TTTPLOTS: A PERL PROGRAM TO CREATE TIME-TO-TARGET PLOTS

RENATA M. AIEX, MAURICIO G.C. RESENDE, AND CELSO C. RIBEIRO

ABSTRACT. This papers describes a perl language program to create time-to-target solution value plots for measured CPU times that are assumed to fit a shifted exponential distribution. This is often the case in local search based heuristics for combinatorial optimization, such as simulated annealing, genetic algorithms, iterated local search, tabu search, WalkSAT, and GRASP. Such plots are very useful in the comparison of different algorithms or strategies for solving a given problem and have been widely used as a tool for algorithm design and comparison. We first discuss how TTT plots are generated. This is followed by a description of the perl program `tttplots.pl`.

1. INTRODUCTION

It has been observed that in many implementations of local search based heuristics for combinatorial optimization problems, such as simulated annealing, genetic algorithms, iterated local search, tabu search, WalkSAT, and GRASP [4, 5, 11, 12, 23, 19, 31, 38, 43, 47], the random variable *time to target solution value* is exponentially distributed or fits a two-parameter shifted exponential distribution, i.e. the probability of not having found a given target solution value in t time units is given by $P(t) = e^{-(t-\mu)/\lambda}$, with $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}$. Hoos and Stützle [22, 23] conjecture that this is true for all local search based methods for combinatorial optimization.

Time-to-target (TTT) 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. TTT plots were used by Feo, Resende, and Smith [13] and have been advocated by Hoos and Stützle [18, 21] as a way to characterize the running times of stochastic algorithms for combinatorial optimization.

This paper describes a perl program to create time-to-target plots for measured CPU times that are assumed to fit a shifted exponential distribution. Such plots are very useful in the comparison of different algorithms or strategies for solving a given problem and have been widely used as a tool for algorithm design and comparison. In the next section, we discuss how TTT plots are generated, following closely Aiex, Resende, and Ribeiro [4]. The perl program `tttplots.pl` is described in Section 3 and can also be downloaded from `http://www.research.att.com/˜mgcr/tttplots`. A listing is given in the appendix. Section 4 presents an example and concluding remarks are made in Section 5.

2. TIME-TO-TARGET PLOTS

The hypothesis here is that CPU times fit a two parameter, or shifted, exponential distribution. For a given problem instance, we measure the CPU time to find a solution with an objective function value at least as good as a given target value. The heuristic is run n times on the fixed instance and using the given target solution value. For each of the n runs,

Date: November 3, 2005.
AT&T Labs Research Technical Report: TD-6HT7EL.

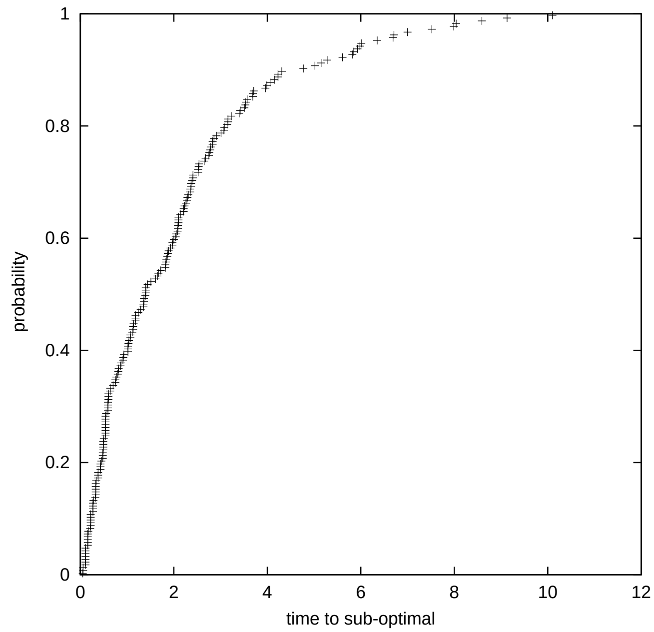

FIGURE 1. Cumulative probability distribution plot of measured data.

the random number generator is initialized with a distinct seed and, therefore, the runs are assumed to be independent. To compare the empirical and the theoretical distributions, we follow a standard graphical methodology for data analysis [7]. This methodology is used to produce the TTT plots. In the remainder of this section we describe this methodology.

For each instance/target pair, the running times are sorted in increasing order. We associate with the i-th sorted running time t_i a probability $p_i = (i-1/2)/n$, and plot the points $z_i = [t_i, p_i]$, for $i = 1,\ldots,n$. Figure 1 illustrates this cumulative probability distribution plot for a instance/target pair obtained by repeatedly applying a GRASP heuristic to find a solution with objective function value at least as good as a given target value. In this figure, we see that the probability of the heuristic finding a solution at least as good as the target value in at most 2 seconds is about 50%, in at most 4 seconds is about 80%, and in at most 6 seconds is about 90%.

The plot in Figure 1 appears to fit a shifted exponential distribution. We would like to estimate the parameters of the two-parameter exponential distribution. To do this, 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 $F(t) = 1 - e^{-(t-\mu)/\lambda}$, where λ is the mean of the distribution data (and also is the standard deviation of the data) and μ is the shift of the distribution with respect to the ordinate axis.

For each value p_i, $i = 1,\ldots,n$, we associate a p_i-quantile $Qt(p_i)$ of the theoretical distribution. For each p_i-quantile we have, by definition, that $F((Qt(p_i)) = p_i$. Hence, $Qt(p_i) = F^{-1}(p_i)$ and therefore, for the two-parameter exponential distribution, we have $Qt(p_i) = -\lambda \ln(1-p_i) + \mu$. The quantiles of the data of an empirical distribution are simply the (sorted) raw data.

