The Optimization Test Environment Tutorial

Martin Fuchs[1]

[1]CERFACS, Parallel Algorithms Team, Toulouse, France
martin.fuchs81@gmail.com

July 27, 2010

Abstract. The Test Environment is an interface to efficiently test different optimization solvers. It is designed as a tool for both developers of solver software and practitioners who just look for the best solver for their specific problem class. It enables users to:

- Choose and compare diverse solver routines;
- Organize and solve large test problem sets;
- Select interactively subsets of test problem sets;
- Perform a statistical analysis of the results, automatically produced as LaTeX, PDF, and JPG output.

The Test Environment is free to use for research purposes.

This tutorial guides the reader through the installation and the basic functionality of the Test Environment illustrated with simple examples. We also give a short outlook on more advanced options and capabilities of the program.

We thank you for your interest in the Test Environment and hope you enjoy using it.

1 Installation

The installation of the TEST ENVIRONMENT is straightforward:

Download. The TEST ENVIRONMENT is available on-line at `http://www.mat.univie.ac.at/~dferi/testenv_download.html`.

Install. In Windows run the installer and follow the instructions. In Linux unzip the `tar.gz` file. Afterwards you can start the TEST ENVIRONMENT at the unzip location via `java -jar TestEnvironment.jar`.

Requirements. The graphical user interface (GUI) of the TEST ENVIRONMENT is programmed in Java, hence Java JRE 6, Update 13 or later is required to be installed. This is the only prerequisite needed.

Note that a folder that contains user specific files is created: for Windows the folder `TestEnvironment` in the application data subdirectory of your home directory; for Linux the folder `testenvironment` in your home directory. We refer to this directory as the TEST ENVIRONMENT working directory **twd**. All subdirectories of the twd are set as default paths in the TEST ENVIRONMENT configuration which can be modified by the user, cf. Section 3.1. The TEST ENVIRONMENT configuration file (`TestEnvironment.cfg`), however, remains in the twd.

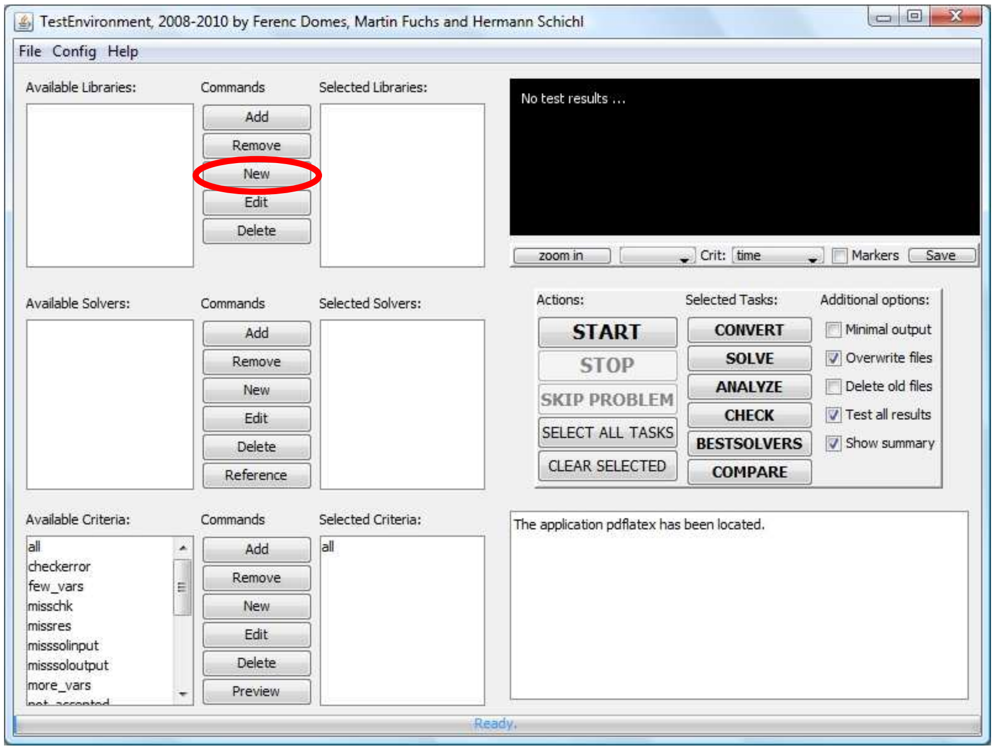
The TEST ENVIRONMENT does not include any solver software. Installation of a solver and obtaining a valid license is independent of the TEST ENVIRONMENT and up to the user.

