On the use of run time distributions to evaluate and compare stochastic local search algorithms

Celso C. Ribeiro (Brazil)
Isabel Rosseti (Brazil)
Reinaldo Vallejos (Chile)

SLS'2009 - Brussels
September 2009

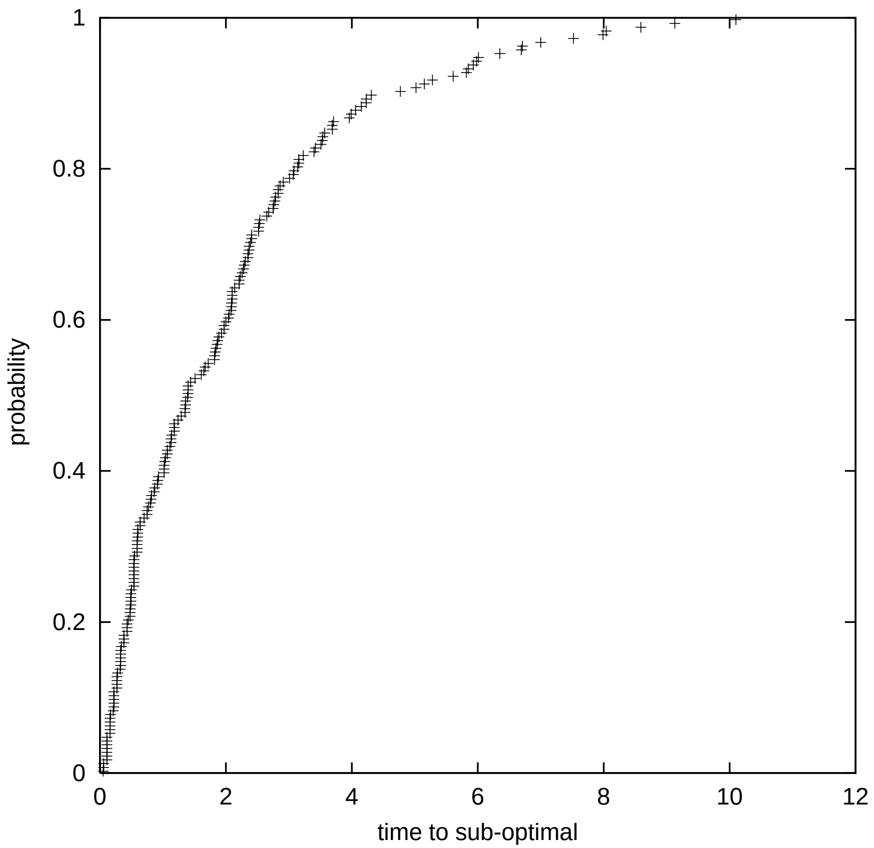
Summary

- Run time distributions
- Algorithms with exponential run time distributions
  - Closed-form result
  - Applications
  - Examples of non-exponential run time distributions
- Algorithms with non-exponential run time distributions
- Case studies
- Parallel implementations
- Convergence and sensitivity
- Concluding remarks

Run time distributions

- Run time distributions or time-to-target plots display the probability that an algorithm will find a solution at least as good as a given target value within a given running time:
  - Useful tool to characterize the running times of stochastic local search (SLS) algorithms.
- Experimental results show that random variable time-to-target-value fits an exponential (or shifted exponential) distribution for a number of SLS-based metaheuristics (SA, TS, ILS, GRASP, etc.).

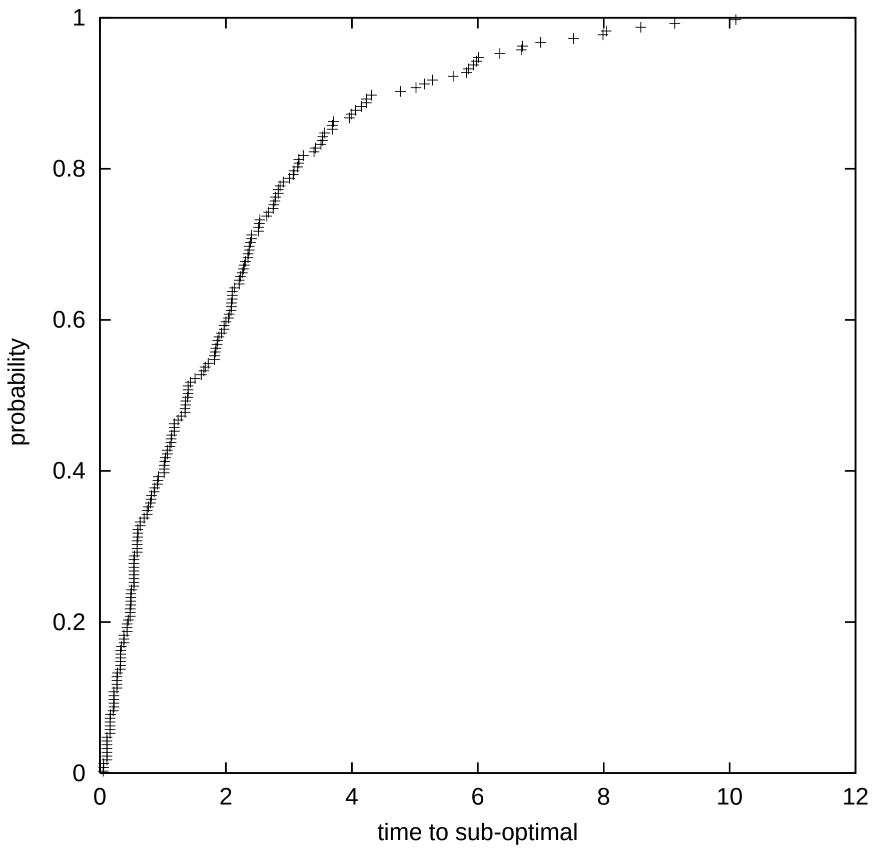
Run time distributions



- Define problem instance and target value.
- N runs: stop when solution as good as target value is found.
- Sort times in ascending order.
- Plot i-th time t_i against probability p_i=i/N.

Cumulative probability distribution plot of the running times =

Run time distributions

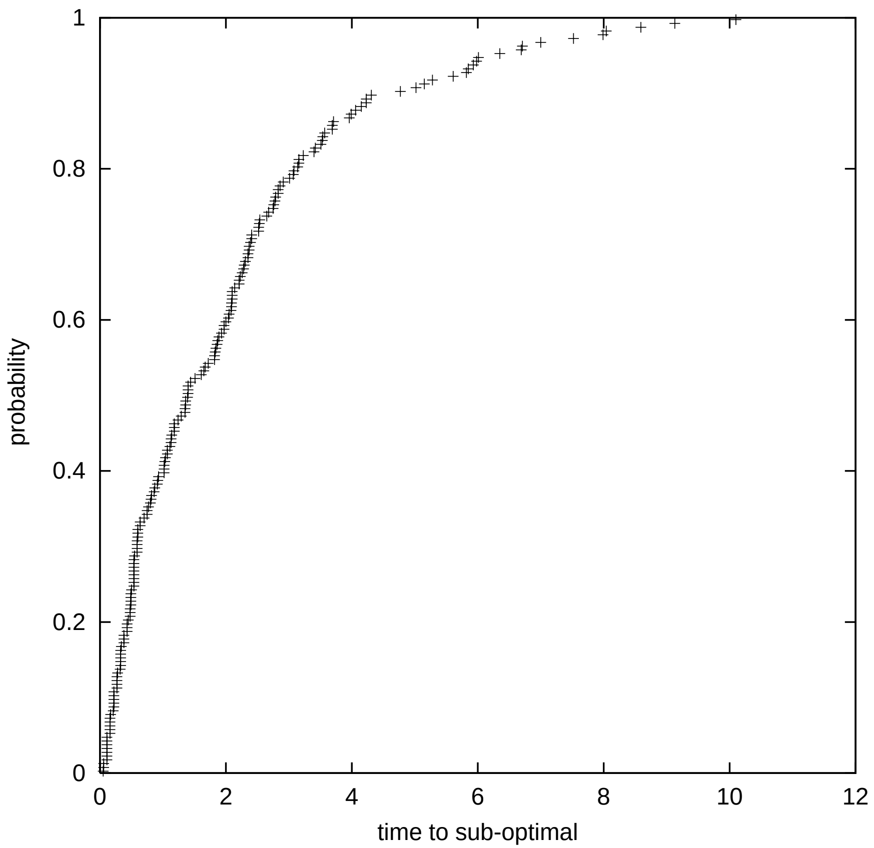


- Define problem instance and target value.
- N runs: stop when solution as good as target value is found.
- Sort times in ascending order.
- Plot i-th time t_i against probability p_i=i/N.

= Run time distribution =

Hoos and Stutzle, Art. Intel. 112 (1999)

Run time distributions

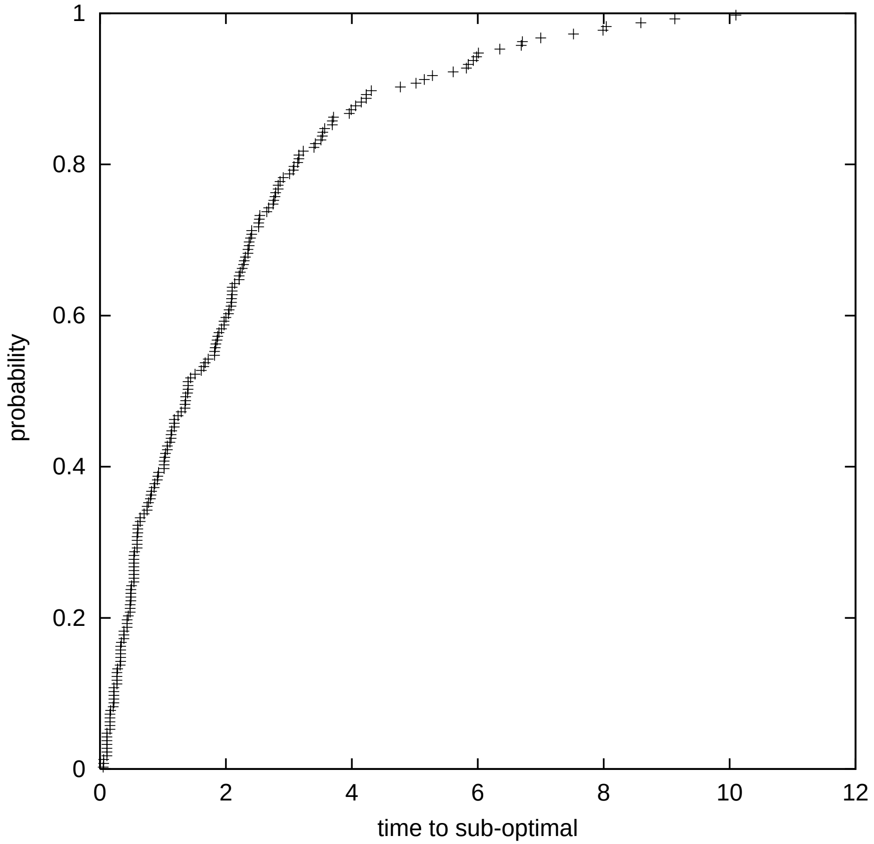


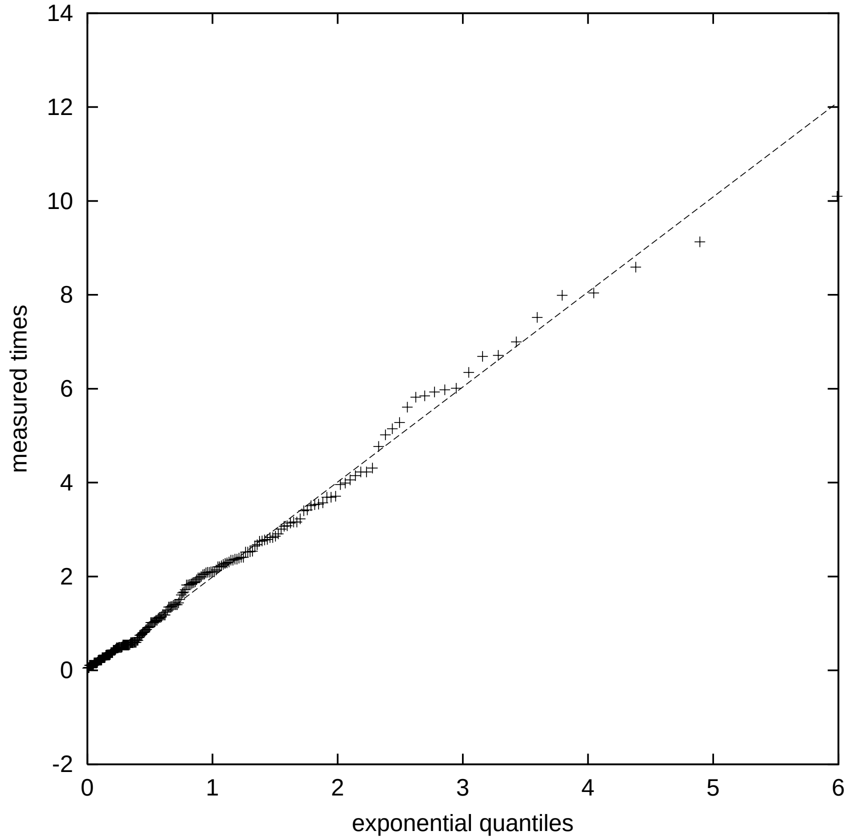
- Define problem instance and target value.
- N runs: stop when solution as good as target value is found.
- Sort times in ascending order.
- Plot i-th time t_i against probability $p_i = i/N$.

