3-10-2019

Productive Cluster Hire

PARTH ATULKUMAR PATEL
University of Windsor

Productive Cluster Hire

By

Parth Patel

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

2019

Productive Cluster Hire

By

Parth Patel

APPROVED BY:

M. Hlynka

Department of Mathematics and Statistics

S. Samet

School of Computer Science

M. Kargar, Co-Advisor

School of Computer Science

Z. Kobti, Advisor

School of Computer Science

January 10, 2019

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Discovering a group of experts to complete a set of tasks that require various skills is known as Cluster Hire Problem. Each expert has a set of skills which he/she can offer and charges a monetary cost to offer their expertise. We are given a set of projects that need to be completed and on completion of each project, the organization gets a Profit. For performing a subset of given projects, we are given a predetermined budget. This budget is spent on hiring experts. We extend this problem by introducing productivity and capacity of experts. We want to hire experts that are more productive, and this factor is determined on the basis of their past experience. We also want to make sure that no expert is overworked as it is not possible for a single expert to provide his/her expertise for unlimited times. Our goal is to hire as many experts as possible in which the sum of their hiring costs (i.e., salary) is under the given budget as we are interested to maximize the profit and also maximize the productivity of the group of experts, our problem is a bi-objective optimization problem. To achieve this, we propose two different approaches that maximize our Profit and Productivity.

DEDICATION

Dedicated to my grandfather Late Shri Arvindbhai Punambhai Patel, my grandmother Jashodaben Arvindbhai Patel, my parents Atulkumar Patel and Hema Patel, my uncle Jitendra Patel, my aunt Kumudben Patel and to my sisters and brothers Janki, Niki, and Dev.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor Dr. Ziad Kobti and Dr. Mehdi Kargar for their continuous support in my research, for their patience, motivation, and immense knowledge. Their guidance helped me throughout my time of research and writing of this thesis. I could not have imagined having better mentors for my study.

I would also like to thank my thesis committee members Dr. Saeed Samet and Dr. Myron Hlynka for their valuable comments and suggestions for the completion of this thesis. I would like to sincerely thank Dhruvi Patel and Meet Patel, who have been the fundamental support throughout my work.

Also, I would like to thank my fellow lab mates for the stimulating discussions during the development of this work. Last but not the least, my deepest gratitude to my family for their unconditional love, support, and belief.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

Finding the best combination of experts to complete a given task is of utmost importance to any company and is considered as an asset. Searching for the right group of experts involves assigning various skills possessed by experts to complete all projects undertaken by the company and ensuring that they are completed within the given deadline. Hiring such individuals will definitely need a budget which can be used to pay the experts in exchange for the services they provide to the company. Hiring a group of individuals who work together to complete a set of projects is known as Cluster Hire Problem and was first introduced by [1]. This problem is growing rapidly and is one of the biggest challenges faced by online labor markets. Recruiters connect with labor industries to find the best people for the job.

Team Formation concept has been in the industry for quite a while and it involves finding the best combination of experts from a pool of experts to complete a task. In other words, given a set of experts, who possess different skills, we need to select a combination that can complete the given task within the given deadline. On completion of each task, a company gets a profit value in dollars. The whole process consists of a set of experts, who possesses a set of skills and each task requires a set of skills [2]. We select experts who help us to cover the subset of required skills for completing the task. To hire an expert, the organization needs to pay a fee to an expert to provide his/her expertise and an organization is always given a predetermined budget for each task. Recruiters try to minimize this budget which will eventually end up in maximizing profit as they will be able to save a part of given

budget [3].

In addition to this, it is important that we hire experts who are productive as we do not want to hire someone who cannot complete the task in allocated time. The productivity of an expert can be decided based on their experience. For instance, we would prefer an expert who has published more research papers over the one who has published a smaller number of papers. Hiring productive experts will increase the rate of successful completion of the project and will ultimately help the company to earn a higher amount of profit as they will be able to complete more projects.