2 First steps

This section guides you through the first steps after installation of the TEST ENVIRONMENT. You learn by way of a simple example how to add test problems, how to configure a solver, and how to run the TEST ENVIRONMENT on the given problems.

2.1 Adding a new test library

After starting the TEST ENVIRONMENT, the first step is to add a set of optimization problems, what we call a **test library**. The problems are assumed to be given as `.dag` files, an input format originating from the COCONUT Environment [3]. In case you do not have your problems given as `.dag` files, but as AMPL code, you need to convert your AMPL model to `.dag` files first which is possible via the COCONUT Environment or easily via a converter script separately available on the Test Environment website [1]. Also you can find a huge collection of test problem sets from the Coconut Benchmark [4] given as `.dag` files on the Test Environment website.



Additional options: [ ] Minimal output, [✓ Overwrite files], [ ] Delete old files, [✓ Test all results], [✓ Show summary]

The application pdflatex has been located.

Figure 2.1: Click the **New** button to set up a new test problem library.

Adding a new test library can be done in two ways: Either directly copy the `.dag` files to the directory of your choice within the `twd/libs` directory before starting the Test Environment. Or click the **New** button in the GUI as shown in Figure 2.1. This creates a new directory in `twd/libs` with the entered name. Then copy the `.dag` files into the directory created. There is also the possibility to use **Browse path** to select a directory outside the `twd`.

To start we create a new library **newlib**. We click the **New** button and enter 'newlib', then we copy 3 simple test problems to `twd/libs/newlib`:

2.1.1 Simple example test library

t1.dag: The intersection of two unit circles around $x = (\pm 0.5, 0)^T$, cf. Figure 2.2. The problem formulation is as follows:

$$
\begin{aligned}
\min_x \quad & x_2 \\
\text{s.t.} \quad & (x_1 - 0.5)^2 + x_2^2 = 1, \\
& (x_1 + 0.5)^2 + x_2^2 = 1, \\
& x_1 \in [-3, 3], x_2 \in [-3, 3].
\end{aligned}
\tag{2.1}
$$

The feasible points of (2.1) are $x = (0, \pm\sqrt{3}/2)^T$. Minimizing x_2 results in the optimal solution $\widehat{x} = (0, -\sqrt{3}/2)^T \approx (0, -0.8660)^T$.

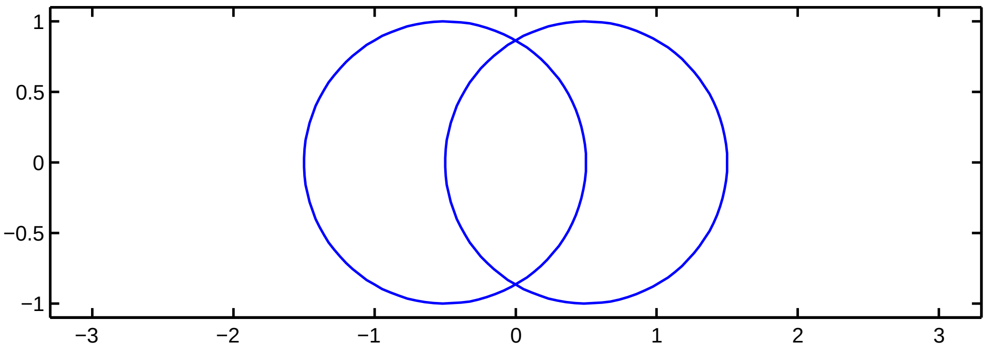


Figure 2.2: The feasible domain of (2.1) consists of the intersection points of two unit circles around $x = (\pm 0.5, 0)^T$.

t2.dag: Two touching unit circles around $x = (\pm 1, 0)^T$, cf. Figure 2.3. The problem formulation is as follows:

$$
\begin{aligned}
\min_x \quad & 1 \\
\text{s.t.} \quad & (x_1 - 1)^2 + x_2^2 = 1, \\
& (x_1 + 1)^2 + x_2^2 = 1, \\
& x_1 \in [-3, 3], x_2 \in [-3, 3].
\end{aligned}
\tag{2.2}
$$

The only feasible point of (2.2) is $x = (0, 0)^T$.

t3.dag: Two disjoint unit circles around $x = (\pm 2, 0)^T$, cf. Figure 2.4. The problem formulation is as follows:

$$
\begin{aligned}
\min_x \quad & 1 \\
\text{s.t.} \quad & (x_1 - 2)^2 + x_2^2 = 1, \\
& (x_1 + 2)^2 + x_2^2 = 1, \\
& x_1 \in [-3, 3], x_2 \in [-3, 3].
\end{aligned}
\tag{2.3}
$$

There is no feasible solution for (2.3).

