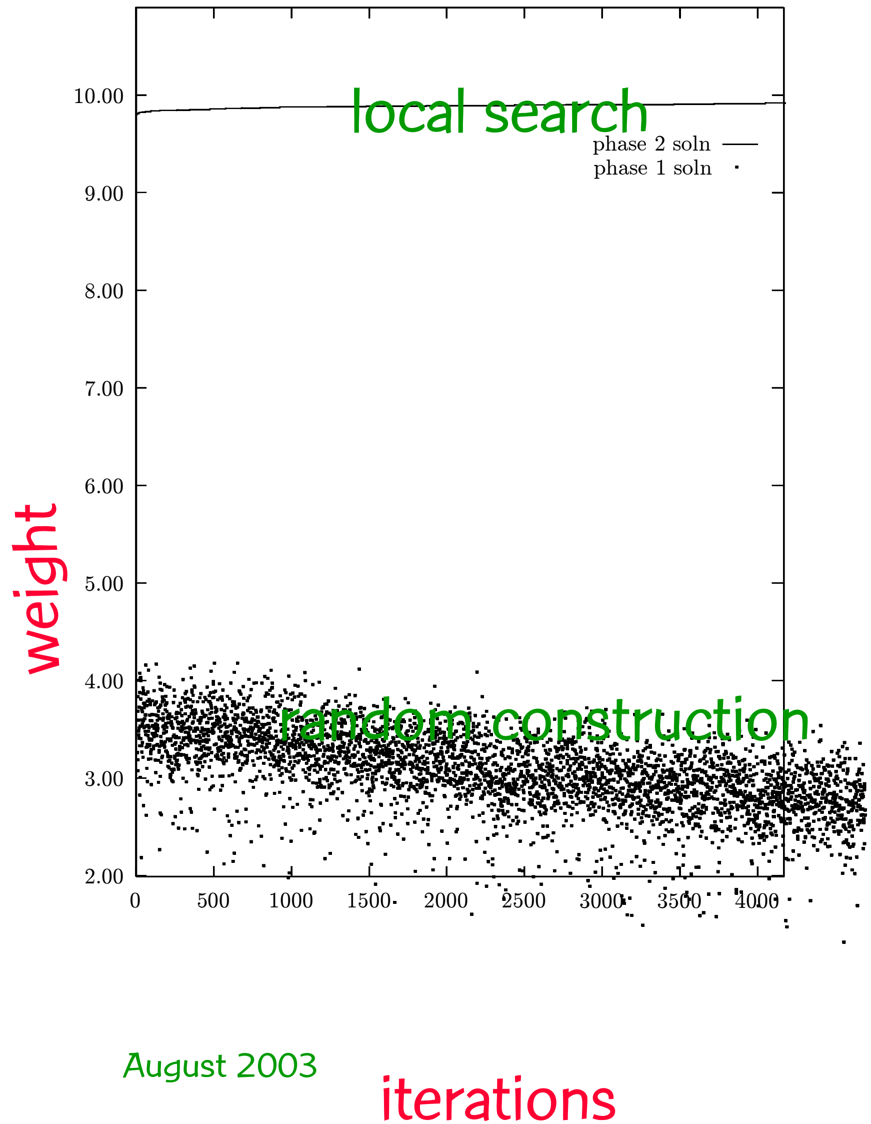
- GRASP:
  - Multistart metaheuristic:
    - Feo & Resende (1989): set covering
    - Festa & Resende (2002): annotated bibliography
    - Resende & Ribeiro (2003): survey

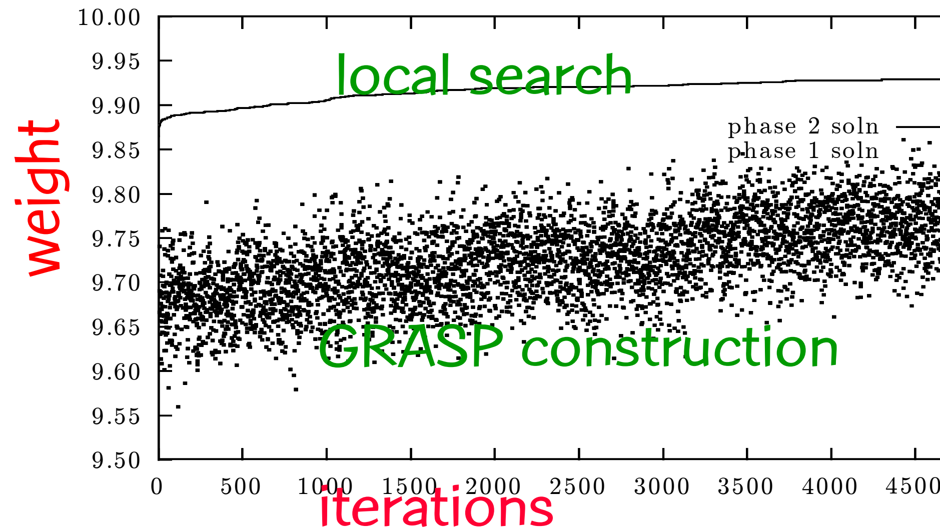
- Repeat for Max_Iterations:
  - Construct a greedy randomized solution.
  - Use local search to improve the constructed solution.
  - Update the best solution found.

GRASP: Basic algorithm

- Construction phase: greediness + randomization
  - Builds a feasible solution:
    - Use greediness to build restricted candidate list and apply randomness to select an element from the list.
    - Use randomness to build restricted candidate list and apply greediness to select an element from the list.
- 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: Basic algorithm





Effectiveness of greedy randomized vs purely randomized construction:

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

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Construction phase

- Greedy Randomized Construction:
  - Solution $\leftarrow \emptyset$
  - Evaluate incremental costs of candidate elements
  - While Solution is not complete do:
    - Build restricted candidate list (RCL).
    - Select an element s from RCL at random.
    - Solution $\leftarrow$ Solution $\cup \{s\}$
    - Reevaluate the incremental costs.
  - endwhile

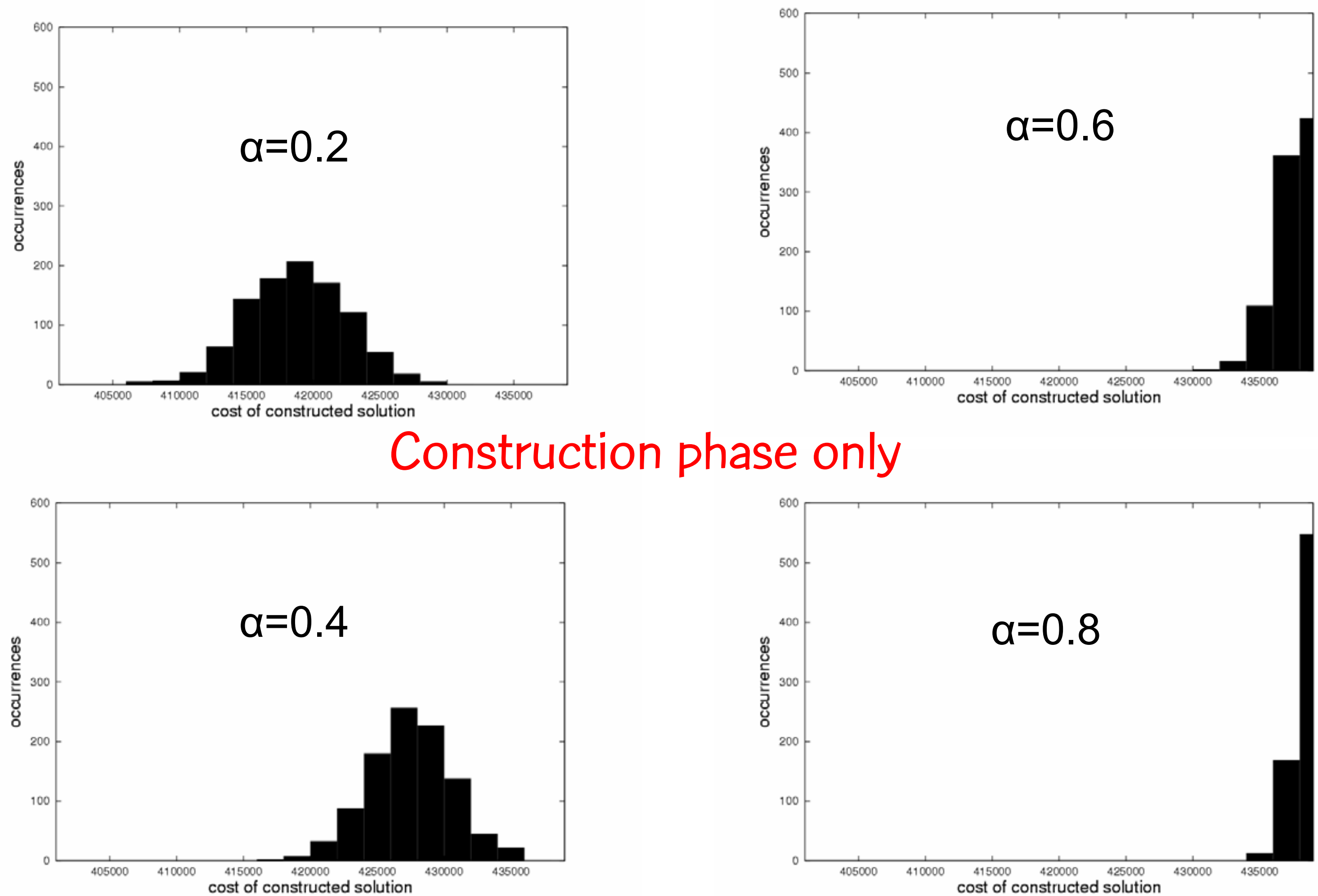
Construction phase

- 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

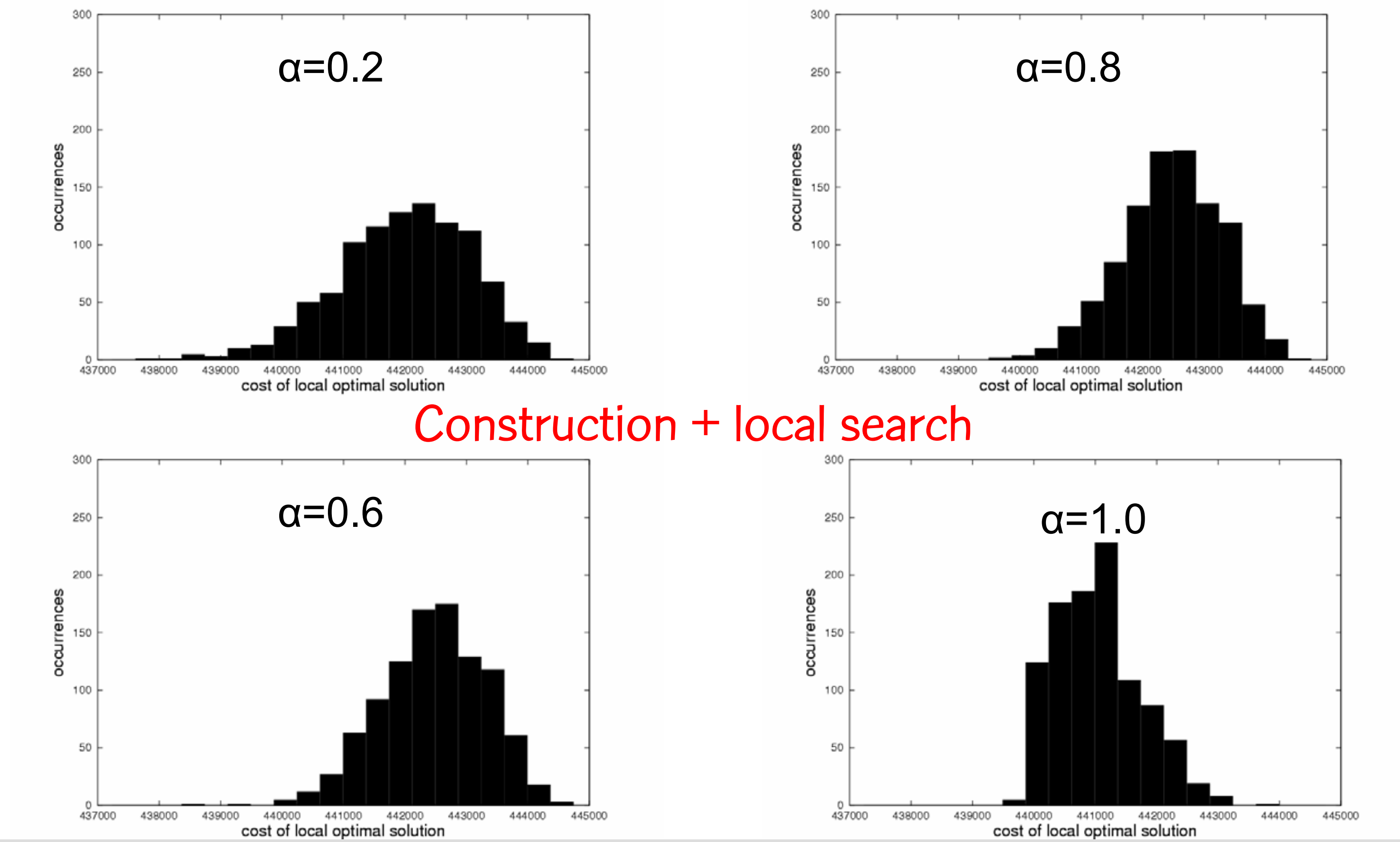
- Cardinality-based construction:
  - p elements with the smallest incremental costs
- Quality-based construction:
  - Parameter α defines the quality of the elements in RCL.
  - RCL contains elements with incremental cost $c^{min} \leq c(e) \leq c^{min} + \alpha\,(c^{max} - c^{min})$
  - $\alpha = 0$: pure greedy construction
  - $\alpha = 1$: pure randomized construction
- Select at random from RCL using uniform probability distribution

Illustrative results: RCL parameter



weighted MAX-SAT instance, 1000 GRASP iterations

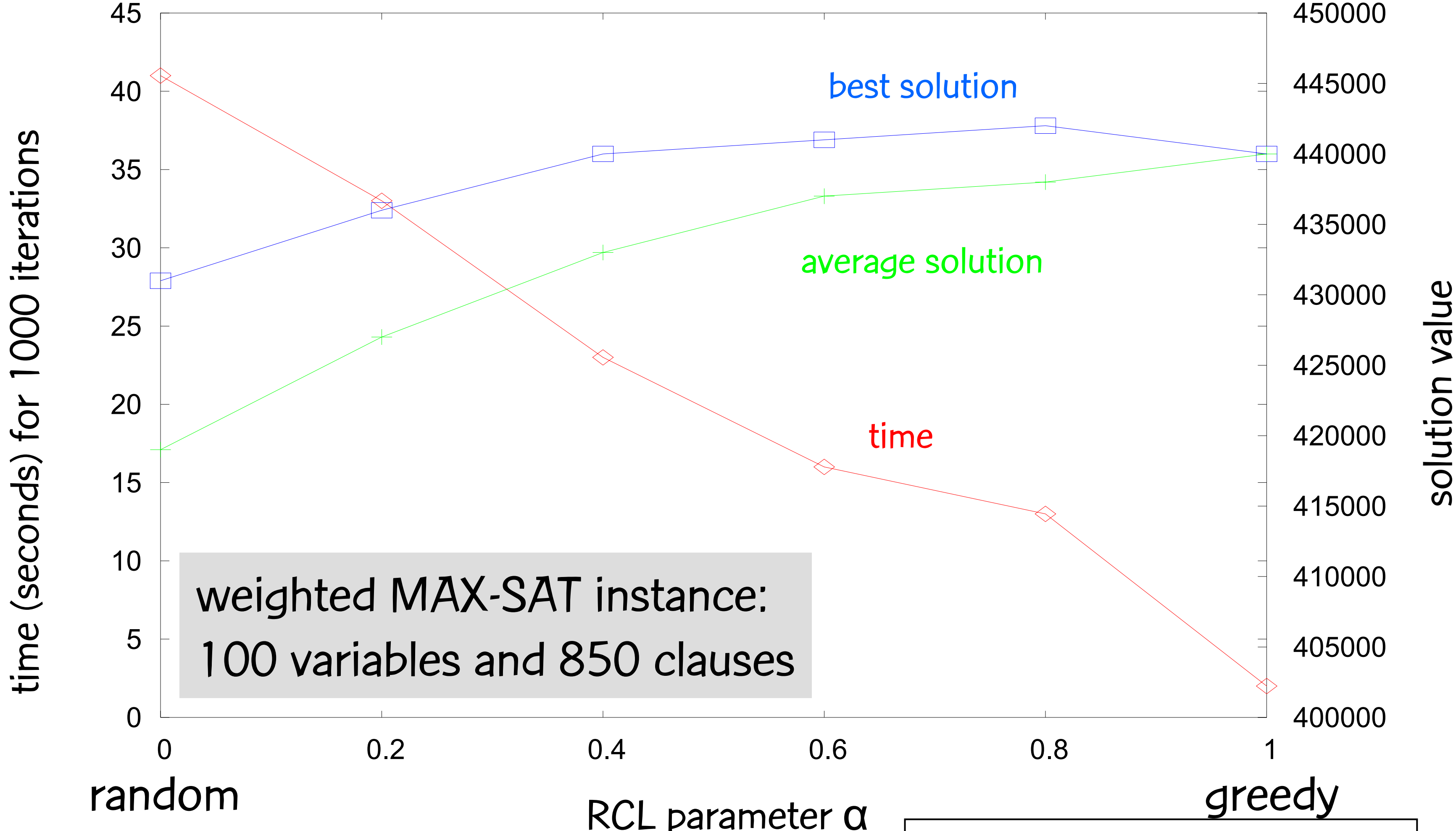
Illustrative results: RCL parameter



5'

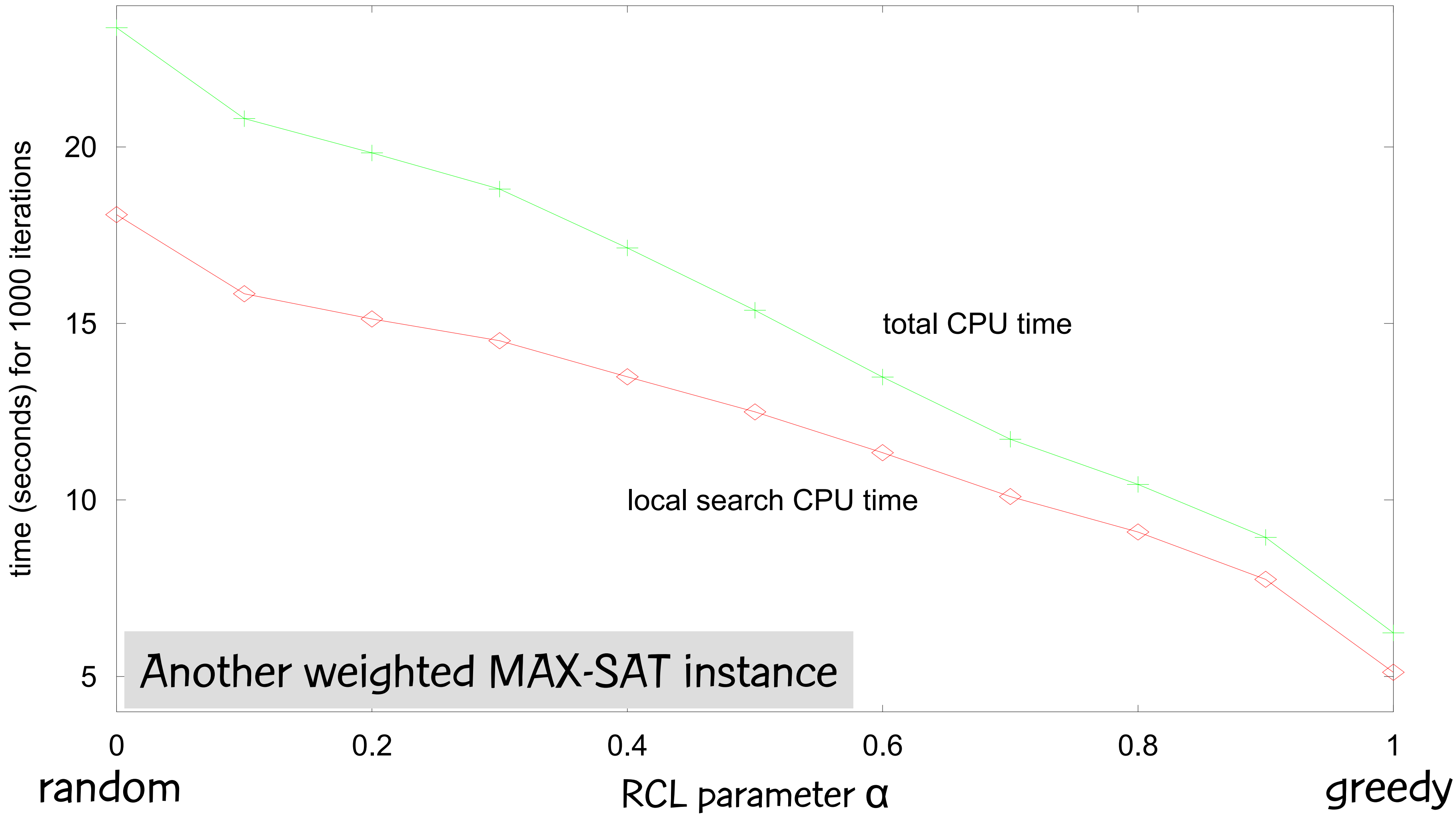
weighted MAX-SAT instance, 1000 GRASP iterations

Illustrative results: RCL parameter



SGI Challenge 196 MHz

Illustrative results: RCL parameter



SGI Challenge 196 MHz

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Enhanced construction strategies

- Reactive GRASP: Prais & Ribeiro (2000) (traffic assignment in TDMA satellites)
  - 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.
  - Other applications, e.g. to graph planarization, set covering, and weighted max-sat:
  - Better solutions, at the cost of slightly larger times.

