Runtime distributions

Celso C. Ribeiro (celso@ic.uff.br)

Universidade Federal Fluminense

Metaheuristics – May 9, 2019

Overview of talk

- Time-to-target plots
- Runtime distribution of GRASP
- Comparing algorithms with exponential runtime distributions
- Comparing algorithms with general runtime distributions
- Numerical applications to sequential algorithms
  - DM-D5 and GRASP algorithms for server replication
  - Multistart and tabu search algorithms for routing and wavelength assignment
  - GRASP algorithms for 2-path network design
- Comparing and evaluating parallel algorithms

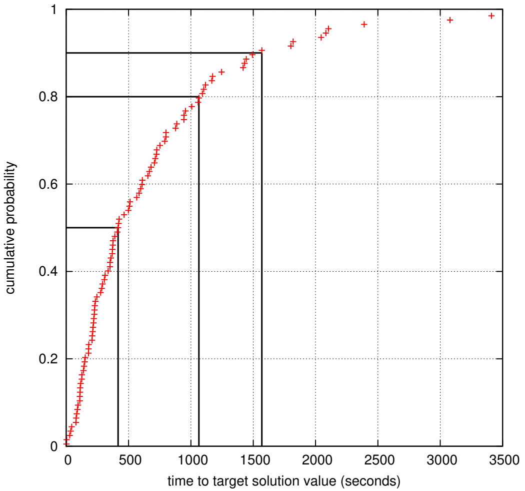
Introduction

- Runtime distributions or time-to-target plots display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis.
- They provide a very useful tool to characterize the running times of stochastic algorithms for combinatorial optimization problems and to compare different algorithms or strategies for solving a given problem.
- They have been widely used as a tool for algorithm design and comparison.

Time-to-target plots

- Let $\mathcal{P}$ be an optimization problem and $\mathcal{H}$ a randomized heuristic for this problem.
- Furthermore, let $\mathcal{I}$ be a specific instance of $\mathcal{P}$ and let *look4* be a solution cost *target value* for this instance.
- Heuristic $\mathcal{H}$ is run N times on the fixed instance $\mathcal{I}$ and the algorithm is made to stop as soon as a solution whose objective function is at least as good as the given target value *look4* is found.
- For each of the N runs, the random number generator used in the implementation of the heuristic is initialized with a distinct seed and, therefore, the runs are assumed to be independent.
- The solution time of each run is recorded and saved.
- To compare their empirical and theoretical distributions, we follow a standard graphical methodology for data analysis.
- This methodology is used to produce the time-to-target plots (TTT-plots).

Time-to-target plots



- After concluding the N independent runs, solution times are sorted in increasing order.
- The i-th sorted solution time t_i is associated with a probability $p_i = (i - 1/2)/N$, and the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, N$, are plotted.
- The figure illustrates this estimated cumulative probability distribution plot for problem $\mathcal{P}$, a GRASP heuristic $\mathcal{H}$, instance $\mathcal{I}$, and target *look4*.
- Observation: The probability that the heuristic finds a solution at least as good as the target value in at most 416 seconds is about 50%, in at most 1064 seconds is about 80%, and in at most 1569 seconds is about 90%.

Runtime distribution of GRASP

- The previous plot appears to fit an exponential distribution, or more generally, a shifted exponential distribution.
- To estimate the parameters of this two-parameter exponential distribution, we first draw the theoretical quantile-quantile plot (or Q-Q plot) for the data.
- To describe Q-Q plots, we recall that the cumulative distribution function for the two-parameter exponential distribution is given by:

$$F(t) = 1 - e^{-(t-\mu)/\lambda}$$

  - λ is the mean of the distribution data (and also is the standard deviation of the data).
  - μ is the shift of the distribution with respect to the ordinate axis.

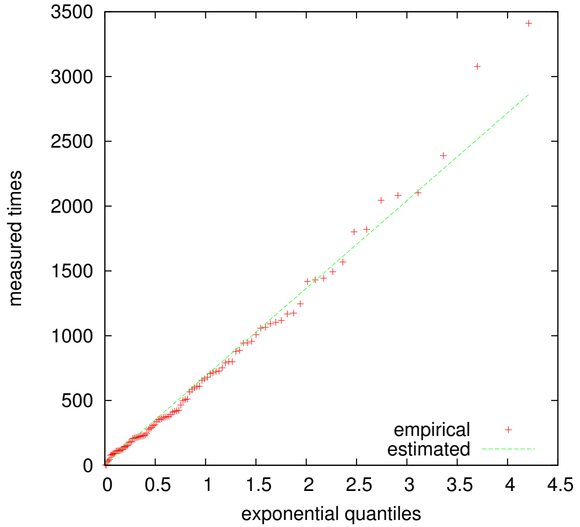
Runtime distribution of GRASP

- The *quantiles* of the data of an empirical distribution are derived from the (sorted) raw data, which in our case are N measured (sorted) running times.
- Quantiles are cutpoints that group a set of sorted observations into classes of equal (or approximately equal) size.
- For each value p_i, $i = 1, \ldots, N$, we associate a p_i-quantile $q(p_i)$ of the theoretical distribution.
- For each p_i-quantile we have, by definition, that $F((q(p_i)) = p_i$.
- Hence, $q(p_i) = F^{-1}(p_i)$ and therefore, for the two-parameter exponential distribution, we have $q(p_i) = -\lambda \cdot \ln(1 - p_i) + \mu$.
- Note that if we were to use $p_i = 1/N$, for $i = 1, \ldots, N$, then $q(p_N)$ would be undefined.

Runtime distribution of GRASP

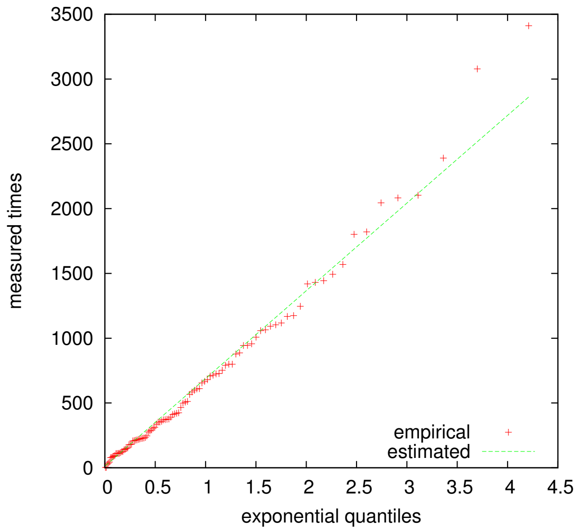
- A theoretical quantile-quantile plot (or theoretical Q-Q plot) is obtained by plotting the quantiles of the data of an empirical distribution against the quantiles of a theoretical distribution. This involves three steps:
  - First, the data (in this case, the measured solution times) are sorted in ascending order.
  - Second, the quantiles of the theoretical exponential distribution are obtained.
  - Finally, a plot of the data against the theoretical quantiles is made.
- In a situation where the theoretical distribution is a close approximation of the empirical distribution, the points in the Q-Q plot will have a nearly straight configuration.
- In a plot of the data against a two-parameter exponential distribution with $\lambda = 1$ and $\mu = 0$, the points would tend to follow the line $y = \hat{\lambda} \cdot x + \hat{\mu}$.
- Consequently, parameters λ and μ of the two-parameter exponential distribution can be estimated, respectively, by the slope $\hat{\lambda}$ and the intercept $\hat{\mu}$ of the line depicted in the Q-Q plot.

Runtime distribution of GRASP



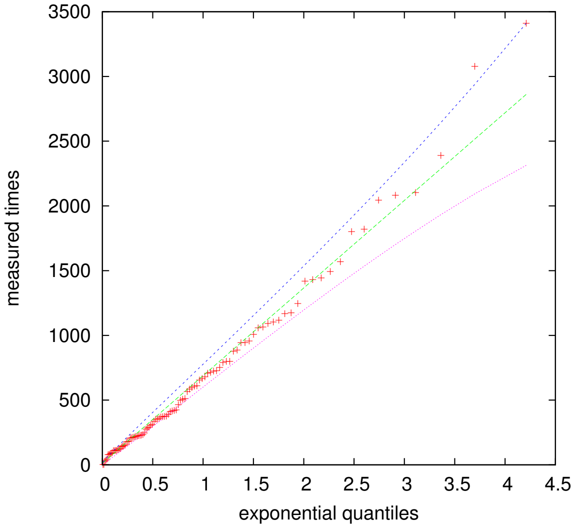
- The Q-Q plot shown is obtained by plotting the measured times in the ordinate against the quantiles of a two-parameter exponential distribution with $\lambda = 1$ and $\mu = 0$ in the abscissa, given by $q(p_i) = -\ln(1 - p_i)$, for $i = 1, \dots, n$.
- To avoid possible distortions caused by outliers, we do not estimate the distribution mean with the data mean or by linear regression on the points of the Q-Q plot.
- Instead, we estimate the slope $\hat{\lambda}$ of the line $y = \lambda \cdot x + \mu$ using the upper quartile q_u and lower quartile q_l of the data.

Runtime distribution of GRASP



- The *upper quartile* q_u and *lower quartile* q_l are, respectively, the $q(1/4)$ and $q(3/4)$ quantiles.
- We take $\hat{\lambda} = (z_u - z_l)/(q_u - q_l)$ as an estimate of the slope, where z_u and z_l are the u-th and l-th points of the ordered measured times, respectively.
- This informal estimation of the distribution of the measured data mean is robust since it will not be distorted by a few outliers.
- Consequently, the estimate for the shift is $\hat{\mu} = z_l - \hat{\lambda} q_l$.