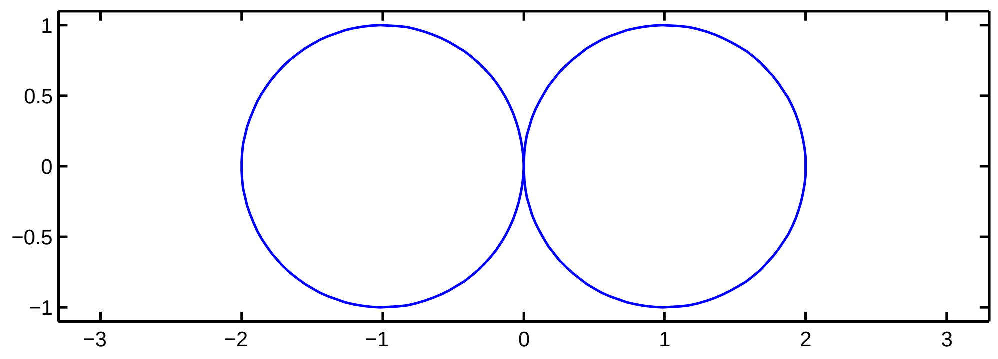


Figure 2.3: The only feasible point of (2.2) is at $x = (0,0)^T$.

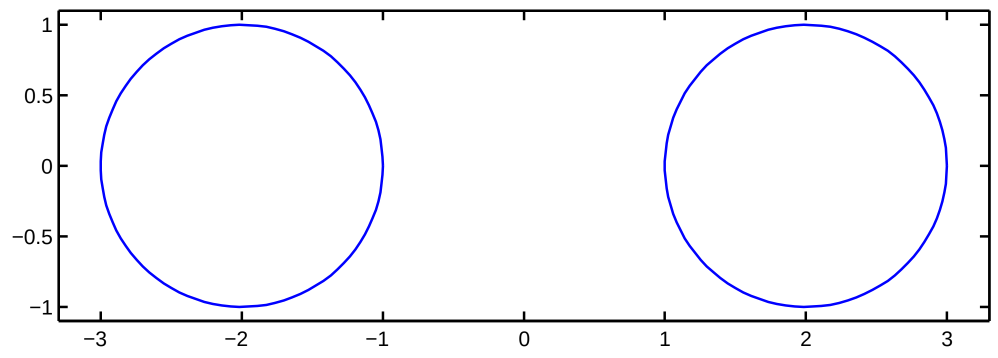


Figure 2.4: Obviously there is no feasible point for (2.3).

See Section 2.1.2 for the AMPL code of (2.1), (2.2), and (2.3), respectively. The AMPL code is converted to `.dag` files via the converter script mentioned. Also see Section 3.3 for some more details on conversion issues.

In our examples we actually know the results of the three problems. In this case the user can optionally provide reference solutions as described in Section 3.2.

The files `t1.dag`, `t2.dag`, and `t3.dag` have to be copied to `twd/libs/newlib` to finish the creation of the 'newlib' library. We also need to click **Edit**, and **Generate Dag Infos**. The **Library settings** window shows the number of correct dag info files, and it can also be used to view single problems from a test library and to check reference solutions, cf. Section 3.2. To select 'newlib' as one of the libraries that will be processed the user selects the 'newlib' entry among the 'Available Test Libraries' in the TEST ENVIRONMENT and clicks **Add**.

2.1.2 AMPL code of the example problems

t1.mod:
```
var x1 >=-3, <=3;
var x2 >=-3, <=3;

minimize obj: x2;

s.t. c1: (x1-0.5)^2+x2^2=1;
s.t. c2: (x1+0.5)^2+x2^2=1;
```

t2.mod:
```
var x1 >=-3, <=3;
var x2 >=-3, <=3;

minimize obj: 1;

s.t. c1: (x1-1)^2+x2^2=1;
s.t. c2: (x1+1)^2+x2^2=1;
```

