Team Formation

Balancing Task Coverage and Expert Workload

Karan Vombatkere Evimaria Terzi

Department of Computer Science
Boston University

Under review at CIKM '22

Roadmap

- Introduction
- Preliminaries
- The Balanced Coverage problem
- `ThresholdGreedy` algorithm and approximation guarantee
- Running time and computational speedups
- Tuning the importance of coverage and workload
- Experimental evaluation
- Conclusions

Introduction

Labor markets. Amazon mechanical turks, Guru, Freelancer, scientific collaborations, hiring committees.

Introduction

Labor markets. Amazon mechanical turks, Guru, Freelancer, scientific collaborations, hiring committees.

Experts, tasks and skills. Tasks *require* a set of skills, while experts *master* a set of skills.

Introduction

Labor markets. Amazon mechanical turks, Guru, Freelancer, scientific collaborations, hiring committees.

Experts, tasks and skills. Tasks *require* a set of skills, while experts *master* a set of skills.

Classical team-formation.

Identify a "good" team of experts such that the skills of these experts cover *all* the skills required by a given task.

Introduction - example

Experts

Anne: $\{excel, java, python\}$
Bob: $\{latex, python\}$
Charlie: $\{sql, tensorflow\}$

Tasks

T_1: $\{java, python\}$
T_2: $\{excel, python, sql\}$
T_3: $\{excel, latex, plotly, tableau\}$

Introduction - example

Experts	**Tasks**
Anne: $\{excel, java, python\}$	T_1: $\{java, python\}$
Bob: $\{latex, python\}$	T_2: $\{excel, python, sql\}$
Charlie: $\{sql, tensorflow\}$	T_3: $\{excel, latex, plotly, tableau\}$

Assignment

Assign Anne to tasks T_1 and T_2; Bob to tasks T_2 and T_3.

Introduction - example

Experts	**Tasks**
Anne: {*excel*, *java*, *python*}	T_1: {*java*, *python*}
Bob: {*latex*, *python*}	T_2: {*excel*, *python*, *sql*}
Charlie: {*sql*, *tensorflow*}	T_3: {*excel*, *latex*, *plotly*, *tableau*}

Assignment

Assign Anne to tasks T_1 and T_2; Bob to tasks T_2 and T_3.

Task coverage

T_1 is 100% covered; T_2 is 67% covered; T_3 is 25% covered.

Introduction - example

Experts	**Tasks**
Anne: $\{excel, java, python\}$	T_1: $\{java, python\}$
Bob: $\{latex, python\}$	T_2: $\{excel, python, sql\}$
Charlie: $\{sql, tensorflow\}$	T_3: $\{excel, latex, plotly, tableau\}$

Assignment

Assign Anne to tasks T_1 and T_2; Bob to tasks T_2 and T_3.

Task coverage

T_1 is 100% covered; T_2 is 67% covered; T_3 is 25% covered.

Expert workload

Anne and Bob each have a workload of 2, since they are assigned to two tasks each. Charlie has a workload of 0.

Our work

Problem. Maximize total task coverage while also trying to minimize the maximum workload among the experts. We combine these two goals into one objective function, and define the (NP-hard) Balanced Coverage problem.

Our work

Problem. Maximize total task coverage while also trying to minimize the maximum workload among the experts. We combine these two goals into one objective function, and define the (NP-hard) Balanced Coverage problem.

Algorithm. Polynomial-time algorithm, `ThresholdGreedy` with a provable approximation guarantee. We give a set of computational speedups for scalability.

Our work

Problem. Maximize total task coverage while also trying to minimize the maximum workload among the experts. We combine these two goals into one objective function, and define the (NP-hard) Balanced Coverage problem.

Algorithm. Polynomial-time algorithm, `ThresholdGreedy` with a provable approximation guarantee. We give a set of computational speedups for scalability.

Experiments. Evaluate on real datasets to demonstrate practical utility of formulation, and efficacy of algorithm.

Preliminaries

- Set of m tasks $\mathcal{J} = \{J_1, \ldots, J_m\}$, and a set of n experts $\mathcal{E} = \{E_1, \ldots, E_n\}$
- Set of skills S such that for every task J_j we have $J_j \subseteq S$, and for every expert E_i we have $E_i \subseteq S$.