Enhanced construction strategies

- Cost perturbations: Canuto, Resende, & Ribeiro (2001) (prize-collecting Steiner tree)
  - Randomly perturb original costs and apply some heuristic.
  - Adds flexibility to algorithm design:
    - May be more effective than greedy randomized construction in circumstances where the construction algorithm is not very sensitive to randomization.
    - Also useful when no greedy algorithm is available.

Enhanced construction strategies

- Sampled greedy: Resende & Werneck (2002) (p-median)
  - Randomly samples a small subset of candidate elements and selects element with smallest incremental cost.
- Random+greedy:
  - Randomly builds first part of the solution and completes the rest using pure greedy construction.

Enhanced construction strategies

- Memory and learning in construction: Fleurent & Glover (1999) (quadratic assignment)
  - Uses long-term memory (pool of elite solutions) to favor elements which frequently appear in the elite solutions (consistent and strongly determined variables).

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Local search

- First improving vs. best improving:
  - First improving is usually faster.
  - Premature convergence to low quality local optimum is more likely to occur with best improving.
- VND to speedup search and to overcome optimality w.r.t. to simple (first) neighborhood: Ribeiro, Uchoa, & Werneck (2002) (Steiner problem in graphs)
- Hashing to avoid cycling or repeated application of local search to same solution built in the construction phase: Woodruff & Zemel (1993), Ribeiro et. al (1997) (query optimization), Martins et al. (2000) (Steiner problem in graphs)

10'

Local search

- Filtering to avoid application of local search to low quality solutions, only promising unvisited solutions are investigated: Feo, Resende, & Smith (1994), Prais & Ribeiro (2000) (traffic assignment), Martins et. al (2000) (Steiner problem in graphs)
- Extended quick-tabu local search to overcome premature convergence: Souza, Duhamel, & Ribeiro (2003) (capacitated minimum spanning tree, better solutions for largest benchmark problems)
- Complementarity GRASP-VNS:
  - Randomization at different levels: construction in GRASP vs. local search in VNS

Summary

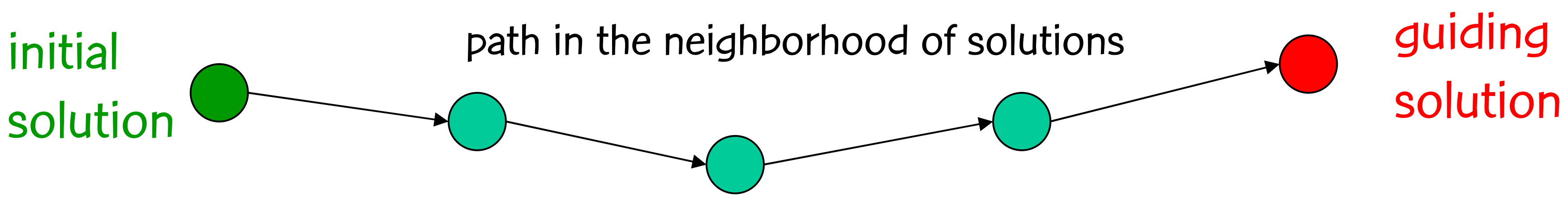
- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Path-relinking

- 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:
    - selection of moves that introduce attributes of the guiding solution into the current solution

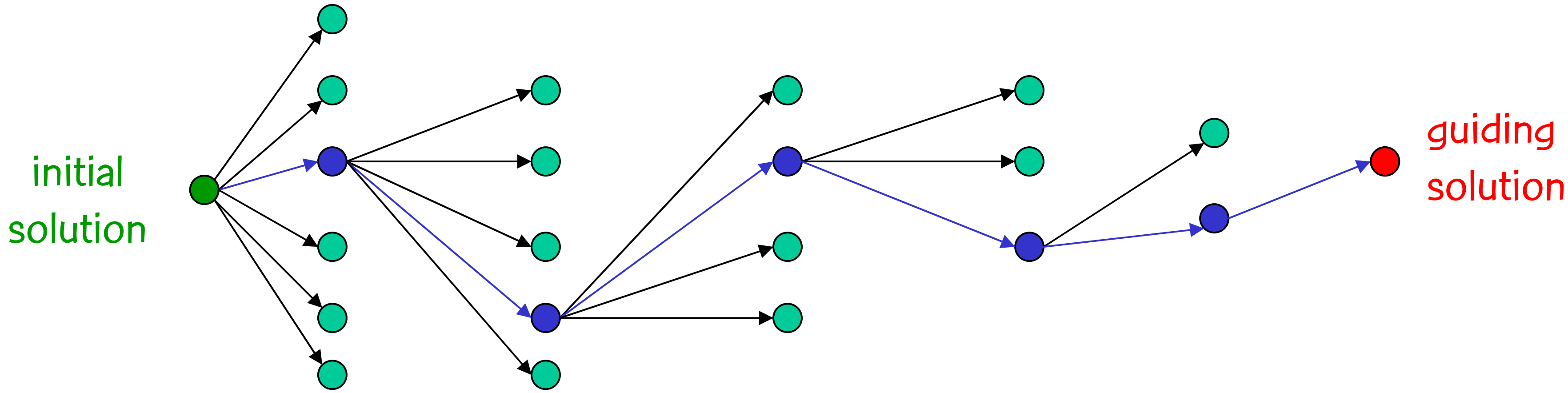
Path-relinking

- 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:



Path-relinking

Elite solutions x and y

$\Delta(x,y)$: symmetric difference between x and y

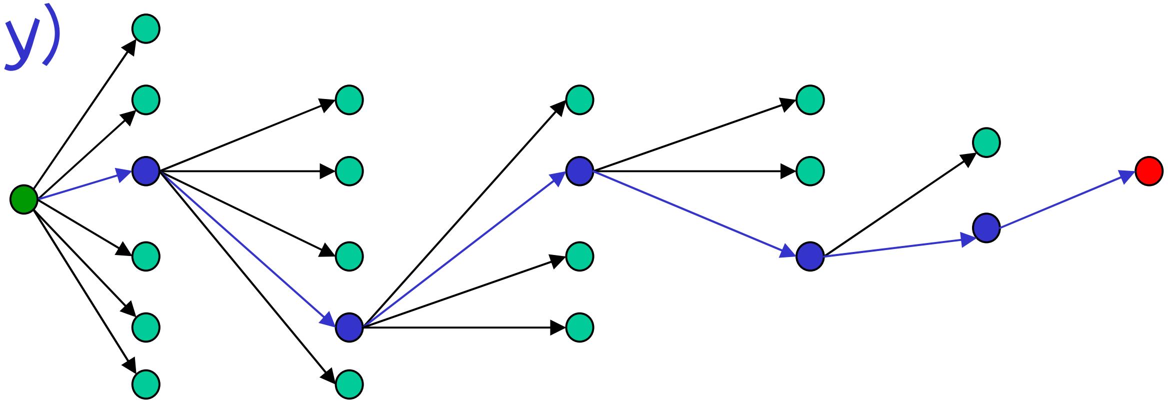
while ($|\Delta(x,y)| > 0$) {

evaluate moves corresponding in $\Delta(x,y)$

make best move

update $\Delta(x,y)$

}



Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

GRASP with path-relinking

- Originally used 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 (may also be selected with probabilities proportional to the symmetric difference w.r.t. the initial solution).
  - Perform path-relinking between these two solutions.

GRASP with path-relinking

- Repeat for Max_Iterations:
  - Construct a greedy randomized solution.
  - Use local search to improve the constructed solution.
  - Apply path-relinking to further improve the solution.
  - Update the pool of elite solutions.
  - Update the best solution found.

GRASP with path-relinking

- Variants: trade-offs between computation time and solution quality
  - Explore different trajectories (e.g. backward, forward): better start from the best, neighborhood of the initial solution is fully explored!
  - Explore both trajectories: twice as much the time, often with marginal improvements only!
  - Do not apply PR at every iteration, but instead only periodically: similar to filtering during local search.
  - Truncate the search, do not follow the full trajectory.
  - May also be applied as a post-optimization step to all pairs of elite solutions.

GRASP with path-relinking

- Successful applications:
  1) Prize-collecting minimum Steiner tree problem: Canuto, Resende, & Ribeiro (2001) (e.g. improved all solutions found by approximation algorithm of Goemans & Williamson)
  2) Minimum Steiner tree problem: Ribeiro, Uchoa, & Werneck (2002) (e.g., best known results for open problems in series dv640 of the SteinLib)
  3) p-median: Resende & Werneck (2002) (e.g., best known solutions for problems in literature)

15'

GRASP with path-relinking

- Successful applications (cont'd):
  4) Capacitated minimum spanning tree: Souza, Duhamel, & Ribeiro (2002) (e.g., best known results for largest problems with 160 nodes)
  5) 2-path network design: Ribeiro & Rosseti (2002) (better solutions than greedy heuristic)
  6) Max-Cut: Festa, Pardalos, Resende, & Ribeiro (2002) (e.g., best known results for several instances)
  7) Quadratic assignment: Oliveira, Pardalos, & Resende (2003)

GRASP with path-relinking

- Successful applications (cont'd):
  8) Job-shop scheduling: Aiex, Binato, & Resende (2003)
  9) Three-index assignment problem: Aiex, Resende, Pardalos, & Toraldo (2003)
  10) PVC routing: Resende & Ribeiro (2003)
  11) Phylogenetic trees: Ribeiro & Vianna (2003)

GRASP with path-relinking

- P is a set (pool) of elite solutions.
- 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.

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Time-to-target-value plots

- Proposition: Let $P(t,p)$ be the probability of not having found a given target solution value in t time units with p independent processors. If $P(t,1) = \exp[-(t-\mu)/\lambda]$ with non-negative λ and μ (two-parameter exponential distribution), then $P(t,p) = \exp[-p.(t-\mu)/\lambda]$.
  $\Rightarrow$ if $p\mu << \lambda$, then the probability of finding a solution within a given target value in time p.t with a sequential algorithm is approximately equal to that of finding a solution with the same quality in time t with p processors.

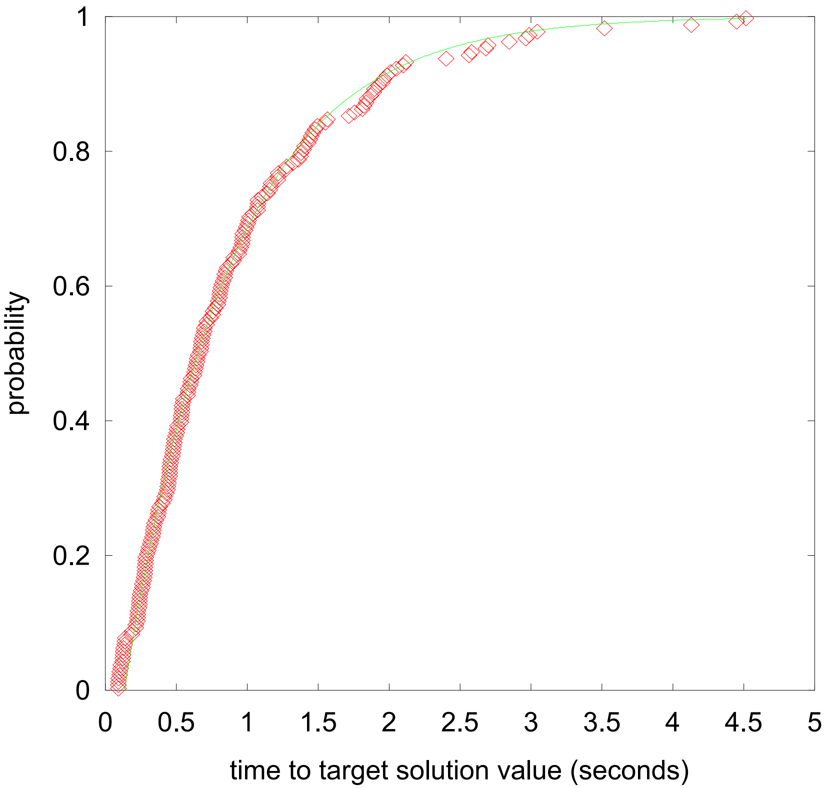
Time-to-target-value plots

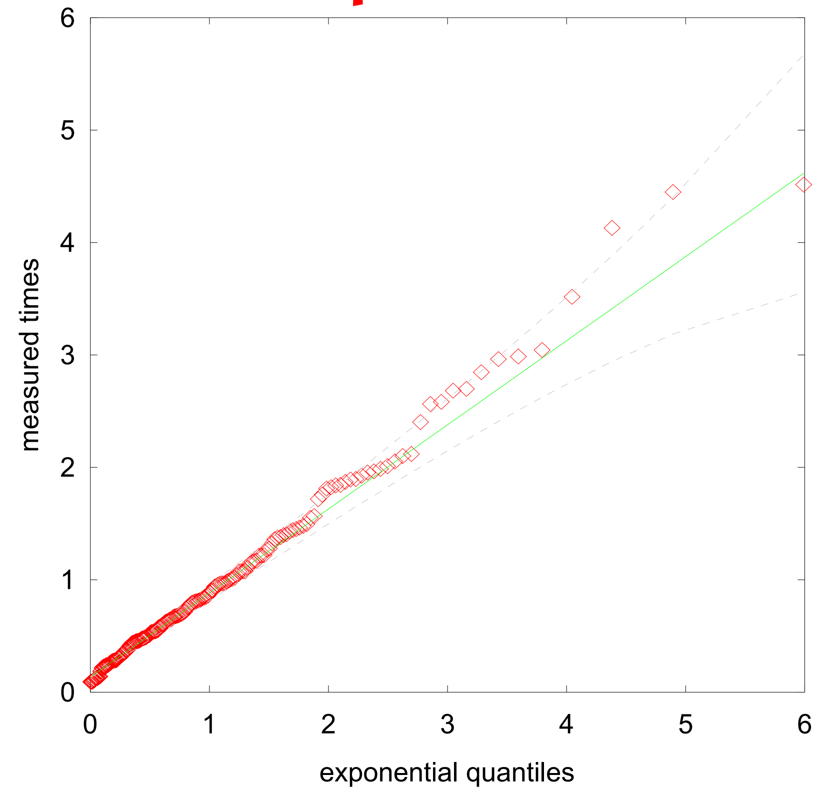
- Probability distribution of time-to-target-solution-value: Aiex, Resende, & Ribeiro (2002) and Aiex, Binato, & Resende (2003) have shown experimentally that both pure GRASP and GRASP with path-relinking present this behavior.

Time-to-target-value plots

- Probability distribution of time-to-target-solution-value: experimental plots
- Select an instance and a target value.
- For each variant of GRASP with path-relinking:
  - Perform 200 runs using different seeds.
  - Stop when a solution value at least as good as the target is found.
  - For each run, measure the time-to-target-value.
  - Plot the probabilities of finding a solution at least as good as the target value within some computation time.

Time-to-target-value plots





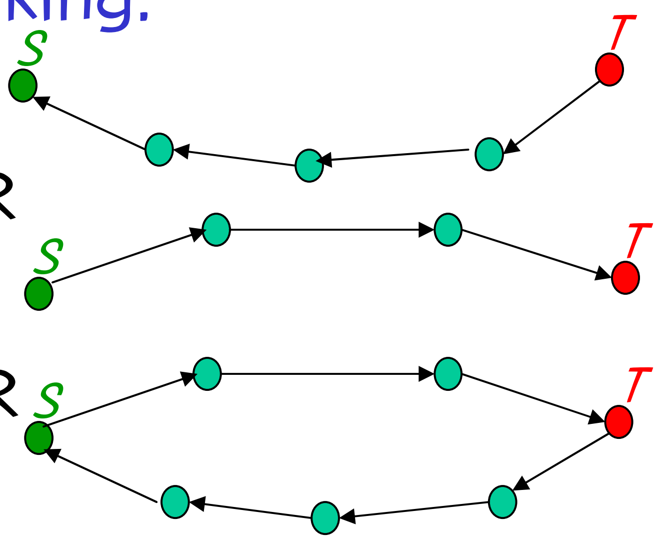
Random variable time-to-target-solution value fits a two-parameter exponential distribution.

Therefore, one should expect approximate linear speedup in a straightforward (independent) parallel implementation.

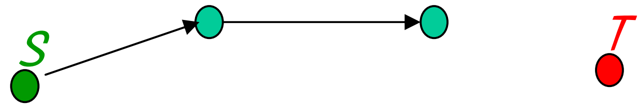
Variants of GRASP + PR

- Variants of GRASP with path-relinking:
  - GRASP: pure GRASP
  - G+PR(B): GRASP with backward PR
  - G+PR(F): GRASP with forward PR
  - G+PR(BF): GRASP with two-way PR

  T: elite solution *S:* local search



- Other strategies:
  - Truncated path-relinking



  - Do not apply PR at every iteration (frequency)

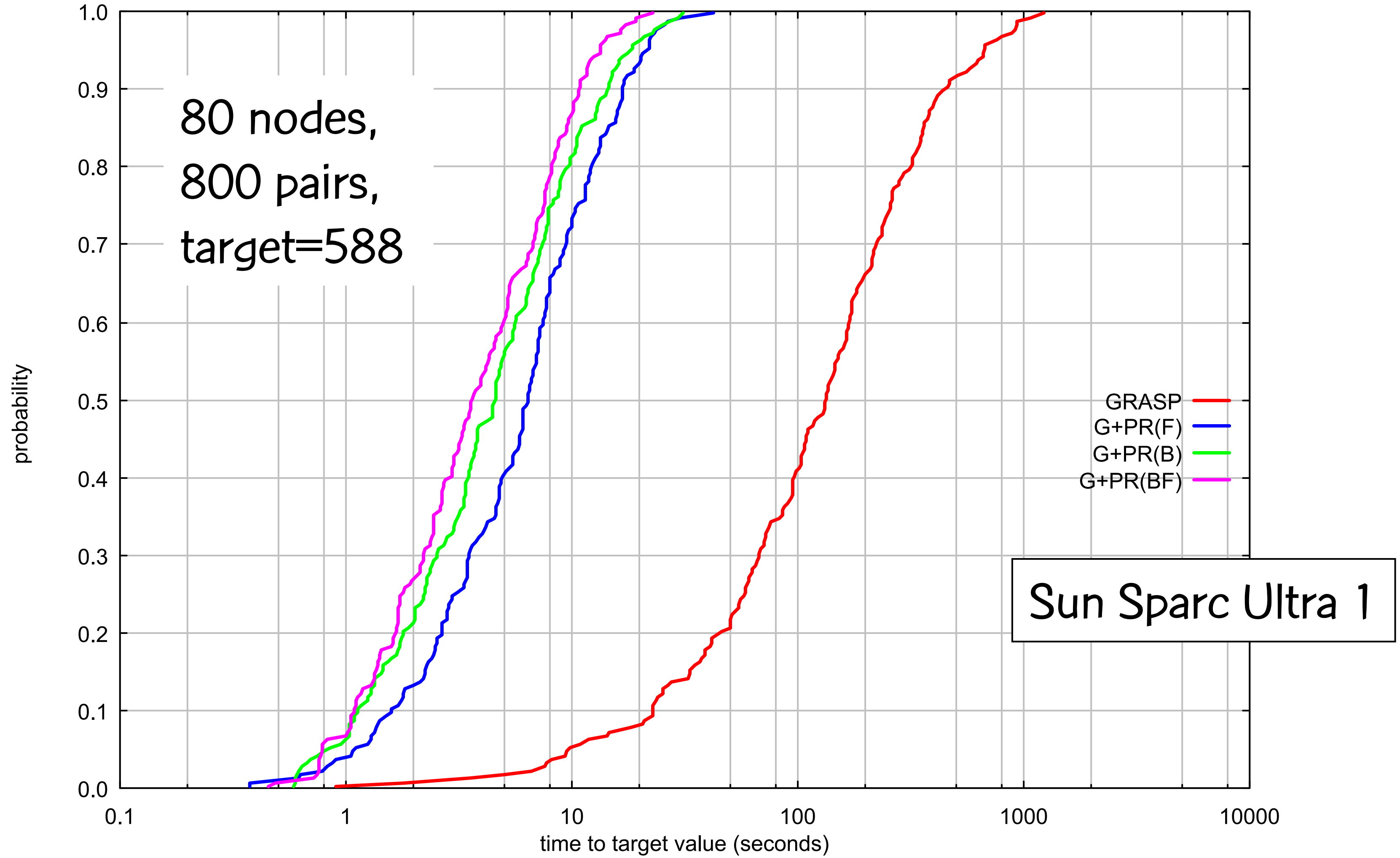
20'

2-path network design problem

- 2-path network design problem:
  - Graph $G=(V,E)$ with edge weights w_e and set D of origin-destination pairs (demands): find a minimum weighted subset of edges $E' \subseteq E$ containing a 2-path (path with at most two edges) in G between the extremities of every origin-destination pair in D.
- Applications: design of communication networks, in which paths with few edges are sought to enforce high reliability and small delays