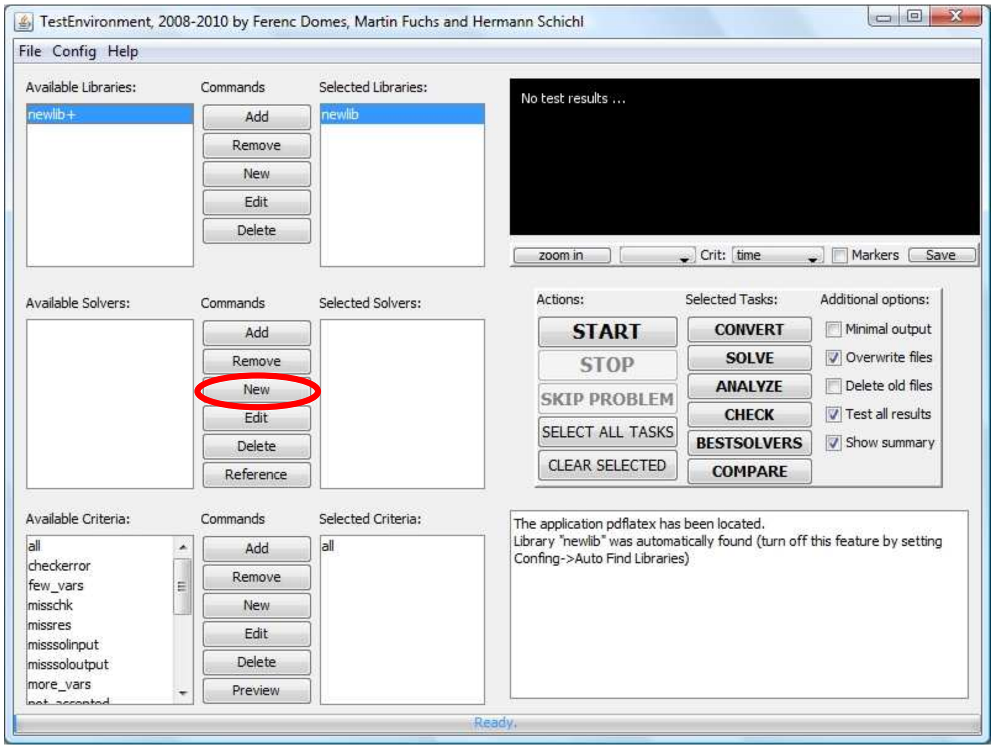
t3.mod:
```
var x1 >=-3, <=3;
var x2 >=-3, <=3;

minimize obj: 1;

s.t. c1: (x1-2)^2+x2^2=1;
s.t. c2: (x1+2)^2+x2^2=1;
```

2.2 Adding a solver

To add a solver to the 'Available Solvers' list we simply click the **New** button, cf. Figure 2.5. The 'Solver Configuration' window pops up.

First we enter a name. We solve our example test library using a KNITRO Student Edition with AMPL interface in its demo version (both are freely available on the internet), so we enter 'knitro'. Afterwards we enter as 'Input File Extensions' first 'mod', then 'lst'. The file extensions should consider the file types that are required in the different phases of the testing. In our case the solve phase uses `.mod` files and the solver output comes as an `.lst` file.

We use the predefined solver configuration for KNITRO from the Test Environment website [1] which provides the solver configurations and installation guides for many well-known solvers, also see Section 3.3. We download the configuration archive for KNITRO and extract it to the `twd/solvers/knitro` directory. We only need to modify the path names for the AMPL and KNITRO path, i.e., edit the fields 'Language path' and 'Solver path', respectively, in the solver configuration window.



TestEnvironment, 2008-2010 by Ferenc Domes, Martin Fuchs and Hermann Schichl

Available Libraries: newlib+

Selected Libraries: newlib

Available Solvers:

Selected Solvers:

Available Criteria: all, checkerror, few_vars, misschk, missres, misssolinput, misssoloutput, more_vars

Selected Criteria: all

No test results ...

- [ ] Minimal output
- [x] Overwrite files
- [ ] Delete old files
- [x] Test all results
- [x] Show summary

The application pdflatex has been located.
Library "newlib" was automatically found (turn off this feature by setting Confing->Auto Find Libraries)

Figure 2.5: Add a solver by clicking **New**.

To commit the modifications one clicks **Overwrite** and uses **Add** to select the KNITRO solver. Different configurations for the same solver can be managed by using the tab **Save & Load** in the solver configuration.

