A Server for Automated Performance Analysis of Benchmarking Data*

Hans D. Mittelmann † Armin Pruessner‡

September 3, 2003

Abstract

As part of Performance World, we describe an automation server (PAVER: `http://www.gamsworld.org/performance/paver`) to help facilitate reproducible performance analysis of benchmarking data for optimization software. Although PAVER does not solve optimization problems, it automates the task of performance data analysis and visualization, taking into account various performance metrics. These include not only robustness and efficiency, but also quality of solution. This paper discusses the tools and the performance metrics used, as well as the design of the server. We give illustrative examples of performance data analysis using an instance of the COPS test case for nonlinear programming.

Keywords: Performance data analysis, performance metrics, automation

1 Introduction

Benchmarking is an important tool in the evaluation and development of solvers for mathematical programming problems, in uncovering solver deficiencies, and in the quality assurance process of mathematical programming software.

Many researchers have devoted considerable work to collecting suitable test problems, benchmarking, and performance testing of optimization software, see for example [1, 2, 5, 27, 29, 31], and more notably [15, 25, 26]. Unfortunately, before the seminal paper by Crowder, Dembo, and Mulvey [10], the first to give clear guidelines and standards on how to report computational experiments, little emphasis was placed on the reproducibility of experiments and data analyses.

Historically, most of the results involve the use of tables listing solver resource times, number of iterations, and if no optimal solution was found, the solver

*Revised Manuscript

†Department of Mathematics and Statistics, Arizona State University, Tempe, AZ 85287 (`mittelmann@asu.edu`). This author supported in part by the National Science Foundation under grant 9981984.

‡GAMS Development Corporation, 1217 Potomac St, NW Washington, DC 20007 (`apruessner@gams.com`).

return status. While the analysis and interpretation of these tables is the most factual and objective form of presenting benchmark results (if displayed in full), large data sets can sometimes be overwhelming. In these cases it would be useful to obtain a quick and comprehensive overview of the results in more compact form and in an automated fashion.

As part of Performance World [32], an online forum devoted to all aspects of performance testing of optimization software, we have developed an online server to help facilitate and automate performance analysis and visualization of benchmarking data. This paper describes the goals of this server and the implementation of the tools. We describe the various performance metrics that are used and how these are implemented in the various tools.

This paper is organized as follows: in §2 we give an overview on the design and implementation of the server itself. In §3 we discuss various performance metrics that are useful in performance data analysis and in §4 describe the tools that we provide as part of the server. Section 5 discusses some key issues in benchmarking and in §6 we give numerical results using the COPS set of nonlinear models. Finally, in §7 we draw conclusions.

2 PAVER Server Design

The PAVER Server (`http://www.gamsworld.org/performance/paver`) is a web-based service for reproducible performance analysis of benchmarking data obtained via optimization software. While the Network-Enabled Optimization System (NEOS) [30, 11, 23, 14] is an environment for the solution of optimization problems and thus the data collection phase in the benchmarking process, PAVER seeks to automate and simplify specific tasks in the performance *data analysis phase*. PAVER provides simple online tools for automated performance analysis, visualization, and processing of benchmarking data.

An optimization engine, either a modeling environment such as AMPL [19] or GAMS [6], or a stand-alone solver, generally provides solution information such as objective function value, resource time, number of iterations, and the solver status. The latter gives information of optimality or feasibility of the solution, or infeasibility or unboundedness of the model. Within GAMS, solve and model statistics are captured automatically by means of *trace files*. Benchmark data (in trace file format) obtained by running several solvers over a set of models can be automatically analyzed via online submission to the PAVER server. A detailed performance analysis report is returned via e-mail in HTML format and is also initially available online.

2.1 Data Submission and Processing

PAVER accepts benchmarking data in the form of trace files, where each trace file contains data for a single solver over a model test set. Trace files contain information such as model statistics (number of variables, constraints, nonzeros, discrete variables), resource time, iterations, objective function value, as well as

GAMS model and solver return status codes. The latter codes are identifiers for specifying solve return states, such as optimal (globally), locally optimal, infeasible, unbounded, etc. For more information on GAMS return codes see [6].

Users submit their trace files via the online (World-Wide Web) submission tool at

`http://www.gamsworld.org/performance/paver/pprocess_submit.htm`

The tool accepts the submission of up to 8 trace files, automatically performing cross comparisons of each solver with respect to every other solver.

After submission, the trace files are analyzed for correctness and formatting. If a trace file does not have the proper format, for example because a data column is missing or because multiple solvers are specified in a single trace file, users receive an error message online. Provided the trace files are valid, a scheduler is invoked, which schedules the performance analysis job for a particular workstation. The scheduling task takes the form of optimizing a mixed integer program, taking into account current workstation workloads. The trace files are then sent from the server to the next available workstation specified by the schedule. The performance analysis is completed and the results returned to the server. The server then sends the results via e-mail attachment to the user. The process is illustrated in Figure 1.

Note that although a single performance analysis of 8 trace files may only take 30 seconds, the submission of multiple jobs can result in an untimely backlog. The scheduler guarantees that jobs are placed from the queue to the next available workstation, resulting in a more efficient process. Workstations can be added or removed as needed without any downtime of the PAVER server.

2.2 Trace File Format

While initial applications of the server involved models written in GAMS format and performance data collected using GAMS, the PAVER server provides a free service and interface which users of other software can make use of.

Model runs using GAMS automatically capture performance data in the form of trace files. Users of other modelling systems or stand-alone solvers can also run PAVER, provided the performance data is submitted in the same format. The trace file format consists of comma-delimited text files containing model and solve statistics and solution information. The necessary data is listed in Table 1.

The first five column fields are required text fields. The text fields must not contain any spaces, unless the element is quoted. If a text field value is not known, for example no MIP solver (MIP def.) exists, then the user could enter a default value such as `MIPdef` or similar. The other fields are numeric, where values of 0 can be specified if the data is not known.

