The Task Space Searcher (TSS) Project Report

Tyler Bletsch (tkbletsc [AT] unity.ncsu.edu)
North Carolina State University
For CSC714, Prof. Mueller
30 November 2005

Introduction

I implemented a schedulability analysis tool in Perl that supports EDF and DM scheduling policies, as well as the PIP and PCP priority schemes. This tool allows one to search a problem space for feasible task sets.

This document reports the final development status. Full project information is available at http://www4.ncsu.edu/~tkbletsc/714/project.html. The **current source code** is available there, as well.

Development Summary

I successfully implemented almost all of the requirements I set for myself, and created what I consider to be a fairly useful tool for real-time development. The final product supports the following:

- Command-line interface with multiple output formats:
  - A simple grep-able list of schedulable task sets
  - An ASCII table of every task-set with pass/fail result & test used
  - A highly detailed hierarchical view of every step of the analysis
  - A simple list of all task sets (sans analysis)
  - A machine-readable Comma-Separated Values (CSV) file suitable for Excel, Gnumeric, etc.
- Specification of tasks with full support for the combinatorial variables I devised
- EDF analysis: utilization analysis with blocking
- DM analysis: utilization analysis with blocking, GTDA with blocking
- Blocking term calculation under PIP and PCP

These features are adequately demonstrated using the sample input files included with the program distribution (`test*.tss`, see comments within each file for details). During the development process, the project evolved significantly from the original proposal. For example, there were a number of features that turned out to be unnecessary:

- Phase is unrelated to timing analysis, so it can be omitted.
- GTDA reduces to TDA when all $D_i \leq p_i$, so a separate TDA implementation isn't needed.

Finally, I was unfortunately not able to fulfill every goal I set for myself. Specifically, my implementation of resource locking does not allow recursive locks at present. This would be a significant step forward, and would make for excellent future development. Other sources of future development include:

- Applying additional reasoning to rule out task sets without performing analysis on them
- Adding support for parallel execution to combat exponential growth in running times
- Allowing for random sampling of a large number of task sets rather

Completed Milestone History

24 Oct: TDA (without blocking term)

This milestone has been completed. This tool had been started but never completed during my earlier development for previous homework assignments and my own studying efforts.

31 Oct: Generalized TDA (without blocking term)

This milestone has also been completed. Much of the work had already been done, as I developed a simple TDA tool for previous homework assignments and my own studying efforts.

7 Nov: System density (not including blocking term)

This milestone was completely straightforward; the summation involved was a simple calculation.

14 Nov: File parser

This was a major step, but some regular expression magic did most of the heavy lifting. With this step completed, the TSS is now a functioning stand-alone tool. To get started, just run the program can be run without arguments to find out the calling syntax. The analysis is completely implemented with the exception of the blocking term (see next milestones).

21 Nov: Blocking term b_i calculation for PIP and PCP

This was a straightforward undertaking. The mechanics of blocking term calculation were covered in the course notes, and I had no trouble implementing them. However, I did not have time to include support for full recursive locks, so only sequential locks are supported.

28 Nov: Add blocking term b_i to all analysis methods

This introduced a bit more work than expected to the utilization measure. As discussed in the Liu text (pages 163-164), utilization must be checked on a per-task basis rather than a global basis. The change to GTDA, on the other hand, was simply the addition of one extra term.

Program usage (taken from the README)

INSTALLATION

Simply extract the archive to the directory of your choice. You can get started quickly by using the included sample input files as a template. The following sample session demonstrates extraction, shows a sample input file, shows the usage information, and finally runs the sample file through the system:

Sample session

```
$ tar -xvzf tss.tar.gz
tss.pl
test1.tss
test2.tss
test3.tss
Task.plx
TaskSet.plx
blocking.plx
common.plx
gtda.plx
util.plx
README

$ cat test1.tss
# Here's a basic input file.  Some of the tasks are schedulable, some aren't.
# Note what the test results show regarding what is schedulable under DM.

task [1,16,3];[1,5,0.1]
task 7;2;7
task 11;4;10

try DM with PIP
try EDF with PIP
```

```
$ tss.pl
Task Space Searcher (TSS) by Tyler Bletsch

A schedulability analysis tool for real-time applications.  See project
homepage for complete documentation.

Usage:
  tss.pl [-v[#]] [-c<CSVfile>] <TSSFile>

Options:
  -v  Set verbosity to #, or 1 if no number is specified. Defaults to 0. Key:
        0 = Emit successful task sets only
        1 = Print TaskSets & analysis result
        2 = Print all of 1, and include in-depth calculation details
  -c  Output results in Comma-Separated Values (CSV) format to <CSVFile>.
  -p  Just print the cases that would be tested (but don't test them).

Arguments:
    TSSFile: A Task Space Searcher input file, see documentation for format


$ tss.pl test1.tss
DM/PIP:  (13,1,13) (7,2,7) (11,4,10)
DM/PIP:  (16,1,16) (7,2,7) (11,4,10)
DM/PIP:  (13,1.1,13) (7,2,7) (11,4,10)
DM/PIP:  (16,1.1,16) (7,2,7) (11,4,10)
DM/PIP:  (13,1.2,13) (7,2,7) (11,4,10)
DM/PIP:  (16,1.2,16) (7,2,7) (11,4,10)
DM/PIP:  (16,1.3,16) (7,2,7) (11,4,10)
DM/PIP:  (16,1.4,16) (7,2,7) (11,4,10)
DM/PIP:  (16,1.5,16) (7,2,7) (11,4,10)
EDF/PIP: (4,1,4) (7,2,7) (11,4,10)
EDF/PIP: (7,1,7) (7,2,7) (11,4,10)
EDF/PIP: (10,1,10) (7,2,7) (11,4,10)
EDF/PIP: (13,1,13) (7,2,7) (11,4,10)
EDF/PIP: (16,1,16) (7,2,7) (11,4,10)
EDF/PIP: (4,1.1,4) (7,2,7) (11,4,10)
     ... a lot of results ...
```

OPTIONS

You can get increasing amounts of output on stdout by specifying higher verbosity with the `-v` option. Use the highest level, `-v2`, to get a detailed hierarchical report of every step of the analysis. You can output the results of analysis to a Comma-Separated Values (CSV) file with the `-c` option. This file can be opened readily in Excel, Gnumeric, OpenOffice Calc, and most any other table-oriented data analysis tool. Finally, you can use `-p` to simply get a list of all task sets (without analysis).