8-31-2020

Team formation using recommendation systems

Shreyas Patil
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

TEAM FORMATION USING RECOMMENDATION SYSTEMS

by
Shreyas Patil

The importance of team formation has been realized since ages, but finding the most effective team out of the available human resources is a problem that persists to the date. Having members with complementary skills, along with a few must-have behavioral traits, such as trust and collaborativeness among the team members are the key ingredients behind team synergy and performance. This thesis designs and implements two different algorithms for the team formation problem using ideas adapted from the recommender systems literature. One of the proposed solutions uses the Glicko-2 rating system to rate the employees' skills which can easily separate the skill ability and experience of the employees. The final contribution of this thesis is to build a system with "plug-in" capability, meaning any new recommendation algorithm could be easily plugged in inside the system. Our extensive experimental analyses explore nuances of data sources, data storage methodologies, as well as characteristics of different recommendation algorithms with rating and ranking sub-systems.

TEAM FORMATION USING RECOMMENDATION SYSTEMS

by
Shreyas Patil

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Data Science

Department of Computer Science

August 2020

APPROVAL PAGE

TEAM FORMATION USING RECOMMENDATION SYSTEMS

Shreyas Patil

Dr. Craig Gotsman, Thesis Advisor Date
Dean and Distinguished Professor, Ying Wu College of Computing, New Jersey Institute of Technology, Newark, NJ

Dr. James Geller, Committee Member Date
Professor and Former Chair, Department of Computer Science, New Jersey Institute of Technology, Newark, NJ

Dr. Senjuti Basu Roy, Committee Member Date
Assistant Professor, Department of Computer Science, New Jersey Institute of Technology, Newark, NJ

BIOGRAPHICAL SKETCH

Author: Shreyas Patil

Degree: Master of Science

Date: August 2020

Undergraduate and Graduate Education:

- Bachelor of Engineering in Computer Engineering,
  University of Mumbai, 2016
- Master of Science in Data Science,
  New Jersey Institute of Technology, Newark, NJ, 2020

Major: Data Science

I dedicate this work to my Dad, Mr. Sudhakar Patil, who fought against the odds to get me to this position; to my Mom, Mrs. Jyoti Patil, who wouldn't mind staying hungry as long as my tummy is full; and to my girlfriend & future wife, Ms. Nikita Patil, who would always be supportive and optimistic no matter what. I know how erratic I can sometimes be, but thank you all for always being by my side even if my behavior may hurt you. I am sorry for all the sufferings I caused you. I love you.

Shreyas Patil

ACKNOWLEDGMENT

I thank Dr. Craig Gotsman for pulling me out of a critical situation and helping me complete my work. I also thank him for being my advisor and giving me his precious time. I want to thank Dr. Suresh U. Kumar for being the first to believe that my concept is practically implementable and for sharing his knowledge and expertise. I thank him for pointing out my mistakes whenever needed and being kind enough to keep forgiving me for the mistakes I kept making again and again. I also thank Dr. Senjuti Basu Roy for pointing towards the right way when I was stuck on what to do next and for sharing her research related to the topic. I thank Dr. James Geller for a brief discussion over the existing solutions, techniques, and the red flags of such systems, which helped me look through a different point of view. I commend all my friends who gave me valuable inputs and critics. At last, but no the least, I thank God for giving me this chance to serve the world in the way I can and getting me through this literature and research.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There are many companies that are still not aware of the art of team formation. The project managers usually form the teams greedily, which may affect an employee or the company overall. The organizations that care about their profit optimization think of team formation as the first thing to work on.

Not only the employees but the organization too benefits by optimizing the team. Employees benefit by getting hassle-free work culture and the organization by saving time, money, and unnecessary efforts and complications among the employees. Current solutions do not focus on these factors and making the best decision regarding team formation for the specific scenario.

The chapter two covers all the related works in the same research space or the ones related to it which includes general recommendation systems, Glicko and Glicko-2 rating systems, and the variants of solutions to the team formation problem.

The work hereon discusses the making of an integrative framework using a recommendation system for team formation. Chapter three covers the problem definition and preparation for implementation. The preparation includes the technologies and server configurations used, data sources, and the data storage structures. Since a NoSQL document-based database is used, the data storage structures may be called document structures in this literature.

Chapter four describes various approaches of a recommendation system application and the way a recommendation system is designed for this system. The implementation of a recommendation system includes a rating system and then a scoring system that is used to rank and recommend the employees for a particular role in the project.

The fifth chapter discusses the use of the recommendation system discussed in chapter four to suggest Top-N employees for each role in a project. A team is then formed for each type of team requested by the project manager by selecting an employee from the Top-N for each role. This section also explains the criteria that each employee has to pass through to be selected.

Chapter six is the last chapter and considers a few other approaches for implementing a recommendation system and matching the employees to form a team. It also covers the project's future scope and extensions along with the challenges the solutions to the team formation problem face. Finally, it concludes the implementation literature for team formation using a recommendation system.

CHAPTER 2

RELATED WORK

2.1 Recommendation Systems

Author in the book [1] discusses the various types of recommendation systems along with their goals. According to the author in [1], the basic principle of recommendations is that significant dependencies exist between user and item-centric activity. The author also explains the ways in which a recommendation problem can be formulated among which the primary ones are Prediction version of a problem and Ranking version of a problem. The prediction version predicts a rating value for a user-item combination. This problem is also referred to as the matrix completion problem because the missing values in an incompletely specified matrix of m users and n items are predicted by the learning algorithm. On the other hand, in the Ranking version, the prediction is not necessary. Instead, the top-k items could be recommended to a user or determine the top-k users to target a particular item. This problem is also referred to as the top-k recommendation problem.

The author in the book [1] has mentioned a numerous real-world recommendation systems such as Amazon.com to recommend books and other products, Netflix to recommend DVDs and streaming videos, Jester to recommend jokes, GroupLens to recommend news, MovieLens to recommend movies, last.fm to recommend music, Google News for news, Google Search for advertisements, Facebook recommends friends and advertisements, Pandora for music, YouTube for online videos, Tripadvisor for travel products, and IMDb for movies.

2.2 Glicko and Glicko-2 Rating Systems

In early 1960's, Arpad Elo developed the Elo rating system which was the first chess rating system that had probabilistic underpinnings. It was adopted by saveral chess federations and in many other games but it has some problems. Hence, in 1995 author of [25] created the Glicko rating system and later the Glicko-2 rating system.

The problem with the Elo rating system that the Glicko and Glicko-2 rating systems address to is related to the reliability of a player's rating. For example, suppose two players, both rated 1700, played a tournament where the first player defeats the second, according to the US Chess Federation's version of the Elo system, the first player would gain 16 points and the other player will lose the same. But suppose that the first player returned to the game after many years, which makes his rating of 1700 more unreliable rather that the second player's rating who plays every weekend. Hence, according to the author in [25], when the first player defeated the second player, the first player's rating must increase more than 16 points as it is clear that the first player's skill level is superior to the second player who has a more precise rating of 1700. And on the other hand, the second player should lose less that 16 points for the same reason.

To overcome this flaw, author in [25] extends the Elo rating system in the Glicko rating system by computing not only a rating, which is the best guess of one's skill level, but also a rating deviation(RD), which measures the uncertainty in the rating where high RD corresponds to unreliable ratings. The author in [25] further introduces a rating volatility σ in the Glicko-2 rating system which is a measure of the degree of expected fluctuation in a player's rating.

2.3 Variations in Team Formation Problem

The social networks have been the point of interest for the team formation researchers for a while now. Researchers find the social networks as a good starting point for their research on online or offline team formation problems. Some chose the generalized social networks or some tried to use a specific social network that is focusing on grouping a particular type of individuals. Either way they require a pre-built social community as their starting point. Most of these researchers use approximation algorithms to find a optimal team out of the network and then address or solve the specific issues related to team formation problem. While some authors focus on the networks such as in [31, 3, 10, 29, 16, 30, 19, 28,] to find the team fit for a task, others focus on the issues related to the team formation problem itself such as seen in [2].

Authors in [31] believe their work to be the first to consider the team formation problem in the presence of a social network of individuals. They tried to find a group from a pool of individuals with different skills to perform a task given a social network that captures he compatibility between these individuals. The authors in [31] try to form a team that not only meet the requirements of a task but also perform effectively by measuring the effectiveness using the communication cost among the team members.

Each team formation problem has a primary goal to find a best group of individuals who meet the requirements of a task while considering a few other important factors that come along the problem such as fairness, workload balancing, employee compatibility, communication and cost. Authors in [2] focus on forming a team of people fit for the task in a fair way where no-one in the team is overloaded or singled out. They recognized the trade-off between the individual workload and the team size while forming the team.

Authors in [3] claim this to be the first paper to address the problem of team

formation with a solution where all the required skills are covered by the team with small communication overhead and load balancing together. They tried to solve the problems addressed by the authors from [31, 2] together to find a group of individuals from a social network that are capable of performing the given task effectively. While keeping the primary goal of the team formation intact and using a social network, the authors in [10] introduced a team member's capacity in a team formation problem while trying to balance the workload among the team members.

The authors in [29] consider the social network as a graph where a node is an expert with the weight representing the cost for using the expert and an edge between the nodes represents the communication between two experts where the weight is the communication cost. They aim to find a team of experts that can cover all the skills required for performing a task while minimizing both the expert and communication cost for a project. One specific application of the team formation problem was discussed by the authors in [16] where they formulated finding influential event organizers as the problem of mining influential cover set to find a team of organizers in social network that have the required skills to organize an event while motivating more individuals to participate in the event.

An another bi-criteria optimization problem similar to [29] proposed by authors in [30] is to find a team of experts from an expert network that covers all the required skills for a given task while minimizing the communication and personnel cost of the team. The first approach they took to solve this problem is by using the (α, β)-approximation algorithms, where one objective is considered one after another and the other one is to find a set of pareto optimal teams. One another attempt to minimize the communication cost while forming top-k teams is made by authors in [28] but with one little addition that is to find a leader for a team.

The authors in [12] noticed that the expert-centric properties such as skill are insufficient to assemble an effective team and focus on the balance between skills and

collaboration instead. They address the team composition problem which consists of expert interaction network extraction, skill profile creation, and ultimately team formation in their work.

Similar to [29], authors in [19] consider the social network as a graph where each node is an expert in one or more skills and the edge weights specific affinity or collaborative compatibility between respective nodes. Their goal is to identify a team to maximize the collaborative compatibility which is measured as the density of the induced sub-graph on selected nodes.

A unique application of the team formation was implemented by authors in [22] who focus on multi-agent complex networks of autonomous yet interdependent agents such as supply chain and sensor networks. They aim to find a group of agents who must coordinate effectively in order to solve problems and achieve collective goals.

Other works based on social networks are discussed by the respective authors in [7, 21, 38, 14, 45]. Similarly a few experiments on team formations using optimization algorithms were performed by respective authors in [6, 39, 18, 43, 46, 13, 35]. Authors in [4] discuss a few algorithms for a fair team composition in an online labour marketplace who focus mainly on the fairness aspect of the team formation problem.

One other study was described by authors in [26] where they conducted an experiment on 530 participants who used a team formation system to assemble project teams. They describe how users' traits and social networks influence their teammate searches and choices and ultimately team compositions. They found out how the final results differ from what the users originally searched and how users' decisions lead to noon-diverse and segregated teams.

The few other works related to team recommendation systems are explained in [20, 45, 9, 36] by the respective authors where they discuss the aspects like the expertise, team formation with members from multiple disciplines, and the effects of team recommendation systems.

CHAPTER 3

PROBLEM DEFINITION AND PREPARATION

The problem of team formation has persisted for ages. Many organizations, such as MITRE and Atlassian, have been working on their team formation strategies. Though there has been a surge in research related to team performance and active formation, there is no such solution that has all the features and considers all the parameters related to a team and individual employee skills. Also, no solution can evaluate and measure an employee's actual skill level.

Having a team that closely matches the requirements makes a big difference for a company in terms of cost, time, and progress. It is also beneficial for the employees working in such a team as they may get overwhelmed or underutilized otherwise. Such systems also help the smooth growth of the employees and the organization overall.