2-path network design problem

Each variant: 200 runs for one instance of 2PNDP



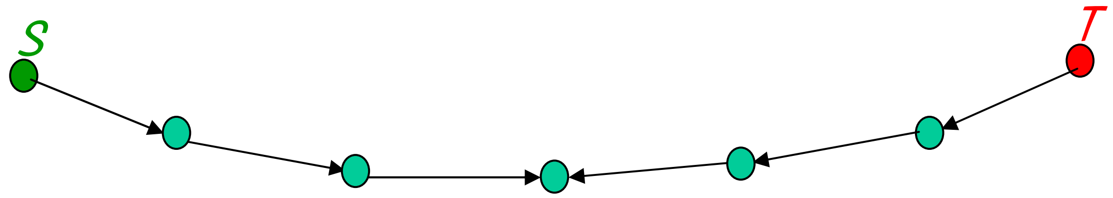
2-path network design problem

- Same computation time: probability of finding a solution at least as good as the target value increases from GRASP $\rightarrow$ G+PR(F) $\rightarrow$ G+PR(B) $\rightarrow$ G+PR(BF)
- P(h,t) = probability that variant h finds a solution as good as the target value in time no greater than t
  - P(GRASP,10s) = 2% P(G+PR(F),10s) = 56%
    P(G+PR(B),10s) = 75% P(G+PR(BF),10s) = 84%

Variants of GRASP + PR

- More recently:
  - G+PR(M): mixed back and forward strategy

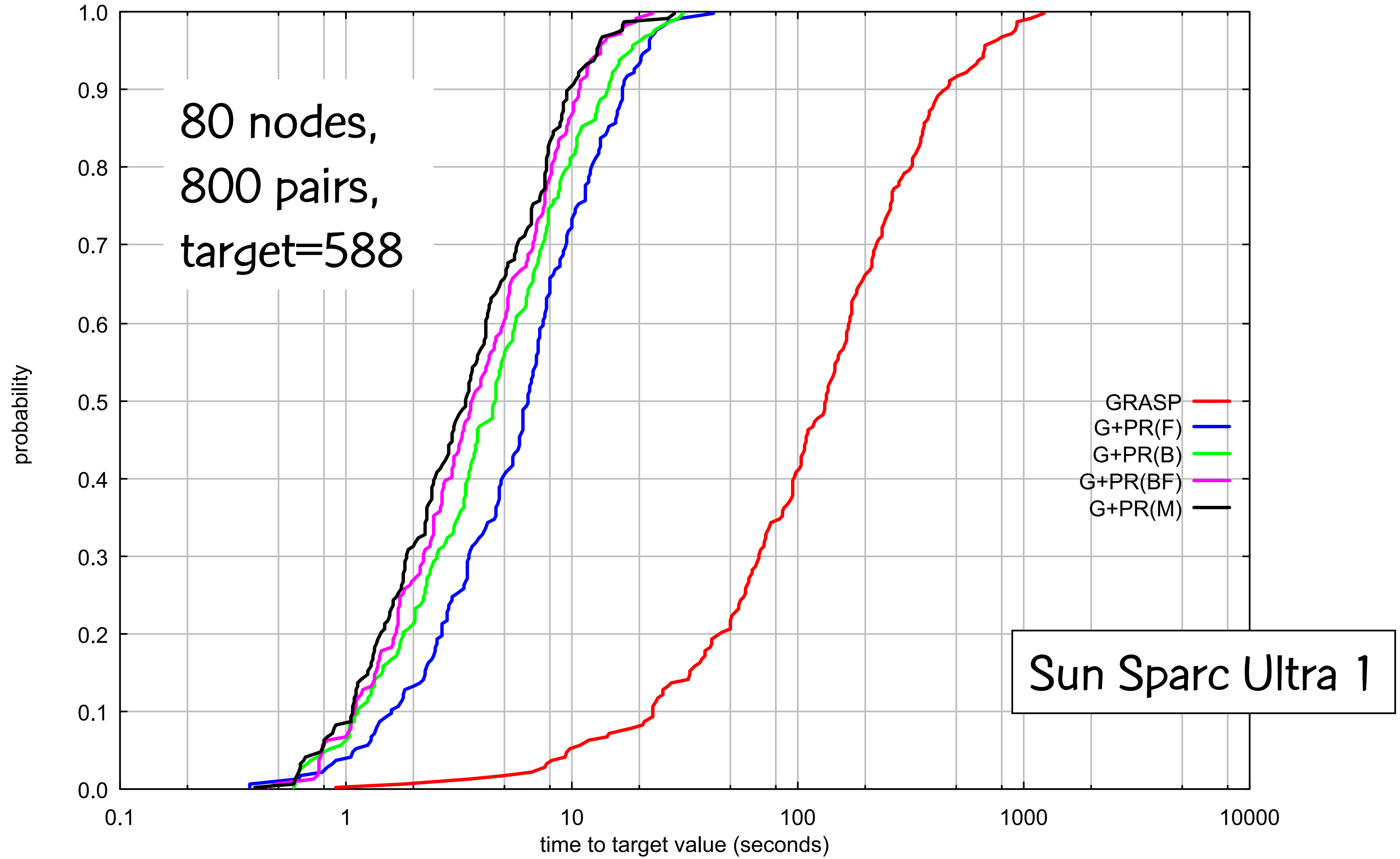
    T: elite solution   *S:* local search



  - Path-relinking with local search

2-path network design problem

Each variant: 200 runs for one instance of 2PNDP



2-path network design problem

Instance	GRASP	G+PR(F)	G+PR(B)	G+PR(FB)	G+PR(M)
100-3	773	762	756	757	754
100-5	756	742	739	737	728
200-3	1564	1523	1516	1508	1509
200-5	1577	1567	1543	1529	1531
300-3	2448	2381	2339	2356	2338
300-5	2450	2364	2328	2347	2322
400-3	3388	3311	3268	3227	3257
400-5	3416	3335	3267	3270	3259
500-3	4347	4239	4187	4170	4187
500-5	4362	4263	4203	4211	4200

10 runs, same computation time for each variant, best solution found

2-path network design problem

- Effectiveness of G+PR(M):
  - 100 small instances with 70 nodes generated as in Dahl and Johannessen (2000) for comparison purposes.
  - Statistical test *t* for unpaired observations
  - GRASP finds better solutions with 40% of confidence (unpaired observations and many optimal solutions): Ribeiro & Rosseti (2002)

	G+PR(M) Sample A	D&J Sample B
Size	100	30
Mean	443.7 (-2.2%)	453.7
Std. dev.	40.6	61.6

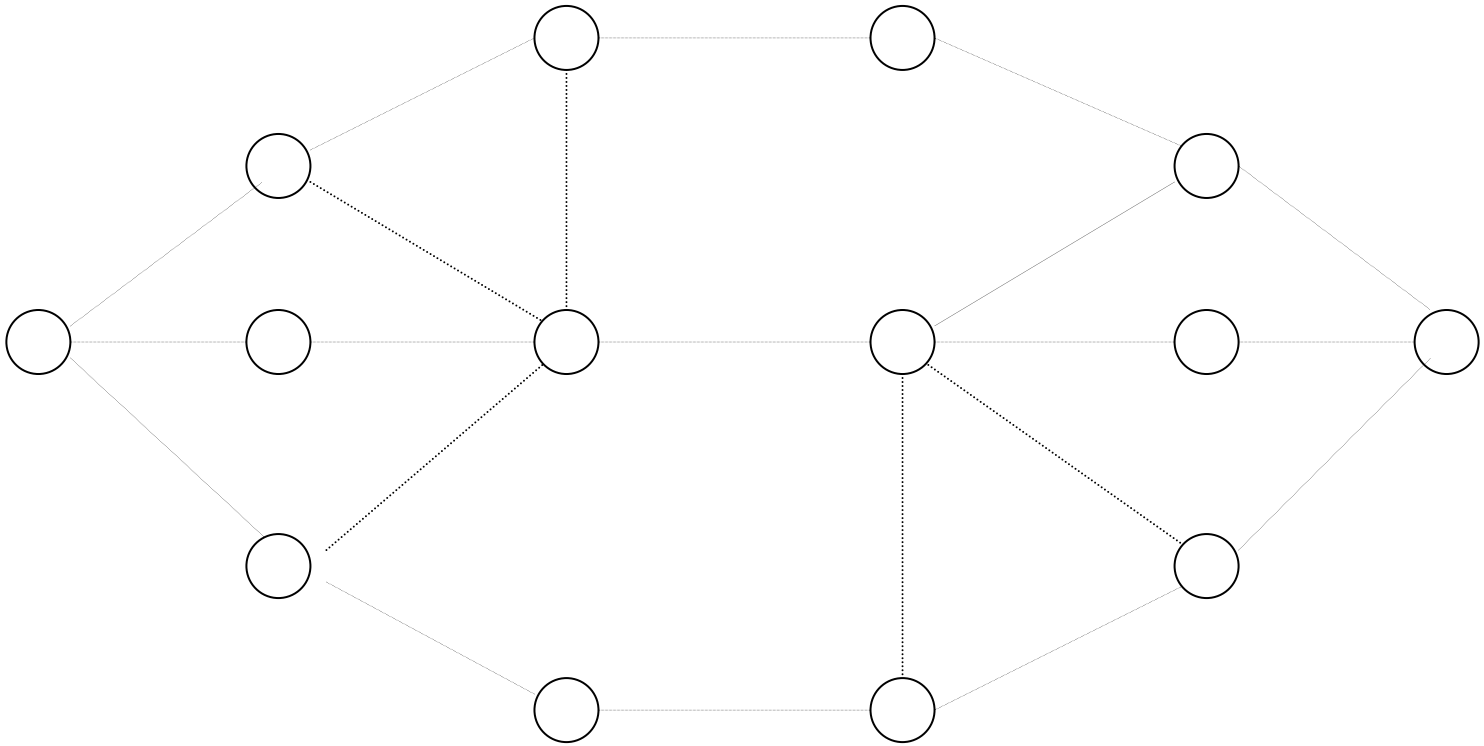
2-path network design problem

- Effectiveness of path-relinking to improve and speedup the pure GRASP.
- Strategies using the backwards component are systematically better.

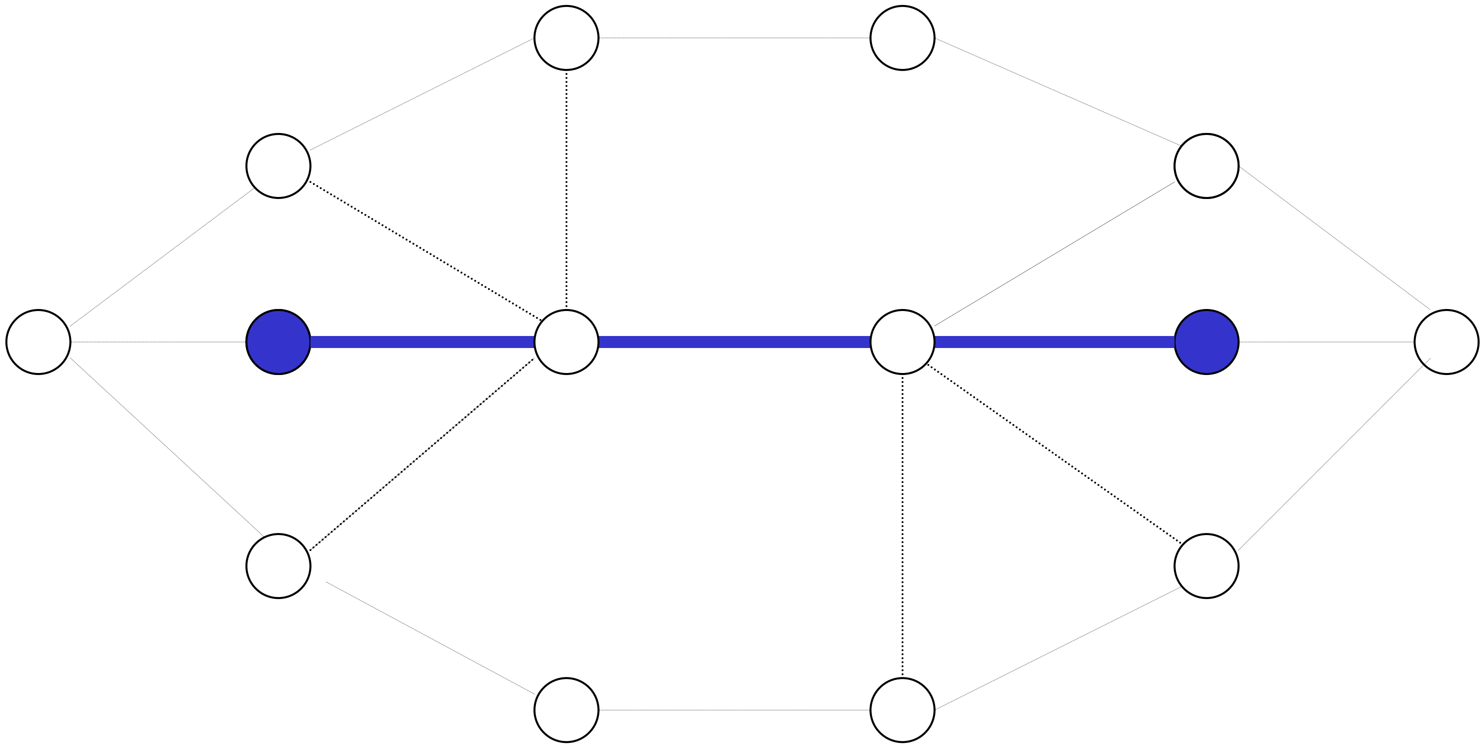
PVC routing

- Frame relay service offers virtual private networks to customers by providing long-term private virtual circuits (PVCs) between customer endpoints on a backbone network.
- Routing is done either automatically by switch or by the network designer without any knowledge of future requests.
- Over time, these decisions cause inefficiencies in the network and occasionally offline rerouting (grooming) of the PVCs is needed:
  - integer multicommodity network flow problem: Resende & Ribeiro (2003)

PVC routing

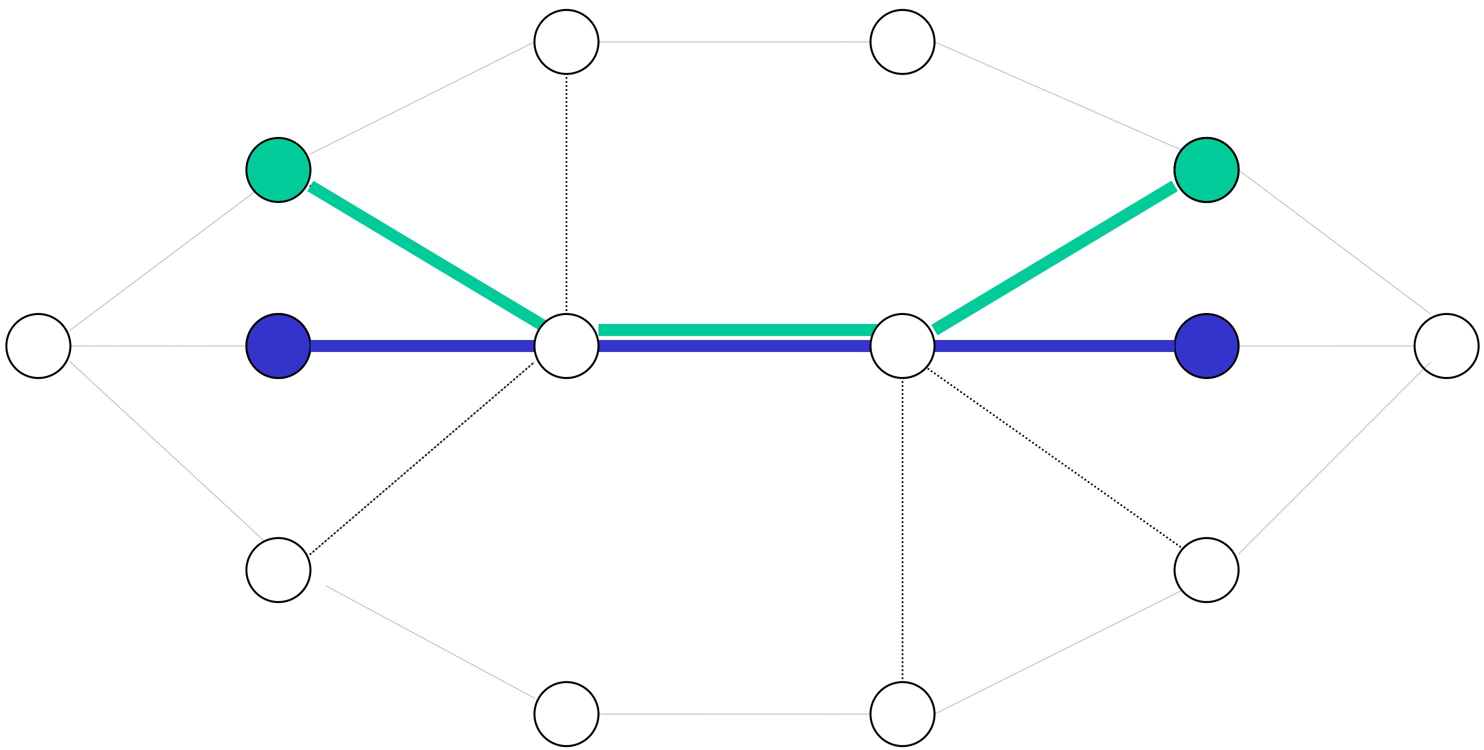


PVC routing

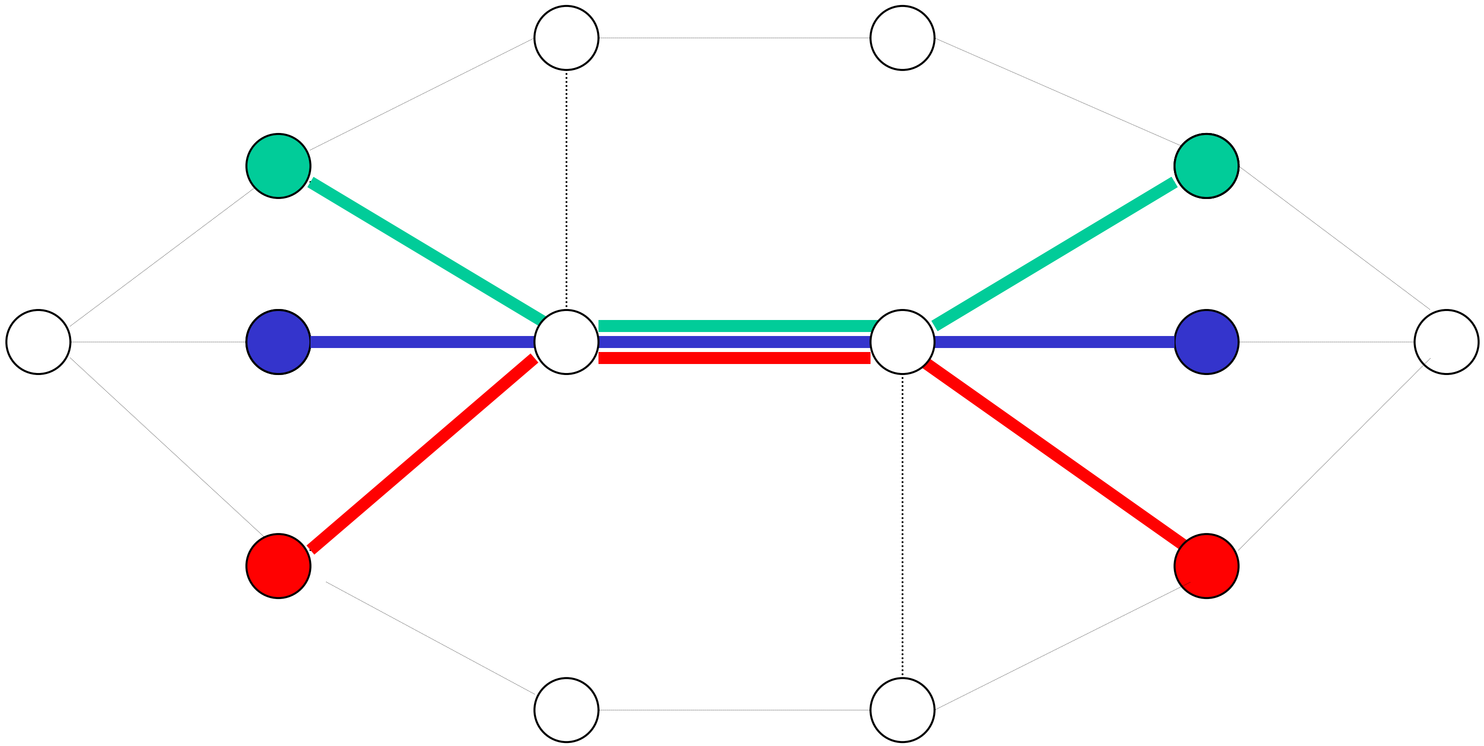


25'