To make an efficient and productive team, we need to consider the capacity of each expert. The capacity of an expert is the amount of time an expert can provide his/her expertise. We know that no one can work for an unlimited time and every human has a capacity beyond which he/she can not work. We need to consider this aspect while selecting an expert [4]. It is very important to see that we do not overburden any expert as this will help the company to get the best results out of them.

Cluster Hire problem is an optimization problem and is slightly related to existing optimization problems. For instance, let's consider the Job Scheduling Problem. Job scheduling problem has a set of resources that need to be used in order to complete a set of tasks that are maintained in a priority-based queue. The goal is to allocate these already existing resources to these tasks using which it can be completed. In our problem, we have a set of projects which needs to be completed using a set of experts. The difference between the two problems is that in our problem we have to first generate a subset of experts from a set of experts based on various constraints like skillset, expert cost, given budget, expert capacity, profit of the project etc. For example, while selecting an expert we check whether the expert has the skills required by the project or not, we also check if he is capable enough to provide his expertise to complete the given set of projects. We also make sure that the expert is not over-worked, and the cost provided to the expert is within the budget to hire

experts. We consider all such factors to form a subset of experts. Moreover, our problem is a bi-objective optimization problem where we have to maximize the profit and minimize budget. Hence, our problem can be considered slightly related to Job Scheduling Problem.

Let's consider Constraint Optimization, it can be considered as a process that optimizes the given objective based on given variables which are restricted by constraints on them. In our problem, we have a similar situation where we have to optimize the objective function by minimizing the Budget and maximizing the Profit and Productivity. Here the constraint factor is Cost of hiring experts where the cost of hiring experts should never exceed the given Budget. Considering this part of the whole problem we can say our problem is slightly related to Constraint Optimization. What makes it stand apart is that the optimization objective of our problem is bi-objective.

1.2 Motivation

In the real world, forming a team to complete a task has been a challenging task as it involves considering so many factors that can affect the final outcome. A lot of researchers are working in this area to find an optimal solution by stretching it to various dimension. There are certain works that focus on factors such as communication cost [4], compatibility amongst the team members [5], the capacity of an expert [4] etc. to find an optimal solution based on the requirement of the project. Communication cost here means the communication overhead between the team members, compatibility is the ability to which the team members can work with each other without any problems, and capacity of an expert means checking that no expert in the team is overworked.

In today's era, the start-up company is growing rapidly and the owners need to make many important decisions. The owners need to hire many employees to run his/her company. Finding the right group of experts can make a huge difference to the future of the

company. They need to hire a group of experts who can meet the needs of the undertaken projects. This adds up to a huge motivation to our problem.

Another relevant motivation is the consultancy companies. Nowadays, consultancy companies get a lot of projects with specific skills required for its completion. On completion of each project consultancy companies get their share of profit. To complete the projects, the consultancy companies hire a group of experts who can work on more than one project so that their hiring cost is minimized, and profit is maximized. These companies have grown rapidly over the past few years and they also add up to be a huge motivation to our problem. Online labor markets like Guru (www.guru.com) and Freelancer (www.freelancer.com) were the first ones to help companies hire experts to complete projects. Experts register on these portals and when a project comes up they try to find the best set of experts who can work on it and helps the company to complete it.

Team Formation Problem is used for forming a team for a particular task but when it comes to completing more than one task it becomes inefficient as it forms a new team for each task. When it comes to the real world, it is not affordable to form one team for each task and as an improvement [1] introduced Cluster Hire Problem which forms one team that can complete all tasks by working simultaneously. Let's see the problem in detail in the next section.

1.3 Problem Statement

1.3.1 Existing Problem

Cluster Hire Problem was first introduced by [1] and it states that instead of forming one team for each task it finds one team that can cover all tasks. Let's understand it in detail. Let $E = \{e_1, e_2, \ldots, e_n\}$ determines a set of n experts, and $S = \{s_1, s_2, \ldots, s_m\}$ determines

a set of m skills. Each expert e posses a set of skills, which is denoted as $ES(e)$. Clearly, $\forall\, e \in E, ES(e) \subseteq S$. Each expert e demands a monetary cost (i.e., salary), to participate in performing different tasks. This is shown by $C(e)$ and is measured by dollar value. We also have a set of given projects which is denoted by $P = \{p_1, p_2, \ldots, p_k\}$. Each project is also composed of a set of required skills that need to be covered by experts in order for the project to be completed. This set is shown by $PS(p_j)$ for project p. Again, $\forall\, p \in P, PS(p) \subseteq S$ [1].

