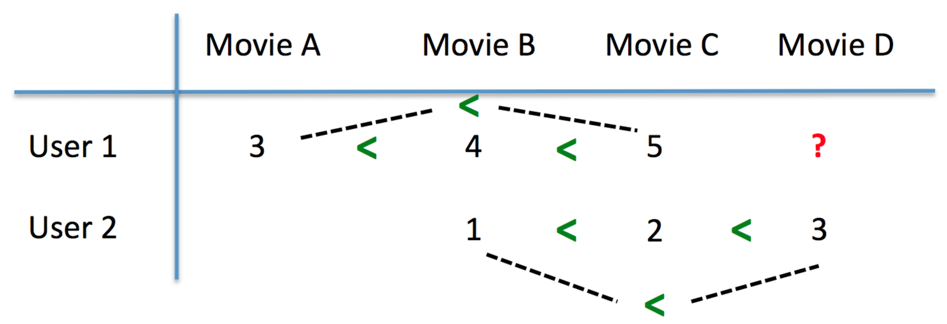
LARGE-SCALE COLLABORATIVE RANKING IN NEAR-LINEAR TIME

LIWEI WU, CHO-JUI HSIEH, AND JAMES SHARPNACK
{ LIWU, CHOHSIEH AND JSHARPNA}@UCDAVIS.EDU

PROBLEM

- Collaborative Ranking can be applied to classical recommender system
  - Given d_1 users, d_2 movies
  - Each user has a subset of observed movie comparisons
  - Goal: predict movie rankings for each user
- For classical data (e.g., Netflix), we can transform original ratings to pairwise comparisons
- With the same data size, **ranking loss outperforms point-wise loss**



MODEL

- Collaborative Ranking:

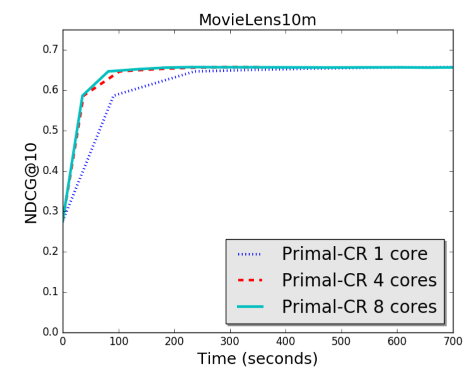
$$\min_{U,V} \sum_{(i,j,k)\in\Omega} \ell\Big(Y_{i,j,k}\cdot\left[(UV^T)_{ij}-(UV^T)_{ik}\right]\Big)+\lambda(\|U\|_F^2+\|V\|_F^2)$$

- The loss function ℓ we used is $\mathcal{L}_2$ hinge loss $\ell(a)=\max(0,1-a)^2$
- The set $(i,j,k)\in\Omega$:
  - User i rates item $j >$ item $k \Leftrightarrow Y_{i,j,k}=1$
  - If user i rates $\bar{d}_2$ movies, there will be $O(\bar{d_2}^2)$ pairs per user.
- Time Complexity Comparison
  - Classical matrix factorization: $O(d_1\bar{d}_2 r)$ time, $O(d_1\bar{d}_2)$ memory
  - Previous collaborative ranking: $O(d_1\bar{d_2}^2 r)$ time, $O(d_1\bar{d_2}^2)$ memory
  - Our **Primal-CR**: $O(d_1\bar{d_2}^2+d_1\bar{d}_2 r)$ time, $O(d_1\bar{d}_2)$ memory
  - Our **Primal-CR++**: $O(d_1\bar{d}_2(r+\log(\bar{d}_2)))$ time, $O(d_1\bar{d}_2)$ memory

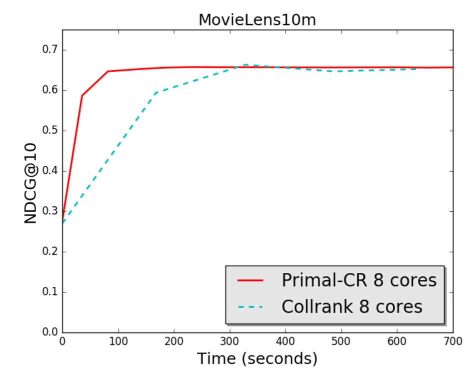
  where d_1 is number of users, $\bar{d}_2$ is averaged ratings per user, r is the target rank.