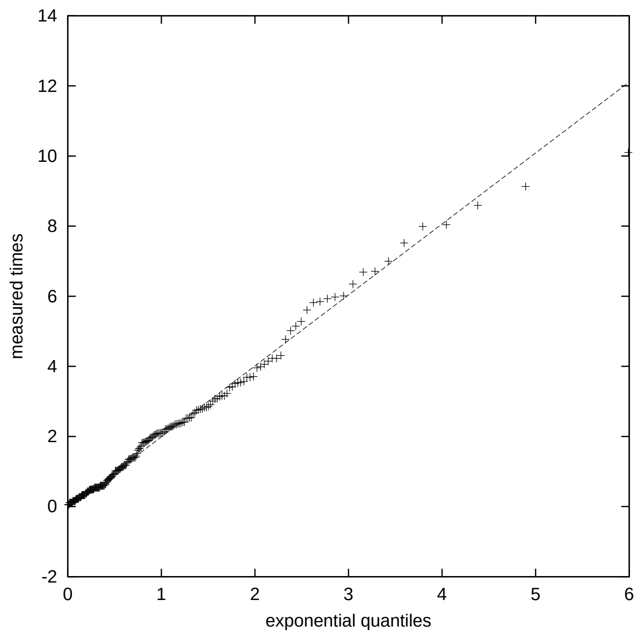

FIGURE 2. Q-Q plot showing fitted line.

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 our case, the measured 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 $y = \hat{\lambda}x + \hat{\mu}$. 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.

The Q-Q plot shown in Figure 2 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 $-\ln(1-p_i)$ for $i = 1, \ldots, n$. 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 $y = \lambda x + \mu$ using the upper quartile q_u and lower quartile q_l of the data. The upper and lower quartiles 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 [7]. Consequently, the estimate for the shift is $\hat{\mu} = z_l - \hat{\lambda} q_l$. 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, $i = 1, \ldots, n$, of the Q-Q plot is $\hat{\sigma} =$

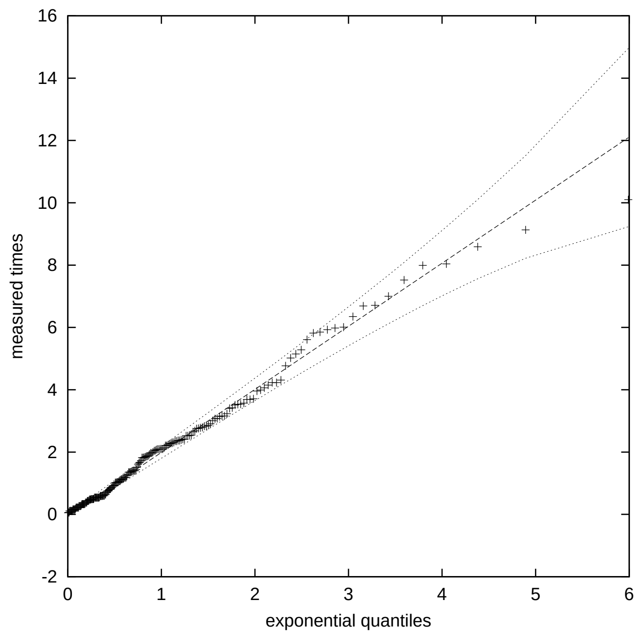

FIGURE 3. Q-Q plot with variability information.

$\hat{\lambda}[p_i/(1-p_i)n]^{\frac{1}{2}}$. Figure 3 shows an example of a Q-Q plot with superimposed variability information.

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 at least one of the data points deviates from the line by two standard deviations is probably much greater. 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 can be observed in the Q-Q plot in Figure 3 that the standard deviation of the points near the high end are substantially larger than the standard deviation of the other end.

Once the two parameters of the distribution are estimated, a superimposed plot of the empirical and theoretical distributions can be made. Figure 4 shows this plot corresponding to the Q-Q plot in Figure 3.

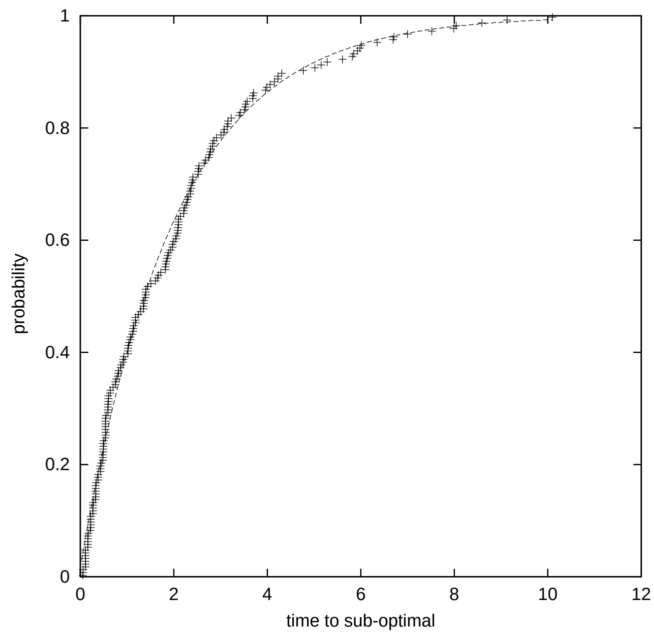

FIGURE 4. Superimposed empirical and theoretical distributions.

3. THE PERL PROGRAM

`tttplots.pl` [1] is a perl program that takes as input a file with with CPU times. To be able to produce the plots, `tttplots.pl` requires that `gnuplot` [2] be installed.

To run `tttplots.pl`, simple type: `perl tttplots.pl -f input_filename` where `input_filename.dat` is the input data file with n CPU time data points, one time point per line.

Two plots are produced by `ttplots.pl`:

(1) Q-Q plot with superimposed variability information (as in Figure 3); and
(2) Superimposed empirical and theoretical distributions (as in Figure 4).

Besides printing to the standard output some basic statistics of the data file and the estimated parameters, `tttplots.pl` also creates some output files. A list of the files produced by `tttplots.pl` is shown in Table 1.

4. AN EXAMPLE

In this section, we show an example of the plots produced by `tttplots.pl`. We ran the GRASP with path-relinking heuristic for the MAX-CUT problem described in [16] on instance G13 with a target solution value of 572. We produce plots after 10, 20, 30, 50, 75, 100, 125, 150, and 200 runs. These plots are shown in Figures 5, 6, and 7.