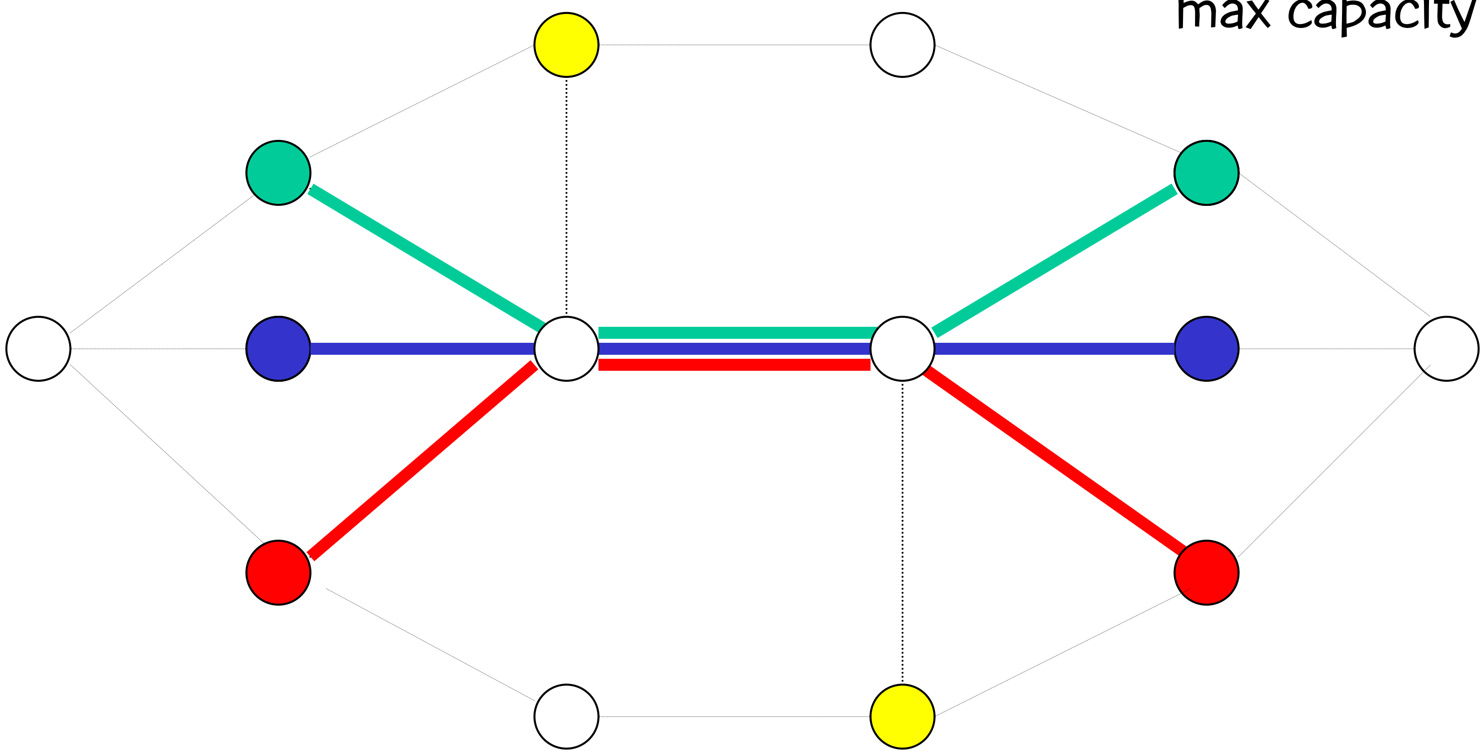
PVC routing



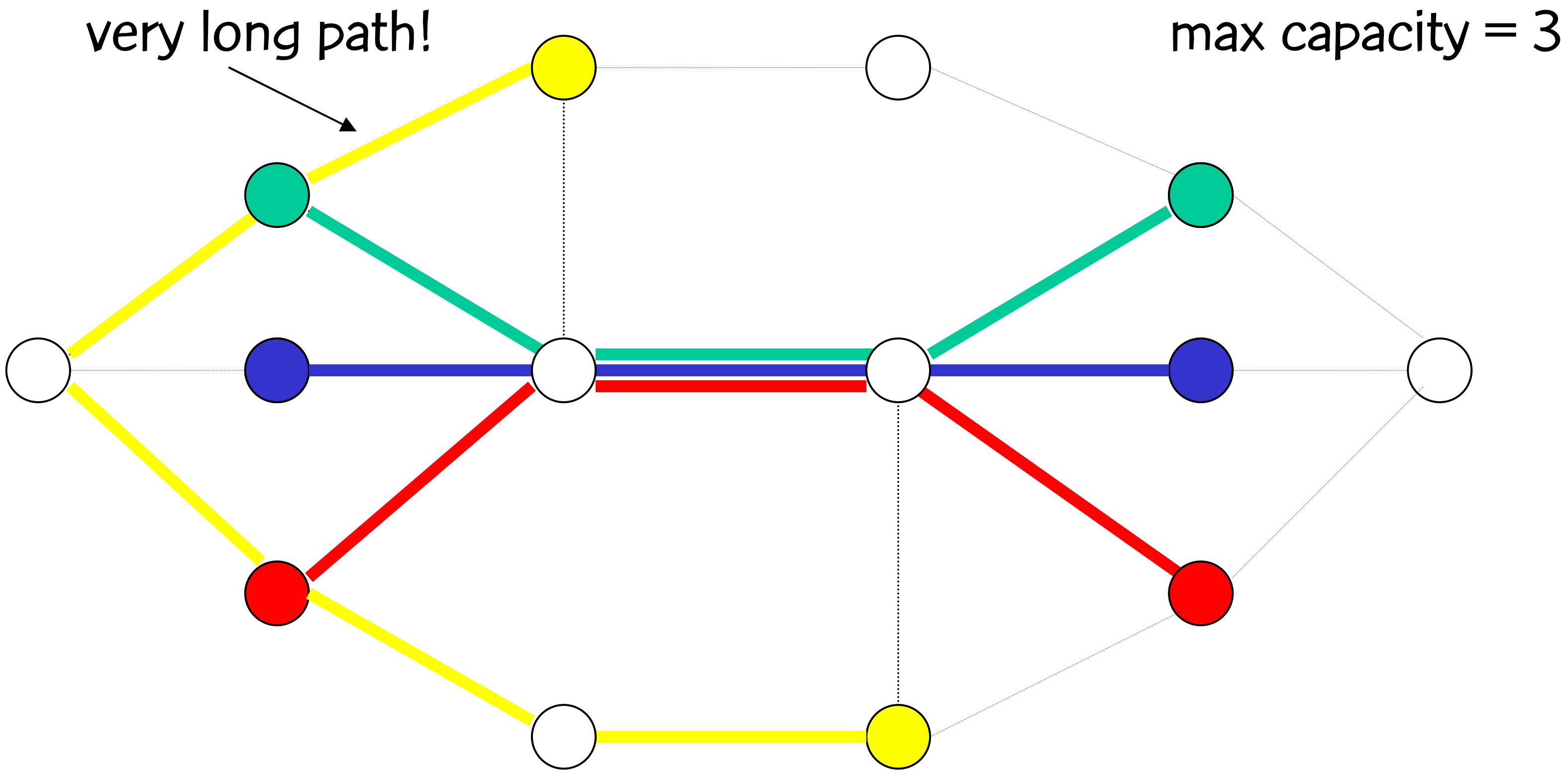
PVC routing



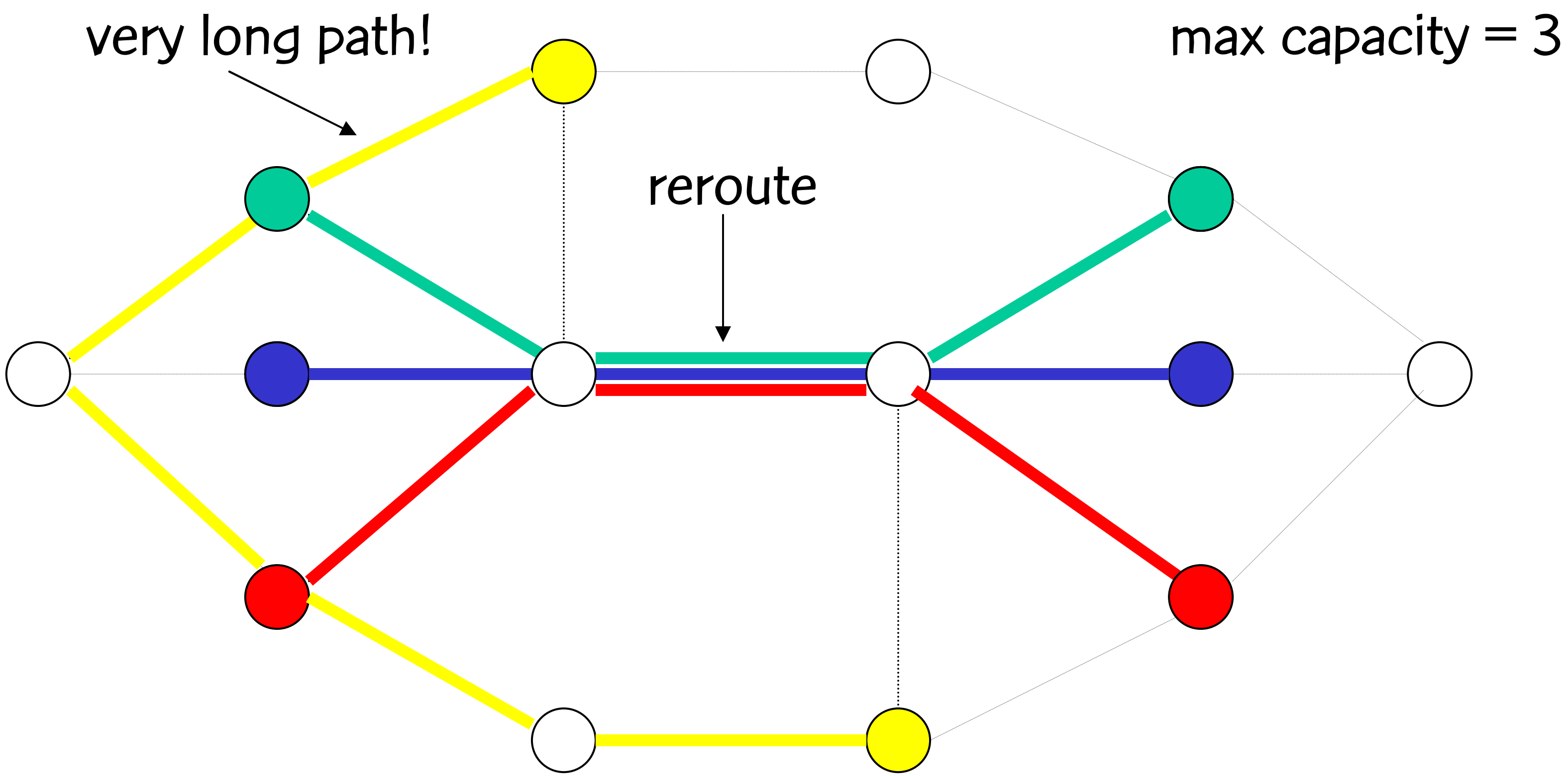
PVC routing



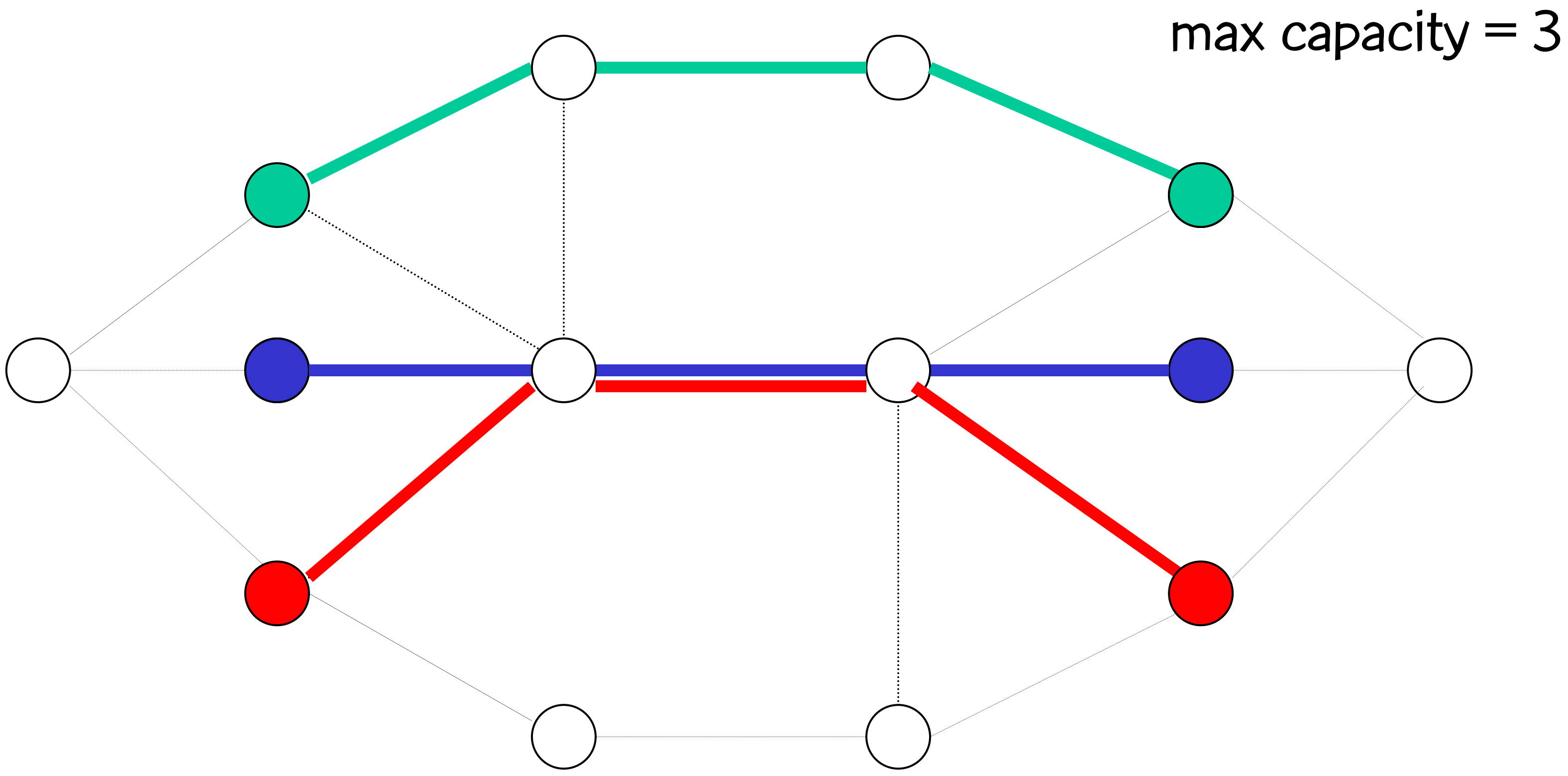
PVC routing



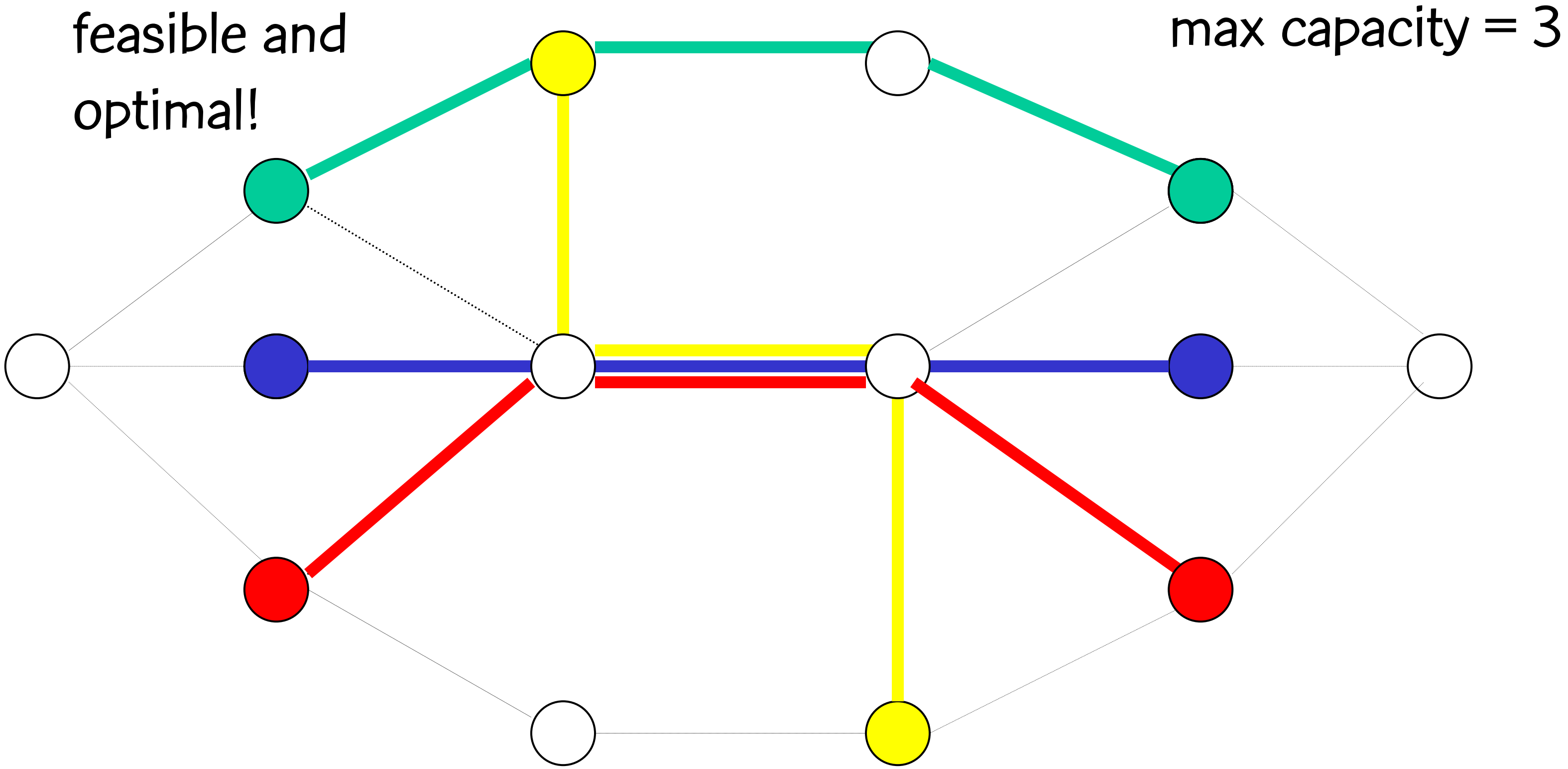
PVC routing



PVC routing

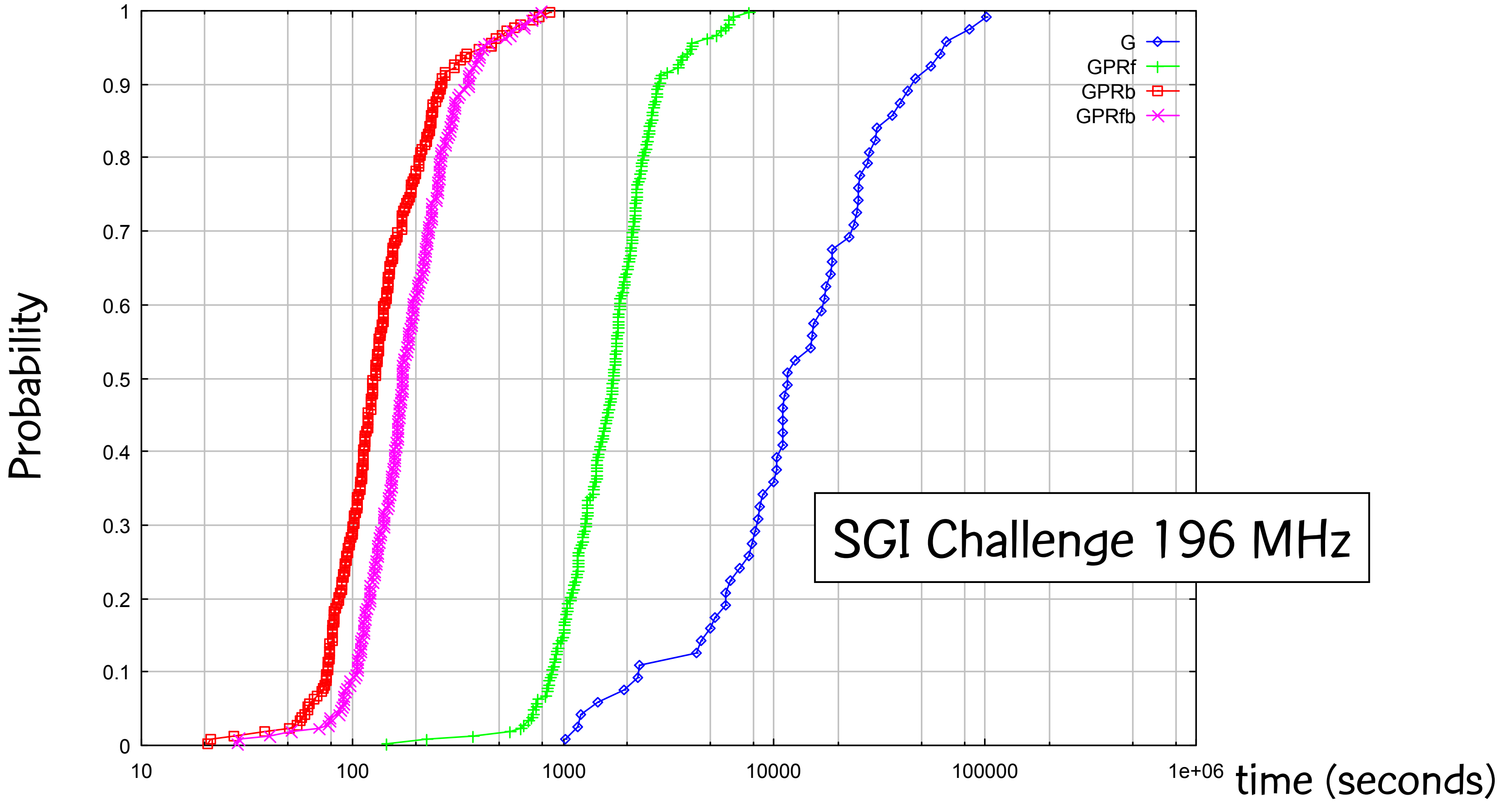


PVC routing



PVC routing

Each variant: 200 runs for one instance of PVC routing problem
(60 nodes, 498 edges, 750 origin-destination pairs)



