International Conference on Web Intelligence : Thessaloniki, Greece — October 14 - 17, 2019. ACM, New York, NY, USA, 293-297.
ISBN: 978-1-4503-6934-3.
http://dx.doi.org/10.1145/3350546.3352533

Fair Team Recommendations for Multidisciplinary Projects

Lucas Machado
Kostas Stefanidis
lucas.machado@tuni.fi
konstantinos.stefanidis@tuni.fi
Tampere University, Finland

ABSTRACT

The focus of this work is on the problem of team recommendations, in which teams have multidisciplinary requirements and team members' selection is based on the match of their skills and the requirements. When assembling multiple teams there is also a challenge of allocating the best members in a fair way between the teams. We formally define the problem and propose a brute force and a faster heuristic method as solutions to create team recommendations to multidisciplinary projects. Furthermore, to increase the fairness between the recommended teams, the *K-rounds* and *Pairs-rounds* methods are proposed as variations of the heuristic approach. Several different test scenarios are executed to analyze and compare the effectiveness of these methods.

1 INTRODUCTION

Recommendation algorithms for team formation aim to assemble teams of individuals based on some specified criteria. For team formation, we need to extract and identify individual characteristics of the individuals, topics from documents and perform analysis and visualization of relation graphs [3]. The individuals are then grouped together based on how near they are from each other in a relations graph (implicit relation identification) or by expert finding [7]. While it is a difficult task to assemble the "best" team, due to the several different subjective factors that could define a team as best, a decision support system such as a recommender system may help on that [3]. Furthermore, most of the papers describe team formation in software development context [16], or based on users common interests and attributes [1, 4].

Thinking about a team as a package of items in which its members have skills that correspond to individual items attributes, a multidisciplinary team is a package of items with diversity in their attributes. Complex tasks often demand multidisciplinary teams, and an increase of a team output could be achieved through selecting members with specific skills to maximize that output. Forming such teams requires aligning people with different skills and backgrounds and should also consider people that are not similar as a possible good choice, while recommender systems are usually based on similarities as an indicator of good alignment [8, 13]. In addition, the concept of diversity is positive in a multidisciplinary team context [6, 12] and may also trigger serendipity [11].

Other challenges may apply to this multidisciplinary team formation problem, e.g., when a team member is restricted to work for only one project. If several teams are being formed, all the best candidates could be assigned to the first team, leaving the remaining less suitable candidates for the other projects. Thus, the *fairness* aspect [9, 10] of this team formation should be also taken into account, in a way that good members could be assigned to all teams.

In this work, our inspiration comes from a real world problem in which multidisciplinary teams need to be formed and allocated to work on different projects with requirements for members' skills and some constraints. We pay special attention to fairness when multiple teams need to be formed. That is, we focus on *"how to create team recommendations based on members' skills and projects requirements, while ensuring fairness?"*. Recommender systems could be very helpful in multidisciplinary team formation problems for projects, bringing several benefits. Particularly when there are thousands of candidates and dozens of teams to be formed, the amount of needed human labor can be significantly reduced due to systematic analysis of candidates. In addition, it is a difficult task to maintain the fairness aspect between teams manually, a problem which that recommender system algorithm can solve.

2 PROBLEM DEFINITION

2.1 Motivating example

Assume that there are several projects that aim to create products and satisfy needs. Each one of them has different needs of skilled people based on their requirements, restrictions, context and goals. For example, a project on developing a new website for a company would require individuals with skills of front-end development, design of interfaces, and user experience. However, a project aimed at creating a device for measuring heart rate would need individuals with expertise in health sciences, engineering and ergonomics.

The individuals that could work on a project possess different sets of skills. The ability or expertise to do something well is defined as a *skill*. A *project* is a collaborative effort to reach a goal, which is carefully designed and planned [14] and that requires a team of people with specific skills for that.