Runtime distribution of GRASP



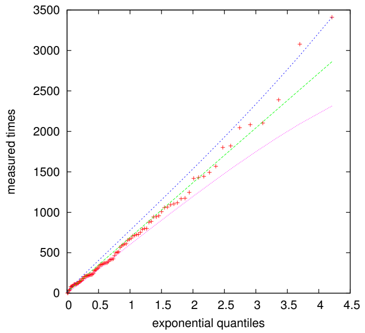
- To analyze the straightness of the Q-Q plots, we superimpose them with variability information.
- For each plotted point, we show plus and minus one standard deviation in the vertical direction from the line fitted to the plot.
- An estimate of the standard deviation for point z_i, $i = 1, \dots, n$, of the Q-Q plot is $\hat{\sigma} = \hat{\lambda}[p_i/(1 - p_i)n]^{\frac{1}{2}}$.
- The figure shows an example of a Q-Q plot with superimposed variability information.

Runtime distribution of GRASP

- When observing a theoretical quantile-quantile plot with superimposed standard deviation information, one should avoid turning such information into a formal test.
- One important fact that must be kept in mind is that the natural variability of the data generates departures from the straightness, even if the model of the distribution is valid.
- The most important reason for portraying standard deviation is that it gives us a sense of the relative variability of the points in the different regions of the plot.
- However, since one is trying to make simultaneous inferences from many individual inferences, it is difficult to use standard deviations to judge departures from the reference distribution.
- For example, the probability that a particular point deviates from the reference line by more than two standard deviations is small.
- However, the probability that any of the points deviates from the line by two standard deviations is probably much greater.

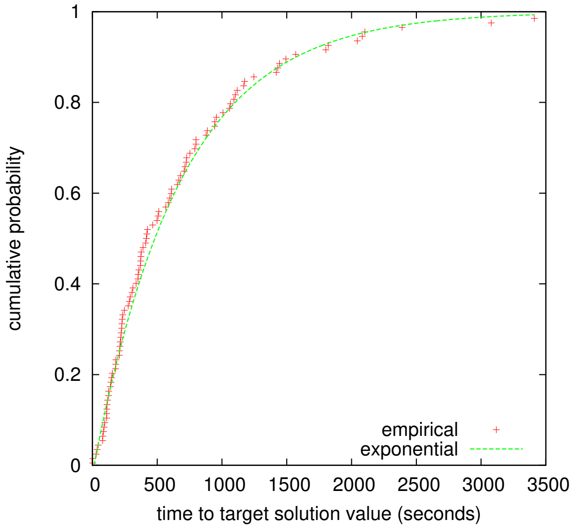
Runtime distribution of GRASP

- In order statistics, this is made more difficult by the high correlation that exists between neighboring points.
- If one plotted point deviates by more than one standard deviation, there is a good chance that a whole bunch of them will too.



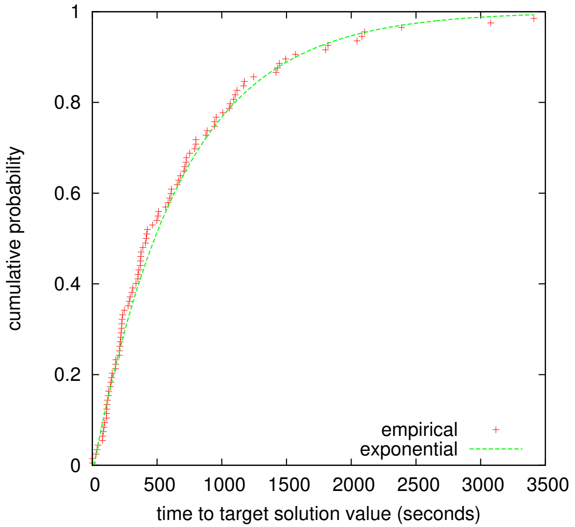
- Another point to keep in mind is that standard deviations vary substantially in the Q-Q plot.
- As one can observe in the previous Q-Q plot, the standard deviation of the points near the high end is substantially larger than the standard deviation of the points near the other end.

Runtime distribution of GRASP



- Once the two parameters of the distribution have been estimated, a superimposed plot of the empirical and theoretical distributions can be made.
- The figure depicts the superimposed empirical and theoretical distributions corresponding to the Q-Q plot in the previous figure.
- The runtime distribution of a pure GRASP heuristic has been shown experimentally to behave as a random variable that fits an exponential distribution.

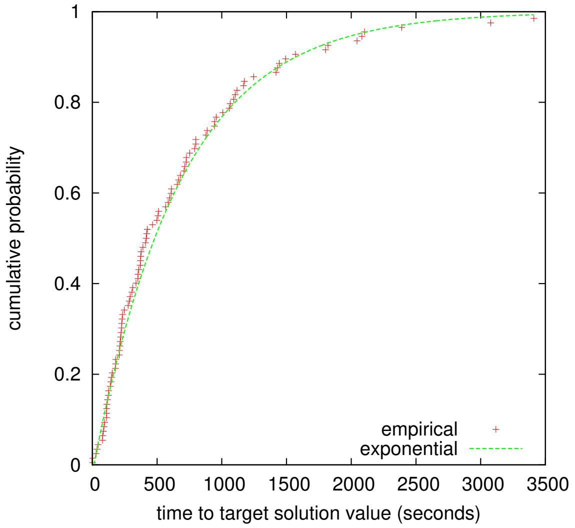
Runtime distribution of GRASP



- However, in the case of more elaborate heuristics where setup times are not negligible, the runtimes fit a two-parameter or shifted exponential distribution.

- Therefore, the probability density function of the time-to-target random variable is given by $f(t) = (1/\lambda) \cdot e^{-t/\lambda}$ in the first case (exponential distribution) and by $f(t) = (1/\lambda) \cdot e^{-(t-\mu)/\lambda}$ in the second (shifted exponential distribution), with the parameters $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}^+$ being associated with the shape and the shift of the exponential function, respectively.

Runtime distribution of GRASP



- The figure illustrates this result, depicting the superimposed empirical and theoretical distributions observed for an instance of the maximum covering problem where one wants to choose 500 out of 1000 facility locations such that, of the 10,000 customers, the sum of the weights of those that are covered is maximized.

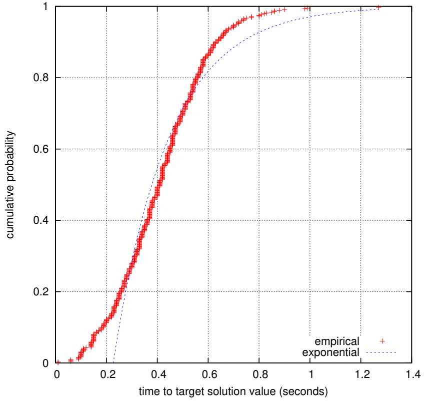
- The best known solution for this instance is 33,343,542 and the target solution value used was 33,339,175 (about 0.01% off of the best known solution).

Runtime distribution of GRASP

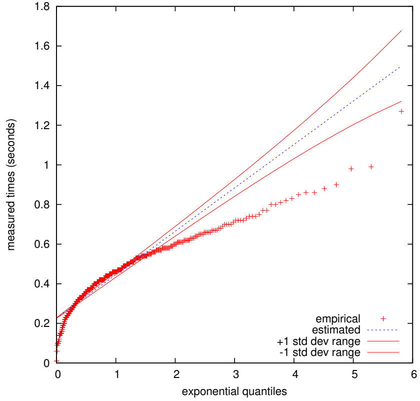
- However, if path-relinking is applied as an intensification step at the end of each GRASP iteration, then the iterations are no longer independent and the memoryless characteristic of GRASP is destroyed.
- This also happens in the case of cooperative parallel implementations of GRASP.
- Consequently, the time-to-target random variable may not fit an exponential distribution in such situations.
- This result is illustrated by two implementations of GRASP with bidirectional path-relinking:
  - The first is an application to the 2-path network design problem.
  - The second is an application to the three-index assignment problem.

Runtime distribution of GRASP

The figure depicts the runtime distribution and the corresponding quantile-quantile plot for GRASP with bidirectional path-relinking of an instance of the 2-path network design problem with 80 nodes and 800 origin-destination pairs, with target set to 588.



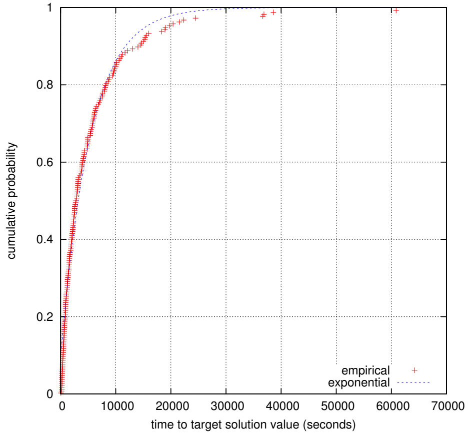
(a) Runtime distribution



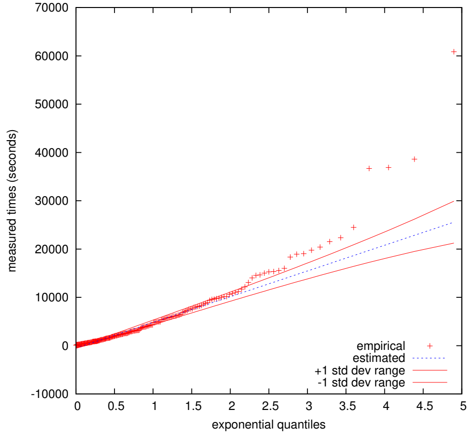
(b) Quantile-quantile plot

Runtime distribution of GRASP

The figure shows the runtime distribution and the corresponding quantile-quantile plot for GRASP with bidirectional path-relinking on Balas and Saltzman problem 22.1, with the target value set to 8.



