Forming Cohesive Teams of Experts

***Abstract*—Problem of team selection from a group of individuals with a variety of expertize has been of great importance in past and is increasingly becoming important with the ever increasing online collaborations over internet. Operation research community has studied this problem extensively but unfortunately, the previous collaboration history between the experts was never taken into account by them. Recently there are a few papers which try to bring in the cohesion dimension by extracting knowledge from static collaboration network among the various experts. In this work on similar lines we are trying to capture cohesion but rather than predicting a team of individuals we are predicting team of group of experts. Our work models the intuition prevalent in social science literature that rather than dyadic links between individuals it is the various higher-order groups, in which an individual has been been collaborating, important. We therefore, have used Hypergraphs to capture these higher order collabarations. Moreover, rather than combining the cohesion and skill fulfillment as a single optimization task we first perform a node ranking over the original expert hypergraph to come up with both the smaller as well as more relevant smaller hypergraph. Then the task of team selection is performed as a multi-set multi-cover (MM) problem over the extracted smaller hypergraph.**

I. Problem

Given a set of nodes (experts) $V = \{v_1, ..., v_n\}$ we have a hypergraph $HG = \{E, V\}$ where, $E = \{e_1, ..., e_m\}$ is the set of m hyper-edges (collaborations). The incidence matrix $\mathbf{H}$ is a $n \times m$ matrix where $\mathbf{H}(e, v) = 1$ when vertex (expert) v belongs to a hyper-edge (collaboration) e. Each vertex v is associated with expertize vector $\mathbf{p}_v$ of size $(p \times 1)$ where each $\mathbf{p}_v(i, 1)$ is a R value representing the expertize in the i^{th} skill. We therefore have an expertize matrix $\mathbf{P}$ is a $(n \times p)$ matrix where $\mathbf{P}(v, :) = \mathbf{p}_v$. In the problem of team selection we have to find out the most cohesive teams of experts (nodes) which together also fulfills the requirements of the given task. Task is represented as $\mathcal{T} = \{k, \mathbf{q}\}$ where k is the cap on the team size and $\mathbf{q}$ is the requirement expertize vector of size $(p \times 1)$. We can design different aims:

- Our aim is to find out the most cohesive set of vertices $S \subseteq V$ where $|S| \leq k$ which satisfy the expertize requirement $\mathbf{q}$.
- Our aim is to find out the most cohesive set of hyper-edges $S \subseteq E$ where $|S| \leq k$ which satisfy the expertize requirement $\mathbf{q}$. In this case k is taken as the cap on the maximum number of hyper-edges.
- Or we can develop other aims that have no size restrictions etc.

Further, an expertize requirement $\mathbf{q}$ is satisfied by a selection S if,

$$\sum_{v \in S} \mathbf{p}_v(i, 1) = \mathbf{q}(i, 1), \forall i \in \{1, ..., p\}. \tag{1}$$

II. Approaches

The approaches we have in mind are as follows:

- Modeling this problem as a completely combinatorial optimization problem of finding the densest sub-hypergraph problem satisfying the skill constraints.
- We model the cohesiveness by coming up with a ranking of the most cohesive vertices for a given skill requirement. We take some top-k out of this ranked list and use this for the densest sub-hypergraph to come up with the team. We thus, reduce our search space to these top-k.

A. Two-Step Process

The two below sections deal with the two steps that are involved in the two-step process.

1) Filtering the intial Hypergraph: In this step our aim is to come up with the top-k most relevant collaborations (hyperedges) out of the huge intial collaboration hypergraph. The incentive for this is that, rather than hitting this NP-Hard MM problem with the complete graph, can we intelligently come up with the relevant vertices/edges. By relevance here we mean those experts which are either having these expertize or have high collaboration links with such experts or (if we are lucky enough) then both. We model this using label propagation over hypergraph. Given the task $\mathcal{T}$ at hand we intialize each node v_i with the skill labels $\mathbf{y}(i)$ such that $\mathbf{y} = \mathbf{P}\tilde{\mathbf{q}}$ where $\tilde{\mathbf{q}}$ is the normalized $\mathbf{q}$. More formally:

$$\text{y(i)} = \sum_{j \in \{1,....,p\}} \mathbf{P}(i, j)\tilde{\mathbf{q}}(j). \tag{2}$$

In this manner we initialize each vertex with sum of all the required skills that vertex possesses weighted by the relevance of that skill for the given task. This initial label vector $\mathbf{y} = [y_1, y_2, ...y_n]$ takes care of the skill requirements. Next we find out the ranking (labels when stationary state is reached) vector $\mathbf{f} = [f_1, f_2, ..., f_n]$ of other nodes with respect to this reference initial vector. This is accomplished by propagation of these labels over hypergraph $\mathbf{H}$. Hypergraph label propagation was formalized by Zhou et al. [6] by providing a hypergraph combinatorial cut and then relaxing it to a convex real valued problem. We have used it to come up with the ranking (final labels) similar to