The problem is to find the best suitable employee $e \in \mathcal{E}$ out of the available ones for each role $\mathcal{R}$ in a project $\mathcal{P}$ to form a complete team $\mathcal{T}$. This research aims to implement a team formation system that measures each skill for an employee individually and then recommends the employees with the required skill levels for a project. It also considers the team members' compatibility and updates the skill rating after the project is complete.

3.1 Technologies Used

3.1.1 Database: MongoDB

Database selection is a vital task as it may cause trouble later if chosen without thinking, especially for the projects with high scaling possibilities. Due to the unstructured nature of data in this project, a relational database is not a good option. For projects like these, a NoSQL database is a proper choice. Among all the NoSQL

options, MongoDB is perfectly suited for projects like this and fulfills all their needs.

The factors considered when selecting MongoDB as the database for this project are:

1. MongoDB is a schema-less database that means the way a developer writes the code defines the schema.

2. It derives a document-based data model that stores its data in the Binary-JSON (BSON) format. It helps to store data in a natural and simple manner.

3. This efficient data storage helps simplify the otherwise complex relational join queries. Unlike SQL, the document query language plays a vital role in supporting dynamic queries.

4. Since it is a NoSQL database, SQL injection is not possible hence making it more secure.

5. One of its key features is Sharding, which allows it to store the data on various machines or a cluster. This feature makes MongoDB easy to scale horizontally.

There are many other pros of MongoDB, but the factors discussed here are sufficient to choose this database for this system which are among the many discussed by the author in [41].

3.1.2 Backend Programming: Python

Python is the most widely used language for data science projects. Other factors to be considered are its simplicity, a large number of libraries like NumPy, Pandas, SciPy, PyPlot, PyMC that make data handling, processing, and visualization very easy. It also has high integration capabilities with other programming languages like Java, C, and C++. Python also supports functional, procedural, object-oriented, and aspect-oriented programming approaches. All these benefits make it a sound programming language for this work.

3.1.3 Web Framework: Flask

Flask is a micro and lightweight web development framework based on python. Being lightweight, it is speedy when it comes to a small web application like this implementation. It is also well-suited for API management, where the backend python processes and fulfills an API request from the browser. Flask uses Jinja2 as a template engine to integrate the data received from the server into the HTML code. Flask is the best way to convert a small python script into a small, user interactable, web application.

3.1.4 Frontend: Bootstrap/JQuery/AJAX

Bootstrap is a popular framework for building responsive and mobile-friendly websites. It is a light and elegant front-end development framework. Once a user interface is built using bootstrap, JQuery is used to create events and make changes to DOM as required. On the other hand, AJAX stands for Asynchronous Javascript and XML, which handles the data management without reloading the page. AJAX is triggered either on page load or when an event like a click, change, or hover occurs.

3.1.5 Server Config: MongoDB Atlas, PythonAnywhere

1. MongoDB Atlas Data Cluster
    (a) Server: AWS M0 Sandbox
    (b) Location: N. Virginia (us-east-1)
    (c) RAM: 512MB, Shared
    (d) vCPU: 1, Shared
    (e) Cluster Nodes: 3 Replica Set (Sharding)
    (f) Database Capacity: 100 databases
    (g) Collection Capacity: 500 collections (all databases inclusive)

(h) Connection Capacity: 500 database connections

2. PythonAnywhere Flask Web App Server

   (a) Python: v3.8

   (b) Web workers: 3

   (c) CPU Allowance: 4000 seconds

   (d) Disk Space: 5GB

3.2 Data

3.2.1 Data Sources

Datasets of jobs at Google and Amazon, the jobs listed on Indeed can be found on Kaggle, and a dataset of software developer jobs in the USA found on data.world gave an idea of skills by groups and roles. Google and Amazon datasets consist of the job descriptions collected from various listing sources. The job titles related to software, web, and database development are considered for further filtration. The descriptions include responsibilities, soft skills, and technical skills. The technical skills are needed for this project. Similarly, the Indeed dataset includes the skills required for the job roles related to data science projects. The data.world dataset is not formatted correctly and contains repeated data; hence it is of little use. The next section lists the data collected from all these datasets and their descriptions.

3.2.2 Data Collection

The data in the table below consists of the skill groups, skills and a few roles collected from the datasets mentioned in the previous section. Since, the scope of the implementation is still limited to a few projects types and roles, the skill groups and skills are limited to be collected. And because this data is sufficient enough

for further user data generation by random combination of skills, it was collected manually for the sake of limiting the scope of research and implementation.

Table 3.1 Data Filtered and Collected from the Datasets

Data Type	*Data Collected*
Skill Groups	Front-end designing, front-end development, database development, scripting, software development, mobile application development, and data science
Skills	HTML, CSS, Visual Design, Material Design, UX, Print Design, Adobe Photoshop, Adobe Illustrator, CoralDraw, Adobe InDesign, Affinity Designer, Sketch, Javascript, JQuery, Wordpress, Responsive Design, AngularJS, Bootstrap, Git, SQL, MySQL, MongoDB, XML, JSON, PostgreSQL, Redis, MariaDB, Elastic Search, SQLLite, ETL, Shell, Bash, NodeJS, Ruby, Python, Perl, C, CPP, CSharp, Java, Typescript, PHP, Android SDK, Kotlin, Android Studio, XCode, iOS API, Swift, Core Data, Linux, Machine Learning, R, Data Mining, Data Analysis, Natural Language Processing, Computer Vision, TensorFlow, Tableau, Hive, Spark, Hadoop, AWS, GCP, Azure, SAS
Roles	UI/UX Designer, Front-end Developer, Database Developer, Backend Developer, Software Developer, Android Developer, iOS Developer, Data Scientist, Full-Stack Developer

3.3 Data Storage

This section consists of document storage structures for skills, groups, roles, projects, and employees. The data is stored in BSON format which resembles closely with the format shown in figures below. the square brackets indicate that the value is an array.

Note: The data shown in this section is just the snapshot of what the stored data looks like and is subject to change in the course of the implementation. Also the data parameters are discussed and used further in the implementation sections throughout the document.

3.3.1 Skill Groups and Skills

A skill may belong to one to many skill groups. For example, in the following figure, skill Python belongs to both Scripting and Data Science skill groups. Whereas, there may be a skill that does not belong to any group yet.

```
{"skill-groups":[
    {"id": 1, "title": "Data Science"},
    {"id": 2, "title": "Scripting"}
 ]
}

{"skills":[
    {"id": 1,
     "title": "Python"
     "groups":[
        {"id": 1},
        {"id": 2}
     ]
    },
    {"id": 2,
     "title": "Shell"
     "groups":[
        {"id": 2}
     ]
    },
    {"id": 3,
     "title": "Machine Learning"
     "groups":[
        {"id": 1}
     ]
    }
 ]
}
```

3.3.2 Roles

Each role consists of skills with their respective required level and weight. The level can be further converted into rating and rating deviation with rating as median and rating deviation as 75.

```
{"roles":[
    {"id": 1,
     "title": "Data Scientist"
     "skills":[
        {"id": 1,
         "level": 2,
         "weight": 0.4
        },
        {"id": 3,
         "level": 3,
         "weight": 0.5
        },
        {"id": 2,
         "level": 1,
         "weight": 0.1
        }
     ]
    }
 ]
}
```

Figure 3.1 Document structure for roles.

3.3.3 System Variables

The system variable store a few global values that will be explained and used throughout the implementation.

```
{"system":[
    {"max-workload": 1},
    {"tau": 0.5},
    {"min-skill-levels": 0.5}
    {"enable-employee-duplication": false}
 ]
}
```

Figure 3.2 Document structure for system variables.

3.3.4 Projects

```
{"projects":[
    {"id": 1,
     "title": "Weather Prediction Web App"
     "roles":[
        {"id": 1,
         "weight": 0.4
        },
        {"id": 2,
         "weight": 0.3
        }
     ],
     "team":[
        {"emp-id": 1,
         "role-id": 1,
         "skills":[
            {"id": 1, "rating": 1500, "rd": 80},
            {"id": 3, "rating": 1650, "rd": 75},
            {"id": 2, "rating": 1350, "rd": 95}
         ]
        },
        {"emp-id": 2,
         "role-id": 2,
         "skills":[
            {"id": 2, "rating": 1500, "rd": 85},
            {"id": 4, "rating": 1700, "rd": 70},
            {"id": 5, "rating": 1450, "rd": 100}
         ]
        }
     ],
     "template": 0
    }
 ]
}
```

Figure 3.3 Document structure for projects.

3.3.5 Employees

```
{"employees":[
    {"id": 1,
     "name": "John Doe"
     "skills":[
        {"id": 1, "rating": 1500, "rd": 80},
        {"id": 3, "rating": 1650, "rd": 75},
        {"id": 2, "rating": 1350, "rd": 95}
     ]
     "projects":[
        {"id": 1,
         "role-id": 1,
         "skills":[
            {"id": 1, "rating": 1500, "rd": 80},
            {"id": 3, "rating": 1650, "rd": 75},
            {"id": 2, "rating": 1350, "rd": 95}
         ],
         "team":[
            {"emp-id": 2, "role-id": 2}
         ],
         "status": 0
        }
     ],
     "relations":[
        {"emp-id": 2,
         "role-id": 1,
         "rating": 3.6
        }
     ]
    }
 ]
}
```

Figure 3.4 Document structure for employees.

3.4 Implementation

3.4.1 Preparation

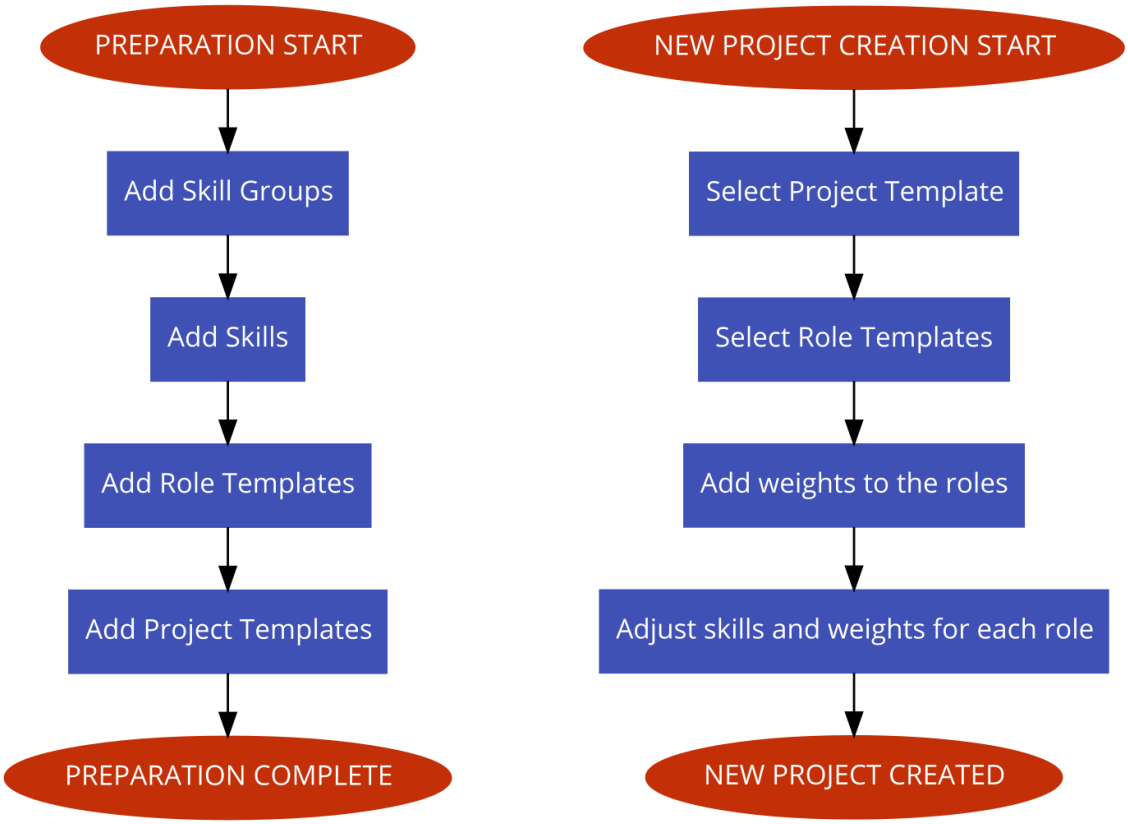

Figure 3.5 Prerequisite data for team formation.