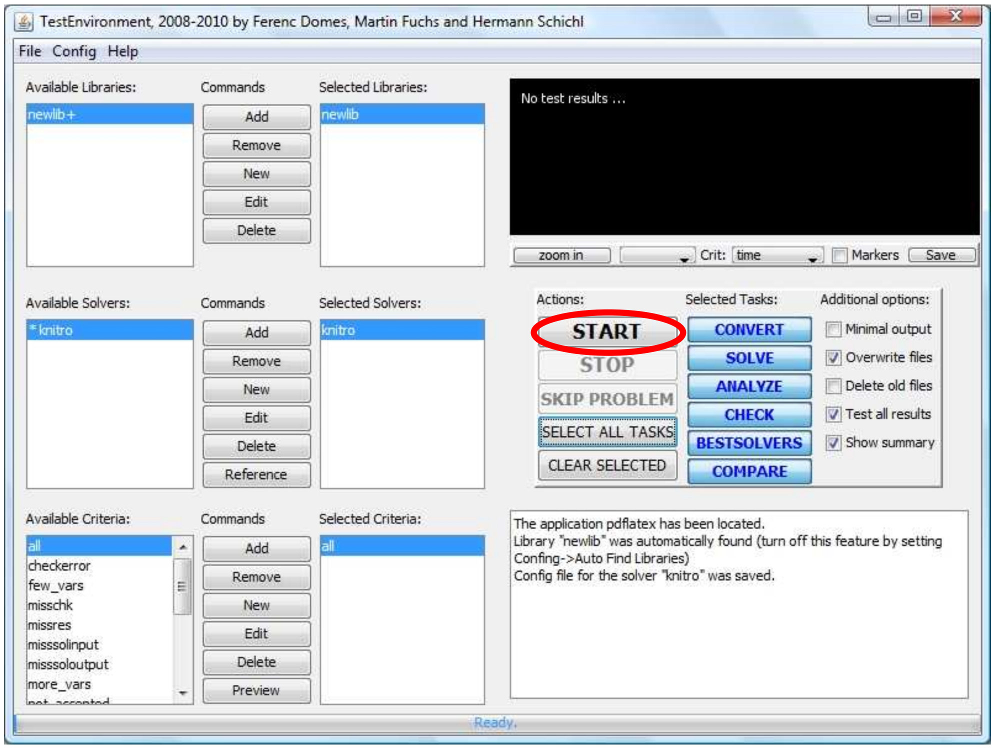
Eventually, one has to add a criterion from the 'Available Criteria'. We add 'all' (also see Section 3.4), and we are ready to solve and analyze the 'newlib' test problems.

2.3 Solve the test problems

Just hit the **Select All Tasks** button and press **Start**, cf. Figure 2.6.

The selected problems are solved by the selected solvers, the results are analyzed by the Test Environment, the output files (in LaTeX, pdf, and jpg) are written to the `twd/results` folder, and a pdf file shows up that summarizes the results.

There are essentially three different kinds of generated output pdfs. The `problems.pdf` analyzes the performance of the selected solvers on each problem from the selected test libraries. The `solvers.pdf` summarizes the performances of each selected solver according to each selected test library. The **summary** pdfs are similar to `solvers.pdf`, but they



Available Libraries: newlib+
Selected Libraries: newlib
Available Solvers: *knitro
Selected Solvers: knitro
Available Criteria: all, checkerror, few_vars, misschk, missres, misssolinput, misssoloutput, more_vars, not_accepted
Selected Criteria: all

No test results ...

[ ] Minimal output
[✓] Overwrite files
[ ] Delete old files
[✓] Test all results
[✓] Show summary

The application pdflatex has been located.
Library "newlib" was automatically found (turn off this feature by setting Confing->Auto Find Libraries)
Config file for the solver "knitro" was saved.

Figure 2.6: **Select All Tasks** and **Start**.

additionally provide a solver summary concerning the whole test set that consists of all selected test libraries.