:

$$\text{Q(f)} = \frac{1}{2} \sum_{i,j=1}^{n} \sum_{e \in E} \frac{1}{\delta(e)} \sum_{\{v_i, v_j\} \subseteq e} w(e) \left\| \frac{f_i}{\sqrt{d(v_i)}} - \frac{f_j}{\sqrt{d(v_j)}} \right\|^2 + \mu \sum_{i=1}^{n} \|f_i - y_i\|^2 \tag{3}$$

where $\delta(e)$ is the degree of hyperedge e, $w(e)$ is the weight of hyperedge e and $d(v_i)$ is the degree of a vertex v_i which is nothing but the number of edges of which this vertex is a part. Let the optimal ranking is given by $\mathbf{f}^* = \underset{\mathbf{f}}{\operatorname{argmin}} Q(\mathbf{f})$. The above cost function contains two terms of which the second term penalizes if the predicted label is not same as the initial labels assigned. The first term comes from the [6] and makes sure that the random walks happen such that: the any vertices that share many hyperedges (i.e. experts who have strongly collaborated in past) are likely to have similar labels and vertices within a hyperedge have similar labels. Note that the random walk jumps in proportional to the weights of the hyperedges (i.e. strong collaborations) and then chooses another vertex within that hyperedge randomly.

We finally take the top-k ranked vertices and also all the hyperedges of which these vertices are a part of and make this new hypergraph. Note that it is not necessary that all the vertices in this new hypergraph are a part of the top-k vertex list.

The method we adopt is to model the real-world team recruitment process in organizations. The process followed by HR in organizations is that of consulting heads of departments with various major skill required by the task at hand. These heads then recurringly follow the same process with the skill requirement getting more specific at deeper levels. In the end of this process small groups which have worked together frequently and are seemingly quiet adequate for the skill requirement at hand are returned. In other words it is similar to a kind of $rippling$ effect where word of mouth is used to form a team. Therefore, the labels emulate the $ripples$.

2) Multi-set Multi-cover (MM) Step: In this step the skill coverage is taken care of. Some top-K most cohesive hyperedges from the above step (A-2) are taken. Each of them is converted from a set of nodes to a set of skills. For each skill we take sum of the values of of this skill in each node. This now changes to a MM problem as described in Section 2.2 of Barna et al. [7]. MM outputs the minimum number of sets (effectively hyperedges) required to fulfill the skill requirement.

B. Densest Sub-hypergraph Multi-cover (DSHyM)

This integer programming problem captures both the cohesiveness and skill requirement, aspects at once. Let us define variable $x(e)$ such that $x(e) = 1$ if hyperedge e is chosen and 0 otherwise. Also for every vertex v we define a variable $y(v)$ such that $y(v) = 1$ when vertex v is chosen and 0 otherwise. We define the ***DSHyM*** as the following LP:

Objective

$$\max \quad \sum_{e \in E} x(e) \tag{1}$$

Constraints

$$x_e \leq y_v \qquad \forall e \in E, v \in V \tag{2}$$

$$\sum_{v \in V} \mathbf{P}(v,a) y(v) \geq \mathbf{q}(a,1) \qquad \forall a \in \{1,...,p\} \tag{3}$$

$$\sum_{v \in V} y(v) = k \tag{4}$$

where,

$$y(v) \in \{0,1\} \tag{5}$$

$$x(e) \in \{0,1\} \tag{6}$$

$$P(v,a) \in \{0,1\} \tag{7}$$

$$k \text{ is the number of vertices chosen} \tag{8}$$

with a LP relaxation of,

$$y(v) \in [0,1] \tag{9}$$

$$x(e) \in [0,1] \tag{10}$$

$$P(v,a) \in R \tag{11}$$

REFERENCES

[1] Bu, Jiajun and Tan, Shulong and Chen, Chun and Wang, Can and Wu, Hao and Zhang, Lijun and He, Xiaofei, *Music recommendation by unified hypergraph: combining social media information and music content*. In Proceedings of the international conference on Multimedia (MM '10). ACM, New York, NY, USA, 391-400.

[2] TaeHyun Hwang, Ze Tian, Jean-Pierre Kocher, and Rui Kuang. *Learning on Weighted Hypergraphs to Integrate Protein Interactions and Gene Expressions for Cancer Outcome Prediction*,Proc. of Eighth IEEE International Conference on Data Mining (ICDM), pages 293-302, 2008.

[3] Ze Tian, TaeHyun Hwang and Rui Kuang. *A Hypergraph-based Learning Algorithm for Classifying Gene Expression and ArrayCGH Data with Prior Knowledge*, Bioinformatics, Vol. 25, No. 21, pages: 2831-2838, 2009.

[4] Koren, Yehuda. *Collaborative filtering with temporal dynamics.* Communications of the ACM 53.4 (2010): 89-97.

[5] Bezdek, James C., and Richard J. Hathaway. *Some notes on alternating optimization.* Advances in Soft ComputingAFSS 2002. Springer Berlin Heidelberg, 2002. 288-300.