CHAPTER 4

RECOMMENDATION SYSTEMS

As the name suggests, a recommendation system outputs one or many items, given the user preferences and criteria as the inputs. For example, on an online e-commerce website, user's search and order history are used to recommend the relevant items in the hope of attracting the user and eventually increasing sales. This basic use case works just fine with most of the small online e-commerce retailers. An example of an advanced recommendation system is Netflix's movie recommendation system which takes personalization to the next level. This chapter covers the fundamentals of a recommendation system. Author in [1] covers most of them in great detail while authors in the book [32] discuss the application of recommender systems in the technology enhanced learning (TEL) domain.

4.1 Approaches

4.1.1 Collaborative Filtering

It assumes that people who have similar preferences are likely to continue having similar choices in the future. This approach uses rating information of items from the users and predicts the rating of an unexplored item for a user based on similar users' preferences. This approach is used by Netflix to find related users.

4.1.2 Content-based Filtering

Solely based on items' description and user's preferences, this approach is best suited for user-specific item recommendations considering the user history. As mentioned earlier, a well-known example of this approach is Netflix's movie recommendation system.

4.1.3 Multi-criteria Recommendation System

This thesis focuses partially on this type of recommendation system, specifically, the subclass called multi-criteria single-rating recommendation system. This way, multiple item factors, as per the required criteria, are modeled into a single rating value and generating recommendations accordingly. The authors in [23] has defined the recommendation problem as a multi-criteria decision making (MCDM) problem and discuss the techniques that provide recommendations by modelling a user-item interaction as a vector of ratings along several criteria along with the algorithms that user these ratings for predictions and recommendations.

4.1.4 Hybrid Recommendation System

A common approach for most modern recommendation systems because it has the power to combine other strategies like collaborative filtering and content-based filtering using hybridization techniques. A few examples of these techniques are weighting, switching, cascading, feature combination, and feature augmentation. The one this thesis uses is weighting with the multi-criteria single-rating recommendation system.

4.1.5 Others

There exist many other approaches that include a risk-aware recommendation system, a mobile recommendation system, a group recommendation system as per author in [5], a knowledge-based recommendation system, and a critiquing-based recommendation system. Authors in [15] the three major recommendation approaches which includes a knowledge-based recommendation system.

4.2 Roles, Skills and Skill Groups

As discussed earlier, during the preparation, each employee has a set of skills where each skill may or may not belong to one to many skill groups. Similarly, each project has roles where each role is required to have a specific skill set to complete the project.

Let $\mathcal{S} = \{s_1, s_2, s_3, \ldots, s_n\}$ be a set of skills and $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_m\}$ be a set of roles in a project, where role skill set $\mathcal{R} \subset \mathcal{S}$.

The goal here is to create a recommendation system to recommend top-N employees for each job role in the project. The algorithm is completed in the implementation section of the next chapter.

Input: Project Roles $\mathcal{P}$ and index i.

Output: Top-N employees for role $\mathcal{R}_i$.

4.3 Skill Rating

An employee's current skill level is estimated and stored as a numerical value called skill rating. That means every employee has their skills accompanied by a score that determines their experience in the respective skill. This work uses the Glicko-2 rating system as per explained by author in [25] to calculate this numerical value that estimates the change in the employee skill level. Other such rating systems are Glicko (old version of Glicko-2) and Elo. Primarily, the Glicko-2 rating system is used to evaluate a player's skill level in chess and other sports like tennis. One application other than sport of a Glicko-based algorithm is measuring in-course learning as discussed by the author in [37].

4.3.1 Introducing Glicko-2 Rating System

The Elo, Glicko, or Glicko-2 rating systems are common in chess or other sports tournaments. The general principle here is that every participant has a score, and this rating is updated every one or a series of matches. This number of matches is called the rating period. The rating update is massive if the outcome is unexpected.

For example, if a novice player defeats an experienced one, the novice player sees a substantial increase while the experienced player sees a significant decrease in ratings. On the other hand, if the obvious has to happen, that is, the veteran player defeats the novice one, the rating updates for both the players are minimal.

4.3.2 Selection of Scale

This implementation uses the original Glicko scale instead of the Glicko-2 scale as it is convenient to use. The ratings are easy to convert back and forth among the two scales.

The Glicko-2 system assumes that the individual score is roughly constant during a tournament as the players encounter each other. The complete competition is the rating period when the matches keep accumulating, and then the score is updated. However, the score may change for the employees working on multiple projects at a time, setting the rating period to a single project by default, which means every project completion updates the members' skill ratings. But termination of an employee as a project member can trigger this update as well.

The Glicko-2 system has three possible outcomes per match, and the system accepts the match outcome as a numerical value - 1 for a win, 0.5 for a draw, and 0 for a loss. This work uses numerical values as it is where - 1 is for project completion; 0.5 for the project or employee termination for valid positive reasons like voluntary employee resignation or management decided to stop the project; 0 for the project or employee termination due to poor performance. As there is a minimal possibility of the latter two to happen, that is the project or employee termination for any reason; the updates must be a little slower. Also, the skill rating initialization with a lower score than the original Glicko-2 system recommends is required.

There are two other employee attributes, a rating deviation RD and a rating volatility σ and a system constraint τ according to the Glicko-2 rating system. The

rating volatility σ indicates the degree of expected fluctuation in an employee's rating, whereas the system constraint τ is the change in volatility over time. The value of σ for each skill of each employee and τ must be as small as 0.05 and 0.2, respectively.

The original Glicko-2 system recommends the player rating initialization to be 1500 and the rating deviation to be 350. However, this work initializes the skill rating for each skill of each inexperienced employee to be 1200 and RD to be 150. Lower RD initialization is because there is less possibility of uncertainty in the skill ratings given the reason discussed earlier. A rating of 1200 and RD of 150 implies that the system is 95% confident that the actual employee skill rating falls between skill rating minus twice RD and skill rating plus twice RD that is 900 and 1500. Rating Deviation RD is also the measure of experience of the employee in that particular skill. As an employee keeps on completing the projects, the skill RD keeps on decreasing. Hence, a lower RD represents a higher experience in that particular skill.

Employee skills are divided into four categories: Beginner, Intermediate, Advanced, and Expert. For employees new to this system, an initial value according to the class, the recruiter seems the employee fits in, is assigned. It is highly not recommended to appoint an expert level to a recruit. Let the system take its course to move the employee to that level. As for those who are migrating to another company using this system, the employees can carry their scores.

This work identifies the project skills difficulty in several categories: Beginner, Intermediate, Advanced, Expert, and all those that fall between the adjacent categories.

4.3.3 Glicko-2 Rating System for Skill Level Estimation

The Glicko-2 rating system comprises of eight steps.

Step 1. Start with initializing τ to 0.2 permanently across the whole system.

(a) If an employee skill is unrated, set the rating r to 1200, rating deviation RD to 150, and volatility σ to 0.05.

(b) Else, use the employee's most recent skill rating, rating deviation, and volatility.

Step 2. Convert the rating r and rating deviation RD of both, the employee skill and the required skill, to Glicko-2 scale rating μ and rating deviation ϕ using the following formulae.

$$\mu = (r - 1500)/173.7178 \tag{4.1}$$

$$\phi = RD/173.7178 \tag{4.2}$$

The value of employee skill volatility σ remains the same. Let the Glicko-2 skill rating for the employee be μ_e and required skill be μ_s. Similarly, let the Glicko-2 rating deviation for the employee be ϕ_e and that for the required skill be ϕ_s.

Step 3. Compute the estimated variance υ of the employee's skill rating.

$$g(\phi_s) = \frac{1}{\sqrt{1 + 3\phi_s^2/\pi^2}} \tag{4.3}$$

$$E(\mu_e, \mu_s, \phi_s) = \frac{1}{1 + \exp(-g(\phi_s)(\mu_e - \mu_s))} \tag{4.4}$$

$$\upsilon = [g(\phi_s)^2 E(\mu_e, \mu_s, \phi_s)\{1 - E(\mu_e, \mu_s, \phi_s)\}]^{-1} \tag{4.5}$$

Step 4. Compute the estimated improvement Δ in the employee's skill rating. As per the events to update the skill rating discussed earlier, the event value e is used here. For example, project completion event sets e = 1.

$$\Delta = \upsilon(g(\phi_s)\{e - E(\mu_e, \mu_s, \phi_s)\}) \tag{4.6}$$

Step 5. An iterative computation is required to determine the new value of volatility σ'.

1. Let $a = \ln(\sigma^2)$, a convergence tolerance variable ε be as small as 0.000001, and

$$f(x) = \frac{\exp(x)(\Delta^2 - \phi_e^2 - \upsilon - \exp(x))}{2(\phi_e^2 + \upsilon + \exp(x))^2} - \frac{x - a}{\tau^2} \tag{4.7}$$

2. Prepare the initial values before starting the iterations.

   - Set $A = a$
   - If $\Delta^2 > \phi_e^2 + \upsilon$, then set $B = \ln(\Delta^2 - \phi_e^2 - \upsilon)$.

     Else,

     (a) Let $k = 1$

     (b) While $f(a - kr) < 0$,

     $k+ = 1$

     (c) Set $B = a - kr$

3. Let $f_A = f(A)$ and $f_B = f(B)$.

4. While $|B - A| > \varepsilon$,

   (a) Let $C = A + (A - B)f_A/(f_B - f_A)$, and $f_C = f(C)$.

   (b) If $f_C f_B < 0$, then $A \leftarrow B$ and $f_A \leftarrow f_B$

   Else $f_A \leftarrow f_A/2$.

   (c) Set $B \leftarrow C$ and $f_B \leftarrow f_C$.

5. Set $\sigma' \leftarrow \exp(A/2)$.

Step 6. The rating deviation to the new pre-rating period value ϕ_e^*

$$\phi_e^* = \sqrt{\phi_e^2 + \sigma\prime 2} \tag{4.8}$$

Step 7. Compute the new values of employee's skill rating as μ'_e and rating deviation ϕ'_e.

$$phi'_e = 1/\sqrt{\frac{1}{\phi^{*2}_e} + \frac{1}{\upsilon}} \tag{4.9}$$

$$\mu'_e = \mu_e + \phi'^2_e g(\phi_s)\{e - E(\mu_e, \mu_s, \phi_s)\} \tag{4.10}$$

Step 8. Convert the employee's skill rating μ'_e and rating deviation ϕ'_e back to original scale.

$$r'_e = 173.7178\mu'_e + 1500 \tag{4.11}$$

$$RD'_e = 173.7178\phi'_e \tag{4.12}$$

4.4 Finding the Employees with Required Skills

As discussed earlier in the preparation section, each project has job roles, and each job role requires a particular skillset. But there is more to it than just that. In a skillset, each skill needs to have a skill level and a weight.

4.4.1 Finding the Employees

Let $\mathcal{E} = \{e_1, e_2, e_3, \ldots, e_k\}$ be a set of all the employees working in a domain in an organization and $\mathcal{R} = \{s_1, s_2, s_3, \ldots, s_l\}$ be the set of required skills for a job role. The following steps determine the employees that are eligible to be ranked:

Step 1: Finding employees with required Skill s.

$$\mathcal{E}_s \subseteq \mathcal{E} \tag{4.13}$$

Step 2: After finding employees for each required skill in a job role, find the intersection among them to get the employees with all the required skills.

$$\mathcal{E}_{\mathcal{R}} \bigcap_{i=1}^{l} \mathcal{E}_{s,i} \tag{4.14}$$

Step 3: It's time to check the employees for their workload. Let e_{el} be the workload for an employee and wl be the maximum load decided by the administrator. if $e_{i,el} < wl$, $\forall e \exists \mathcal{E}_{\mathcal{R}}$,

$$\mathcal{E}_{\mathcal{R},load} \subseteq \mathcal{E}_{\mathcal{R}} \tag{4.15}$$

Step 4: Now that all the employees with the required skills are collected, generate a number e_{nl} for each employee. This number e_{nl} indicates the score for required skill levels the employee satisfies. Let $a(e, s, l)$ show if the employees e has required skill s within level l and returns 1 if it does, otherwise, returns 0.

$$e_{nl} = \sum^{s \in \mathcal{R}} w_s a(e, s, l) \tag{4.16}$$

For example, out of five skills $\{s_1, s_2, s_3, s_4, s_5\}$ with weights $\{0.2, 0.3, 0.1, 0.10.3\}$, if an employee e_1 has four required skills $\{s_1, s_2, s_3, s_4\}$ within the respective required skill levels and employee e_2 has four required skills $\{s_1, s_2, s_3, s_5\}$, then $e_{1,nl}$ is $\{0.2 \times 1 + 0.3 \times 1 + 0.1 \times 1 + 0.1 \times 1 + 0.3 \times 0\}$ that is equal to 0.7 and similarly $e_{2,nl}$ is 0.9. Hence, even though both the employees have same number required skills within the required skills, the nl score varies depending on the skill weights. Let every employee in set $\mathcal{E}_{\mathcal{R}}$ carry e_{nl} as $e_{i,nl}$ for employee i. Let $\mathcal{E}_{\mathcal{R},nl}$ be a set of all e_{nl} values for $\mathcal{E}_{\mathcal{R}}$. The goal is to select the employees with the highest two numbers.