We notice that the larger is the number of runs n (i.e. the number of points plotted), the closer the empirical distribution is to the theoretical distribution. We have observed in practice that using $n = 200$ gives very good approximations of the theoretical distributions.

[1] `tttplots.pl` can be downloaded from `http://www.research.att.com/~mgcr/tttplots`.
[2] `gnuplot` can be downloaded from the gnuplot homepage at `http://www.gnuplot.info`.

TABLE 1. Files produced by `tttplots.pl`.

empirical exponential distribution data file	`input_filename-ee.dat`
theoretical exponential distribution data file	`input_filename-te.dat`
empirical QQ-plot data file	`input_filename-el.dat`
theoretical QQ-plot data file	`input_filename-tl.dat`
theoretical upper 1 standard deviation QQ-plot data	`input_filename-ul.dat`
theoretical lower 1 standard deviation QQ-plot data	`input_filename-ll.dat`
theoretical vs empirical TTT plot gnuplot file	`input_filename-exp.gpl`
theoretical vs empirical QQ-plot gnuplot file	`input_filename-qq.gpl`
theoretical vs empirical TTT plot PostScript file	`input_filename-exp.ps`
theoretical vs empirical QQ-plot PostScript file	`input_filename-qq.ps`

Furthermore, we also notice that the use of "easy" target solution values should be discouraged, since in this case the CPU times are very small in almost all runs and the exponential distribution degenerates to a step function.

5. CONCLUDING REMARKS

In this paper, we described a perl language program to create time-to-target plots from a set of running times that are exponentially distributed.

Most time-to-target plots seen in the literature are created from a set of repeated runs of an algorithm on a fixed problem instance. An exception to this was in Feo, Resende, and Smith [13], where the time-to-target plots were created by running an algorithm a single time on many randomly generated instances having a fixed characteristic (e.g. size and density).

Besides being used to help establish the probability distribution of time-to-target random variables for various stochastic algorithms [1, 2, 3, 4, 9, 19, 24, 32, 34, 35, 42], TTT plots have been used in a number of studies to analyze the comparison of

- different heuristics [1, 2, 6, 8, 10, 14, 15, 16, 20, 24, 27, 30, 34, 35, 36, 37, 41, 42, 44, 45];
- parallel implementations using different number of processors or parallelization strategies [1, 2, 3, 28, 34, 35];
- the same algorithm on several instances [1, 2, 13, 17, 23, 29, 42];
- algorithms using different strategies [2, 28, 33, 34, 35, 39, 40, 42, 45, 46]; and
- an algorithm using different parameter settings [6, 25, 26, 39, 40, 46].

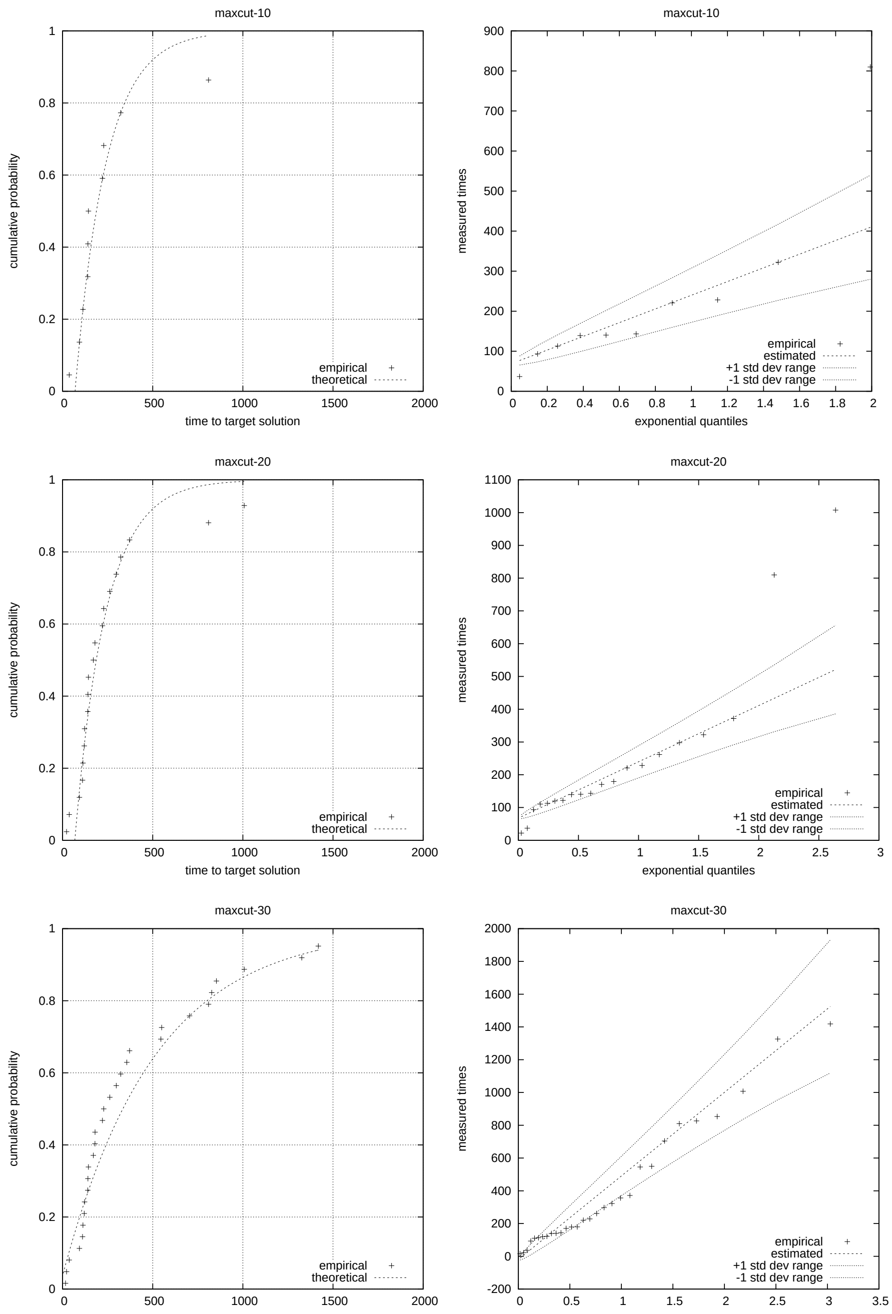

FIGURE 5. Empirical versus theoretical distributions on left and QQ-plots with variability information on right: 10, 20, and 30 data points.