PARALLELIZATION

- Our algorithm scales up well (see comparisons for 1 core, 4 cores and 8 cores) in the multi-core shared memory setting
- Primal-CR still much faster than Collrank (Park et al., ICML 2015) when 8 cores are used
- In all the plots in the poster, we were comparing our Julia codes with others' C++ codes back then (Despite Julia is slower than C++, our proposed algorithms still **won by a lot**)



MoviLens10M $(71,567\times 65,134)$

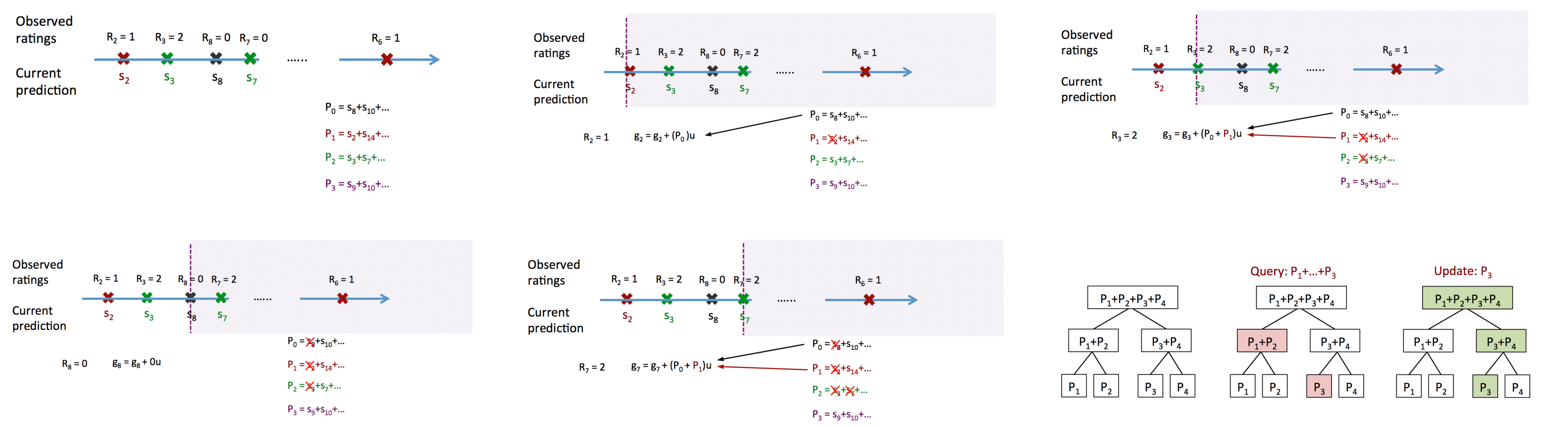


MoviLens10M

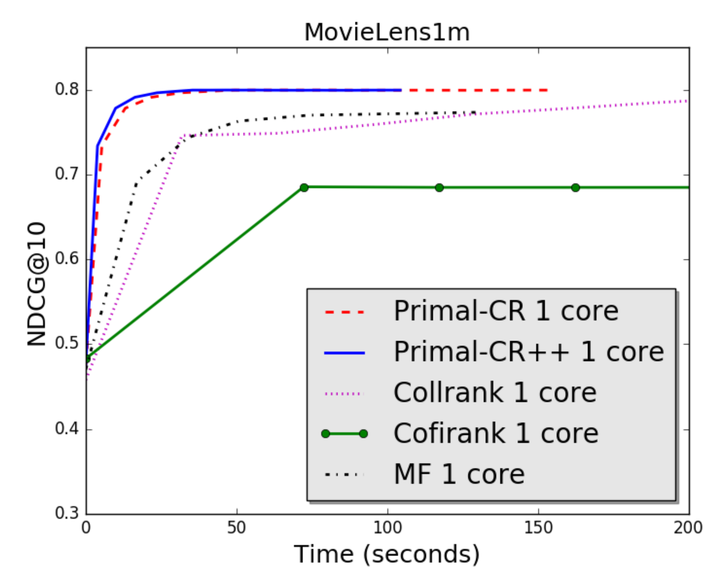
METHOD

$$\nabla_V f(V)=\sum_{i=1}^{d_1}\sum_{(j,k)\in\Omega_i} 2\max\Big(0,1-(\boldsymbol{u}_i^T\boldsymbol{v}_j-\boldsymbol{u}_i^T\boldsymbol{v}_k)\Big)(\boldsymbol{u}_i\boldsymbol{e}_k^T-\boldsymbol{u}_i\boldsymbol{e}_j^T)+\lambda V$$