This work investigates a team formation problem in the context of a platform in which several different projects are available to

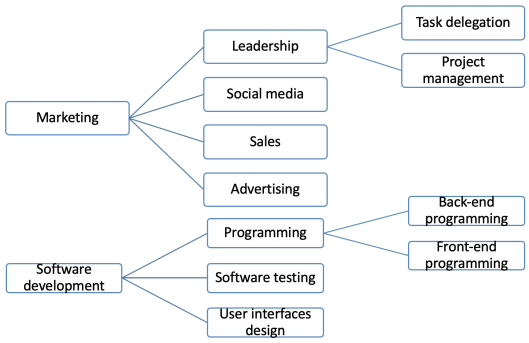

Figure 1: Example of a set of skills in a hierarchy

receive applications from interested individuals (applicants) to work on them. For all projects, a team should be formed by matching the project requirements with applicants' skills. Furthermore, each project has a determined number of team members required.

The teams should be formed in a way that maximizes matches between project requirements and applicants' skills. A perfectly maximized but unrealistic team formation would be when for every applicant in the team, the applicant possesses all of the project required skills. However, a given applicant cannot belong to more than one team, which poses a restriction since forming a team limits the available choices of applicants for other teams. Therefore, when forming all the teams, some fairness is required in such a way that all teams are similarly good and choices are not made purely on finding the best possible applicants for a project, leaving other projects with the remaining less suitable applicants.

2.2 Model

Skills, which might be attributes of applicants or project requirements, are represented by textual tags and relate to each other in a graph relationship. Figure 1 shows an example of a set of skills and their relations. In this example, skills are represented by nodes which are connected by edges.

Skills are not only attributes of applicants, but also requirements for the projects. Let P be a set of projects, in which each project p, $p \in P$, is described by a set of required skills $p = \{r_1, \ldots, r_n\}$ that the team members of the project must possess, in which r_i is a textual tag representing a skill. It can be assumed that all projects have the same amount of required skills.

Example 2.1 (Projects p_1 and p_2 and their sets of required skills).
p_1 = {programming, sales, user interfaces, design}
p_2 = {programming, user experience, marketing, advertising}

To determine if an applicant a has the skill r that is needed in the project p, or if the applicant has any skills related to r within its skill set, the function $scoreAR(a, r)$ is used. The function returns a score of how well an applicant has skills that relate to a specific required skill of a project. The $similarity$ function returns a score in the interval $[0, 1]$, depending on how related the skills are: $scoreAR(a, r) = \sum_{\forall s_i \in a} similarity(s_i, r)$.

Knowing how well an applicant's skills suit one project requirement, it is possible to calculate how well an applicant fits in a project in consideration with all of the project's required skills. The function $scoreAP(a, p)$ is used to calculate the matches between the set of skills of an applicant a and all the required skills of a project p, based on the function $scoreAR(a, r)$: $scoreAP(a, p) = \sum_{\forall r_i \in p} scoreAR(a, r_i)$.

Furthermore, with the information of how well applicants could fit within a project, the function $scoreTP(t, p)$ denotes how well a team of k applicants t, $t \subset A$, matches with all the required skills of a project p. By using this function it is possible to compare how well different team formations fit to a project, according to Definition 2.2 below. It is assumed that all teams are formed with the same amount of applicants k: $scoreTP(t, p) = \sum_{\forall a_i \in t} scoreAP(a_i, p)$.

Definition 2.2. Given a project $p = \{r_1, \ldots, r_n\}$, and a set of applicants $A = \{a_1, \ldots, a_m\}$, where each applicant a_i is associated with a set of skills $\{s_{i_1}, \ldots, s_{i_x}\}$, the best team of k applicants for the project p is the team T^* for which: $T^* = argmax_{|T|=k} scoreTP(T, p)$, such that, $\forall r_j \in p$, $\exists a_i \in T$, with $s_{i_y} = r_j$; and there are at least k applicants in the set A.

Therefore, to form q teams of k members, the set A must contain at least $q \times k$ applicants.

