CVPR 2019

Reviewer paper assignments

Reviewer paper assignments: Overview

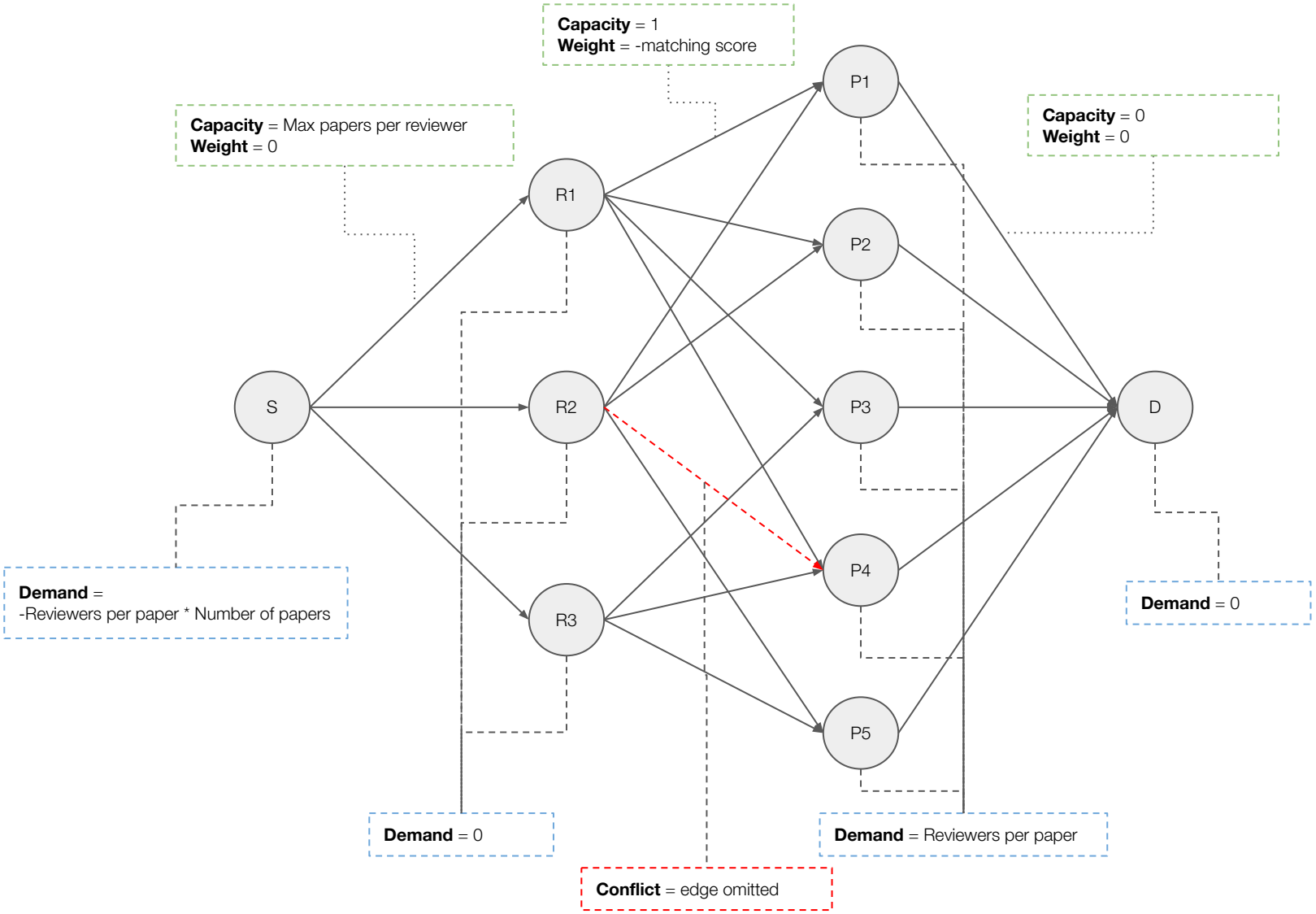
Problem

- Assign papers to appropriate reviewers under the following constraints
  - Objective: Maximize sum of assignment scores
  - Hard constraint: Limit of N_i papers for each reviewer
  - Hard constraint: Cannot be assigned to a reviewer that is conflicted.