For more information on trace files, see for example [8]

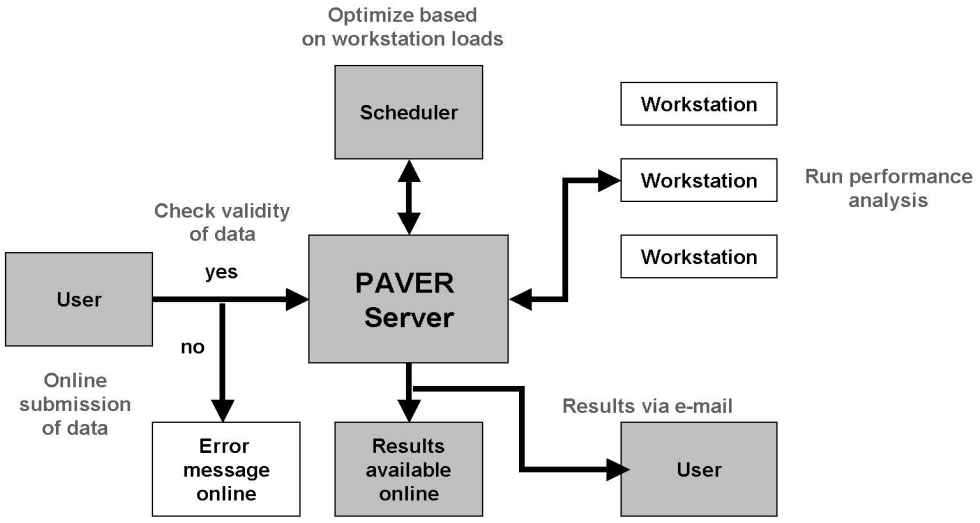

Figure 1: PAVER: Process Overview

3 Measures of Performance

3.1 Metrics

The main concern in presenting benchmark results is in removing some of the ambiguity in interpreting results. Metrics should give information on

- solver robustness
- solver efficiciency
- quality of solution

While the first two metrics have long been viewed as standard performance measures, the latter is becoming more important with recent advances in global optimization. See for example the algorithms in [33, 35, 36].

In general, performance metrics used to compare solvers should not be dominated by single test cases, as is the case in some results using (arithmetic mean) averages of all ratios of a solve time of a single solver with respect to the fastest time of all solvers in the comparison. In [3] Bixby makes use of geometric mean averages which are less influenced by outliers, but the benchmark omitted models which were not solved by all solvers in the comparison.

As Dolan and Moré note in [12], metrics should not only consider efficiency, but also robustness. Unfortunately, some measures of performance are applicable only if all solvers used can solve all problems in the benchmark test case. If a comparison only considers models which all solvers can solve, then the most

Table 1: Trace File Format

Heading	**Description**
Modelname (required)	Model Filename
Modeltype (required)	LP, MIP, NLP, etc.
Solvername (required)	
NLP def. (required)	default NLP solver
MIP def. (required)	default MIP solver
Juliantoday	start day/time of job in Julian format
Direction	0=min, 1=max
Equnum	Total number of equations
Varnum	Total number of variables
Dvarnum	Total number of discrete variables
Nz	Number of nonzeros
Nlnz	Number of nonlinear nonzeros
Optfile	1=optfile included, 0=none
Modelstatus	GAMS model status
Solverstatus	GAMS solver status
Obj	Value of objective function
Objest	Estimate of objective function
Res used	Solver resource time used (sec)
Iter used	Number of solver iterations
Dom used	Number of domain violations
Nodes used	Number of nodes used

robust solvers may be unduly penalized if they are able to solve more difficult models that others were not able to solve.

The work of Dolan and Moré [12] using performance profiles has been very useful in removing some of the ambiguity involved in interpreting benchmark results. The use of profiles, cumulative distribution functions over a given performance metric, present a descriptive measure providing a wealth of information such as solver efficiency, robustness, and probability of success in compact form.

The tools available as part of PAVER take into account the various metrics using the solver square and resource time comparisons, which are described in detail in §4. Furthermore, we show how the profiles of Dolan and Moré can be used to include quality of solution information.

3.2 Subjectivity and Model Selection

The choice of test problems for benchmarks is difficult and inherently subjective. While there is no consensus on choosing appropriate models, many interesting and diverse model libraries exist which come from a wide variety of application areas. See for example MIPLIB [4] for mixed integer programs, the COPS

Table 2: PAVER Tools Overview

Tool	**Robustness**	**Efficiency**	**Solution Quality**
Square	×		
Resource Time		×	×
Profiles	×	×	×

[9] library for nonlinear programs, and the MINLPLib [7] for mixed integer nonlinear programs.

Nonetheless, even the use of models from standard benchmark libraries, does not make the process completely objective. For example, it is easily possible to choose a subset of models where one solver outperforms all others and another subset where another solver outperforms all others. See the example in [34] and the results in §6.

Therefore, in order to reduce the risk of bias in benchmark results, it is sometimes helpful to perform benchmarks on various sets of models and observing solver performance trends over the whole rather than relying on a single benchmark test set. The automated performance analysis tools described herein may simplify the data analysis process and help in obtaining results over various data sets quickly. Finally, any representation of benchmark data using performance tools is only as complete as the data itself and it is important that users understand the test set, the solvers, and the solver options used.

4 Tools

The tools available in PAVER allow either direct comparisons between two solvers or comparisons of more than two solvers simultaneously. The *solver square* and *resource time* utilities belong to the former and the *performance profile* utility to the latter. An overview of the tools and the (primary) performance measure employed by each is given in Table 2.

4.1 Solver Square

Robustness comparisons between two solvers can be achieved using the PAVER comparison utility we refer to as solver square. Given possible solver outcomes of optimal (globally), locally optimal, feasible, unbounded, and fail, the utility displays the number of models that fall into each category for each solver. The outcome category is determined by the solver and specified in the trace file via the model and solve status codes. Furthermore, the utility gives information on the number of models that fall into a solver outcome category pair.

For example, a user may be interested in the number of models that were solved locally optimal by one solver, but were found infeasible by another. The utility lists the models that fall into a particular outcome pair category and

gives resource times and objective function information for each model in that category.

The utility allows for quick identification of models where one solver is more robust than another. It has been used, for example in [16], for fine tuning a new solver version.

4.2 Resource Time Comparisons

The *resource time utility* compares solver resource times of two given solvers. In order to give information on solver robustness, solvers which do not find a feasible solution are assigned an infinite time for the particular model. The utility lists the number of models which were solved in the same amount of time, where one solver was faster and where one solver was much faster than the other.

Furthermore, models are disaggregated by objective value as well. Thus, models in each category are further subdivided into models where both solvers found the same objective function (within a given tolerance), or where one solver found a better solution.

Again, this information has been used in [16] for comparisons of solver versions.

4.3 Performance Profiles

Performance profiles are cumulative distribution functions over a given performance metric and give perhaps the most complete information in terms of robustness, efficiency and solution quality. We will give a brief overview of performance profiles, much of which comes from [12]. Furthermore, we discuss how performance profiles can be used to include quality of solution information.

