Algorithms for Online Labour Marketplaces

Stefano Leonardi
Sapienza University of Rome

Based on work with

Aris Anagnostopoulos (Sapienza Univ.)
Carlos Castillo (UPF, Barcelona)
Adriano Fazzone (Sapienza Univ.)
Aris Gionis (Aalto Univ.)
Evimaria Terzi (Boston University)
Online labor marketplaces

• We will see an increase in the sophistication of systems that use and guide user actions

• Require models and algorithms to capture the human elements
  
  • What skills people have
  • Efficiency
  • Time availability
  • Human-human relationship
  • Incentive and behavioral issues
  • Human errors / disagreements
  • Work organization
Online collaborative systems

Several success stories indicate that much more is possible:

• Tagging/geotagging systems:

• Content creation systems:

• Online labor markets:

• Crowdsourcing:

• Polymath project:

• Open source community:
This lecture

We will look at two specific problems:

• **How can we form teams** of experts online when compatibility between experts is modelled by a social network

• How can we decide online when to use outsourced workers, when to hire workers in a team and when to fire inactive workers
We like to solve the above problems while achieving:

- Good performance of formed teams on allocated tasks
- Fair distribution of the task load between experts
- Low coordination overhead within a team
- Good trade-offs between outsourcing and hiring/salary cost
Online collaborative systems

Several success stories indicate that much more is possible:

- Tagging/geotagging systems:
- Content creation systems:
- Online labor markets:
  - Elance
  - VWorker
  - ODesk
  - freelancer
  - guru
  - Upwork
- Crowdsourcing:
  - Amazon mechanical turk
  - Google CrowdFlower
- Polymath project:
- Open source community:
Team formation
Team formation

Industrial and business settings

Cluster hires: Which experts should be hired?

Online collaborations: Can teams really work online?
Team formation

Educational settings

Traditional classroom: How to create good study groups?

Massive Online Courses (MOOCs): How to bring in social aspects?
Team formation

Research environments

- Writing proposals with others
- Cluster hires with diversity
- Collaborative problem solving
OCEAN'S ELEVEN
Place your bets.

George Clooney
Matt Damon
Andy Garcia
Brad Pitt
Julie Roberts
That Big One!!

FRANK SINATRA \ DEAN MARTIN
SAMMY DAVIS JR. \ PETER LAWFORD
ANGIE DICKINSON

OCEANS 11

Nobody else would have dared it because nobody else would have the nerve! Just Danny Ocean and his 11 pals – the crazy night they blew all the lights in Las Vegas...

TECHNOLOR \ PANAVISION \ WARNER BROS.

RICHARD CONTE \ CESAR ROMERO \ PATRICE WYMORE \ JOEY BISHOP
oDesk – Team sizes over time

Number of contractors working in teams of given team size, over time, on oDesk.com
The Online Team Formation Problem
Related work

**Business & Management Science**
- [Lau et al. 1998]
- [Li et al. 2005]
- [Choi et al. 2010]
- [Thatcher et al. 2003]
- [Molleman 2005]
- [Polzer et al. 2006]
- [Bezrukova et al. 2009]
- [Pearsall et al. 2008]
- [Jehn et al. 2010]
- [Gratton et a. 2007]
- [Shaw 2004]

**Education Sciences**
- [Slavin 1987]
- [Kulik 1982]
- [Kerchoff 1986]
- [Kulik et al. 1992]
- [Mislevy 1983]
- [Lazarowitz et al. 1995]
- [Vygotsky et al. 1978]

**Social Research**
- [DeGroot 1974]
- [Friedkin et al. 1990]
- [Jackson et al. 2008]
- [Friedkin et al. 1999]

**Computer Science**
- [Anagnostopoulos et al. 2010]
- [Okimoto et al. 2015]
- [Agrawal et al. 2014]
- [Lappas et al. 2009]
- [Sozio et al. 2010]
- [Gajewar et al. 2012]
- [Anagnostopoulos et al. 2012]
- [Yildiz et al. 2013]
- [Kargar et al. 2013]
- [Dorn et al. 2010]
- [Kargar et al. 2011]
- [Li et al. 2010]
- [Bell, 2007]
- [Majumder 2012]
- [Golshan et al. 2014]
Set-cover view of team formation