PVC routing

10 runs	10 seconds		100 seconds	
Variant	best	average	best	average
GRASP	126603	126695	126228	126558
G+PR(F)	126301	126578	126083	126229
G+PR(B)	125960	126281	125666	125883
G+PR(BF)	125961	126307	125646	125850

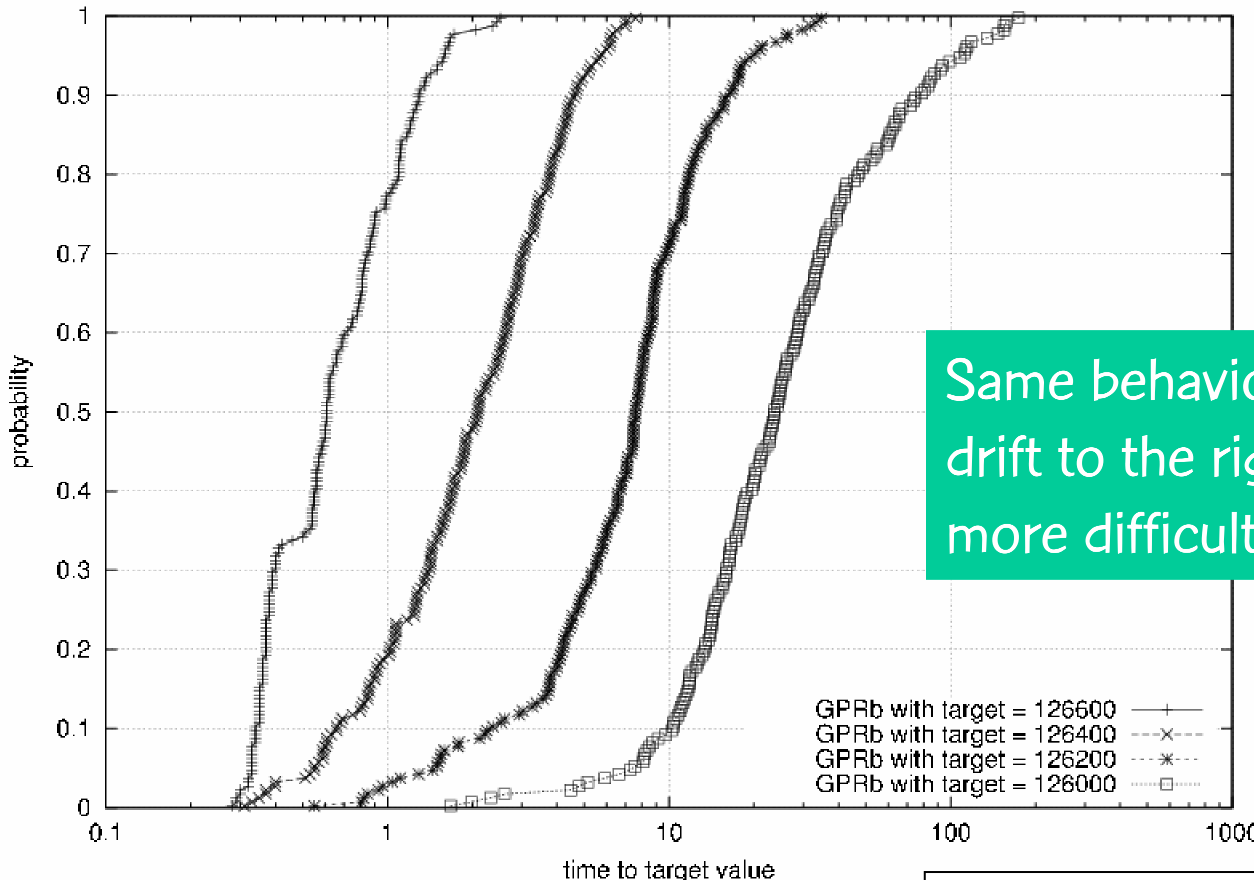
PVC routing

10 runs	10 seconds		100 seconds	
Variant	best	average	best	average
GRASP	126603	126695	126228	126558
G+PR(F)	126301	126578	126083	126229
G+PR(B)	125960	126281	125666	125883
G+PR(BF)	125961	126307	125646	125850

30'

PVC routing

GRASP + PR backwards: four increasingly difficult target values



Same behavior, plots drift to the right for more difficult targets

SGI Challenge 196 MHz

GRASP with path-relinking

- Post-optimization "evolutionary" strategy:

a) Start with pool P_0 found at end of GRASP and set $k = 0$.

b) Combine with path-relinking all pairs of solutions in P_k.

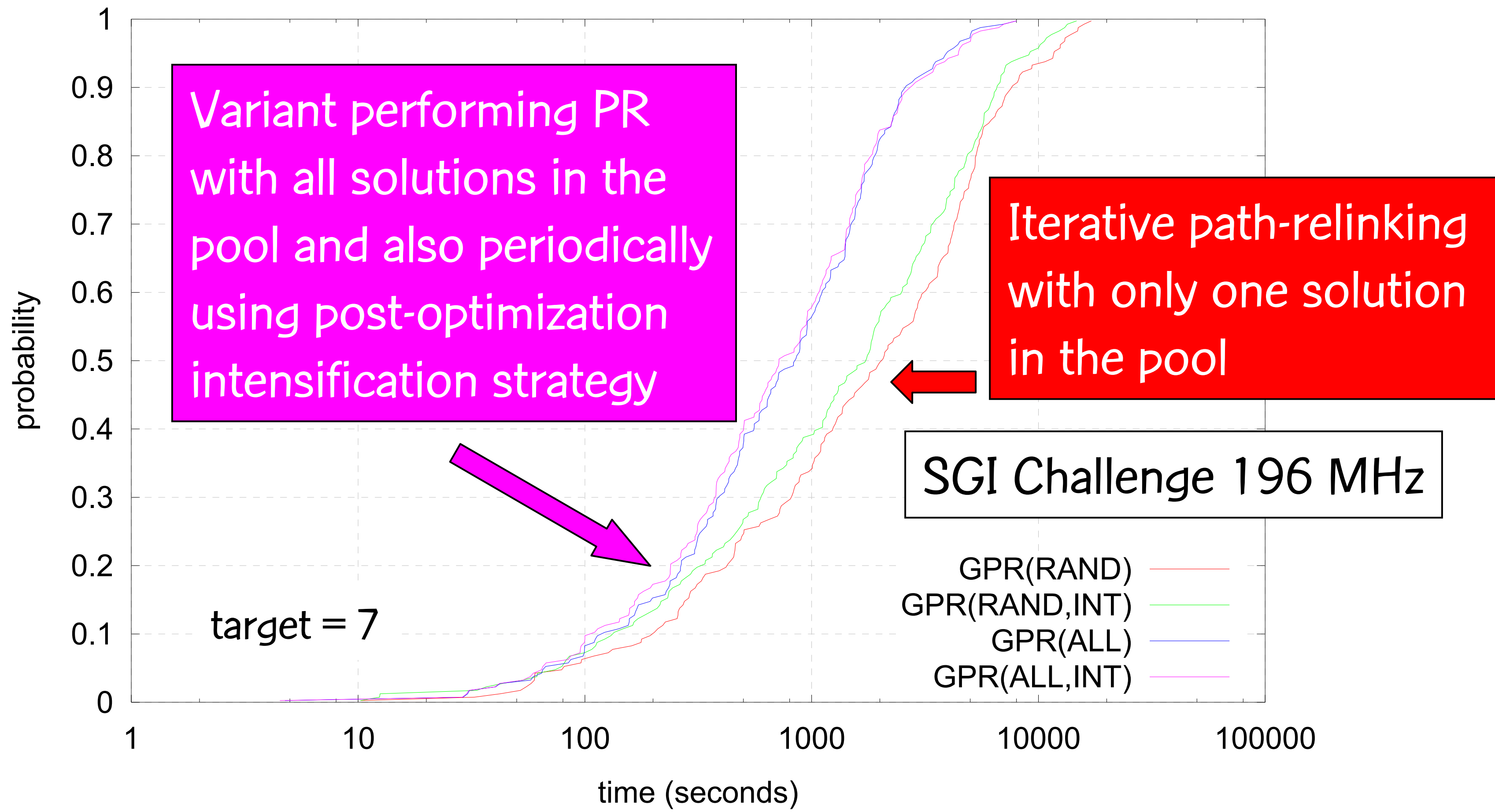
c) Solutions obtained by combining solutions in P_k are added to a new pool P_{k+1} following same constraints for updates as before.

d) If best solution of P_{k+1} is better than best solution of P_k, then set $k = k + 1$, and go back to step (b).

- Succesfully used by Ribeiro, Uchoa, & Werneck (2002) (Steiner) and Resende & Werneck (2002) (p-median)

3-index assignment (AP3)

Each variant: 200 runs for instance Balas & Saltzman 20.1 of 3AP



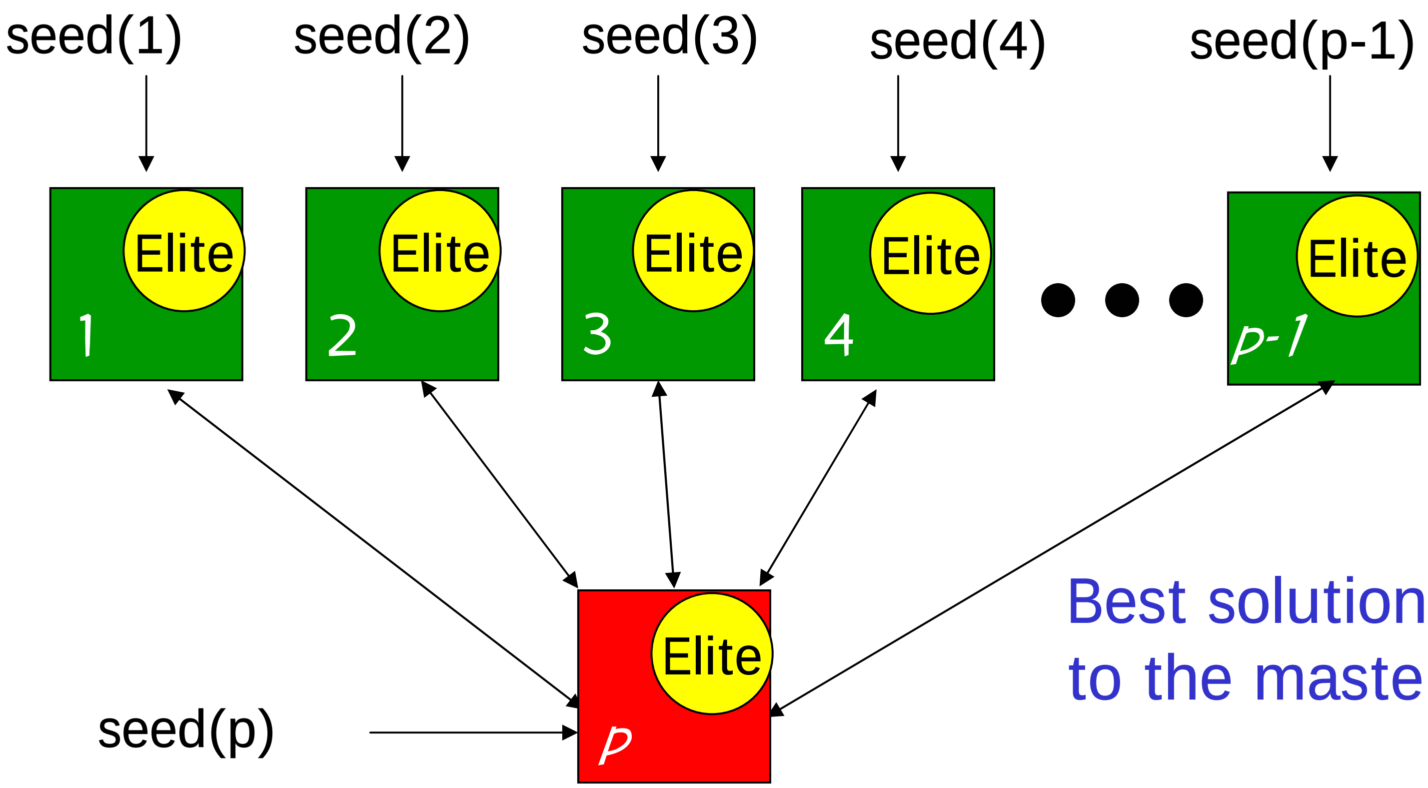
Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Parallel independent implementation

- Parallelism in metaheuristics: robustness
  Cung, Martins, Ribeiro, & Roucairol (2001)
- Multiple-walk independent-thread strategy:
  - p processors available
  - Iterations evenly distributed over p processors
  - Each processor keeps a copy of data and algorithms.
  - One processor acts as the master handling seeds, data, and iteration counter, besides performing GRASP iterations.
  - Each processor performs Max_Iterations/p iterations.

Parallel independent implementation

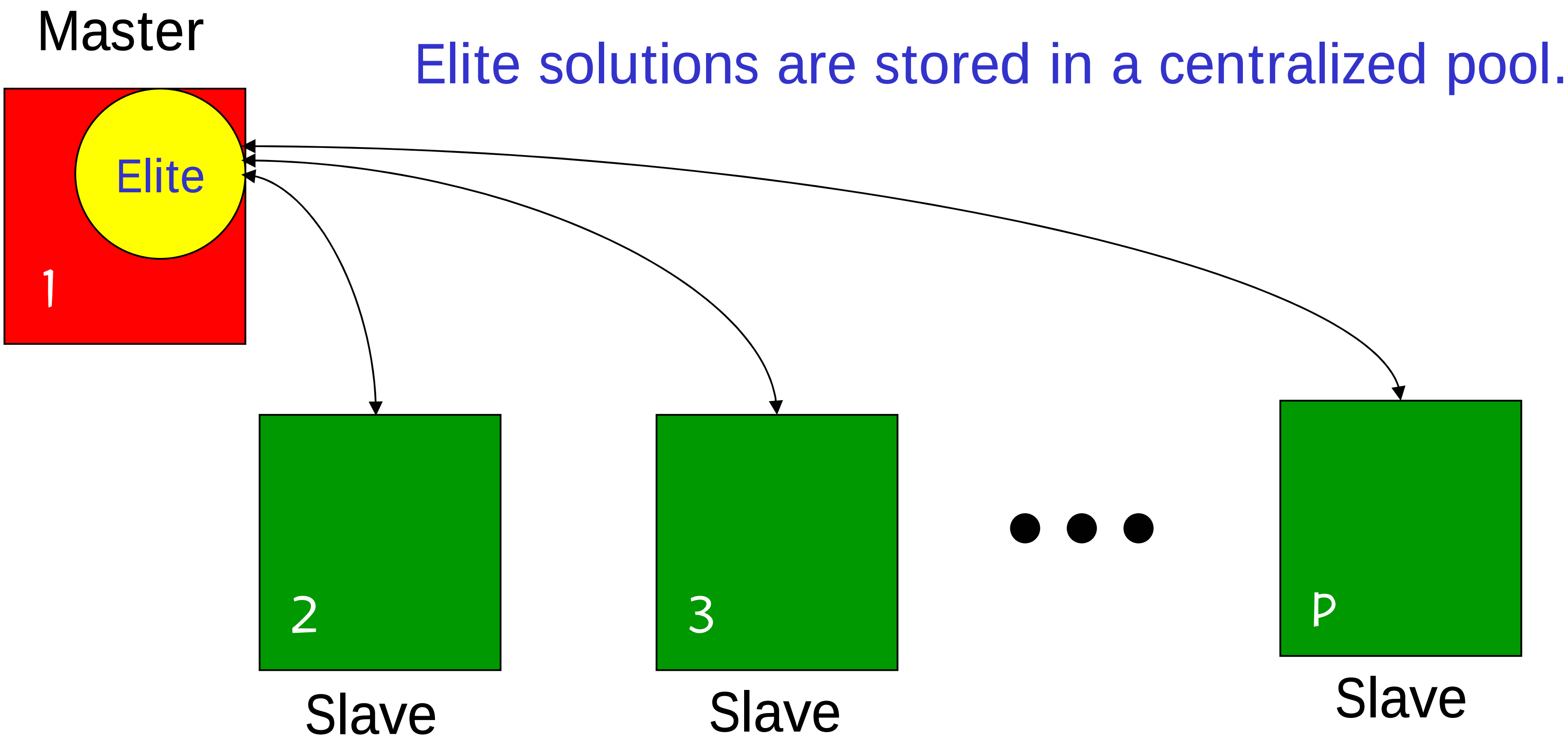


Best solution is sent to the master.

Parallel cooperative implementation

- Multiple-walk cooperative-thread strategy:
  - p processors available
  - Iterations evenly distributed over p-1 processors
  - Each processor has a copy of data and algorithms.
  - One processor acts as the master handling seeds, data, and iteration counter and handles the pool of elite solutions, but does not perform GRASP iterations.
  - Each processor performs Max_Iterations/(p–1) iterations.

Parallel cooperative implementation



Parallel environment

- 2-path network design
- Linux cluster with 32 Pentium II-400 MHz processors with 32 Mbytes of RAM each
- IBM 8274 switch with 96 ports (10 Mbits/s)
- Implementations in C using MPI LAM 6.3.2 and bidirectional path-relinking (BF)

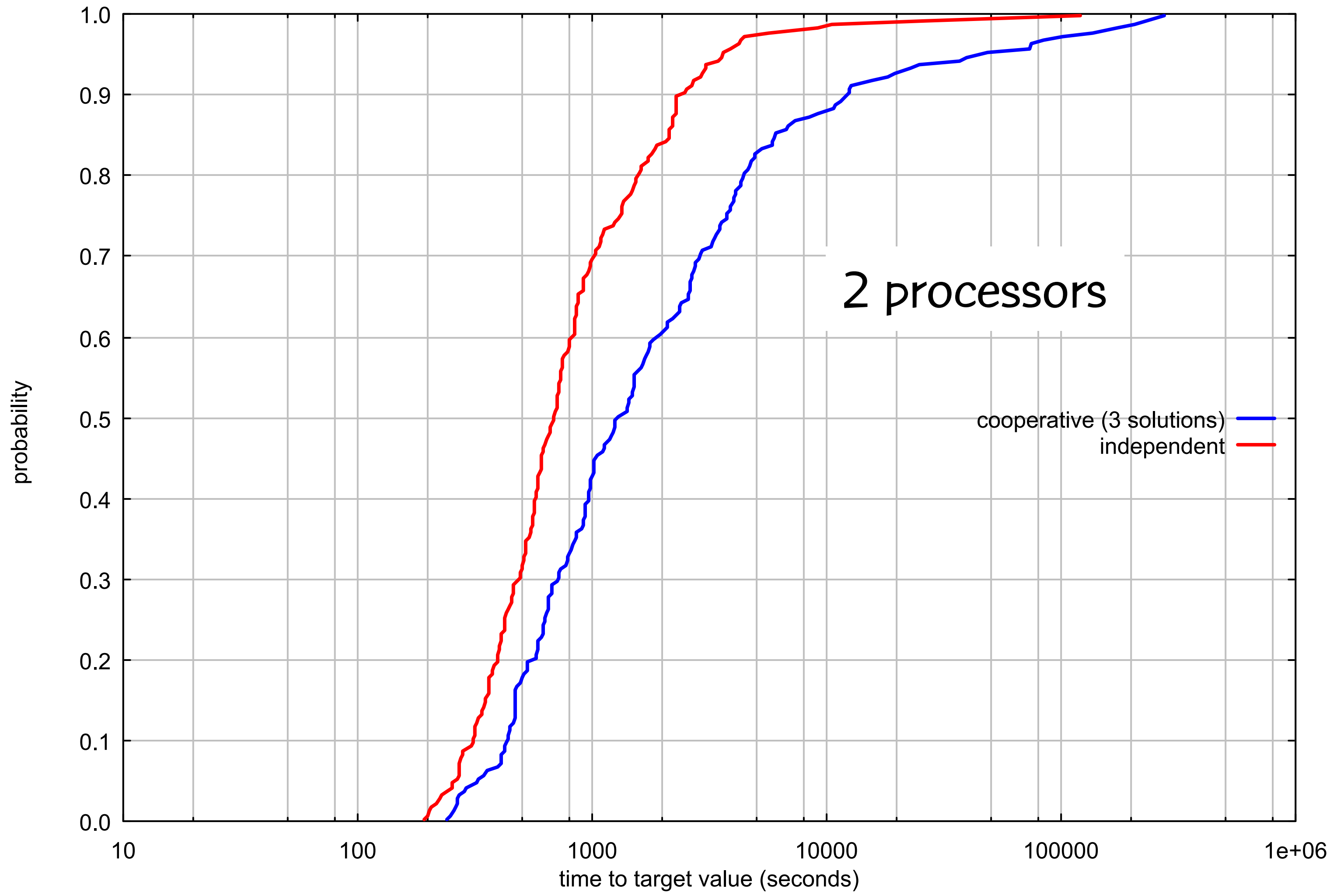
Cooperative vs. independent strategies

- Same instance: 15 runs with different seeds, 3200 iterations
- Pool is poorer when fewer GRASP iterations are done and solution quality deteriorates