4.3.1 Background

Suppose we have n_s solvers which we run over a set P of n_p problems. Let $t_{p,s}$ be the solver resource time for solver s on problem p. Now define a *performance ratio* as

$$\rho_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : 1 \leq s \leq n_s\}} \tag{1}$$

For solvers s that do not solve problem p, choose a parameter $\rho_M \geq \rho_{p,s}$. Now define

$$p_s(\tau) = \frac{1}{n}\text{size}\{p \in P : \rho_{p,s} \leq \tau\} \tag{2}$$

Then $p_s(\tau) : \mathbb{R} \mapsto [0, 1]$ is the probability that a performance ratio $\rho_{p,s}$ is within τ of the best ratio. The function in (2) is a *performance profile* and is the cumulative distribution function for the performance ratio in (1). Furthermore, it is piecewise constant, monotonically increasing and continuous from the right at each of the breakpoints.

We remark that the performance ratio is not confined to the one chosen in equation (1), but can be chosen based on the purpose of the profile.

4.3.2 Interpretation

The profile gives much information including information about solver robustness and efficiency. If a user is only interested in solver efficiency, then the user can examine profile values $p_s(\tau)$ for $\tau = 1$ of different solvers s. The values $p_s(1)$ specifies the probability that a solver will "win" over all other solvers. For the profile given above, we define a "win" as the solver who finds any optimal solution (local or global) in the least amount of time. It is possible to choose different definitions of "win" based on different performance ratios.

If a user is only interested in the probability of sucess of a solver for the problem set P, then the user may examine

$$\lim_{\tau \to \infty} \quad p_s(\tau) \tag{3}$$

For ratios τ approaching ∞ we are looking at the probability of success of a solver given unlimited resource time. Again, we remark that success is defined as finding an optimal (local or global) solution.

4.3.3 Quality of Solution

The performance ratio defined in Equation (1) can be modified if the user is also interested in information on quality of the solution returned by a solver. This is of particular interest for nonconvex and discrete models.

Recall that in the original performance ratio we assigned a suitably large ratio ρ_M to those problem/solver pairs p, s if a solver s failed to solve problem p. If we are interested in quality of solution information in the ratio, we can modify the ratio. If $o_{p,s}$ is the solution found by solver s for problem p and b_p is the best solution found by all solvers $s \in S$ for problem p, then we define a new performance ratio as

$$\rho_{p,s} = \left\{ \begin{array}{lll} \frac{t_{p,s}}{\min\{t_{p,s} : 1 \le s \le n_s\}} & \text{if} & \left| \frac{o_{p,s} - b_s}{b_s} \right| \le \delta \\ \\ \rho_M & \text{if} & \left| \frac{o_{p,s} - b_s}{b_s} \right| > \delta \end{array} \right\} \tag{4}$$

where δ is a user-defined relative objective function difference threshold and ρ_M is again an upper bound on $\rho_{p,s}$ over all problems p and solvers s. The ratio here is similar as before, except that we consider a solver successful only if the solution returned by a solver is within δ of the best solution found. We note that the performance ratio defined in Equation (1) is a special case of the general ratio defined in (4). For the new ratio, the performance profile is the same as defined previously.

Interpretation of the profile is similar as before. Consider the case for $\delta = 0$: if we are only interested in the solver that finds the best solution of all solvers in the fastest time, we consider $p_s(1)$ for all solvers. If we are interested in the probability that a solver will find the best solution over all solvers, then we consider $p(\tau)$ as $\tau \to \infty$. Interpretations for other values of δ are analogous.

Note that in practice we usually do not choose $\delta = 0$ but rather some small positive value, say 10^{-6}.

5 Benchmarking Issues

In this section we discuss some of the key issues in benchmarking. In particular we discuss solver precision and solution verification.

Because solvers use different optimality metrics (measurements of optimality) and feasibility, the comparison of benchmarking data from different solvers is inherently imprecise. In order to compare solvers in an unbiased manner, it can be argued that the solution should be verified independently. In particular the solution provided by each solver should be verified using the same optimality and feasibility criterion.

Some work in this direction has been done already. Within the NEOS framework, a benchmark solver exists which determines if the solution satisfies the user-specified complementarity and optimality tolerances. In particular, the utility returns the feasibility, complementarity, optimality and scaled optimality errors. For more information see [30] and [13]. Within GAMS, the Examiner [20] solver can be used to independently verify the solution for feasibility, complementarity and optimality. Examiner works inbetween the modeling environment and the solver in a seamless fashion.

While independent verification is a step in the right direction, we argue that this additional measure still does not guarantee a completely fair comparison. In particular, the specification of solver options work to satisfy feasibility and optimality criteria in the *solver frame of reference*, whereas verification takes place in a separate reference frame. Thus, tightening of solver tolerances may not necessarily lead to tighter solutions in the *verification frame of reference*.

On the other hand, the choice of sufficiently tight solver options is difficult to make without choosing a too tight option. In particular, one solver may work unduly hard to satisfy the verification tolerances, where a lower solver tolerance may suffice. This can lead to longer resource times than necessary or lead to the solver not being able to satisfy the tolerances at all, even though some sufficient tolerance may exist.

Completely fair data comparisons can only occur if solvers have a *uniform stopping criteria* which is indeed different than satisfying solution verification tolerances. In practice, the former criteria is unfortunately difficult, if not impossible, to satisfy.

6 Illustrative Example: COPS Models

In order to illustrate the features of the PAVER performance analysis tools we chose the COPS [9] library of nonlinear test problems. Our selection of solvers include two solvers run through AMPL (using the GAMS/AMPL interface) and two through GAMS directly.

Table 3: COPS Model Statistics. Number of models in COPS benchmark set having specified number of variables (Vars), number of nonzeros (NZ) and number of nonlinear nonzeros (NLNZ). NLNZ refers to the number of nonzero coefficients of nonlinear variables. For example, 21 of the 64 COPS models have between $100-999$ variables. Only 10 models have NLNZ between $10000-99999$.

	10-99	**100-999**	**1000-9999**	**10000-99999**
# Models (Vars)	2	21	37	4
# Models (NZ)	0	6	37	21
# Models (NLNZ)	0	10	44	10

Our experiments involve two test sets. One uses the complete COPS test set except for the `flowchan` models, and the other a subset of models, illustrating the subjectivity in model selection. In particular, we show that depending on the performance metric of interest any of the four solvers can be deemed "superior."

Note that we did not verify the solution independently for the reasons discussed in the previous section. While some work in the verification of the solution makes sense for precise benchmarks, our intent rather is to illustrate the features of PAVER.