Finishing each project brings a profit in dollar value which is shown by $PF(p)$ for project p. We are interested to choose a set of projects in which the sum of their profit is maximized.

Given a set of projects P, the profit of completing these projects is defined as follows:

$$Profit(\mathscr{P}) = \sum_{p \in \mathscr{P}} PF(p)$$

The authors in [1] introduced the Cluster Hire problem and were able to solve it successfully under the given circumstances but it had few drawbacks. It did not consider the capacity of the experts. In other words, according to their approach, an expert can be allocated to any number of projects which makes it impractical as one expert cannot provide their expertise for unlimited times. To overcome this problem [4] extended their work by introducing a constraint on a number of times an expert can provide their expertise. They assume that each expert e is able to offer her expertise at most $Cap(e)$ times [4]. This is a reasonable assumption since we do not want to overload an expert by assigning her to many projects. Therefore, each expert has a maximum capacity to participate in a number of tasks. We take this constraint into consideration when designing the algorithms [4].

1.3.2 Our Contribution

In the previous section we discussed about the significant work done by [1] and [4] to solve Cluster Hire problem. In spite of considerable work done by them, there are certain

areas which need to be considered to make it more efficient. While selecting an expert we need to consider the Productivity of an expert as it is important to see that the experts we hire are productive as they hold more chances to complete the project as compared to the experts who are less productive. Let's discuss it in detail.

Each Expert e is assigned a Productivity Score $PR(e)$. This score is determined based on the past performance of the expert. For example, among a group of researchers, the one that publishes the most is considered to be more productive. Clearly, we prefer to have a group of experts that have high productivity. For experts who are new and do not have any productivity score are assigned with an average value of the productivity amongst the group of experts who are competing for the work. This gives the expert a fair chance to be considered to be a part of the final team. The productivity of a group of experts are defined as follows:

Given a group of experts E, the productivity of this group E is defined as follows:

$$Productivity(\mathscr{E}) = \sum_{i=1}^{|\mathscr{E}|} PR(e_i)$$

For performing a subset of given projects, we are given a predetermined budget (also in dollar value) denoted as B. This budget is spent on hiring experts. Our goal is to hire as many experts as possible in which the sum of their hiring costs (i.e., salary) is under the given budget B. Since we are interested to maximize the profit and also maximize the productivity of the group of experts, our problem is a bi-objective optimization problem. A common approach to solve a bi-objective optimization problem is to convert it into a single objective problem. This can be done by introducing a trade-off parameter λ that varies between 0 and 1 and determines whether we want to put more weight towards profit or productivity.

Given a set of n experts E, a set of m skills S, a set of k projects P, a trade off λ between the profit and productivity, we are interested to choose a group of experts $\mathscr{E} \subseteq E$ and a set

of projects $\mathscr{P} \subseteq S$ in which the following objective is maximized:

$$PP(\mathscr{P}, \mathscr{E}) = (\lambda).Profit(\mathscr{P}) + (1-\lambda)Productivity(\mathscr{E})$$

Furthermore, the following budget constraint must be satisfied:

$$\sum_{e \in \mathscr{E}} C(e) \leq B$$

Note that since the dollar values of the projects profit and the productivity of experts will have different scales, both of these values have been normalized before using the above objective so that they both fall into the same range(Between 0 and 1).