Let $\mathcal{E}_{\mathcal{R},load,levels}$ be a set of N employees with highest nl scores from set $\mathcal{E}_{\mathcal{R},load}$.

$$\mathcal{E}_{\mathcal{R},load,levels} \subseteq \mathcal{E}_{\mathcal{R},load} \tag{4.17}$$

Hence $\mathcal{E}_{\mathcal{R},load,levels}$ consists of Top-N employees with the required skills and their respective levels required job role and has some workload available. And for the purpose of simplicity, let $\mathcal{E}_{\mathcal{R},load,levels}$ be $\mathcal{E}_{\mathcal{R}}$ for job role $\mathcal{R}$ again from hereon.

$$\mathcal{E}_{\mathcal{R}} = \mathcal{E}_{\mathcal{R},load,levels} \tag{4.18}$$

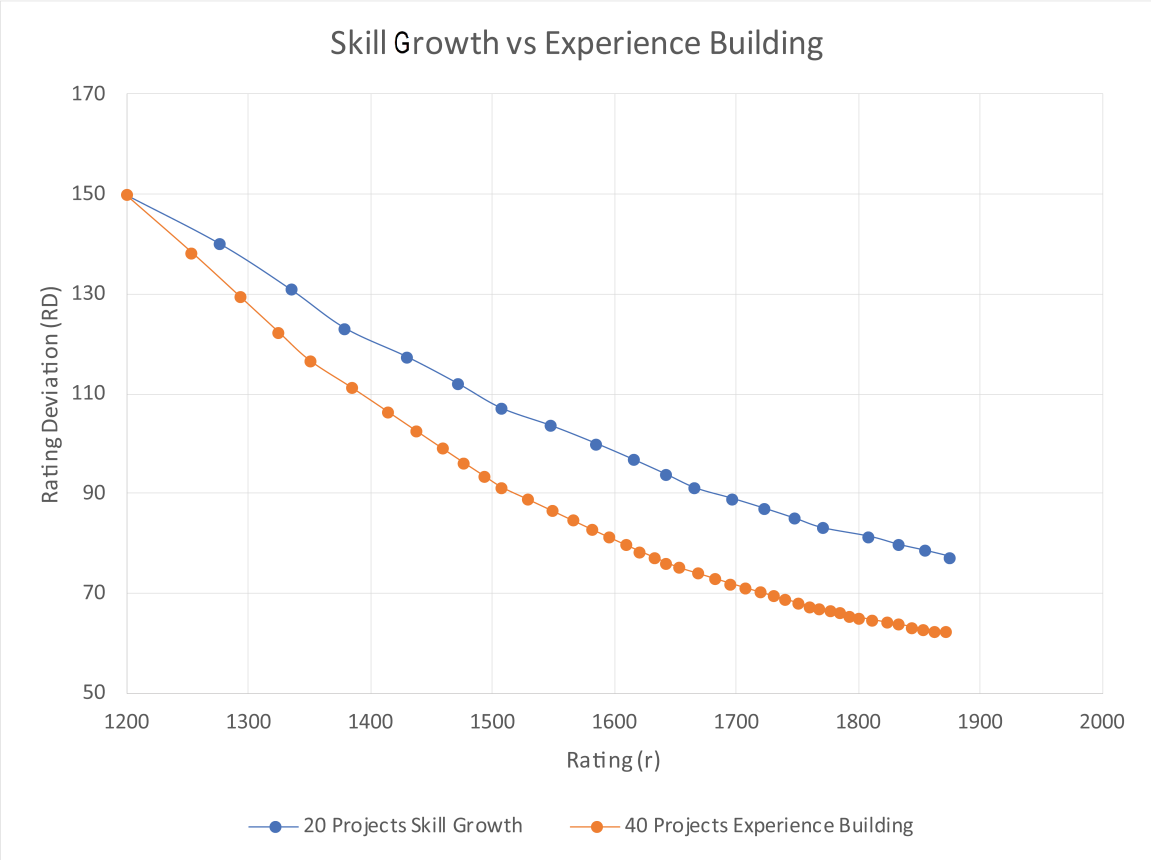

Figure 4.1 Difference between experience building and skill learning.

4.4.2 Mean vs. Median

Every company or organization may have a different definition of project difficulty levels. A challenging project for a company can be intermediate or even easy to implement for another company. The capacity for handling the project depends on various factors, such as employee quality, hiring requirements, and company funding.

So when a project manager states that the project difficulty level is intermediate, then it is mostly considering the quality of relevant employees in that particular organization. To get a better estimate of the skill level value w.r.t. to the employees' skill levels, finding an average of all those skill levels is a valid option. But finding a mean of those values has a drawback. It is highly sensitive to outliers and biased

data. For example, if a company chooses to recruit a bunch of employees, then the mean may be pushed to the lower end of the distribution. To preserve the central tendency, especially in a skewed distribution, finding a median is a better option. In this work, $\tilde{\mu}$ denotes a median.

4.4.3 Required Skill Levels

When a project manager adds a job role to a project, the job role comes with a set of required skills. Each skill has its level requirement that specifies how good an employee has to be at that skill for the system to consider him for the role. As per discussed previously in the Selection of scales section, the required skill level can either of the following:

Table 4.1 Values for Levels of a Required Skill

	Level	*Abbr.*	*Value*	*Range*
(i)	Beginner	b	$\tilde{\mu}_{s_b}$	< 1350
(ii)	Between Beginner and Intermediate	bi	$\tilde{\mu}_{s_{bi}}$	$1200 - 1500$
(iii)	Intermediate	i	$\tilde{\mu}_{s_i}$	$1350 - 1650$
(iv)	Between Intermediate and Advanced	ia	$\tilde{\mu}_{s_{ia}}$	$1500 - 1800$
(v)	Advanced	a	$\tilde{\mu}_{s_a}$	$1650 - 1950$
(vi)	Between Advanced and Expert	ae	$\tilde{\mu}_{s_{ae}}$	$1800 - 2100$
(vii)	Expert	e	$\tilde{\mu}_{s_e}$	> 2100

Let n be the number of employees with the skill in the required *level*. Median index

mi can be found using the following formula:

$$mi = \frac{n+1}{2} \tag{4.19}$$

Let $\tilde{\mu}_{s,level}$ be the median value of the employee ratings for skill s with the required level, $\mathcal{E}_{s,level,asc}$ be the set of employees with skill s and required level ordered in ascending and $\lfloor mi \rfloor$ be the value of mi when rounded to lower whole number and $\lceil mi \rceil$ be the value of mi when rounded to higher whole number.

$$\therefore \tilde{\mu}_{s,level} = \frac{\mathcal{E}_{s,level,asc}(\lfloor mi \rfloor) + \mathcal{E}_{s,level,asc}(\lceil mi \rceil)}{2} \tag{4.20}$$

The rating r for the required skill is set to the level's respective median $\tilde{\mu}_{level}$ and the rating deviation RD is set to 75. Therefore, the actual required skill level lies between the median minus 150 and the median plus 150.

4.5 Skill Weights and Ranking

4.5.1 Adding weights to skills

A project manager can add weights to each required skill. These weights help to define the job role in a better way. The sum of the weights allotted has to be equal to 1. If the project manager chooses not to put on any weights manually, then the skills will be equally weighted. Although the sum of the weights is still equal to 1, the system uses 0 to identify that the skills are by default equally weighted and also to prevent the project manager to manually putting a 0 to every skill. The r_s and w_s are the rating and the weight for skill s respectively.

4.5.2 Ranking System

It is crucial to focus upon the actual rating range of an employee's skill rather than just the rating value because both ability and experience determine the actual skill rating. In contrast, the skill rating alone is just a measure of ability. While talking

Table 4.2 Assign a Level and a Weight to Each Skill in a Job Role

Index	*Skill (s)*	*Level*	*Rating* (r_s)	*Weight* ($\sum w_s = 1$)
(i)	A	*bi*	$\tilde{\mu}_{A_{bi}}$	w_A
(ii)	B	*i*	$\tilde{\mu}_{B_i}$	w_B
(iii)	C	*b*	$\tilde{\mu}_{C_b}$	w_C
(iv)	D	*ia*	$\tilde{\mu}_{D_{ia}}$	w_D
(v)	E	*i*	$\tilde{\mu}_{E_i}$	w_E

about the range, the upper limit is above which an employee may be overwhelmed and hence is very important. Whereas on the other hand, the lower limit is not as necessary due to the face that a simple project won't stress an employee even though it's a waste of a useful resource.

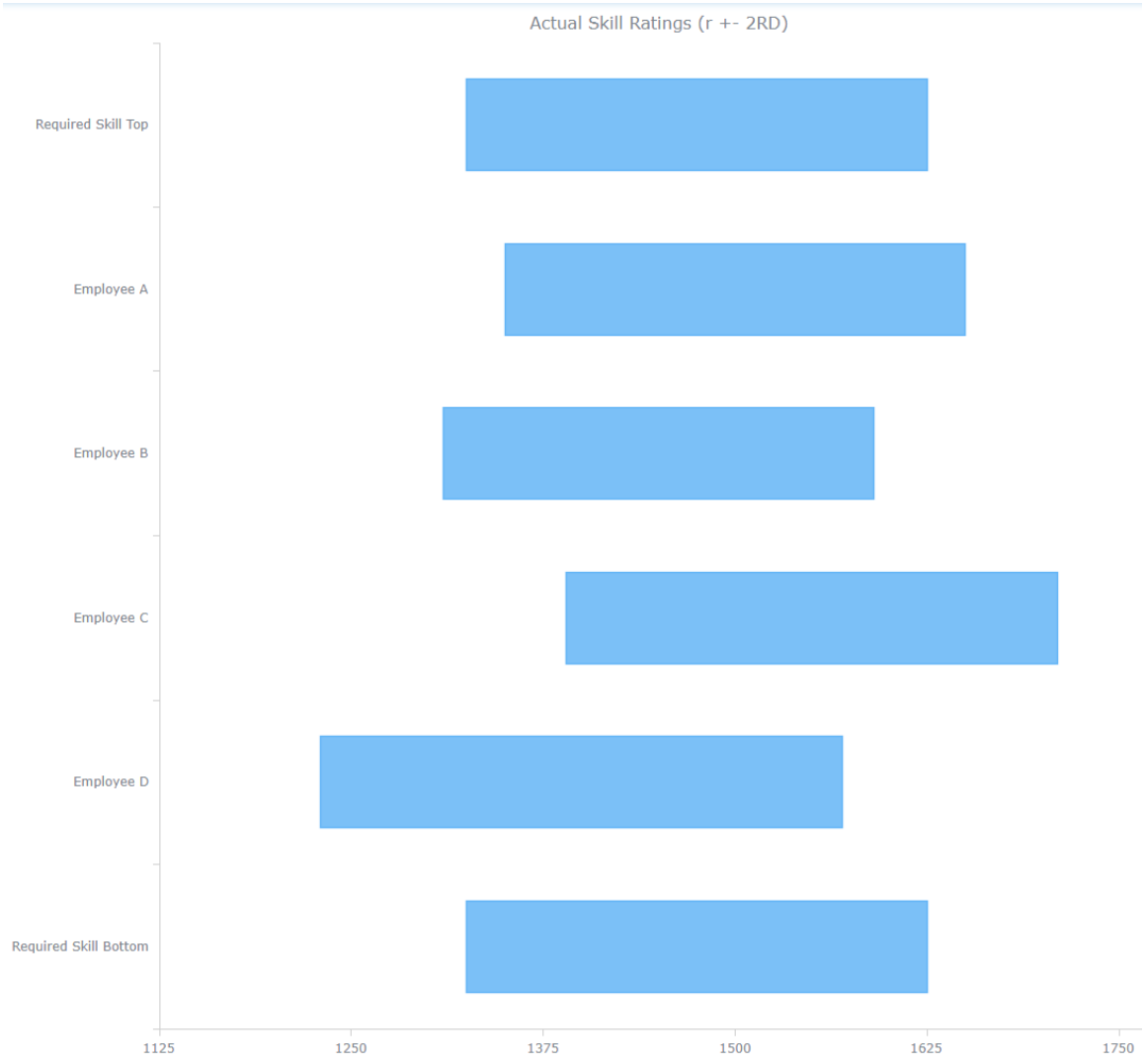

Figure 4.2 An example of actual employee skill rating alongside actual role skill rating.