= Time-to-target value plot

Aiex et al., J. of Heuristics 8 (2002)

Run time distributions

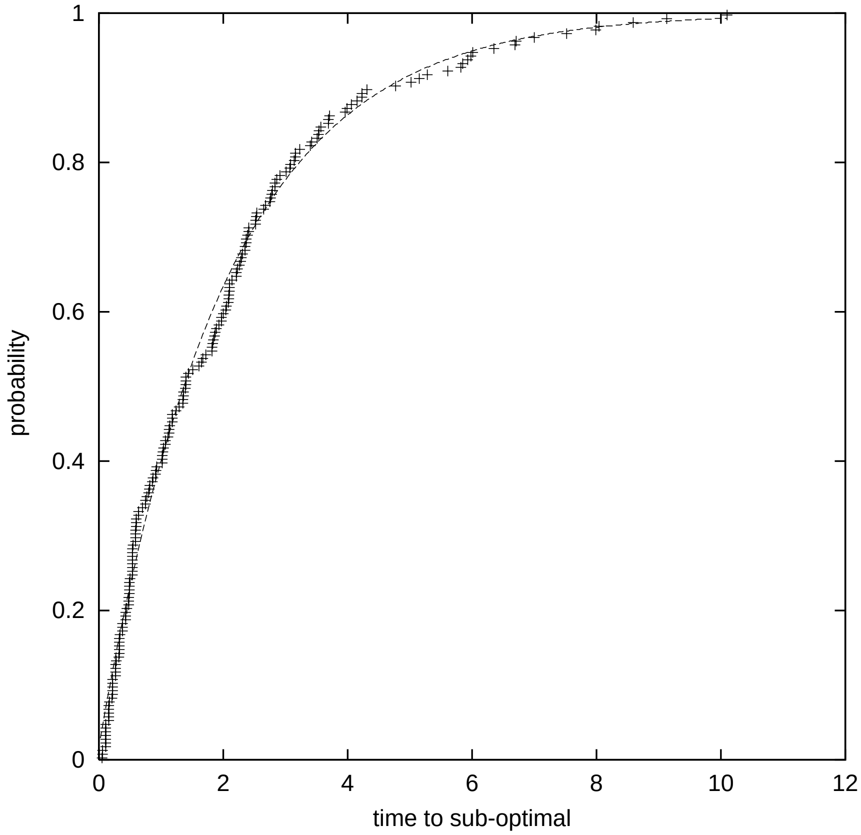


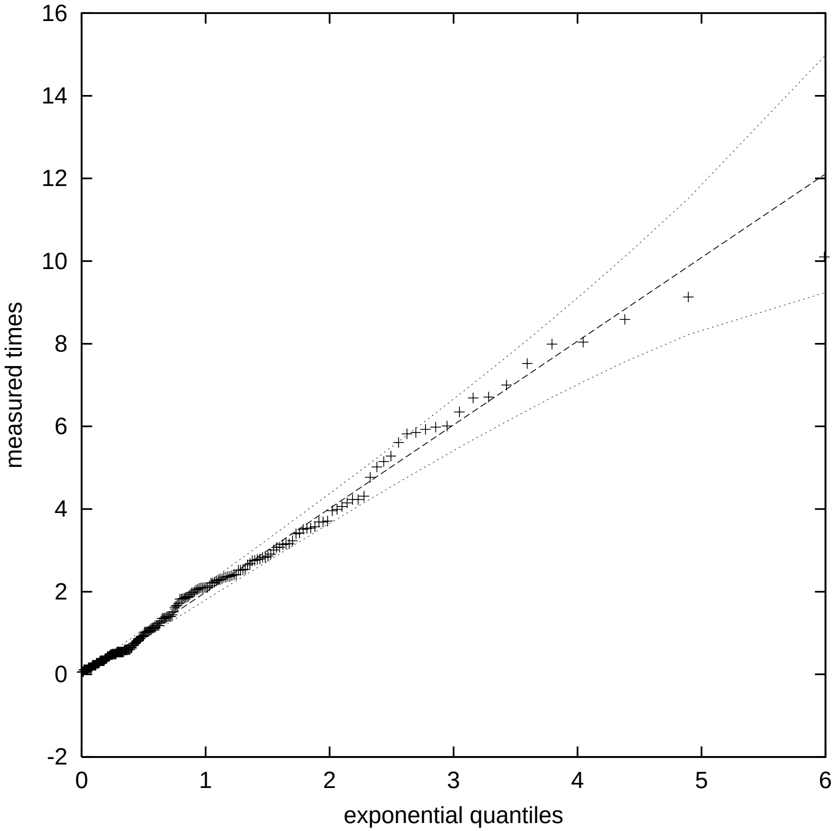


Time-to-target value plot

Quantile-quantile plot

Run time distributions





Empirical and theoretical plots

Q-Q plot with variability information

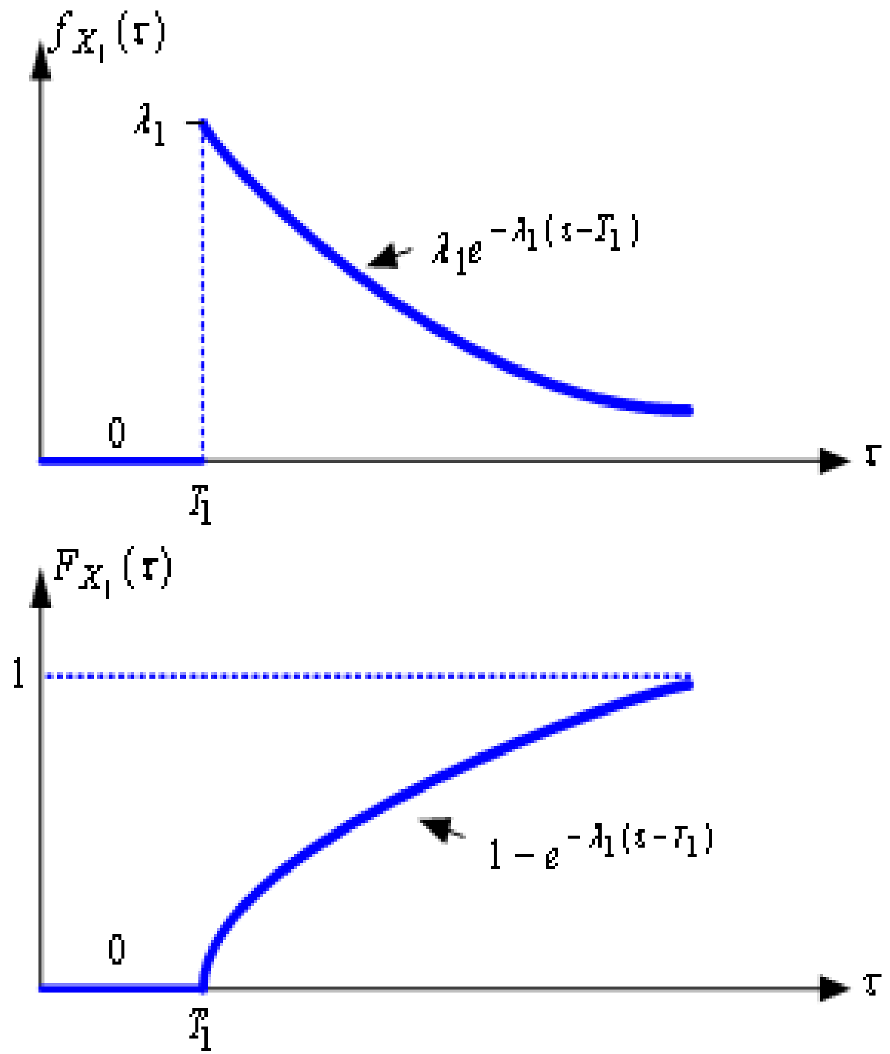
Run time distributions

- This work: new tool to compare a pair of different heuristics based on stochastic local search algorithms.
  - Applications to sequential and parallel algorithms

Exponential run time distributions

- We assume the existence of two SLS algorithms A_1 and A_2 for approximately solving some combinatorial optimization problem.
  - Running times of A_1 and A_2 fit exponential (or shifted exponential) distributions.
  - X_1 (resp. X_2): continuous random variable denoting the time needed by algorithm A_1 (resp. A_2) to find a solution as good as a given target value:

Exponential run time distributions



Exponential run time distributions

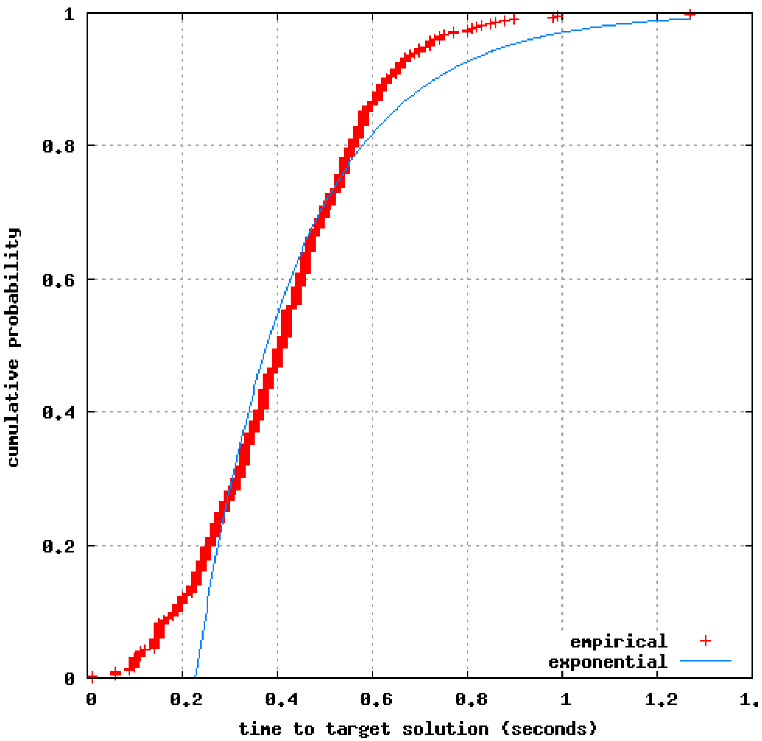
- Both algorithms A_1 and A_2 stop when they find a solution at least as good as the target value:
  - Algorithm A_1 performs better than A_2 if the former stops before the latter.
- Evaluate the probability that the random variable X_1 takes a value smaller than or equal to X_2:

$$P(X_1 \leq X_2) = \int_{-\infty}^{+\infty} P(X_1 \leq X_2 \mid X_2 = \tau) . f_{X_2}(\tau) d\tau$$

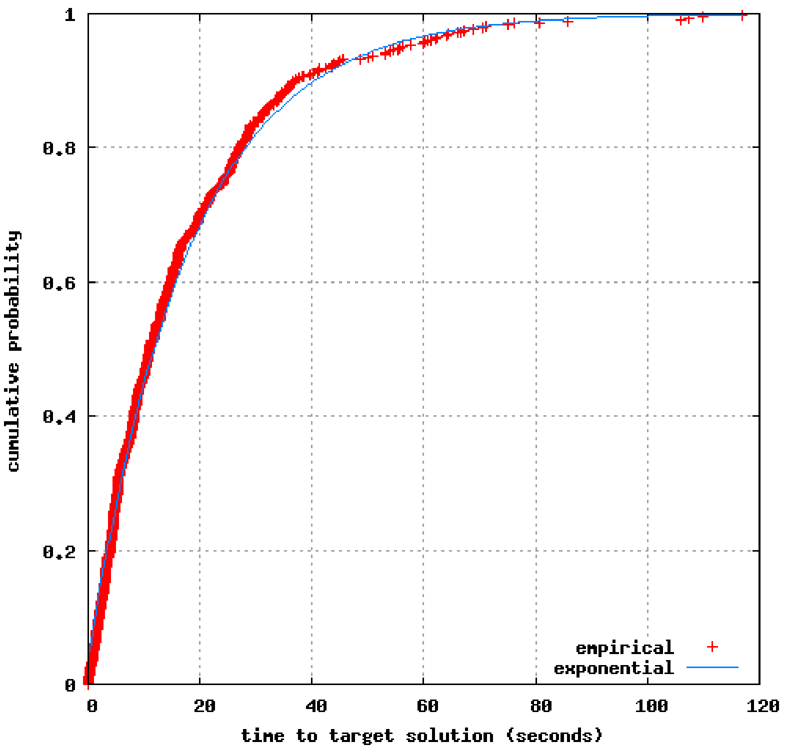
$$\boxed{P(X_1 \leq X_2) = 1 - e^{-\lambda_1 (T_2 - T_1)} \cdot \frac{\lambda_1}{\lambda_1 + \lambda_2}}$$

2-path network design problem

A_1: GRASP with bi-PR

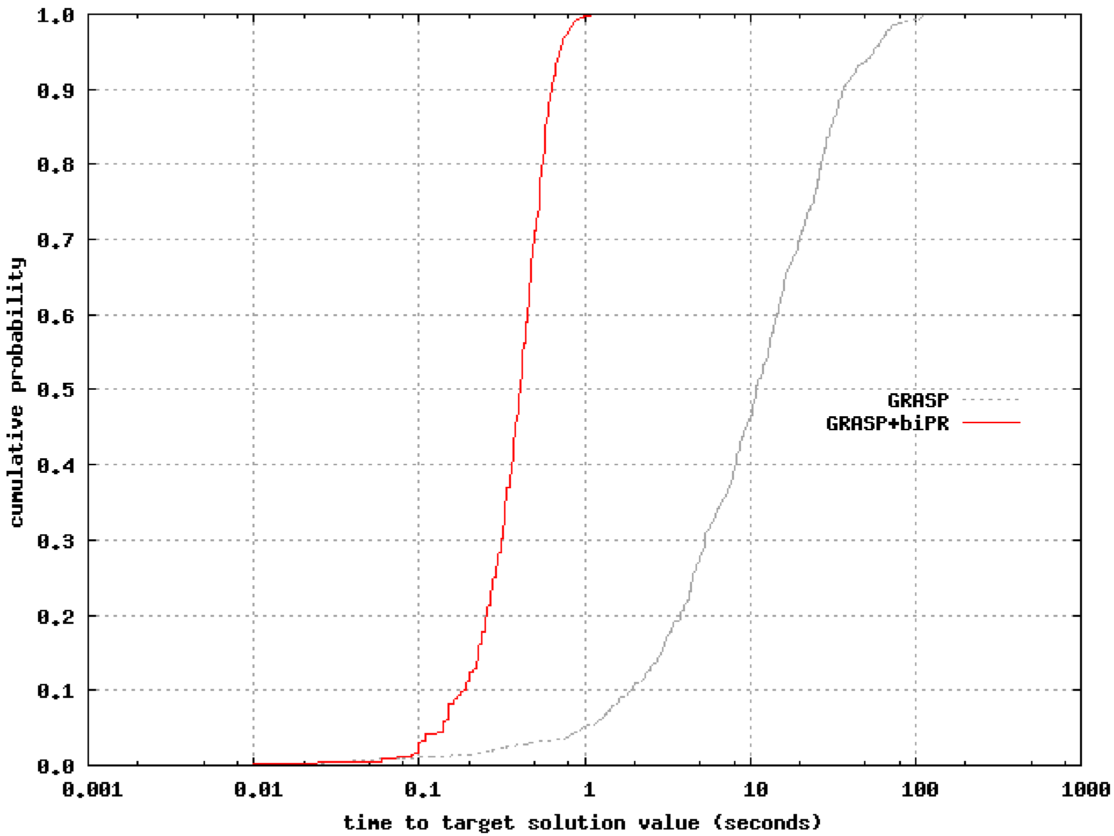


A_2: pure GRASP



500 runs: $\lambda_1 = 0.218988$, $T_1 = 0.01$, $\lambda_2 = 17.829236$, and $T_2 = 0.01$

2-path network design problem



$P(X_1 \leq X_2) = 0.943516$

TTTplot of A_1 is clearly to the left of that of A_2.

Algorithm A_1 is faster than A_2 for this instance and target.