The model presented above differs from the methods in research literature in the sense that teams are not formed based on similarities between its members as in content-based methods, neither on past ratings as in collaborative filtering approaches. The concept of ratings itself is not used in a traditional way, but replaced by the relation between required skills of projects and applicants' skills, which for this problem would be better called *scores*. Since the relations between applicants' skills — items' attributes — are not taken into consideration, content-based filtering is not suitable. Furthermore, recommended teams are also not calculated based on historic data of past formed teams. Projects are often unique and rarely the assumption that there are two or more projects with the same requirements can be made. Hence, collaborative filtering would also be unfit for this problem, since the sparsity problem would be taken to an extreme in which the subset of rated items for a user would consist of at most one item.

Nonetheless, the model seems to fit better within knowledge-based approaches. It could be related to constraint-based knowledge-based systems, as the projects possess requirements for the desired applicants' skills. Furthermore, knowledge-based methods have as their strengths the ability to work well with sparsity, complex and specific problems, which is the case presented by this work.

Moreover, the concept of group recommendations to individuals is used. The formation of a team involves calculating the score of applicants for a project, then combining them into the team, based on the requirements (criteria) set by the projects.

2.3 Fairness-aware Team Formation

It is not sufficient to find the best team for a project using the Definition 2.2, since the assignment of applicants to a specific team makes them unavailable to other teams. Therefore, the property of fairness needs to be applied when suggesting multiple teams to multiple projects, so that there is a balance between the teams.

Definition 2.3. **Fairness to teams:** Let $\mathcal{T}$ be a set of n teams $(T_1, \ldots, T_n)$, assigned to a set $\mathcal{P}$ of projects $(p_1, \ldots, p_n)$. Given a

set $\mathcal{TS}$ including all pairs of teams $(T_i, T_j) \in \mathcal{T}$, to ensure fairness in group formation, minimize: $\sum_{(T_i,T_j)\in\mathcal{T}} |scoreTP(T_i, p_i) - scoreTP(T_j, p_j)|$.

The implication of applying Definition 2.3 is that the best possible team is not always going to be chosen for some projects. However, by choosing teams with slightly lower scores (*scoreTP*) for some projects, it is possible to choose teams with greater scores for others, thus minimizing the differences and increasing the fairness between them. The implementation of fairness in this work is novel relating to the research literature, as most of the approaches are implemented by reducing the variation between predicted ratings, i.e. by increasing similarity between items. On the other way, this work proposes to achieve fairness through the way in which teams are formed, considering the presented restrictions of the context.

3 METHODS

3.1 Brute force algorithm

Assume that $combinations(k, L)$ is a function that calculates the binomial coefficient (generates a list of all possible combinations) of k elements from the set of elements L. In our context k is the amount of applicants in a team, and L is the set of all applicants. Moreover, the function $scoreAP(a, p)$ calculates how well a given applicant a is suited to a given project p, based on the model in Section 2. Our brute force algorithm uses the $combinations(k, L)$ and $scoreAP(a, p)$ functions to implement a brute force method to generate the best teams recommendations. For every project p in the set of projects P, all the possible team combinations T of k members are generated from the set of available applicants A. Then for every possible team t in T, its score relating to the project p is calculated with the function $scoreTP(t, p)$. The team with the maximum score is chosen as the best team for that project and its members are removed from the set of available applicants A. This team formation process is repeated for all projects.

3.2 Heuristic algorithm

The brute force approach is computationally expensive, due to the calculation of all possible team combinations. Therefore, a heuristic which could be applied to minimize the computations while keeping the recommendation efficacy is proposed. Instead of generating all possible team combinations, our heuristic first calculates $scoreAP(a, p)$ between every applicant a in the set A and every project p in the set of projects P. These values are stored as a set in the *projectApplicantsScores* variable. Then the applicant a who had the maximum *scoreAP* for a given project p is chosen as a member of that project team and is removed from the set of available applicants A. This previous step is repeated k times until the project p team has all its k members chosen. This team formation process is then repeated for all other projects of the set P.

