Task routing

Problems:
- Distributing to many non-expert and aggregate outputs to approximate expert answers.
- Identifying expert in a crowd.
- Trust users based on their (estimated) expertise.

Methods:
- Passive: assign randomly and discard un-trusted output afterward,
- Active: Assign HIT to the most suited workers.

Examples
- Paying Turkers to translate a long text,
- Hiring a small team of worker to evaluate documents,
- Find an expert to check a mathematical proof,
- Getting gamers to make an alignment with Phylo,
- Asking someone to find a recycling box near their current location.

Expert vs. non-expert

Expert:
- Generate better, faster and more accurate solutions,
- Can identify deeper structures in a problem,
- Efficient information retrieval.

Non-expert:
- Bigger force task,
- Promote learning and assign difficult task to non-experts.

Which factors influence the routing strategy?

- Human-computer interactions
  - Assign task iff a task is available,
  - Always assign task,
  - Seek experts and assign them a task,

- Scale of the system
  - How many tasks?
  - How many potential workers?

- Characterizing the crowd
  - Duration of human-computer interaction,
  - Can we estimate the reliability of the workers?
Push vs. Pull

**Push**: System controls the distribution of tasks. The worker is *passive*.

**Pull**: Workers can browse, visualize & search data. The workers are *active*.

---

**Push approach: Matching**

Workers have strict preferences. Solve the stable marriage problem (Gale-Shapley algorithm).

**Task assignment**

- **Goal**: Given a set of preferences among workers and tasks, design a self-reinforcing assignment process.
- **Unstable pair**: worker *w* and task *t* are unstable if:
  - *w* prefers *t* to its assigned task.
  - *t* prefers *w* to its assigned student.
- **Stable assignment**: Assignment with no unstable pairs.
  - Natural and desirable condition.
  - Individual self-interest.

**Stable marriage problem**

**Goal**: Given *n* workers and *n* tasks, find a "suitable" matching.
- Each worker lists tasks in order of preference from best to worst.
- Each task has preference for most qualified workers.

<table>
<thead>
<tr>
<th>Worker's preferences</th>
<th>Task's preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier A B C</td>
<td>A Yuri Xavier Zoran</td>
</tr>
<tr>
<td>Yuri B A C</td>
<td>B Xavier Yuri Zoran</td>
</tr>
<tr>
<td>Zoran A B C</td>
<td>C Xavier Yuri Zoran</td>
</tr>
</tbody>
</table>

**Example**

Q: Is X-C, Y-B, Z-A a good assignment?
Stable marriage problem

Consider a complete bipartite graph such that |A|=|B|=n.
- Each member of A has a preference ordering of members of B.
- Each member of B has a preference ordering of members of A.

Algorithm for finding a matching:
- Each A member tries to match a member of B, in preference order.
- Each B member accepts the first assignment from a member of A, but can reject it if it receives an assignment from a member of A that it prefers more.

Gale-Shapley algorithm

For each $\alpha \in A$, let $\text{pref}[\alpha]$ be its ordering of preferences in B
For each $\beta \in B$, let $\text{pref}[\beta]$ be its ordering of preferences in A
Let matching be a set of crossing edges between A and B

while there is $\alpha \in A$ not yet matched do
  $\beta \leftarrow \text{pref}[\alpha].\text{removeFirst}()$
  if $\beta$ not yet matched then
    matching $\leftarrow$ matching $\cup \{(\alpha, \beta)\}$
  else
    $\gamma \leftarrow \beta$’s current match
    if $\beta$ prefers $\alpha$ over $\gamma$ then
      matching $\leftarrow$ matching $\cup \{(\gamma, \beta)\} - \{(\alpha, \beta)\}$
  end
return matching
Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

Example

Workers

Tasks

Worker’s preferences

Task’s preferences

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran

1st 2nd 3rd

Xavier B A C

Zoran A C B

1st 2nd 3rd

A Zoran Xavier Yuri

C Xavier Yuri Zoran
Worker’s expertise is known (or estimated). A coalition is a group of agents which cooperate in order to achieve a common task. (Coalition Problem). Given \( \langle A, H, T \rangle \), the coalition problem is to assign tasks \( t \in T \) to coalitions of agents \( C \subseteq A \) such that the total utility is maximized and the precedence order is respected.

(Shahaf & Horvitz, 2010)

### Single task scenario

Allocation based on estimate of worker’s attributes:
- System has knowledge of worker’s skills.
- Participants associated with cost and utility (e.g. location)

In both case, routing is an NP-hard problems (reduction from weighted exact set cover problem) but can be approximated using greedy algorithms

**Problem**: How to perform allocation if no one has the skills to perform a specific task?

**Approach**: Decompose

(Shahaf & Horvitz, 2010)

### Approximation scheme

Workers are vertices and allocations of tasks are hyperedges. Color indicates the task and weight the expected reward.