6.1 COPS NLP Models

Although the COPS models were originally in AMPL format, we chose an implementation of the models in the GAMS modeling language. The models come from 17 different application areas and exhibit multiple maxima and minima. Our test instance consists of four sizes of each of the COPS models, varying a parameter in the application. The parameter can, for example, specify the number of grid points in a discretization . All of the models used are available as part of the GLOBALLib [22] library of models. We omitted the family of `flowchan` models, as their implementation in GAMS is as a constrained nonlinear system (CNS), where objective function comparisons are not of interest. This resulted in a total of 64 models in the benchmark test set. Table 3 gives an overview of the model sizes. For full model statistics information see [22].

6.2 Solvers and Experimental Conditions

We chose CONOPT3 [17] (Library 301F), KNITRO [38] (version 2.1 08/01/02), IPOPT [24] (version 2.0.1), and SNOPT [21] (version 5.3-5(2)) as our solvers. CONOPT3 and SNOPT were run through GAMS, while KNITRO and IPOPT were run through AMPL, employing the GAMS/AMPL interface so that all solvers make use of the COPS implementation in the GAMS modeling language. For the AMPL solvers, we additionally wrote scripts to parse the log output in order to obtain additional performance data necessary for input to PAVER. This additional data is not available directly through the GAMS/AMPL interface.

All benchmark runs were obtained by running the tests on an Ultra60 with dual 450 MHz Ultrasparc 2 processors and 2 Gb of memory, running Solaris 8. We set a resource time limit of $3,600$ seconds for each model for each solver. To ensure consistency, we have verified that solver resource times can be reproduced to within 10% accuracy. Solvers were run in default mode, except for SNOPT, where we increased the superbasics limit to $10,000$. The AMPL solvers were run with specific output options to get the desired log information needed to create trace files.

6.3 Results - All Models

We show the results using PAVER with all 64 COPS models and all four solvers. The complete results of this benchmark are available online at

`http://www.gamsworld.org/performance/cops/`

Figure 2: Summary File

Figure 2 lists the performance analysis summary HTML file returned by PAVER. The first part under `Trace files used` provides links to the trace files that were submitted. The second part under `Performance profile summary` gives a link to the performance profiles obtained from the submitted data. The plots show the results for all solvers in a single plot.

The next part under `Solver square summary` gives links to the solver square for two solvers at a time. The matrix shown gives the solver square results for all possible combinations of two solvers. In this case we have 6 different solver square tables.

Similarly, under the `Resource time summary`, we have links to the resource time comparison tables for all possible combinations of solvers (a total of 6 different tables).

Figure 3: Solver Square

Figure 3 shows the results of the solver square utility. We only show results for the comparisons between CONOPT3 and KNITRO, although comparisons for all solver pairs are computed automatically by PAVER (see the online results). The utility gives solver robustness information of two solvers in compact form and allows quick cross comparisons of two solvers. The results for CONOPT3 are listed in the white cells and the results for KNITRO in the dark grey-shaded cells. The intersection (denoted by light grey cells) lists the number of models falling into a given outcome pair.

For example, in Figure 3, 41 models were solved locally optimal by both solvers. KNITRO found 45 locally optimal solutions and another 11 feasible solutions (see the column `total AMPL/KNITRO`). Of the 45 locally optimal models, CONOPT3 found 4 of them infeasible. On the other hand, CONOPT3 found

locally optimal solutions for 6 models, where KNITRO failed. The square utility also lists resource times and objective value information for all models (not shown here). The table entries are links to the models falling into a particular category. Models found to have a better objective function value are listed in boldface.

Figure 4: Performance Profiles (Efficiency)

Figure 4 shows the performance profile of all solvers using the performance ratio in Equation (1), where we are interested only in solver efficiency. For example, the fastest solver is KNITRO, which solves roughly 45% of the models the fastest, followed by CONOPT3 at 25%. (See the profile for a `Time Factor` of 1). If we are interested in overall probability of success, then SNOPT has the highest probability of success at roughly 95%, even though it had the lowest profile for efficiency alone. KNITRO and CONOPT3 followed with success rates of roughly 87% and 82%. (see the profile as `Time Factor`$\rightarrow \infty$). Note the profile of `CAN_SOLVE`, which is the probability of success that any solver in the comparison can solve the problem.

Figure 5: Performance Profiles (Quality of Solution)

Figure 5 shows the profile using the performance ratio given in Equation (4), where we take into account the solver quality of solution. We used a value of $\delta = 10^{-5}$, indicating that the relative objective function error can be no greater than 10^{-5} with respect to the best solution found.

In terms of quality of solution, CONOPT3 is the fastest solver overall in finding the best solution, solving roughly 35% of the models in the most efficient manner. KNITRO follows at about 30%. (See the profile for a `Time Factor` of 1). If we are interested in probability of success in finding the best solution, then SNOPT and CONOPT3 "win" most of the time (at about 78% and 76%) followed closely by KNITRO at roughly 70%. (See the profile as `Time Factor` $\rightarrow \infty$).

6.4 Results - Subset of Models

In this example we will illustrate the subjectivity in the selection of models. In particular, using PAVER, it is easy to choose a subset of models where any one solver performs the best subject to a given performance metric. In this case, the metric is based on efficiency only.

Figure 6: Resource Times (IPOPT Best)

In order to choose a subset we make use of the Resource Time Utility. We will choose a subset of models where IPOPT performs best in terms of solver efficiency. In Figure 6 we show results for resouce time comparisons between CONOPT3 and IPOPT for all COPS models in the benchmark. For example in column `Total` we can see that for 6 models IPOPT was faster, for 8 much

faster and for 2 infinitely faster. A solver is considered faster, if it is 10% faster with respect to the faster time, much faster if it is 50% faster, and inifinitely faster if one of them finds a solution whereas the other does not.

We can thereby choose a subset of models, for example the $6 + 8 + 2$ models which gurantees that IPOPT will perform better than CONOPT. In particular, we will choose these 16 models plus the 5 models where both solvers perform the same and show in the next figure that IPOPT performs the "best" for this model test set. The numbers are links to the models that fall into a particular category.

Models are further categorized by objective function value. As the square utility, the resource time utility gives detailed objective function value and timing information for individual models (not shown).

Figure 7: Performance Profiles - Efficiency (IPOPT Best)

For this plot we choose the 21 models as defined from the previous resource time results. If we look at the performance profile plot in Figure 7 we can see that IPOPT is the fastest solver, finding a solution the fastest for more than 60% of the models. Note that for each of the other two profile plots (efficiency and quality of solution) in Figures 4 and 5 IPOPT appeared to look less effective than the other solvers. If we look at probability of success, then IPOPT again is the "best" and can find a solution to all models.