Furthermore, this algorithm could be optimized to improve fairness according to Definition 2.3. That optimization is specified in the following two novel variants $V1$ and $V2$, which implement a *k-rounds choosing* method to generate more fair teams recommendations. Specifically, instead of choosing all the k best applicants as members of a project team and then repeating the process for other project teams, $V1$ chooses only one applicant a who had the best calculated *scoreAP* for every project p as a member of that project team, and also removes it from the set of available applicants A. This procedure happens in k rounds to add the *k-nth-member* until all the teams have k members. Similarly, $V2$ implements a variation *pairs-rounds choosing* method to form teams. Based on the calculations of $scoreAP(a, p)$ between every applicant a in the set A and every project p in the set of projects P, there are $k/2$ rounds in which the pair of applicants a_1 and a_2 who had the best values of *scoreAP* for p are assigned as team members and removed from the set of available applicants A. Again, this process is repeated until all the projects have teams of k members. If k is an odd number, then during the last round only one team member will be assigned.

4 EXPERIMENTAL EVALUATION

4.1 Dataset

A pre-processed DBLP dataset (Wang et al. [15]), consisting of a CSV file with 7428 lines, is used for testing. Each line corresponds to a researcher, and contains the person's name and a varying number of skill tags related to that person. Each researcher has at least one skill and there are 4480 unique skills among all people. Due to the nature of DBLP, the skills associated with the researchers correspond to keywords used in those researchers publications. Truthfully, it may not represent the same definition of skills used in this work. However, since the skills derived from the DBLP dataset represent an area of knowledge or expertise in which a person published research, it could be considered as a sufficient approximation. To assemble the hierarchy relationship between the skills, Wordnet [5] was used.

4.2 Measurements

Since the *scoreTP* value is an indicator of how well a team fits into a project requirements, the analysis of the success of the recommendations is focused on it. The sum of all the *scoreTP* values in a set of recommended teams indicate how successful the recommendation method is, relative to its parameters (amount of projects, amount of required skills by project and amount of members in each team). Therefore, this measurement is taken into account as it conveys s better quality and quantity of matches between applicants' skills and project requirements. Furthermore, based on the *scoreTP* values, a *fairness-deviation* indicator is proposed to measure the fairness digression between recommended teams. Assuming that an absolute fair set of teams would be a set in which all of the teams have the same *scoreTP*, the fairness-deviation indicates how much in average the teams deviated from this absolute fair situation. The fairness-deviation between a set of recommended teams $\mathcal{T}$ is defined by: $\text{fairness-deviation}(\mathcal{T}) = \frac{\sum_{\forall t_i \in \mathcal{T}} \mid t_i - \text{mean}(\mathcal{T}) \mid}{\text{len}(\mathcal{T})}$, where $\text{mean}(\mathcal{T})$ is the mean of the *scoreTP* values of the teams in the set $\mathcal{T}$, and $\text{len}(\mathcal{T})$ is the count (length) of teams in the set.

4.3 Methods

A test scenario receives as input p projects and a set of 20 randomly sampled skills that could be used as project requirements. This set of projects together with the parameter k of how many members each team should have are subsequently used as input to the algorithms. The results returned by the algorithms are measured and

recorded. In all experiments, we use the full set of 7428 applicants as input to the algorithms. We target at comparing the fairness-related results between the algorithms. From the range of numbers between 5 and 35 a sample of 10 numbers was selected to be used as p values (amount of projects). Then from the range between 3 and 12 a sample of 5 numbers was used as k values (amount of members in each team). These range values were chosen trying to represent the extreme situations in a real scenario, as Agile teams, for example, usually have less than 10 members [2]. All the possible combinations between the 10 sampled p and 5 sampled k values are used to generate a total of 50 test scenarios. Those 50 scenarios are executed receiving as input only the 200 most frequent skills as possible project requirements. Likewise, another 50 test scenarios are created and executed with another samples of p and k values following the same constraints, however using the 200 less frequent out of the 2000 most frequent skills to create project requirements. The behaviour of the algorithms during the extreme situations of overfitting and underfitting of data can be analyzed with this variation between the most and less frequent skills.