Experts

- JAVA
- Python
- HTML 5

- C++
- Objective C

Single task

- JAVA, C++

- SEO
- HTML 5
Set-cover view of team formation

Experts

[ ![JAVA](image), Python, HTML 5 ]

[ ![C++](image), Objective C ]

[ ![SEO](image), HTML5 ]

Single task

JAVA, C++
Basic formulation: set cover

**Problem:** Given a pool of experts, a single task hire the minimum-cost subset of experts that can complete (i.e., cover) the task

**Facts:**
- The problem is NP-hard
- Greedy algorithm is a good approximation algorithm
Setting

- Pool of people with different skills
- Stream of tasks/jobs arriving online
- Tasks have some skill requirements
- Create teams on-the-fly for each job
  - Select the right team
  - Satisfy various criteria
Criteria

- **Fitness**
  - E.g. success rate, maximize expected number of successful tasks
  - Depends on:
    - People skills
    - Ability to coordinate

- **Fairness**: everybody should be involved in roughly the same number of tasks

- **Efficiency**:
  - Cost of outsourced tasks vs cost of hired workers

- **Trade-offs may appear**: do you see how?
Basic formulation: Skills and people

- $n$ People/Experts
- $m$ Skills
- Each person has some skills

\[
p^1, p^2, \ldots, p^n
\]
\[
S = \{0, 1\}^m
\]
\[
p^i \in S
\]
Basic formulation: jobs & teams

- Stream of $k$ Jobs/Tasks
- A job requires some skills
- $k$ Teams are created online
- A team must cover all job skills

\[
J^1, J^2, \ldots, J^k
\]
\[
J^j \in S
\]
\[
Q^j \subseteq \{p^1, p^2, \ldots, p^n\}
\]

Load:
\[
L(p) = |\{j; p \in Q^j\}|
\]
Coordination cost

- **Coordination cost** measures the compatibility of the team members
- Example of $d(p^i, p^j)$:
  - Degree of knowledge
  - Time-zone difference
  - Past collaboration

- Select teams that minimizes **coordination cost** $c(Q)$:
  - Steiner-tree cost
  - Diameter
  - Sum of distances
Framework

- Jobs/Tasks \( (k) \)
- People \( (n) \)
- Skills \( (m) \)
- Teams \( (k) \)
- Distance between people
- Team coordination cost
- Score/fitness
- Load

\[
\mathcal{J} = \{ J^j; \ j = 1, 2, \ldots, k \} \\
\mathcal{P} = \{ p^j; \ j = 1, 2, \ldots, n \} \\
S = \{0, 1\}^m \quad \text{or} \quad S = [0, 1]^m \\
Q^j \subseteq \mathcal{P} \\
d(p^i, p^j) \\
c(Q^j) \\
s(Q^j, J^j) \\
L(p) = |\{j; p \in Q^j\}| 
\]
Binary Profiles

In this talk (and most the work): Binary skill profiles

\[ S = \{0, 1\}^m \]

- A person either has a skill or not
- Team has a skill if a person has it
- A job either requires it or not
- Score of a team \( Q \) for task \( J \)

\[ s(Q, J) = \begin{cases} 1, & \text{if } Q \text{ has all the skills of } J, \\ 0, & \text{otherwise.} \end{cases} \]

- Covering problem
- Other options are available
Online Balanced Task Covering
1. Balanced task covering

- Cover all the jobs
  \[ s(Q^j, J^j) = 1 \quad \forall j = 1, \ldots, k \]

- Objective
  \[ \min \max_j L(p^j) \]

- NP-hard problem even with \( k = 2 \)

- Offline setting has a randomized approx. algo.
  That succeeds with prob \( 1 - \delta \) with ratio
  \[ O\left( \log \left( \frac{mk + n}{\delta} \right) \right) \]

- Does it exist an O(1)-APX?
Our modeling approach

- Set a desirable coordination cost upper bound $B$
- **Online** solve

\[
\min_i \max \sum_{j} L(p_i^j) \\
Q_j^i \text{ covers } J_j^i \quad \forall j \\
c(Q_j^i) \leq B \quad \forall j.
\]

- Must concurrently solve various combinatorial problems:
  - Set cover
  - Steiner tree
  - Online makespan minimization
Our modeling approach