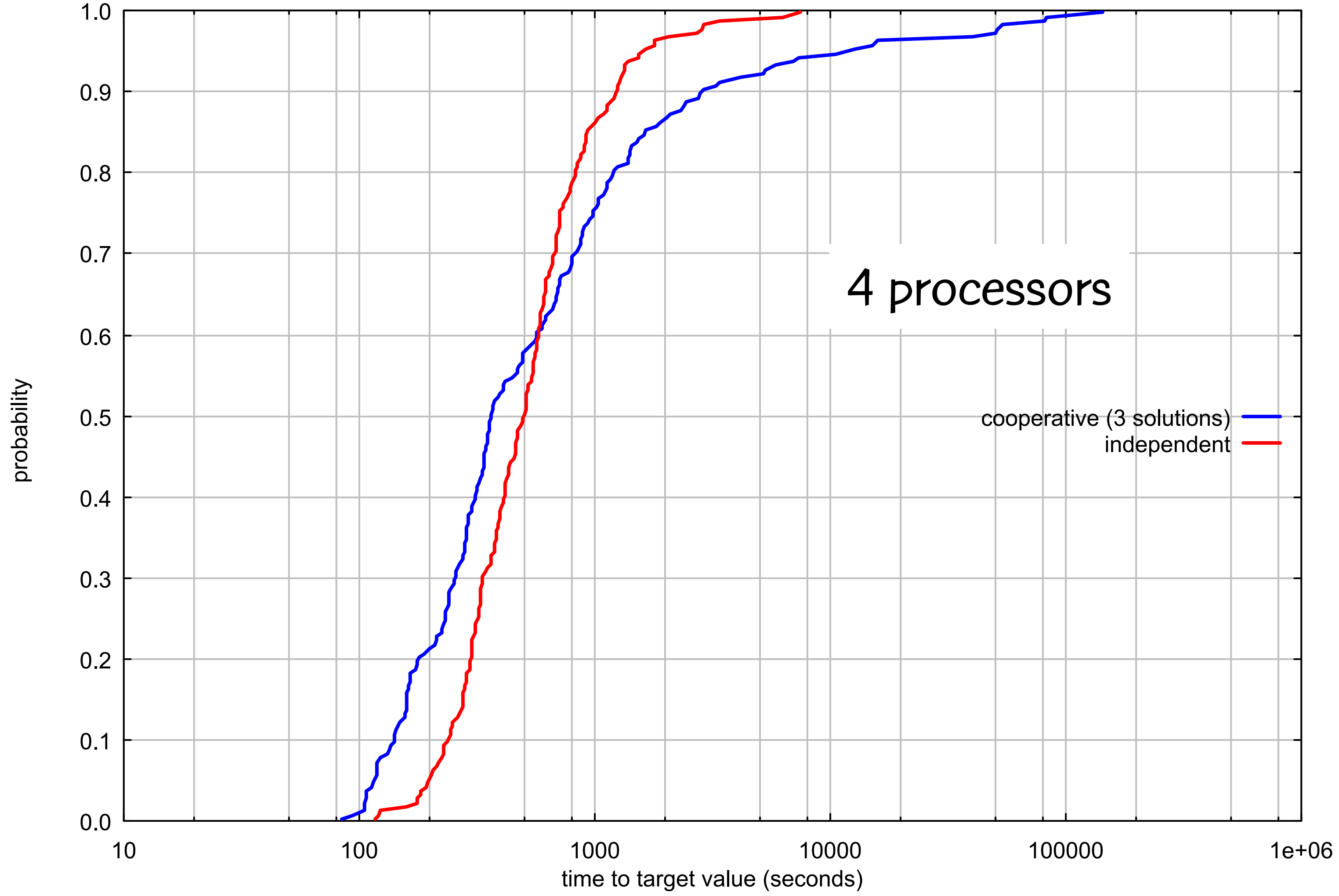
	Independent		Cooperative	
procs.	best	avg.	best	avg.
1	673	678.6	-	-
2	676	680.4	676	681.6
4	680	685.1	673	681.2
8	687	690.3	676	683.1
16	692	699.1	674	682.3
32	702	708.5	678	684.8

35'

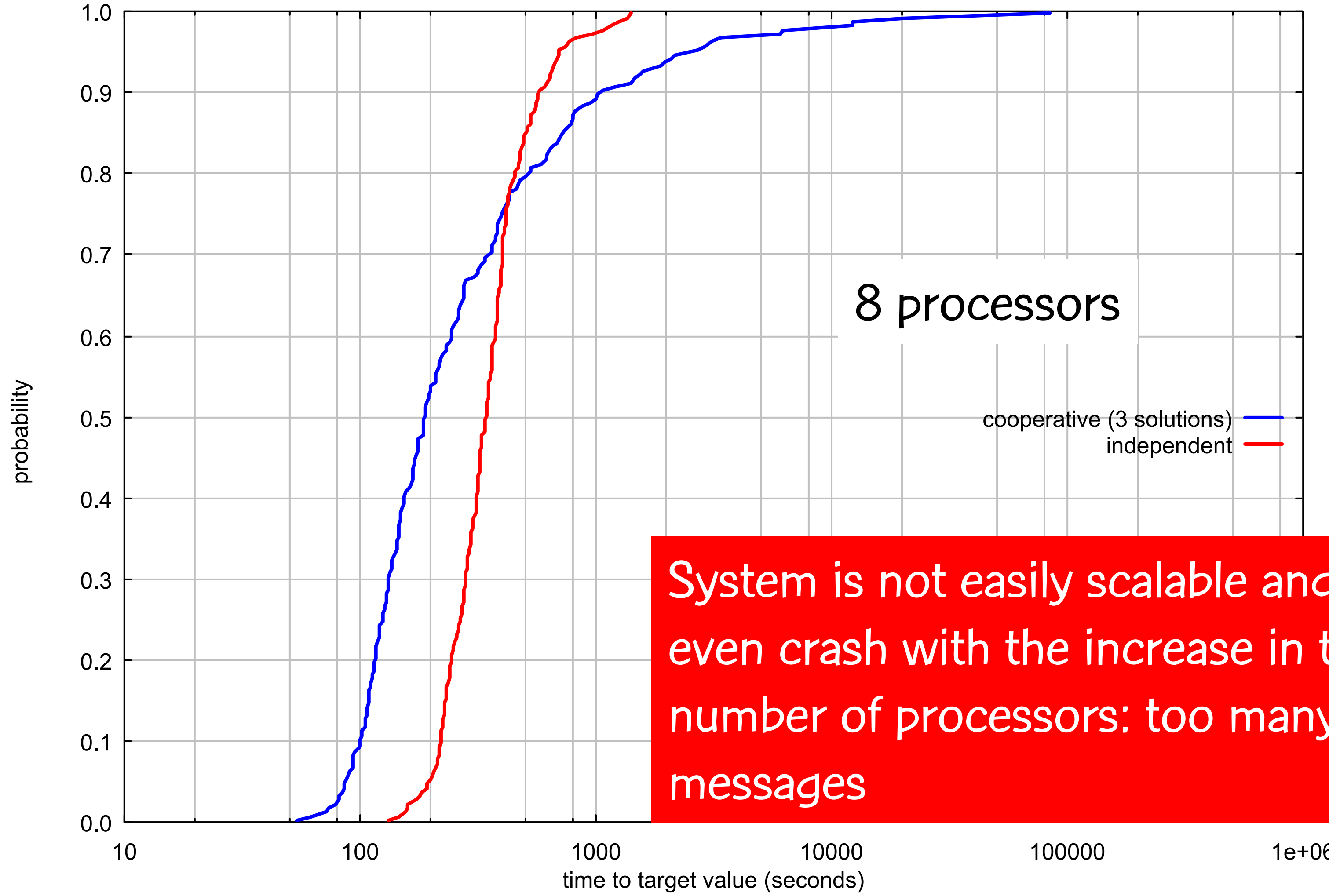
Cooperative vs. independent strategies



Cooperative vs. independent strategies



Cooperative vs. independent strategies

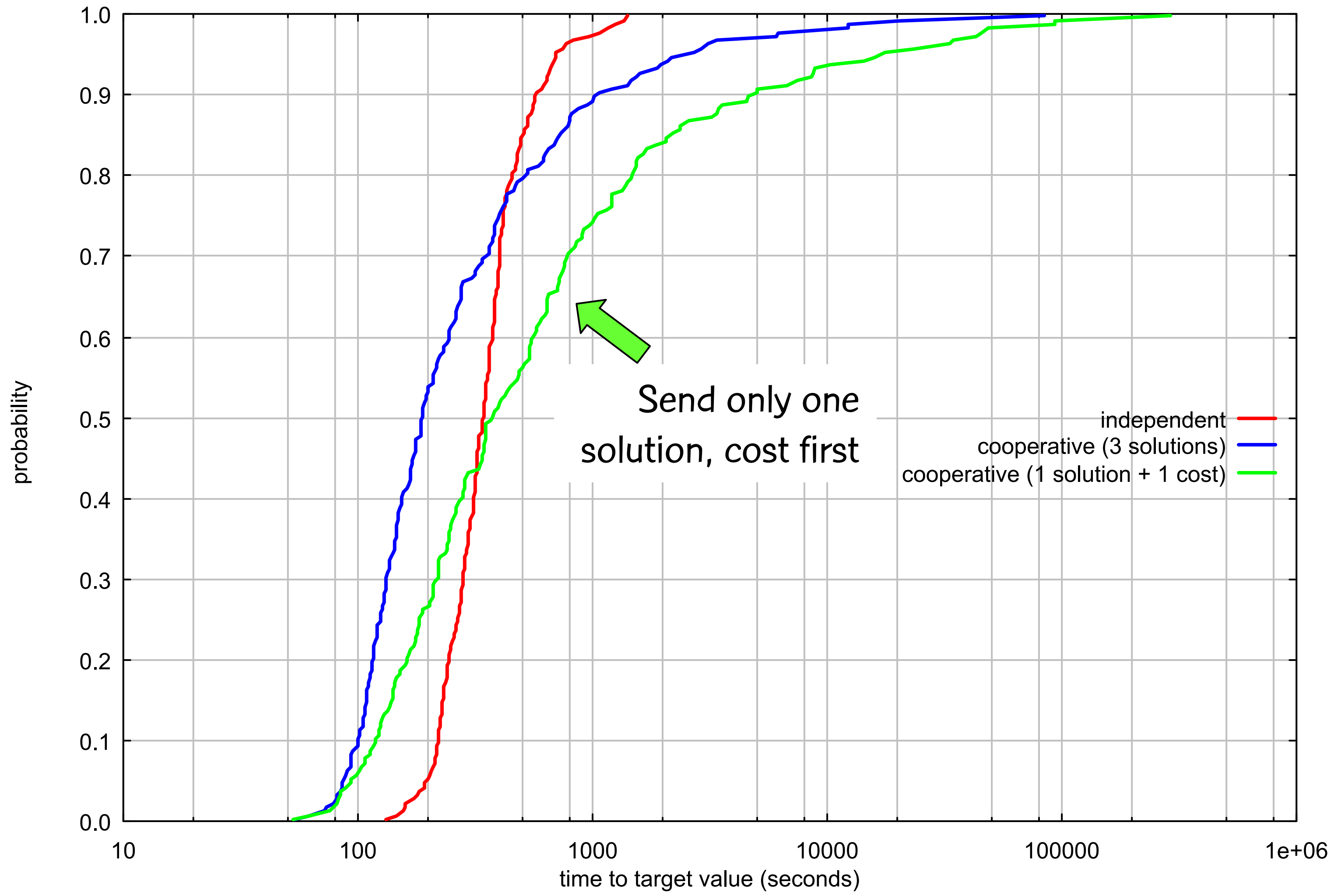


System is not easily scalable and may even crash with the increase in the number of processors: too many large messages

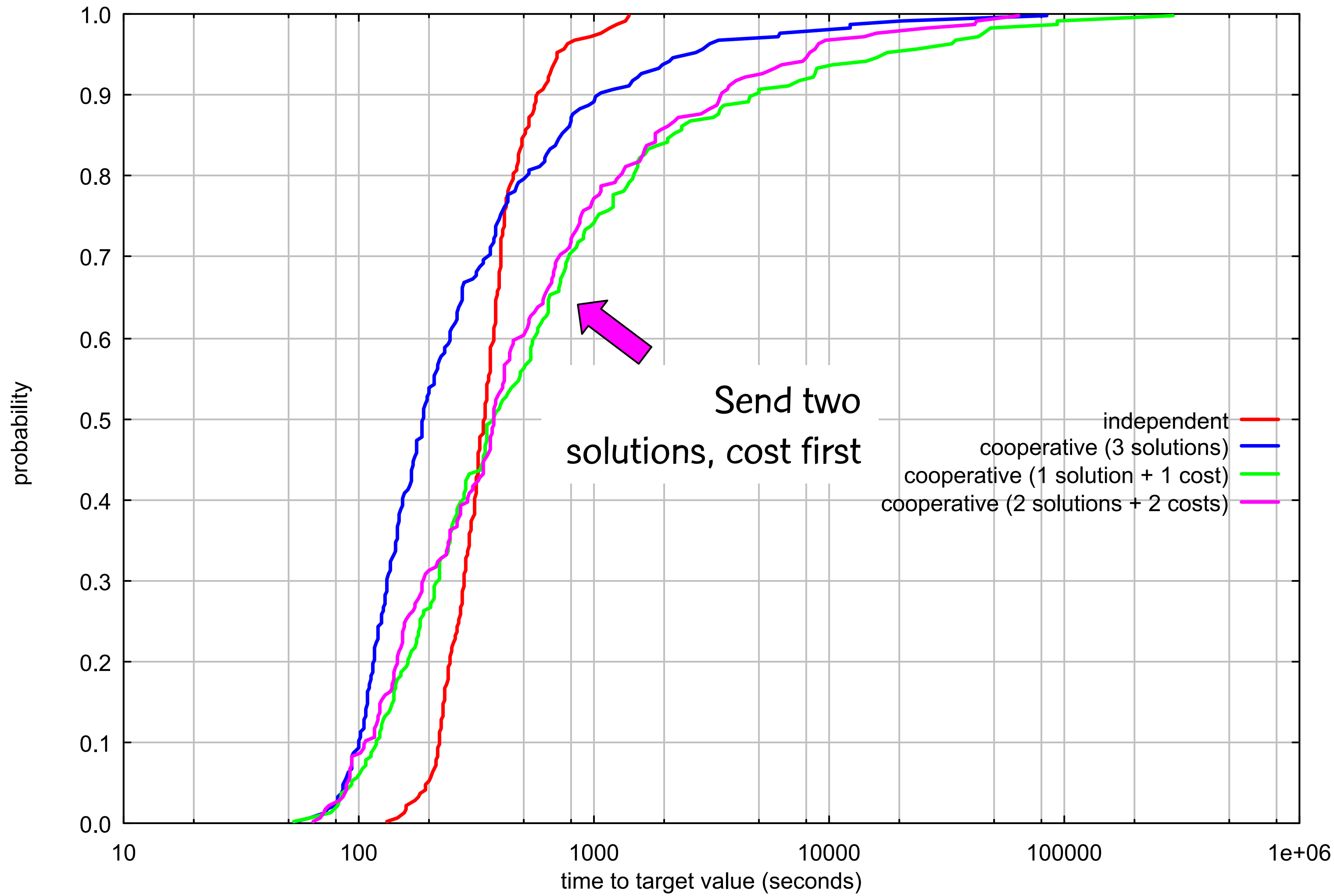
Parallel cooperative implementation

- Improved multiple-walk cooperative-thread strategy:
  - Locally keep the value of the worst elite solution (eventually outdated).
  - Only consider a solution as a candidate to be sent to the pool if its value is best than the above.
  - First send the solution value, then compare its value with worst elite value in the pool, next send the solution itself only if its value is better.
  - Significant reductions in communications and memory requirements: smaller and fewer messages are sent!

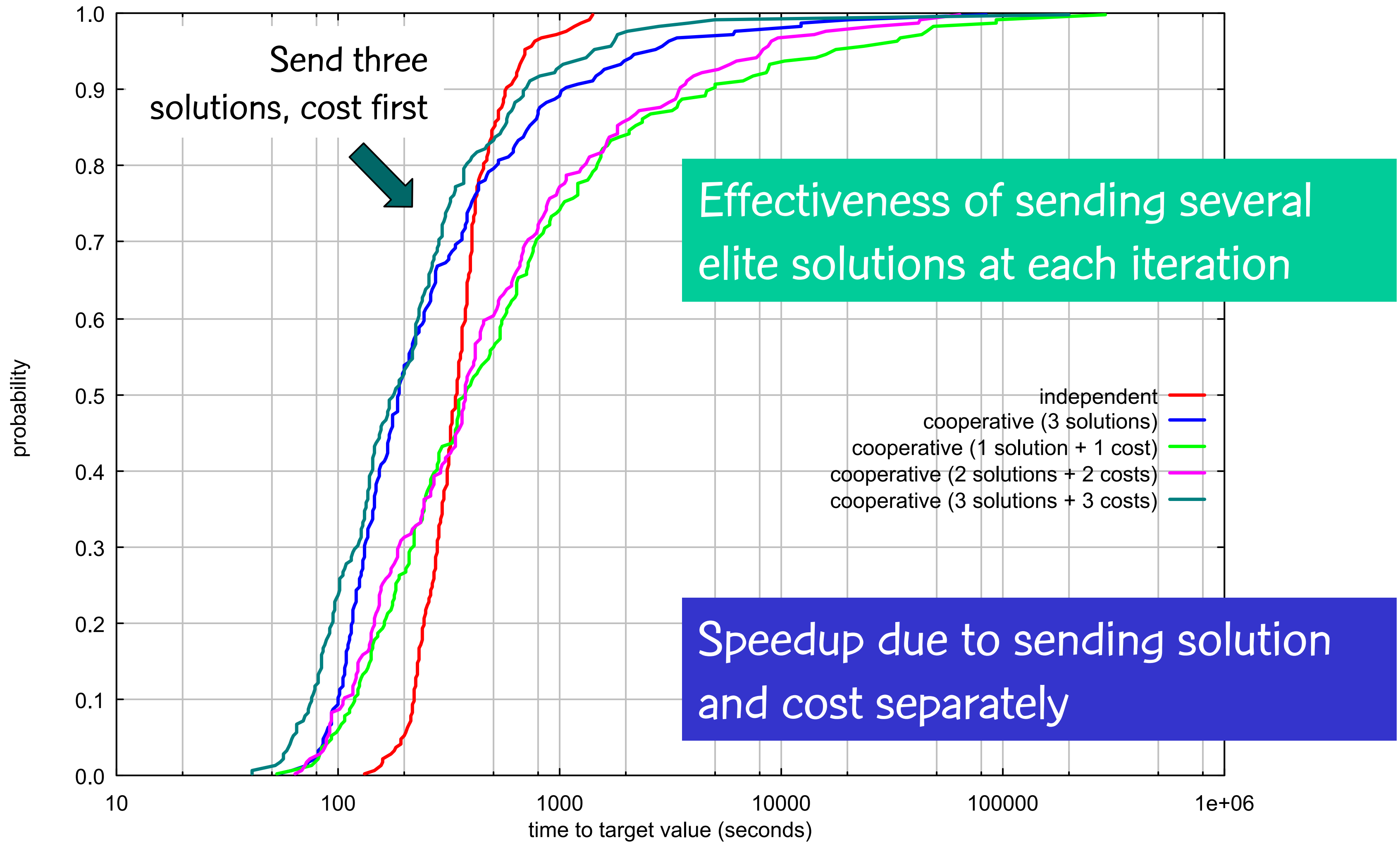
Cooperative vs. independent strategies



Cooperative vs. independent strategies



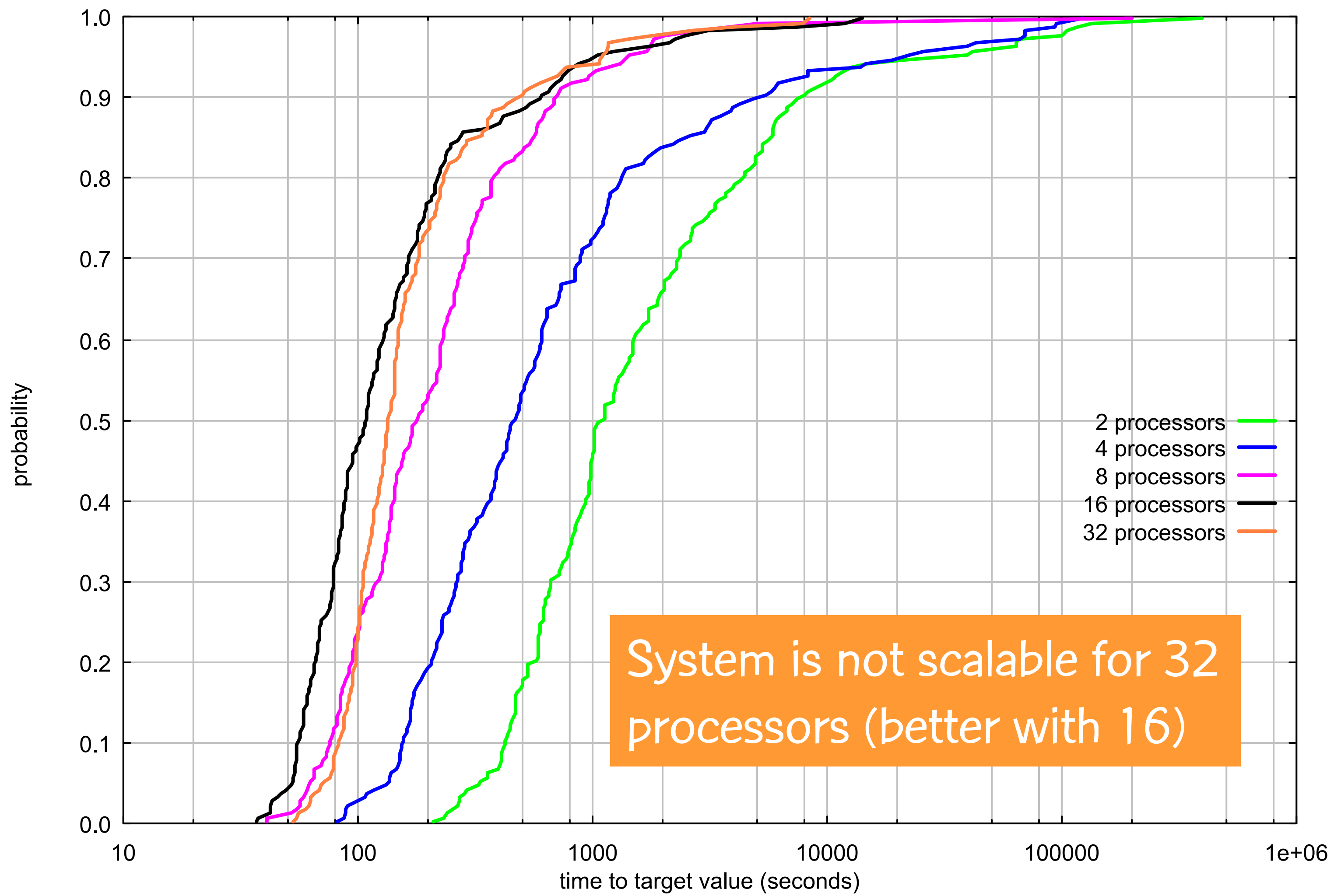
Cooperative vs. independent strategies



Parallel cooperative implementation

- Recall that when p processors are used:
  - All of them perform GRASP iterations in the independent strategy
  - Only p-1 processors perform GRASP iterations in the cooperative strategy
- Cooperative strategy improves w.r.t. the independent strategy when the number of processors increases.
- Cooperative strategy is already better for $p \geq 4$ processors.