Solution

- Information extraction (from CMT)
- Assignment Algorithm
  - Preprocessing
  - Min cost flow assignment

Reviewer paper assignments: Algorithm



- Solved by linear programming

Reviewer paper assignments: Example

Inputs

- Scores matrix: N x M (reviewers x papers)
- Conflicts matrix: N x M
- Capacities matrix: N x 1

Scores

0.44594859	0.24831998	0.20821929	0.96067386	0.59818093
0.99525685	0.87566416	0.92307729	0.50964754	0.62490232
0.04263246	0.59708164	0.92083302	0.10097003	0.25485641

Conflicts

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0

Capacities

5
3
3

Solution

Assignments

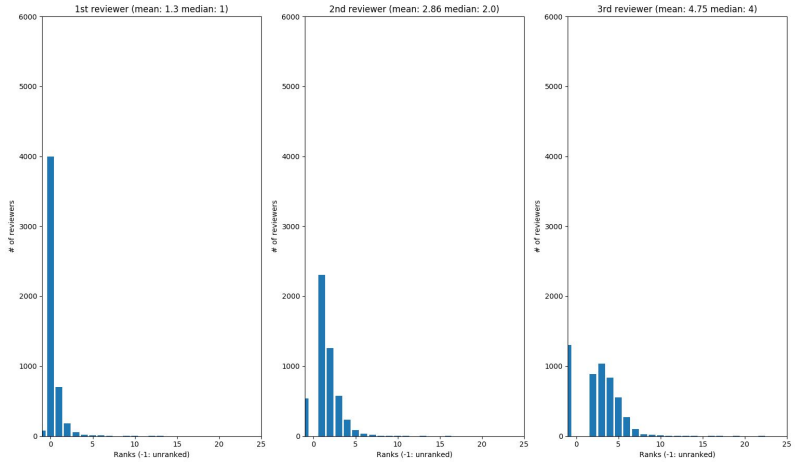
1	0	1	1	1
1	1	1	0	0
0	1	0	1	1

Reviewer paper assignments: Greedy approach

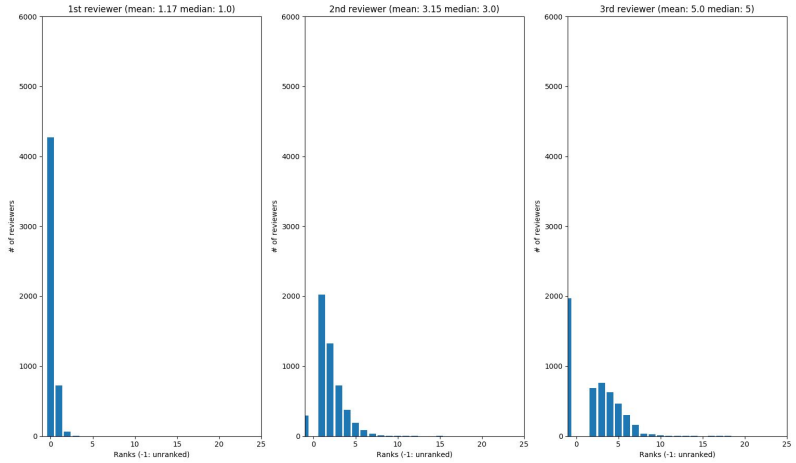
- Not a globally optimal solution!
  - 3 iterative steps:
    - Set number of reviews per paper: 1
    - Run assignment algorithm
    - Adjust reviewer capacities (for those who were assigned papers)
    - Set conflicts for reviewers with assigned paper (to ensure they cannot be assigned this paper again)
    - Repeat