Preliminaries

- Set of m tasks $\mathcal{J} = \{J_1, \ldots, J_m\}$, and a set of n experts $\mathcal{E} = \{E_1, \ldots, E_n\}$
- Set of skills S such that for every task J_j we have $J_j \subseteq S$, and for every expert E_i we have $E_i \subseteq S$.
- An *assignment* of experts to tasks is represented by a $n \times m$ binary matrix A, such that $A(i, j) = 1$ if expert E_i is assigned to task J_j; otherwise $A(i, j) = 0$.

Preliminaries

- Set of m tasks $\mathcal{J} = \{J_1, \ldots, J_m\}$, and a set of n experts $\mathcal{E} = \{E_1, \ldots, E_n\}$
- Set of skills S such that for every task J_j we have $J_j \subseteq S$, and for every expert E_i we have $E_i \subseteq S$.
- An *assignment* of experts to tasks is represented by a $n \times m$ binary matrix A, such that $A(i,j) = 1$ if expert E_i is assigned to task J_j; otherwise $A(i,j) = 0$.
- Alternatively, we can view an assignment as a bipartite graph with nodes on one side corresponding to experts and the nodes on the other side corresponding to tasks; edge (i,j) exists iff $A(i,j) = 1$.

Preliminaries

Given an assignment A, we define the *coverage* of a single task J_j as the fraction of the skills in J_j covered by the experts assigned to J_j.

$$C(J_j \mid A) = \frac{|(\cup_{i:A(i,j)=1} E_i) \cap J_j|}{|J_j|}$$

Note that $0 \leq C(J_j \mid A) \leq 1$.

Preliminaries

Given an assignment A, we define the *coverage* of a single task J_j as the fraction of the skills in J_j covered by the experts assigned to J_j.

$$C(J_j \mid A) = \frac{|(\cup_{i:A(i,j)=1} E_i) \cap J_j|}{|J_j|}$$

Note that $0 \leq C(J_j \mid A) \leq 1$.

Overall task coverage

$$C(A) = \sum_{j=1}^{m} C(J_j \mid A)$$

Preliminaries

Given an assignment A, we define the *load* of an expert E_i in A to be the number of tasks that E_i is assigned to.

$$L(E_i \mid A) = \sum_j A(i,j)$$

Preliminaries

Given an assignment A, we define the *load* of an expert E_i in A to be the number of tasks that E_i is assigned to.

$$L(E_i \mid A) = \sum_j A(i,j)$$

Maximum load

$$L_{\max}(A) = \max_i L(E_i \mid A)$$

The Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks $\mathcal{J} = \{J_1, \ldots, J_m\}$, and a set of n experts $\mathcal{E} = \{E_1, \ldots, E_n\}$ find an assignment A of experts to tasks such that

$$F(A) = C(A) - L_{\max}(A)$$

is maximized

The Balanced Coverage problem

Observations:

1. $C(A)$ is a sum of normalized coverages, and is a quantity that takes real values between $[0, m]$. $L_{\max}(A)$ takes integer values between $\{0, m\}$. We introduce a *balancing coefficient* to tune the relative importance of the two parts of $F()$.

The Balanced Coverage problem

Observations:

1. $C(A)$ is a sum of normalized coverages, and is a quantity that takes real values between $[0, m]$. $L_{\max}(A)$ takes integer values between $\{0, m\}$. We introduce a *balancing coefficient* to tune the relative importance of the two parts of $F()$.
2. The number of experts is *not* constrained in the definition of the problem, since a good solution needs to identify the ideal "balance point" between the experts used and the coverage achieved.