Parallel cooperative implementation

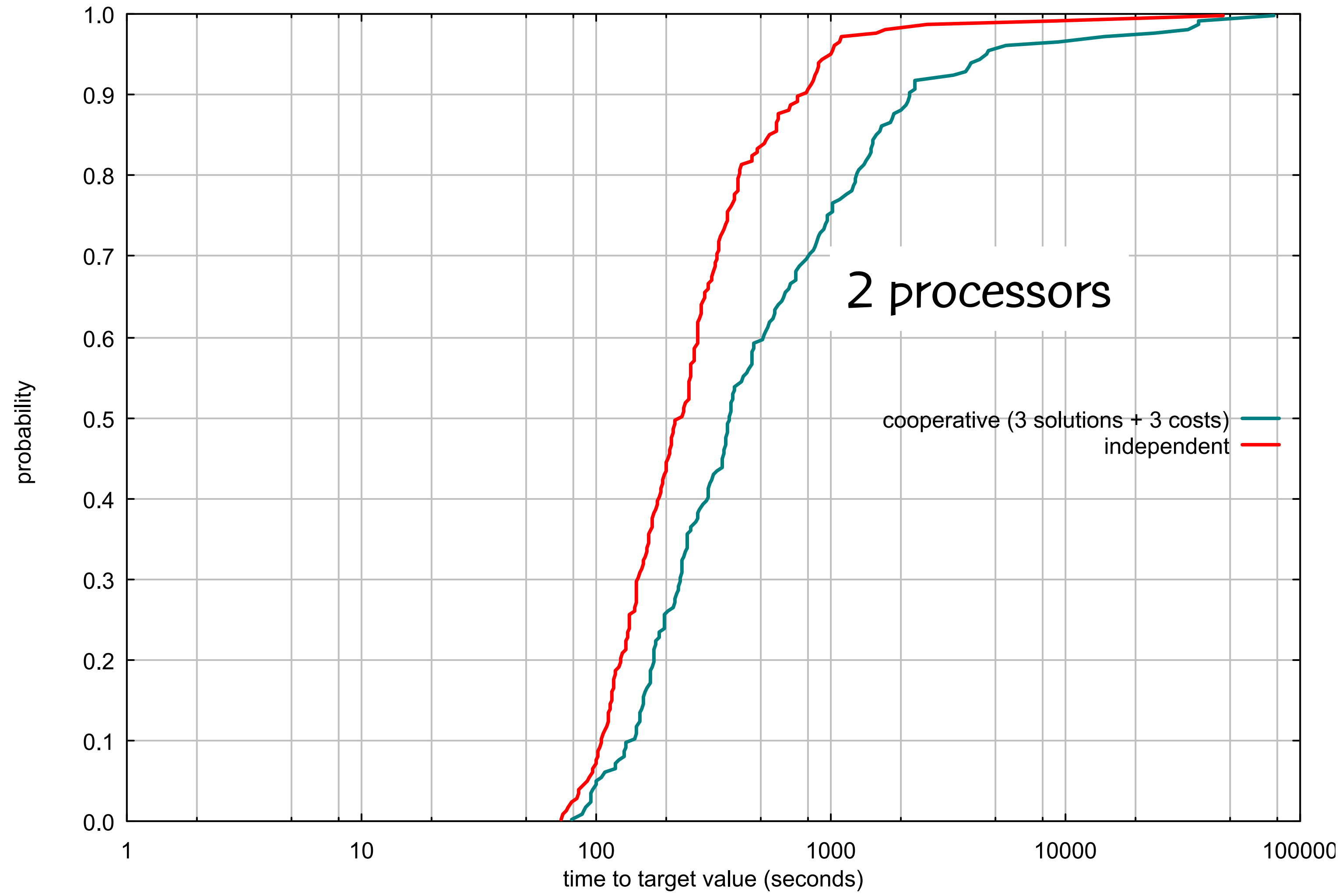


System is not scalable for 32 processors (better with 16)

Improved parallel environment

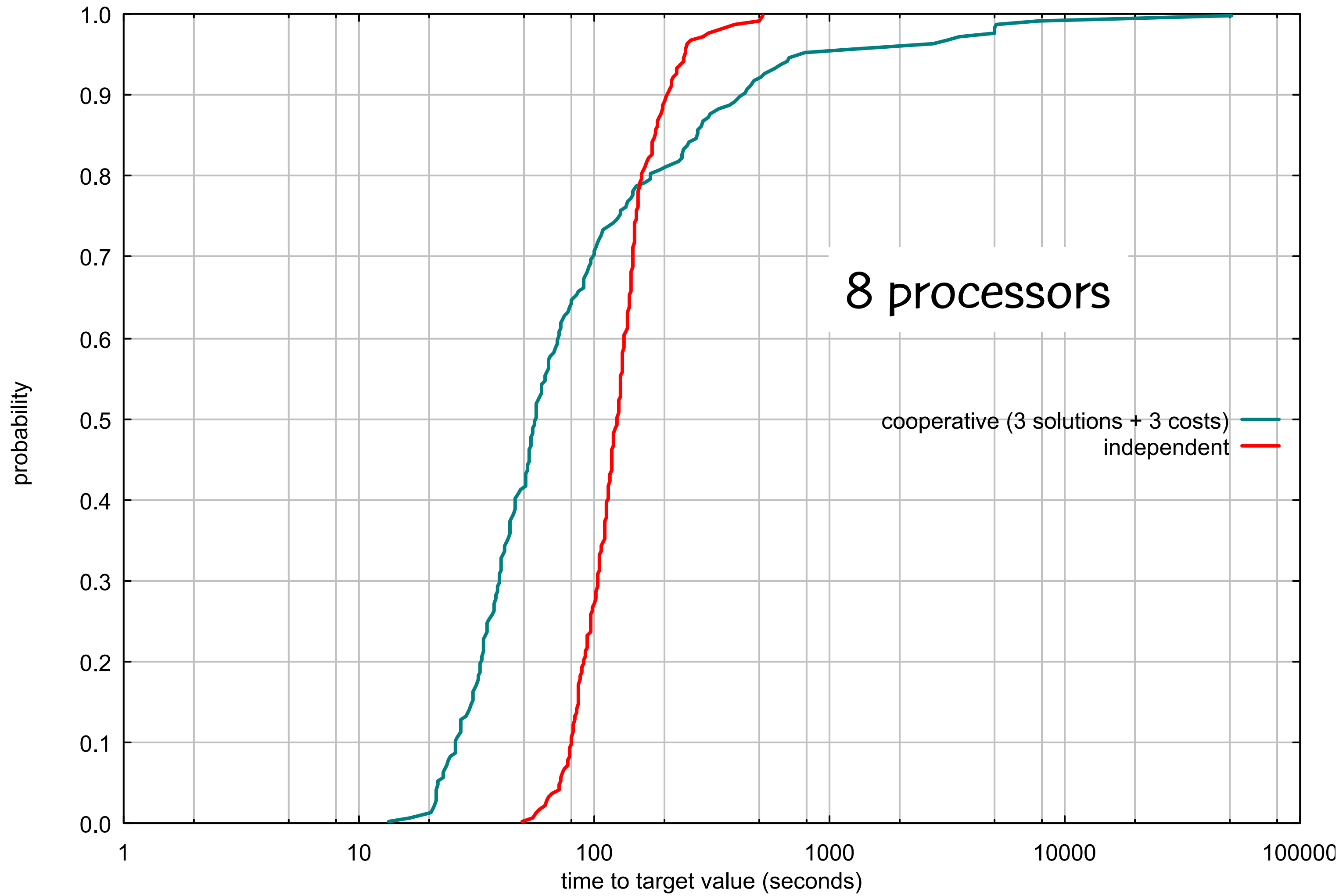
- Linux cluster with 32 Pentium IV 1.7 GHz processors with 256 Mbytes of RAM each
- Extreme Networks switch with 48 10/100 Mbits/s ports and two 1 Gbits/s ports

Improved parallel environment

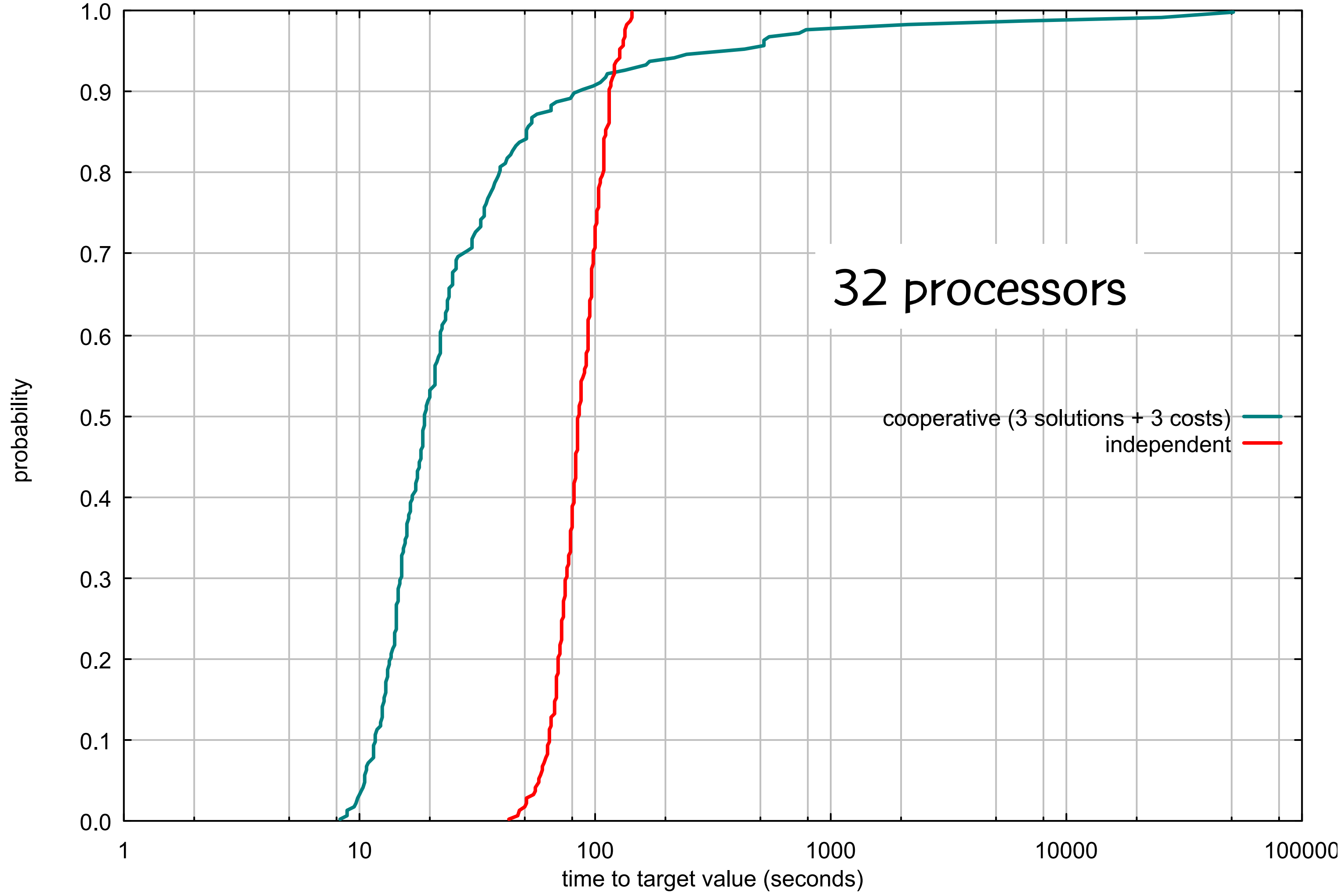


40'

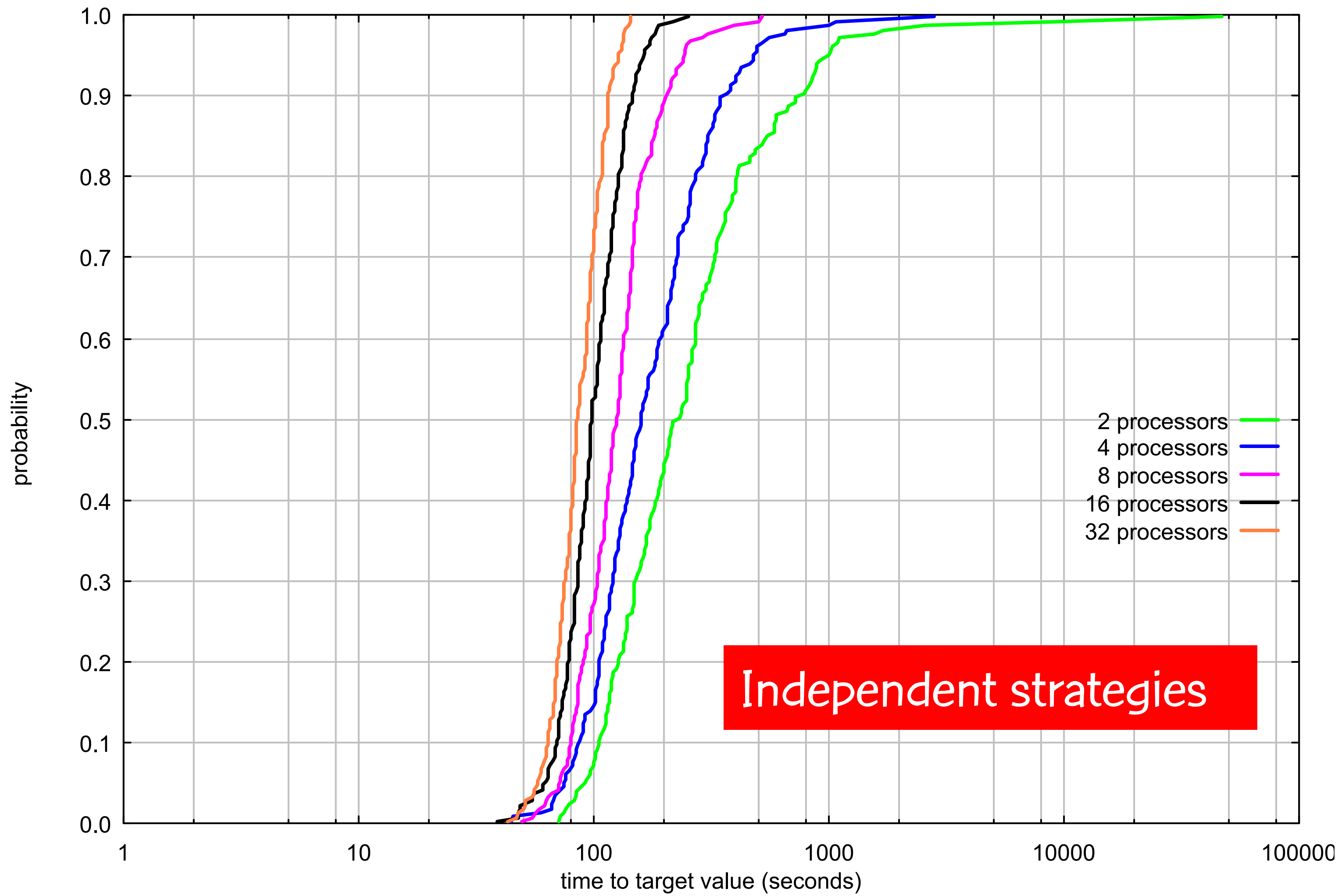
Improved parallel environment



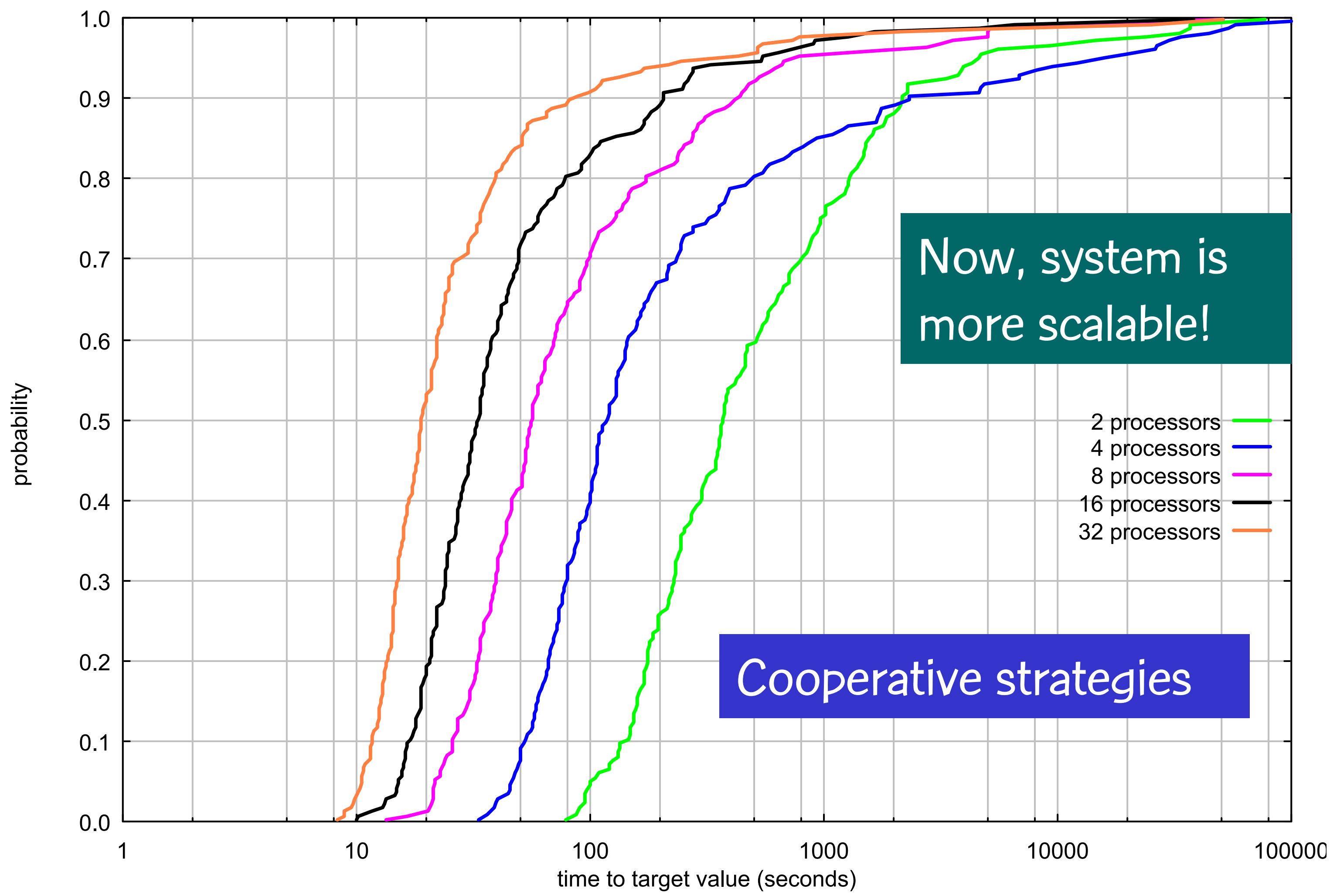
Improved parallel environment



Improved parallel environment



Improved parallel environment



Job shop scheduling

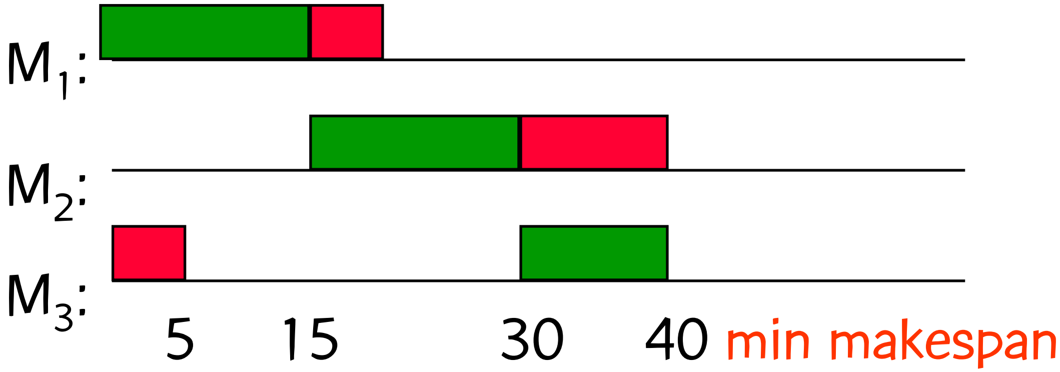
Schedule a set of jobs on a set of machines, such that

- each job has a specified processing order on the set of machines
- machines can process only one job at a time
- each job has a specified duration on each machine
- machine must finish processing job before it can begin processing another job (no preemption allowed)

minimizing makespan.