- The greedy approach has more desirable statistics
  - Every paper is assigned a suggested reviewer

Globally optimal



Greedy



Reviewer paper assignments: Additional Notes

- Conflicts
  a. Common organizations
     i. Extracted from CMT
  b. Co-authors
     i. Co-authors of one paper cannot review each others papers (different papers)

- Emergency Reviewers
  a. Capacity decreased by 1 (since they may have to review additional papers)

Running code

Command line:

```
python -u calculate_scoring_matrix.py \
    -u ./$inputs_folder/Users.txt \
    -r ./$inputs_folder/reviewers.csv \
    -t ./$inputs_folder/ReviewerTpmsScores_CVPR2019.csv \
    -p ./$inputs_folder/Papers.csv \
    -q ./$inputs_folder/quotas.csv \
    -s ./$inputs_folder/ReviewerSuggestions.txt \
    -c ./$inputs_folder/ReviewerConflicts.txt \
    -n 3 \
    -g $g \
    -w_t $t \
    -w_a $a \
    -w_s $s \
    -w_e $e \
    -o $c \
    --cached_folder ./output-w_t-$t-w_a-$a-w_s-$s-w_e-$e-g-$g-n-3-config-$c/
```

Bash script (specific arguments in the script):

```
bash o-run.sh
```

Outputs

Under ***output-w_t-$t-w_a-$a-w_s-$s-w_e-$e-g-$g-n-3-config-$c*** folder:

Cached files:

- Cache inputs in a format used by our algorithm
- paper_conflicts.json; papers.json; reviewer_quotas.json; reviewers.json; reviewer_suggestions.json; users.json
- capacities.npy; conflicts.npy

Mapping files:

- Mapping paper and reviewer ids to integer (and vice versa). Used to lookup entries into the scoring and assignment matrices
- paper-mapping.json; reverse-paper-mapping.json; reverse-reviewer-mapping.json; reviewer-mapping.json

Debugging files:

- Files used for debugging purposes
- assignments-R0.npy; assignments-R1.npy; assignments-R2.npy
- v-capacities-step-0.json; v-capacities-step-1.json; v-capacities-step-2.json

Results files:

- Scoring matrices and final assignments
- Assignments.npy; assignments.xml; experience_scores.npy; subject_area_scores.npy; suggestion_scores.npy; tpms_scores.npy;

Statistics:

- Charts on assignment statistics
- O-assignments-of-suggested-reviewers-detailed.png; o-assignments-of-suggested-reviewers.png; o-capacity-limits-of-reviewers.png; o-max-scores-per-paper.png; o-mean-scores-non-suggested-reviewers-detailed.png; o-mean-scores-per-paper.png; o-reviewer-distribution.png

OpenReview files:

- Files for openreview algorithm (produces identical results as our fully optimal solution)
- o-conflicts.csv; o-matching-scores.csv; o-max-reviewers-per-paper.csv; o-papers.csv; o-reviewers.csv; o-reviewers-per-paper.csv

Input Files - Unmodified

- **Users.txt:**
  a. **How it's used:** To identify domain conflicts (in addition to what CMT identifies). Allows us to identify same institutions with different domains (uiuc.edu = illinois.edu; facebook.com = fb.com, etc.)
  b. **Extraction:** Users -> Conference User -> Actions -> Export

- **ReviewerConflicts.txt:**
  a. **How it's used:** Conflicts identified by CMT
  b. **Extraction:** Submissions -> Actions -> Export to Tab Delimited -> Reviewer Conflicts

- **ReviewerSuggestions.txt:**
  a. **How it's used:** Used in calculating scoring matrix (reviewer/paper scoring matrix).
     i. If reviewer is not suggested: Score = 0.0
     ii. If rank <= 7: Score = (8.0 - rank of reviewer for paper / 8.0)
     iii. Otherwise: Score = 0.1
  b. **Extraction:** Submissions -> Actions -> Export to Tab Delimited -> Reviewer Suggestions