Exponential run time distributions

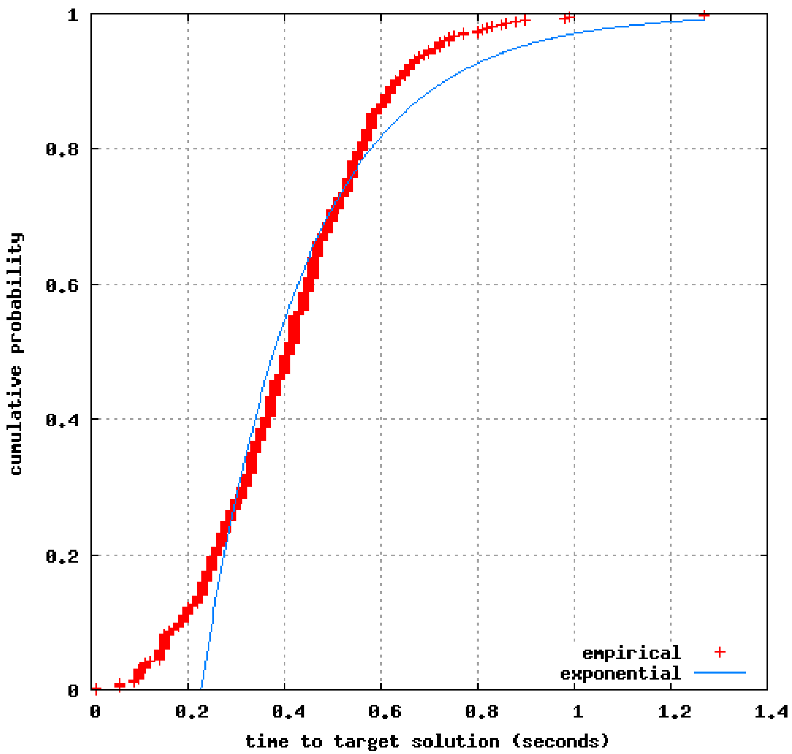
- Aiex, Resende & Ribeiro (J. of Heuristics, 2002): time taken by a GRASP heuristic to find a solution at least as good as a given target value fits an exponential distribution
  - If the setup times are not negligible: running times fit a two-parameter shifted exponential distribution.
  - Experimental result involving 2,400 runs of five problems: maximum stable set, quadratic assignment, graph planarization, maximum weighted satisfiability, and maximum covering.

Exponential run time distributions

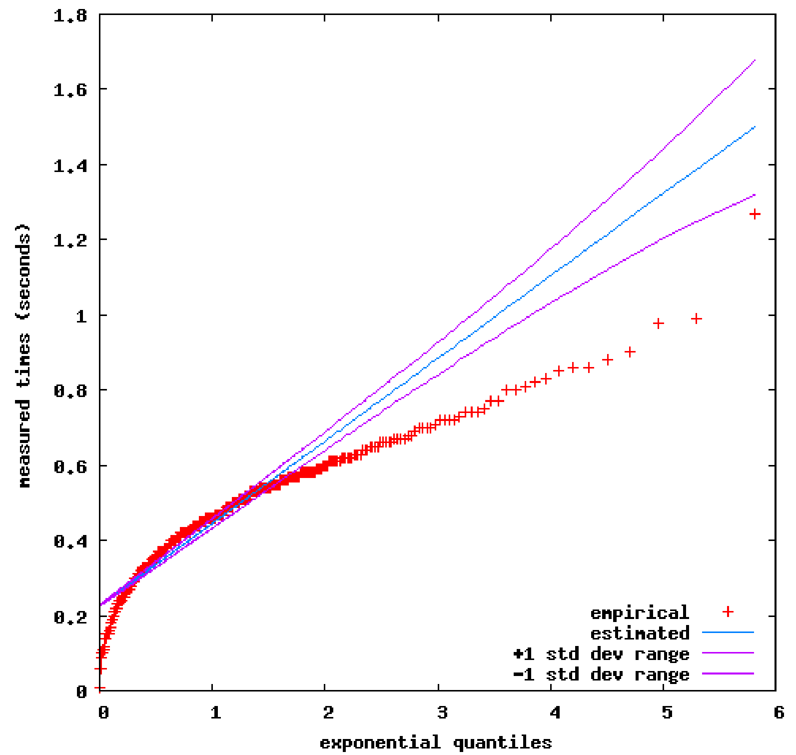
- If path-relinking is applied as an intensification step at the end of each GRASP iteration:
  - Iterations are no longer independent.
  - Memoryless characteristic of GRASP is destroyed.
- Therefore, time-to-target-value random variable may not fit an exponential distribution.
- Examples: GRASP with PR for...
  - 2-path network design problem
  - three-index assignment problem

2-path network design problem

Run-time distribution



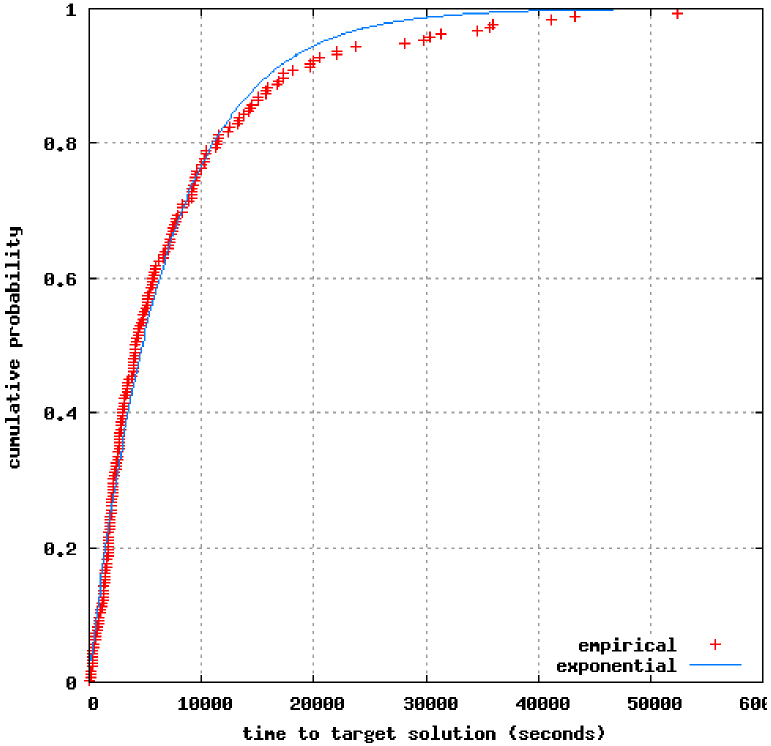
Quantile-quantile (q-q) plot



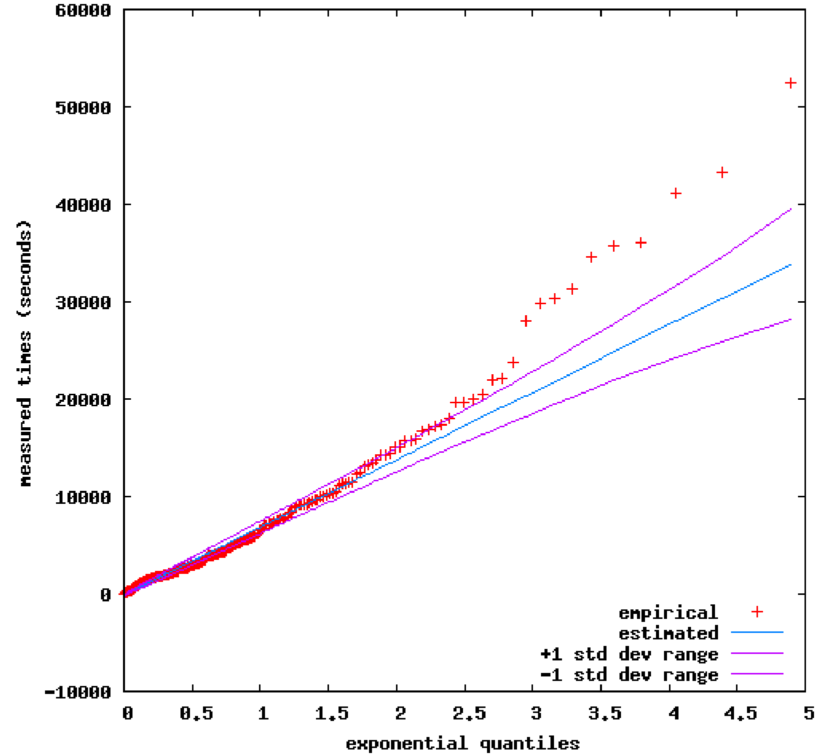
Points steadily deviate by more than one standard deviation from the estimate for the upper quantiles in the q-q plots: run time distributions are not exponential.

Three-index assignment problem

Run-time distribution



Quantile-quantile (q-q) plot



Points steadily deviate by more than one standard deviation from the estimate for the upper quantiles in the q-q plots: run time distributions are not exponential.

Exponential run time distributions

- If the running times do not fit an exponential distribution, then the previous closed-form does not hold.
  - Approach has to be extended to general run time distributions.

Non-exponential running times

- Once again, we assume the existence of two independent SLS algorithms A_1 and A_2 for the same problem.
- Time-to-target-values X_1 and X_2 are continuous random variables, with empirical cumulative probability distributions $F_{X1}(\tau)$ and $F_{X2}(\tau)$ and probability density functions $f_{X1}(\tau)$ and $f_{X2}(\tau)$:

$$P(X_1 \leq X_2) = \int_{-\infty}^{+\infty} P(X_1 \leq \tau).f_{X_2}(\tau)d\tau =$$

arbitrarily small $\varepsilon > 0$

$$= \int_{0}^{+\infty} P(X_1 \leq \tau).f_{X_2}(\tau)d\tau = \sum_{i=0}^{\infty} \int_{i=\varepsilon}^{(i+1)\varepsilon} P(X_1 \leq \tau).f_{X_2}(\tau)d\tau$$

Non-exponential running times

$$P(X_1 \le X_2) \le \sum_{i=0}^{\infty} F_{X_1}((i+1)\varepsilon) \int_{i=\varepsilon}^{(i+1)\varepsilon} f_{X_2}(\tau)d\tau = R(\varepsilon)$$

$$L(\varepsilon) = \sum_{i=0}^{\infty} F_{X_1}(i\varepsilon) \int_{i=\varepsilon}^{(i+1)\varepsilon} f_{X_2}(\tau)d\tau = R(\varepsilon) \le P(X_1 \le X_2)$$

If $\Delta(\varepsilon) = R(\varepsilon) - L(\varepsilon)$ is sufficiently small, then

$$P(X_1 \le X_2) \approx \frac{L(\varepsilon) + R(\varepsilon)}{2}.$$

In practice, probability density functions $f_{X_1}(\tau)$ and $f_{X_2}(\tau)$ are unknown.

Non-exponential running times

Let N be the number of observations of X_1 and X_2.

In the computation of $L(\varepsilon)$ and $R(\varepsilon)$, replace $f_{X_2}(\tau)$ by estimate $\hat{f}_{X_2}(\tau)$ obtained from the sample histogram.

As before, compute

$$P(X_1 \leq X_2) \approx \frac{L(\varepsilon) + R(\varepsilon)}{2}.$$

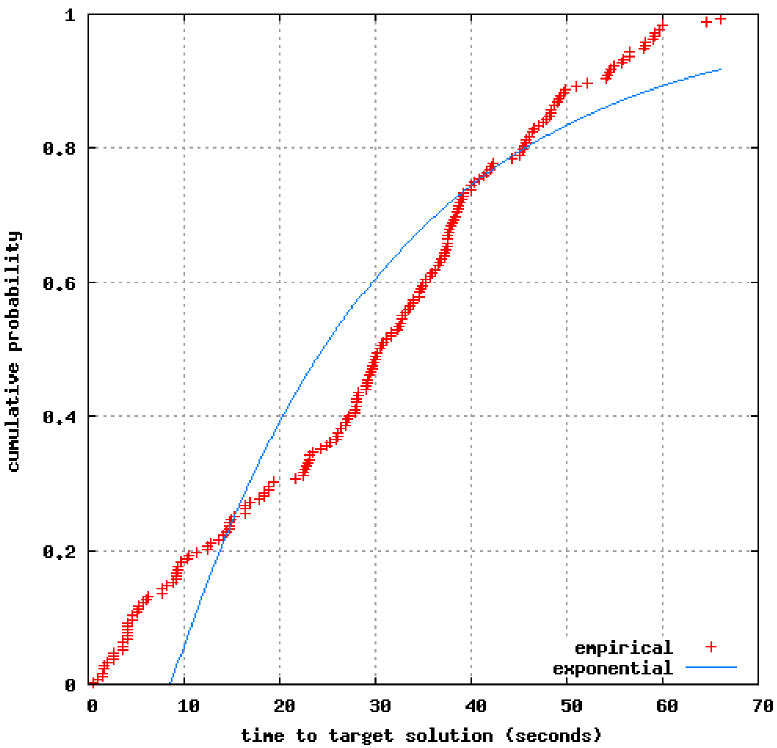
by numerical integration.