J_1: $M_1(15)$, $M_2(15)$,$M_3(10)$

J_2: $M_3(5)$, $M_1(5)$, $M_2(10)$



Job shop scheduling

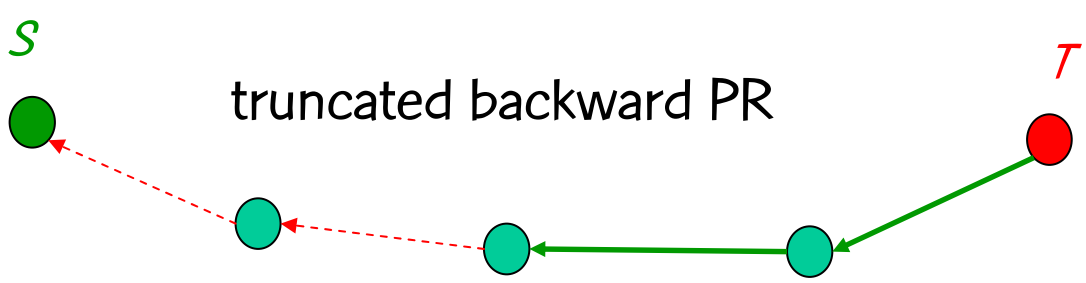
- Construction: solution is built by scheduling all operations, one at a time, biased by greedy function (makespan or job time remaining).
- Local search: on standard disjunctive graph representation of job shop schedule

  Roy & Sussmann (1964)

  Binato, Hery, Loewenstern, & Resende (2001)

Job shop scheduling

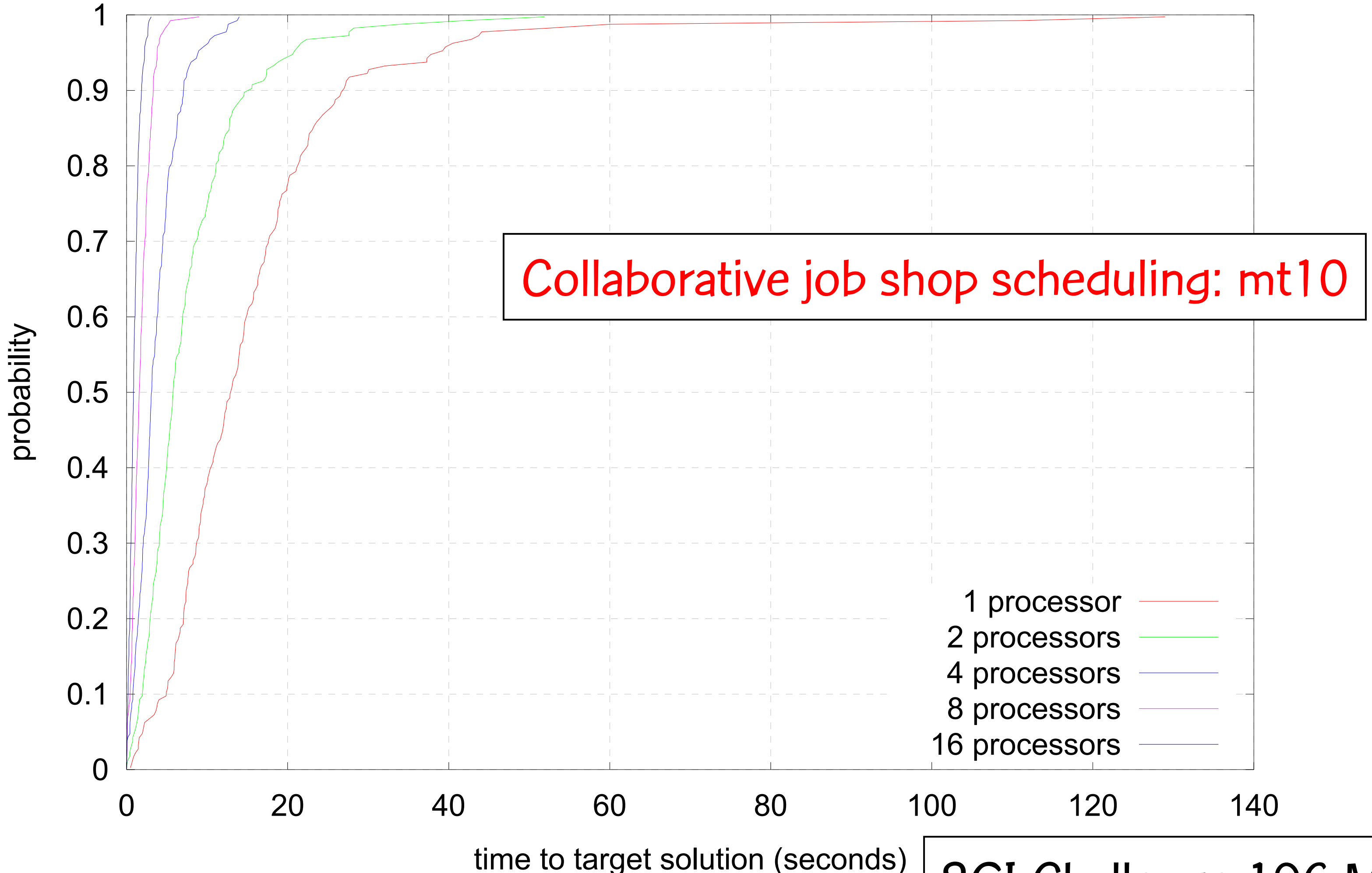
- Path-relinking between m permutation arrays, similar to PR for 3-index assignment (2 permutation arrays)
- Computing path-relinking is much more expensive than computing GRASP component:
  - Limit to backward path-relinking
  - Truncated path relinking



Job shop scheduling

SGI Challenge 196 MHz

Job shop scheduling



SGI Challenge 196 MHz

Job shop scheduling

SGI Challenge 196 MHz

Summary

- Basic algorithm
- Construction phase
- Enhanced construction strategies
- Local search
- Path-relinking
- GRASP with path-relinking
- Variants of GRASP with path-relinking
- Parallel implementations
- Applications and numerical results
- Concluding remarks

Concluding remarks (1/3)

- Path-relinking adds memory and intensification mechanisms to GRASP, systematically contributing to improve solution quality:
  - better solutions in smaller times
  - some implementation strategies appear to be more effective than others.
  - mixed path-relinking strategy is very promising
  - backward relinking is usually more effective than forward
  - bidirectional relinking does not necessarily pay off the additional computation time

Concluding remarks (2/3)

- Difficulties:
  - How to deal with infeasibilities along the relinking procedure?
  - How to apply path-relinking in "partitioning" problems such as graph-coloring, bin packing and others?
- Other applications of path-relinking:
  - VNS+PR: Festa, Pardalos, Resende, & Ribeiro (2002)
  - PR as a generalized optimized crossover in genetic algorithms: Ribeiro & Vianna (2003)

Concluding remarks (3/3)

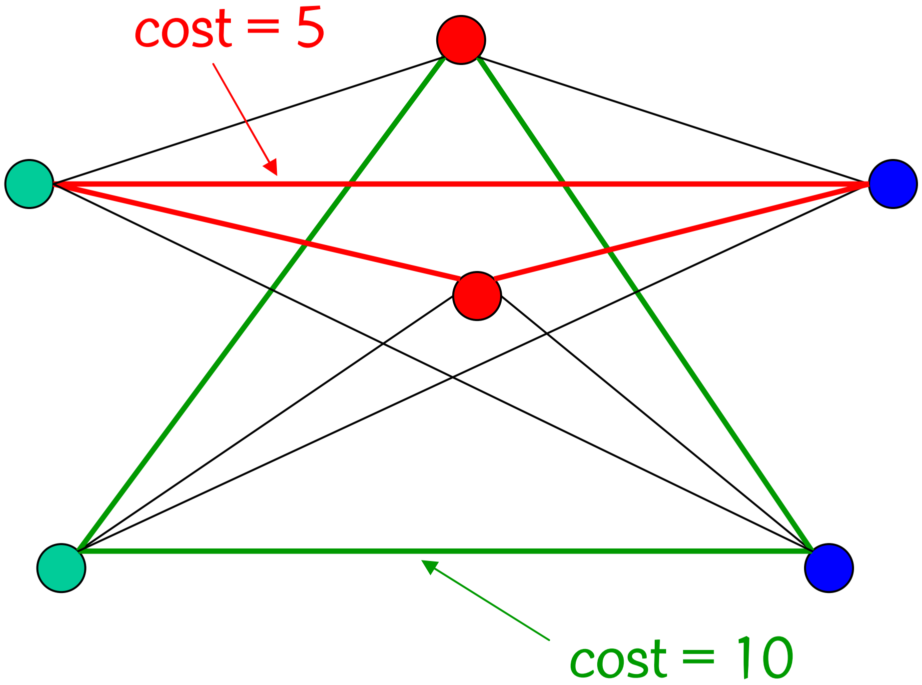
- Cooperative parallel strategies based on path-relinking:
  - Path-relinking offers a nice strategy to introduce memory and cooperation in parallel implementations.
  - Cooperative strategy performs better due to smaller number of iterations and to inter-processor cooperation.
  - Linear speedups with the parallel implementation.
  - Robustness: cooperative strategy is faster and better.
  - Parallel systems are not easily scalable, parallel strategies require careful implementations.

Slides, publications, and acknowledgements

- Slides of this talk can be downloaded from: http://www.inf.puc-rio/~celso/talks
- Papers about GRASP, path-relinking, and their applications available at:
  http://www.inf.puc-rio.br/~celso/publicacoes
  http://www.research.att.com/~mgcr
  http://graspheuristic.org
- Joint work done with several M.Sc. and Ph.D. students from PUC-Rio, who are all gratefully acknowledged: S. Canuto, M. Souza, M. Prais, S. Martins, D. Vianna, R. Aiex, R. Werneck, E. Uchoa, and I. Rosseti.

```
procedure GRASP+PR_2PNDP;
    f* ← ∞;
    Pool ← ∅;
    for k = 1, ..., Max_Iterations do;
        Construct a randomized solution x (construction phase);
        Find y by applying local search to x (local search phase);
        if y satisfies the membership conditions then insert y into Pool;
        Randomly select a solution z ∈ Pool (z ≠ y) with uniform probability;
        if f(z) > f(y) then exchange y and z;
        Compute Δ(z, y);
        Let ȳ be the best solution found by applying path-relinking to (z, y);
        if ȳ satisfies the membership conditions then insert ȳ into Pool;
        if f(ȳ) < f* then do;
            x* ← ȳ;
            f* ← c(ȳ);
        end if;
    end for;
    return x*;
end GRASP+PR_2PNDP;
```

3-index assignment (AP3)



Complete tripartite graph: Each triangle made up of three distinctly colored nodes has a cost.

AP3: Find a set of triangles such that each node appears in exactly one triangle and the sum of the costs of the triangles is minimized.

3-index assignment (AP3)

- Construction: Solution is built by selecting n triplets, one at a time, biased by triplet costs.
- Local search: Explores $O(n^2)$ size neighborhood of current solution, moving to better solution if one is found

Aiex, Pardalos, Resende, & Toraldo (2003)

3-index assignment (AP3)

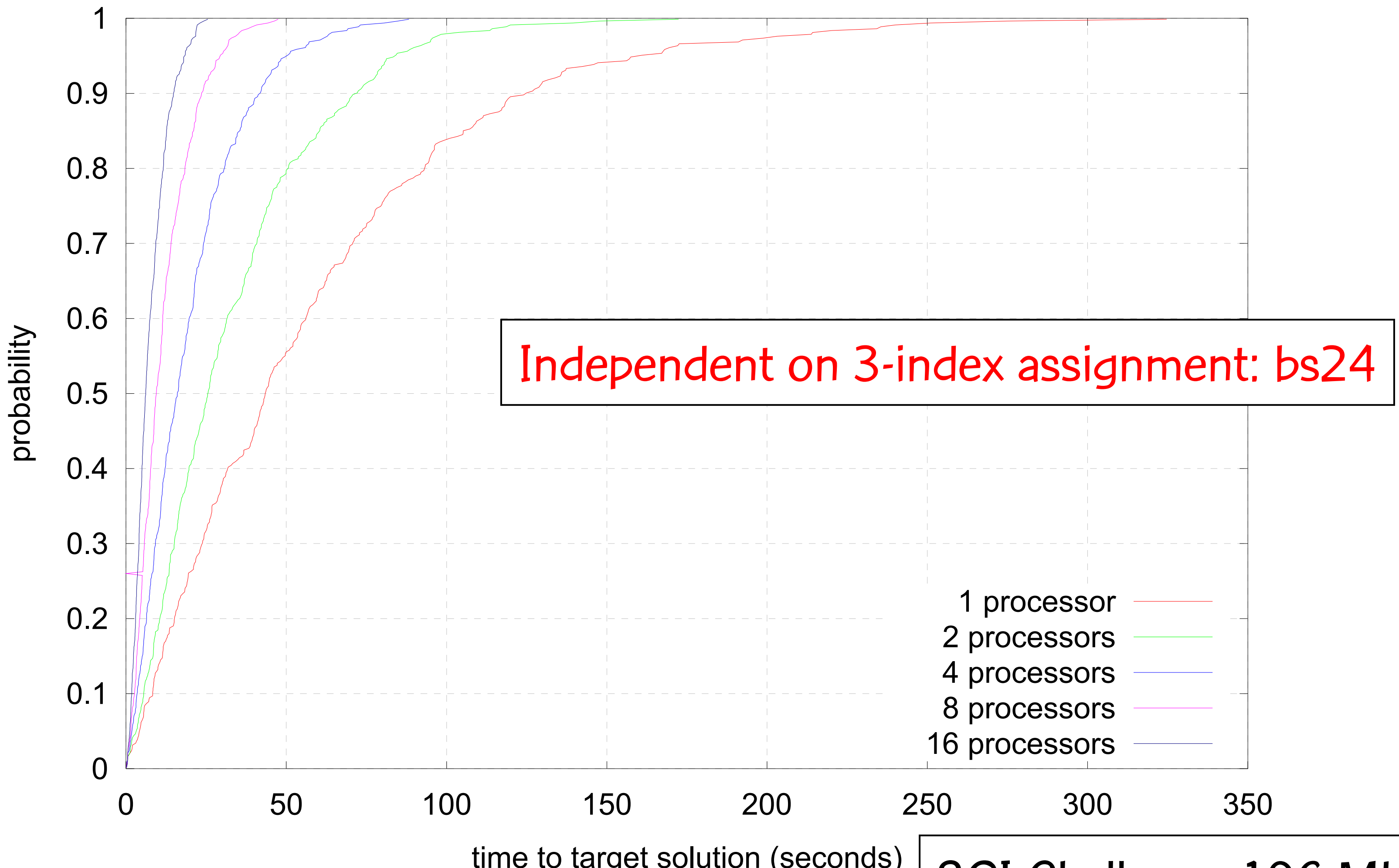
- Path relinking is done between:
  - Initial solution

$$S = \{ (1, j_1^S, k_1^S), (2, j_2^S, k_2^S), \ldots, (n, j_n^S, k_n^S) \}$$

  - Guiding solution

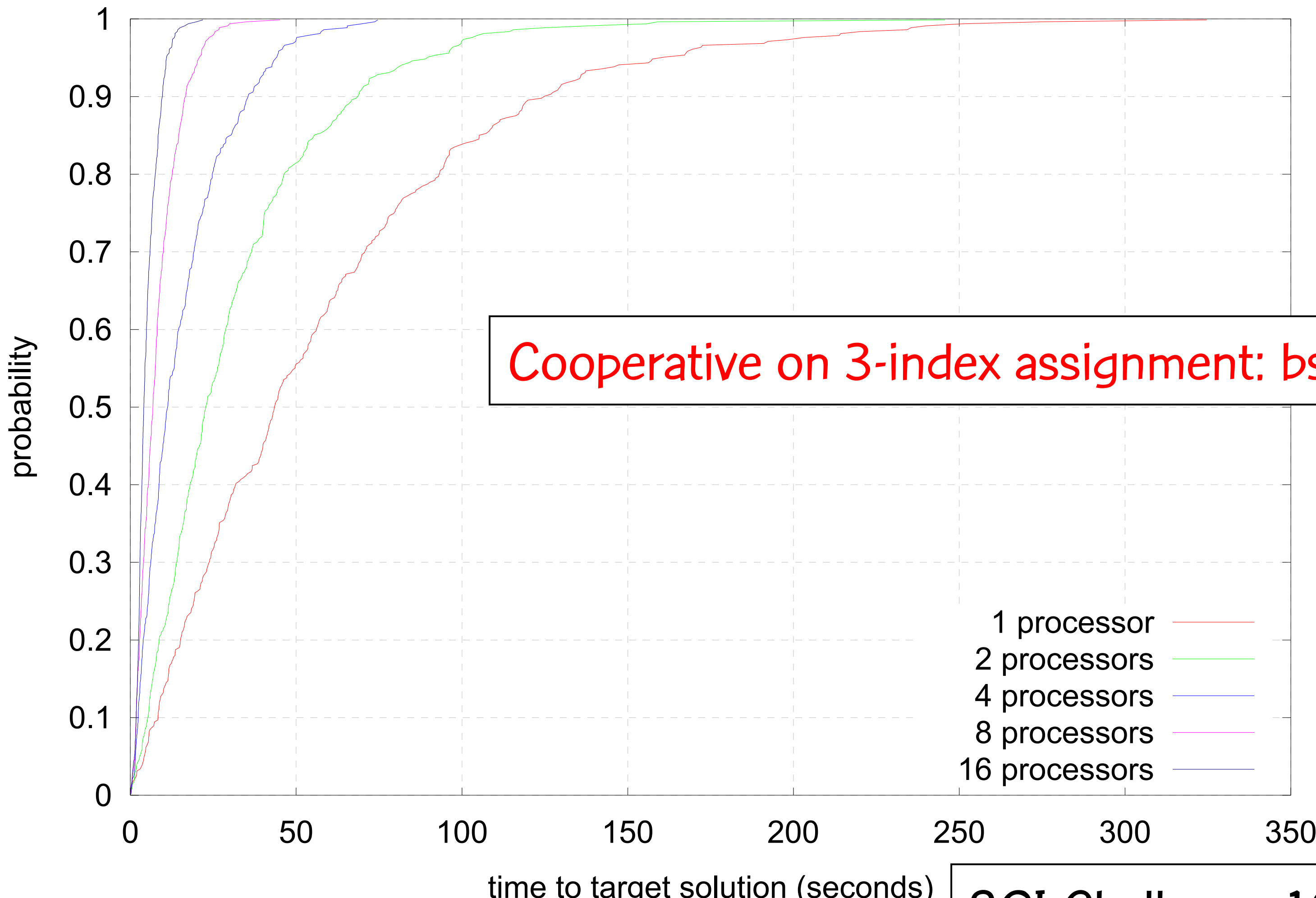
$$T = \{ (1, j_1^T, k_1^T), (2, j_2^T, k_2^T), \ldots, (n, j_n^T, k_n^T) \}$$

3-index assignment (AP3)



SGI Challenge 196 MHz

3-index assignment (AP3)



Cooperative on 3-index assignment: bs24

SGI Challenge 196 MHz

3-index assignment (AP3)

SGI Challenge 196 MHz

45'