Team Formation in Social Networks

Md. Toufikuzzaman (0419052008), Ali Haisam Muhammad Rafid (0419052019) and Makhdum Ahsan Rupak (0419052060)

ARTICLE INFO

Keywords:

ABSTRACT

1. Introduction

Team formation in social network deals with forming a team of socially connected experts to perform a task requiring a set of skills. The team must perform satisfactorily and there should be effective communication among the team members.

In regular team formation problem, there is a given task which requires certain number of members possessing a set of skills. From a set of candidates we have to choose the most suitable ones to meet the requirements. This is similar to the task assignment problem Easterfield (1946) which can be solved in polynomial time Kuhn (2005). But efficacy of the team depends on how the team members interact and communicate with each other McDonald (2003); Wolf, Schröter, Damian, Panjer and Nguyen (2008). Specially, in software product development and knowledge creation, collaboration is the main key. Good communication ensures best utilization of individual skills for achieving the goal. Thus notion of considering social connection in team formation was constructed and solution to these types of problems was first put forward by Lappas, Liu and Terzi (2009).

The core of the solution is a social graph. In this graph nodes represents possible candidates and their set of skills. Whereas edges contain weights inferring social affinity and ability of working together of two candidates. Our target is to find a subset of nodes in the social graph which will ensure that a) chosen candidates have the required skills to perform the task, b) communication cost among them is minimum and c) workload will be balanced as fairly as possible.

2. Motivation

This problem mainly gained familiarity in the scientific community because of the hardness that it poses in solving it at optimality after it was introduced by Lappas et al. (2009). In real world, we can readily assume that there exists a social network communication between individuals of a team. In many team based projects, it is necessary to ensure that the members possess the required skills to fulfill the demands of the projects and also that they collaborate effectively and efficiently. There are many social networks for professionals which can be exploited to form effective teams which meets the required skills. We can also develop a model using the idea of this problem to predict approximately the performance of a team

3. Problem Description

3.1. Notations

Let us assume that we are given a list of n candidates denoted by the set $C = \{1, 2, \dots, n\}$. We are also given a set of all possible skills required to perform all possible tasks under a domain denoted by the set $A = \{a_1, a_2, \dots, a_m\}$. Depending on the domain of the tasks, members of the set A will vary. Each candidate has some skills which is a subset of A. If the skill set of an individual i is C_i, then $C_i \subseteq A$ where $C_i = \{a_j \in A \mid \text{individual } i \text{ has skill } a_j\}$. From the information given in C, A and C_i we can define a support set $S(a_j) \subseteq C$ which is the set of all candidates possessing the skill a_j. In a more formal definition $S(a_j) = \{i \in C \mid a_j \in C_i\}$. Finally the skills required to perform a specific task T is represented by the set $A(T) \subseteq A$.

ORCID(s):

3.2. Defining the Task

Task T is defined by a set of tuples, $T = \{(a_1, k_{a_1}), (a_2, k_{a_2}), \ldots, (a_m, k_{a_m})\}$, where the first element of each tuple is a skill and the second element is the required number of persons with that skill required to complete the task. So for each tuple (a_j, k_{a_j}), $a_j \in A(T)$ and $k_{a_j} \in \mathbb{N}$. For performing a task, a team (C') will be formed where $C' \subseteq C$. As we need k_{a_j} number of members with the skill a_j in the team C', this team must fulfill the requirement $\forall_{a_j \in A(T)}(\mid C' \cap S(a_j) \mid \geq K_{a_j})$. As each member of the team can have multiple skills, team size will be bounded by $\mid C' \mid \leq \sum_{j=1}^{m} k_{a_j}$.

3.3. Social Graph

Social graph $G(C, E)$ is composed of nodes representing each candidates in C and the edges (E) correspond to relation of candidates. Weight (w) of the edges can mean either the gain or the cost of collaboration depending on the implementation. Each node is also associated with a set of skills which the member possesses.

3.4. Problem Formulation

Our target is to find the subgraph which will minimize the collaboration cost while meeting the required skills of a task. We denote this subgraph with $G'(C', E')$ where C' is the set of members of the team for performing the task T and E' contains those edges in E whose nodes are in C'.