REFERENCES

[1] R.M. Aiex, S. Binato, and M.G.C. Resende. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29:393–430, 2003.

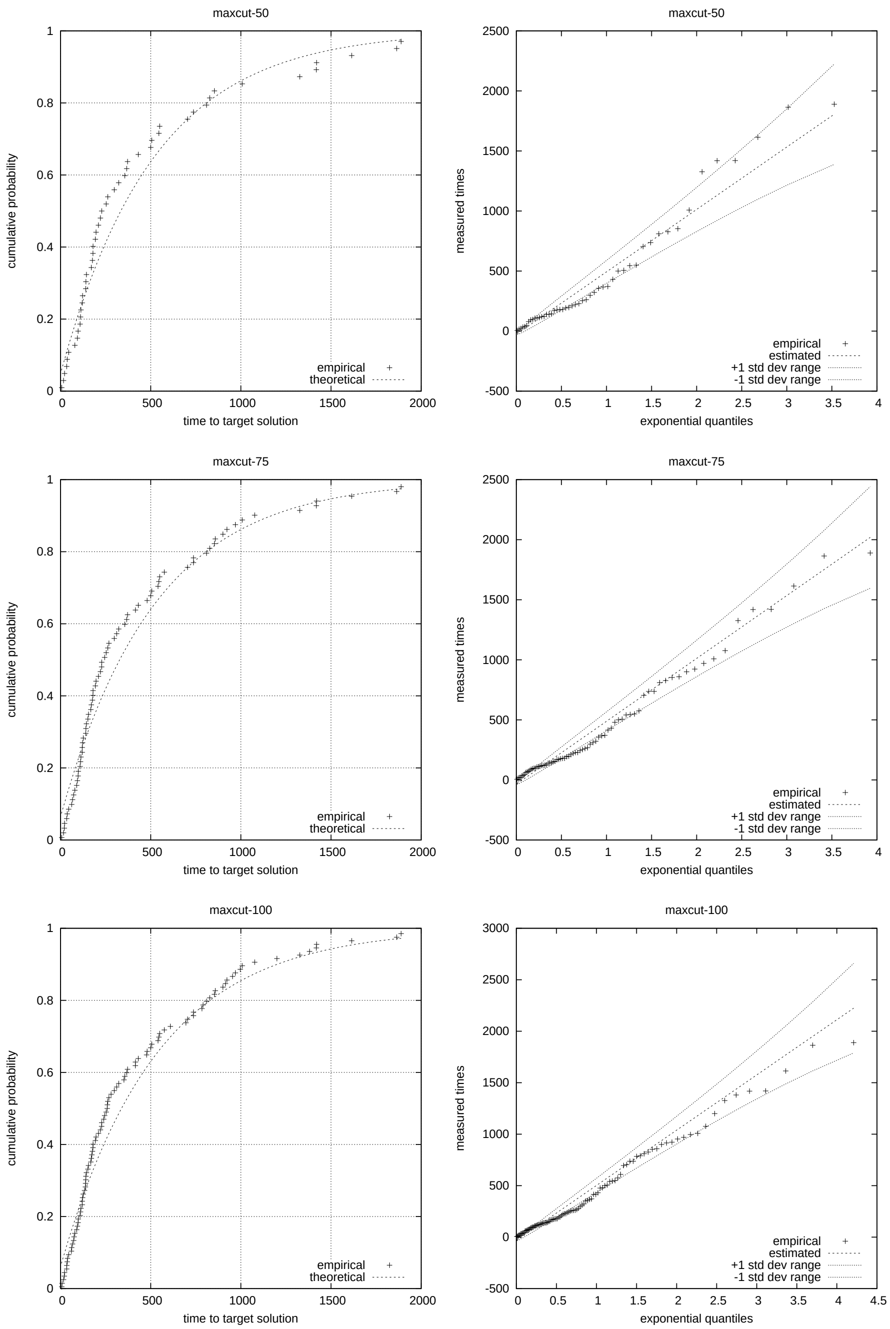

FIGURE 6. Empirical versus theoretical distributions on left and QQ-plots with variability information on right: 50, 75, and 100 data points.

[2] R.M. Aiex, P.M. Pardalos, M.G.C. Resende, and G. Toraldo. GRASP with path relinking for three-index assignment. *INFORMS J. on Computing*, 17:224–247, 2005.

