CENTER FOR HEALTHCARE ENGINEERING & PATIENT SAFETY
UNIVERSITY OF MICHIGAN

2015 Healthcare Engineering and Patient Safety Symposium

Scheduling Medical Residents With Conflicting Requests for Time-Off

Brian Lemay, Amy Cohn, Marina Epelman

Introduction

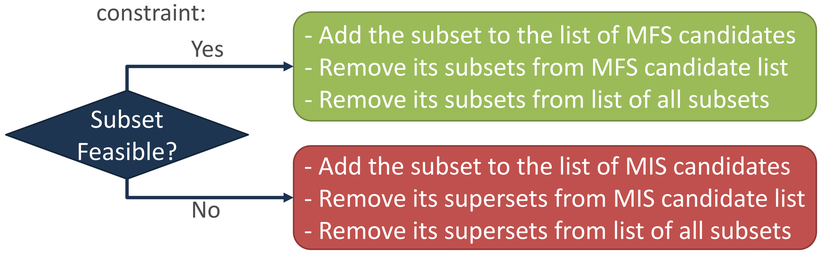
- In scheduling medical residents, many *hard constraints* must be satisfied to ensure appropriate patient coverage, adequate training opportunities, etc.
- A common scheduling objective is to maximize resident satisfaction, which can be measured by counting how many *soft constraints* (e.g., *requests* for time-off) are satisfied
- However, not all requests have equal importance and determining appropriate weights for each request is challenging
- GOAL: Identify the exhaustive collection of *maximally feasible* and *minimally infeasible* sets of requests which can in turn be used by the scheduler to select their preferred solution
- We develop an algorithm, which we call **Request Selection Via Pruning (RSVP),** to identify these sets by solving a sequence of feasibility problems

Definitions

- **Hard constraints** – constraints that must be satisfied. Example: A limit on the total number of hours worked per week.
- **Soft constraints** – constraints that are preferred, but not required to be satisfied (i.e., *requests*). Example: A request for time-off.
- **Maximally Feasible Set (MFS)** – A set of requests is maximally feasible if it is possible to satisfy all requests in the set and it is not possible to satisfy any additional requests.
- **Minimally Infeasible Set (MIS)** – A set of requests is minimally infeasible if it is not possible to satisfy all requests in the set, but it is possible to satisfy every subset of requests (i.e., if any single request is removed from the set, all remaining requests in the set can be satisfied).

Method Overview

- In the broadest sense, given a set of requests, we evaluate the set of all possible subsets
- For each subset, actions are taken based on the feasibility of granting all requests in the subset while also satisfying every hard constraint:

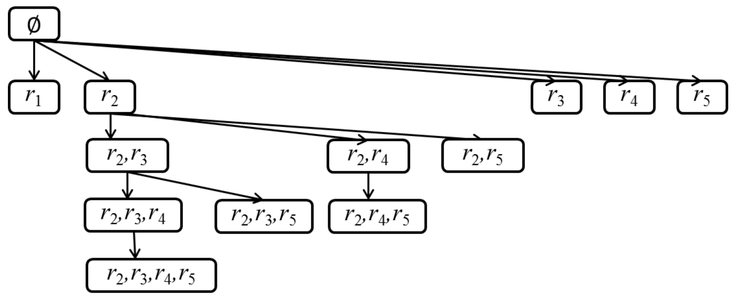


An Example

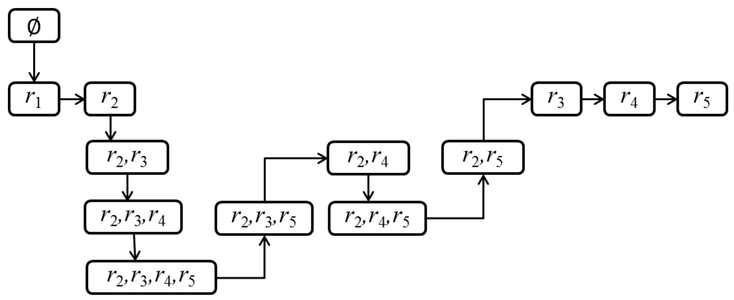
- Consider an example that involves the following time-off requests:
  - r_1: Gene wants to attend his sister's wedding
  - r_2: Billy want to go to a Michigan football game
  - r_3: Marina wants to attend a Red Sox playoff game
  - r_4: Brian wants to watch his wife's Ironman race
  - r_5: Amy wants to run the Chicago Marathon
- Simply maximizing the number of granted requests could result in granting , , and
- It may be more useful to know all MFSs: {}, {, }, {, , }, and {, , }
- Knowing all MISs can also be useful since they indicate incompatible sets of requests: {, }, {, }, {, }, {, }, {, , }, and {, , }

Algorithm Details

- Subsets are evaluated in lexicographical order
- When an infeasible subset is identified, its supersets are never added to the evaluation queue (they are known to be infeasible)
- When a feasible subset is identified, its *children* are added to the evaluation queue. In the following figure, arrows point from *parents* to their *children* (only r_2 is expanded):



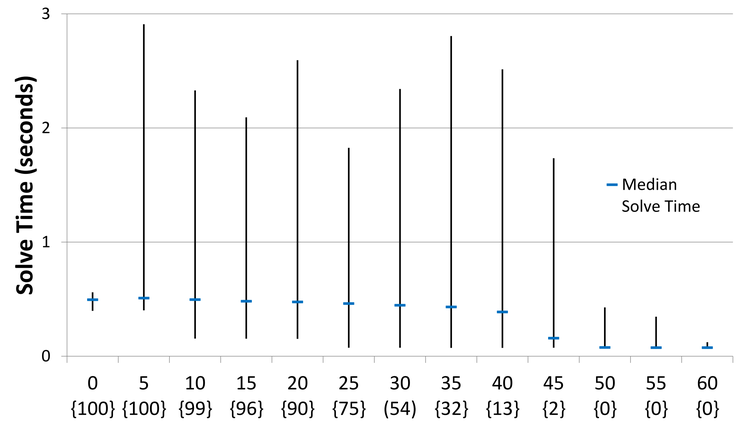
- Given these subsets, we evaluate them in the following order:



Computational Results

- Since it may be necessary to solve many feasibility problems, it is important to be able to solve each problem quickly
- GOAL: Determine how long it takes to solve a variety of realistically-sized feasibility problems
- Method: Time-off requests for 20 residents were randomly selected from a pool of 60 requests. The number of requests selected was varied from 0 to 60 in increments of 5. For each number of requests selected, 100 random problem instances were solved. The minimum, maximum, and median solve times are reported below:

Minimum, Maximum, and Median Solve Times



Number of Random Requests (out of 60)
{Number of Feasible Instances (out of 100)}

- Conclusions:
  - The majority of instances solve in less than 0.5 seconds
  - Infeasible problem instances generally take less time to solve
- Next Steps:
  - Conduct computational testing on the RSVP algorithm
  - Identify opportunities for performance improvements

Acknowledgements

This research is supported by the Center for Healthcare Engineering and Patient Safety (CHEPS), the Seth Bonder Foundation, and the UM College of Engineering SURE Program. We are also appreciative to all of the CHEPS students who helped design and write the computer code.