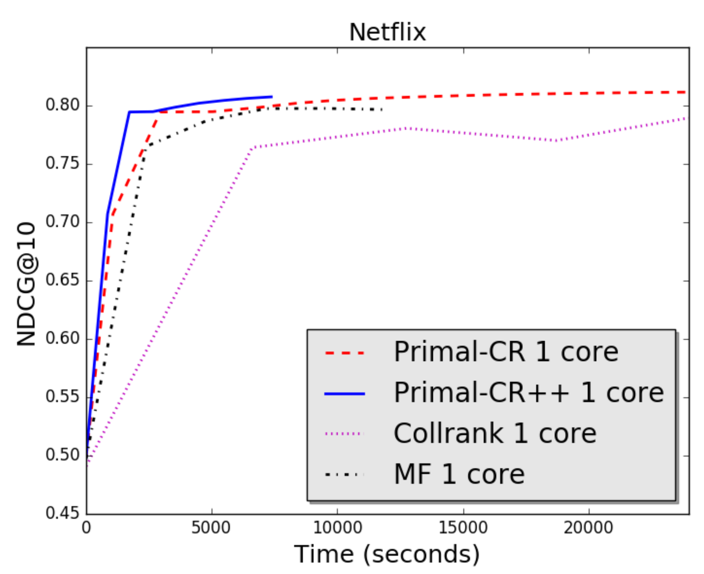
- **Primal-CR**: *Pre-compute $\boldsymbol{u}_i^T\boldsymbol{v}_j$ for all j; Initial $c_j=0$ for all j; Update c_j for each $(j,k)\in\Omega_i$; Finally compute $\nabla f(V)=\nabla f(V)+\sum_j c_j\boldsymbol{u}_i\boldsymbol{e}_j^T$*
- **Primal-CR++**: *Fix k, do a linear scan of j after sorting; Initially $P_\ell=\sum_{j:R_j=\ell}s_j$; For $j=\pi(1),\pi(2),\cdots$: Add $(\sum_{\ell<R_j}P_\ell)\cdot u$ to g_j; Update $P_{R_j}\leftarrow P_{R_j}-s_j$* (Can be done in $O(\log(T))$ time per query for T levels of ratings using **Segment tree or Fenwick tree!**)



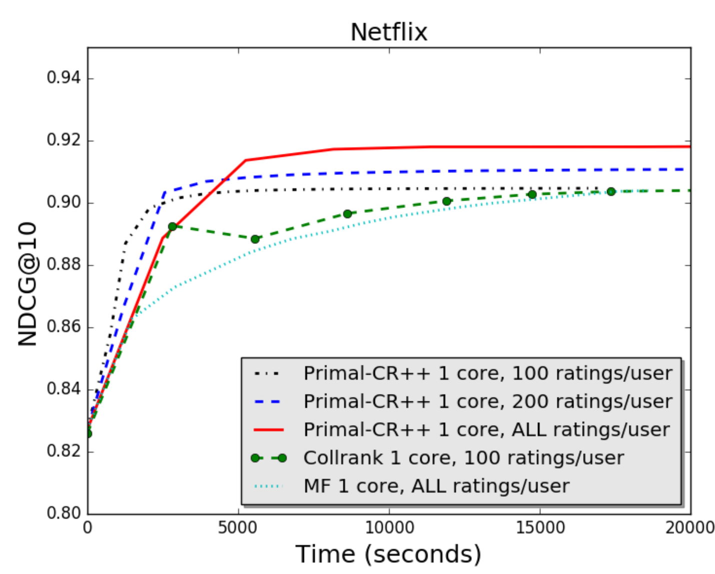
RESULTS



MoviLens1M $(6,040\times 3,952)$



Netflix $(2,649,430\times 17,771)$



Netflix $(2,649,430\times 17,771)$

- Single Core Subsampled Data (Plots Leftmost and Center):
  - We subsampled each user to have exactly 200 ratings in training set and used the rest of ratings as test set, since previous approaches cannot scale up
  - Users with fewer than $200+10$ ratings not included
- Single Core Full Data (Plot Rightmost):
  - Due to the $O(|\Omega|r)$ complexity, existing algorithms always sub-sample a limited number of pairs per user
  - Our algorithm is the **first** ranking-based algorithm that can scale to full Netflix data set using a single core, and without sub-sampling
- A natural question:
  Does using more training data help us predict and recommend better?
  The answer is yes!

CONCLUSION

- We show that CR can be used to replace matrix factorization in recommender systems
- We show that CR can be solved efficiently (same time complexity with matrix completion)
- We show that it is always best to use all available pairwise comparisons if possible (subsampling gives suboptimal recommender results for top k items)

SOURCE CODE

- Julia codes: `https://github.com/wuliwei9278/ml-1m`
- C++ codes (2x faster than Julia codes): `https://github.com/wuliwei9278/primalCR`