1.4 Hypothesis

Given a set of projects where each project requires a set of skills to be completed, a set of experts where each expert possesses a set of skills, capacity, and productivity of experts, a predetermined budget which can be used to hire a team of experts. We need to find a group of experts who can cover all the projects such that the cost of hiring experts is less than or equal to the given budget and we want to maximize the profit. The Profit value for each project is in dollar value and is known beforehand. Cost of hiring an expert is also in dollar values and is termed as the fees we need to pay to an expert to provide his/her expertise. We propose to find a team of experts that can cover all the projects and maximize the profit by using a greedy approach and want to achieve the solution in polynomial time. We plan on applying two different algorithms: expert greedy and project greedy algorithm which we will see in detail in Chapter 3 of the thesis and after applying this we expect to find a team of experts who can cover all the projects by maximizing the productivity of experts and profit of the projects. We also make sure that no expert is overworked and we plan to achieve all this in cubic time in the worst case scenario.

1.5 Objective

Team Formation problem has become a well-known problem and is being researched for about almost a decade now. This problem involves finding a set of individuals that match a given skill set to complete the given task. In our case, we take various factors like profit, capacity, and productivity into consideration. We are given a set of experts who possess different skills, we are also given a set of projects that require a certain skill set to be completed and on completion of which the organization gets a profit value. Our task is to find a group of experts who possess all the skills required to complete all the projects within the given budget. While finding a group of experts we take various factors such as the capacity and productivity of experts.

The capacity of an expert means while assigning an expert to many projects we check if he/she has the potential to complete all the projects in the given time. In other words, we make sure that no expert is overworked as no one can provide his/her expertise for unlimited times. Any individual has a limit up to which he/she can work, and it is important that we take it into consideration while hiring an expert. For example, we have 10 projects that need to be completed and we have 15 experts. If we choose an expert who possesses skills that matches to all the 10 projects and we assign him to all of them. It will be practically impossible for that expert to give each project equal attention and this will lead to a bad result for the company and in the worst case, the project will never be completed. To avoid such circumstances, it is very important to maintain the balance between the team members and make sure that everyone is given an equal amount of work.

The productivity of an expert is defined in terms of the performance of an expert. When we hire an expert, we would prefer an expert who is more productive as he/she will stand more chances of providing successful results. The productivity of an expert in our case is defined based on their past performance. An expert who has published a greater number of

research papers is more productive than an expert who has published a smaller number of research papers. It is an important factor in our problem as we need to maximize profit and productivity of an expert is directly related to profit because if we hire a more productive expert there are more chances of completing the project and hence company will get more profit.

Overall, to complete a set of projects we are given a predetermined budget to hire experts. Our goal is to hire as many as experts possible such that the hiring cost of experts is less than the given budget and make sure that no expert is overworked. We want to maximize the profit and minimize the budget and this makes our problem a bi-objective optimization problem.

1.6 Structure of Thesis

The rest of the thesis is organized in the following way: In chapter 2, we do the background study of our thesis. We discuss the basic and fundamental concepts. It also includes a detailed discussion of related work and literature review. In chapter 3, we discuss our proposed approach in detail with illustration to understand it better. Chapter 4 includes the experiments done in order to see the performance of our algorithms and validate the results. Chapter 5 includes the conclusion and future work.

Chapter 2

Background Study

2.1 Key Concepts

In this section, we will discuss the key concepts to understand the research work done in this thesis. We will first discuss the research area of the thesis and then we will describe what is Team Formation, Cluster Hire and the difference between Team Formation and Cluster Hire Problem.

2.1.1 Data Mining

Data Mining is a process of finding interesting and meaningful patterns from large data sets [6]. Data mining is a subfield of Computer Science with a goal to extract meaningful data from a dataset which can be used in future to take strategic decisions. Each and every sector of the industry is going paperless and with the increase in the volume of the data generated it is necessary to have tools that can process this data [6]. Processing data here means the ability to find valuable information form the whole lot of data present in any companys database. Data Mining is a part of Knowledge Discovery in Database process. The KDD process is shown in figure 2.1 it is commonly defined by the following stages [8]:

1. Data Cleaning: It is a process of removing irrelevant and noisy data from the collection.

2. Data Integration: Combining data from different sources into a single source by making it consistent.

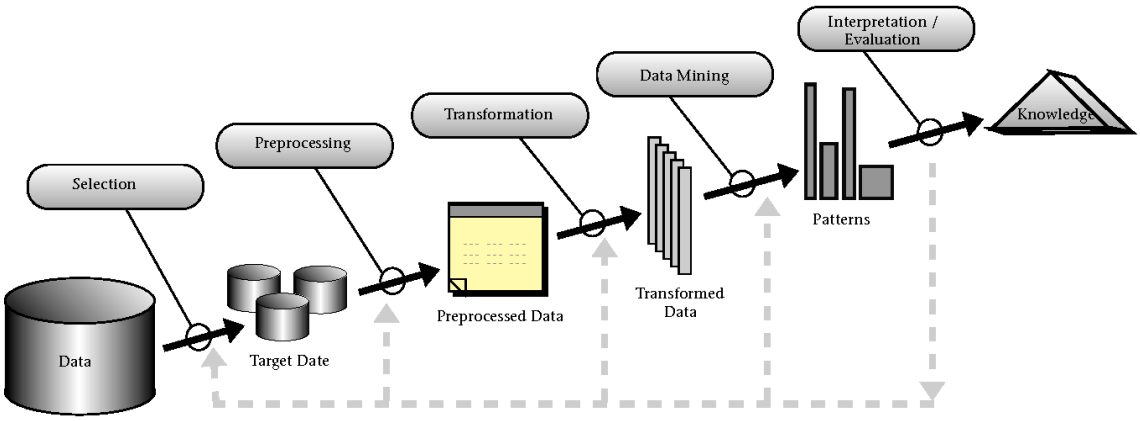

Figure 2.1: Knowledge Discover in Database Process [7]

3. Data Selection: It is a process of selecting and retrieving relevant data to conduct the analysis from the data warehouse.

4. Data Transformation: It is a process of transforming the data into a form processable by the data mining techniques.

5. Data Mining: It is a technique of extracting relevant and important patterns which can be used by the high-level authorities to make business strategies.

6. Pattern Evaluation: This stage helps us to decide how relevant are the patterns and tells about the interestingness of the discovered patterns.

7. Knowledge Representation: This stage helps to look at the interesting patterns that are discovered using data mining techniques. It becomes very easy for the users to look at the pattern and decide on the possible outcomes and helps in decision making.

2.1.2 Team Formation

It is a process of finding a set of individuals that can work together to complete a task. The selection of individuals can be done on the basis of the skills possessed by them. It

depends on the skillset required to complete the given task and no. of people needed to complete it. Working in a team is a lot different than working individually as it requires each member of the team to work together without any problems [9]. We have to find individuals who can work together to complete any task and at the same time, we have to make sure they are compatible with each other. For example, if we are looking to form a soccer team, in particular, the forward players. The required skills will be physicality, dribbling skills, powerful shot, accuracy, and pace. We need to find 3 players to form the forward of the team. Lets consider we have 5 candidates who possess all the skills. While shortlisting them we will need 3 players who run at a similar pace as it becomes easy to play one to one when you are at a similar pace. It might be possible that the 2 left out candidates are far better than the selected candidates individually but under the given circumstance they might fail as a team.

2.1.3 Cluster Hire

Cluster Hire problem means hiring a group of experts who can collectively as a team complete a set of projects. In other words, we find a team of experts who can work on more than one project simultaneously to complete them [1]. Here the hiring of experts depends on the budget, profit value that will be achieved by completing the project, skills possessed by an expert, the skills required to complete a project, the capacity of the expert, productivity of expert etc. We have to make sure that we find a cluster of experts who hold a maximum chance of completing all the projects under the given constraints. For example, we have 5 projects that require a variety of skills to be completed. The company gets a profit value on completion of each project. We need a team which can complete all 5 projects. So, from a pool of experts with different skills, we select a team that can complete all the projects within the given budget. This process is known as Cluster Hire.

