Balancing Task Coverage and Expert Workload in Team Formation

Karan Vombatkere   Evimaria Terzi

Department of Computer Science
Boston University

SDM '23

Introduction

Online and offline labor markets.

Introduction

Online and offline labor markets.

Introduction

Online and offline labor markets.

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

Introduction - example

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Introduction - example

Alice

excel, **java**, **python**

Bob

latex, *python*

Charlie

sql, *tensorflow*

David

excel, **latex**, **tableau**

Task 1

java, python, *sql*, **latex, tableau**

Possible Assignment 1.

Alice and David cover 80% of Task 1

Introduction - example

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, *tensorflow*

David

excel, latex, tableau

Task 1

java, python, sql, latex, *tableau*

Possible Assignment 2.

Bob and Charlie cover 80% of Task 1

Introduction - example

Alice

excel, **java**, **python**

Bob

latex, *python*

Charlie

java, **sql**, *tensorflow*

David

excel, **latex**, **tableau**

Task 1

java, python, sql, latex, tableau

Possible Assignment 3.

Alice, Charlie and David cover 100% of Task 1

Introduction - example

Alice

excel, *java*, *python*

Bob

latex, **python**

Charlie

java, **sql**, *tensorflow*

David

excel, *latex*, **tableau**

Task 1

java, **python**, **sql**, **latex**, **tableau**

Possible Assignment 4.

Bob, Charlie and David cover 100% of Task 1

What if we add more tasks...

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Task 2

excel, tensorflow, tableau

Task 3

java, sql, latex

What if we add more tasks...

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Task 2

excel, tensorflow, tableau

Task 3

java, sql, latex

Possible Assignment 1 - 100% Coverage

Task 1: Bob, Charlie, David
Task 2: Charlie, David
Task 3: Bob, Charlie

Workload:
Alice: 0, Bob: 2, Charlie: 3, David: 2

What if we add more tasks...

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Task 2

excel, tensorflow, tableau

Task 3

java, sql, latex

Possible Assignment 2 - 100% Coverage

Task 1: Alice, Charlie, David
Task 2: Charlie, David
Task 3: Charlie, David

Workload:
Alice: 1, Bob: 0, Charlie: 3, David: 3

What if we add more tasks...

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Task 2

excel, tensorflow, tableau

Task 3

java, sql, latex

Possible Assignment 3 - $<$ 100% Coverage

Task 1 (80%): Alice, David
Task 2 (67%): David
Task 3 (100%): Bob, Charlie

Workload:
Alice: 1, Bob: 1, Charlie: 1, David: 2

What if we add more tasks...

Alice

excel, java, python

Bob

latex, python

Charlie

java, sql, tensorflow

David

excel, latex, tableau

Task 1

java, python, sql, latex, tableau

Task 2

excel, tensorflow, tableau

Task 3

java, sql, latex

Possible Assignment 4 - $<$ 100% Coverage

Task 1 (60%): Alice, Bob
Task 2 (67%): David
Task 3 (67%): Charlie

Workload:
Alice: 1, Bob: 1, Charlie: 1, David: 1

Roadmap

- Preliminaries
- The BALANCED COVERAGE problem
- Related work and approximation guarantee
- The `ThresholdGreedy` algorithm
- Experiments

Preliminaries

An *assignment* of n experts to m 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

An *assignment* of n experts to m 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 $A(i,j) = 0$.

Total task coverage

$C(J_j \mid A)$ denotes the *coverage* of a *single* task J_j: it is the fraction of skills in J_j covered by the experts assigned to J_j.

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

Preliminaries

An *assignment* of n experts to m 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 $A(i,j) = 0$.

Total task coverage

$C(J_j \mid A)$ denotes the *coverage* of a *single* task J_j: it is the fraction of skills in J_j covered by the experts assigned to J_j.

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

Maximum load

$L(E_i \mid A)$ denotes the *load* of an expert E_i: corresponds to the number of tasks that E_i is assigned to.

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

Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks and a set of n experts, find an assignment A of experts to tasks such that.

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

is maximized.

We tune the relative importance of task coverage vs. expert workload by adding a balancing coefficient $\lambda > 0$, to the coverage function.

Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks and a set of n experts, find an assignment A of experts to tasks such that.

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

is maximized.

We tune the relative importance of task coverage vs. expert workload by adding a balancing coefficient $\lambda > 0$, to the coverage function.

- Balanced Coverage is NP-Hard.

Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks and a set of n experts, find an assignment A of experts to tasks such that.

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

is maximized.

We tune the relative importance of task coverage vs. expert workload by adding a balancing coefficient $\lambda > 0$, to the coverage function.

- Balanced Coverage is NP-Hard.
- C is a monotone, submodular function.

Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks and a set of n experts, find an assignment A of experts to tasks such that.

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

is maximized.

We tune the relative importance of task coverage vs. expert workload by adding a balancing coefficient $\lambda > 0$, to the coverage function.

- Balanced Coverage is NP-Hard.
- C is a monotone, submodular function.
- $L_{\max}$ does not have a concrete form, i.e. it is not linear or convex.

Balanced Coverage problem

Problem 1 (Balanced Coverage)

Given a set of m tasks and a set of n experts, find an assignment A of experts to tasks such that.

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

is maximized.

We tune the relative importance of task coverage vs. expert workload by adding a balancing coefficient $\lambda > 0$, to the coverage function.

- Balanced Coverage is NP-Hard.
- C is a monotone, submodular function.
- $L_{\max}$ does not have a concrete form, i.e. it is not linear or convex.
- F is not guaranteed to be positive.

Related Work

- Complete coverage as a hard constraint. [Lappas et al. 2009] [Bhowmik et al. 2014]
- Online setting [Anagnostopoulos et al. 2010]. Offline setting is framed as a load minimization problem under full coverage constraint.
- Partial coverage with a single task, and sum of expert costs [Nikolakaki et al. 2020]

Approximation guarantee

Beyond a multiplicative approximation...

Let A be the assignment returned by `ThresholdGreedy` and let OPT be the optimal assignment for the BALANCED COVERAGE problem. Ideally, we want a multiplicative approximation for some $\alpha < 1$.

$$F(A) \geq \alpha F(OPT)$$
$$C(A) - L_{\max}(A) \geq \alpha\big(C(OPT) - L_{\max}(OPT)\big)$$

Approximation guarantee

Beyond a multiplicative approximation...

Let A be the assignment returned by `ThresholdGreedy` and let OPT be the optimal assignment for the Balanced Coverage problem. Ideally, we want a multiplicative approximation for some $\alpha < 1$.

$$F(A) \geq \alpha F(OPT)$$
$$C(A) - L_{\max}(A) \geq \alpha\big(C(OPT) - L_{\max}(OPT)\big)$$

We adopt a weaker notion of approximation from [Harshaw et al. 2019], [Mitra et al. 2021]

$$C(A) - L_{\max}(A) \geq \alpha C(OPT) - \beta L_{\max}(OPT)$$

Approximation guarantee

Beyond a multiplicative approximation...

Let A be the assignment returned by `ThresholdGreedy` and let OPT be the optimal assignment for the BALANCED COVERAGE problem. Ideally, we want a multiplicative approximation for some $\alpha < 1$.

$$F(A) \geq \alpha F(OPT)$$
$$C(A) - L_{\max}(A) \geq \alpha\big(C(OPT) - L_{\max}(OPT)\big)$$

We adopt a weaker notion of approximation from [Harshaw et al. 2019], [Mitra et al. 2021]

$$C(A) - L_{\max}(A) \geq \alpha C(OPT) - \beta L_{\max}(OPT)$$

Our result: $\alpha = \left(1 - \frac{1}{e}\right), \beta = 1$

ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$
Output: An assignment, A of experts to tasks

```
for τ = 1, ..., m do
    Optimize for C(), A_τ = Submodular-Optimization(ℰ, 𝒥, τ)
    Compute objective, F_{A_τ} = C(A_τ) − τ
end for
return A_τ that maximizes F.
```

- **for** $\tau = 1, ..., m$ **do**
  - Optimize for $C()$, $A_\tau = \texttt{Submodular-Optimization}(\mathcal{E}, \mathcal{J}, \tau)$
  - Compute objective, $F_{A_\tau} = C(A_\tau) - \tau$
- **end for**
- **return** A_τ that maximizes F.

The ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$
Output: An assignment, A of experts to tasks

```
for τ = 1, ..., m do
    Optimize for C, A_τ = Greedy(ℰ, 𝒥, τ)
    Compute objective, F_{A_τ} = C(A_τ) − τ
end for
return A_τ that maximizes F.
```

The ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$
Output: An assignment, A of experts to tasks

for $\boxed{\tau = 1, ..., m}$ **do**
  Optimize for C, $\boxed{A_\tau = \texttt{Greedy}(\mathcal{E}, \mathcal{J}, \tau)}$
  Compute objective, $F_{A_\tau} = C(A_\tau) - \tau$
end for
return A_τ that maximizes F.

The ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$
Output: An assignment, A of experts to tasks

for $\boxed{\tau = 1, ..., m}$ **do**
  Optimize for C, $\boxed{A_\tau = \texttt{Greedy}(\mathcal{E}, \mathcal{J}, \tau)}$
  Compute objective, $F_{A_\tau} = C(A_\tau) - \tau$
end for
return A_τ that maximizes F.