Hence, the only two factors needed to calculate how far away the employee skill from the required skill is, are the rating value and the upper limit of the actual skill rating.

As discussed earlier, Actual Rating Upper Limit = rating +2× rating deviation

Let $d(e_s, \mathcal{R}_s)$ be the difference between the employee skill and the required skill. $d(e_s, \mathcal{R}_s)$ = (role skill actual rating upper limit - employee skill actual rating upper limit) + (role skill rating - employee skill rating)

Let $\mathcal{R}_{s,r}$, $\mathcal{R}_{s,RD}$, $e_{s,r}$ and $e_{s,RD}$ be the role skill rating, role skill rating deviation, employee skill rating and employee skill rating deviation respectively.

$$\therefore d(e_s, \mathcal{R}_s) = ((\mathcal{R}_{s,r} + 2\mathcal{R}_{s,RD}) - (e_{s,r} + 2e_{s,RD})) + (\mathcal{R}_{s,r} - e_{s,r}) \tag{4.21}$$

Now, simplifying the equation,

$$d(e_s, \mathcal{R}_s) = \mathcal{R}_{s,r} + 2\mathcal{R}_{s,RD} - e_{s,r} - 2e_{s,RD} + \mathcal{R}_{s,r} - e_{s,r} \tag{4.22}$$

$$\therefore d(e_s, \mathcal{R}_s) = 2\mathcal{R}_{s,r} + 2\mathcal{R}_{s,RD} - 2e_{s,r} - 2e_{s,RD} \tag{4.23}$$

Now further removing the constants,

$$d(e_s, \mathcal{R}_s) = \mathcal{R}_{s,r} + \mathcal{R}_{s,RD} - e_{s,r} - e_{s,RD} \tag{4.24}$$

$$\therefore d(e_s, \mathcal{R}_s) = (\mathcal{R}_{s,r} - e_{s,r}) + (\mathcal{R}_{s,RD} - e_{s,RD}) \tag{4.25}$$

This will calculate a score for each employee's skill with respect to the required role skill. Let n be the total number of required skills and w_s be the weight allocated to skill s in the role $\mathcal{R}$. Now to calculate the cumulative score of an employee with respect to the complete job role, weights are used as follows:

$$rank(e, \mathcal{R}) = \sum_{i=1}^{n} w_{s_i} d(e_{s_i}, \mathcal{R}_{s_i}) \tag{4.26}$$

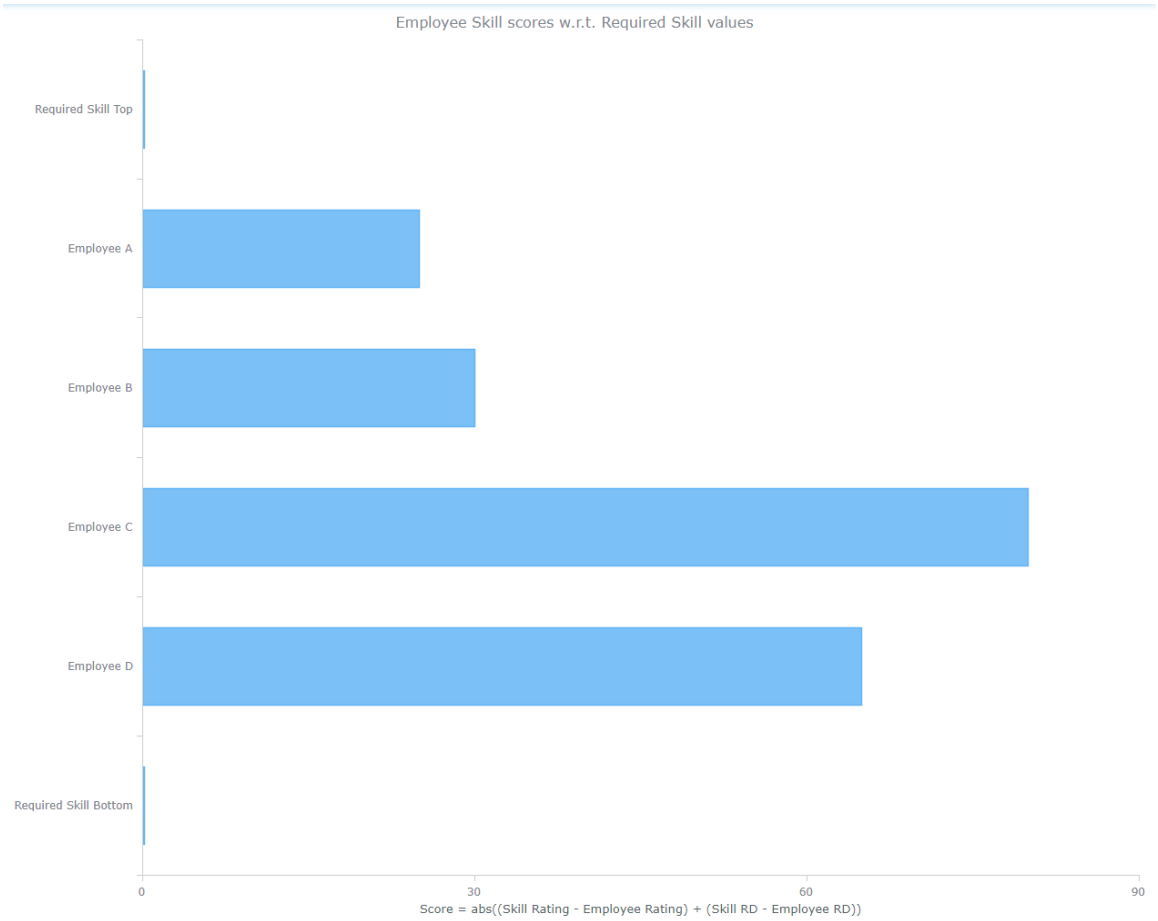

Figure 4.3 Absolute scores for employees in previous figure - smaller value is better.

4.6 Implementation

4.6.1 Ranking

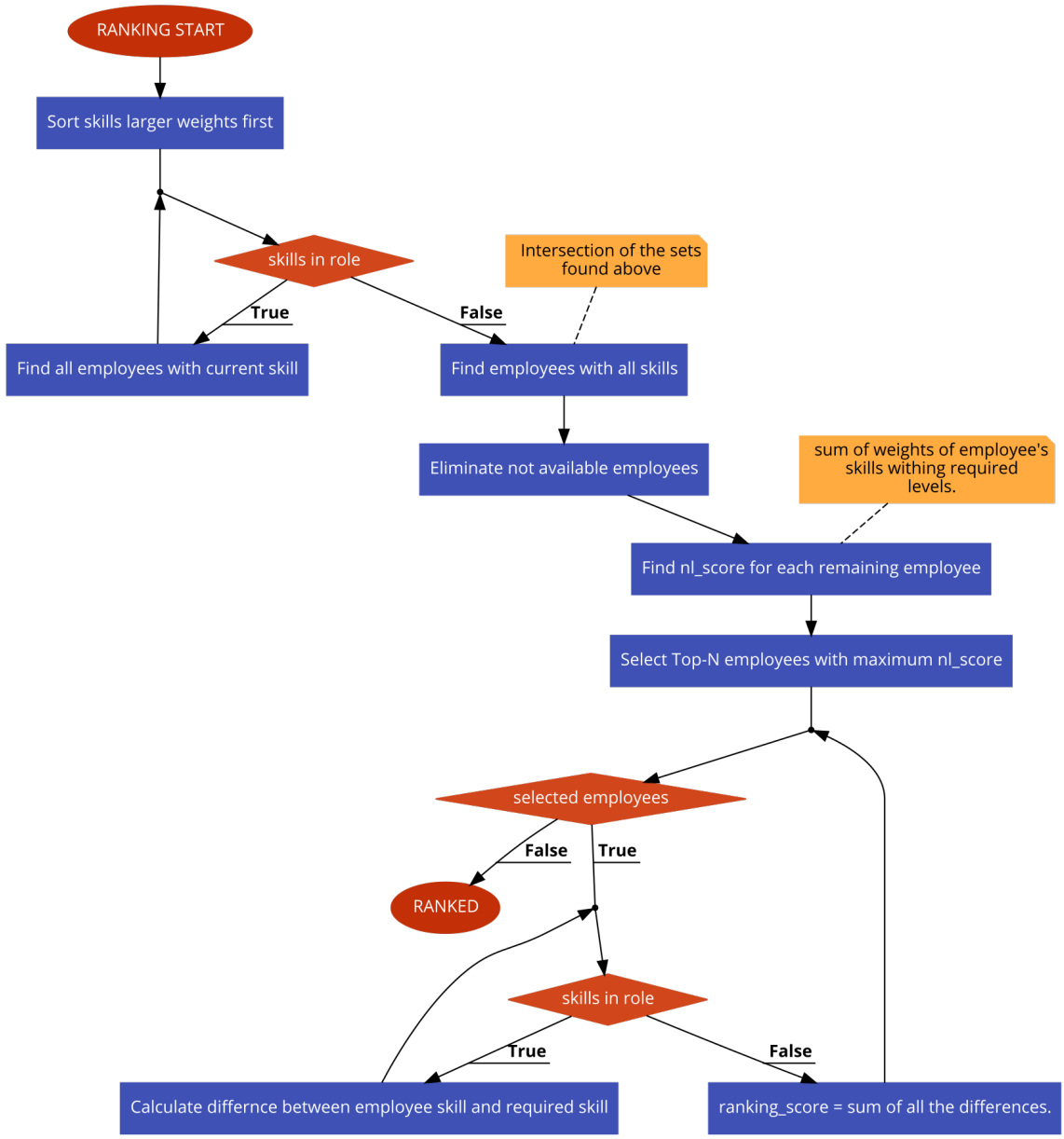

Figure 4.4 Ranking employees for a role.

Algorithm 1 Ranking employees for a role

```
1: procedure RECOMMEND_EMPLOYEES(role_skills)
2:     role_skills.sort(key=weight,decending)
3:     for skill s in role_skills do
4:         emp_skill[s] = all employees with skill s
5:     end for
6:     emp_allskills = emp_skill[0].intersection(*emp_skill)
7:     emp_available = all employees in emp_allskills with workload ¡ max_workload
8:     emp_nl = []
9:     for emp in emp_available do
10:        nl = 0
11:        for skill s in role_skills do
12:            if emp.s.r is in range(role_skills.s.level) then
13:                nl += role_skills.s.weight
14:            end if
15:        end for
16:        emp_nl.append([emp, nl])
17:    end for
18:    emp_nl.sort(key=nl, decending)
19:    emp_final = [i[0] for i in emp_nl]
20:    emp_final = emp_final[: N]
21:    emp_ranked = []
22:    for emp in emp_final do
23:        score, cr, crd = 0
24:        for skill s in role_skills do score += (role.s.r - emp.s.r) + (role.s.rd -
       emp.s.rd) cr += role.s.weight * emp.s.r crd += role.s.weight * emp.s.rd
25:        end for
26:        emp_ranked.append([emp, cr, crd, score])
27:    end for
28:    return emp_ranked
29: end procedure
```

CHAPTER 5

TEAM FORMATION

As discussed earlier, a project team consists of various job roles to fulfill their respective tasks and complete the project. Even though it sounds straightforward, the execution is very complicated. Forming an ideal team of existing employees to serve a common goal needs a great deal of consideration. The previous chapter successfully recommends the list of employees for a particular job role. This chapter explores further process using the approved employees for each role and forms multiple teams. As stated earlier, $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_m\}$ is a set of roles in a project.