Application #1: server replication

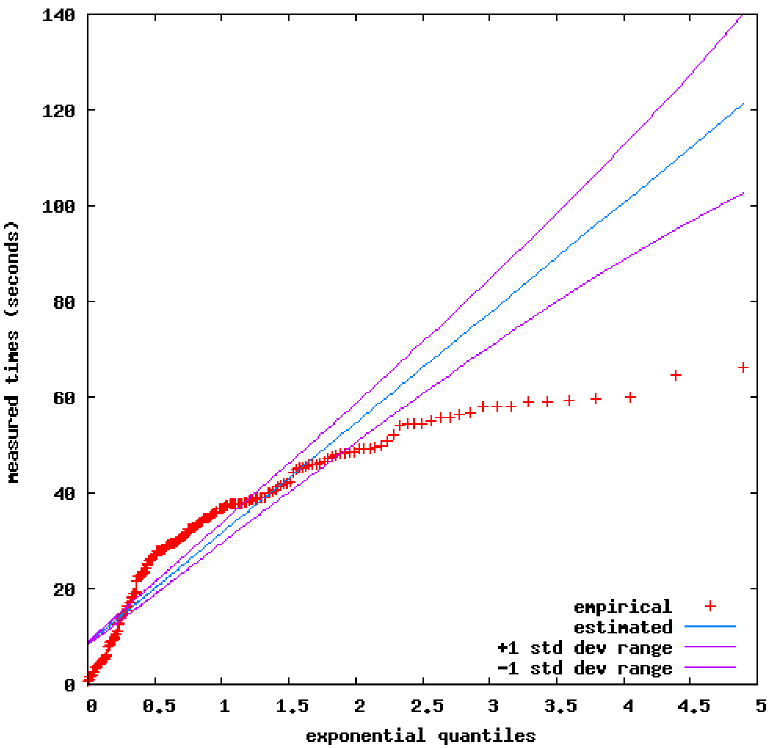
- DM-GRASP vs. pure GRASP algorithms for server replication
  - DM-GRASP: hybrid version of GRASP incorporating a data-mining process in the construction phase
  - Basic principle: mining for patterns found in good-quality solutions, to guide the construction of new solutions (similar to vocabulary building)
  - Algorithm A_1: DM-D5 version of DM-GRASP
  - Algorithm A_2: pure GRASP (exponential run time distribution)
  - Sample size: N = 200

Application #1: server replication

Run-time distribution

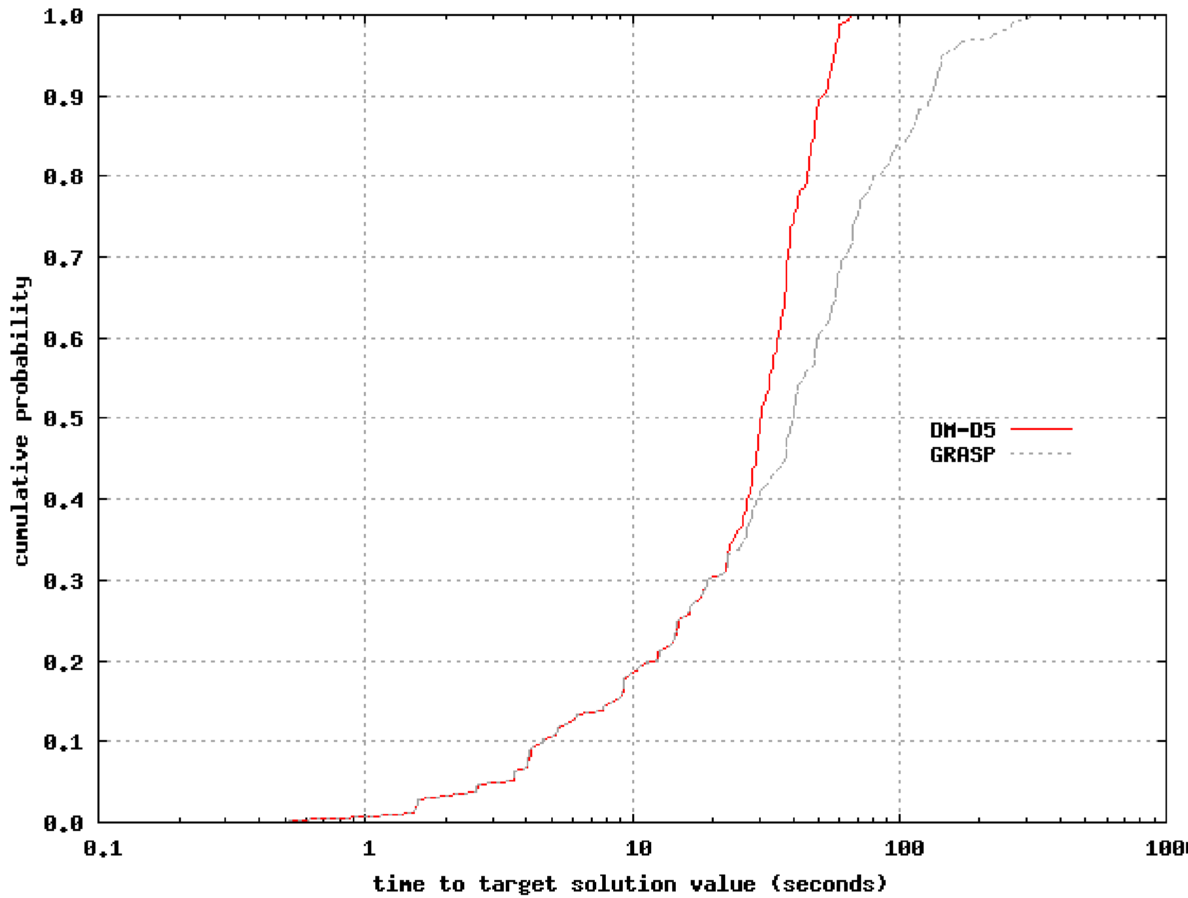


Quantile-quantile (q-q) plot



Run time distribution of DM-D5 GRASP is clearly non-exponential.

Application #1: server replication



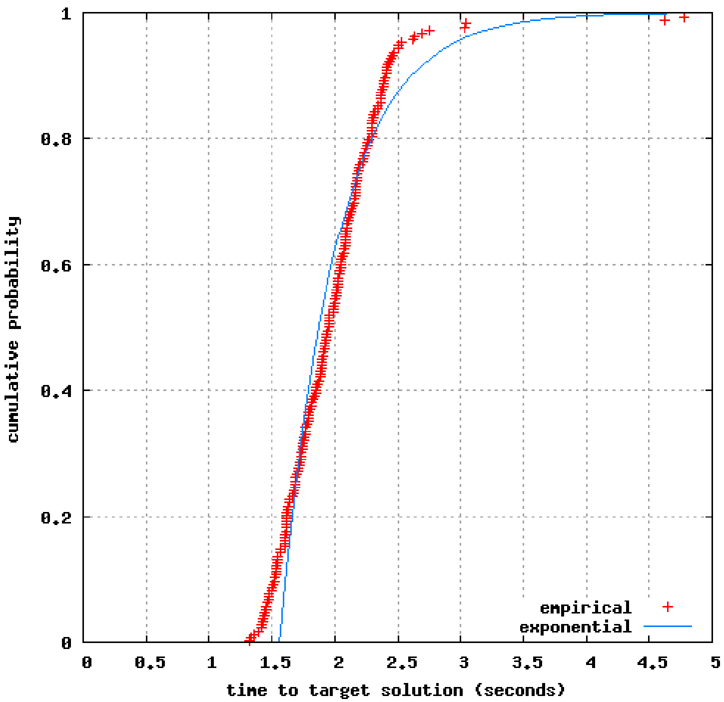
Algorithm DM-D5 outperforms GRASP: P(DM-D5 ≤ GRASP) = 0.614775.

Application #2: wavelength assignment

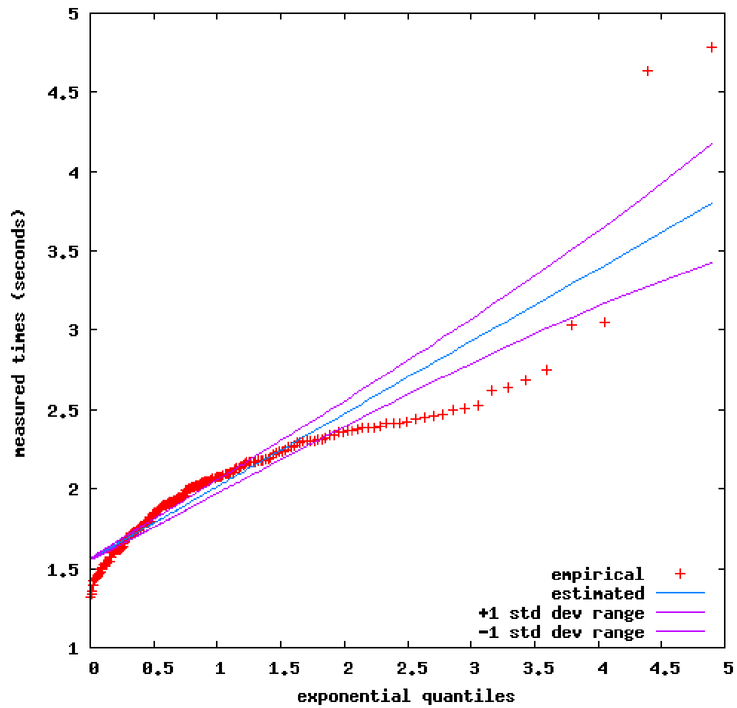
- Multistart greedy vs. tabu search algorithms for wavelength assignment
  - Algorithm A_1: multistart greedy (exponential run time distribution)
  - Algorithm A_2: tabu search
  - Sample size: N = 200

Application #2: wavelength assignment

Run-time distribution

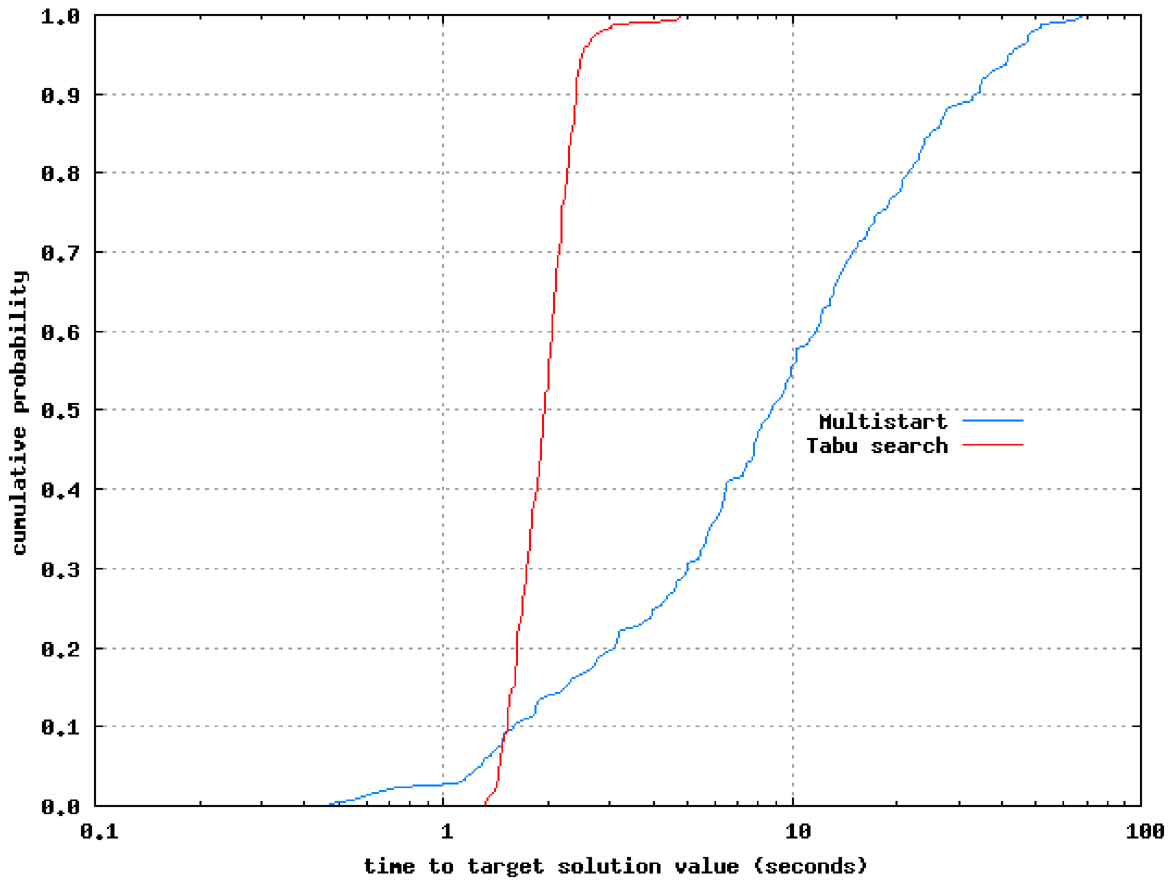


Quantile-quantile (q-q) plot



Run time distribution of tabu search is non-exponential (instance Brazil).

Application #2: wavelength assignment

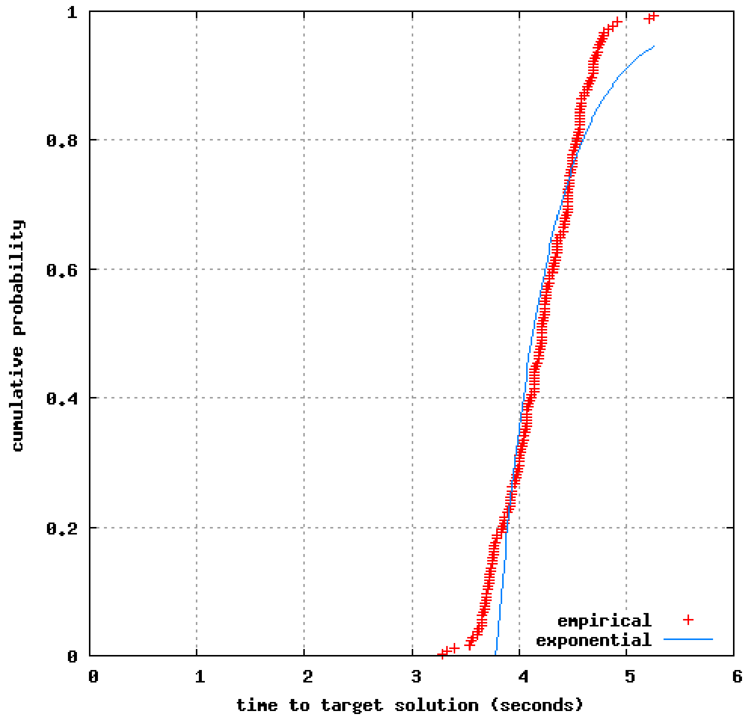


Brazil

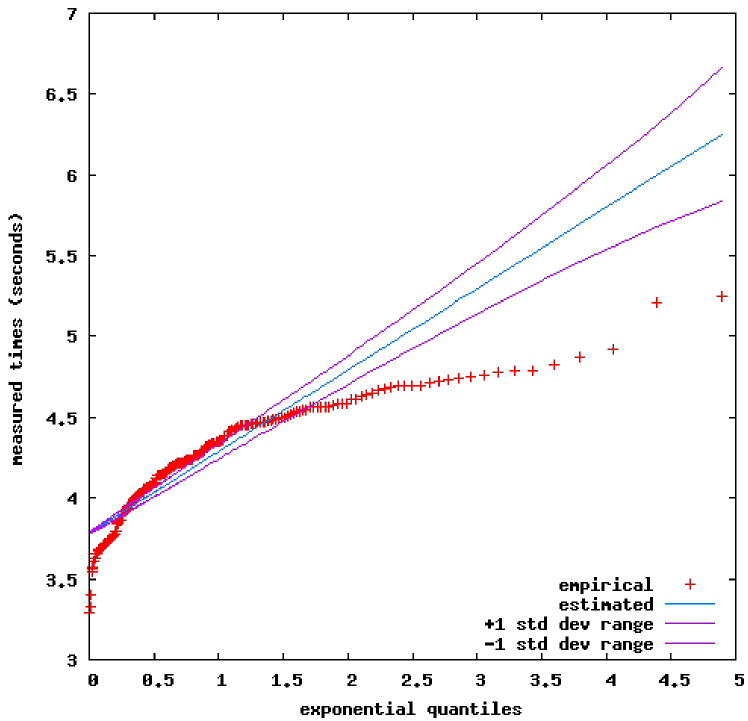
Tabu search clearly outperforms multistart: $P(MS \leq TS) = 0.106766$.