4. Related Work

Team formation has been discussed in Operations Research for many years. But team formation in social network was introduce in 2009 by Lappas et al. (2009). In their work, the two communication cost functions were considered based on the subgraph formed from social graph G. They are weight of the minimum spanning tree (MST) and diameter of the subgraph formed by the team. They proposed two algorithms. One is Rarest First Algorithm for the MST based cost function and the other is Enhanced Steiner Algorithm based on diameter of subgraph function.

Based on the work of Lappas et al. (2009), Majumder, Datta and Naidu (2012) proposed a slightly different solution. With the two given cost functions they added Bottleneck Edge cost and incorporated the constraint that each candidate has a capacity so that team members are prevented from being overburdened.

Kargar and An (2011) pointed out the shortcomings of the cost functions presented by Lappas et al. (2009). They showed that tweak in the input graph may result in a completely different solution. In a later literature Kargar proposed a scalarization based bi-objective algothim. The objectives were minimizing communication and personal cost. In scalarization, a budget is given to one objective and the other objective is minimized while maintaining the budget. Their algorithm merged two objectives by getting threshold input from user. As users do not always want to provide the threshold value they also proposed an approximation based pareto set finding algorithm.

Niveditha, Swetha, Poornima and Senthilkumar (2017) explored a tri-objective optimization approach to the problem. In addition to the communication and personal cost in Kargar, An and Zihayat (2012), they tried to minimize the number of members in a team. They argued that merging objective functions like proposed in Kargar et al. (2012) may cause complication of adding more objectives later and the teams formed in this way tend to be larger. They applied Non-dominated Sorting Genetic Algorithm for Team Formation (NSGA-II TF) Deb, Agrawal, Pratap and Meyarivan (2000) to design a robust and extensible framework.

Juárez and Brizuela (2018) formed a multi objective solution. They considered two conflicting objectives, collaborative density and total level of expertise, to solve the problem. In a more recent work Selvarajah, Zadeh, Kargar and Kobti (2019) have proposed a cultural algorithm and tried to minimize communication cost. In their work communication cost was defined as diameter of the graph of sum of distance.

5. Methodology

Overview of our workflow is shown in the block diagram 1. The first two steps are discussed in 6. In problem definition we have defined a task as a set of skills (a_i) where k_{a_i} is the number of experts required to perform the skill. In our preliminary experiment we are setting k_{a_i} to 1 as done in most of the previous works and in the latest work by Selvarajah et al. (2019).

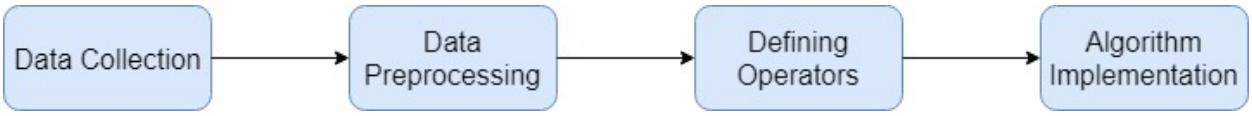

Figure 1: Block diagram of our workflow

5.1. Defining Operators

We have defined a cost function based on previous collaboration history between authors. Our crossover and mutation operators are a slight modification of those used by Juárez and Brizuela (2018). Before getting into explanation of operators first we discuss the initialization function shown in 1.

Algorithm 1 Initialization Function

```
procedure INITIALIZE(Social Graph G, Skill Author Map S_map, Tasks T)
    team ← []
    for skill in T do
        author ← RANDOM AUTHOR FROM S_map(skill)
        team.append(author)
    return team
```

For each skill in the required task a random author is chosen who is an expert at that skill. A team is a collection of these chosen author. By using this initialization method we are ensuring that no infeasible team or in other words team not fulfilling all required skills will occur.