**Objective**: find a maximum value matching in the graph such that only one edge of each color is used.

**Result**: A greedy algorithm yields a \( \frac{1}{2} \)-approximation.

(Shahaf & Horvitz, 2010)

### Push approach: Allocation

Worker’s expertise is known (or estimated). A coalition is a group of agents which cooperate in order to achieve a common task.

(Shahaf & Horvitz, 2010)

### Weighted Exact Set Cover

**Definition (weighted exact set cover)**: Given a set \( X \) and a set \( S \) of subsets of \( X \) with associated rewards \( R(s) \) for each \( s \in S \), an exact set cover \( S^* \) is a subset of \( S \) such that every element in \( X \) is contained in exactly one set in \( S^* \). The goal is to find the exact cover with the maximal reward.

### Tasks with multiple scenarios

Multi-commodity flow problem on \( G = (V, E, w) \).
- \( s \in V \) represents a language.
- Directed edges are resources.
- Edge’s weight represents the degree of competence.
- Source & sink nodes represent the source & target language.

NP-complete problem, typically solved with a fully polynomial time approximation scheme and Linear Programming.

(Shahaf & Horvitz, 2010)
Linear Programming Approach

**Concept:** Use flow networks, when flow goes through an edge, it suffers some loss in quality. We seek some threshold quality at the sink.

Each edge has flow in ($f^i$), flow out ($f^j$), and capacity $c_e$

$$\max \sum \sum_{e \in E} f^e \ s.t.
\sum_{e \in E} f^e \leq c_e
\sum_{e \in E} f^e = d_i

f^e = w(e) \cdot f^{e\in} \quad (*) \text{Quality reduction}
\sum_{u \in V} f^u(w, u) = \sum_{u \in V} f^u(u, e) \quad u \neq s, t. \quad (*) \text{Flow*}$$

---

**Push approach: Inference**

How to proceed when worker’s expertise, motivation & cost are unknown?

- Decision theoretic model
- Exploration-exploitation tradeoff

---

**Decision theoretic model**

- Probability/cost $C(w)$ of a worker to achieve a task
- Utility $U(w)$ of the task (E.g. quality, worker’s satisfaction)

**Goal:** maximize the expected utility.

**Technique:** Proactive learning

(Donmez & Carbonell, 2008) 2 step algorithm:

1. Discovery: Probing worker’s characteristics
2. Task assignment: Choose iteratively task-worker pair maximizing the cost-benefit trade-off.

---

**Exploration-exploitation tradeoff**

At every step the algorithm can explore (information gathering) or exploit (i.e. assign a task).

(Donmez et al., 2009)

- For each action $a$, we record the number of time it has been performed $n_i$ and its success rate $w_i$. 
- At each step, we estimate the confidence interval of the success rate and choose the one with highest upper bound.

Remark: Time changes (experience, fatigue) can be integrated.

---

**Pull approach**

Workers tend to choose tasks in which they have the most expertise, interest & understanding.

- More effective on platform with high turn-over.
- Coverage & completion time.

(Law et al., 2011)

---

**Search & visualization**

Mechanical Turk UI is very primitive. Users constantly refresh the web page to find most recent HITs.

Better classification of HIT using Web Ontology Language.

(Odesk & SciStarter)
Task recommendation

Objective: Match users to set of objects.

Applications:
- Wikipedia.

Challenges in human-computation systems:
- It must satisfy the requester’s needs as well,
- Tasks are typically non-persistent.

Content based methods: Detect similarities between user profile & item characteristics.

Collaborative filtering methods: Use preferences (i.e. ratings) to infer similarities between individuals.

Evaluation of routing

Primary criteria to evaluate the quality of a routing strategy:
- Accuracy
- Discovery
- Efficiency
- Worker motivation

Tentative schedule

Jan 24: Task design I (Understanding workers & requesters)
Jan 29: Paper & project presentation
Jan 31: Task design II (The art of asking questions)
Feb 5: Invited lecture – Doina Precup
Feb 7: Invited lecture – Derek Ruths
Feb 12: Invited lecture – Ali Mohammadi
Feb 14: Scientific games
Feb 19-28: Paper presentations
Winter break!
Mar 12-14: Paper presentations
Mar 19: Invited lecture – Eric Bourget
Mar 21 – Apr 9: Lectures on "Networked Science"