Application #2: wavelength assignment

Run-time distribution

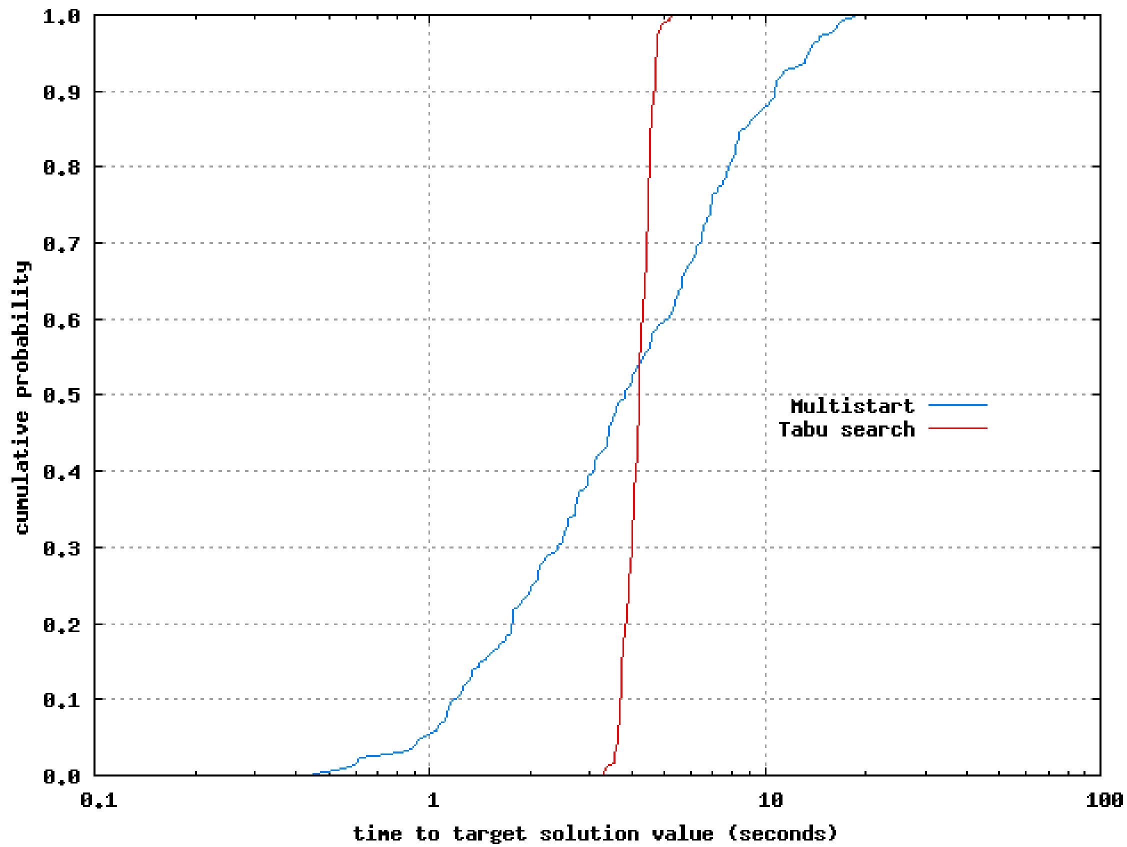


Quantile-quantile (q-q) plot



Run time distribution of tabu search is non-exponential (instance Finland).

Application #2: wavelength assignment



Finland

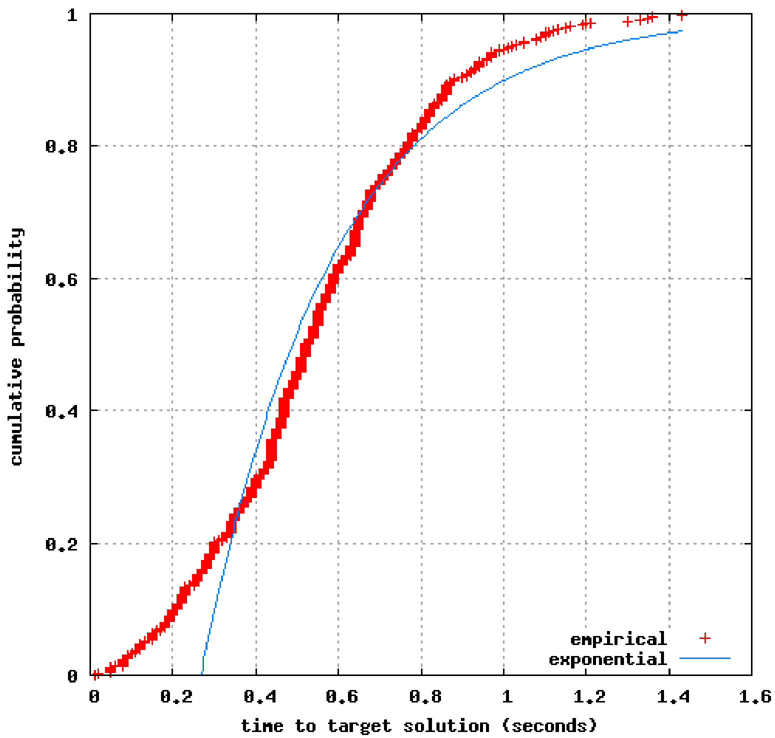
Now, multistart slightly outperforms tabu search: $P(MS \leq TS) = 0.545619$.

Application #3: 2-path network design

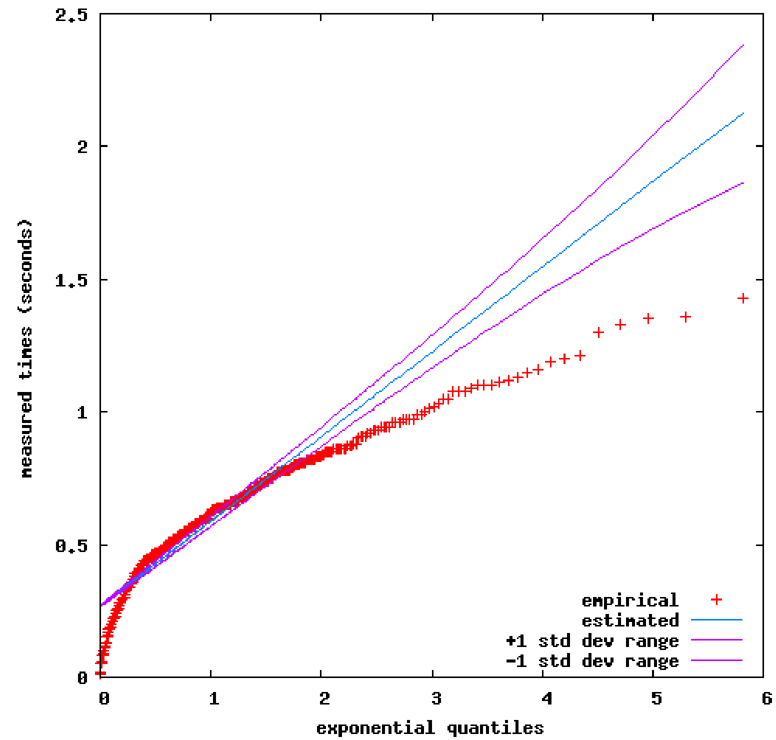
- Given a connected graph, edge weights, and a set of origin-destination nodes, find a minimum weighted edge subset containing a path formed by at most two edges between every o-d pair.
- GRASP algorithms for 2-path network design
  - Algorithm A_1: pure GRASP (exponential running times)
  - Algorithm A_2: GRASP with forward path-relinking
  - Algorithm A_3: GRASP with bidirectional path-relinking
  - Algorithm A_4: GRASP with backward path-relinking
  - Instance with 90 nodes and 900 origin-destination pairs
  - Sample size: N = 500