<table>
<thead>
<tr>
<th>Job</th>
<th>p₁</th>
<th>p₂</th>
<th>p₃</th>
<th>p₄</th>
<th>p₅</th>
<th>p₆</th>
<th>p₇</th>
<th>Qⱼ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Q₁ = {p₂, p₄, p₅}</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Q₂ = {p₁, p₄, p₆}</td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Q₃ = {p₃, p₄}</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Q₄ = {p₁, p₅, p₇}</td>
</tr>
<tr>
<td>5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Q₅ = {p₂, p₃, p₄, p₅}</td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Q₆ = {p₃, p₅, p₆}</td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Q₇ = {p₁, p₂}</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>Q₈ = {p₁, p₂, p₃, p₄, p₇}</td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Q₉ = {p₃, p₄, p₅}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load</th>
<th>4</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>5</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
</table>

Balanced task covering – Online

- Evaluate by **competitive ratio**
  - Compare with optimal offline assignment
  - Offline has full information
- Simple heuristics
  - Assemble the team of minimum size
  - Assemble the team that minimize the maximum load of a person: $\max_{p \in Q} L_t^t(p)$
  - Assemble the team that minimize the sum of the loads of the team: $\sum_{p \in Q} L_t^t(p)$
  - Competitive ratios are bad: $\Omega(n), \Omega(k), \Omega(\sqrt{m})$
- In practice some are OK
Algorithm ExpLoad

When a task arrives at time $t$

- Weight each person $p$ by $(2n)^{L_t(p)}$

- Select team $Q$ that covers all task skills and minimizes
  \[ \sum_{p \in Q} (2n)^{L_t(p)} \]

- Weighted set cover problem

- **Theorem.** Competitive ratio = $O(\log m \log k)$
Experiments
### Mapping of data to problem instances

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Experts</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>Movie directors</td>
<td>Audition actors</td>
</tr>
<tr>
<td>Bibsonomy</td>
<td>Prolific scientists</td>
<td>Interview scientists</td>
</tr>
<tr>
<td>Flickr</td>
<td>Prolific photographers</td>
<td>Judge photos</td>
</tr>
</tbody>
</table>

### Summary statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Experts</th>
<th>Tasks</th>
<th>Skills</th>
<th>Skills/expert</th>
<th>Skills/task</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>725</td>
<td>2173</td>
<td>21</td>
<td>2.96</td>
<td>11.10</td>
</tr>
<tr>
<td>Bibsonomy</td>
<td>816</td>
<td>35506</td>
<td>793</td>
<td>7.64</td>
<td>4.44</td>
</tr>
<tr>
<td>Flickr.art</td>
<td>504</td>
<td>59869</td>
<td>12913</td>
<td>49.90</td>
<td>15.73</td>
</tr>
<tr>
<td>Flickr.nature</td>
<td>2879</td>
<td>112467</td>
<td>26379</td>
<td>31.25</td>
<td>15.45</td>
</tr>
</tbody>
</table>
We report mean, maximum, and additional columns as follows: $\phi_{.9}$ denotes the 90\% quantile; $\sigma_{.9}$ is the maximum team size that an algorithm allocates provided that each task is covered only up to 90\% of the required skills; finally, $\lambda_{.1}$ is the mean load of the 10\% more loaded experts.

<table>
<thead>
<tr>
<th>Method</th>
<th>Team size statistics</th>
<th>Experts load statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>$\phi_{.9}$</td>
</tr>
<tr>
<td>IMDB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>2.31</td>
<td>4</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>3.27</td>
<td>4</td>
</tr>
<tr>
<td>SumLoad</td>
<td>4.75</td>
<td>7</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>3.80</td>
<td>5</td>
</tr>
<tr>
<td>Bibsonomy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>2.70</td>
<td>5</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>2.92</td>
<td>5</td>
</tr>
<tr>
<td>SumLoad</td>
<td>3.13</td>
<td>6</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>2.83</td>
<td>5</td>
</tr>
<tr>
<td>Flickr.nature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>6.34</td>
<td>10</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>7.38</td>
<td>11</td>
</tr>
<tr>
<td>SumLoad</td>
<td>7.53</td>
<td>12</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>7.08</td>
<td>11</td>
</tr>
</tbody>
</table>
We report mean, maximum, and additional columns as follows: $\phi_{.9}$ denotes the 90% quantile; $\sigma_{.9}$ is the maximum team size that an algorithm allocates provided that each task is covered only up to 90% of the required skills; finally, $\lambda_{.1}$ is the mean load of the 10% more loaded experts.