5.1 Team Roles and Weights

As each skill in a job role had a weight to determine the most critical skills first; similarly, each job role in a project has a weight too. These weights are to identify the driving job roles first. The system ranks the employees for the most critical job roles first. The default case, just like with the skills in a job role, is an equal distribution of weights to the job roles if the weights are not given. The job roles $\mathcal{R}$ are listed in descending order of their weighs in the project $\mathcal{P}$.

Table 5.1 An Example of the Roles, Weights And Ranking Sequence

Index	*Role ($\mathcal{R}$)*	*Weight ($\sum w_{\mathcal{R}} = 1$)*	*Ranking Sequence*
(i)	A	0.4	1
(ii)	C	0.3	2
(iii)	D	0.2	3
(iv)	B	0.1	4

5.2 Team Speciality Options

It's not necessary to have a perfect team. Sometimes a bit more and special is expected from a team. For example, sometimes, creativity can be a special requirement for a project, or the project may need a cost-effective solution due to its low budget. Hence, for such diversity, the system recognizes five different types of a team that can be formed with varying specialties. This section explores every kind of team a project manager may request.

5.2.1 Balanced

The original type of team this system tends to suggest by default. The system designates this type when a project manager asks for a team to be as close to the requirements as possible. In other words, the employees with the score as close as possible to zero are selected for each role.

Let $\mathcal{T}_i$ be the i_{th} member of the team who is selected for role $\mathcal{R}_i$ and the roles are listed in the descending order of their weights in project $\mathcal{P}$.

$$\mathcal{T}_{balanced}[i] = \min(|\mathcal{E}_{\mathcal{R}_i,ranked}[score]|), \forall \mathcal{R} \in \mathcal{P} \tag{5.1}$$

$\therefore \mathcal{T}_{balanced} = \{e_1, e_2, e_3, \ldots, e_m\}$ is the balanced team offered for project $\mathcal{P}$.

5.2.2 Skill Growth

As the name suggests, this type focuses solely on the growth of the employees, pushing them further towards their limits. This type forms a team of employees with skill ratings less than the required skill level. In other words, employees with a maximum score from the ranked list of employees for each job role. Let $\mathcal{T}_{growth} = \{e_1, e_2, e_3, \ldots, e_m\}$ be the team of employees who needs skill growth for project $\mathcal{P}$.

$$\mathcal{T}_{growth}[i] = \max(\mathcal{E}_{\mathcal{R}_i,ranked}[score]), \forall \mathcal{R} \in \mathcal{P} \tag{5.2}$$

5.2.3 Quality focused

The sole purpose of this type of team is to deliver a creative solution to the project. To be creative, one need not have experience. The experience forces a person to fit in the pattern according to the author in [11]. Breaking the patterns is a difficult process. Experience teaches oneself to make a linear movement, whereas creativity demands non-linearity.

In this system, rating r of a skill defines the ability, whereas rating deviation RD measures employee experience. Finding a creative mind means finding an employee with highest cumulative rating of skills in Role $\mathcal{R}$. Let $cr(e, \mathcal{R})$ compute cumulative rating for employee e with respect to skills in role $\mathcal{R}$.

$$cr(e, \mathcal{R}) = \sum^{s \in \mathcal{R}} w_s e_s(r) \tag{5.3}$$

Let $\mathcal{T}_{quality} = \{e_1, e_2, e_3, \ldots, e_m\}$ be the team of employees who are creative enough for the project $\mathcal{P}$.

$$\mathcal{T}_{quality}[i] = \max(cr(\mathcal{E}_{\mathcal{R}_i, ranked}), \mathcal{R}_i), \forall \mathcal{R} \in \mathcal{P} \tag{5.4}$$

5.2.4 Time Saver

This type is similar to the previous type as this one needs employees with least cumulative rating deviation. To save time on a project, employees need to work linearly and procedurally. Experience, in this case, rating deviation RD is a more important factor here than creativity. Let $crd(e, R)$ calculate cumulative rating deviation for employee e with respect to skills in role $\mathcal{R}$.

$$crd(e, \mathcal{R}) = \sum^{s \in \mathcal{R}} w_s e_s(RD) \tag{5.5}$$

Let $\mathcal{T}_{time} = \{e_1, e_2, e_3, \ldots, e_m\}$ be the team of employees who can complete the project $\mathcal{P}$ faster than the rest.

$$\mathcal{T}_{time}[i] = \min(crd(\mathcal{E}_{\mathcal{R}_i, ranked}), \mathcal{R}_i), \forall \mathcal{R} \in \mathcal{P} \tag{5.6}$$

5.2.5 Cost Effective

Even though not usually recommended, an organization may need this type for limited and low budget projects. Provided that the cost for each employee is available and accurate, a team with low-cost employees for all the job roles can be built. Let $\mathcal{T}_{cost} = \{e_1, e_2, e_3, \ldots, e_m\}$ be the cheapest team of employees to complete the project $\mathcal{P}$.

$$\mathcal{T}_{cost}[i] = \min(\mathcal{E}_{\mathcal{R}_i, ranked}[cost]), \forall \mathcal{R} \in \mathcal{P} \tag{5.7}$$

Respective authors in [29, 30, 31, 35] have tried to solve the team formation problem with the focus aspect being the cost effectiveness by minimizing the expert costs and the communication costs if any.

5.3 Criteria Check and Teams Completion

As discussed earlier, the job roles are listed in the descending order of their weights in a project, which makes it easier to recommend employees for the most important job roles first. The system starts checking for the following criteria while recommending for each role after the first one. The checks are performed on the team being built, that is, set $\mathcal{T}$. The system does the workload check while recommending an employee and hence, is not needed to be done again.

5.3.1 Criterion 1: Avoiding Employee Duplication

Currently, this system allows only one role per employee. The system can be modified later to let the project manager decide whether an employee is allowed to have multiple roles in a project. Multiple roles consume multiple workloads. As of now, to avoid duplication, the recommended employees from the second job role onwards are checked one by one if they exist in the team $\mathcal{T}$. If not, they are added to the team, and if they do, then the next possible option is selected, and the checks are performed again. Let

$e_{\mathcal{R}_i}$ be the employee recommended for i^{th} role $\mathcal{R}$.

$$\nexists e_{\mathcal{R}_i} \in \mathcal{T} \Rightarrow \mathcal{T}[i] = e_{\mathcal{R}_i} \tag{5.8}$$

5.3.2 Criterion 2: Respecting Mutual Ratings

Let $c(e, \mathcal{T})$ be a compatibility function that returns the number of non-compatible employees in team $\mathcal{T}$ with employee e by checking the employee relations rating stored in the database. The function checks on the mutual rating between employee e and each team member that has already been selected in team $\mathcal{T}$ and returns zero if the employee is compatible with the other team members. The employee is added to the team only if the compatibility function returns 0 or else the function return 1 and the next possible employee option is suggested. This new employee has to go through the compatibility test too.

$$c(e_{\mathcal{R}_i}, \mathcal{T}) = 0 \Rightarrow \mathcal{T}[i] = e_{\mathcal{R}_i} \tag{5.9}$$

5.4 Feedback

Currently, this system relies heavily on the feedback loop for the mutual ratings among project team members. The system asks for feedback from every team member, including the project manager. It requests each employee to rate every other team member after each project completion. As of now, the system only makes use of the mutual ratings collected from the feedbacks. Other factors, such as project difficulty, communication, and management related queries, can be asked in the feedback.

Feedback helps improve employee engagement and working relationships as per the author in [33] Positive feedback loops are a fundamental concept in psychology. Give people feedback about their actions promptly without fear of reprisal, and it allows them to work toward better behaviors according to the discussion by the author in [27]

Such feedback practices boost the collaboration where any researchers use this employee inter-relation data for team formation experiments. Respective authors in

[30, 12, 19, 7, 43, 26] use collaboration as the key aspect while trying to solve the team formation problem. These feedback loops are necessary after project completion to free the workload for the next assignment.

5.5 Implementation

5.5.1 Criteria Check

Algorithm 2 Checking Employee Eligibility

```
1: procedure CRITERIA_CHECK(emp, team)
2:     if ! emp in team then
3:         flag = 0
4:         for member in team do
5:             if emp.mutual_rating.member ≠ 0 & < rating_threshold then
6:                 flag = 1
7:             end if
8:         end for
9:         if flag == 0 then
10:            return true
11:        end if
12:    end if
13:    return false
14: end procedure
```

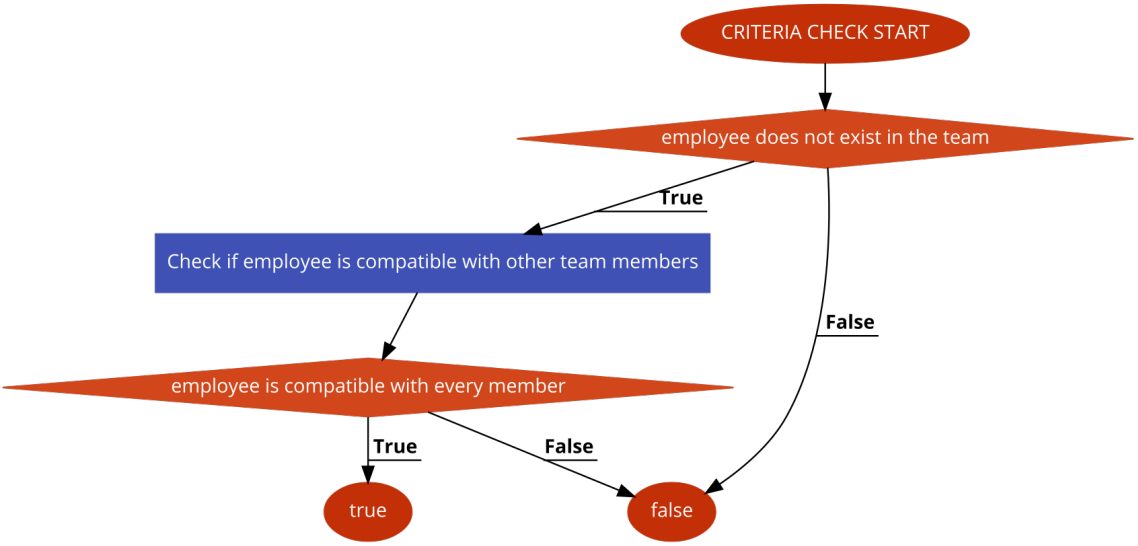

Figure 5.1 Check if the employee is eligible to be in the team.

