MMSConjecture User Guide

Version 1.0

Derrick Stolee
University of Illinois
stolee@illinois.edu

January 7, 2013

Abstract

The *MMSConjecture* software is an implementation a branch-and-cut method using linear programs to prove the Manickam-Miklós-Singhi Conjecture for fixed values of k. This software is in support of the paper "A Branch-and-Cut Strategy for the Manickam-Miklós-Singhi Conjecture" by Hartke and Stolee [1].

1 Acquiring *MMSConjecture*

The latest version of *MMSConjecture* and its documentation is available online as part of the *SearchLib* collection at the address

`http://www.math.illinois.edu/~stolee/SearchLib/`

MMSConjecture is made available open-source under the GPL 3.0 license.

Fix a workspace directory which contains the `MMSConjecture` folder. Install the *TreeSearch* and *Utilities* projects into the `TreeSearch/` and `Utilities/` folders in the workspace. To compile *MMSConjecture*, use a terminal to access the `MMSConjecture/src/` folder. There are two different types of executables:

1. To use GLPK, install GLPK into a directory `glpk/` in the workspace directory. Type `make -f Makefile-GLPK` to compile `mmsconj-glpk.exe`.
2. To use CPLEX, install CPLEX into a directory `cplex/` in the workspace directory. Type `make -f Makefile-CPLEX` to compile `mmsconj-cplex.exe`.

The executables will be placed in `MMSConjecture/bin/`

1.1 Acquiring Necessary Libraries

There are two external libraries and two *SearchLib* projects used by *MMSConjecture*.

1. *GLPK*, the GNU Linear Programming Kit, is an open-source linear programming interface. When using GLPK with *MMSConjecture*, the exact solver is used.

   `http://www.gnu.org/software/glpk/`

2. *CPLEX*, the IBM ILOG CPLEX Optimizer, is a commercial solver. An academic license is available.

http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

3. *TreeSearch* is a project in *SearchLib* that abstracts the structure of a backtrack search in order to allow for parallelization. *TreeSearch* is available on the same web site as *MMSConjecture*. Consult the *TreeSearch* documentation for details about the arguments and execution processes.

4. *Utilities* is a project in *SearchLib* containing useful objects and functions necessary by other projects in *SearchLib*. *Utilities* is available on the same web site as *MMSConjecture*.

1.2 Full Directory Structure

For proper compilation, place the different dependencies in the following directory structure:

- `MMSConjecture/` – The *MMSConjecture* project.
  - `src/` – Contains source code. Compilation occurs here.
  - `bin/` – The final binaries are placed here.
  - `scripts/` – Contains shell scripts with the parameters required to verify the conjectures. See Section 4.
  - `data/` – Contains data files from previous executions of the programs, including detailed timing information. See Section 5.
  - `docs/` – This folder contains documentation, including this guide.
- `TreeSearch/` – A support project from *SearchLib*.
- `Utilities/` – A support project from *SearchLib*.
  - `src/` – Type `make` in this directory to compile the *Utilities* project.

2 Execution

There are three executables in the *MMSConjecture* project.

- `mmsconj-glpk.exe`
- `mmsconj-cplex.exe`
- `mmslight-cplex.exe`

2.1 `mmsconj-glpk.exe` and `mmsconj-cplex.exe`

The first two executables have identical behavior except for the linear program solver implementations. **Note:** The `mmsconj-cplex.exe` executable requires `libcplex123.so` to be present in the current directory, and the shell variable `ILOG_LICENSE_FILE` to be set to the path of the license file.

```
mmsconj-*.exe [TreeSearch args] -K # -N # [--target #] [-gm#v#] [--prop
  none|gac|sgac|bsgac|pos] [--strong] [--rule balanced|middle|opthalf]
                [--proof] [--stochastic] [--branchless]
```