<table>
<thead>
<tr>
<th>Method</th>
<th>Team size statistics</th>
<th>Experts load statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean  $\phi_{.9}$  $\sigma_{.9}$ max</td>
<td>mean  $\phi_{.9}$ $\lambda_{.1}$ max</td>
</tr>
<tr>
<td><strong>IMDB</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>2.31 4 3 5</td>
<td>6.92 11 58 1260</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>3.27 4 3 7</td>
<td>9.80 45 53 65</td>
</tr>
<tr>
<td>SumLoad</td>
<td>4.75 7 3 10</td>
<td>14.23 32 46 65</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>3.80 5 3 9</td>
<td>11.38 32 47 64</td>
</tr>
<tr>
<td><strong>Bibson</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>2.70 5 5 22</td>
<td><strong>117.66</strong> 251 397 1417</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>2.92 5 3 22</td>
<td><strong>127.13</strong> 248 353 700</td>
</tr>
<tr>
<td>SumLoad</td>
<td>3.13 6 7 25</td>
<td><strong>136.05</strong> 244 343 701</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>2.83 5 4 22</td>
<td><strong>123.27</strong> 258 365 700</td>
</tr>
<tr>
<td><strong>Flickr</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>6.34 10 25 29</td>
<td>247.85 439 823 6645</td>
</tr>
<tr>
<td>MaxLoad</td>
<td>7.38 11 27 31</td>
<td>288.22 468 571 941</td>
</tr>
<tr>
<td>SumLoad</td>
<td>7.53 12 30 35</td>
<td>294.09 438 535 937</td>
</tr>
<tr>
<td>ExpLoad</td>
<td>7.08 11 28 34</td>
<td>276.60 475 587 964</td>
</tr>
</tbody>
</table>
Online Balanced Task Covering with Coordination Cost
2. Coordination cost

- Have not taken into account **coordination cost**

- Distance between people $d(p^i, p^j)$

- Team coordination cost $c(Q^j)$

- Select teams that minimizes $c(Q^j)$
  - Steiner-tree cost
  - Diameter
  - Sum of distances
Coordination cost

- Steiner-tree cost
- Diameter
- Sum of distances

$$\sum_{p^i, p^j \in Q} d(p^i, p^j)$$
Conflicting goals

• We want solutions that minimize
  – Load
  – Coordination cost

and satisfy each job.
Our modeling approach

- Set a desirable coordination cost upper bound $B$
- Online solve

\[
\min \max_i L(p^i) \\
\text{s.t. } s(J^j, Q^j) = 1 \quad \forall j \in \mathcal{J} \\
c(Q^j) \leq B \quad \forall j \in \mathcal{J}.
\]

- 3 different problems for the 3 different coordination costs
- This talk: focus on Steiner tree coordination cost
Algorithm

At every step t:

• Combine ExpLoad with coordination cost constraint ⇒

• Find a team that:
  – Covers all required skills
  – Satisfies $c(Q) \leq B$
  – Minimizes $\sum_{p \in Q} (2n)^{L_t(p)}$

• How?
At every step $t$

- Incorporate to the graph $\lambda (2n)^{L_t(p)}$
- Solve a **variant of Steiner tree**. Get a solution that
  - Covers all required skills
  - Satisfies $c(Q) \leq \beta B$
  - $\alpha$-approximates $\sum_{p \in Q} (2n)^{L_t(p)}$
- Different graphs in the **family** tradeoff between $\alpha, \beta$
Result

We wanted: \[ \min \max_i L(p^i) \]
\[ s(J^j, Q^j) = 1 \quad \forall j \in \mathcal{J} \]
\[ c(Q^j) \leq B \quad \forall j \in \mathcal{J}. \]

Theorem. The algorithm satisfies:

\( \alpha \)-approximates \[ \min \max_i L(p^i) \]
\[ s(J^j, Q^j) = 1 \quad \forall j \in \mathcal{J} \]
\[ c(Q^j) \leq \beta B \quad \forall j \in \mathcal{J}. \]