- **ReviewerTpmsScores_CVPR2019.csv:**
  a. **How it's used:** Part of the scoring matrix along with the reviewer suggestion score
  b. **Extraction:** Submissions -> Actions -> TPMS -> Download Scores

Input Files - Modified

Slightly modified - Deleted first 3 rows and saved it as a csv file

- **Papers.csv:**
  a. **How it's used:** To identify co-author conflicts and assign reviewers to papers that are "Awaiting Decision", i.e., not desk rejected
  b. **Extraction:** Submissions -> Actions -> Export to Excel -> Submissions

Manually extracted - Copied and pasted the data from the from browser directly

- **quotas.csv:**
  a. **How it's used:** Limits the number of papers assigned to the author based on the quota
  b. **Extraction:** Submissions -> Actions -> Automatic Assignment -> Reviewer -> Next

- **reviewers.csv:**
  a. **How it's used:** Primarily used for subject area based scoring
    i. Primary subject area of reviewer(psar) == primary subject area of paper(psap): Score = 0.6
    ii. Secondary subject area of reviewer(ssar) == primary subject area of paper(psap): Score = 0.4
    iii. Score = 0.4 * len(ssap ∩ ssar) / len(ssap)
  b. **Extraction:** Users -> Reviewers -> [Click All]

Command line arguments

-u, -r, -t, -p, -q, -s, -c: input files discussed in the previous slides
-n: number of reviewers per paper ***(int)***
-g: number of greedy steps for assignment problem ***(int)***
- ***0***: fully optimal solution
- ***1***: first step is greedy and second step optimally assigns 2 more reviews per paper
- ***2***: assignment is performed in 3 different steps, each step is greedy (CVPR solution)

-w_t: weighting for TPMS scores ***(float)***
-w_a: weighting for subject area scores ***(float)***
-w_s: weighting for reviewer suggestion scores ***(float)***
-w_e: weighting for experienced reviewer scores ***(float)***
-o: configuration key for different quotas for different user types ***(str)***

Derek:

```
'Faculty/Researcher, >10 times as reviewer for CVPR, ICCV, or ECCV': 10,
'Faculty/Researcher, 3-10 times as reviewer for CVPR, ICCV, or ECCV': 10,
'Faculty/Researcher, 0-2 times as reviewer for CVPR, ICCV, or ECCV': 6,
'Student, >3 times as reviewer for CVPR, ICCV, or ECCV': 6,
'Student, 0-2 times as reviewer for CVPR, ICCV, or ECCV': 4,
'': 4
```

Abhinav:

```
'Faculty/Researcher, >10 times as reviewer for CVPR, ICCV, or ECCV': 9,
'Faculty/Researcher, 3-10 times as reviewer for CVPR, ICCV, or ECCV': 9,
'Faculty/Researcher, 0-2 times as reviewer for CVPR, ICCV, or ECCV': 7,
'Student, >3 times as reviewer for CVPR, ICCV, or ECCV': 7,
'Student, 0-2 times as reviewer for CVPR, ICCV, or ECCV': 4,
'': 4
```

--cached_folder: output location of results and intermediate files (used for faster future runs) ***(str)***