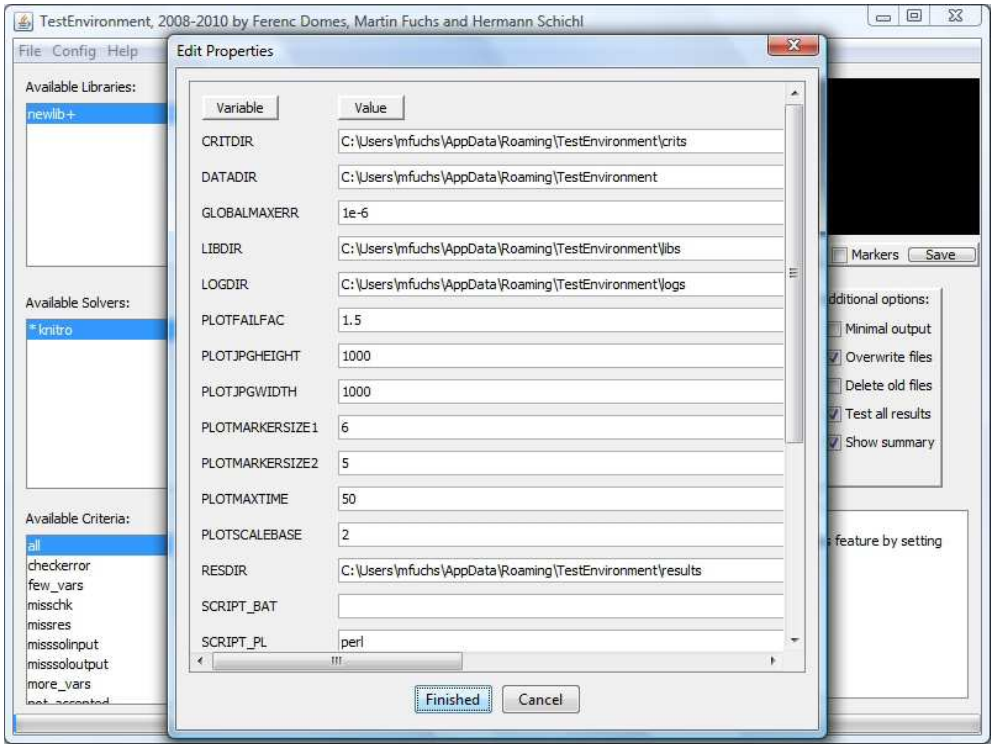
Moreover, we also generate performance profiles saved as jpg files in the results folder and plotted in the top right corner of the GUI.

3 Further things you can do

Now you know the basic procedure of solving a set of optimization problems within the Test Environment. Of course, there are plenty of things left that can be explored by the advanced user. In this section we introduce some of the most important options that the interested user will be looking for.

3.1 Change default paths

To change the default path location of your libraries, result files etc. one clicks **Config** → **Variables**. The menu opened enables the user to add variables and to modify variables, cf. Figure 3.1.



Variable	Value
CRITDIR	C:\Users\mfuchs\AppData\Roaming\TestEnvironment\crits
DATADIR	C:\Users\mfuchs\AppData\Roaming\TestEnvironment
GLOBALMAXERR	1e-6
LIBDIR	C:\Users\mfuchs\AppData\Roaming\TestEnvironment\libs
LOGDIR	C:\Users\mfuchs\AppData\Roaming\TestEnvironment\logs
PLOTFAILFAC	1.5
PLOTJPGHEIGHT	1000
PLOTJPGWIDTH	1000
PLOTMARKERSIZE1	6
PLOTMARKERSIZE2	5
PLOTMAXTIME	50
PLOTSCALEBASE	2
RESDIR	C:\Users\mfuchs\AppData\Roaming\TestEnvironment\results
SCRIPT_BAT	
SCRIPT_PL	perl

Figure 3.1: Here you can change the default paths or add or modify further variables.

Among these variables are the default paths for the criteria selection (CRITDIR), the test libraries (LIBDIR), the log files (LOGDIR), the result files (RESDIR), the data directory (DATADIR, typically the parent directory of the directories before), and the solver configurations (SOLVEDIR).

Also a general timeout can be set as TIMEOUT (for more details on timeouts see Section 3.4).

The variables 'SCRIPT_<ext>' define how scripts with the extension <ext> should be called.

The variables 'PLOT' adjust the setting of the plot in the top right corner of the GUI.

3.2 Provide reference solutions

As mentioned in Section 2.1.1, a reference solution is a known solution for one of the test problems in our test problem library. It can be provided simply as a `.res` file which has to be put into the `twd/libs/newlib_sol` folder. This folder is automatically generated upon creation of the 'newlib' entry.

The generic format of `.res` files within the Test Environment for our example problem `t1.dag` reads as follows:

```
modelstatus = 0
x(1) = 0
x(2) = -0.8660254037844386
obj = -0.8660254037844386
infeas = 0.00
nonopt = 0.00
```

which we write to `t1.res`. There are several fields that can be entered in .res files. See [2] for more details. For problem `t2.dag` we provide a **wrong** solution in `t2.res`:

```
modelstatus = 0
x(1) = 1
x(2) = 2
obj = 0
infeas = 0.00
nonopt = 0.00
```

For problem `t3.dag` we do not provide any `.res` file.

To finish the setup of the 'newlib' test library with the given reference solutions we click **Edit**, and finally **Check Solutions**, cf. Figure 3.2. Note that the solution check for the reference solution of `t2.dag` has failed as we provided a wrong solution.