Application #3: 2-path network design

Run-time distribution



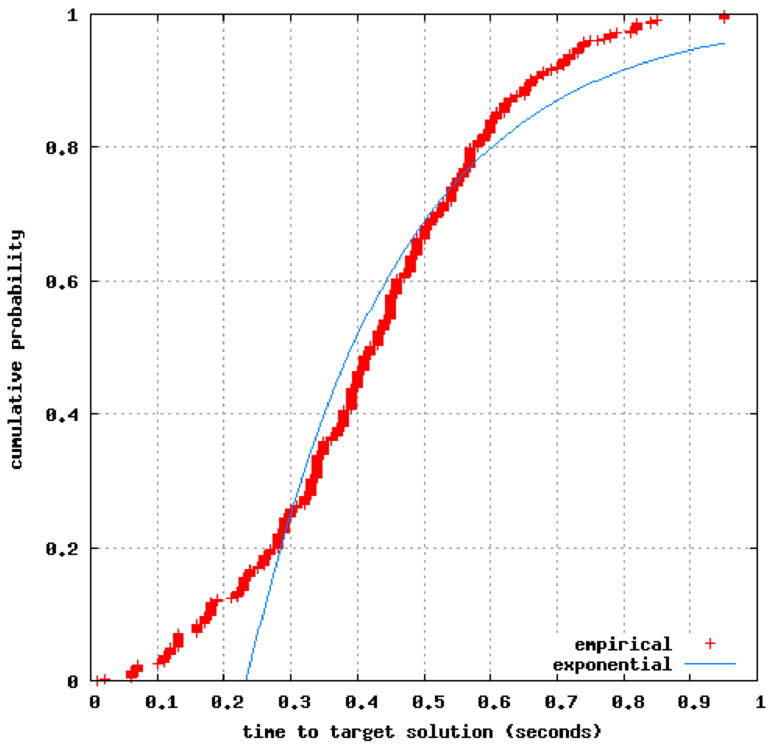
Quantile-quantile (q-q) plot



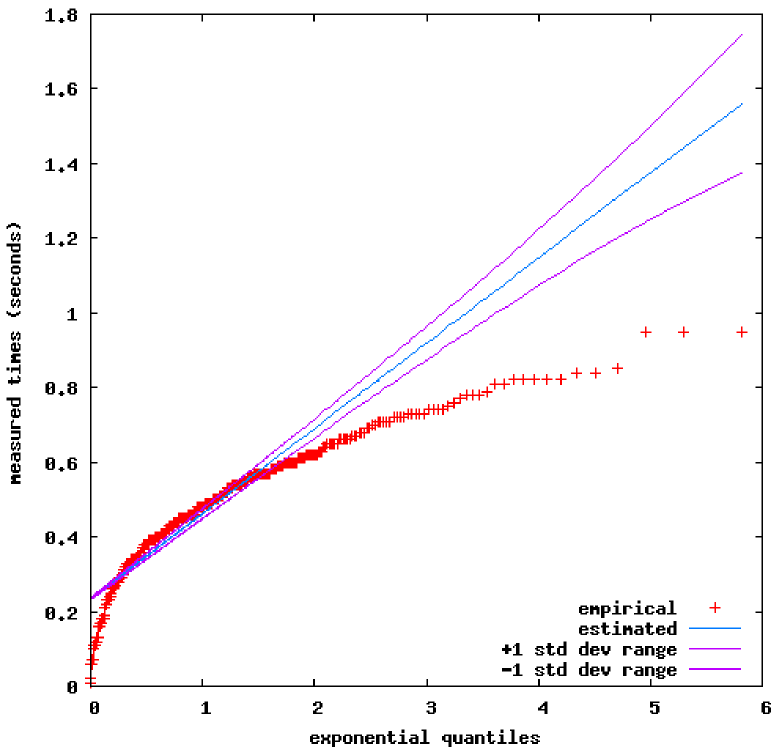
GRASP with forward path-relinking

Application #3: 2-path network design

Run-time distribution



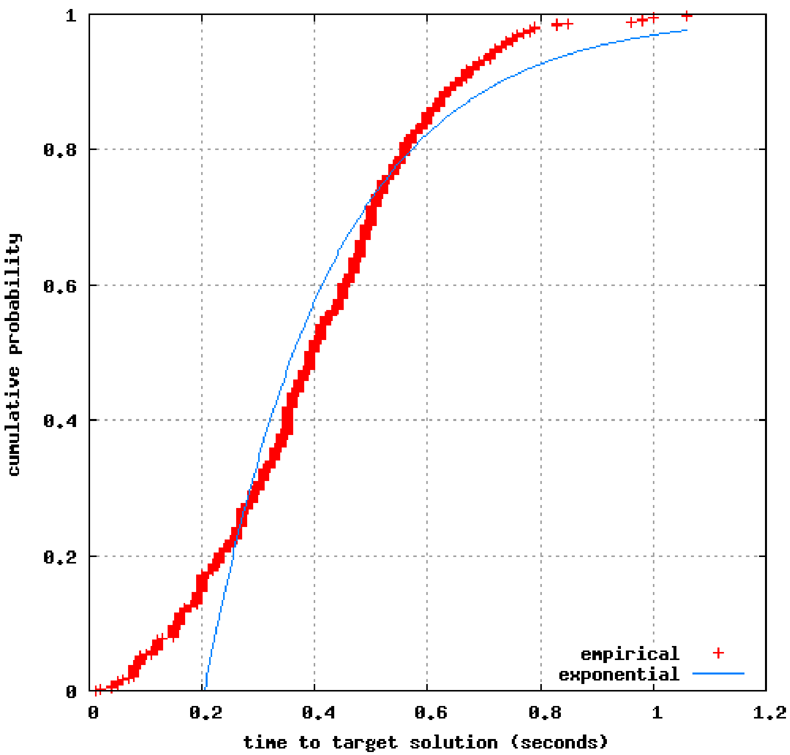
Quantile-quantile (q-q) plot



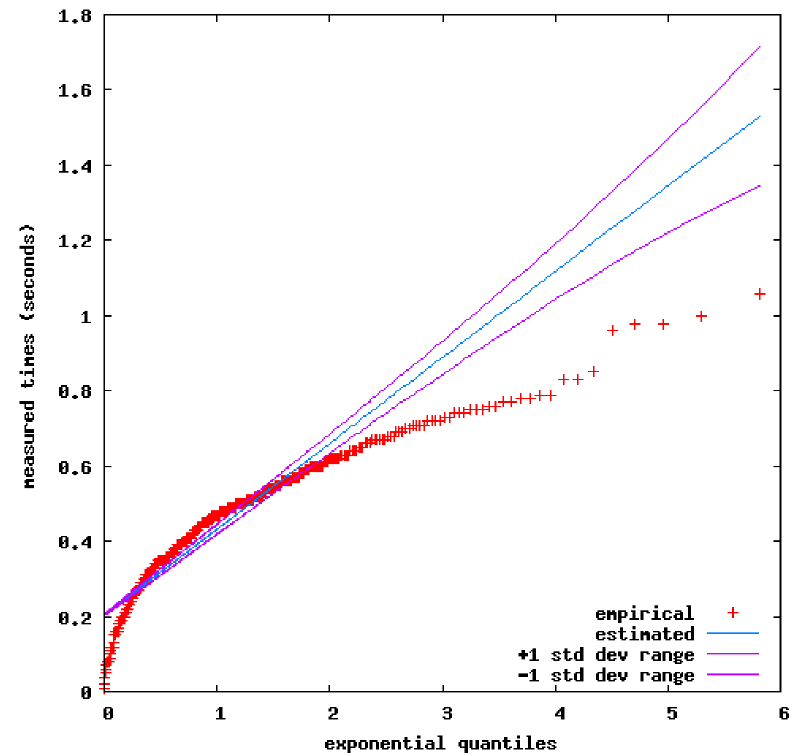
GRASP with bidirectional path-relinking

Application #3: 2-path network design

Run-time distribution

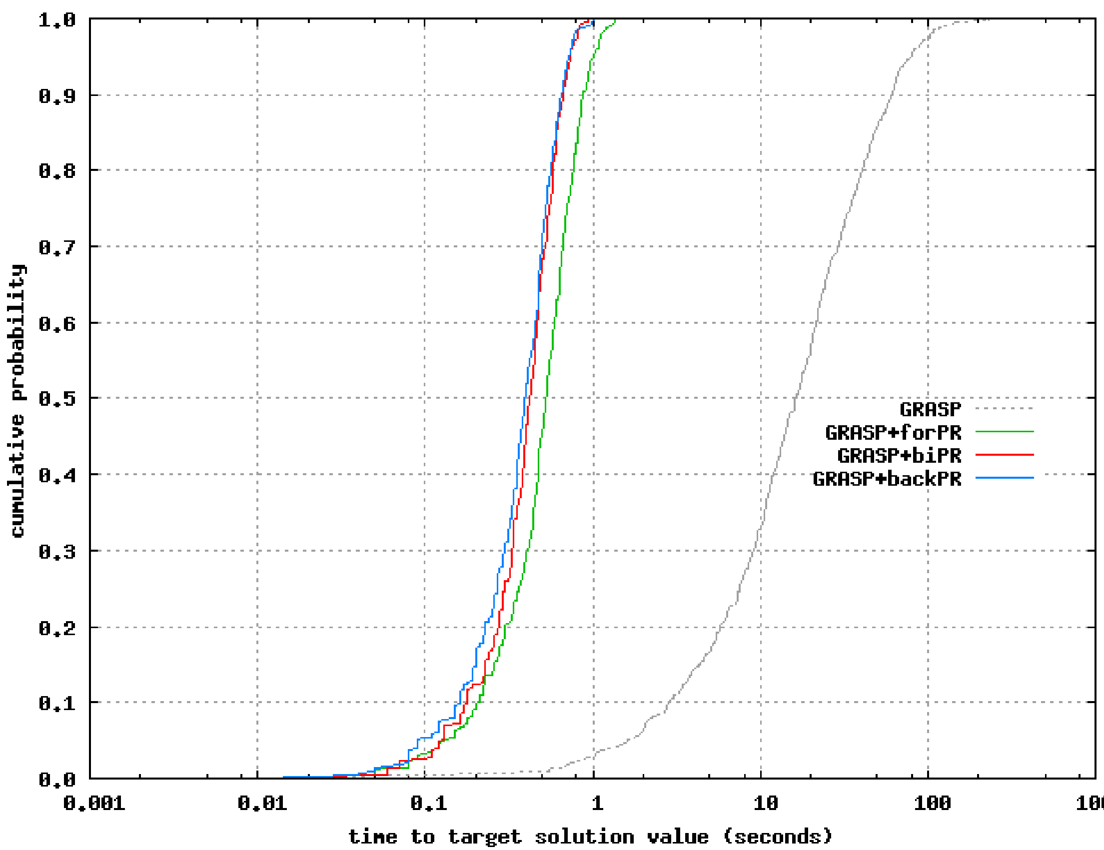


Quantile-quantile (q-q) plot



GRASP with backward path-relinking

Application #3: 2-path network design



Versions with path-relinking perform much better than pure GRASP.

P(GRASP+forPR ≤ GRASP) = 0.98447

P(GRASP+biPR ≤ GRASP+forPR) = 0.63400

P(GRASP+backPR ≤ GRASP+biPR) = 0.53602

Versions with backward and bidirectional PR perform very similarly.

Application #3: 2-path network design

- Application: another instance of the 2-path network design problem
  - Algorithm A_1: pure GRASP (exponential running times)
  - Algorithm A_2: GRASP with forward path-relinking
  - Algorithm A_3: GRASP with bidirectional path-relinking
  - Algorithm A_4: GRASP with backward path-relinking
  - Algorithm A_5: GRASP with mixed path-relinking
  - Instance: 80 nodes and 800 origin-destination pairs
  - Sample size: N = 500

Mixed path-relinking

- Given initial and guiding solutions: start from the initial solution, obtain the new current solution, exchange the roles of the current and guiding solutions, and repeat the procedure.

Mixed path-relinking

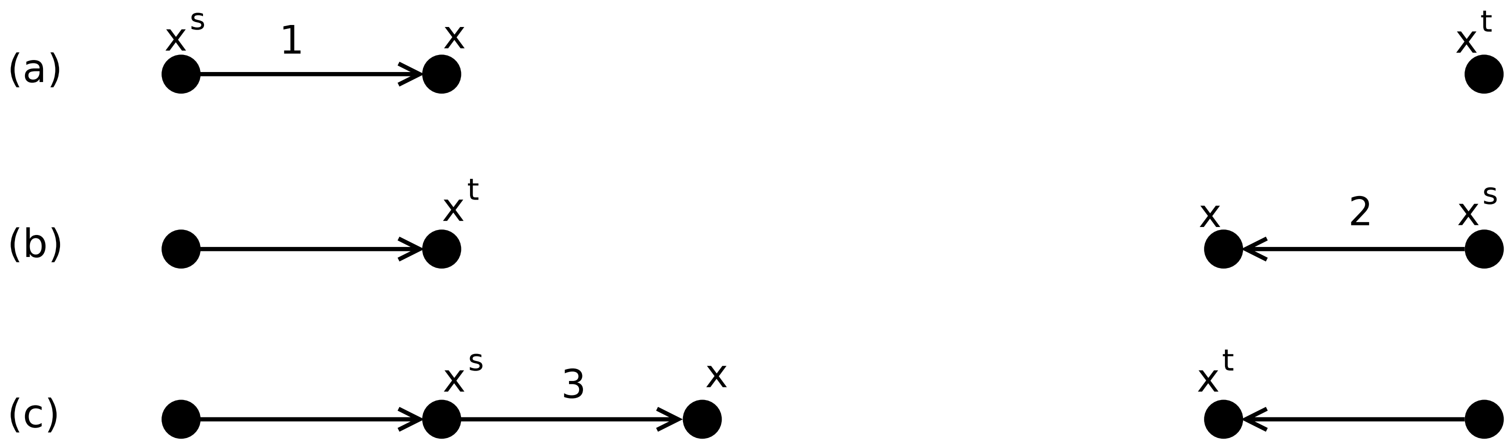
- Given initial and guiding solutions: start from the initial solution, obtain the new current solution, exchange the roles of the current and guiding solutions, and repeat the procedure.

Mixed path-relinking

- Given initial and guiding solutions: start from the initial solution, obtain the new current solution, exchange the roles of the current and guiding solutions, and repeat the procedure.

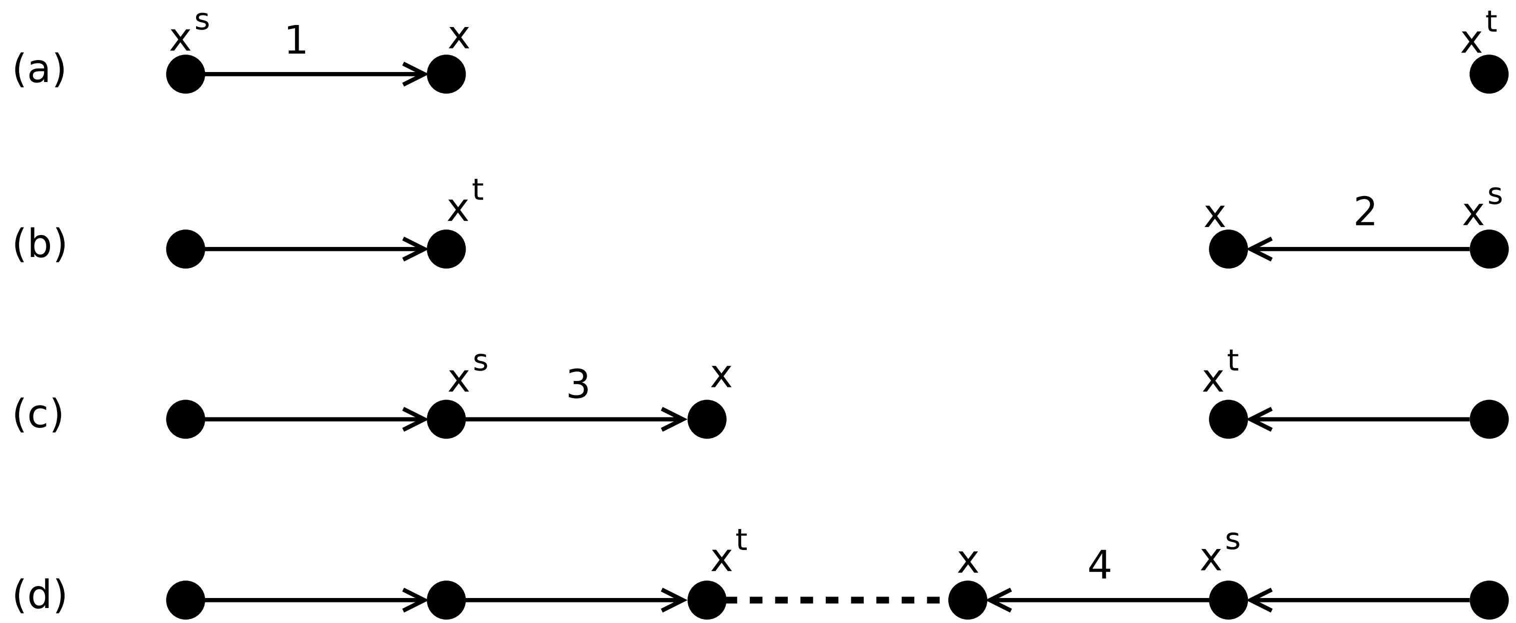
Mixed path-relinking

- Given initial and guiding solutions: start from the initial solution, obtain the new current solution, exchange the roles of the current and guiding solutions, and repeat the procedure.



Mixed path-relinking

- Given initial and guiding solutions: start from the initial solution, obtain the new current solution, exchange the roles of the current and guiding solutions, and repeat the procedure.



Application #3: 2-path network design

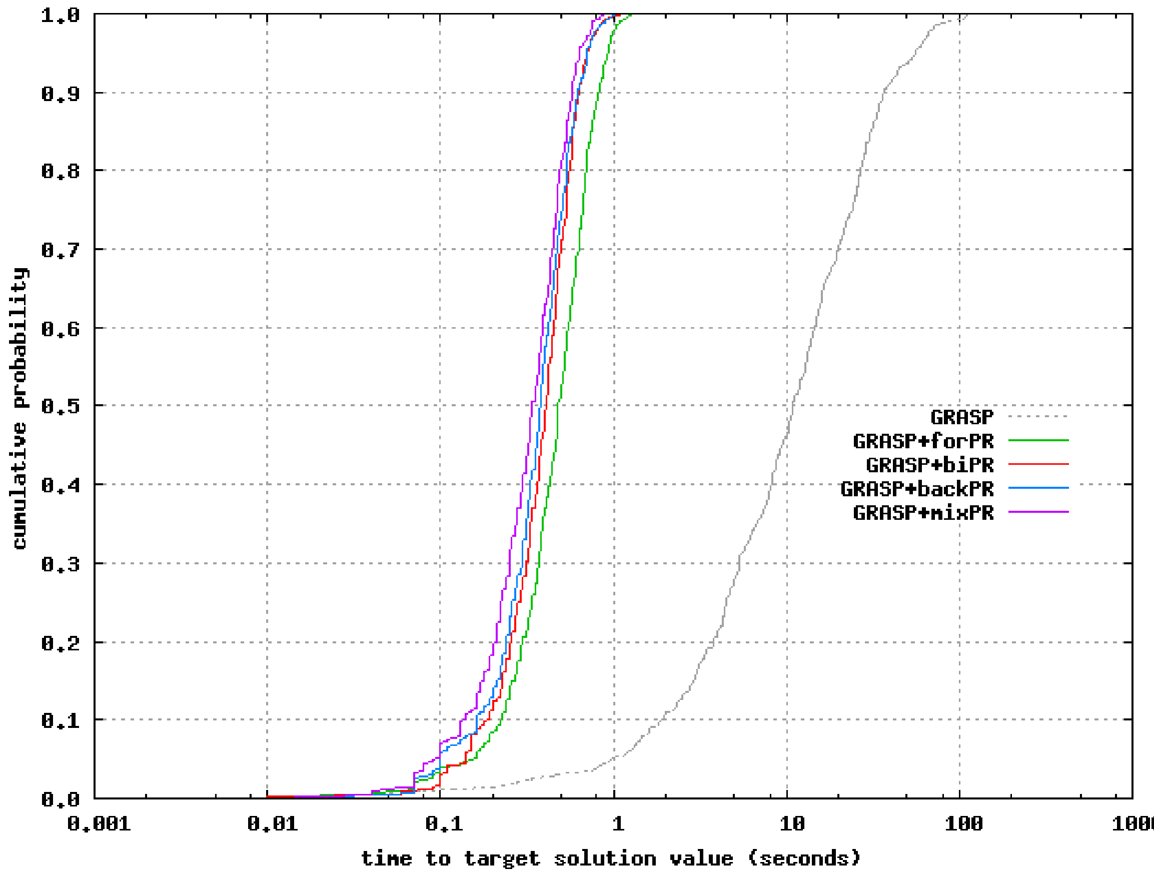
Versions with path-relinking perform much better than pure GRASP.