The Balanced Coverage problem

Monotonicity, Submodularity and NP-hardness:

1. $C(A)$ is a *monotone* and *submodular* function.
   - **Monotonicity:** For any two assignments A and A', if $A \subseteq A'$ then $C(A) \leq C(A')$.
   - **Submodularity:** For any two assignments A and A', such that $A \subseteq A'$ and for any (i,j) such that $A'(i,j) \neq 1$,

$$C(A \cup (i,j)) - C(A) \geq C(A' \cup (i,j)) - C(A')$$

The Balanced Coverage problem

Monotonicity, Submodularity and NP-hardness:

1. $C(A)$ is a *monotone* and *submodular* function.
   - **Monotonicity:** For any two assignments A and A', if $A \subseteq A'$ then $C(A) \leq C(A')$.
   - **Submodularity:** For any two assignments A and A', such that $A \subseteq A'$ and for any (i,j) such that $A'(i,j) \neq 1$,

$$C(A \cup (i,j)) - C(A) \geq C(A' \cup (i,j)) - C(A')$$

2. Balanced Coverage problem is NP-hard even for $m = 2$. Proof uses reduction from *load balancing with restricted assignment*.

The ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$

Output: An assignment of experts to tasks A

1: $A \leftarrow \emptyset, F_{\max} = 0$
2: **for** $\tau = 1, ..., m$ **do**
3: Create the set of experts $\mathcal{E}_\tau$, with τ copies of each expert
4: $A_\tau = \texttt{Greedy}(\mathcal{E}_\tau, \mathcal{J})$
5: Compute $F_\tau = C(A_\tau) - \tau$
6: **if** $F_\tau \geq F_{max}$ **then**
7: $F_{max} = F_\tau$
8: $A \leftarrow A_\tau$
9: **end if**
10: **end for**
11: **return** A

The ThresholdGreedy algorithm

Lemma 2

Let A_τ be the assignment of experts to tasks returned by `Greedy` *(Line 4) for fixed threshold workload τ. Let OPT_τ be the optimal assignment of experts $\mathcal{E}_\tau$ to tasks $\mathcal{J}$ with respect to the coverage objective $C(OPT_\tau)$. Then, it holds that:*

$$C\left(A_\tau\right) \geq \left(1 - \frac{1}{e}\right) C\left(OPT_\tau\right).$$

The ThresholdGreedy algorithm

Lemma 2

Let A_τ be the assignment of experts to tasks returned by `Greedy` *(Line 4) for fixed threshold workload τ. Let OPT_τ be the optimal assignment of experts $\mathcal{E}_\tau$ to tasks $\mathcal{J}$ with respect to the coverage objective $C(OPT_\tau)$. Then, it holds that:*

$$C(A_\tau) \geq \left(1 - \frac{1}{e}\right) C(OPT_\tau).$$

Proof.

The proof of this lemma follows the standard proof of the `Greedy` algorithm being an $\left(1 - \frac{1}{e}\right)$-approximation algorithm to the coverage problem. □

Approximation guarantee

The overall objective function $F()$ does not have a concrete form, i.e. it is not linear or convex. Our approximation guarantee is a weaker form of the standard multiplicative approximation guarantees since $F()$ is not guaranteed to be positive.

Approximation guarantee

The overall objective function $F()$ does not have a concrete form, i.e. it is not linear or convex. Our approximation guarantee is a weaker form of the standard multiplicative approximation guarantees since $F()$ is not guaranteed to be positive.

Theorem 3

Let A be the assignment returned by `ThresholdGreedy` and let OPT be the optimal assignment for the Balanced Coverage *problem. Then we have the following approximation:*

$$C(A) - L_{\max}(A) \geq \left(1 - \frac{1}{e}\right) C(OPT) - L_{\max}(OPT)$$

Approximation guarantee proof

Proof.

Let's assume that $L_{\max}(OPT) = \tau$; note that $L_{\max}(A)$ may or may not be equal to τ. Then, we have the following:

$$
\begin{aligned}
F(A) &\geq F(A_\tau) \ \text{(True for any } \tau\text{)} \\
&= C(A_\tau) - \tau \\
&\geq \left(1 - \frac{1}{e}\right) C(OPT_\tau) - \tau \ \text{(Lemma 2)} \\
&\geq \left(1 - \frac{1}{e}\right) C(OPT) - \tau \ \text{(OPT_τ is optimal for } \tau\text{)} \\
&= \left(1 - \frac{1}{e}\right) C(OPT) - L_{\max}(OPT).
\end{aligned}
$$

Running time and speedups

A naive implementation of `ThresholdGreedy` takes time $O(m^2 n^2)$. We give two methods to significantly improve running time in practice.