- Can obtain \( \alpha, \beta = O(\log(n, m, k)) \)
Group Steiner Tree

- Group Steiner Tree: Construct a Steiner tree that connects at least one node for each group
- Heuristics for Group Steiner Tree:

  1. LLT [Lappas, Liu, Terzi, KDD 2009]
     - Connect each skill $J_l$ to all experts that own the skill
     - Construct a Steiner tree connecting all skills of $J$
Group Steiner tree

2. Set Cover (SC): Cover all skills with experts.

At each step select the most effective expert with cost-effectiveness:

\[
\frac{\text{gain}(p^j)}{\text{loss}(p^j)}
\]

- \text{gain}(p^j) \quad \# \text{newly covered skills}
- \text{loss}(p^j) \quad \text{distance to experts selected so far plus } \lambda \times \text{ExpLoad of the expert}
Experiments Bibsonomy

Experts = prolific authors
Task = interview scientists
Distance = f( #collaborations )
Optimize over $\lambda$
Experiments Bibsonomy

Experts = prolific authors
Task = interview scientists
Distance = f( #collaborations )
Experiments IMDB

Experts = directors
Task = find a cast
Distance = \( f( \# \text{common actors directed} ) \)
Online Team Formation with Outsourcing
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online.
- Tasks and workers are represented as a set of skills.
- At each time step a new task arrives.
- A team must be created to cover all the task skills.
- Each member of the team can be either hired or a freelance worker.
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost.
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker \( w_r \) has a hiring \((C_r)\), salary \((\sigma_r)\), and outsourcing \((\lambda_r)\) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

\[
\begin{align*}
w_1 &= \{s_1\} \\
w_2 &= \{s_2, s_4\} \\
w_3 &= \{s_2, s_3\}
\end{align*}
\]
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online.
- Tasks and workers are represented as a set of skills.
- At each time step a new task arrives.
- A team must be created to cover all the task skills.
- Each member of the team can be either hired or a freelance worker.
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost.
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

\[ w_1 = \{s_1\} \quad w_2 = \{s_2, s_4\} \quad w_3 = \{s_2, s_3\} \quad J^1 = \{s_1, s_2\} \]
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online.
- Tasks and workers are represented as a set of skills.
- At each time step a new task arrives.
- A team must be created to cover all the task skills.
- Each member of the team can be either hired or a freelance worker.
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost.
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

\[ w_1 = \{s_1\} \quad w_2 = \{s_2, s_4\} \quad w_3 = \{s_2, s_3\} \]

\[ J^1 = \{s_1, s_2\} \]

**Hired:** $w_1$, $w_2$

**Outsourced:** $w_1$, $w_2$

**Cost:** $\lambda_1 + \lambda_2$
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

### Example

- $w_1 = \{s_1\}$
- $w_2 = \{s_2, s_4\}$
- $w_3 = \{s_2, s_3\}$
- $J^1 = \{s_1, s_2\}$
- $J^2 = \{s_1, s_3\}$

**Hired:**
- $w_1$, $w_2$

**Outsourced:**
- $w_1$, $w_2$

**Cost:**
- $\lambda_1 + \lambda_2$
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

\[w_1 = \{s_1\}\]
\[w_2 = \{s_2, s_4\}\]
\[w_3 = \{s_2, s_3\}\]
\[J^1 = \{s_1, s_2\}\]
\[J^2 = \{s_1, s_3\}\]

**Hired:**

- $w_1$

**Outsourced:**

- $w_1, w_2$
- $w_3$

**Cost:**

- $\lambda_1 + \lambda_2$
- $C_1 + \sigma_1 + \lambda_3$
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker \( w_r \) has a hiring (\( C_r \)), salary (\( \sigma_r \)), and outsourcing (\( \lambda_r \)) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

\[
\begin{align*}
J^1 &= \{s_1, s_2\} \\
J^2 &= \{s_1, s_3\} \\
J^3 &= \{s_2, s_3\}
\end{align*}
\]

\[
\begin{align*}
w_1 &= \{s_1\} \\
w_2 &= \{s_2, s_4\} \\
w_3 &= \{s_2, s_3\}
\end{align*}
\]

\[
\begin{align*}
\text{Hired:} & \quad - \quad w_1 \\
\text{Outsourced:} & \quad w_1, w_2 \\
\text{Cost:} & \quad \lambda_1 + \lambda_2 \\
\text{Cost:} & \quad C_1 + \sigma_1 + \lambda_3
\end{align*}
\]
Team Formation with Outsourcing