- `-K #` – Let # be the number k, for the size of the partial sums.
- `-N #` – Let # be the number n of variables.
- `--target #` – Let # be the target t.
- `--strong` – If present, will enforce that the set $\{1, n-k+2, \ldots, n\}$ has negative sum and hence the non-negative sums will not include the star example.
- `-gm#v#` – If present, the first # is a number m and the second number # is a value v such that $g(m,k) = v$. When $m = n - k$, this value is used in the initial propagation step (see Algorithm 3 [**?**]) to create more negative k-sums.
- `--stochastic` – If present, the propagation step will use STOCHASTICPROPAGATION, a method for randomly selecting a set and testing feasibility with that set chosen.
- `--branchless` – If present, the propagation step will use BRANCHLESSSEARCH, which will test all sets using the LP.
- `--prop none|gac|sgac|bsgac|pos` – Select the propagation method:
  - `none` – Skips the propagation step. (Not recommended.)
  - `gac` – Global Arc Consistency checks at every step if making a set non-negative would increase the number of sets to at least the target value, and if so adds the constraint that the set must be strictly negative (see Algorithm 4 [**?**]). This is the option used for the performance statistics in [**?**].
  - `sgac` – Singleton Global Arc Consistency checks at every step what Global Arc Consistency would do if we changed any single set to non-negative or positive. If one change results in a contradiction, then we take the other choice. (Not recommended.)
  - `bsgac` – Branch Singleton Global Arc Consistency selects the set that would be chosen by the branching rule with the current configuration and tests each branch option. If GAC then reports a contradiction, it sets the other option without requiring a branch. Useful to reduce the depth of the search tree, but usually slower. (Not recommended.)
  - `pos` – Positive Propagation performs the check for Global Arc Consistency (which forces some sets to be negative) and then checks remaining sets if setting them to be strictly negative keeps the linear program feasible. (This is very similar to performing BRANCHLESSSEARCHat every search node.) If the LP check reports infeasible, the set is set to positive (see Algorithm 5 [1]). This check greatly slows the search, but reduces the search tree significantly. This option is used when printing the proof for the case $k = 4$.
- `--rule balanced|middle|opthalf` – Select the branching rule.
  - `balanced` – Select the set S which maximizes $\min\{|\mathcal{L}(S) \cap \mathcal{C}^*|, |\mathcal{R}(S) \cap \mathcal{C}^*|\}$. The goal is to guarantee the most sets change when S is set to be negative or non-negative. This option is used in the performance statistics.
  - `middle` – Select the set which has the median co-lexicographic index among sets in $\mathcal{C}^*$. (Not recommended.)
  - `opthalf` – Using the current LP optimal vector $\mathbf{x}$, find the set S with current sum $\sum_{i \in S} x_i$ closest to $-\frac{1}{2}$, since the branch will either select the sum to be at least 0 or at most -1. (Not recommended.)

- `--proof` – If present, the steps of a proof will be written to standard out. In particular, at the propagation steps, the choices of sets to set as non-negative or negative will be written. Also, at the branching steps, cases will be created to split the search tree. Not recommended in general, but used to write the proofs for the cases $k = 3$ and $k = 4$ in the Appendix of [1].

2.2 `mmslight-cplex.exe`

```
mmslight-cplex.exe [TreeSearch args] -K # -N # [--target #] [-gm#v#]
[--strong] [--stochastic] [--steps #] [--samples #] [--sampletime #]
```

- `-K #` – Let # be the number k, for the size of the partial sums.
- `-N #` – Let # be the number n of variables.
- `--target #` – Let # be the target t.
- `--strong` – If present, will enforce that the set $\{1, n-k+2, \ldots, n\}$ has negative sum and hence the non-negative sums will not include the star example.
- `-gm#v#` – If present, the first # is a number m and the second number # is a value v such that $g(m, k) = v$. When $m = n - k$, this value is used in the initial propagation step (see Algorithm 3 [**?**]) to create more negative k-sums.
- `--stochastic` – If present, the propagation step will use STOCHASTICPROPAGATION, a method for randomly selecting a set and testing feasibility with that set chosen.
- `--branchless` – If present, the propagation step will use BRANCHLESSSEARCH, which will test all sets using the LP.
- `--steps #` – Specify the number of steps in the probability distribution for selecting a random set in $\mathcal{C}^*$ (weighted to the left).
- `--samples #` – Specify the number M such that if M random samples are not decided to be positive or negative in STOCHASTICPROPAGATION, we should terminate the propagation and return to branching.
- `--sampletime #` – Specify the maximum number of seconds to spend in STOCHASTICPROPAGATION.

3 *TreeSearch* Arguments

- `-k #` — The killtime: How many seconds before halting the process and reporting a partial job.
- `-m #` — The maximum depth: the maximum number of steps to go before halting (or in generation mode, a new job is written at this depth).
- `run` — Run mode: The input jobs are run until finished or the killtime is reached.
- `generate` — Generation mode: The input jobs are run and new jobs are listed when reaching the maximum depth.
- `--maxjobs #` — The maximum number of jobs to generate before halting with a partial job (default: 1000).

- `--maxsols #` — The maximum number of solutions to output before halting with a partial job (default: 100).

4 Execution with CPLEX

Our implementation uses IBM ILOG CPLEX version 12.3. To compile and execute any file using CPLEX, we require the binary file `libcplex123.so` in the current directory. Further, for execution, the environment parameter `ILOG_LICENSE_FILE` must be set to the file containing the license information. Typically, this is contained in `cplex/licenses/access.ilm`.

5 Scripts

In the `scripts/` folder, there are some files which allow for execution of the

6 Data

The `data/` folder contains some output from previous executions, as well as tabulated performance data.

References

[1] S. G. Hartke, D. Stolee, A Branch-And-Cut Strategy for the Manickam-Miklós-Singhi Conjecture, preprint (2013).

[2] D. Stolee, TreeSearch user guide, available at http://www.math.illinois.edu/ stolee/SearchLib/TreeSearch.pdf (2011).