RosterBot

Elliott "Seylerius" Seyler

December 11, 2016

Contents

RosterBot is a tool for automatically generating duty rosters for teams. Operating on Constraint Programming, the software narrows the pool of schedules based on the specified restrictions, then iterates through them to find an optimal selection.

Built in Clojure, RosterBot is released under the GPL3. The following libraries are used by RosterBot:

aengelberg/loco Constraint Programming

yogthos/clj-pdf PDF generation

SparkFund/google-apps-clj Google Calendar sync

clj-time Date/time library

1 Workspace

```
(clj-workspace (concat "~/src/freelance/marco-peters/rosterbot"
                       "/src/com/seriouslyseylerius/rosterbot.clj")
               "~/src/freelance/marco-peters/rosterbot/project.clj")
```

2 TODO Core Logic

2.1 TODO Base Constraint Model

Constraint programming operates on a model of bounded variables which are reduced and restricted until a solution is found that meets all constraints. Constraints are defined in terms of variables.

2.1.1 DONE Model timespan with each day's shifts

The most flexible option seems to be generating an `employee-day` matrix, by iterating through the date range, then iterating through the employee list. I'll map each assignment-state to a number:

Assigned elsewhere	`-2`
Requested time off	`-1`
Unassigned	`0`
Shifts	`1-N`

2.1.2 DONE Model shift recurrence

Shift restrictions will be added as constraints as the `employee-day` variables are generated. Shifts will have schedule options as follows:

- Every `X` days

- Specified days-of-week every `X` weeks

Shift exceptions can be specified as dates and date ranges, causing the modified shift to not be generated for those dates.

2.1.3 DONE Model shift restrictions

I redesigned the shift-constraint model to generalize shift constraints from a single function. All arguments are keyword-args, as follows:

`:shifts` The collection of defined shifts

`:shift` The shift being restricted, may be a sequence or number

- `count` An optional count, included as the second item in `:shift` as a sequence, defaults to `1`

`:follower` Optional shift-set to require after the shift. Can be `:out` (off, PTO, or outside assignment), `:other` (anything but the specified shift), `:off` (unscheduled or PTO), or a specified list of shifts; may be wrapped in a sequence

- `count` An optional count, included as the second item in `:follower` as a sequence, defaults to `1`

These shift-constraints are generated as finite state automata and combined. The resulting automaton is used as the row constraint for each employee.

I can optionally add the ability to set employee shift preferences, but that's a feature to consider later.

2.1.4 TODO Constrain employees by other obligations and personal requests

2.1.5 TODO Model balanced shift distribution

3 TODO Data I/O

3.1 TODO Store Definitions Locally

3.2 TODO Sync Employee Availability

3.3 TODO Sync Produced Roster

4 TODO Graphical User Interface

4.1 TODO Shift Definition Screen

4.2 TODO Employee Definition Screen

4.2.1 TODO Employee Availability Screen

4.3 TODO Generated Roster View

4.3.1 TODO Enable Roster Modification