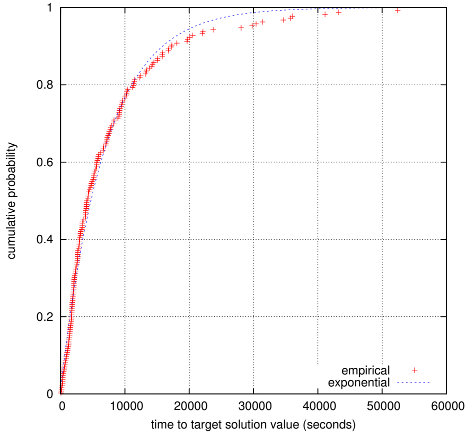
(a) Runtime distribution



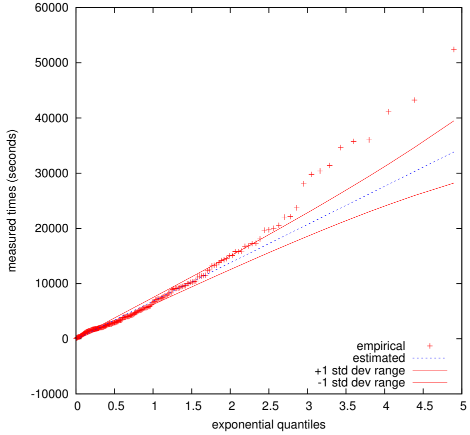
(b) Quantile-quantile plot

Runtime distribution of GRASP

The figure shows the runtime distribution and the corresponding quantile-quantile plot for GRASP with bidirectional path-relinking on Balas and Saltzman problem 24.1, with the target value set to 7.



(a) Runtime distribution



(b) Quantile-quantile plot

Runtime distribution of GRASP

Observations:

- For both heuristics and these three example instances, we observe that points steadily deviate by more than one standard deviation from the estimate for the upper quantiles in the quantile-quantile plots.
    - That is, many points associated with large computation times fall outside the plus or minus one standard deviation bounds.
- Therefore, we cannot say that these runtime distributions are exponentially distributed.

Comparing algorithms with exponential runtime distributions

- We assume the existence of two randomized algorithms A_1 and A_2 for the approximate solution of some optimization problem.
- Furthermore, we assume that their solution times fit exponential (or shifted exponential) distributions.
- We denote by X_1 (resp. X_2) the continuous random variable representing the time needed by algorithm A_1 (resp. A_2) to find a solution as good as a given target value:

$$X_1 \mapsto \begin{cases} 0, & \tau < T_1 \\ \lambda_1 \cdot e^{-\lambda_1(\tau - T_1)}, & \tau \geq T_1 \end{cases}$$

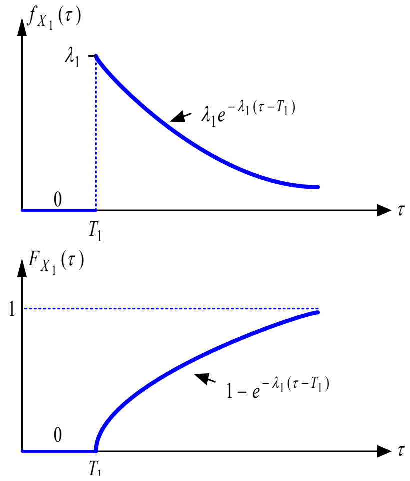
and

$$X_2 \mapsto \begin{cases} 0, & \tau < T_2 \\ \lambda_2 \cdot e^{-\lambda_2(\tau - T_2)}, & \tau \geq T_2 \end{cases}$$

where T_1, λ_1, T_2, and λ_2 are parameters (λ_1 and λ_2 define the shape of each shifted exponential distribution, whereas T_1 and T_2 denote by how much each of them is shifted).

Comparing algorithms with exponential runtime distributions

The figure depicts the cumulative probability distribution and the probability density function of the random variable X_1.



Comparing algorithms with exponential runtime distributions

- Since both algorithms stop when they find a solution at least as good as the target, we can say that algorithm A_1 performs better than A_2 if the former stops before the latter.
- Therefore, we must evaluate the probability $Pr(X_1 \leq X_2)$ that the random variable X_1 takes a value smaller than or equal to X_2.
- Conditioning on the value of X_2 and applying the total probability theorem, we obtain

$$Pr(X_1 \leq X_2) = \int_{-\infty}^{\infty} Pr(X_1 \leq X_2 | X_2 = \tau) \cdot f_{X_2}(\tau) \cdot d\tau =$$

$$= \int_{T_2}^{\infty} Pr(X_1 \leq X_2 | X_2 = \tau) \cdot \lambda_2 \cdot e^{-\lambda_2(\tau - T_2)} \cdot d\tau = \int_{T_2}^{\infty} Pr(X_1 \leq \tau) \cdot \lambda_2 \cdot e^{-\lambda_2(\tau - T_2)} \cdot d\tau.$$

Let $\nu = \tau - T_2$. Then, $d\nu = d\tau$ and

$$Pr(X_1 \leq X_2) = \int_{0}^{\infty} Pr(X_1 \leq (\nu + T_2)) \cdot \lambda_2 \cdot e^{-\lambda_2 \nu} \cdot d\nu. \tag{1}$$

Comparing algorithms with exponential runtime distributions

Using the formula of cumulative probability function of the random variable X_1 (see previous figure), we obtain

$$Pr(X_1 \leq (\upsilon + T_2)) = 1 - e^{-\lambda_1(\upsilon + T_2 - T_1)}. \tag{2}$$

Replacing (2) in (1) and solving the integral, we conclude that

$$Pr(X_1 \leq X_2) = 1 - e^{-\lambda_1(T_2 - T_1)} \cdot \frac{\lambda_2}{\lambda_1 + \lambda_2}. \tag{3}$$

This result can be better interpreted by rewriting expression (3) as

$$Pr(X_1 \leq X_2) = (1 - e^{-\lambda_1(T_2 - T_1)}) + e^{-\lambda_1(T_2 - T_1)} \cdot \frac{\lambda_1}{\lambda_1 + \lambda_2}. \tag{4}$$

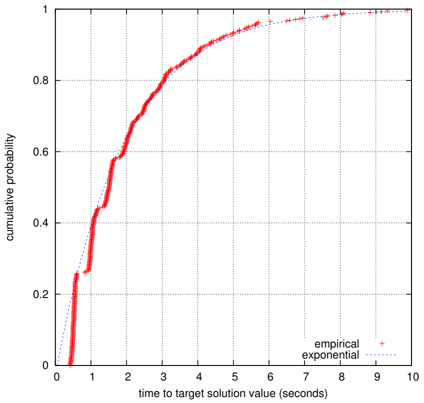
- The first term of the right-hand side of equation (4) is the probability that $0 \leq X_1 \leq T_2$, in which case X_1 is clearly less than or equal to X_2.
- The second term is given by the product of the factors $e^{-\lambda_1(T_2 - T_1)}$ and $\lambda_1/(\lambda_1 + \lambda_2)$, in which the former corresponds to the probability that $X_1 \geq T_2$ and the latter to the probability that X_1 be less than or equal to X_2, given that $X_1 \geq T_2$.

Comparing algorithms with exponential runtime distributions

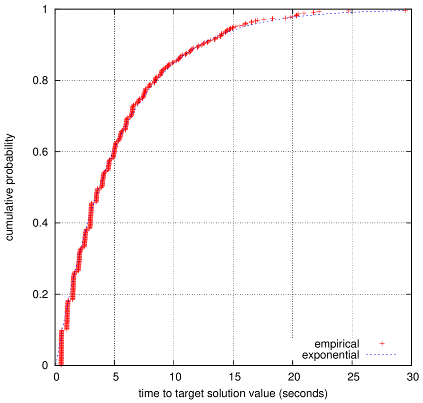
- To illustrate the above result, we consider two algorithms for solving the server replication for reliable multicast problem.
  - Algorithm A_1 is an implementation of pure GRASP with $\alpha = 0.2$.
  - Algorithm A_2 is a pure GRASP heuristic with $\alpha = 0.9$.
- The runs were performed on an Intel Core2 Quad with 2.40 GHz of clock speed and 4 GB of RAM memory.

Comparing algorithms with exponential runtime distributions

The figure depicts the runtime distributions of each algorithm, obtained after 500 runs with different seeds of an instance of the server replication for reliable multicast problem with $m = 25$ and the target value set at 2830.