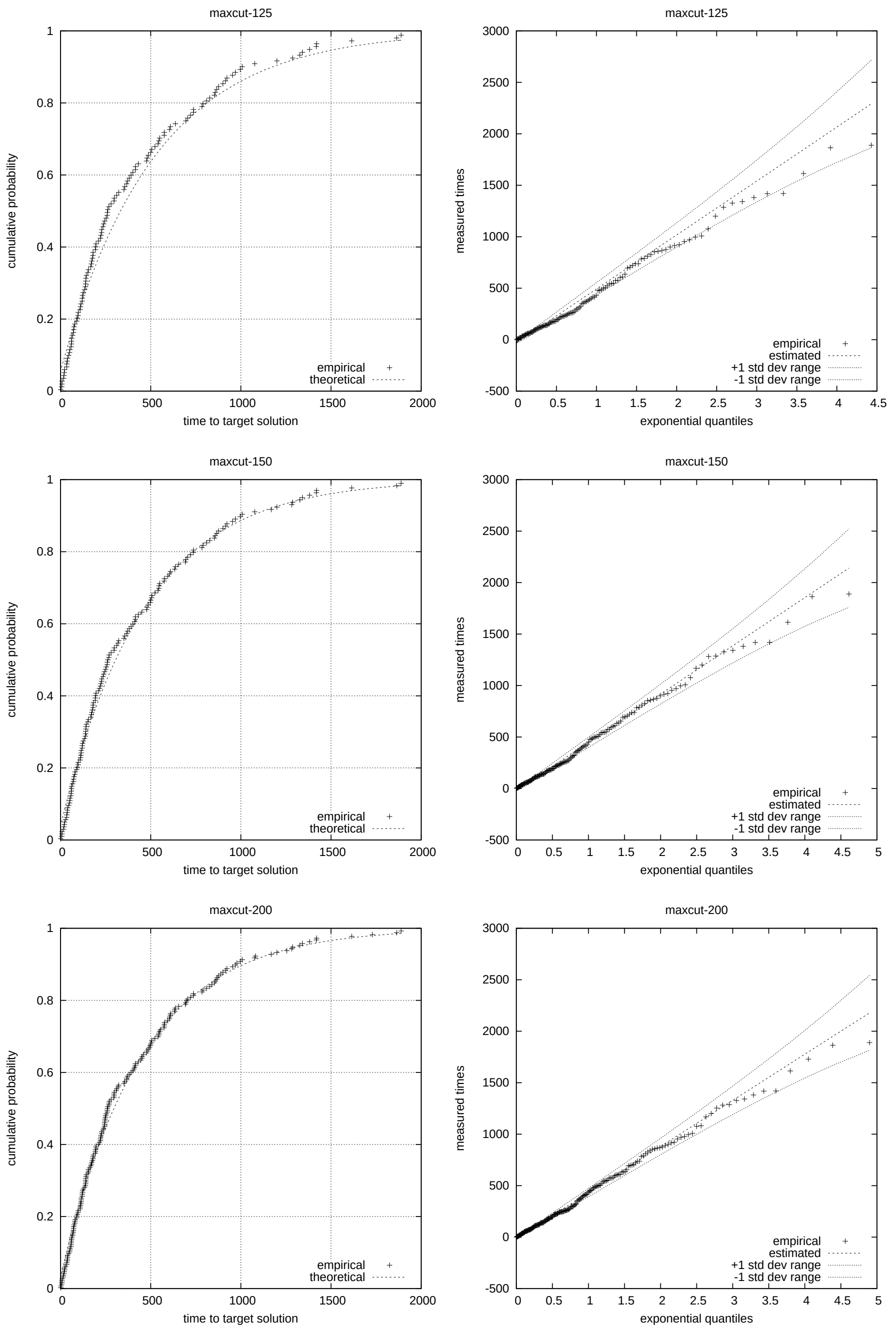

FIGURE 7. Empirical versus theoretical distributions on left and QQ-plots with variability information on right: 125, 150, and 200 data points.

[3] R.M. Aiex and M.G.C. Resende. Parallel strategies for GRASP with path-relinking. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 301–331. Springer, 2005.

[4] R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. *J. of Heuristics*, 8:343–373, 2002.

[5] R. Battiti and G. Tecchiolli. Parallel biased search for combinatorial optimization: Genetic algorithms and TABU. *Microprocessors and Microsystems*, 16:351–367, 1992.

[6] L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46:36–56, 2005.

[7] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Chapman & Hall, 1983.

[8] M. Chiarandini and T. Stützle. Experimental evaluation of course timetabeling algorithms. Technical Report AIDA-02-05, Fachgebiet Intellektik, Fachbereich Informatik Technische Universität Darmstadt, 2002.

[9] I. Czarnowski and P. Jędrzejowicz. Probability distribution of solution time in ANN training using population learning algorithm. In L. Rutkowski et al., editor, *ICAISC 2004*, volume 3070 of *Lecture Notes in Artificial Intelligence*, pages 172–177. Springer-Verlag, 2004.

[10] M.R.Q. de Andrade, P.M.F. de Andrade, S.L. Martins, and A. Plastino. GRASP with path-relinking for the maximum diversity problem. In S.E. Nikoletseas, editor, *WEA 2005*, volume 3503 of *Lecture Notes in Computer Science*, pages 558–569. Springer-Verlag, 2005.

[11] N. Dodd. Slow annealing versus multiple fast annealing runs: An empirical investigation. *Parallel Computing*, 16:269–272, 1990.

[12] H.M.M. Ten Eikelder, M.G.A. Verhoeven, T.W.M. Vossen, and E.H.L. Aarts. A probabilistic analysis of local search. In I.H. Osman and J.P. Kelly, editors, *Metaheuristics: Theory & applications*, pages 605–618. Kluwer, 1996.

[13] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[14] E.R. Fernandes and C.C. Ribeiro. Using an adaptive memory strategy to improve a multistart heuristic for sequencing by hybridization. In S.E. Nikoletseas, editor, *WEA 2005*, volume 3503 of *Lecture Notes in Computer Science*, pages 4–15. Springer-Verlag, 2005.

[15] P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted maximum satisfiability problem. In S.E. Nikoletseas, editor, *WEA 2005*, volume 3503 of *Lecture Notes in Computer Science*, pages 367–379. Springer-Verlag, 2005.

[16] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

[17] I.P. Gent, H.H. Hoos, A.G.D. Rowley, and K. Smyth. Using stochastic local search to solve quantified Boolean formulae. In F. Rossi, editor, *CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 348–362. Springer-Verlag, 2003.

[18] H. Hoos and T. Stützle. On the empirical evaluation of Las Vegas algorithms - Position paper. Technical report, Computer Science Department, University of British Columbia, 1998.

[19] H.H. Hoos. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proc. AAAI-99*, pages 661–666. MIT Press, 1999.

[20] H.H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the Seventeenth Conference on Artificial Intelligence (AAAI 2000)*, pages 22–29, Austin, Texas, 2000. MIT Press.

[21] H.H. Hoos and T. Stützle. Evaluation Las Vegas algorithms - Pitfalls and remedies. In *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, pages 238–245, 1998.

[22] H.H. Hoos and T. Stützle. Some surprising regularities in the behaviour of stochastic local search. In M. Maher and J.-F. Puget, editors, *CP'98*, volume 1520 of *Lecture Notes in Computer Science*, page 470. Springer-Verlag, 1998.