2.1.4 Difference between Cluster Hire and Team Formation

Cluster Hire and Team Formation are two different problems. Team formation problem finds a team for each task independent of other available tasks. Whereas, in Cluster Hire problem we form a single team that can complete all available tasks.

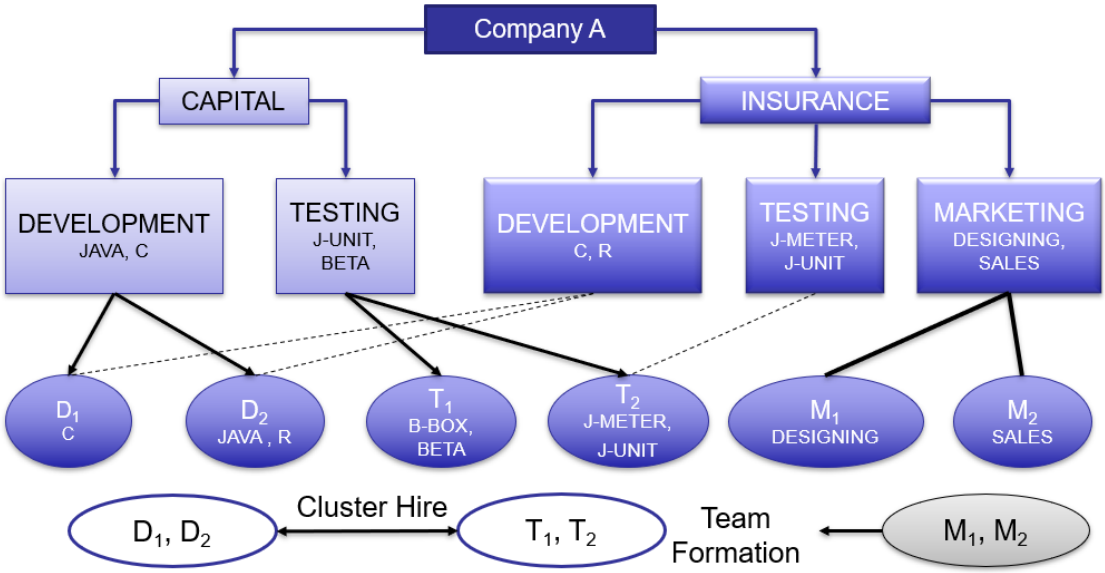

Figure 2.2: Difference between Cluster Hire and Team Formation

Lets understand this using a simple example. In figure 2.2 we have demonstrated a scenario of a random Company A. The company has two main sub-divisions Capital and Insurance. These sub-divisions are further divided into different departments. Capital has the Development and Testing department whereas Insurance has Development, Testing, and Marketing department. Each department has required skills attached to them that are needed to complete the projects. For example, in sub-division Capital development department needs an expert who possesses Java and C. Further we have experts D1, D2, T1, T2, M1, and M2. They possess different skills and based on their skills they will be assigned to different departments. So, hiring D1, D2, T1, and T2 collectively to meet the needs of departments of sub-division Capital and Insurance is known as Cluster Hire whereas hiring

M1 and M2 to meet the requirement of Marketing department of sub-division Insurance is known as Team Formation.

2.2 Fundamental Concepts

In this section, we will discuss the fundamental concepts that are used in this thesis and will help us to understand it better. We first discuss heuristic function then we know about the exact algorithm. We then understand the greedy algorithm and the approach used in the greedy algorithm. Finally, we discuss the bi-objective optimization function.

2.2.1 Heuristic Function

Heuristics are used to provide the solution to the problems that are NP-hard to solve. When there is no possible solution that can be found in reasonable time, we use heuristics to find its solution [10]. In other words, heuristic helps to find the solution in a reasonable amount of time, but it compromises with the optimality of the solution. The main advantage of using heuristic is that it does not need a lot of computer time and it produces results quickly. The results produced using heuristic can be optimal or near optimal.

