SATFC MANUAL

ALEXANDRE FRÉCHETTE[1], KEVIN LEYTON-BROWN[1]

September 25, 2014

CONTENTS

* *Department of Computer Science, University of British Columbia, British Columbia, Canada.*

1 INTRODUCTION

ABSTRACT SATFC (*SAT-based Feasibility Checker*) solves radio-spectrum repacking feasibility problems arising from simulations of the FCC's upcoming reverse auction. It combines a formulation of feasibility checking based on propositional satisfiability with a heuristic pre-solver and a SAT solver tuned for the type of instances observed in auction simulations.

AUTHORS & COLLABORATORS SATFC is the product of the ideas and hard work of Auctionomics, more specifically: Alexandre Fréchette, Guillaume Saulnier-Comte, Nick Arnosti, and Kevin Leyton-Brown.

CONTACT Any question, bug report or feature suggestion should be directed to Alexandre Fréchette - afrechet@cs.ubc.ca.

1.1 Licenses

1.2 System Requirements

SATFC is primarily intended to run on Unix-like platforms. It requires Java 7 to run (although greater stability has been observed with Java 8). One also needs our modified version of the SAT solver CLASP v2.2.3 compiled for JNA library usage, which in turn necessitates GCC v4.8.1 or higher as well as the standard Unix C libraries (see Section 3 to learn how to compile CLASP).

1.3 Version

This manual is for SATFC v1.1.0a.

2 DESCRIPTION

2.1 Problem

At the core of the reverse auction part of the radio-spectrum incentive auction lies the problem of (re)assigning telecom stations to a smaller range of broadcast channels than what they are on currently while satisfying interference constraints (often referred to as the STATION PACKING PROBLEM or the FEASIBILITY CHECKING PROBLEM). As of now, these constraints are pairwise in between stations, hence the whole problem can be cast to an (extended) GRAPH COLORING PROBLEM, where stations are nodes of the graph, edges correspond to interference constraints and colors to available broadcast channels. Unfortunately, the latter problem is known to be **NP**-complete, *i.e.* computationally challenging. Furthermore, as per the latest reverse auction designs, feasibility checking must be executed very frequently, sometimes upwards of a thousand times per auction round. It is thus considered as the fundamental computational bottleneck of the auction. Fortunately, the distribution of feasibility checking problems encountered in a typical auction is very specific, and that is what SATFC leverages.

2.2 SATFC

To take advantage of the specific distribution of encountered feasibility checking problems, SATFC first casts feasibility checking to its propositional satisfiability (SAT) version. This allows us to leverage the vast array of high performance SAT solver developed over the last years, and more specifically their tuning flexibility. Through a series of extensive empirical evaluations, we have identified SAT solver

CLASP as the best solver when tuned on our specific instance set using SMAC, a sequential, model-based algorithm configurator. In addition to using a highly configured version of clasp, SATFC solves easy instances using a heuristic pre-solving algorithm based on previous partial satisfiable assignments. Finally, some engineering is involved in making the whole pipeline as efficient as possible.

2.3 Efficient Usage

SATFC 's development was symbiotic to the concurrent design of reverse spectrum auctions. Hence, it really shines in the setting arising from the specific practical auction designs that are being studied by the FCC. For example, in these designs, one expects to encounter many problems to be solved sequentially, a good fraction of which are simple. SATFC is engineered around these characteristics (and many more), extensively leveraging any given partial feasible assignment to get rid of easy instances quickly, and having a main solver tuned for a specific (empirically defined) distribution of harder instances.

3 USAGE

3.1 Setting Up

Packaged with SATFC are its source code, the necessary libraries, execution scripts as well as a copy of CLASP . The latter needs to be compiled on your machine:

```
> cd satfc-v1.x.xa-release-x/clasp/
> bash compile.sh
```

3.2 Standalone

> **Warning**
> Everytime SATFC is launched from the command-line, it will have to load the Java Virtual Machine, necessary libraries as well as any constraint data corresponding to the specified problem. This is a significant overhead if one is to solve easy instances or a lot of instances. In the latter case, and if efficiency is a primary concern, it is suggested to use SATFC as a Java library.

To use SATFC from the command line to solve feasibility checking problems, go in the SATFC directory and execute the following:

```
> bash satfc -DATA-FOLDERNAME <data folder> -DOMAINS <station domains>
```

where

- `data folder` points to a folder containing the *broadcasting interference constraints data* - see Section 3.2.1 for further information, and
- `station domains` is a string containing each station, and for each station the list of channels to try to pack it. This *domains string* consists of *single station domains strings* joined by ';', where each single domain string consists of a station numerical ID, a ':', and a list of integer channels joined by ','. One can also run

  ```
  > bash satfc --help
  ```

  to get a list of the SATFC options and parameters.