P(GRASP+forPR ≤ GRASP) = 0.96873

P(GRASP+biPR ≤ GRASP+forPR) = 0.61529

P(GRASP+backPR ≤ GRASP+biPR) = 0.53558

P(GRASP+mixPR ≤ GRASP+biPR) = 0.55435



Application #3: 2-path network design

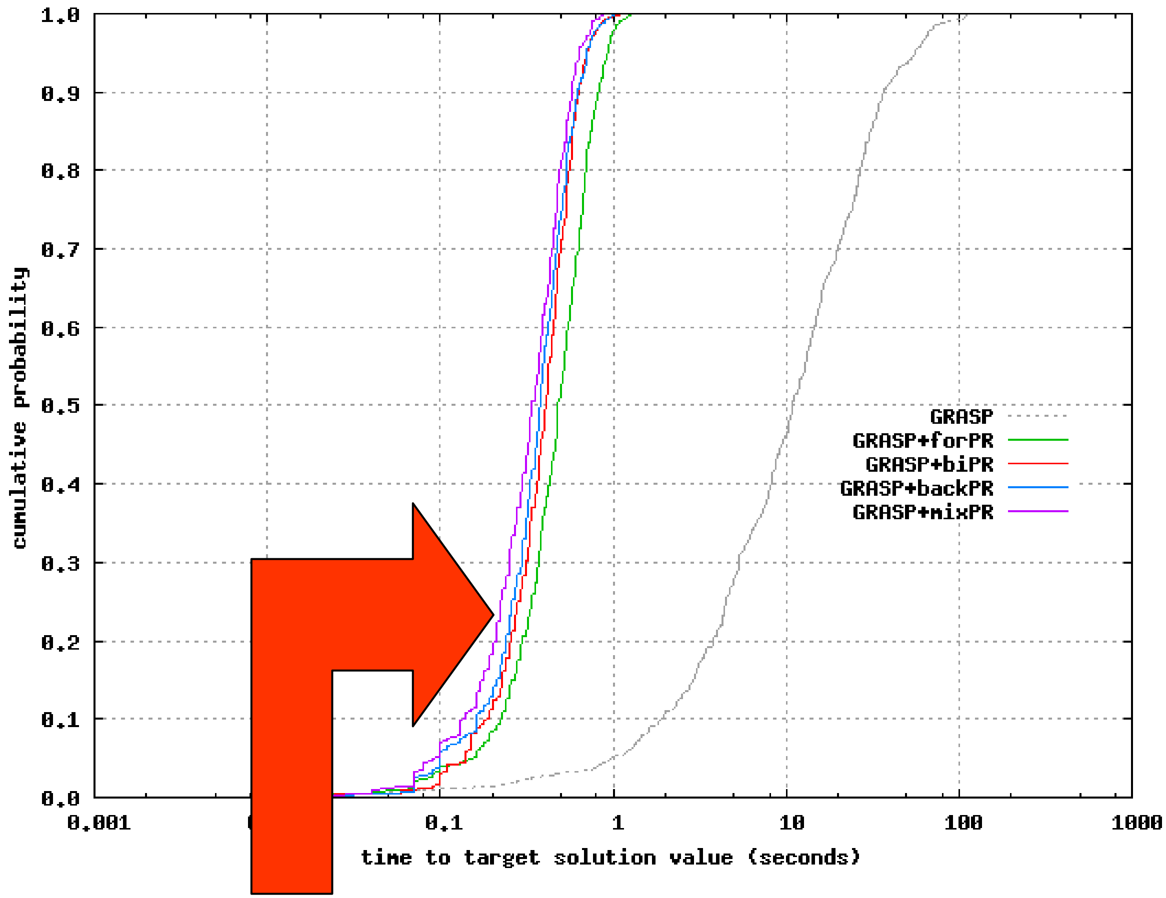
Versions with path-relinking perform much better than pure GRASP.

P(GRASP+forPR ≤ GRASP) = 0.96873

P(GRASP+biPR ≤ GRASP+forPR) = 0.61529

P(GRASP+backPR ≤ GRASP+biPR) = 0.53558

P(GRASP+mixPR ≤ GRASP+biPR) = 0.55435



Mixed path-relinking seems to outperform other versions of path-relinking.

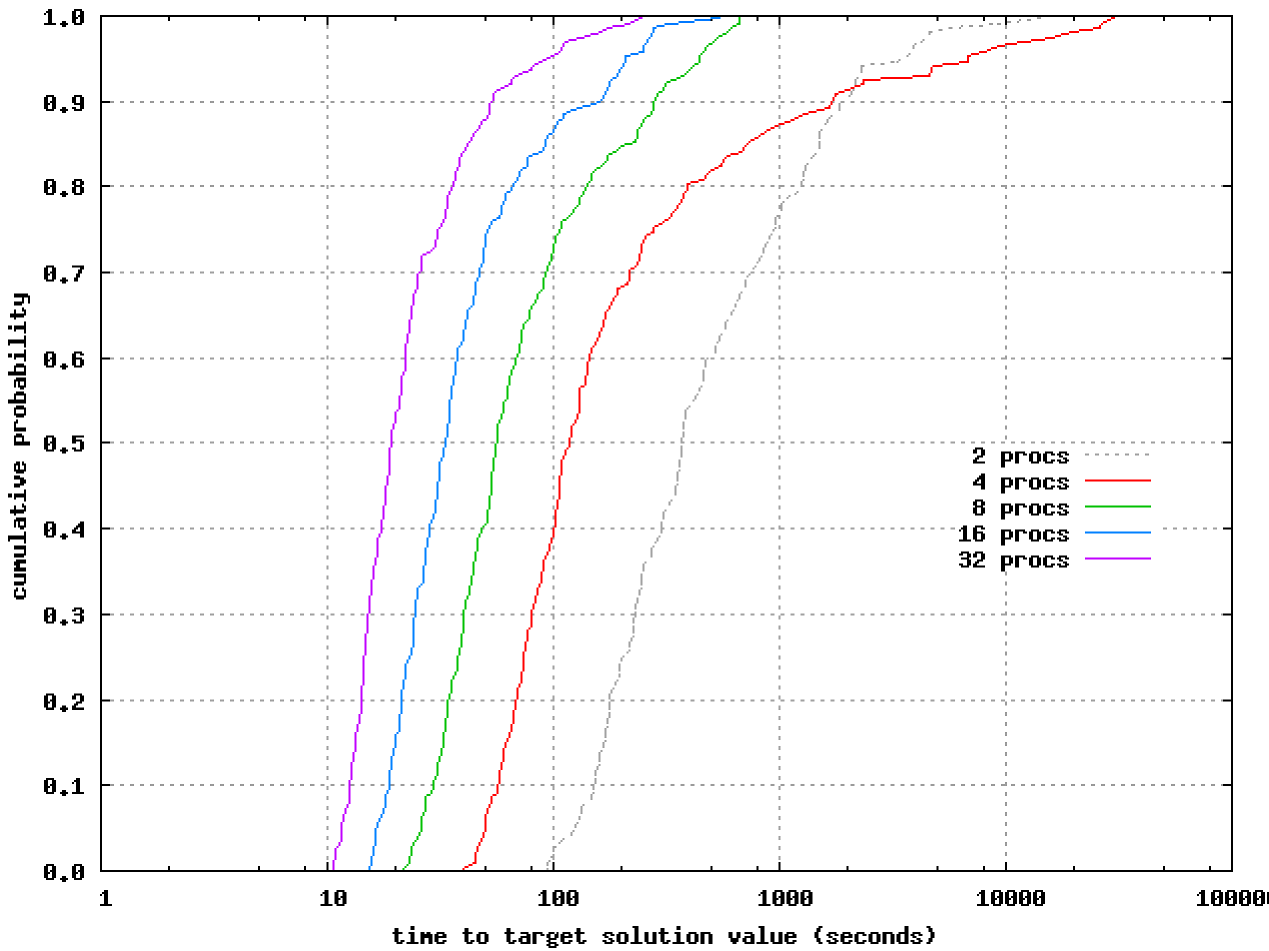
Parallel implementations

- Evaluation of trade-offs between:
  - cooperative and independent implementations
  - running time and number of processors
- Application: 2-path network design problem
  - Algorithm A_3: GRASP with bidirectional path-relinking
  - Cooperative and independent parallel implementations
  - Instance with 80 nodes and 800 origin-destination pairs
  - Sample size: N = 500
- Run time distributions of independent and cooperative implementations of GRASP with bidirectional PR on 2, 4, 8, 16, 32 processsors.

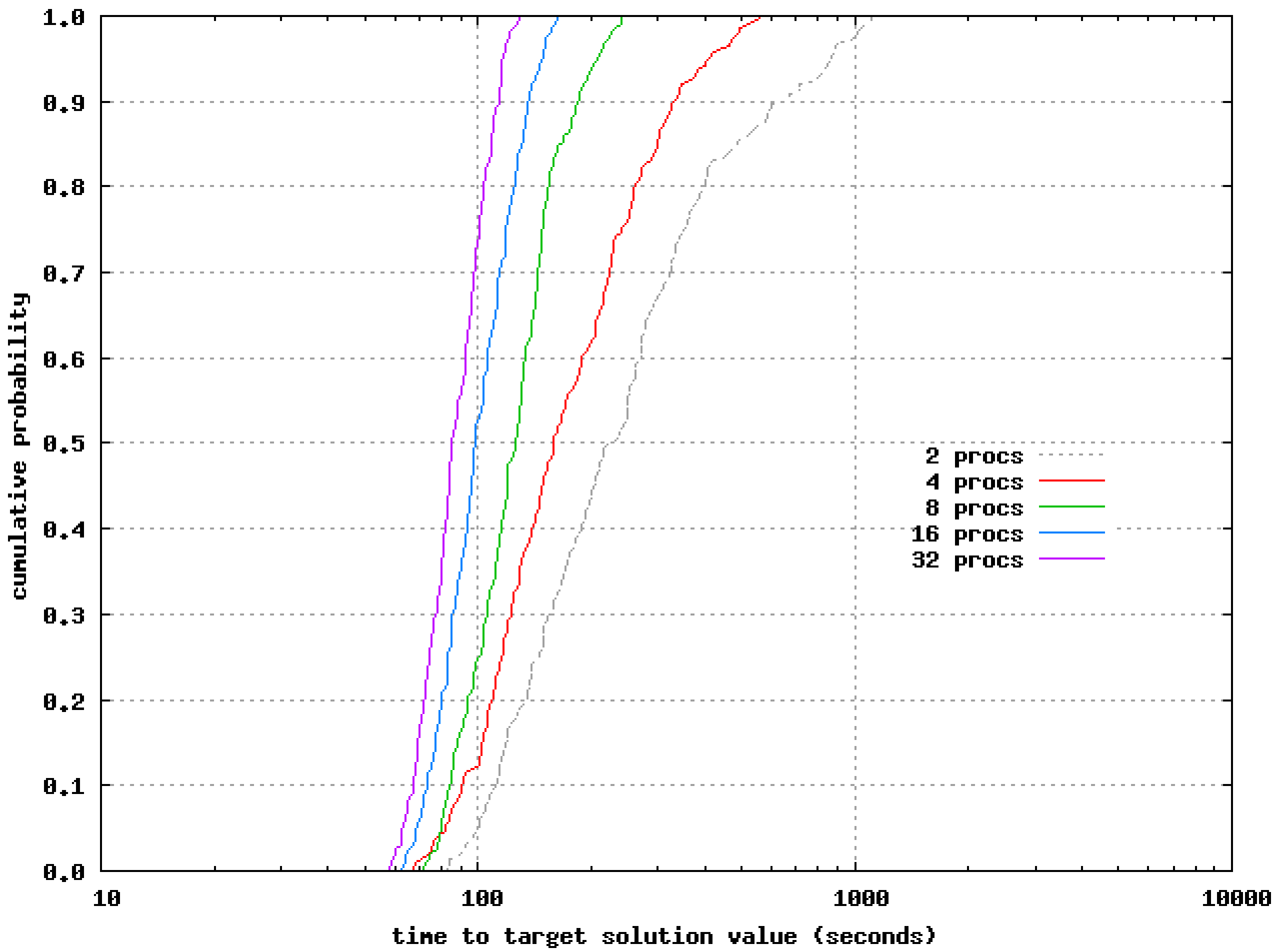
Parallel implementations

- Run time distributions of independent and cooperative implementations of GRASP with bidirectional PR on 2, 4, 8, 16, 32 processsors.
- A_k^1 (resp. A_k^2) denotes the cooperative (resp. independent) parallel implementation of GRASP with bidirectional path-relinking running on k = 2, 4, 8, 16, 32 processors.

Cooperative parallel implementations



Independent parallel implementations



Parallel implementations

- Probabilities that the cooperative parallel implementation performs better than the independent on k=2, 4, 8, 16, 32 processors
- Independent implementation performs better than the cooperative on two processors.
- Cooperative implementation performs better when the number of processors increases, because more processors are devoted to perform iterations.

k	$P(X_1^k \leq X_2^k)$
2	0.309660
4	0.597253
8	0.766698
16	0.860910
32	0.944846

Parallel implementations

- Probabilities that each of the two parallel implementations performs better on 2^{j+1} than on 2^j processors, for j = 1, 2, 3, 4.

# procs. a	# procs. b	$P(X_1^a \leq X_1^b)$ cooperative	$P(X_2^a \leq X_2^b)$ independent
4	2	0.766204	0.629691
8	4	0.748302	0.662932
16	8	0.713272	0.571173
32	16	0.742037	0.224815

Parallel implementations

- Cooperative implementation scales appropriately as the number of processors grows.
- Performance of the independent implementation seems to deteriorate in the same scenario.

Convergence: sample size (1)

- Convergence: influence of the sample size

  2-path network design problem

  90-node instance

  Algorithm 1: GRASP with backward PR

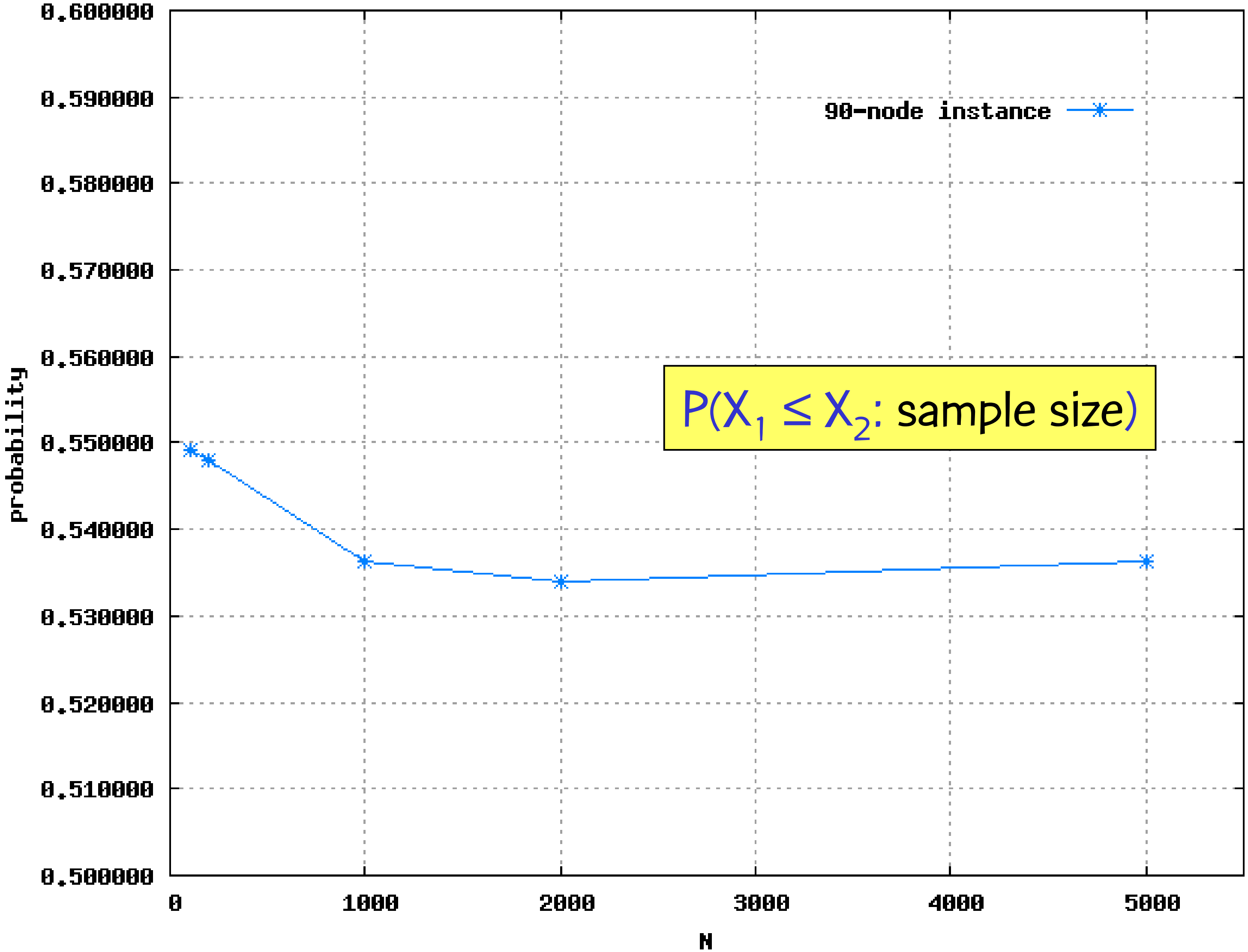
  Algorithm 2: GRASP with bidirectional PR