So although initial results appeared to indicate that the other solvers are more effective, for almost one third of the models, IPOPT still outperforms the other solvers.

Discussion of Results

The results indicate that given the metric of interest each of the solvers can outperform the others. Furthermore, we have illustrated how model selection plays an important role in benchmark analysis and that users must be careful to consider the test set. Any unbiased benchmark should therefore include multiple test sets so that general trends over multiple data sets can be analyzed, thereby minimizing the risk of subjective interpretation of results.

7 Conclusions

We have introduced an online server for reproducible and automated performance analysis and visualization of benchmarking data. While the original intent was for analyzing benchmark data of linear and nonlinear optimization software, the server can be utilized for other algorithmic software, for example, solvers for problems in numerical linear algebra, ordinary or partial differential equations. Provided users supply solve information in trace file format, the server can perform similar performance analyses on non-optimization engines. Future work may focus on expanding PAVER to use a more general format such as XML as data input.

Acknowledgements

We wish to thank the referees and the editor for their helpful comments which greatly improved the original manuscript.

We also would like to thank Michael Bussieck, Steve Dirkse, Arne Drud, and Alex Meeraus for spirited discussions on performance metrics and quality assurance and Michael Ferris, Jorge Moré, and Nick Sahinidis for helpful comments on early versions of the paper.

Furthermore, Michael Bussieck was instrumental in design of the server and Gene Pian devoted considerable hours in helping set up the server. Also, we wish to thank Prakhar Lodha who wrote most of the scripts to parse the AMPL log output and enable benchmark comparisons involving AMPL solvers.

References

[1] H. Y. Benson, D. F. Shanno, and R. J. Vanderbei (2000). Interior-point methods for nonconvex nonlinear programming: Jamming and comparative numerical testing. *Technical Report ORFE-00-02*, Princeton University, Princeton, New Jersey.

[2] S. C. Billups, S. P. Dirkse, and M. C. Ferris (1997). A comparison of large-scale mixed complementarity problem solvers. *Comp. Optim. Appl.*, **7**, 3–25.

[3] R. E. Bixby (2002). Solving real-world linear programs: a decade and more of progress, *Operations Research,* 50^{th} *Anniversary Issue*, **50** (1), 3–15.

[4] R. E. Bixby, S. Ceria, C. M. McZeal and M. W. P. Savelsbergh (1998). An Updated Mixed Integer Programming Library: MIPLIB 3.0., *Optima*, **58**, 12–15.

[5] I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint (1997). A numerical comparison between the LANCELOT and MINOS packages for large-scale constrained optimization. *Namur University*, Report 97/13.

[6] A. Brooke, D. Kendrick, and A. Meeraus (1988). *GAMS: A User's Guide*, The Scientific Press, San Francisco, CA.

[7] M. R. Bussieck, A. S. Drud, and A. Meeraus (2002), MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming, *INFORMS J. Comput.*, **15** (1), 114–119.

[8] M. R. Bussieck, A. S. Drud, A. Meeraus, and A. Pruessner (2003), Quality Assurance and Global Optimization, *LNCS of the 1st International Workshop on Global Constrained Optimization and Constraint Satisfaction (COCOS'2002)*, France, October 2002, Lecture Notes in Computer Science, Springer Verlag, to appear.

[9] COPS. [www http://www-unix.mcs.anl.gov/~more/cops].

[10] H. Crowder, R. S. Dembo, and J. M. Mulevy (1979), On Reporting Computational Experiments with Mathematical Software, *ACM Transactions of Mathematical Software*, **5** (2), 193–203.

[11] J. Czyzyk, M. Mesnier, and J. J. Moré (1998). The NEOS Server, *IEEE J. Comp. Sci. Eng.*, **5**, 68–75.

[12] E. D. Dolan and J. J. Moré (2002). Benchmarking optimization software with performance profiles, *Math. Programming*, **91** (2), 201–213.

[13] E. D. Dolan, J. J. Moré, and T. S. Munson (2002). Measures of optimality for constrained optimization, [www http://www-neos.mcs.anl.gov/neos/ftp/optimality.pdf].

[14] E. D. Dolan (2001). The NEOS Server 4.0 Administrative Guide, *Technical Memorandum ANL/MCS-TM-250*, Mathematics and Computer Science Division, Argonne National Laboratory.

[15] E. D. Dolan and J. J. Moré (2000). Benchmarking optimization software with COPS, *Technical Memorandum ANL/MCS-TM-246*, Argonne National Laboratory, Argonne, Illinois.

[16] A. Drud, Testing and Tuning a New Solver Version Using Performance Tests, Informs 2002, San Jose, Session on "Benchmarking & Performance Testing of Optimization Software." [www http://www.gams.com/presentations/present_performance.pdf].

[17] A. S. Drud (1996). CONOPT: A System for Large Scale Nonlinear Optimization, Reference Manual for CONOPT Subroutine Library, ARKI Consulting and Development A/S, Bagsvaerd, Denmark.

[18] R. Fletcher and S. Leyffer (1998). User manual for filterSQP, *Dundee University Numerical Analysis Report*, NA/181.

[19] R. Fourer D. M. Gay, and B. W. Kernighan (2003). *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press, Brooks/Cole-Thomson Publishing Company, Pacific Grove, CA.

[20] GAMS (2003). *GAMS: The Solver Manuals*, GAMS Development Corporation, Washington, DC.

[21] P. E. Gill, W. Murray, and M. A. Saunders (2002). SNOPT: An algorithm for large-scale constrained optimization, *SIAM J. Opt.*, **12**, 979–1006.

[22] GLOBALlib. [www http://www.gamsworld.org/global/globallib.htm].

[23] W. Gropp and J. J. Moré (1997). Optimization Environments and the NEOS Server, In: M. D. Buhmann and A. Iserles (Eds.), *Approximation Theory and Optimization*, 167–182, Cambridge University Press, Cambridge.

[24] IPOPT: An interior point algorithm for large-scale nonlinear optimization, [www http://www-124.ibm.com/developerworks/opensource/coin/Ipopt].

[25] H. D. Mittelmann and P. Spellucci (2003). *Decision Tree for Optimization Software*, [www http://plato.la.asu.edu/guide.html].

[26] H. D. Mittelmann (2003). An independent benchmarking of SDP and SOCP solvers, *Math. Programming Ser. B*, **95**, 407–430.