4.4 Effectiveness

The recommendations are evaluated by how well the team members of the team recommendations adhere to the required skills of the projects, represented by the sum of *scoreTP* values. In addition, they are also evaluated by how fair the teams are in the context of a set of recommended teams $\mathcal{T}$. The fairness-deviation indicator is used for that. Figure 2 shows the sum of all the *scoreTP* values over the amount of choices made in a set of recommended teams $\mathcal{T}$. The amount of choices made refer to the amount of team members in each team multiplied by the amount of project teams ($k \times \text{len}(\mathcal{T})$). By applying linear regression to this data (as indicated by the lines in the figure), it is possible to notice that the *Pairs-rounds* method achieves slightly better results in overall than the other methods, while the *K-rounds* method seems to improve when the amount of choices made increase. However, the results of the three algorithms are very similar and in practice their differences could be considered negligible. Due to the two variations of skill sets used to generate project requirements, it is also possible to notice two linear areas of concentrated points in Figure 2. More frequent skills in project requirements have a bigger chance to find more similarities with applicants' skills, thus increasing the overall value of *scoreTP*.

Figure 3 shows the fairness-deviation values for all executions, over the *amount of choices made*. With linear regression analysis on this data (as indicated by the lines in the figure), it is observed that the *K-rounds* and *Pairs-rounds* methods produce significantly more fair results than the simple heuristic method without fairness optimization. The brute force and heuristic methods not only produced less fair results, but also had a lower sum of the *scoreTP* values. This occurred because first a full team was formed and its members were made unavailable for other teams.

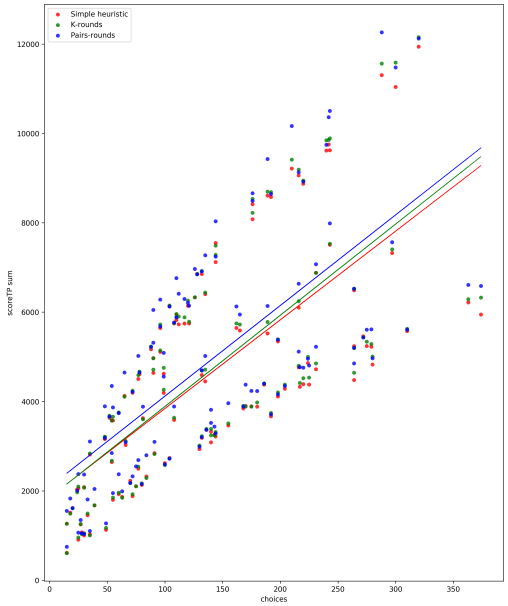

Figure 2: Sum of scoreTP values for different algorithms

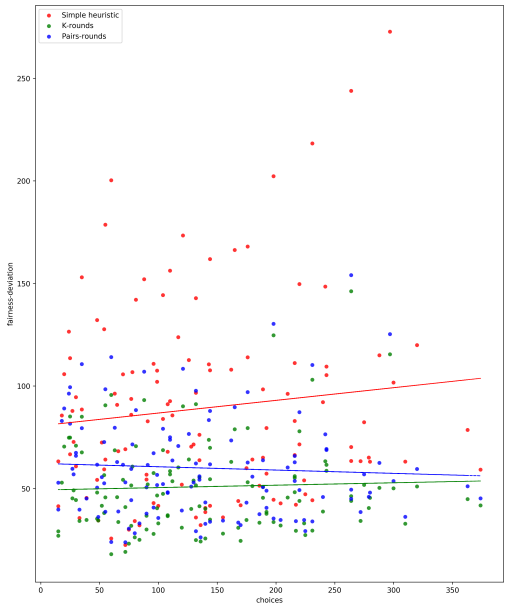

Figure 3: Fairness-deviation for different algorithms

5 SUMMARY

In this work, we focus on the problem of fair team recommendations. We formally define the problem and propose algorithms as solutions to create team recommendations to multidisciplinary projects. The experimental evaluation shows that the proposed heuristic-based methods are able to create team recommendations for multidisciplinary projects in a very successful way.