Running time and speedups

A naive implementation of `ThresholdGreedy` takes time $O(m^2 n^2)$. We give two methods to significantly improve running time in practice.

1. **Lazy evaluation technique.** Leverage submodularity of $C()$ to overcome the computational bottleneck of `Greedy` using a maximum-priority queue.

Running time and speedups

A naive implementation of `ThresholdGreedy` takes time $O(m^2 n^2)$. We give two methods to significantly improve running time in practice.

1. **Lazy evaluation technique.** Leverage submodularity of $C()$ to overcome the computational bottleneck of `Greedy` using a maximum-priority queue.
2. **Early termination.** The value of the objective computed by `ThresholdGreedy` for different values of τ is a unimodal function.

Running time and speedups

Early Termination. The submodularity and monotonicity of the overeall coverage function imply the following observations:

- For every $\tau \in \{1, \ldots, m\}$: $C_\tau \geq C_{\tau-1}$.
- For every $\tau \in \{1, \ldots, m-1\}$: $C_\tau - C_{\tau-1} \geq C_{\tau+1} - C_\tau$.

Theorem 4

If there is a value of the threshold τ^*, such that $F_{\tau^*} \geq F_{\tau^*-1}$ and $F_{\tau^*} \geq F_{\tau^*+1}$, then the values of the objective function $F_\tau = F(A_\tau)$ as computed by `ThresholdGreedy` for $\tau = 1, \ldots, m$ are unimodal. That is, $F_1 \leq F_2 \leq \ldots \leq F_{\tau^*}$ and $F_{\tau^*} \geq F_{\tau^*+1} \geq \ldots \geq F_m$.

Tuning the importance of coverage and workload

We enhance our framework to tune the relative importance of coverage vs. the workload by adding a *balancing coefficient* $\lambda > 0$, to the coverage function:

$$F^{\lambda}(A) = \lambda C(A) - L_{\max}(A)$$

Tuning the importance of coverage and workload

We enhance our framework to tune the relative importance of coverage vs. the workload by adding a *balancing coefficient* $\lambda > 0$, to the coverage function:

$$F^{\lambda}(A) = \lambda C(A) - L_{\max}(A)$$

Efficient search on values of λ.

Lemma 5

Assume that $\lambda_1 > \lambda_2$ and let the best-greedy objectives achieved for those values be $F_{\tau_1}^{\lambda_1}$ and $F_{\tau_2}^{\lambda_2}$ respectively. Then, for the corresponding best-greedy workloads we have that $\tau_1 \geq \tau_2$.

Experimental evaluation

Datasets. IMDB, Bibsonomy, Freelancer, Guru

Table 1: Summary statistics of our datasets.

Dataset	Experts	Tasks	Skills	skills/ expert	skills/ task
IMDB-2015	5551	18109	26	2.4	3.1
IMDB-2018	3871	13183	26	2.1	2.7
IMDB-2020	2176	7858	25	1.9	2.4
Bibsonomy-2010	3044	21981	1000	13.7	4.8
Bibsonomy-2015	1904	9061	1000	10.9	4.3
Bibsonomy-2020	177	834	858	11.5	3.6
Freelancer	1212	993	175	1.5	2.9
Guru	6120	3195	1639	13.1	5.2

Experimental evaluation

Datasets. IMDB, Bibsonomy, Freelancer, Guru

Table 1: Summary statistics of our datasets.

Dataset	Experts	Tasks	Skills	skills/ expert	skills/ task
IMDB-2015	5551	18109	26	2.4	3.1
IMDB-2018	3871	13183	26	2.1	2.7
IMDB-2020	2176	7858	25	1.9	2.4
Bibsonomy-2010	3044	21981	1000	13.7	4.8
Bibsonomy-2015	1904	9061	1000	10.9	4.3
Bibsonomy-2020	177	834	858	11.5	3.6
Freelancer	1212	993	175	1.5	2.9
Guru	6120	3195	1639	13.1	5.2

Baselines. We used the following three intuitive baselines, inspired by related work and heuristic methods.