[27] H. D. Mittelmann (1999). Benchmarking interior point LP/QP solvers, *Opt. Meth. Software*, **12**, 655–670.

[28] B. A. Murtagh and M. A. Saunders (1998). MINOS 5.5 User's Guide, *Stanford Univ. Dept. Op. Research, Rpt. SOL 83-20R.*

[29] S. G. Nash and J. Nocedal (1991). A numerical study of the limited memory BFGS method and the truncated Newton method for large scale optimization. *SIAM J. Optim.*, **1**, 358–372.

[30] NEOS (1997). [www http://www-neos.mcs.anl.gov].

[31] A. Neumaier (2000). [www http://www.mat.univie.ac.at/~neum/glopt.html].

[32] Performance World (2002). [www http://www.gamsworld.org/performance].

[33] J. D. Pintér (2002), *LGO - A Model Development System for Continuous Global Optimization. User's Guide. (Current revised edition)*, Pintér Consulting Services, Halifax, NS.

[34] A. Pruessner (2002). Automated Performance Testing and Analysis, Informs 2002, San Jose, Session on "Benchmarking & Performance Testing of Optimization Software." [www http://www.gams.com/presentations/present_automation.pdf.]

[35] M. Tawarmalani and N. V. Sahinidis (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic Publishers, Dordrecht.

[36] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Marti (2002). A multistart scatter search heuristic for smooth NLP and MINLP problems, *INFORMS J. Comp.*, to appear.

[37] R. J. Vanderbei (1999). LOQO user's manual - Version 3.10, *Opt. Meth. Software*, **12**, 231–252

[38] R. A. Waltz and J. Nocedal (2002). KNITRO 2.0 User's Manual, [www http://www.ziena.com/knitro/manual.htm].

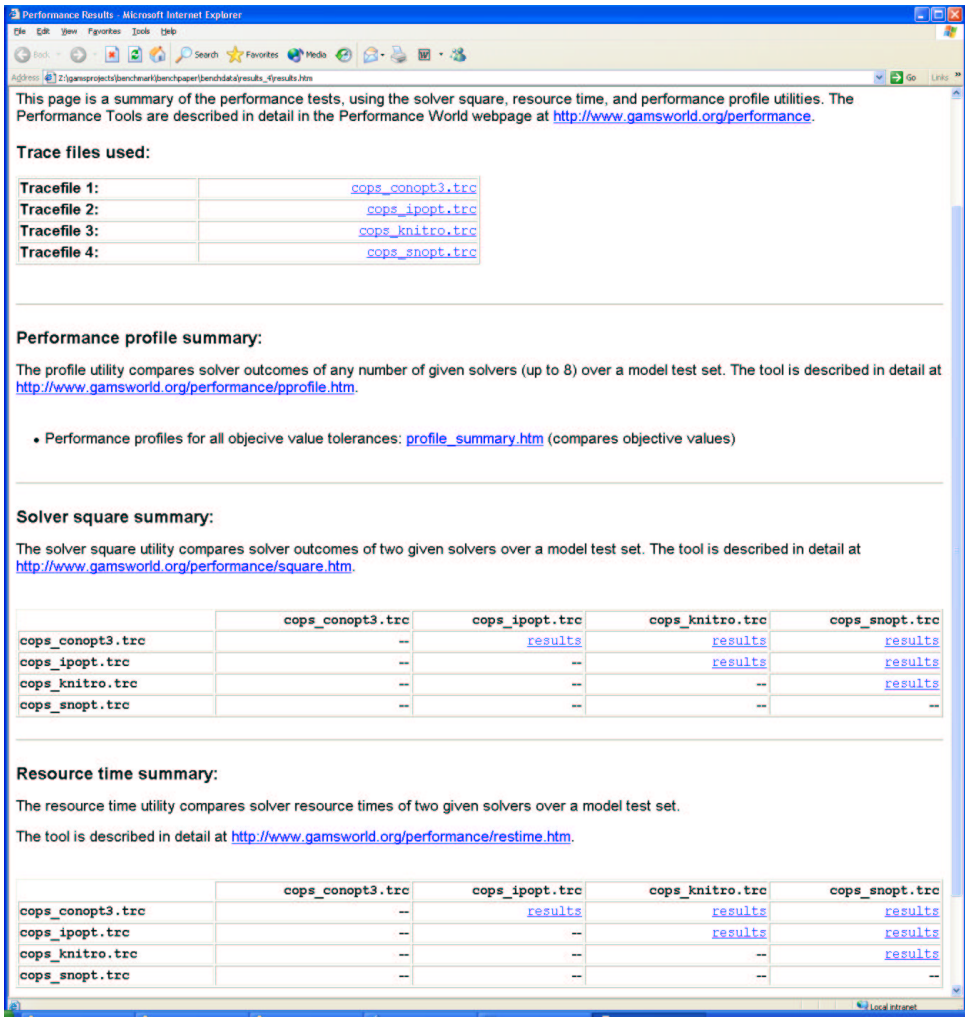

Figure 2: PAVER Summary (All models).