(a) Pure GRASP with $\alpha = 0.2$

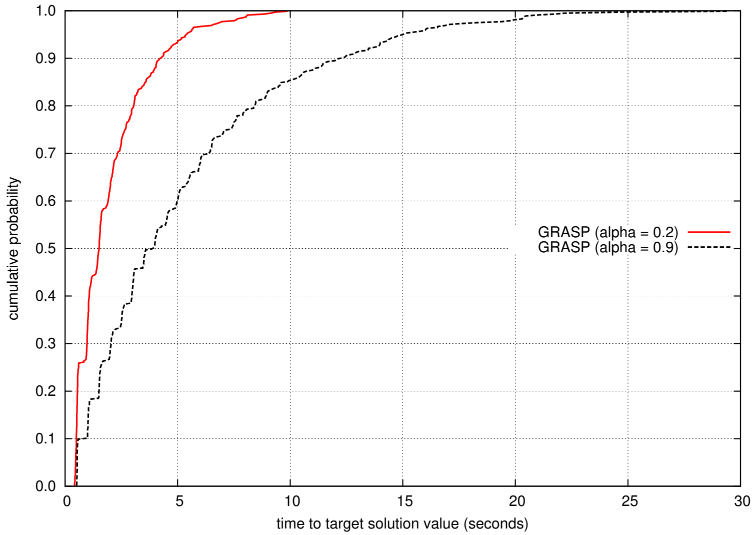


(b) Pure GRASP with $\alpha = 0.9$

Comparing algorithms with exponential runtime distributions

- The parameters of the two distributions are $\lambda_1 = 0.524422349$, $T_1 = 0.36$, $\lambda_2 = 0.190533895$, and $T_2 = 0.51$.
- Applying expression (3), we get $Pr(X_1 \leq X_2) = 0.684125$.
- This probability is consistent with the following figure, in which we superimposed the runtime distributions of the two pure GRASP heuristics for the same instance.

Comparing algorithms with exponential runtime distributions



The plots in this figure show that the pure GRASP with $\alpha = 0.2$ outperforms one with $\alpha = 0.9$, since the runtime distribution of the former is to the left of the runtime distribution of the latter.

Comparing algorithms with exponential runtime distributions

Observations:

- If the solution times do not fit exponential (or two-parameter shifted exponential) distributions, as for the case of GRASP with path-relinking heuristics, the closed form result established in expression (3) does not hold.
- Algorithms in this situation cannot be compared by this approach.
- The next topic extends this approach to general runtime distributions.

Comparing algorithms with general runtime distributions

Let X_1 and X_2 be two continuous random variables, with cumulative probability distributions $F_{X_1}(\tau)$ and $F_{X_2}(\tau)$ and probability density functions $f_{X_1}(\tau)$ and $f_{X_2}(\tau)$, respectively. Then,

$$Pr(X_1 \leq X_2) = \int_{-\infty}^{\infty} Pr(X_1 \leq \tau) \cdot f_{X_2}(\tau) \cdot d\tau = \int_{0}^{\infty} Pr(X_1 \leq \tau) \cdot f_{X_2}(\tau) \cdot d\tau,$$

since $f_{X_1}(\tau) = f_{X_2}(\tau) = 0$ for any $\tau < 0$. For an arbitrary small real number ε, the above expression can be rewritten as

$$Pr(X_1 \leq X_2) = \sum_{i=0}^{\infty} \int_{i\cdot\varepsilon}^{(i+1)\cdot\varepsilon} Pr(X_1 \leq \tau) \cdot f_{X_2}(\tau) \cdot d\tau. \tag{5}$$

Since $Pr(X_1 \leq i \cdot \varepsilon) \leq Pr(X_1 \leq \tau) \leq Pr(X_1 \leq (i+1) \cdot \varepsilon)$ for $i \cdot \varepsilon \leq \tau \leq (i+1) \cdot \varepsilon$, then replacing $Pr(X_1 \leq \tau)$ by $Pr(X_1 \leq i \cdot \varepsilon)$ and by $Pr(X_1 \leq (i+1) \cdot \varepsilon)$ in (5) leads to

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

Comparing algorithms with general runtime distributions

Let $L(\varepsilon)$ and $R(\varepsilon)$ be the value of the left and right hand sides of the above expression, respectively, with $\Delta(\varepsilon) = R(\varepsilon) - L(\varepsilon)$ being the difference between the upper and lower bounds to $Pr(X_1 \leq X_2)$.

Then, we have that

$$\Delta(\varepsilon) = \sum_{i=0}^{\infty} [F_{X_1}((i+1)\cdot\varepsilon) - F_{X_1}(i\cdot\varepsilon)] \int_{i\cdot\varepsilon}^{(i+1)\cdot\varepsilon} f_{X_2}(\tau)\cdot d\tau. \tag{6}$$

Let $\delta = \max_{\tau\geq 0}\{f_{X_1}(\tau)\}$. Since $|F_{X_1}((i+1)\cdot\varepsilon) - F_{X_1}(i\cdot\varepsilon)| \leq \delta\cdot\varepsilon$ for $i \geq 0$, expression (6) turns out to be

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

Consequently,

$$\Delta(\varepsilon) \leq \delta\cdot\varepsilon, \tag{7}$$

i.e., the difference $\Delta(\varepsilon)$ between the upper and lower bounds to $Pr(X_1 \leq X_2)$ (or the absolute error in the integration) is smaller than or equal to $\delta\varepsilon$.

Therefore, this difference can be made as small as desired by choosing a sufficiently small value for ε.

Comparing algorithms with general runtime distributions

In order to numerically evaluate a good approximation to $Pr(X_1 \leq X_2)$, we select the appropriate value of ε such that the resulting approximation error $\Delta(\varepsilon)$ is sufficiently small.

Next, we compute $L(\varepsilon)$ and $R(\varepsilon)$ to obtain the approximation

$$Pr(X_1 \leq X_2) \approx \frac{L(\varepsilon) + R(\varepsilon)}{2}. \tag{8}$$

In practice, the above probability distributions are unknown.

Instead of the distributions, the information available is limited to a sufficiently large number N_1 (resp. N_2) of observations of the random variable X_1 (resp. X_2).

Since the value of $\delta = \max_{\tau \geq 0}\{f_{X_1}(\tau)\}$ is also unknown beforehand, the appropriate value of ε cannot be estimated.

Then, we proceed iteratively as follows.

Comparing algorithms with general runtime distributions

Let $t_1(j)$ (resp. $t_2(j)$) be the value of the j-th smallest observation of the random variable X_1 (resp. X_2), for $j = 1, \ldots, N_1$ (resp. N_2).

We set the bounds $a = \min\{t_1(1), t_2(1)\}$ and $b = \max\{t_1(N_1), t_2(N_2)\}$ and choose an arbitrary number h of integration intervals to compute an initial value $\varepsilon = (b - a)/h$ for each integration interval.

For sufficiently small values of the integration interval ε, the probability density function $f_{X_1}(\tau)$ in the interval $[i \cdot \varepsilon, (i+1) \cdot \varepsilon]$ can be approximated by $\hat{f}_{X_1}(\tau) = (\hat{F}_{X_1}((i+1) \cdot \varepsilon) - \hat{F}_{X_1}(i \cdot \varepsilon))/\varepsilon$, where

$$\hat{F}_{X_1}(i \cdot \varepsilon) = |\{t_1(j), j = 1, \ldots, N_1 : t_1(j) \leq i \cdot \varepsilon\}|. \tag{9}$$

The same approximations hold for random variable X_2.

Finally, the value of $Pr(X_1 \leq X_2)$ can be computed as in expression (8), using the estimates $\hat{f}_{X_1}(\tau)$ and $\hat{f}_{X_2}(\tau)$ in the computation of $L(\varepsilon)$ and $R(\varepsilon)$.

If the approximation error $\Delta(\varepsilon) = R(\varepsilon) - L(\varepsilon)$ becomes sufficiently small, then the procedure stops.

Otherwise, the value of ε is halved and the above steps are repeated until convergence.

Numerical applications to sequential algorithms

We illustrate next an application of the procedure described in the previous topic for the comparison of randomized algorithms (running on the same instance) on three problems:

- server replication for reliable multicast,
- routing and wavelength assignment, and
- 2-path network design.

DM-D5 and GRASP algorithms for server replication

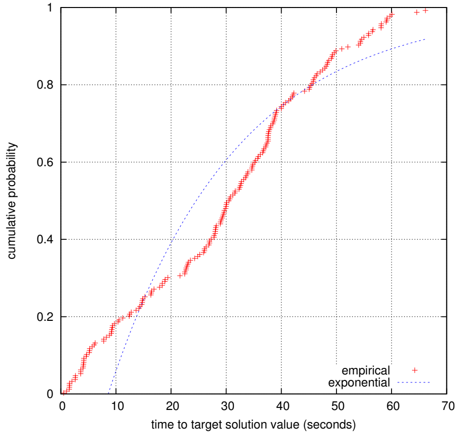
- Multicast communication consists of simultaneously delivering the same information to many receivers, from single or multiple sources.
- Network services specially designed for multicast are needed.
- The scheme used in current multicast services create a delivery tree, whose root represents the sender, whose leaves represent the receivers, and whose internal nodes represent network routers or relaying servers.
- Transmission is performed by creating copies of the data at split points of the tree.
- An important issue regarding multicast communication is how to provide reliable service, ensuring the delivery of all packets from the sender to receivers.
- A successful technique to provide reliable multicast service is the server replication approach, in which data is replicated at some of the multicast-capable relaying hosts (also called replicated or repair servers) and each of them is responsible for the retransmission of packets to receivers in its group.
- The problem consists in selecting the best subset of the multicast-capable relaying hosts to act as replicated servers in a multicast scenario.
- It is a special case of the *p*-median problem.