5.1.1. Fitness Assessment

We have defined a cost function (2) based on sum of normalized publication count between two authors. Publication count is normalized by dividing with total number of publications. If there is a collaboration between two authors say A and B, our social graph will contain two edges one from A to B and other from B to A for them. Though number of publications for both edge is same their normalized publication count can be different as both the authors have different total number of publications. The intuition behind this assumption was that if we let total publication count dominate the solution space, teams may end up with authors who have higher number of publications only. Our assumption is expected to maintain diversity by repelling this effect. As we are forming a minimization function, for each pair of authors we subtract the normalized edge weight from 1. If there is not an edge between two authors the wight is 0. We can examine the two sample teams shown in 2. Here in the team at the right side all the members know each other. So from the perspective of collaboration density this team is more preferable. But in previous works like Selvarajah et al. (2019) they formulated the cost function as sum of distances between all authors. As the graph in the left side has two edges it will have lower sum of distances than the team at the right side. Our fitness function penalizes the nodes who have no edge between them and hence push for teams with more number of interlinks and dense network. Also cost functions like sum of distances require the calculation of all pair shortest path which is a computationally intensive task and it tends to be infeasible as number of nodes in the graph increases. But the simple and easy to calculate fitness function that we have proposed does not suffer from this problem yet expected to produce better collaborative teams.

Algorithm 2 Fitness Function

```
procedure ASSESS FITNESS(Social Graph G, Individual I)
    score ← 0.0
    for all pair of authors A, B in I do
        score ← score + 1 − G(A, B)
        score ← score + 1 − G(B, A)
    return score
```

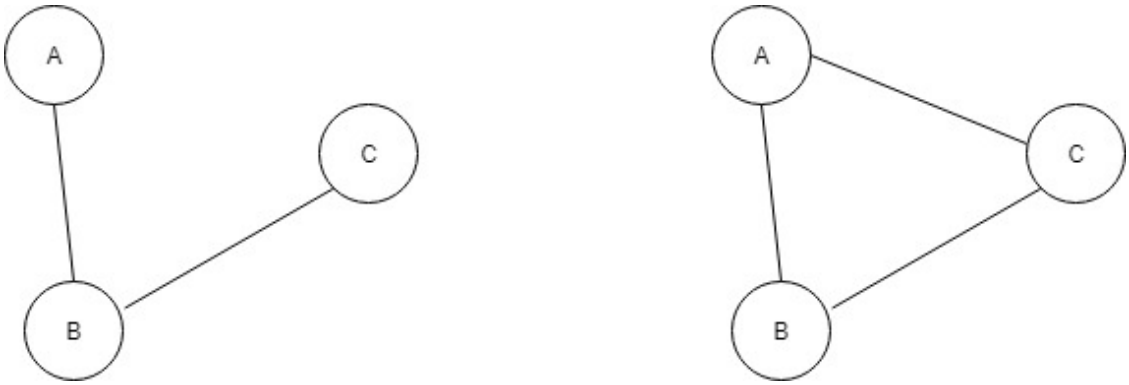

Figure 2: Two sample teams with edges indicating collaboration between the authors

5.1.2. Mutation

In mutation (3), first a random author from the team is removed. Then to make sure that the team is a feasible solution we check if remaining authors have all the skills required to perform the task. If they satisfy the condition, mutation phase ends and returns the new team. Otherwise an author is assigned for one of the skills that the authors of the team do not satisfy. In this selection process a random author is selected from the co-authors of current team who possess that skill. If no such co-author is found, a random author with that skill is added in the team. This process is repeated until authors are assigned for all the skills in the task.

Algorithm 3 Mutation Function

procedure MUTATE(Social Graph G, Skill Author Map S_{map}, Individual I, Task T)
    $I_{new} \leftarrow copy(I)$
    REMOVE RANDOM AUTHOR FROM I_{new}
    while There is a skill in T that is not satisfied by I_{new} **do**
        $authors \leftarrow S_{map}(skill) \bigcap coauthors(I_{new})$
        if $authors$ is not empty **then**
            $author \leftarrow$ RANDOM AUTHOR CHOSEN FROM $authors$
        else
            $author \leftarrow$ RANDOM AUTHOR CHOSEN FROM $S_{map}(skill)$
        $I_{new}.append(author)$
    return I_{new}

5.1.3. Crossover

Crossover operator gets two individuals as input. The individuals are selected using tournament selection. Two children are produced from the crossover operation. For each skill in the task an author is selected from each individuals and inserted in one of the children teams. A coin is tossed for each skill. If the toss returns head candidate author from individual 1 and individual 2 are inserted in child 1 and child 2 respectively. In case of tail child 1 gets author from the candidates of individual 2 and child 2 gets from individual 1. The whole process is shown in 4.