  For example, wanting to pack station 1 and 2 in channels 14,15,16,17,18,19,20 and station 3 in channels 14,15,16 would be done with the following string:

```
1:14,15,16,17,18,19,20;2:14,15,16,17,18,19,20;3:14,15,16
```

3.2.1 *Data*

The broadcasting interference constraint data (as specified by the FCC) consists of two files, one specify station domains, and one specifying (pairwise) interference between stations. As part of its essential arguments, SATFC on the command-line expects a path to a folder containing both of these files exactly named `domains.csv` and `interferences.csv`, respectively.

DOMAINS The domains file consists of a CSV with no header where each row encodes a station's domain (or list of channels on which a station can broadcast). The first entry of a row is always the `DOMAIN` keyword, the second entry is the station ID, and all subsequent entries are domain channels. Note that SATFC

INTERFERENCES The interferences files consists of a CSV with no header where each row is a compact representation of many pairwise interference constraints between a single subject station on a possibly many subject channels with possibly many target stations. Specifically, the entries of a row are, in order, a key indicating the type of constraint, a lowest and highest channel between which the constraint applies, the subject station of the constraint, and then a list of target stations to which the constraint applies. Here it is in more compact format:

```
<key>,<low channel>,<high channel>,<subject station>,<target station 1>,...
```

There are two possible constraint keys: `CO` and `ADJ+1`. The former indicates a *co-channel constraint*, stating that the subject station and any target station cannot be on the same channel, for any channel in between the low and high channel limits. The latter describes an *adjacent plus one constraint* which implies a co-channel constraint as well as stating that, for any channel c in between the low and the high channel, the subject station cannot be on channel c together with any target station on channel $c+1$.

Here are two examples of interference constraints:

```
CO,14,20,1,2,3
```

This constraint means that for any channel c between 14 and 20, station 1 and 2 cannot jointly on c, as well as station 1 and 3.

```
ADJ+1,14,20,1,2,3
```

This constraints implies the corresponding the co-channel constraint "`CO,14,20,1,2,3`", as well as the following: for any channel c between 14 and 20, station 1 cannot be on channel c together with station 2 on $c+1$, and station 1 cannot be on channel c together with station 3 on $c+1$.

3.2.2 *Parameters & Options*

In addition to the required arguments to solve a problem from the command-line, SATFC also has the following options:

- `--help` - display SATFC options and parameters, with short descriptions and helpful information.
- `-PREVIOUS-ASSIGNMENT` - a partial, previously satisfiable channel assignment. This is passed in a string similar to the `-DOMAINS` string: the station and previous channel pairs are separated by ':', and the different pairs are joined by ','. For example, "`1:14,15,16;2:14,15`" means pack station 1 into channels 14,15 and 16 and station 2 into channels 14 and 15.
- `-CUTOFF` - a cutoff time, in seconds, for the SATFC execution.

Warning

It is important to keep in mind that SATFC was tuned and engineered with a runtime of about one minute in mind. So components might not interact in the most efficient way if cutoff times vastly different than a minute are enforced, especially much shorter ones. Moreover, cutoff times below a second are very hard to respect.

- `-SEED` - the seed to use for any (non-CLASP) randomization done in SATFC .
- `-CLASP-LIBRARY` - a path to the compiled "`.so`" CLASP library to use.
- `--log-level` - SATFC 's logging level. Can be one of `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE` (listed in increasing order of verbosity).

3.3 As a Library

The most efficient way of using SATFC is as a Java library. The source code should be packaged with SATFC 's release. The code should be well-documented, and the simplest entry point is the `SATFCFacade` object, and its corresponding builder `SATFCFacadeBuilder`. It might be helpful to Sections 3.2 and 3.2.1 to understand what the components at play are.

4 ACKNOWLEDGEMENTS

Steve Ramage deserves a special mention as he is indirectly responsible for SATFC 's quality. He is the author of AEATK, one of the main libraries in SATFC that was used throughout its development for various prototypes, as well as different other miscellaneous features. He also was a constant source of technical help and knowledge when it came to the software design of SATFC .

Various members of the AUCTIONOMICS team. Ilya Segal for the main idea behind the pre-solver used in SATFC . Ulrich Gall and his team members for working with SATFC during its early stages.