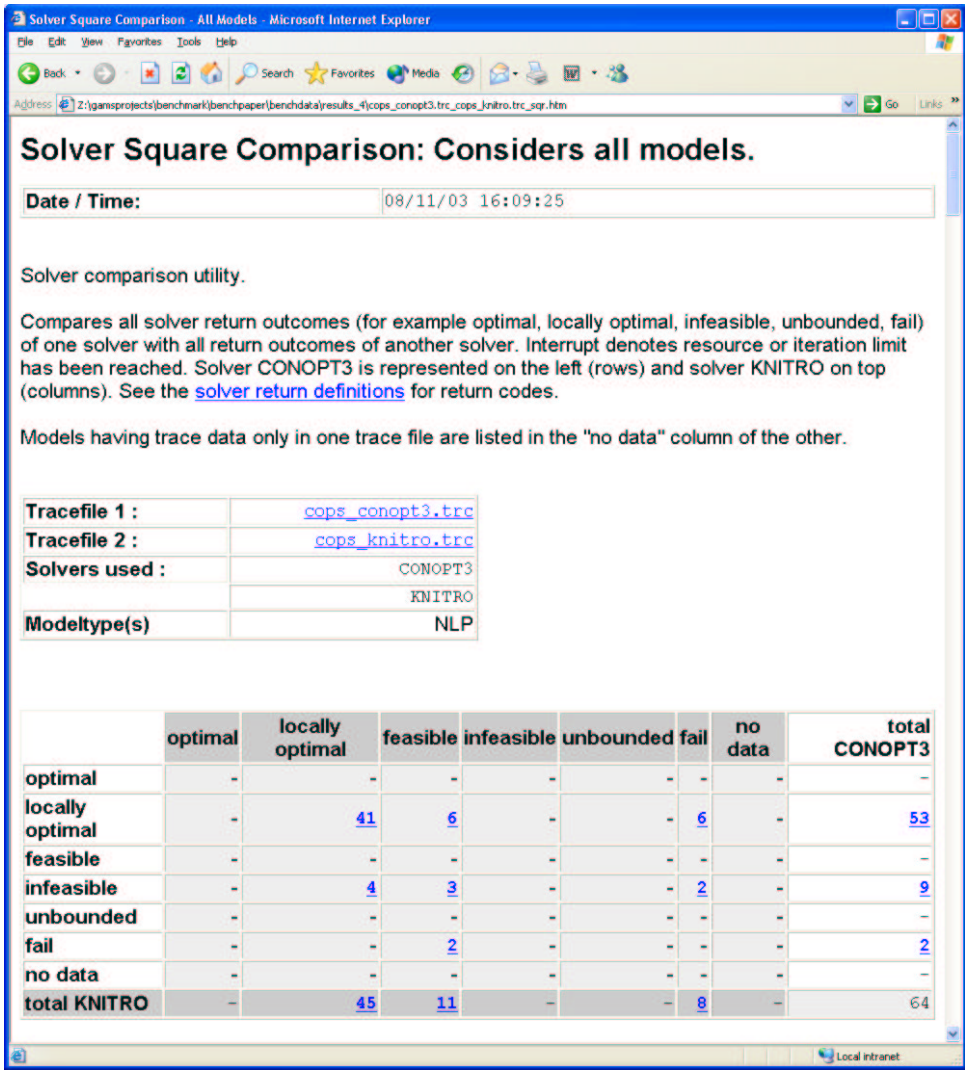

Figure 3: PAVER Square Utility Results (All models. Solvers: CONOPT, KNITRO).

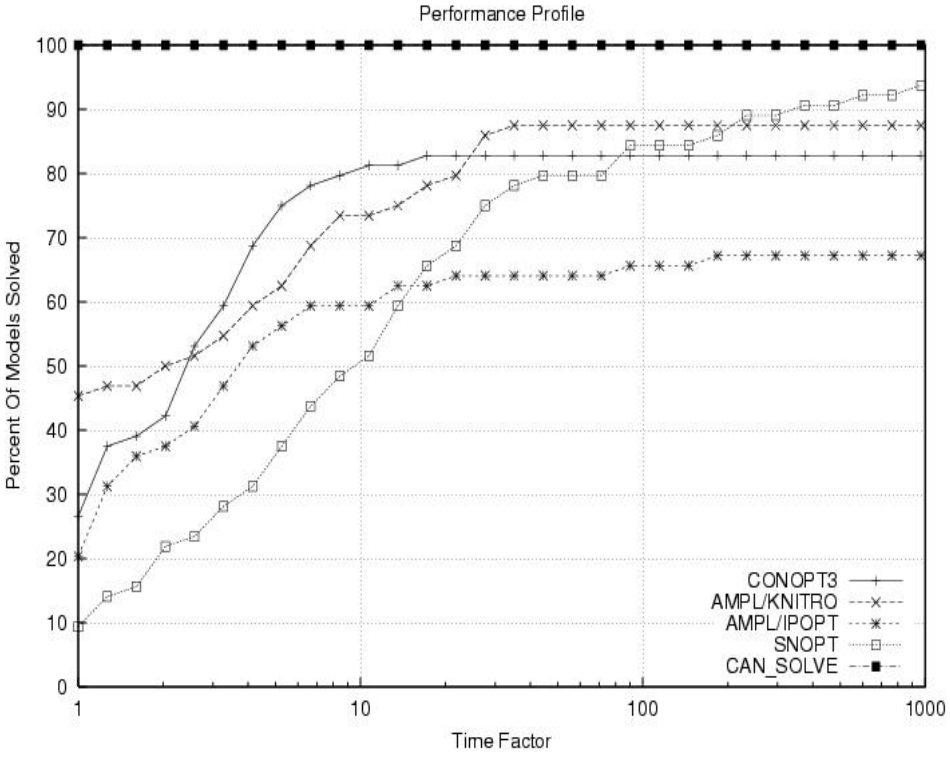

Figure 4: PAVER Performance Profile. Time Factor τ is in $\log_{10}$ scale

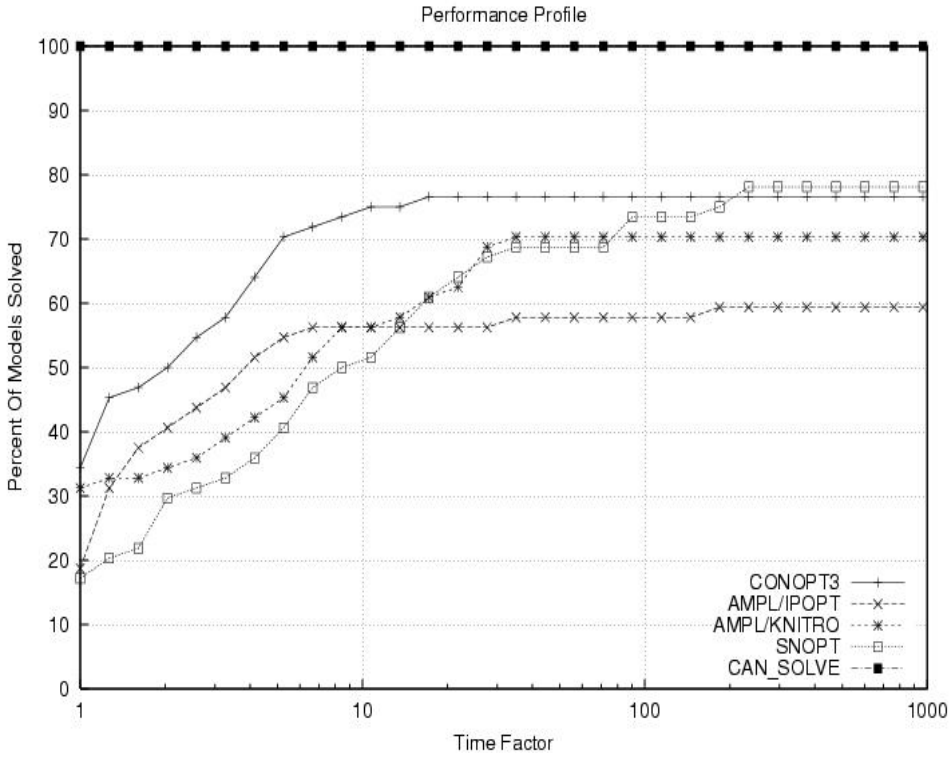

Figure 5: PAVER Performance Profile (Quality of Solution). Time Factor τ is in $\log_{10}$ scale.