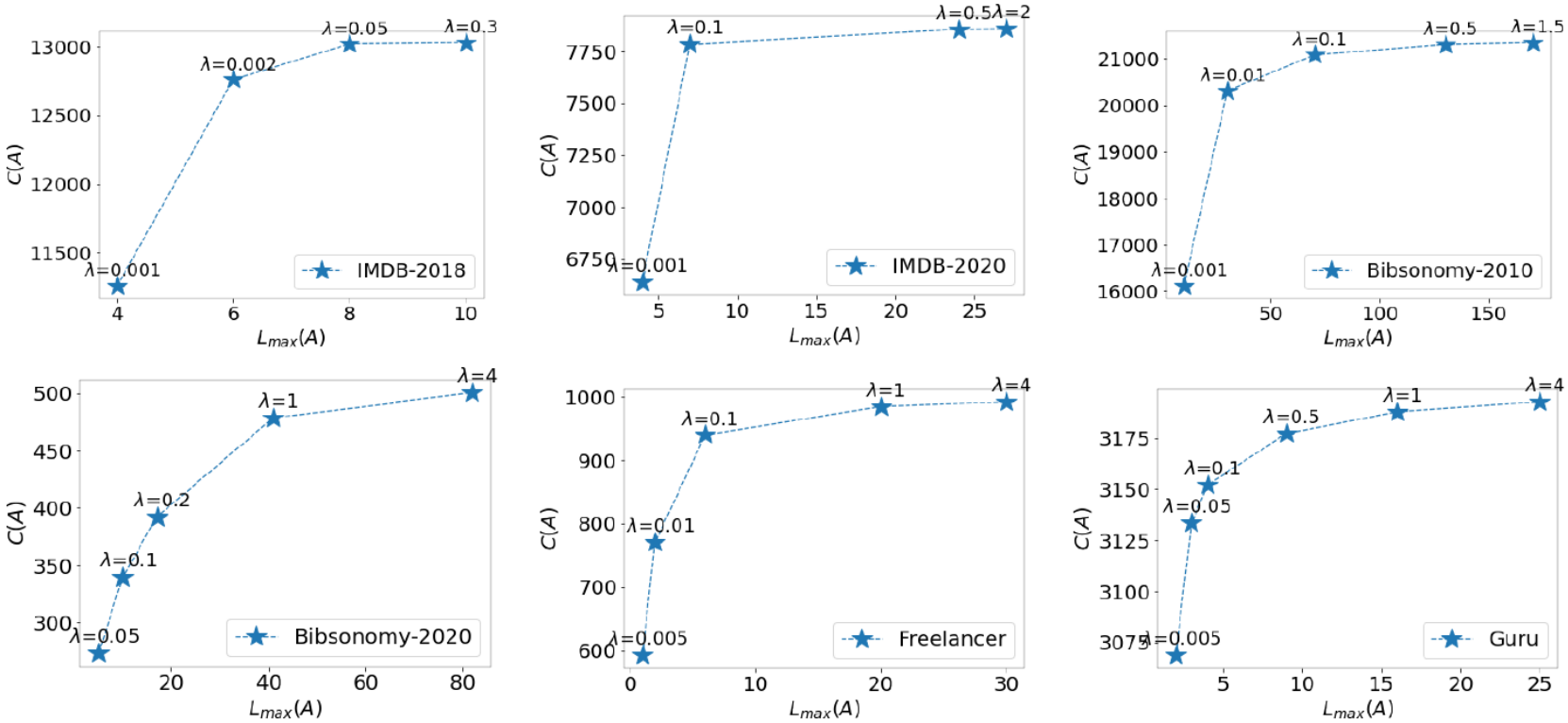
1. `TaskGreedy`
2. `NoUpdateGreedy`
3. `SmartRandom`

Experimental evaluation

Elbow method to choose λ. Graphically the best λ value for each dataset corresponds to the λ value observed at the "elbow" of the plot, where further increase of λ does not give significant increase in coverage.

Experimental evaluation

Elbow method to choose λ. Graphically the best λ value for each dataset corresponds to the λ value observed at the "elbow" of the plot, where further increase of λ does not give significant increase in coverage.