[23] H.H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, 112:213–232, 1999.

[24] H.H. Hoos and T. Stützle. Local search algorithms for SAT: An empirical evaluation. *J. of Automated Reasoning*, 24:421–481, 2000.

[25] F. Hutter. Stochastic local search for solving the most probable explanation problem in Bayesian networks. Master's thesis, Computer Science Department, Darmstadt University of Technology, 2004.

[26] F. Hutter, D.A.D. Tompkins, and H.H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In P. Van Hentenryck, editor, *CP 2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 233–248. Springer-Verlag, 2002.

[27] E.H. Marinho. Heurísticas busca tabu para o problema de programação de tripulações de ônibus urbanos. Master's thesis, Universidade Federal Fluminense, Niterói, Brazil, 2005. In Portuguese.

[28] S.L. Martins, C.C. Ribeiro, and I. Rosseti. Applications and parallel implementations of metaheuristics in network design and routing. In S. Manandhar et al., editor, *AACC 2004*, volume 3285 of *Lecture Notes in Computer Science*, pages 205–213. Springer-Verlag, 2004.

[29] E. Nudelman, K. Leyton-Brown, H.H. Hoos, A. Devkar, and Y. Shoham. Understanding random SAT: Beyond the clauses-to-variables ratio. In M. Wallace, editor, *CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 438–452. Springer-Verlag, 2004.

[30] C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende. GRASP with path-relinking for the quadratic assignment problem. In C.C. Ribeiro and S.L. Martins, editors, *Efficient and Experimental Algorithms – WEA2004*, volume 3059 of *Lecture Notes in Computer Science*, pages 356–368. Springer-Verlag, 2004.

[31] L.J. Osborne and B.E. Gillett. A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. *ORSA J. on Computing*, 3:213–225, 1991.

[32] M.G.C. Resende and J.L. Gonzalez Velarde. GRASP: Procedimientos de búsqueda miope aleatorizado y adaptativo. *Inteligencia Artificial*, 19:61–76, 2003.

[33] M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003.

[34] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer, 2003.

[35] M.G.C. Resende and C.C. Ribeiro. Parallel Greedy Randomized Adaptive Search Procedures. In E. Alba, editor, *Parallel Metaheuristics: A new class of algorithms*, pages 315–346. Wiley, 2005.

[36] H.G. Santos, L.S. Ochi, and J.F. Souza M. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. In E.K. Burke and M. Trick, editors, *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT '04)*, pages 343–358, 2004.

[37] H.G. Santos, L.S. Ochi, and M.J.F. Souza. An efficient tabu search heuristic for the school timetabling problem. In C.C. Ribeiro and S.L. Martins, editors, *Efficient and Experimental Algorithms – WEA2004*, volume 3059 of *Lecture Notes in Computer Science*, pages 468–481. Springer-Verlag, 2004.

[38] B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the AAAI-94*, pages 337–343. MIT Press, 1994.

[39] A. Shmygelska, R. Aguirre-Hernández, and H.H. Hoos. An ant colony optimization algorithm for the 2D HP protein folding problem. In M. Dorigo et al., editor, *ANTS 2002*, volume 2463 of *Lecture Notes in Computer Science*, pages 40–52. Springer-Verlag, 2002.

[40] A. Shmygelska and H.H. Hoos. An improved ant colony optimisation algorithm for the 2D HP protein folding problem. In Y. Xiang and B. Chaib-draa, editors, *AI 2003*, volume 2671 of *Lecture Notes in Artificial Intelligence*, pages 400–417. Springer-Verlag, 2003.

[41] G.C. Silva, L.S. Ochi, and S.L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In C.C. Ribeiro and S.L. Martins, editors, *Efficient and Experimental Algorithms – WEA2004*, volume 3059 of *Lecture Notes in Computer Science*, pages 498–512. Springer-Verlag, 2004.

[42] T. Stützle and H.H. Hoos. Analyzing the run-time behaviour of iterated local search for the TSP. Technical Report IRIDIA/2000-01, IRIDIA, Université Libre de Bruxelles, 2000.

[43] E.D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[44] D.A.D. Tompkins and H.H. Hoos. Scaling and probabilistic smoothing: Dynamic local search for unweighted MAX-SAT. In Y. Xiang and B. Chaib-draa, editors, *AI 2003*, volume 2671 of *Lecture Notes in Artificial Intelligence*, pages 145–159. Springer-Verlag, 2003.

[45] D.C. Tulpan and H.H. Hoos. Hybrid randomised neighbourhoods improve stochastic local search for DNA code design. In Y. Xiang and B. Chaib-draa, editors, *AI 2003*, volume 2671 of *Lecture Notes in Artificial Intelligence*, pages 418–433. Springer-Verlag, 2003.

[46] D.C. Tulpan, H.H. Hoos, and E. Condon A. Stochastic local search algorithms for DNA word design. In M. Hagiya and A. Ohuchi, editors, *DNA8*, volume 2568 of *Lecture Notes in Computer Science*, pages 229–241. Springer-Verlag, 2003.

[47] M.G.A. Verhoeven and E.H.L. Aarts. Parallel local search. *J. of Heuristics*, 1:43–66, 1995.