3.3 Advanced solver configuration

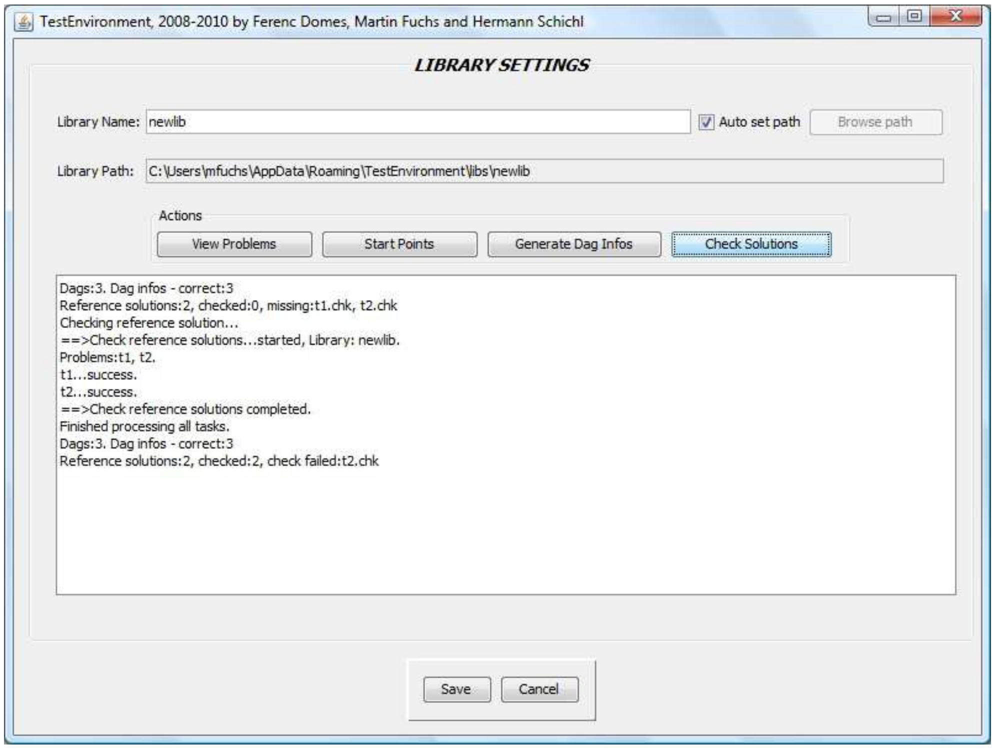
The Test Environment website [1] offers a collection of predefined solver configurations and instructions how to use them, such as the configuration we used to set up the KNITRO solver with AMPL interface in Section 2.2.

In our example we use one command line string and two scripts. Note that for the Linux version the quotation marks have to be omitted and `\` has to be replaced by `/`. The first command line string is in charge of the conversion of the problem `.dag` files to the AMPL modeling language:

```
"%<EXTERNALDIR>%\dag2ampl" "%<SOURCEDIR>%\%<PROBLEM>%"...
        ..."%<TARGETDIR>%\%<PROB>%.%<OUTEXT>%
```

The `solvescriptknitro.py` script calls the solver. The Test Environment calls this script as

```
%<SCRIPT_PY>% "%<SOURCEDIR>%\solvescriptknitro.py" "%<SOURCEDIR>%" "%<TARGETDIR>%"...
        ...%<PROBLIST>% "%<SOLVERPATH>%" "%<LANGUAGEPATH>%" %<TIMEOUT>%
```


TestEnvironment, 2008-2010 by Ferenc Domes, Martin Fuchs and Hermann Schichl

LIBRARY SETTINGS

Library Name: newlib ☑ Auto set path

Library Path: C:\Users\mfuchs\AppData\Roaming\TestEnvironment\libs\newlib

```
Dags:3. Dag infos - correct:3
Reference solutions:2, checked:0, missing:t1.chk, t2.chk
Checking reference solution...
==>Check reference solutions...started, Library: newlib.
Problems:t1, t2.
t1...success.
t2...success.
==>Check reference solutions completed.
Finished processing all tasks.
Dags:3. Dag infos - correct:3
Reference solutions:2, checked:2, check failed:t2.chk
```

Figure 3.2: Test library settings

which would be a command line string equivalent to entering only the script name in the solver configuration (i.e., `solvescriptknitro.py`). Upon solving there is also a file `Acpu` created containing CPU information of the current machine.

A solver is considered to be **incompatible** with the Test Environment if it does not enable the user to set the maximal allowed time for the problem solution process, which is used by the Test Environment to compare different solvers by setting the same value `%<TIMEOUT>%` for each of them.