Experimental results

`ThresholdGreedy` **outperforms the other algorithms both in terms of the objective (consistently higher) and in terms of the workload (predominantly lower).**

Dataset	`ThresholdGreedy`		`TaskGreedy`		`NoUpdateGreedy`		`SmartRandom`	
	$F^{\lambda}(A)$	$L_{\max}(A)$	$F^{\lambda}(A)$	$L_{\max}(A)$	$F^{\lambda}(A)$	$L_{\max}(A)$	$F^{\lambda}(A)$	$L_{\max}(A)$
IMDB-2015 ($\lambda = 0.05$)	885	7	475	362	720	150	456	81
IMDB-2018 ($\lambda = 0.05$)	643	8	339	264	448	200	186	345
IMDB-2020 ($\lambda = 0.1$)	771	7	644	118	650	100	445	160
Bibsonomy-2010 ($\lambda = 0.1$)	2039	70	1319	243	1097	200	711	272
Bibsonomy-2015 ($\lambda = 0.05$)	389	27	96	126	129	250	92	79
Bibsonomy-2020 ($\lambda = 1$)	438	41	402	93	408	94	359	78
Freelancer ($\lambda = 0.1$)	88	6	63	36	25	50	17	33
Guru ($\lambda = 0.1$)	311	4	225	30	152	119	82	60

Experimental results

While slower than the baselines, `ThresholdGreedy` has a reasonable running time in practice, even on the largest datasets which have several thousand tasks and experts.

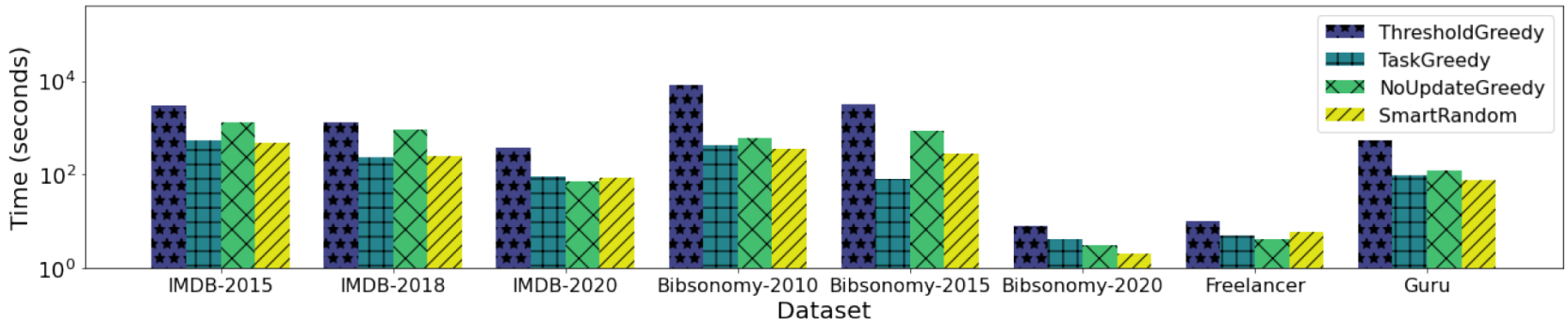


Figure 2: Running times (in seconds) of `ThresholdGreedy` and baselines, in logarithmic scale.

Conclusions

We introduced the Balanced Coverage problem, a new problem in team formation.

Conclusions

We introduced the Balanced Coverage problem, a new problem in team formation.

We showed that Balanced Coverage is an NP-hard problem and then we designed a polynomial-time approximation algorithm for solving it.

Conclusions

We introduced the Balanced Coverage problem, a new problem in team formation.

We showed that Balanced Coverage is an NP-hard problem and then we designed a polynomial-time approximation algorithm for solving it.

We also showed we can exploit the structure of our objective function and design speedups that work extremely well in practice.

Conclusions

We introduced the Balanced Coverage problem, a new problem in team formation.

We showed that Balanced Coverage is an NP-hard problem and then we designed a polynomial-time approximation algorithm for solving it.

We also showed we can exploit the structure of our objective function and design speedups that work extremely well in practice.

Our experimental results with a variety of datasets from various domains demonstrated the utility of our framework and the efficiency and efficacy of our algorithm.

Thank you!