- **Lazy evaluation.** Use submodularity of $C()$ to overcome the computational bottleneck of `Greedy` [Minoux, 1978].

The ThresholdGreedy algorithm

Algorithm `ThresholdGreedy`

Input: Set of m tasks $\mathcal{J}$ and n experts $\mathcal{E}$
Output: An assignment, A of experts to tasks

for $\boxed{\tau = 1, \ldots, m}$ **do**
  Optimize for C, $\boxed{A_\tau = \texttt{Greedy}(\mathcal{E}, \mathcal{J}, \tau)}$
  Compute objective, $F_{A_\tau} = C(A_\tau) - \tau$
end for
return A_τ that maximizes F.

- **Lazy evaluation.** Use submodularity of $C()$ to overcome the computational bottleneck of `Greedy` [Minoux, 1978].
- **Early termination.** The value of the objective computed by `ThresholdGreedy` for different values of τ is a unimodal function.

Experimental evaluation

Real world datasets. IMDB, Bibsonomy, Freelancer, Guru.

Dataset	Experts	Tasks	Skills	skills/ expert	skills/ task
IMDB-15	5551	18109	26	2.4	3.1
IMDB-18	3871	13183	26	2.1	2.7
IMDB-20	2176	7858	25	1.9	2.4
Bbsm-10	3044	21981	1000	13.7	4.8
Bbsm-15	1904	9061	1000	10.9	4.3
Bbsm-20	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

Real world datasets. IMDB, Bibsonomy, Freelancer, Guru.

Dataset	Experts	Tasks	Skills	skills/ expert	skills/ task
IMDB-15	5551	18109	26	2.4	3.1
IMDB-18	3871	13183	26	2.1	2.7
IMDB-20	2176	7858	25	1.9	2.4
Bbsm-10	3044	21981	1000	13.7	4.8
Bbsm-15	1904	9061	1000	10.9	4.3
Bbsm-20	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. `LPCover`: Offline LP-rounding algorithm by [Anagnostopoulos et al. 2010]
2. `TaskGreedy`: Greedy variant inspired by [Nikolakaki et al. 2020]
3. `NoUpdateGreedy`: Heuristic modification of `ThresholdGreedy`

Experimental evaluation

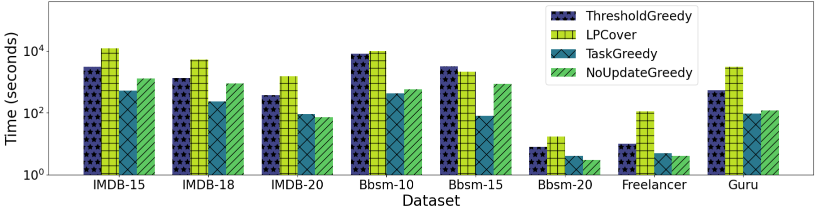
ThresholdGreedy performs well in practice

We observe that ThresholdGreedy outperforms the baselines both in terms of the objective (consistently higher) and in terms of the workload (predominantly lower).

Dataset	ThresholdGreedy			LPCover			TaskGreedy			NoUpdateGreedy		
	F	$L_{\max}$	$\frac{1}{m}C$	F	$L_{\max}$	$\frac{1}{m}C$	F	$L_{\max}$	$\frac{1}{m}C$	F	$L_{\max}$	$\frac{1}{m}C$
IMDB-15 ($\lambda = 0.05$)	**885**	**7**	**0.99**	777	100	0.97	475	362	0.92	720	150	0.96
IMDB-18 ($\lambda = 0.05$)	**643**	**8**	**0.99**	474	122	0.90	339	264	0.91	448	200	0.98
IMDB-20 ($\lambda = 0.1$)	**771**	**7**	**0.99**	676	87	0.97	644	118	0.97	650	100	0.95
Bbsm-10 ($\lambda = 0.1$)	**2039**	**70**	**0.96**	1691	282	0.90	1319	243	0.71	1097	200	0.59
Bbsm-15 ($\lambda = 0.05$)	**389**	**27**	**0.92**	336	55	0.86	96	126	0.49	129	250	0.84
Bbsm-20 ($\lambda = 1$)	**438**	**41**	0.57	418	84	**0.60**	402	93	0.59	408	94	**0.60**
Freelancer ($\lambda = 0.1$)	**88**	**6**	0.95	59	32	0.92	63	36	**0.99**	25	50	0.76
Guru ($\lambda = 0.1$)	**311**	**4**	**0.99**	287	25	0.98	225	30	0.80	17	33	0.16

Experimental evaluation

`ThresholdGreedy` runs fast in practice



Thank you!