To be able to call a solver it is necessary to know the solver location, i.e., the `%<SOLVERPATH>%` and/or the location of an associated modeling language, i.e., the `%<LANGUAGEPATH>%`. Hence we pass all these variables together with the problem list `%<PROBLIST>%` to the solve script. The solver path and the language path can be set in the corresponding boxes in the solver configuration GUI which also allows to set environment variables if this is needed for some solver.

All variables, e.g., `%<PROBLIST>%`, are also explained in the Help tab of the solver configuration. If the available set of variables is not sufficient to satisfy the user's needs there is additionally the possibility to create user-defined variables as described in Section 3.1.

After calling the solver, the script `analyzescriptknitro.py` is used to generate the `.res` files from the solver output. Help on the required fields in a `.res` file can be found by pressing F1 or in Section 3.2. Analyze scripts (and convert scripts analogously) are called by the Test Environment in the same way as solve scripts, i.e.,

```
%<SCRIPT_PY>% "%<SOURCEDIR>%\analyzescriptknitro.py" "%<SOURCEDIR>%" "%<TARGETDIR>%"...
        ...%<PROBLIST>% "%<SOLVERPATH>%" "%<LANGUAGEPATH>%" %<TIMEOUT>%
```

Note that complete `.res` files (cf. Section 3.2) **must** be produced even if no feasible solution has been found or a timeout has been reached or similar. Writing the solve and analyze scripts for solvers that are not available on the Test Environment website requires basic knowledge about scripting string manipulations which is an easy task for advanced users and especially for solver developers.

Also note that the original solver output is also kept and stored, i.e., in our example in the directory `twd/solvers/knitro/res/newlib_<name>` where `<name>` is an abbreviation of the name of the computer.

We explicitly **encourage** people who have implemented a solve or analyze script for the Test Environment to send it to the authors, who will add it to the Test Environment website.

Some solvers can be called setting a command line flag in order to generate `.res` files directly without the need of an analyze script.

3.4 Criteria selection

The criteria selection is another core feature of the Test Environment. The criteria editor can be accessed via the **New** button to create a custom criterion. The criteria are used to specify subsets of test libraries. There are many possibilities to specify the selection of test problems from the libraries by way of connected logical expressions, e.g., `[variables<=10]and[int-vars<=5]` restricts the selected test libraries to those problems with less than $n = 10$ variables and less than 5 integer variables. A criterion is defined by such a connection of logical expressions. The creation of a criterion in the Test Environment GUI is straightforward using the 'Condition' box together with the 'Logicals' box of the 'Criteria editor'.

A click on the **Preview** button in the main Test Environment window shows all selected criteria in correspondence to all selected problems that meet the criteria.

The Test Environment already includes a collection of predefined criteria in the 'Available criteria' box, such as, e.g., `few_vars` to choose problems with only a few variables setting a timeout of 2 minutes, or `misssoloutput` to choose problems for which a solver output file is missing.

One important feature of criteria is their use in setting **timeouts** in connection with further conditions like problem size. For example create 4 criteria:

size1:	`[variables>=1]and[variables<=9]and[timeout==180]`
size2:	`[variables>=10]and[variables<=99]and[timeout==900]`
size3:	`[variables>=100]and[variables<=999]and[timeout==1800]`
size4:	`[variables>=1000]and[timeout==1800]`

and add them to the selected criteria. Any task on the selected test library will now be performed first on problems with $1 \leq n \leq 9$ and timeout 180 s, then on problems with $10 \leq n \leq 99$ and timeout 900 s, then problems with $100 \leq n \leq 999$ and timeout 1800 s, and eventually problems with $n \geq 1000$ and timeout 1800 s.

This concludes the tutorial. There are still several functionalities unreported, e.g., more details on custom solver configurations, numerical tests for large test problem sets and comparing different solvers using the Test Environment. The interested reader is referred to [2] and the Test Environment website [1].

References

[1] F. Domes. Test Environment website, `http://www.mat.univie.ac.at/~dferi/testenv.html`, 2009.

[2] F. Domes, M. Fuchs, and H. Schichl. The optimization test environment. Submitted, 2010. Preprint available on-line at: `http://www.mat.univie.ac.at/~dferi/testenv.html`.

[3] H. Schichl et al. The COCONUT Environment, 2000–2010. Software. URL `http://www.mat.univie.ac.at/coconut-environment`.

[4] O. Shcherbina. COCONUT benchmark, `http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html`, 2009.