DM-D5 and GRASP algorithms for server replication

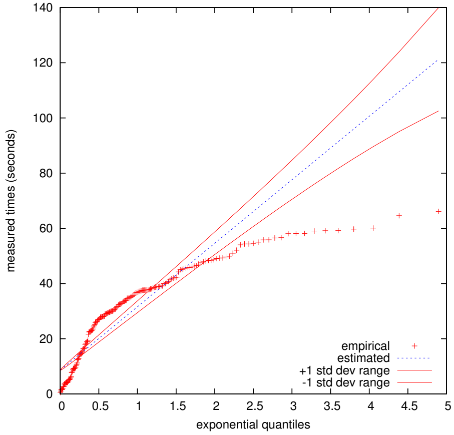
- DM-GRASP is a hybrid version of GRASP which incorporates a data-mining process.
- We compare two heuristics for the server replication problem:
  - algorithm A_1 is an implementation of the DM-D5 version of DM-GRASP, in which the mining algorithm is periodically applied,
  - while A_2 is a pure GRASP heuristic.
- We present results for two instances using the same network scenario, with $m = 25$ and $m = 50$ replication servers.
- Each algorithm was run 200 times with different seeds.
- The target was set at 2,818.925 (the best known solution value is 2,805.89) for the instance with $m = 25$ and at 2,299.07 (the best known solution value is 2,279.84) for the instance with $m = 50$.

DM-D5 and GRASP algorithms for server replication

The figure depicts the runtime distribution and quantile-quantile plot for algorithm DM-D5 on the instance with $m = 25$ and the target value set at 2,818.925.



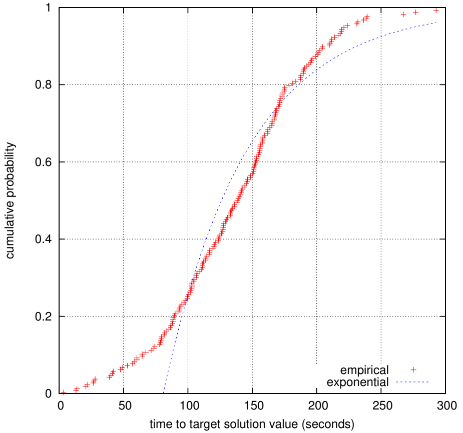
(a) Runtime distribution



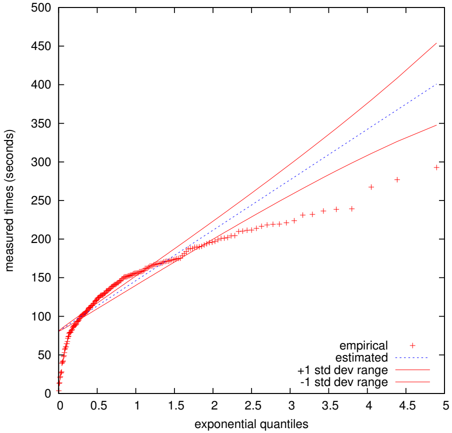
(b) Quantile-quantile plot

DM-D5 and GRASP algorithms for server replication

The figure depicts the runtime distribution and quantile-quantile plot for algorithm DM-D5 on the instance with $m = 50$ and the target value set at 2,299.07.



(a) Runtime distribution



(b) Quantile-quantile plot

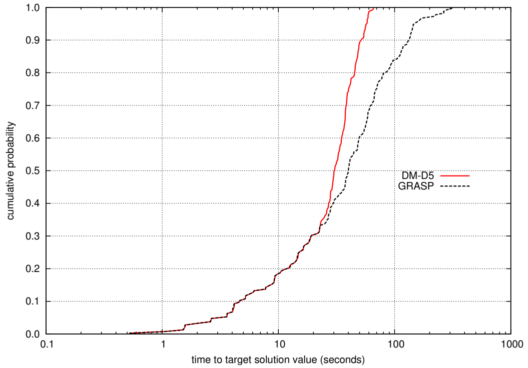
DM-D5 and GRASP algorithms for server replication

Observations:

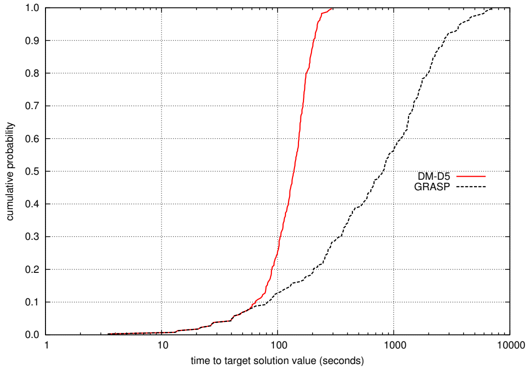
- Running times of DM-D5 did not fit exponential distributions for any of the instances.
- GRASP solution times were exponential for both.

DM-D5 and GRASP algorithms for server replication

The figure shows the superimposed empirical runtime distributions of DM-D5 and GRASP.



(a) $m = 25$ with target 2,818.925
$Pr(X_1 \leq X_2) = 0.619763$



(b) $m = 50$ with target 2,299.07
$Pr(X_1 \leq X_2) = 0.854113$

DM-D5 and GRASP algorithms for server replication

Observations:

- Algorithm DM-D5 outperformed GRASP, since the runtime distribution of the DM-D5 is to the left of the distribution for GRASP on the both instances, with $m = 25$ and $m = 50$.
- Consistently, the computations show that:
  - $Pr(X_1 \leq X_2) = 0.619763$ (with $L(\varepsilon) = 0.619450$, $R(\varepsilon) = 0.620075$, $\Delta(\varepsilon) = 0.000620$, and $\varepsilon = 0.009552$) for the instance with $m = 25$.
  - $Pr(X_1 \leq X_2) = 0.854113$ (with $L(\varepsilon) = 0.853800$, $R(\varepsilon) = 0.854425$, $\Delta(\varepsilon) = 0.000625$, and $\varepsilon = 0.427722$) for the instance with $m = 50$.

DM-D5 and GRASP algorithms for server replication

- We also investigate the convergence of the proposed measure with the sample size (i.e., with the number of independent runs of each algorithm).
- Convergence with the sample size is illustrated next for the same $m = 25$ instance of the server replication problem, with the same target 2,818.925 already used in the previous experiment.
- Once again, algorithm A_1 is the DM-D5 version of DM-GRASP and algorithm A_2 is the pure GRASP heuristic.
- The estimation of $Pr(X_1 \leq X_2)$ is computed for $N = 100, 200, 300, 400, 500, 600, 700, 800, 900,$ $1000, 2000, 3000, 4000,$ and 5000 independent runs of each algorithm.

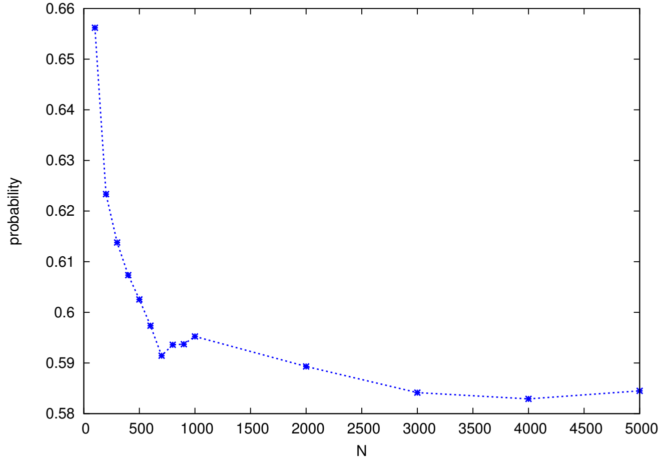
DM-D5 and GRASP algorithms for server replication

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the $m = 25$ instance of the server replication problem.

N	$L(\varepsilon)$	$Pr(X_1 \leq X_2)$	$R(\varepsilon)$	$\Delta(\varepsilon)$	ε
100	0.655900	0.656200	0.656500	0.000600	0.032379
200	0.622950	0.623350	0.623750	0.000800	0.038558
300	0.613344	0.613783	0.614222	0.000878	0.038558
400	0.606919	0.607347	0.607775	0.000856	0.038558
500	0.602144	0.602548	0.602952	0.000808	0.038558
600	0.596964	0.597368	0.597772	0.000808	0.038558
700	0.591041	0.591440	0.591839	0.000798	0.038558
800	0.593197	0.593603	0.594009	0.000812	0.042070
900	0.593326	0.593719	0.594113	0.000788	0.042070
1000	0.594849	0.595242	0.595634	0.000785	0.042070
2000	0.588913	0.589317	0.589720	0.000807	0.047694
3000	0.583720	0.584158	0.584596	0.000875	0.047694
4000	0.582479	0.582912	0.583345	0.000866	0.047694
5000	0.584070	0.584511	0.584953	0.000882	0.050604

DM-D5 and GRASP algorithms for server replication

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the $m = 25$ instance of the server replication problem.



Observation:

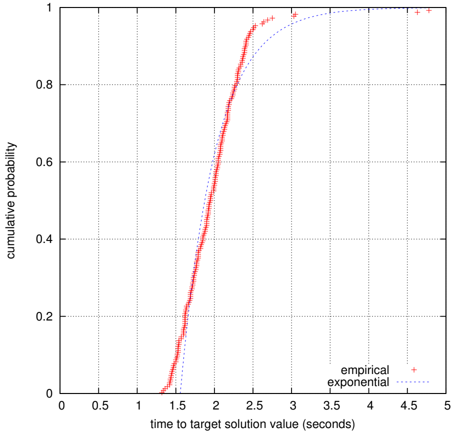
The estimation of $Pr(X_1 \leq X_2)$ stabilizes as the sample size N increases.