  $P(X_1 \leq X_2 : N = 100) = 0.54925$

  $P(X_1 \leq X_2 : N = 200) = 0.54796$

  $P(X_1 \leq X_2 : N = 5000) = 0.53636$

Convergence: sample size (1)



Convergence: sample size (2)

- Convergence: influence of the sample size

  Server replication problem

  25-node instance

  Algorithm 1: DM-D5 version of DM-GRASP

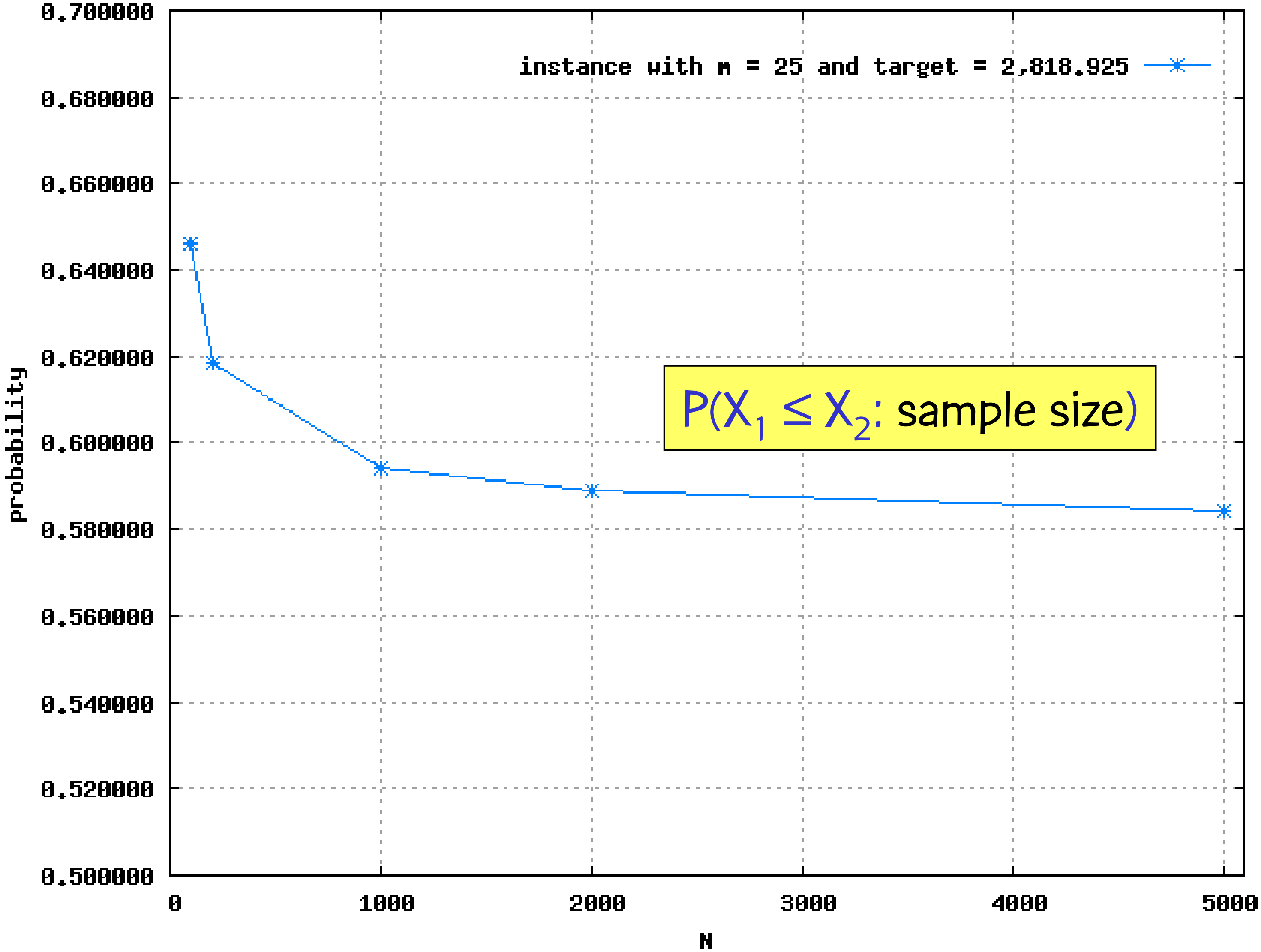
  Algorithm 2: pure GRASP

  $P(X_1 \leq X_2 : N = 100) = 0.64620$

  $P(X_1 \leq X_2 : N = 200) = 0.61834$

  $P(X_1 \leq X_2 : N = 5000) = 0.58432$

Convergence: sample size (2)



Convergence: target value

- Convergence: influence of the target value

  2-path network design problem

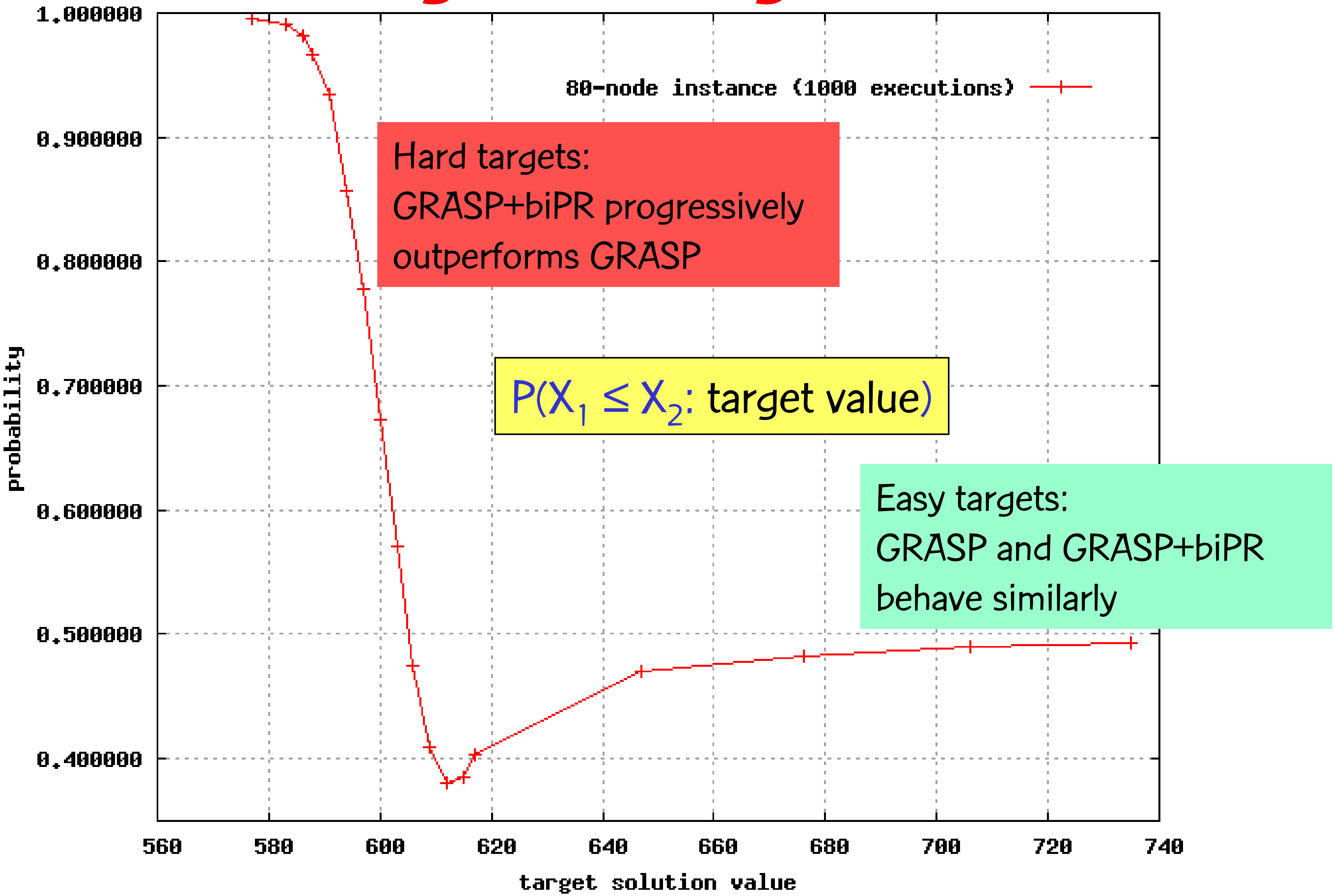
  80-node instance

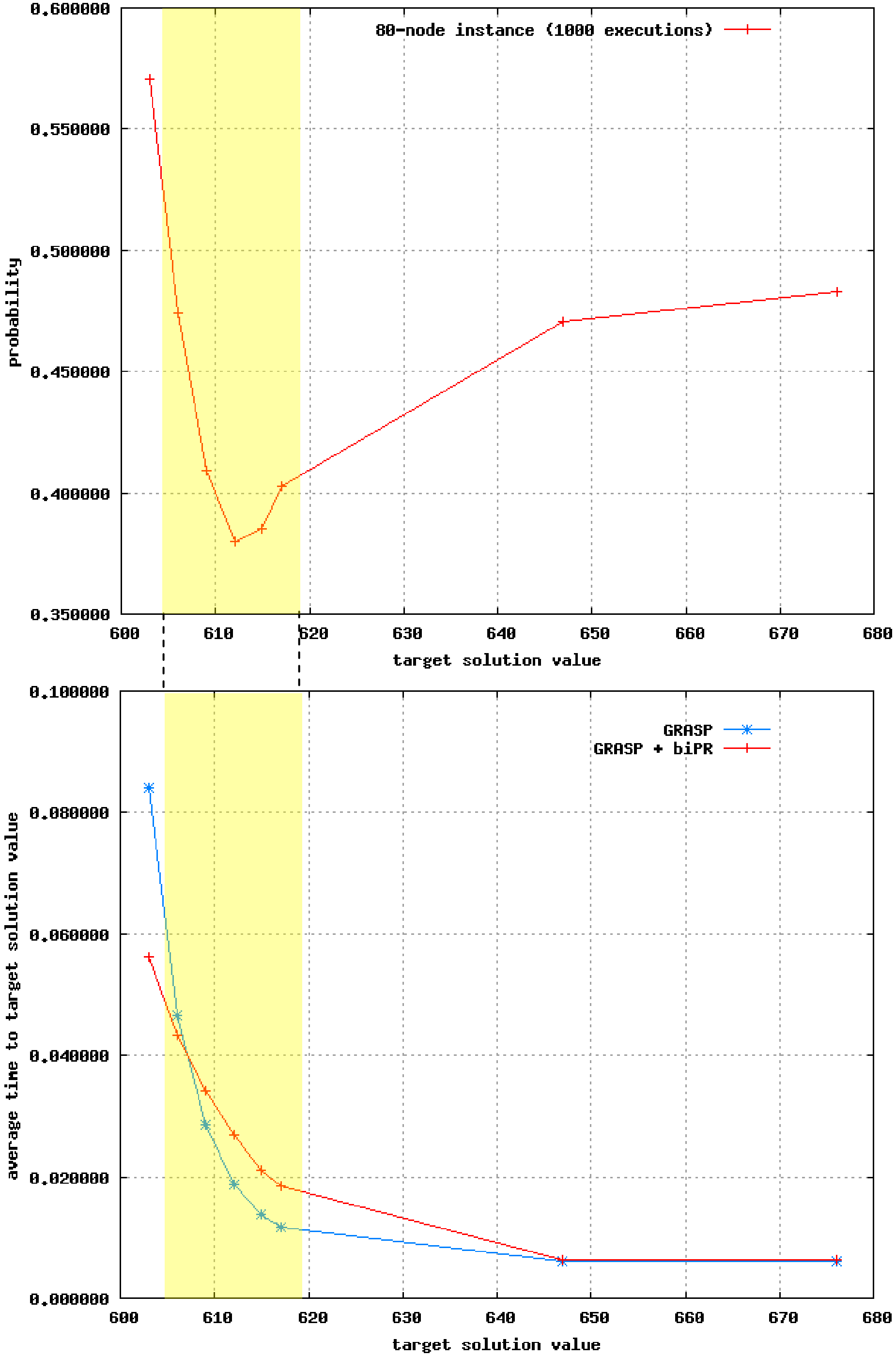
  Algorithm 1: GRASP with bidirectional PR

  Algorithm 2: pure GRASP

  $P(X_1 \leq X_2$: target value)

Convergence: target value





- Convergence: influence of the target value
- Consider the average time to target values.
- PR introduces an overhead in GRASP for easy targets.
- Contrarily, PR strongly improves GRASP when targets get harder: average times increase more slowly.

Concluding remarks

- Run time distributions are very useful tools to characterize SLS algorithms.
- Closed form index to compare exponential run time distributions.
- Numerical procedure to compute the probability that one algorithm finds a target value in less time than another, for general run time distributions.
- New probability index provides an additional measure for comparing the performance of metaheuristics based on SLS algorithms.

Concluding remarks

- Can also be used in the evaluation of parallel SLS algorithms, providing an indicator to evaluate trade-offs between elapsed times and the number of processors (scalability).
- Run time distributions and new probability index are very helpful and give additional insight for algorithm engineering.
- Extension to benchmark sets formed by multiple instances and targets.
- Software available from the authors upon request.