5.2. Algorithms

Using the operator defined in 5.1 we ran our experiment with four algorithms to minimize our defined fitness function. As the algorithms are already well known, we are not providing their details. The algorithms are following:

- **Random Search:** Selecting best team from a set of randomly initialized teams.
- **Hill Climbing:** Selecting a random team and tweaking it with mutation operator to get better team.

Algorithm 4 Crossover Function

```
procedure CROSSOVER(Individual 1 I_1, Individual 2 I_2, Task T)
    C_1 ← []
    C_2 ← []
    for each skill s in T do
        if toss returns HEAD then
            C_1.append(A RANDOM AUTHOR WITH SKILL s FROM I_1)
            C_2.append(A RANDOM AUTHOR WITH SKILL s FROM I_2)
        else
            C_1.append(A RANDOM AUTHOR WITH SKILL s FROM I_2)
            C_2.append(A RANDOM AUTHOR WITH SKILL s FROM I_1)
    return C_1, C_2
```

- **Steepest Ascent Hill Climbing:** Sampling the neighbourhood of a team with mutation operator to find better team.
- **Steepest Ascent Hill Climbing with Replacement** Similar to steepest ascent hill climbing. Additional feature is current team is always replaced with the best neighbour.
- **Genetic Algorithm:** Searching for best team in a population of teams which gets replaced in each generation by the children.
- **Elitist Genetic Algorithm:** Similar to genetic algorithm but portion of elite parents reach the next generation with children.

6. Results

6.1. Dataset

We have used the DBLP[1] dataset in our experiment which have been used in most of the previous works. It is a computer science bibliography dataset which contains bibliographic information on major journals and proceedings. This is collection or raw data in xml format. To make this data useful for our purpose some preprocessing steps are required. It is a very large dataset. But due to our time constraint in this preliminary result we have use a smaller preprocessed version of the dataset provided by Wang, Zhao and Ng (2015). The preprocessed dataset extracted bibliographic information about four major computer science topics Database (DB), Theory (T), Data Mining (DM) and Artificial Intelligence (AI) from the raw data. The considered conferences are: DB = SIGMOD, VLDB, ICDE, ICDT, PODS, T = SODA,FOCS, STOC,STACS, ICALP, ESA, DM=WWW, KDD, SDM, PKDD,ICDM, WSDM, AI = IJCAI, NIPS, ICML, COLT, UAI, CVPR. Finally they generated two csv files. In the co-author file each row represents an author. For each author total number of publications, name of the coauthors and number of publications with the co-authors are given. In the skill file each row contains author name and the set of skills the author possess. We generated social graph from these files. The dataset contains information about only 7159 authors. To prevent the graph from being too sparse and disconnected we only considered those skilled with frequency greater than 5. There were 1350 such skills.

6.2. Algorithm Parameters

In our preliminary study we have experimented with six algorithms mentioned in 5.2. All the algorithms were executed for 1000 iterations. The parameters of these algorithms are described in the following.

- **Random Search**
    1. Iteration: 1000
- **Hill Climbing**

[1] https://dblp.uni-trier.de/xml/

Algorithm	Average Team Size	Average Objective Value
Random Search	5.17	28.070
Hill Climbing	5.02	20.726
Steepest Ascent Hill Climbing	5.11	20.477
Steepest Ascent Hill Climbing with Replacement	4.88	19.312
Genetic Algorithm	4.65	17.306
Elitist Genetic Algorithm	4.46	16.103

Table 1
Performance comparison of algorithms

1. Iteration: 1000

- **Steepest Ascent Hill Climbing**
  1. Iteration: 1000
  2. Samples in Neighbourhood: 100
- **Steepest Ascent Hill Climbing with Replacement**
  1. Iteration: 1000
  2. Samples in Neighbourhood: 100
- **Genetic Algorithm**
  1. Iteration: 1000
  2. Population size: 100
  3. Tournament size: 7
- **Elitist Genetic Algorithm**
  1. Iteration: 1000
  2. Population size: 100
  3. Tournament size: 7
  4. Number of elites: 20

6.3. Experimental Setup

All the experiments were conducted on a workstation with the properties mentioned below.