Multistart and tabu search for routing and wavelength assignment

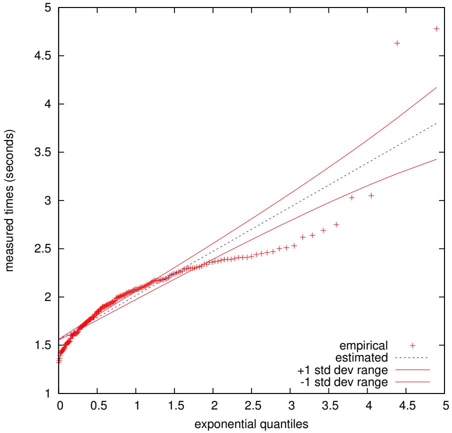
- A point-to-point connection between two endnodes of an optical network is called a lightpath.
- Two lightpaths may use the same wavelength, provided they do not share any common link.
- The routing and wavelength assignment problem is that of routing a set of lightpaths and assigning a wavelength to each of them, minimizing the number of wavelengths needed.
- A decomposition strategy is compared with a multistart greedy heuristic.
- Two networks are used for benchmarking:
  - The first has 27 nodes representing the capitals of the 27 states of Brazil, with 70 links connecting them. There are 702 lightpaths to be routed.
  - Instance Finland is formed by 31 nodes and 51 links, with 930 lightpaths to be routed.
- Each algorithm was run 200 times with different seeds.
- The target was set at 24 (the best known solution value) for instance Brazil and at 50 for instance Finland (the best known solution value is 47).
- Algorithm A_1 is the multistart heuristic, while A_2 is the tabu search decomposition scheme.
- The multistart solution times fit exponential distributions for both instances.

Multistart and tabu search for routing and wavelength assignment

The figure displays the runtime distribution and quantile-quantile plot for tabu search on Brazil instance with the target value set at 24.



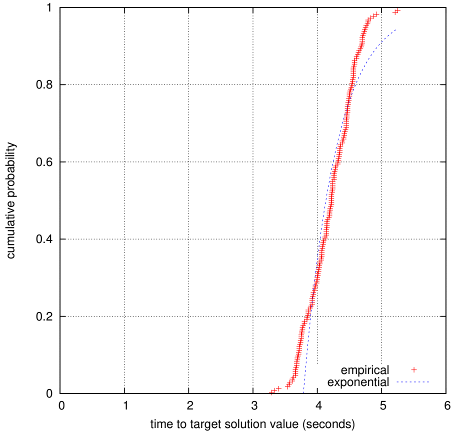
(a) Runtime distribution



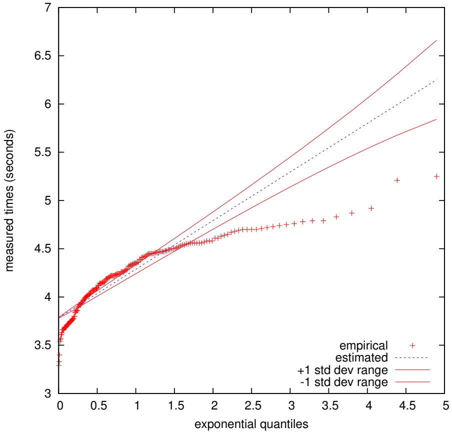
(b) Quantile-quantile plot

Multistart and tabu search for routing and wavelength assignment

The figure displays the runtime distribution and quantile-quantile plot for tabu search on Finland instance with the target value set at 50.



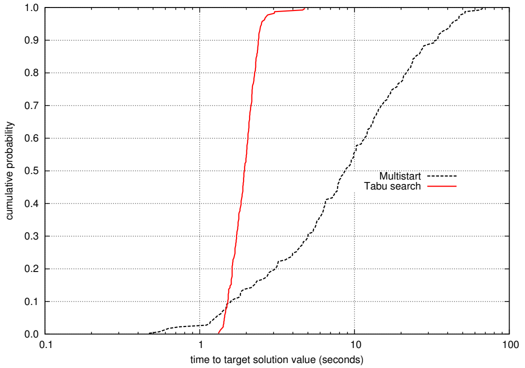
(a) Runtime distribution



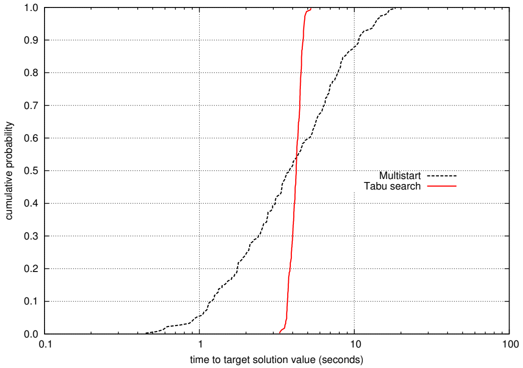
(b) Quantile-quantile plot

Multistart and tabu search for routing and wavelength assignment

The figure shows the superimposed empirical runtime distributions of multistart and tabu search algorithms.



(a) Brazil instance with target 24
$Pr(X_1 \leq X_2) = 0.13$



(b) Finland instance with target 50
$Pr(X_1 \leq X_2) = 0.536787$

Multistart and tabu search for routing and wavelength assignment

Observations:

- The direct comparison of the two approaches shows that decomposition clearly outperformed the multistart strategy for instance Brazil, since $Pr(X_1 \leq X_2) = 0.13$ in this case (with $L(\varepsilon) = 0.129650$, $R(\varepsilon) = 0.130350$, $\Delta(\varepsilon) = 0.000700$, and $\varepsilon = 0.008163$).
- However, the situation changes for instance Finland.
- Although both algorithms have similar performances, multistart is slightly better with respect to the measure proposed in this presentation, since $Pr(X_1 \leq X_2) = 0.536787$ (with $L(\varepsilon) = 0.536525$, $R(\varepsilon) = 0.537050$, $\Delta(\varepsilon) = 0.000525$, and $\varepsilon = 0.008804$).

Multistart and tabu search for routing and wavelength assignment

- As done for the server replication problem , we also investigate the convergence of the proposed measure with the sample size (i.e., with the number of independent runs of each algorithm).
- Convergence with the sample size is illustrated next for the Finland instance of the routing and wavelength assignment problem, with the target set at 49.
- Once again, algorithm A_1 is the multistart heuristic and algorithm A_2 is the tabu search decomposition scheme.
- The estimation of $Pr(X_1 \leq X_2)$ is computed for $N = 100$, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, and 5000 independent runs of each algorithm.

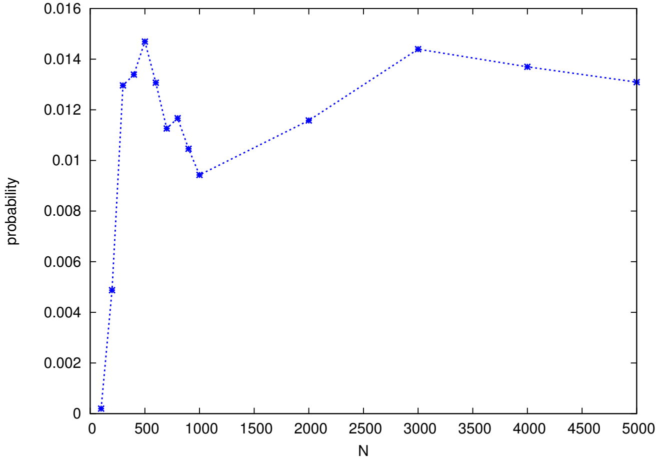
Multistart and tabu search for routing and wavelength assignment

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the Finland instance of the routing and wavelength assignment problem.

N	$L(\varepsilon)$	$Pr(X_1 \leq X_2)$	$R(\varepsilon)$	$\Delta(\varepsilon)$	ε
100	0.000001	0.000200	0.000400	0.000400	1.964844
200	0.000100	0.004875	0.009650	0.009550	0.000480
300	0.006556	0.012961	0.019367	0.012811	0.000959
400	0.007363	0.013390	0.019425	0.012063	0.000959
500	0.007928	0.014694	0.021460	0.013532	0.000610
600	0.006622	0.013069	0.019517	0.012894	0.000610
700	0.005722	0.011261	0.016800	0.011078	0.000610
800	0.005033	0.011667	0.018302	0.013269	0.000610
900	0.004556	0.010461	0.016367	0.011811	0.000610
1000	0.004100	0.009425	0.014750	0.010650	0.000610
2000	0.006049	0.011580	0.017112	0.011063	0.000610
3000	0.007802	0.014395	0.020987	0.013185	0.000610
4000	0.007408	0.013698	0.019988	0.012580	0.000610
5000	0.006791	0.013090	0.019389	0.012598	0.000623

Multistart and tabu search for routing and wavelength assignment

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the Finland instance of the routing and wavelength assignment problem.



Observation:

Once again, the estimation of $Pr(X_1 \leq X_2)$ stabilizes as the sample size N increases.

GRASP algorithms for 2-path network design

- Given a connected undirected graph with non-negative weights associated with its edges, together with a set of origin-destination nodes, the 2-path network design problem consists in finding a minimum weighted subset of edges containing a path formed by at most two edges between every origin-destination pair.

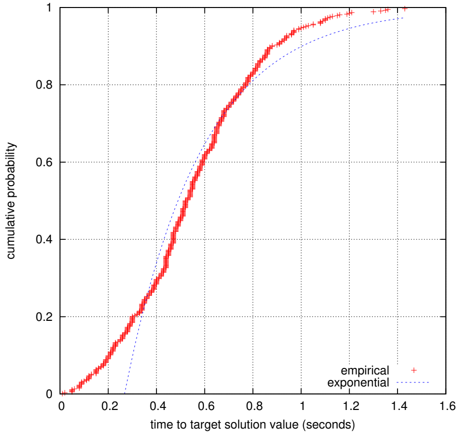
- Applications can be found in the design of communication networks, in which paths with few edges are sought to enforce high reliability and small delays.