5.5.2 Team Types

Algorithm 3 Select a team member for each team for a role

```
1: procedure GET_MEMBER(emps, team, type)
2:     if type == balanced then
3:         emps.sort(key = abs(score))
4:     else if type == growth then
5:         emps.sort(key = score, reverse = true)
6:     else if type == cost then
7:         emps.sort(key = cost)
8:     else if type == quality then
9:         emps.sort(key = cr, reverse = true)
10:    else if type == time then
11:        emps.sort(key = crd)
12:    end if
13:    i = 0
14:    while !criteria_check(emps[i], team) || i ≥ emps.length do
15:        i++
16:    end while
17:    if i ≥ emps.length then
18:        return null
19:    else
20:        return emps[i]
21:    end if
22: end procedure
```

5.5.3 Teams Formation

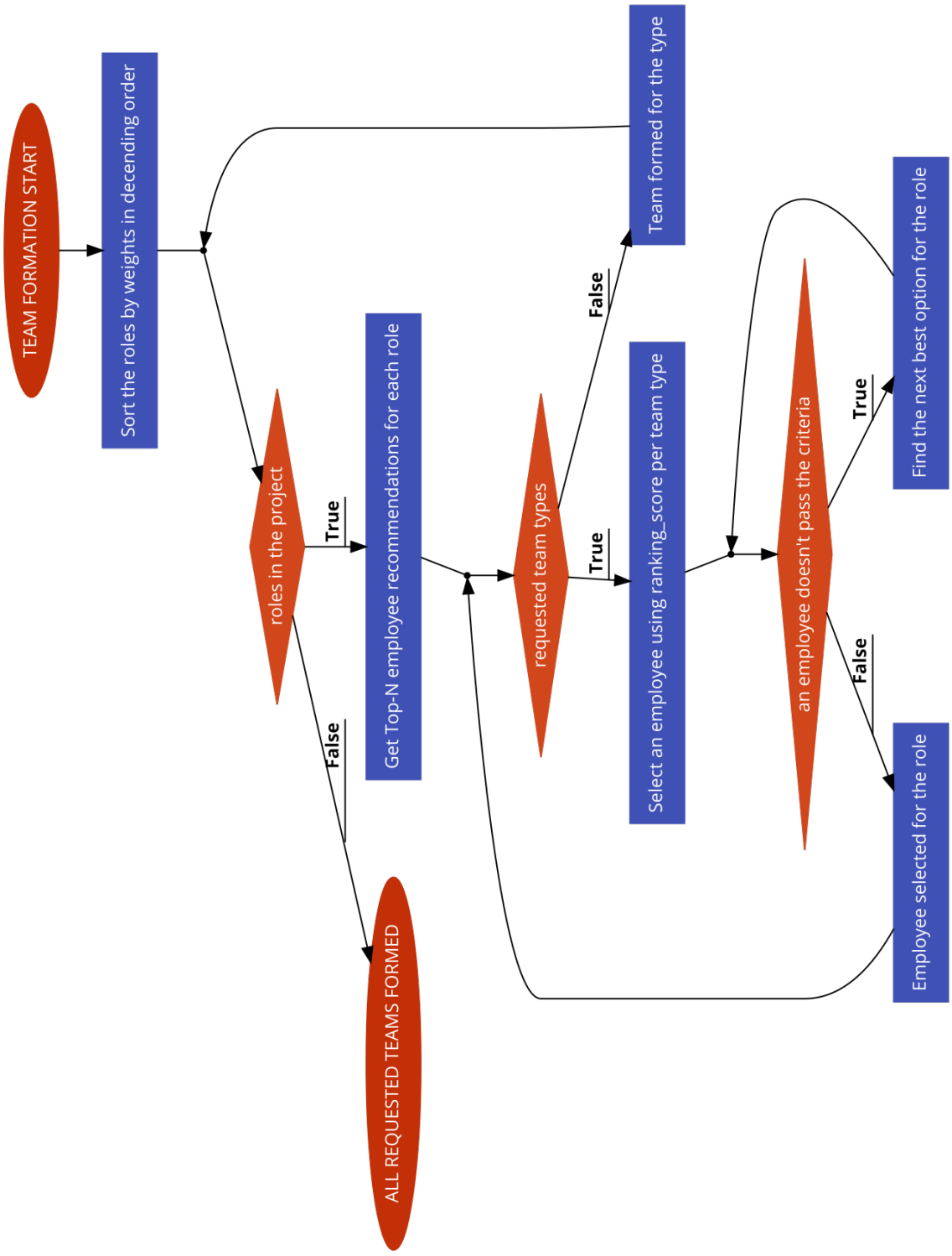

Figure 5.2 Form a team for each requested type.

Algorithm 4 Form a team for each type

```
project = [[[[skill,r,rd,weight],...],rweight],...]
project.sort(key=weight, decending)
team = null
for type in team_types do
    team.append = []
end for
for role in project do
    rec_emps[role] = recommend_employees(project[role])
    for type in team_types do
        team[type].append(get_member(rec_emps[role], type, team[type]))
    end for
end for
```

5.5.4 Skill updates

Algorithm 5 Update the skill ratings when

```
procedure UPDATE_SKILL(emp.skill, role.skill, event)
    if event = project complete then glicko2.skill_update(emp.skill, role.skill, 1)
    else if event = project or employee terminated then
        if event not related to performance then glicko2.skill_update(emp.skill, role.skill, 0.5)
        else if event related to performance then glicko2.skill_update(emp.skill, role.skill, 0)
        end if
    end if
end procedure
```

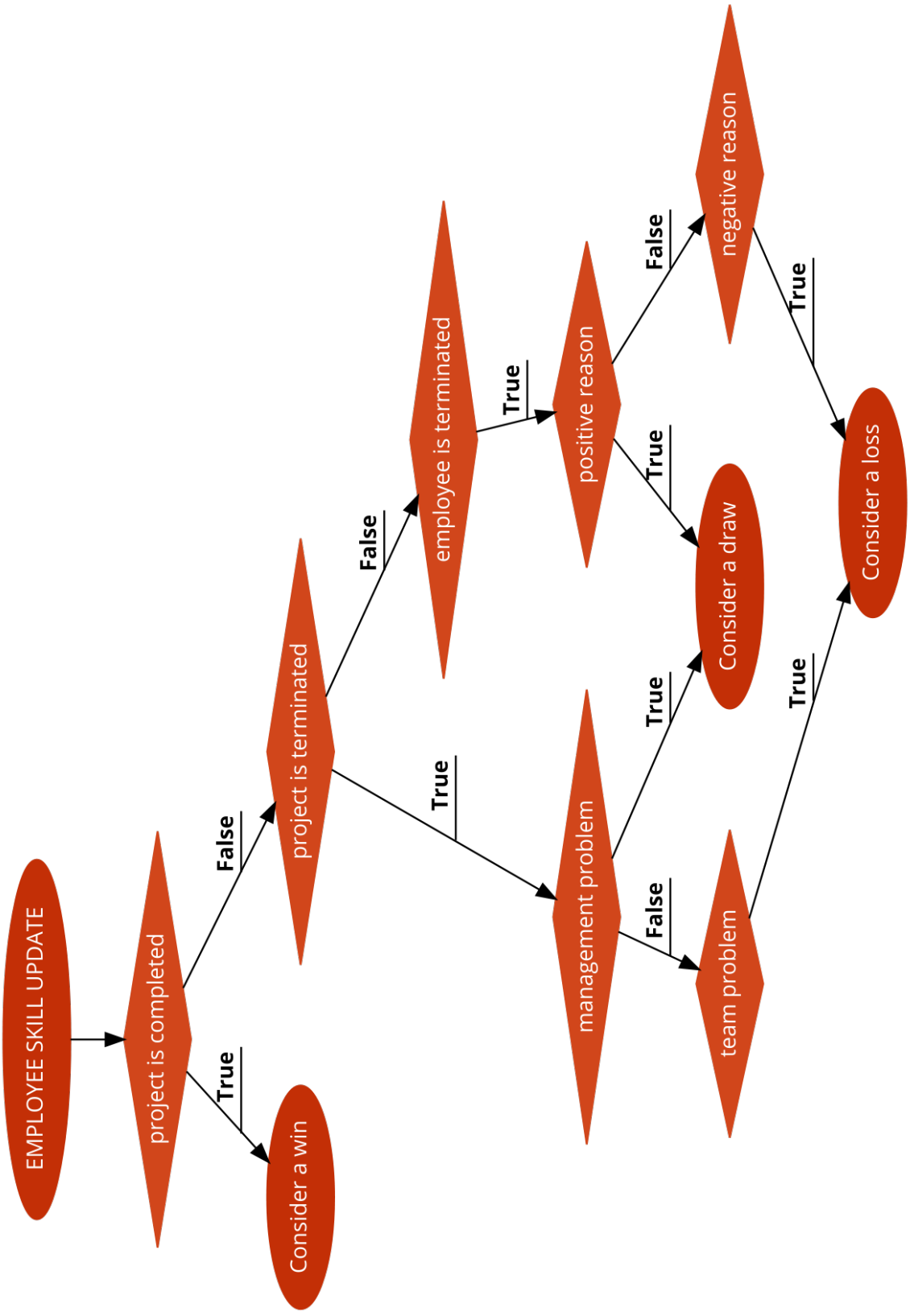

Figure 5.3 Update skill values when an event occurs.

CHAPTER 6

CONCLUSION

By now, it is clear that there are never-ending possibilities to expand the team formation solution and cruises through numerous other domains of project management. This work tried to design and implement a standard solution for the team formation problem using a simple recommendation system. This work focuses on giving the project manager a team as close to the requirement as possible; on the other hand, a project to employees that they deserve.

This work creates a rating environment for skills and uses it to get the project manager's requirements. It then ranks the employees as per the requirements and suggests different types of teams, each one with its unique focus. Lastly, it also sheds some light on a few challenges that the team formation solutions overall have to face and also discuss the plans for this work.

The implementation in this work covers designing and development of a recommendation system based on Glicko-2 rating system which evaluates, estimates, and updates the actual skill rating of an employee. The framework takes the project requirements as the input and uses the recommendation system to recommend a number of employees for each role. One of those employees is then selected for the role which is continued for each role in the project, hence forming the complete team.

The framework also tries to suggest five different teams, if requested, where each team has it's own benefit to the organization such as a balanced team, a employee growth concerned team, a quality focused team, a time saver team, and a cost effective team. This work also shows the potential to expand beyond just team formation to hiring and outsourcing. The implementation can also guide those who are trying to create and implement their team formation solution because the implementation requires a lot more preparation than just the solution.

Hence, all research works on team formation, and its aspects such as this one will eventually lead to better working culture, and the employees will start getting to work on what they deserve.

6.1 Alternative Approaches

Various aspects of experts, teams, and team formation such as the ones discussed by the author in [44] through the organizational point of view. Many other similar yet unique works using different approaches and techniques to the same problem but with different applications while considering various related aspects can be seen in [3, 10, 17, 2, 22, 28, 6, 39, 18, 21, 46, 4, 38, 14, 13, 35]

6.1.1 Employees Recommendation

The authors in [24] describe the implementation of a approach that uses a standard collaborative filtering recommendation algorithm twice to recommend employee for the open job positions in the organization. But in this algorithm, the users and the items are flipped. It recommends users for a given item rather than recommending items for a given user.

First, because the users and items are flipped, skills are considered as users and roles are considered as the items. Instead of each role associated with multiple skills, each skill is associated with multiple roles it belongs to. A similarity matrix is generated between each skill pair by computing the similarity between two skills in the pair using the Tanimoto coefficient, also known as the Jaccard coefficient. It computes the ratio of common elements in the set to the total size for the sets, which is a measure of similarity between the two sets. The recommender then uses the nearest neighbor approach to find Top-few skills for the skillset based on the similarity.

Now that a skillset for a role is found, an employee skillset is considered as a user, and skills are considered as items the second time algorithm is used. A similarity matrix is generated between skillsets, and Top-N employees with the

skillset are recommended. The authors in [34] describe the tools and techniques to provide awareness of team members in the virtual collaboration environment and automated discovery of distributed experts. The experiment resulted in embodiment of three different solutions among which one was to support automated expertise identification. Other such experiments and implementations were discussed in [20, 45, 8, 9, 36] by their respective authors.

Advantages of this approach are:

1. It takes less preparation to recommend.

2. Recommends employees with potential skills and not just the ones that strictly has them.

Disadvantages of this approach are:

1. Instead of recommending the employees who can do the job, this approach recommends the employees you may be able to pull it off.

2. It cannot distinguish between the employees for creativity, experience, and cannot form a balanced team.

6.1.2 Employees Matching to Form a Team

For the recommenders that recommend multiple employees rather than just one, a simple matching algorithm to form a team is needed. A matching algorithm can arrange the employees for each role in the order of a particular parameter or score in ascending or descending. Depending on whether the problem is maximization or minimization. Or another way is to form combinations of the recommended employees and choose the team with the maximum or minimum score. All of this, while checking for employee duplication and compatibility.

6.2 Future Scope

6.2.1 Skill Ratings

The current rating system can be replaced later with a machine learning prediction algorithm when there is sufficient data to understand employee skills while working on the projects. At first, both the algorithms, the current one, and the machine learning may work simultaneously to help the machine learning algorithm detects the patterns in skill rating changes. Later, it can completely replace the current glicko2 rating system.

6.2.2 Skill Cluster

Later, when the system collects enough data about the skills used, a skill cluster can be formed to find related skills, alternative skills and help businesses grow by recommending other projects that can be handled using current employees' skillsets.