• Create teams of workers for solving tasks/jobs that arrive online
• Tasks and workers are represented as a set of skills
• At each time step a new task arrives
• A team must be created to cover all the task skills
• Each member of the team can be either hired or a freelance worker
• Each worker \( w_r \) has a hiring (\( C_r \)), salary (\( \sigma_r \)), and outsourcing (\( \lambda_r \)) cost
• **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

**Hired:**
- \( w_1 \)
- \( w_2 \)

**Outsourced:**
- \( w_3 \)

**Cost:**
- \( \lambda_1 + \lambda_2 \)
- \( C_1 + \sigma_1 + \lambda_3 \)
- \( \sigma_1 + C_3 + \sigma_3 \)
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

**Hired:** $w_1$, $w_2$, $w_3$, $w_1$, $w_3$

**Outsourced:** $w_1$, $w_2$, $w_3$

**Cost:** $\lambda_1 + \lambda_2$, $C_1 + \sigma_1 + \lambda_3$, $\sigma_1 + C_3 + \sigma_3$
Team Formation with Outsourcing

- Create teams of workers for solving tasks/jobs that arrive online
- Tasks and workers are represented as a set of skills
- At each time step a new task arrives
- A team must be created to cover all the task skills
- Each member of the team can be either hired or a freelance worker
- Each worker $w_r$ has a hiring ($C_r$), salary ($\sigma_r$), and outsourcing ($\lambda_r$) cost
- **Goal:** design an online, cost-minimizing algorithm for hiring, firing, and outsourcing.

$w_1 = \{s_1\}$

$w_2 = \{s_2, s_4\}$

$w_3 = \{s_2, s_3\}$

$J^1 = \{s_1, s_2\}$

$J^2 = \{s_1, s_3\}$

$J^3 = \{s_2, s_3\}$

$J^4 = \{s_2, s_3, s_4\}$

**Hired:**
- $w_1$
- $w_2$
- $w_1$, $w_3$
- $w_3$

**Outsourced:**
- $w_1$, $w_2$
- $w_3$
- $w_2$

**Cost:**
- $\lambda_1 + \lambda_2$
- $C_1 + \sigma_1 + \lambda_3$
- $\sigma_1 + C_3 + \sigma_3$
- $\sigma_3 + \lambda_2$
Quality of online algorithms

Goal:
Design a polynomial-time online algorithm for the TFO problem with a small competitive approximation ratio.

Competitive approximation ratio of an online algorithm:

$$\max_{\text{Stream of Tasks}} \frac{\text{Cost of the Algorithm}}{\text{Cost of an Optimal Algorithm}}$$
Methodology

1. **TFO-LumpSum**: no salary and no firing. Design a polynomial time online algorithm with a logarithmic competitive approximation ratio.

2. **TFO**: full version. Design a polynomial time online algorithm with a logarithmic competitive approximation ratio, by modifying the algorithm for TFO-LumpSum.
Methodology

1. **TFO-LumpSum**: no salary and no firing.
   Design a polynomial time online algorithm with a logarithmic competitive approximation ratio.

2. **TFO**: full version.
   Design a polynomial time online algorithm with a logarithmic competitive approximation ratio, by modifying the algorithm for TFO-LumpSum.
TFO-LumpSum: Online primal–dual technique

- $x_r = 1$ if worker $W^r$ is hired, 0 otherwise.
- $f_{rt} = 1$ if worker $W^r$ is outsourced for performing task $J^t$, 0 otherwise.

Linear program for LUMPSUM:

$$\min \sum_{r=1}^{n} \left( C_r x_r + \lambda_r \sum_{t=1}^{T} f_{rt} \right)$$

subject to: $\forall t = 1, \ldots, T, \ell \in J^t$:

$$\sum_{W^r \in P_\ell} (x_r + f_{rt}) \geq 1$$

$\forall t = 1, \ldots, T, r = 1, \ldots, n$:

$$x_r, f_{rt} \geq 0$$

$C_r$ Hiring fee, paid when worker $r$ is hired.
$\lambda_r$ Outsourcing fee, paid every time $r$ performs a task.
TFO-LumpSum: Online primal–dual technique

- \( x_r = 1 \) if worker \( W^r \) is hired, 0 otherwise.
- \( f_{rt} = 1 \) if worker \( W^r \) is outsourced for performing task \( J^t \), 0 otherwise.