GRASP algorithms for 2-path network design – Instance with 90 nodes

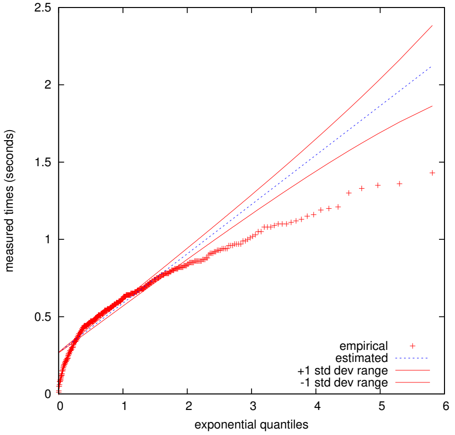
- We first compare four GRASP heuristics for solving an instance of the 2-path network design problem with 90 nodes.
  - The first heuristic is a pure GRASP (algorithm A_1).
  - The others integrate different path-relinking strategies for search intensification at the end of each GRASP iteration:
    - forward path-relinking (algorithm A_2),
    - bidirectional path-relinking (algorithm A_3), and
    - backward path-relinking (algorithm A_4).
- Each algorithm was run 500 independent times on the benchmark instance with 90 nodes and 900 origin-destination pairs, with the solution target value set at 673 (the best known solution value is 639).
- The runtime distributions and quantile-quantile plots for the different versions of GRASP with path-relinking are shown in the following figures.

GRASP algorithms for 2-path network design – Instance with 90 nodes

The figure shows the runtime distribution and quantile-quantile plot for GRASP with forward path-relinking on 90-node instance with the target value set at 673.



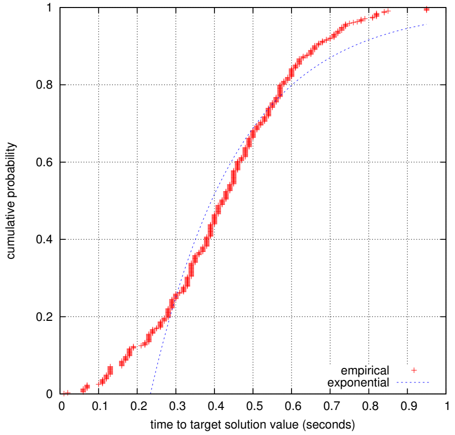
(a) Runtime distribution



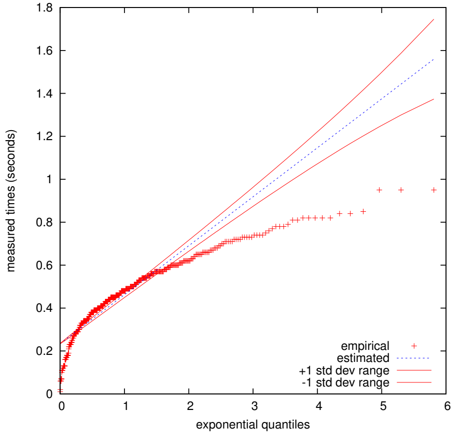
(b) Quantile-quantile plot

GRASP algorithms for 2-path network design – Instance with 90 nodes

The figure shows the runtime distribution and quantile-quantile plot for GRASP with bidirectional path-relinking on 90-node instance with the target value set at 673.



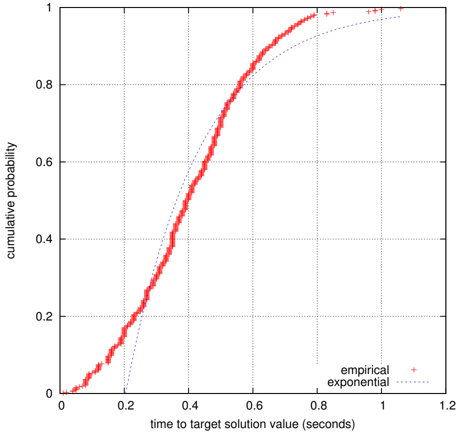
(a) Runtime distribution



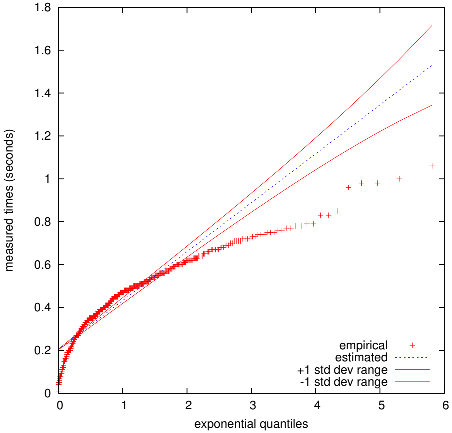
(b) Quantile-quantile plot

GRASP algorithms for 2-path network design – Instance with 90 nodes

The figure shows the runtime distribution and quantile-quantile plot for GRASP with backward path-relinking on 90-node instance with the target value set at 673.



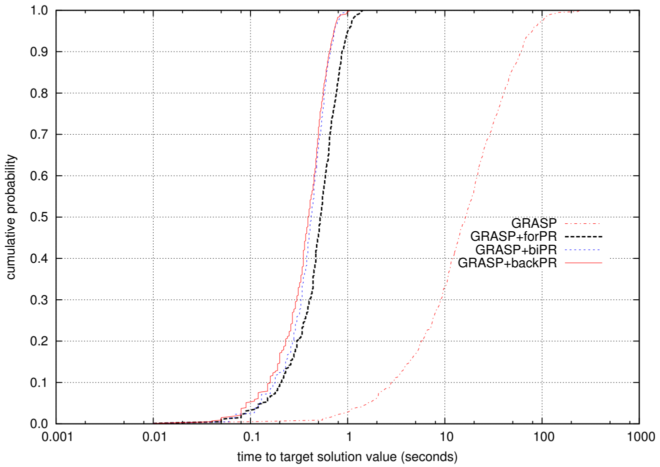
(a) Runtime distribution



(b) Quantile-quantile plot

GRASP algorithms for 2-path network design – Instance with 90 nodes

The figure shows the superimposed empirical runtime distributions of pure GRASP and three versions of GRASP with path-relinking.



GRASP algorithms for 2-path network design – Instance with 90 nodes

Observations:

- Algorithm A_2 (as well as A_3 and A_4) performs much better than A_1, as indicated by $Pr(X_2 \leq X_1) = 0.986604$ (with $L(\varepsilon) = 0.986212$, $R(\varepsilon) = 0.986996$, $\Delta(\varepsilon) = 0.000784$, and $\varepsilon = 0.029528$).
- Algorithm A_3 outperforms A_2, as illustrated by the fact that $Pr(X_3 \leq X_2) = 0.636000$ (with $L(\varepsilon) = 0.630024$, $R(\varepsilon) = 0.641976$, $\Delta(\varepsilon) = 0.011952$, and $\varepsilon = 1.354218 \times 10^{-6}$).
- Finally, we observe that algorithms A_3 and A_4 behave very similarly, although A_4 performs slightly better for this instance, since $Pr(X_4 \leq X_3) = 0.536014$ (with $L(\varepsilon) = 0.528560$, $R(\varepsilon) = 0.543468$, $\Delta(\varepsilon) = 0.014908$, and $\varepsilon = 1.001358 \times 10^{-6}$).

GRASP algorithms for 2-path network design – Instance with 90 nodes

- As for the problems considered in previous topics, we also investigate the convergence of the proposed measure as a function of sample size (i.e., with the number of independent runs of each algorithm).
- Convergence with the sample size is illustrated next for the 90-node instance of the 2-path network design problem, with the same target 673 previously used.
- We recall that algorithm A_1 is the GRASP with backward path-relinking heuristic, while algorithm A_2 is the GRASP with bidirectional path-relinking heuristic.
- The estimation of $Pr(X_1 \leq X_2)$ is computed for $N = 100, 200, 300, 400, 500, 600, 700, 800, 900$, 1000, 2000, 3000, 4000, and 5000 independent runs of each algorithm.

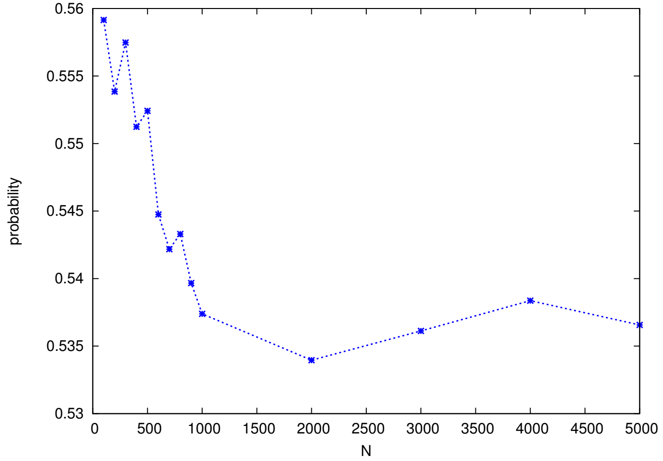
GRASP algorithms for 2-path network design – Instance with 90 nodes

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the 90-node instance of the 2-path network design problem.