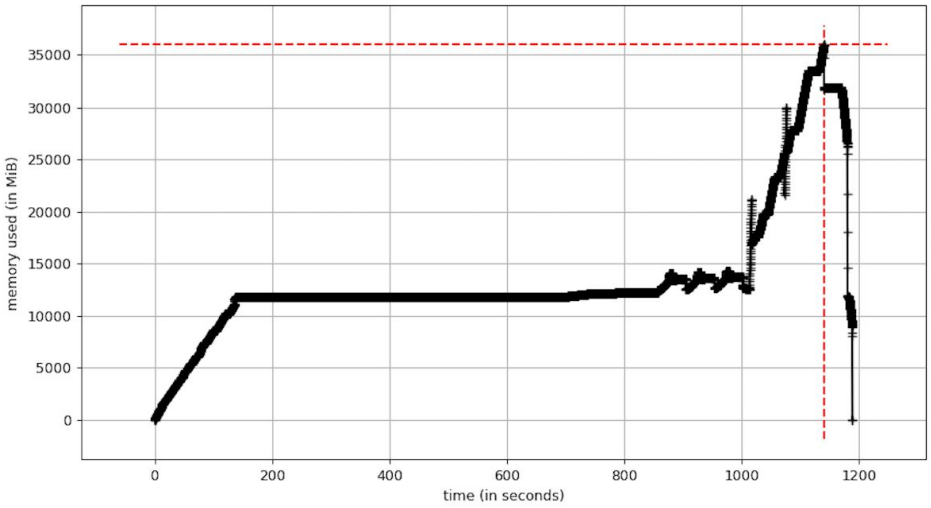
Code Profiling

Tested on Ubuntu 16.04 - 12 CPU Cores with 64G RAM

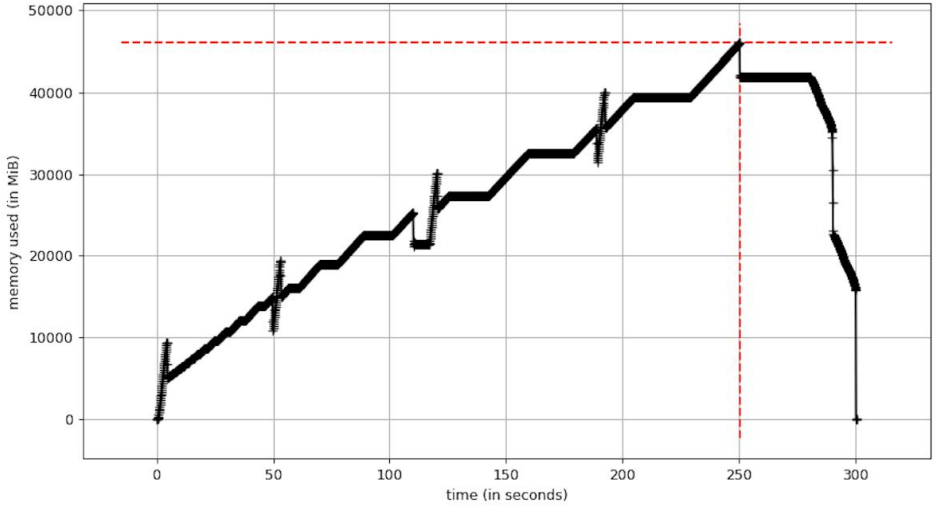
Time:
Cached run: 289.56s
Complete run: 1178.83s

Memory consumption
Cached run usage: 46G
Complete run: 36G

Complete Run



Cached Run



ICCV 2019 Updates

Changes

Revisions made by Daniel McKee

1. Code was modified to no longer need quotas.csv file since the browser page we extracted reviewers.csv from contained reviewer quotas.
2. We did not use experience scores as a factor in matching. We set all experience level scores to 1.0 and set the experience level weight to 0.0 in the code.
3. We changed the weighting of tpms score to 0.8 and subject area metric to 0.2, but we left area chair suggestions with weight 10.0.
4. We changed reviewer suggestion scores to allow for scaled scores of up to 20 reviewer suggestions according to following:
    a. If reviewer is not suggested: Score = 0.0
    b. If rank <= 19: Score = (20.0 - rank of reviewer for paper / 20.0)
    c. Otherwise: Score = 0.05
5. We changed default quota configuration options (used “low” option in the end):

high:
'Researcher/faculty': 10,
'Student': 6,
'': 6

low:
'Researcher/faculty': 9,
'Student': 5,
'': 5