APPENDIX: PROGRAM LISTING

```
#!/usr/bin/perl
##
#       ----------------------------------------------------------------------------
```

```
##
##      tttplots: A Perl program for generating empirical vs
##                theoretical distributions of time-to-target-value
##                plot and corresponding QQ-plots with variability
##                information.
##
##      usage: perl tttplots.pl -f <input-file>
##
##             where <input-file>.dat is the input file of time to
##                   target values (one per line).
##
##
##      authors: Renata M. Aiex, Mauricio G. C. Resende, and
##               Celso C. Ribeiro
##
#       --------------------------------------------------------------------------------
##

##
#       --------------------------------------------------------------------------------
##      Input file name.
#       --------------------------------------------------------------------------------
##
        $datafilethere=0;
        while ($ARGV[0]) {
              if ($ARGV[0] eq "-f") {
                     shift(@ARGV);
                     $filename = $ARGV[0];
                     $datafilename = $filename . ".dat";
                     $datafilethere=1;
                     shift;
              }

        }
        if ($datafilethere == 0) {
              die "Error, data file missing. \nUsage:
                     perl tttplots.pl -f datafile.dat -o outputfile.out \n";
        }

##
#       --------------------------------------------------------------------------------
#       Name output files.
#       --------------------------------------------------------------------------------
##
        $emp_lin_filename = $filename . "-el" . ".dat";
        $the_lin_filename = $filename . "-tl" . ".dat";
        $up_lin_filename = $filename . "-ul" . ".dat";
        $lo_lin_filename = $filename . "-ll" . ".dat";
        $gpl_lin_filename = $filename . "-qq" . ".gpl";
        $ps_lin_filename = $filename . "-qq" . ".ps";
        $emp_exp_filename = $filename . "-ee" . ".dat";
        $the_exp_filename = $filename . "-te" . ".dat";
        $gpl_exp_filename = $filename . "-exp" . ".gpl";
        $ps_exp_filename = $filename . "-exp" . ".ps";

##
#       --------------------------------------------------------------------------------
#       Open input data file.
```

```
#       -------------------------------------------------------------------------------
##
        open (DATFILE,$datafilename) ||
               die "Cannot open file: $datafilename \n";

##
#       -------------------------------------------------------------------------------
#       Read input data file.  Require that data file have one value per line.
#       -------------------------------------------------------------------------------
##
        $n=0;
        while ($line = <DATFILE>){
              chomp($line);
              @field = split(/\s+/,$line);

              $nfields=0;                #
              foreach $fld (@field){     # count number of fields
                     $nfields++;         #
              }                          #

              if ($nfields != 1){
                     die "Number of fields in data file must be 1 \n";
              }
              $time_value[$n] = $field[0];
              $n++;
        }
        close (DATFILE);

##
#       -------------------------------------------------------------------------------
#       Sort times.
#       -------------------------------------------------------------------------------
##
        @sorted_time_value = sort { $a <=> $b } @time_value;

##
#       -------------------------------------------------------------------------------
#       Write file information to standard output.
#       -------------------------------------------------------------------------------
##
        print "\@-------------------------------------------------------@\n";
        print " tttplots: TIME TO TARGET (TTT) DISTRIBUTION PLOTS \n\n";
        print " Input data set > \n\n";
        print "    data file   : $datafilename \n\n";
        print "    data points : $n \n";
        print "    max value   : $sorted_time_value[$n-1] \n";
##
#       -------------------------------------------------------------------------------
#       Compute and write input data mean to standard output.
#       -------------------------------------------------------------------------------
##
        $avg=0;
        for ($k=0; $k < $n; $k++){
              $avg=$avg + $sorted_time_value[$k];
        }
        $avg=$avg/$n;
        print "    avg value   : $avg \n";
        print "    min value   : $sorted_time_value[0] \n\n";
```

```
##
#       --------------------------------------------------------------------------------
#       Compute probabilities for distribution plot.
#       --------------------------------------------------------------------------------
##
        $nn = 0;
        $np1=$n+1;
        while ($nn < $n){
                $prob[$nn] = $nn + .5;
                $prob[$nn] = $prob[$nn] / $np1;
              $nn++;
        }

##
#       --------------------------------------------------------------------------------
#       Compute distribution parameters.
#       --------------------------------------------------------------------------------
##
        $fq = ($np1 * .25);
        $tq = ($np1 * .75);
        $fq = int($np1 * .25);
        $tq = int($np1 * .75);

        $y = $prob[$fq];
        $zl = $sorted_time_value[$fq];
        $ql = -log(1-$y);
        $y = $prob[$tq];
        $zu = $sorted_time_value[$tq];
        $qu = -log(1-$y);

        $lambda = ($zu - $zl)/($qu - $ql);
        $mu = $zl - ($lambda * $ql);
##
#       --------------------------------------------------------------------------------
#       Write distribution parameters to standard output.
#       --------------------------------------------------------------------------------
##
        print " Estimated parameters (theoretical shifted exponential
                distribution) > \n\n";
        print "    shift (mu)         : $mu \n";
        print "    std. dev. (lambda) : $lambda \n";
        $shifted_mean = $mu+$lambda;
        print "    mean (shifted)     : $shifted_mean  \n";
##
#       --------------------------------------------------------------------------------
#       Compute theoretical plot (400 points).
#       --------------------------------------------------------------------------------
##
        $tmax = $sorted_time_value[$n-1];
        $inv_lambda = 1/$lambda;
        $eps = $tmax/400;
        $nn = 1;
        while ($nn <= 400){
                $theory_t[$nn-1]= $eps * $nn;
                $theory_p[$nn-1] = 1-exp(-$inv_lambda*($eps * $nn - $mu));
                $nn++;
        }
```

```
##
#       --------------------------------------------------------------------------------
#       Compute theoretical time values.
#       --------------------------------------------------------------------------------
##
        $nn = 0;
        while ($nn < $n){
                $theoretical_time[$nn] = -log(1-$prob[$nn]);
                $nn++;
        }