N	$L(\varepsilon)$	$Pr(X_1 \leq X_2)$	$R(\varepsilon)$	$\Delta(\varepsilon)$	ε
100	0.553300	0.559150	0.565000	0.011700	4.387188×10^{-7}
200	0.553250	0.553850	0.554450	0.001199	4.501629×10^{-7}
300	0.551578	0.557483	0.563389	0.011811	4.501629×10^{-7}
400	0.545244	0.551241	0.557238	0.011994	4.730511×10^{-7}
500	0.546604	0.552420	0.558236	0.011632	5.035686×10^{-7}
600	0.538867	0.544749	0.550631	0.011764	5.073833×10^{-7}
700	0.536320	0.542181	0.548041	0.011720	5.073833×10^{-7}
800	0.537533	0.543298	0.549064	0.011531	5.073833×10^{-7}
900	0.533912	0.539671	0.545430	0.011517	5.073833×10^{-7}
1000	0.531595	0.537388	0.543180	0.011585	5.073833×10^{-7}
2000	0.528224	0.533959	0.539698	0.011469	5.722427×10^{-7}
3000	0.530421	0.536128	0.541835	0.011414	6.027603×10^{-7}
4000	0.532695	0.538364	0.544033	0.011338	6.027603×10^{-7}
5000	0.530954	0.536566	0.542178	0.011225	6.027603×10^{-7}

GRASP algorithms for 2-path network design – Instance with 90 nodes

Convergence of the estimation of $Pr(X_1 \leq X_2)$ with the sample size for the 90-node instance of the 2-path network design problem.



Observation:

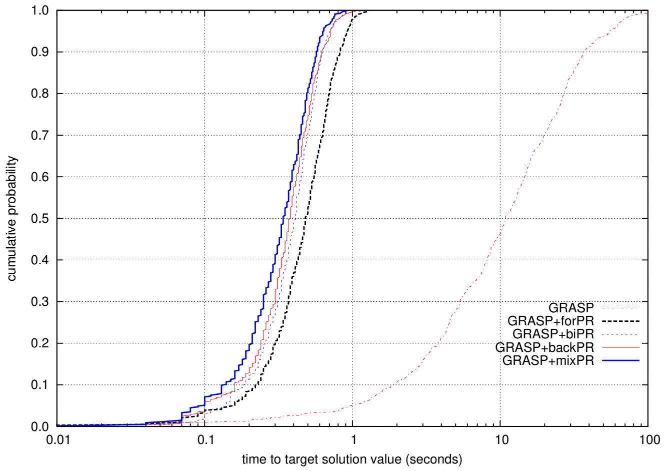
Once again, the estimation of $Pr(X_1 \leq X_2)$ stabilizes as the sample size N increases.

GRASP algorithms for 2-path network design – Instance with 80 nodes

- We next compare five GRASP heuristics for the 2-path network design problem, with and without path-relinking, for solving an instance with 80 nodes and 800 origin-destination pairs, with target value set at 588 (the best known solution value is 577).
- In this example, the first algorithm is a pure GRASP (algorithm A_1).
- The other heuristics integrate different path-relinking strategies at the end of each GRASP iteration:
  - forward path-relinking (algorithm A_2),
  - bidirectional path-relinking (algorithm A_3),
  - backward path-relinking (algorithm A_4), and
  - mixed path-relinking (algorithm A_5).
- As before, each heuristic was run independently 500 times.

GRASP algorithms for 2-path network design – Instance with 80 nodes

The figure shows the superimposed empirical runtime distributions of pure GRASP and four GRASP with path-relinking heuristics.



GRASP algorithms for 2-path network design – Instance with 80 nodes

Observations:

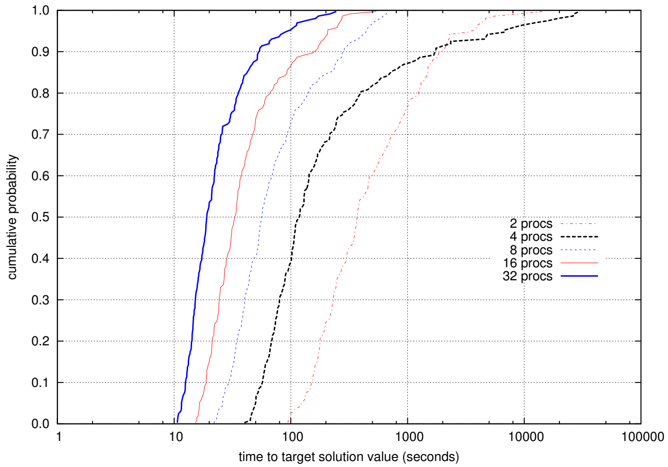
- Algorithm A_2 (as well as A_3, A_4, and A_5) performs much better than A_1, as indicated by $Pr(X_2 \leq X_1) = 0.970652$ (with $L(\varepsilon) = 0.970288$, $R(\varepsilon) = 0.971016$, $\Delta(\varepsilon) = 0.000728$, and $\varepsilon = 0.014257$).
- Algorithm A_3 outperforms A_2, as shown by the fact that $Pr(X_3 \leq X_2) = 0.617278$ (with $L(\varepsilon) = 0.610808$, $R(\varepsilon) = 0.623748$, $\Delta(\varepsilon) = 0.012940$, and $\varepsilon = 1.220703 \times 10^{-6}$).
- Algorithm A_4 performs slightly better than A_3 for this instance, since $Pr(X_4 \leq X_3) = 0.537578$ (with $L(\varepsilon) = 0.529404$, $R(\varepsilon) = 0.545752$, $\Delta(\varepsilon) = 0.016348$, and $\Delta(\varepsilon) = 1.201630 \times 10^{-6}$).
- Algorithms A_5 and A_4 also behave very similarly, but A_5 is slightly better for this instance since $Pr(X_5 \leq X_4) = 0.556352$ (with $L(\varepsilon) = 0.547912$, $R(\varepsilon) = 0.564792$, $\Delta(\varepsilon) = 0.016880$, and $\varepsilon = 1.001358 \times 10^{-6}$).

Comparing and evaluating parallel algorithms

- We conclude this presentation by describing the use of the runtime distribution methodology to evaluate and compare parallel implementations of stochastic local search algorithms.
- Once again, the 2-path network design problem is used to illustrate this application.
- The next two figures superimpose the runtime distributions of, respectively, cooperative and independent parallel implementations of GRASP with bidirectional path-relinking for the same problem on 2, 4, 8, 16, and 32 processors, on an instance with 100 nodes and 1000 origin-destination pairs, using 683 as target value.
- Each algorithm was run independently 200 times.
- We denote by A_1^k (resp. A_2^k) the cooperative (resp. independent) parallel implementation running on k processors, for $k = 2, 4, 8, 16, 32$.

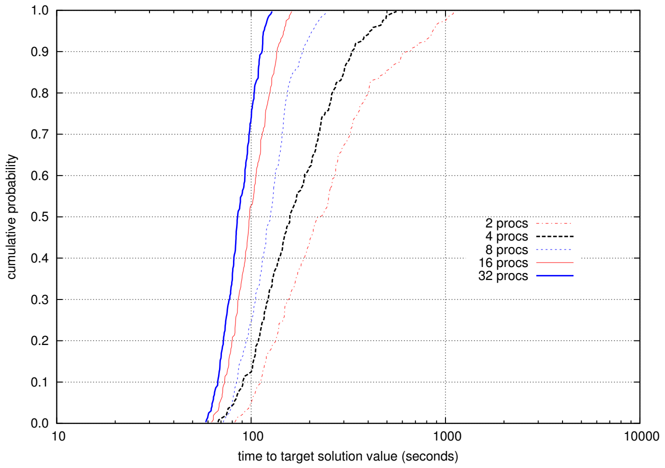
Comparing and evaluating parallel algorithms

The figure shows the superimposed empirical runtime distributions of cooperative parallel GRASP with bidirectional path-relinking running on 2, 4, 8, 16, and 32 processors.



Comparing and evaluating parallel algorithms

The figure shows the superimposed empirical runtime distributions of independent parallel GRASP with bidirectional path-relinking running on 2, 4, 8, 16, and 32 processors.



Comparing and evaluating parallel algorithms

Comparing cooperative (algorithm A_1) and independent (algorithm A_2) parallel implementations.

Processors (k)	$Pr(X_1^k \leq X_2^k)$
2	0.309784
4	0.597253
8	0.766806
16	0.860864
32	0.944938

Observations:

- The table shows the probability that the cooperative parallel implementation performs better than the independent implementation on 2, 4, 8, 16, and 32 processors.

- The independent implementation performs better than the cooperative implementation on two processors.
- In that case, the cooperative implementation does not benefit from the availability of two processors, since only one of them performs iterations, while the other acts as the master.
- However, as the number of processors increases from two to 32, the cooperative implementation performs progressively better than the independent implementation, since more processors are devoted to perform GRASP iterations.

Comparing and evaluating parallel algorithms

- The proposed methodology is clearly consistent with the relative behavior of the two parallel versions for any number of processors.
- Furthermore, it illustrates that the cooperative implementation becomes progressively better than the independent implementation when the number of processors increases.

Comparing and evaluating parallel algorithms

Comparing the parallel implementations on 2^{j+1} (algorithm A_1) and 2^j (algorithm A_2) processors, for $j = 1, 2, 3, 4$.

Processors (a)	Processors (b)	$Pr(X_1^a \leq X_1^b)$	$Pr(X_2^a \leq X_2^b)$
4	2	0.766235	0.651790
8	4	0.753904	0.685108
16	8	0.724398	0.715556
32	16	0.747531	0.669660

Observations:

- Both implementations scale appropriately as the number of processors grows.
- Once again, we can see that the performance measure appropriately describes the relative behavior of the two parallel strategies and provides insight on how parallel algorithms scale with the number of processors.
- The table shows numerical evidence to evaluate the trade-offs between computation times and the number of processors in parallel implementations.

Concluding remarks

The material in this talk is taken from

- Chapter 6 – Runtime distributions

of our book, *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures* (Resende & Ribeiro, Springer. 2016).