6.2.3 Employee Training

Skill upgrades can be done not only by handling projects but also by learning them by taking on pieces of training and courses. Further, courses can be recommended to the employees for their skills that need growth, and they can be added to the skill ratings. The authors in [40] can identify the students who require attention and recommend them the additional courses accordingly to boost their progress. Similarly, the authors in [32] discusses the recommender systems for technology enhanced learning (TEL) to support and enhance learning practices for individuals and organizations. Also, the authors in [42] use various available datasets to evaluate and compare the performance of the recommendation algorithms for TEL.

6.2.4 Manual Employee Selection

A project manager may need to include a specific employee to the team. This feature can be added, and other team members can be selected accordingly. It can also be

extended to allowing the project manager to choose other alternatives recommended by the system.

6.2.5 Handling long projects

Extended projects can be handled by splitting them into multiple small projects. This way, it can benefit in multiple ways like regular employee skill updates and better management of deadlines. It can also help to work on multiple small projects simultaneously to boost the overall project completion. The team can be allowed to be carried to another small project if needed in this case.

6.2.6 Beginner Employees

It's highly unlikely for those starting from beginner level to being allocated a project as there are almost negligible chances for such projects to exist in any organization. Hence, to give them a break, they can be allocated with the existing team for training. The rating system for them needs to be changed.

6.2.7 Performance Evaluation and Feedback Practices

A company needs to perform performance evaluations now and then. These evaluations are intended to boost employee skill levels or promote them to the next level of the skills. Similarly, feedback practices are equally essential after the project is complete. Feedback includes not only the technical details but also the queries about essential aspects like behavior, satisfaction, comfort, and management.

6.2.8 Gamification

This complete system can be converted into a game-like structure where employees compete with each other in the skill scores and are rewarded for the reaching levels. The reward may be something like being eligible to have a beginner level employee

working for the same role assisting in the project. However, this is going to need a lot of game-like rules to be set and an experimental approach.

6.2.9 Freelancers and Hiring

This system can be extended for hiring purposes or selecting a group of freelancers for outsourcing. This application requires the complete system to be accessible to the public, where they can find jobs for themselves while getting picked up by the companies at the same time.

6.2.10 Project Deadline Estimation

After sufficient data is collected, it can be used to estimate the project deadline based on the team selected. The team members' skill levels can be a few of the many other vital factors to determine the approximate time to complete the project. Based on this data, it becomes easier for the system to take some decisions like allocating the employee to a soon starting project before the current one, which is on the verge of ending almost ends.

6.2.11 Workload Estimation

It is imperative to estimate the workload to prevent an employee from being overwhelmed with work and miss deadlines. This estimate can be a deciding factor to check whether the employee can manage one more project over his existing workload. For estimation to be possible, the system needs to be trained for workload balancing first.

6.3 Challenges

Not only the system discussed here, but the team formation problem itself faces many challenges and has various approaches to solve sub-problems. The authors in [44] discuss a few works trying to solve the same problem with their challenges and limitations. A few of these challenges are mentioned below:

1. A project manager plays an important role not only in the team's success but also for the system like this one. A project manager has to be very smart while describing the project to the system and entering the correct data at the correct place. A project manager's single mistake or misunderstanding in feeding inputs to the system can create massive differences in the expected solution. Almost every team formation solution out there needs a project manager's manual intervention.

2. Always finding the best team for a specific type of project may lead to many others far from the requirements to sit idle and never get a chance to ever work on the type of project with the skills required they are way off from.

3. A new employee or an intern may never get a chance to work on the projects they deserve as they would never fit in any requirements right away.

4. Skill updates won't be accurate if the employee never actually worked or got significant help regarding a particular skill in a project.

5. Projects can sometimes be very long to justify the skill updates post project completion. The solution to this can be the skill updates after the regular intervals or converting the long project into a few short ones.

6. Employee compatibility is a tougher challenge than other challenges. Many have tried to solve it with various solutions, such as psychometric tests and regular feedbacks.

7. Cold start is one of the common problems in recommendation systems because of the insufficient data or the default data values. The employees new to the system do not have enough data to start working on the projects they deserve.

8. One of the most sensitive topics and red flags is maintaining diversity, whether ethnicity or gender, while forming a team.

APPENDIX A

WEB SITES & SKILLS DATASETS

	Web Site (Cited March 3, 2020)	*Description*
1.	https://www.onetcenter.org/dictionary/24.2/excel/technology_skills.html	A web library with the skills data.
2.	https://data.world/jobspikr/software-developer-job-listings-from-usa	Software developer job listings from USA.
3.	https://www.kaggle.com/elroyggj/indeed-dataset-data-scientistanalystengineer	Indeed Dataset - Data Scientist/Analyst/Engineer.
4.	https://www.kaggle.com/atahmasb/amazon-job-skills	Amazon Software Development Job Skills.
5.	https://www.kaggle.com/niyamatalmass/google-job-skills	Google Job Skills.
6.	http://dataatwork.org/data/	An API for skills.

REFERENCES

[1] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[2] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Power in unity: Forming teams in large-scale community systems. pages 599–608, 01 2010.

[3] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 839–848, New York, NY, USA, 2012. Association for Computing Machinery.

[4] Giorgio Barnabò, Adriano Fazzone, Stefano Leonardi, and Chris Schwiegelshohn. Algorithms for fair team formation in online labour marketplaces. 02 2020.

[5] Senjuti Basu Roy, Laks V.S. Lakshmanan, and Rui Liu. From group recommendations to group formation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1603–1616, New York, NY, USA, 2015. Association for Computing Machinery.

[6] Adil Baykasoglu, Turkay Dereli, and Sena Das. Project team selection using fuzzy optimization approach. *Cybern. Syst.*, 38(2):155–185, February 2007.

[7] Michelle Cheatham and Kevin Cleereman. Application of social network analysis to collaborative team formation. In *Proceedings of the International Symposium on Collaborative Technologies and Systems*, CTS '06, page 306–311, USA, 2006. IEEE Computer Society.

[8] Anwitaman Datta, Jackson Tan Teck Yong, and Anthony Ventresque. T-recs: Team recommendation system through expertise and cohesiveness. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, page 201–204, New York, NY, USA, 2011. Association for Computing Machinery.

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

[10] Samik Datta, Anirban Majumder, and K. Purushotam Naidu. Capacitated team formation problem on social networks. In *KDD*, 2012.

[11] Dinesh Divekar. Knowledge vs experience vs creativity.

[12] Christoph Dorn and Schahram Dustdar. Composing near-optimal expert teams: A trade-off between skills and connectivity. pages 472–489, 12 2010.

[13] Bowen Du, Qian Tao, Feng Zhu, and Tianshu Song. Finding optimal team for multiskill task based on vehicle sensors data. *Journal of Sensors*, 2017:1–10, 10 2017.

[14] Walaa El Ashmawi. An improved african buffalo optimization algorithm for collaborative team formation in social network. *International Journal of Information Technology and Computer Science*, 10:16–29, 05 2018.

[15] A. Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, Stefan Reiterer, and Martin Stettinger. *Basic Approaches in Recommendation Systems*, pages 15–37. 12 2014.

[16] Kaiyu Feng, Gao Cong, Sourav S. Bhowmick, and Shuai Ma. In search of influential event organizers in online social networks. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, page 63–74, New York, NY, USA, 2014. Association for Computing Machinery.

[17] Kaiyu Feng, Gao Cong, Sourav S. Bhowmick, and Shuai Ma. In search of influential event organizers in online social networks. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, page 63–74, New York, NY, USA, 2014. Association for Computing Machinery.

[18] Erin L. Fitzpatrick and Ronald G. Askin. Forming effective worker teams with multi-functional skill requirements. *Comput. Ind. Eng.*, 48(3):593–608, May 2005.

[19] Amita Gajewar and Atish Das Sarma. Multi-skill collaborative teams based on densest subgraphs. *Computing Research Repository - CORR*, 02 2011.

[20] Dawei Gao, Yongxin Tong, Jieying She, Tianshu Song, Lei Chen, and Ke Xu. Top-k team recommendation and its variants in spatial crowdsourcing. *Data Science and Engineering*, 2, 03 2017.

[21] M Gaston, John Simmons, and M DesJardins. Adapting network structure for efficient team formation. In *Proceedings of the AAAI 2004 fall symposium on artificial multi-agent learning*, 2004.

[22] Matthew E. Gaston and Marie desJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, page 230–237, New York, NY, USA, 2005. Association for Computing Machinery.

[23] YoungOk Kwon Gediminas Adomavicius, Nikos Manouselis. Multi-criteria recommender systems.

[24] Abigail Gertner and Susan Lubar. Recommendations to support staffing decisions. 2014.

[25] Mark E. Glickman. Example of the glicko-2 system.

[26] Diego Gómez-Zara, Matthew Paras, Marlon Twyman, Jacqueline Ng, Leslie Dechurch, and Noshir Contractor. Who would you like to work with? pages 1–15, 04 2019.

[27] Cord Himelstein. The importance of the employee feedback loop.

[28] Mehdi Kargar and Aijun An. 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*, CIKM '11, page 985–994, New York, NY, USA, 2011. Association for Computing Machinery.

[29] Mehdi Kargar, Aijun An, and Morteza Zihayat. Efficient bi-objective team formation in social networks. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 483–498, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[30] Mehdi Kargar, Morteza Zihayat, and Aijun An. *Finding Affordable and Collaborative Teams from a Network of Experts*, pages 587–595. 05 2013.

[31] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 467–476, New York, NY, USA, 2009. Association for Computing Machinery.

[32] Nikos Manouselis, Hendrik Drachsler, Katrien Verbert, and Erik Duval. *Recommender systems for learning*. Springer Science & Business Media, 2012.

[33] Emily Marsh. Why feedback is important in the workplace.

[34] Mark T. Maybury, Raymond J. D'Amore, and David House. Awareness of organizational expertise. *International Journal of Human–Computer Interaction*, 14:199 – 217, 2002.

[35] Yashar Najaflou and Kris Bubendorfer. In pursuit of the wisest: Building cost-effective teams of experts. pages 158–167, 10 2017.

[36] Z. Ning, X. Zeng, M. Fu, T. Megersa Bekele, and X. Wang. A catfish effect based team recommendation system. In *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 203–208, 2018.

[37] Rachel Reddick. Using a glicko-based algorithm to measure in-course learning. In *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, page 754–759, 2019.

[38] Kalyani Selvarajah, Pooya Moradian Zadeh, Mehdi Kargar, and Ziad Kobti. Identifying a team of experts in social networks using a cultural algorithm. *Procedia Computer Science*, 151:477–484, 01 2019.

[39] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[40] Kelly Spoon, Joshua Beemer, John C Whitmer, Juanjuan Fan, James P Frazee, Jeanne Stronach, Andrew J Bohonak, and Richard A Levine. Random forests for evaluating pedagogy and informing personalized learning. *Journal of educational data mining*, 8(2), 2016.

[41] Studytonight. Advantages of mongodb.

[42] Katrien Verbert, Hendrik Drachsler, Nikos Manouselis, Martin Wolpers, Riina Vuorikari, and Erik Duval. Dataset-driven research for improving recommender systems for learning. In *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*, pages 44–53, 2011.

[43] Hyeongon Wi, Seungjin Oh, Jungtae Mun, and Mooyoung Jung. A team formation model based on knowledge and collaboration. *Expert Syst. Appl.*, 36(5):9121–9134, July 2009.

[44] M.S.A.V.P.V. Wulf, M.S. Ackerman, A.P.C.S.C.W.S.M.V. Pipek, V. Pipek, V. Wulf, Inc Books24x7, and P.I.S.N.M.V. Wulf. *Sharing Expertise: Beyond Knowledge Management*. EBL-Schweitzer. MIT Press, 2003.

[45] Qinghai Zhou, Liangyue Li, Nan Cao, Norbou Buchler, and Hanghang Tong. Extra: Explaining team recommendation in networks. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 492–493, New York, NY, USA, 2018. Association for Computing Machinery.

[46] ARMEN Zzkarian and Andrew Kusiak. Forming teams: an analytical approach. *IIE transactions*, 31(1):85–97, 1999.