##
#       --------------------------------------------------------------------------------
#       Compute qqplot line, lower and upper error lines.
#       --------------------------------------------------------------------------------
##
        $nn = 0;
        while ($nn < $n){
                $pi = $prob[$nn];
                $x[$nn] = -log(1-$pi);
                $qq_err[$nn] = $lambda * $x[$nn] + $mu;
                $dev = $lambda  * (sqrt($pi/((1-$pi)*$np1)));
                $lo_error_point[$nn] = $qq_err[$nn] - $dev;
                $up_error_point[$nn] = $qq_err[$nn] + $dev;
                $nn++;
        }

##
#       --------------------------------------------------------------------------------
#       Write output files ...
#       --------------------------------------------------------------------------------
##
        open (EMP_LIN_FILE,">$emp_lin_filename") ||
                    die "Cannot open file: $emp_lin_filename \n";
        open (UP_LIN_FILE,">$up_lin_filename") ||
                    die "Cannot open file: $up_lin_filename \n";
        open (LO_LIN_FILE,">$lo_lin_filename") ||
                    die "Cannot open file: $lo_lin_filename \n";
        open (EMP_EXP_FILE,">$emp_exp_filename") ||
                    die "Cannot open file: $emp_exp_filename \n";
        open (THE_LIN_FILE,">$the_lin_filename") ||
                    die "Cannot open file: $the_lin_filename \n";

        $nn = 0;
        while ($nn < $n){
                print EMP_EXP_FILE
                     "$sorted_time_value[$nn] $prob[$nn] \n";
                print EMP_LIN_FILE
                     "$theoretical_time[$nn] $sorted_time_value[$nn] \n";
                print LO_LIN_FILE "$x[$nn]  $lo_error_point[$nn] \n";
                print UP_LIN_FILE "$x[$nn]  $up_error_point[$nn] \n";
                print THE_LIN_FILE "$x[$nn]  $qq_err[$nn] \n";
                $nn++;
        }
        close (EMP_EXP_FILE);

##
```

```
#       --------------------------------------------------------------------------------
#       Theoretical exponential distribution file.
#       --------------------------------------------------------------------------------
##
        open (THE_EXP_FILE,">$the_exp_filename") ||
                die "Cannot open file: $the_exp_filename \n";
        $nn = 0;
        while ($nn < 400){
              print THE_EXP_FILE "$theory_t[$nn] $theory_p[$nn] \n";
              $nn++;
        }

##
#       --------------------------------------------------------------------------------
#       Create qqplot gnuplot file.
#       --------------------------------------------------------------------------------
##
        open (GPL_LIN_FILE,">$gpl_lin_filename") ||
                       die "Cannot open file: $gpl_lin_filename \n";
        print GPL_LIN_FILE <<EOF;
              set xlabel \'exponential quantiles\'
              set size ratio 1
              set ylabel \'measured times\'
              set key right bottom
              set title \'$filename\'
              set terminal postscript color \'Helvetica\'
              set output \'$ps_lin_filename\'
              plot "$emp_lin_filename" t "empirical" w points,
                     "$the_lin_filename" t "estimated" with lines 3,
                     "$up_lin_filename" t "+1 std dev range" w lines 4,
                     "$lo_lin_filename" t "-1 std dev range" w lines 4
              quit
EOF
##
#       --------------------------------------------------------------------------------
#       Create qqplot postscript graphic file.
#       --------------------------------------------------------------------------------
##
        open (PS_EXP_FILE,">$ps_exp_filename") ||
                     die "Cannot open file: $ps_exp_filename \n";
        system("gnuplot $gpl_lin_filename") == 0 ||
                     die "gnuplot (needed for plotting) not found \n";
##
#       --------------------------------------------------------------------------------
#       Create empirical-theoretical distributions gnuplot file.
#       --------------------------------------------------------------------------------
##
        open (GPL_EXP_FILE,">$gpl_exp_filename") ||
                die "Cannot open file: $gpl_exp_filename \n";
        print GPL_EXP_FILE <<EOF;
              set xlabel \'time to target solution\'
              set size ratio 1
              set ylabel \'cumulative probability\'
              set yrange [0:1]
              set key right bottom
              set grid
              set title \'$filename\'
              set terminal postscript color \'Helvetica\'
```

```
              set output \'$ps_exp_filename\'
              plot "$emp_exp_filename" t "empirical" w points,
                     "$the_exp_filename" t "theoretical" w lines 3
              quit
EOF
##
#       --------------------------------------------------------------------------------
#       Create empirical-theoretical distributions postscript
#       graphic file.
#       --------------------------------------------------------------------------------
##
        open (PS_EXP_FILE,">$ps_exp_filename") ||
                 die "Cannot open file: $ps_exp_filename \n";
        system("gnuplot $gpl_exp_filename") == 0 ||
                     die "gnuplot (needed for plotting) not found \n";
##
#       --------------------------------------------------------------------------------
#       Write file names to standard output.
#       --------------------------------------------------------------------------------
##
        print "\n Output data files > \n\n";
        print "    empirical exponential distribution data  : $emp_exp_filename  \n";
        print "    theoretical exponential distribution data: $the_exp_filename  \n";
        print "    empirical qq-plot data                   : $emp_lin_filename\n";
        print "    theoretical qq-plot data                 : $the_lin_filename\n";
        print "    theoretical upper 1 std dev qq-plot data : $up_lin_filename\n";
        print "    theoretical lower 1 std dev qq-plot data : $lo_lin_filename\n";
        print "    theor. vs empir. ttt plot gnuplot file   : $gpl_exp_filename\n";
        print "    theor. vs empir. qq-plot gnuplot file    : $gpl_lin_filename\n";
        print "    theor. vs empir. ttt plot postscript file: $ps_exp_filename\n";
        print "    theor. vs empir. qq-plot postscript file : $ps_lin_filename\n";
        print "\n DONE \n";
        print "\@-------------------------------------------------------@\n";
        print "\n";
##
#       --------------------------------------------------------------------------------
#       End of program.
#       --------------------------------------------------------------------------------
##
```

(R.M. Aiex) DEPARTMENT OF COMPUTER SCIENCE, CATHOLIC UNIVERSITY OF RIO DE JANEIRO, R. MARQUÊS DE SÃO VICENTE, 225, RIO DE JANEIRO, RJ 22453-900, BRAZIL
E-mail address, R.M. Aiex: rmaiex@yahoo.com

(M.G.C. Resende) ALGORITHMS AND OPTIMIZATION RESEARCH DEPARTMENT, AT&T LABS RESEARCH, 180 PARK AVENUE, ROOM C241, FLORHAM PARK, NJ 07932-0971, USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com

(C.C. Ribeiro) DEPARTMENT OF COMPUTER SCIENCE, UNIVERSIDADE FEDERAL FLUMINENSE, RUA PASSO DA PÁTRIA, 156, NITERÓI, RJ 24210-240, BRAZIL.
E-mail address, C.C. Ribeiro: celso@ic.uff.br