ACKNOWLEDGMENT

This work has been partially supported by the Virpa D project funded by Business Finland.

REFERENCES

[1] Gaganmeet Kaur Awal and K. K. Bharadwaj. 2014. Team formation in social networks based on collective intelligence - an evolutionary approach. *Applied Intelligence* 41, 2 (9 2014), 627–648.

[2] Denise Canty. 2015. *Agile for Project Managers*. Auerbach Publications. 135 pages. https://doi.org/10.1201/b18052

[3] Anwitaman Datta, Jackson Tan Teck Yong, and Stefano Braghin. 2014. The zen of multidisciplinary team recommendation. *Journal of the Association for Information Science and Technology* 65, 12 (12 2014), 2518–2533.

[4] Christoph Dorn, Florian Skopik, Daniel Schall, and Schahram Dustdar. 2011. Interaction mining and skill-dependent recommendations for multi-objective team composition. *Data & Knowledge Engineering* 70, 10 (10 2011), 866–891.

[5] Christiane Fellbaum (Ed.). 1998. *WordNet: an electronic lexical database*. MIT Press.

[6] Marialena Kyriakidi, Kostas Stefanidis, and Yannis E. Ioannidis. 2017. On Achieving Diversity in Recommender Systems. In *ExploreDB*. 4:1–4:6.

[7] Ching Yung Lin, Nan Cao, Shi Xia Liu, Spiros Papadimitriou, Jimeng Sun, and Xifeng Yan. 2009. SmallBlue: Social network analysis for expertise search and collective intelligence. In *ICDE*.

[8] Ioanna Lykourentzou, Robert E. Kraut, and Steven P. Dow. 2017. Team Dating Leads to Better Online Ad Hoc Collaborations. In *CSCW*.

[9] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. 2012. Fast Group Recommendations by Applying User Clustering. In *ER*. 126–140.

[10] Eirini Ntoutsi, Kostas Stefanidis, Katharina Rausch, and Hans-Peter Kriegel. 2014. "Strength Lies in Differences": Diversifying Friends for Recommendations through Subspace Clustering. In *CIKM*. 729–738.

[11] Violina Ratcheva. 2007. Redefining multidisciplinary team boundaries in resolving heterogeneous knowledge dilemmas. *International Journal of Intelligent Enterprise* 1, 1 (2007), 81.

[12] Kostas Stefanidis, Marina Drosou, and Evaggelia Pitoura. 2010. PerK: personalized keyword search in relational databases through preferences. In *EDBT*. 585–596.

[13] Kostas Stefanidis and Evaggelia Pitoura. 2013. Finding the Right Set of Users: Generalized Constraints for Group Recommendations. *CoRR* abs/1302.6580 (2013). http://arxiv.org/abs/1302.6580

[14] A Stevenson. 2010. *Oxford Dictionary of English* (3rd edition ed.). Oxford University Press. https://doi.org/10.1093/acref/9780199571123.001.0001

[15] Xinyu Wang, Zhou Zhao, and Wilfred Ng. 2015. A Comparative Study of Team Formation in Social Networks. In *DASFAA*.

[16] Murat Yilmaz, Ali Al-Taei, Rory V O'Connor, and Rory V. O'Connor. 2015. A Machine-Based Personality Oriented Team Recommender for Software Development Organizations. In *Systems, Software and Services Process Improvement*.