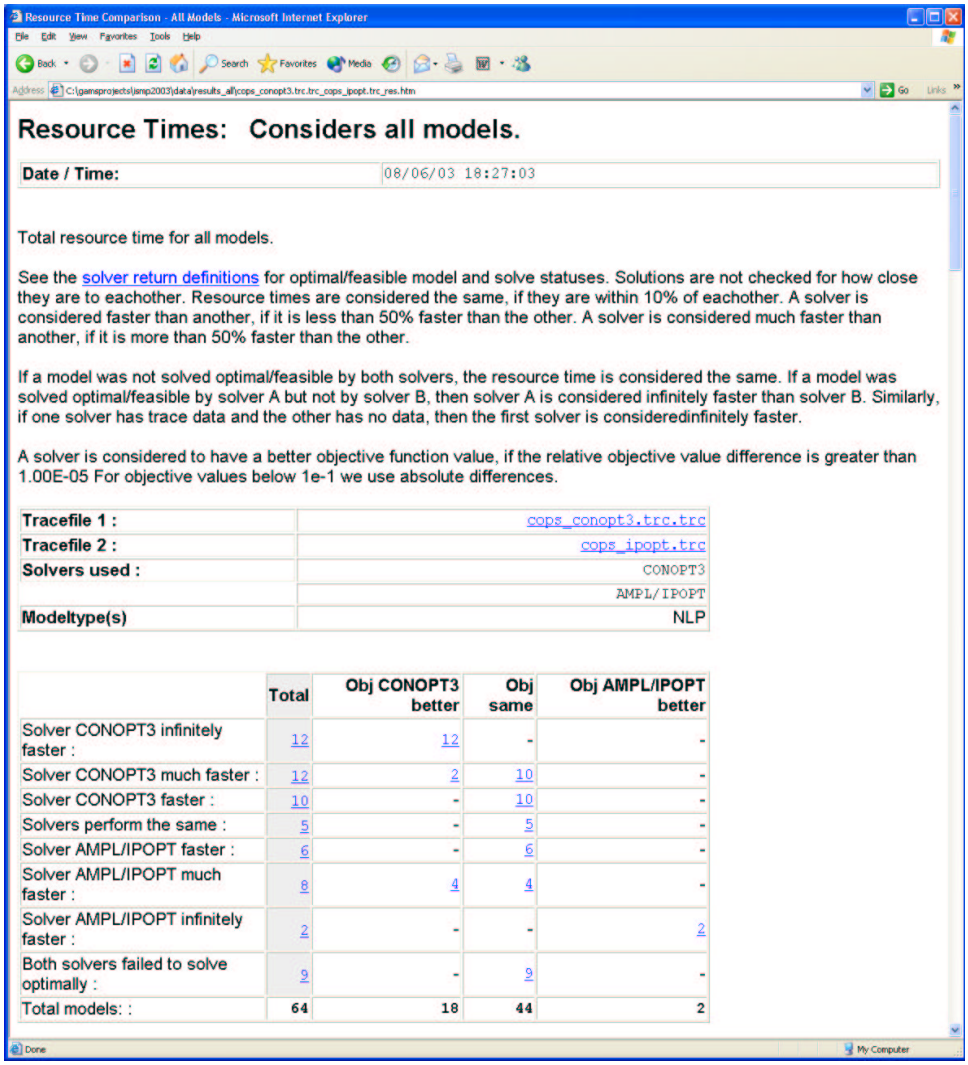

Figure 6: PAVER Resource Time Utility Results - IPOPT Best.

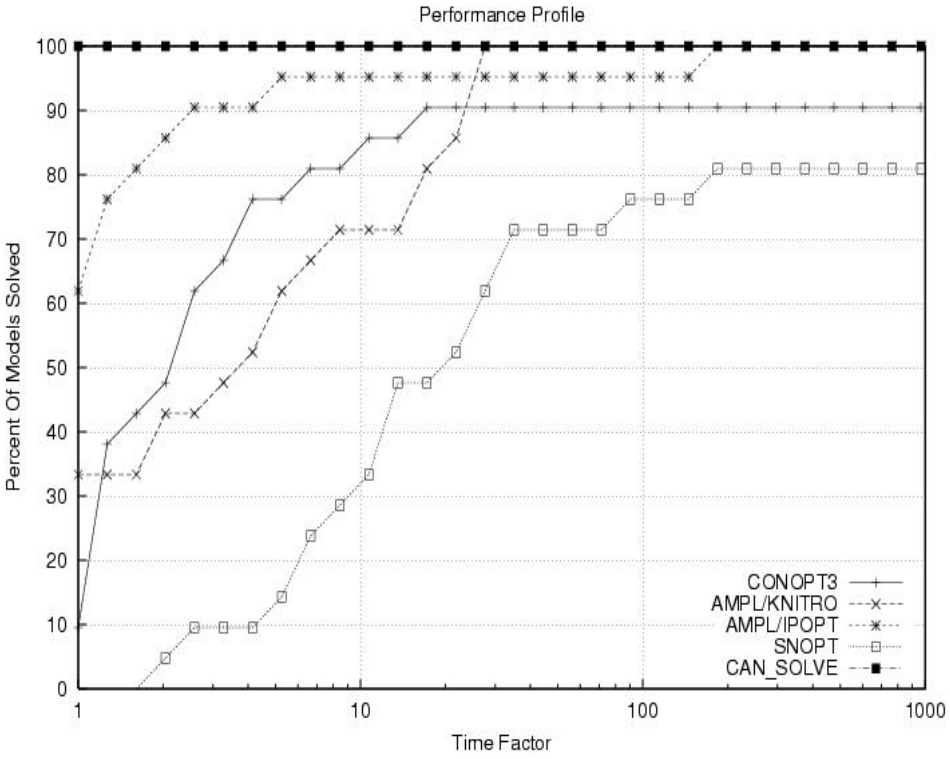

Figure 7: PAVER Performance Profile (Efficiency - IPOPT Best). Time Factor τ is in $\log_{10}$ scale.