Linear program for LUMPSUM:

\[
\begin{align*}
\min & \sum_{r=1}^{n} \left( C_r x_r + \lambda_r \sum_{t=1}^{T} f_{rt} \right) \\
\text{subject to: } & \forall t = 1, \ldots, T, \ell \in J^t : \\
& \sum_{W^r \in P_\ell} (x_r + f_{rt}) \geq 1 \\
\forall t = 1, \ldots, T, r = 1, \ldots, n: \\
& x_r, f_{rt} \geq 0
\end{align*}
\]

The dual of the linear program for LUMPSUM:

\[
\begin{align*}
\max & \sum_{t=1}^{T} \sum_{\ell \in J^t} u_{\ell t} \\
\text{subject to: } & \forall r = 1, \ldots, n: \\
& \sum_{t=1}^{T} \sum_{\ell \in J^t \cap W^r} u_{\ell t} \leq C_r \\
& \forall t = 1, \ldots, T, r = 1, \ldots, n: \\
& \sum_{\ell \in J^t \cap W^r} u_{\ell t} \leq \lambda_r \\
\forall t = 1, \ldots, T, \ell \in J^t: \\
& u_{\ell t} \geq 0,
\end{align*}
\]

\( C_r \)  Hiring fee, paid when worker \( r \) is hired.
\( \lambda_r \)  Outsourcing fee, paid every time \( r \) performs a task.
TFO-LumpSum: Algorithm

When job $J^T$ arrives:

Step 1: Increase potentials:

```latex
\textbf{for each} skill $\ell \in J^T_F$:
\begin{align*}
\text{while } \sum_{W^r \in P_{\ell}} (\tilde{x}_r + \tilde{f}_{rT}) < 1: \\
& u_{\ell t} \leftarrow u_{\ell t} + 1 \\
& \text{for each } W^r \in P_{\ell}: \tilde{x}_r \leftarrow \tilde{x}_r \left(1 + \frac{1}{C_r}\right) + \frac{1}{nC_r} \\
& \text{for each } W^r \in P_{\ell}: \tilde{f}_{rT} \leftarrow \tilde{f}_{rT} \left(1 + \frac{1}{\lambda_r}\right) + \frac{1}{n\lambda_r}
\end{align*}
```

Step 2: Perform randomized rounding to decide which worker to hire and to whom to outsource

```latex
\textbf{repeat } \rho \text{ times: }
\begin{align*}
& \text{for each } W^r \in P^F_T \\
& \quad \text{with probability } \Delta \tilde{x}_r: \\
& \quad \text{hire worker } W^r \text{ (set } x_r \leftarrow 1) \\
& \quad \text{with probability } \tilde{f}_{rT}: \\
& \quad \text{outsource worker } W^r \text{ (set } f_{rT} \leftarrow 1)
\end{align*}
```

Running time:

\[ O \left( n \left( |J^T| \log n + \log m + \log C^* \right) \right) \]

Competitive approximation ratio:

\[ O(\log n(\log m + \log C^*)) \]
Methodology

1. **TFO-LumpSum**: no salary and no firing. Design a polynomial time online algorithm with a logarithmic competitive approximation ratio.

2. **TFO**: full version. Design a polynomial time online algorithm with a logarithmic competitive approximation ratio, by modifying the algorithm for TFO-LumpSum.
**Theorem.** There exists a polynomial time online algorithm for TFO with competitive approximation ratio

\[ O((\log m + \log C^* + \log T^*) \log n) \]

**Proof.** Use online primal–dual schema with a more complicated set of integer and linear programs.
Theorem. There exists a polynomial time online algorithm for TFO with competitive approximation ratio

\[ O((\log m + \log C^* + \log T^*) \log n) \]

Proof. Use online primal–dual schema with a more complicated set of integer and linear programs.