- **Operating System:** Windows 10 Pro 64-bit (10.0, Build 10240)
- **Processor:** Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz (4 CPUs), ~2.2GHz
- **Memory:** 16384MB
- **Python Version:** 3.6

6.4. Performance

We used two metrics to evaluate performance. One metric was the fitness function and the other was team size. To produce this preliminary result we conducted experiments for 100 tasks where each task require 6 skills. These skills were chosen at random from the set of skills. Performance metrics were reported as the average result from the optimization result of 100 tasks. Table 1 compares performance of different algorithms.

All the codes and resources of our experiment are availabel at GitHub[2].

[2] https://github.com/tzpranto/Team-Formation

7. Discussions

Results obtained from our experiment suggest that genetic algorithms show significant improvement in performance. Introduction of elitism to make the algorithm more exploitative has made slight improvement in performance. As our dataset is relatively smaller, we are not drawing the conclusion that elitist genetic algorithm is the better choice. Our dataset produces sparse social graph because it contains only around 7000 nodes in social graph compared to almost 200k nodes in the paper of Selvarajah et al. (2019). Even in our sparse dataset fitness function has directed the solutions towards finding teams with more previous collaborations. So we are assuming our fitness function will perform better on a dense graph.

Due to the unavailability of dataset and code used by most recent literature we could not perform a head to head comparison. The dataset we have, lacks enough information to reproduce the results of Juárez and Brizuela (2018) and Selvarajah et al. (2019). We could not prepare similar dataset from raw DBLP data due to time constraint. Also the authors did not want to share their codes so we have built all the functions from scratch. As our current dataset does not contain enough information, we could not apply multi objective algorithms.

References

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, in: International conference on parallel problem solving from nature, Springer. pp. 849–858.

Easterfield, T.E., 1946. A combinatorial algorithm. Journal of the London Mathematical Society 1, 219–226.

Juárez, J., Brizuela, C.A., 2018. A multi-objective formulation of the team formation problem in social networks: preliminary results, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM. pp. 261–268.

Kargar, M., An, A., 2011. Discovering top-k teams of experts with/without a leader in social networks, in: Proceedings of the 20th ACM international conference on Information and knowledge management, ACM. pp. 985–994.

Kargar, M., An, A., Zihayat, M., 2012. Efficient bi-objective team formation in social networks, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 483–498.

Kuhn, H.W., 2005. The hungarian method for the assignment problem. Naval Research Logistics (NRL) 52, 7–21.

Lappas, T., Liu, K., Terzi, E., 2009. Finding a team of experts in social networks, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 467–476.

Majumder, A., Datta, S., Naidu, K., 2012. Capacitated team formation problem on social networks, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 1005–1013.

McDonald, D.W., 2003. Recommending collaboration with social networks: a comparative evaluation, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM. pp. 593–600.

Niveditha, M., Swetha, G., Poornima, U., Senthilkumar, R., 2017. A genetic approach for tri-objective optimization in team formation, in: 2016 Eighth International Conference on Advanced Computing (ICoAC), IEEE. pp. 123–130.

Selvarajah, K., Zadeh, P.M., Kargar, M., Kobti, Z., 2019. Identifying a team of experts in social networks using a cultural algorithm. Procedia Computer Science 151, 477–484.

Wang, X., Zhao, Z., Ng, W., 2015. A comparative study of team formation in social networks, in: International conference on database systems for advanced applications, Springer. pp. 389–404.

Wolf, T., Schröter, A., Damian, D., Panjer, L.D., Nguyen, T.H., 2008. Mining task-based social networks to explore collaboration in software teams. IEEE Software 26, 58–66.