[6] Zhou, Dengyong, Jiayuan Huang, and Bernhard Schlkopf. *Learning with hypergraphs: Clustering, classification, and embedding.* Advances in Neural Information Processing Systems. 2006.

[7] Saha, Barna, and Samir Khuller. "Set cover revisited: hypergraph cover with hard capacities." Automata, Languages, and Programming. Springer Berlin Heidelberg, 2012. 762-773.

Algorithm 1 MM(x^*, α, $\mathbf{M}(=\mathbf{P})$, $\mathbf{r}(=\mathbf{q})$, $\mathcal{S}$, p, η, β)

/* STEP 1*/
$\mathcal{H} = \{s | x^*(s) \geq \alpha\}$, $\bar{\mathbf{r}} = \phi$
for $a \in \{1, ..., p\}$ **do**
  for $S \in \mathcal{H}$ **do**
    $\bar{\mathbf{r}}(a) = \mathbf{r}(a) - \mathbf{M}(idx(S), a)$
  end for
end for

/* STEP 2*/
$\mathcal{C} = \mathcal{S} - \mathcal{H}$
$\mathbf{M^1} =$ getPowOfTwo(M)
$\bar{\mathbf{r}}^{\mathbf{1}} =$ getPowOfTwo($\bar{\mathbf{r}}$)
$\mathbf{y^1} = 4\mathbf{y}$

/* STEP 3*/
for $S \in \mathcal{C}$ **do**
  for $a \in \{1, ..., p\}$ **do**
    if $\mathbf{M^1}(idx(S), a) \geq \bar{\mathbf{r}}^{\mathbf{1}}(a)$ **then**
      $\mathcal{H} = \mathcal{H} \cup S$
    end if
  end for
end for
$C = S - H$
$n = |\mathcal{C}|$

/* Grouping into small and big sets.*/

$B = \phi$, $L = \phi$
for $S \in C$ **do**
  for $a \in \{1, ..., p\}$ **do**
    if $\mathbf{M^1}(idx(S), a) \geq \eta \bar{\mathbf{r}}^{\mathbf{1}}(a)$ **then**
      $B = B \cup S$
    else
      $L = L \cup S$
    end if
  end for
end for
/* Grouping into small and big elements.*/
$l = \phi$, $b = \phi$
for $a \in \{1, ..., p\}$ **do**
  $t = 0$
  for $S \in L$ **do**
    $t = t + \mathbf{M}^1(S, a)$
  end for
  if $t \geq \bar{\mathbf{r}}^{\mathbf{1}}(a)$ **then**
    $l = l \cup a$
  else
    $b = b \cup a$
  end if
end for
$\beta_1 = \beta - 1$

/* STEP 4: Covering small elements.*/
$\mathcal{C}_{small} = randRounding(\gamma \mathbf{y}^1_S, L, l, \bar{\mathbf{r}}^1, \mathbf{M}^1(S, a))$

/* STEP 5: Covering big elements.*/
$\mathcal{C}_{big} = coverBig(\mathbf{y}^1_S, B, b, \bar{\mathbf{r}}^1, \mathbf{M}^1(S, a), \beta^1)$
return $H \cup C_{small} \cup C_{big}$ =0

Algorithm 2 randRounding(γ, $\mathbf{y}^1_S$, $\mathbf{M}^1$, $\mathcal{C}_{small}$, $\bar{\mathbf{r}}^1$, L, l, n)

$C_{small} = \phi$
Find 'c' s.t. $(\frac{1}{e})^{c \log n} \leq \frac{1}{4n}$
for $i \in \{1, ..., n\}$ **do**
  for $j \in \{1, ..., c \log n\}$ **do**
    if 1 == {1 with probability $\gamma \mathbf{y}^1_S$} **then**
      $C_{small} = C_{small} \cup \{S | idx(S) == i\}$
    end if
  end for
end for
$\sum_{S \in C_{small}, a \in l} \mathbf{M}^1(S, a) \mathbf{y}^1_S \geq \bar{\mathbf{r}}^1$ and cost of $C_{small} \leq (4c \log n) OPT_f$

return C_{small}
=0

Algorithm 3 coverBig($\mathbf{y}^1_S$, $\mathbf{M}^1$, C_{small}, $\bar{\mathbf{r}}^1$, B, b, n)

$k = (\ln \ln n + 3)$
$T^a_i = \{S \in B | \mathbf{M}^1 = \frac{\bar{\mathbf{r}}^1}{2^i}\} \forall i \in \{1, ..., k\}$
$C_{big} = \phi$
for $i \in \{1, ..., k\}$ **do**
  $J = \{x | x \in C_{small}, x \in T^a_i\}$
  $R^a_i = \sum_{s \in T^a_i} \mathbf{y}^1_S$
  if $R^a_i > i$ and $|J| < \frac{R^a_i}{\beta_1 - 2}$ **then**
    $C_{big} = C_{big} \cup (\lceil \frac{R^a_i}{\beta_1 - 2} \rceil - J)$
  end if
end for

return C_{small}
=0