Linear program for TFO:

\[
\min \sum_{r=1}^{n} \left[ \sum_{I \in I} C_r x(r, I) + \sum_{t=1}^{T} \lambda_r f_{rt} + \sum_{t=1}^{T} \sigma_r g_{rt} \right]
\]

subject to

\[ \forall t = 1 \ldots T, \ell \in J^t : \]

\[ \sum_{W^r \in P_t} \left( f_{rt} + \sum_{I : t \in I} x(r, I) \right) \geq 1. \]

\[ \forall t = 1 \ldots T, r = 1 \ldots n : \]

\[ \sum_{I \in I : t \in I} x(r, I) \leq g_{rt} \]

\[ \forall t = 1 \ldots T, r = 1 \ldots n, I \in I : \]

\[ x(r, I), f_{rt}, g_{rt} \geq 0 \]

\[ m: \text{ total number of skills.} \]
\[ C^*: \text{ maximum hiring cost.} \]
\[ T^*: \text{ number of tasks in the stream.} \]
\[ n: \text{ total number of workers.} \]
Experiments: Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>upwork</th>
<th>freelancer</th>
<th>guru</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skills ($m$)</td>
<td>2,335</td>
<td>175</td>
<td>1,639</td>
</tr>
<tr>
<td>Workers ($n$)</td>
<td>18,000</td>
<td>1,211</td>
<td>6,119</td>
</tr>
<tr>
<td>Tasks ($T$)</td>
<td>50,000</td>
<td>992</td>
<td>3,194</td>
</tr>
<tr>
<td>... distinct</td>
<td>50,000</td>
<td>600</td>
<td>2,939</td>
</tr>
<tr>
<td>... avg. similarity (Jaccard)</td>
<td>0.095</td>
<td>0.045</td>
<td>0.018</td>
</tr>
<tr>
<td>Average Skills/worker</td>
<td>6.29</td>
<td>1.45</td>
<td>13.07</td>
</tr>
<tr>
<td>Average Skills/task</td>
<td>41.88</td>
<td>2.86</td>
<td>5.24</td>
</tr>
</tbody>
</table>

Generation of the stream of tasks:

- Pick a random task as pivot.
- With probability $1-1/p$, pick the next task within those whose Jaccard similarity with the pivot is at least 0.5.
- With probability $1/p$, pick another random task as a pivot.
Experiments: TFO vs. Heuristics

\[
C_r = 4\lambda_r \quad \sigma_r = \lambda_r/10 \quad p = 100
\]

Generation of the stream of tasks:

- Pick a random task as pivot.
- With probability 1-\textbf{1/100}, pick the next task within those whose Jaccard similarity with the pivot is at least 0.5.
- With probability \textbf{1/100}, pick another random task as a pivot.
Experiments: TFO vs. Always Outsource

(a) UpWork: TFO vs. Always-Outsource

(c) Freelancer: TFO vs. Always-Outsource

(e) Guru: TFO vs. Always-Outsource
Experiments: TFO vs. Always Outsource

(a) UpWork: TFO vs. Always-Outsource

(b) UpWork: TFO-Adaptive vs. Always-Outsource

(c) Freelancer: TFO vs. Always-Outsource

(d) Freelancer: TFO-Adaptive vs. Always-Outsource

(e) Guru: TFO vs. Always-Outsource

(f) Guru: TFO-Adaptive vs. Always-Outsource

Aris Anagnostopoulos  Algorithms for Hiring and Outsourcing in the Online Marketplace  London, KDD 2018
Conclusions

- Defined a novel online team formation problem in a hire-or-outsource setting
- Designed polynomial-time online algorithms with competitive approximation ratios
- Shown the applicability of our algorithmic solutions, by performing experiments using data from online outsourcing marketplaces
- Showed the practical use of the online primal–dual schema

Future work:

- Relax/test some of the modeling assumptions
- **k-TFO**: # of hired workers can be at most a fixed number k
Future directions

Modeling
• Several human elements: capabilities, cooperation, etc.
• Application dependent

Learning
• Learning profiles of experts
• Learn coordination based on performance

Algorithmic
• Matching problems
• How to train experts
• Explore-exploit tradeoff
Future directions

Game-theoretic

- Incentives for participation and rewarding mechanisms
- Issues on cooperation / altruism / trust
Thanks!

Questions, comments, etc.:

Stefano:  http://www.dis.uniroma1.it/~leon