2.2.2 Exact Algorithm

The exact algorithm finds a solution to a problem optimally. The results produced by the exact algorithm is most optimal [11]. The main drawback of this approach is that it can not be used when the search space is huge. It takes into consideration all possible combinations to reach the solution. The solution generated by this algorithm compromises with the runtime, but it produces an optimal solution. When dealing with huge search space it fails to produce results is feasible time [11]. Moreover, to calculate the solution of the problem with the large search space we need a lot of computer power and it is very costly.

2.2.3 Greedy Algorithm

Greedy Algorithm solves the problem using heuristics. It is capable of solving NP-hard problems in feasible time at an affordable cost. It produces optimal or near to optimal solutions [12]. It works in a step by step procedure where at each step it chooses a solution that is optimal at that given point of time. It finds a locally optimal solution and proceeds further until it reaches its goal. It considers the best choice available at that point and proceeds further but it never reconsiders the choice it has made, and this is one of the major drawbacks of this approach. Generally, there are five components that are used to solve a problem using the greedy algorithm [13]:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a solution, or a partial solution
5. A solution function, which will indicate when we have discovered a complete solution

Figure 2.3: Solution to coin change problem using greedy algorithm [14]

Lets understand it with an example. If we are given 1, 5, 10, 25, and 50 cents coins and we want to make a change of 41 cents such that we use a minimum number of coins. The greedy algorithm will try to use most of the 25 cents coins as it is the maximum value coin. It will then try to use the next highest coin which is 5 cents and so on until it reaches the solution which is 41. The figure 2.3 shows the optimal solution to the problem which can be achieved using the greedy approach.

2.2.4 Bi - Objective Optimization Function

Bi-objective optimization means optimizing two objectives at the same time using a single function. Such functions are useful when there are more than one deciding factors [15]. In our case, we have two deciding factors which is why we use a bi-objective function. The function balances both objectives using a balancing factor and the contribution of each objective to the final score depends on the balancing factor. Lets understand it using a simple example.

Let SCORE = (BF * (Objective 1)) + ((1 BF) * (Objective 2)) where BF is balancing factor and its value lies between 0 and 1. Depending on the value of BF the importance of the Objectives can be balanced. In other words, if the value of BF is higher then the contribution of Objective 1 in the score value will be higher and if the value of BF is low then the contribution of Objective 2 will be higher.

2.3 Literature Review and Related Work

In this section, we will discuss the work done by other researchers in the past and how our work is related to their work. Cluster Hire and Team Formation problem are closely related to each other but are two different problems. We will discuss the work done in both team formation and cluster hire problem and see how they are different from each other.

We will start with the work done in team formation by various authors and then we will discuss the work done in cluster hire problem and at the end, we will discuss the paper that is most closely related to our topic.

The authors in [16] addresses Team Formation problem in a Network of Experts. Team Formation problem has been researched by many researchers and they have considered various parameters to find the best possible teams. In this paper, authors [16] have considered two new factors: Personal Cost and Communication Cost of the experts. Personal cost is the cost of hiring an expert to provide his/her expertise and Communication Cost is the cost considered for communicating between the team members. For example, if there are two experts who are physically present in two different countries then the cost to communicate with each other is known as communication cost [16]. They have proposed two approximation algorithms. The first approach presents a budget on one objective and they minimize the other objective and in the second approach, they present a set of Pareto-optimal solutions in which there is no other team that dominates in both the costs [16]. Given a project P and a graph G representing a network of experts the task is to find a team for P from G such that the Communication Cost between the team members and the personal cost of the experts is minimized. Finding a team of experts while minimizing communication cost is an NP-hard problem and is proved to be NP-Hard by [17]. This is a bi-criteria optimization problem and authors have proposed two variations of minimizing communication cost which we will discuss in detail. The first approach proposes an alpha-beta approximation function where the first parameter alpha means that in our bi-objective minimization problem the first objective is at most alpha times the budget and the second parameter is at most beta times the minimum distance [16]. In this approach the authors use two different functions: first considers diameter and the second considers the sum of distance for the communication cost function. The algorithm looks for the expert who possesses the rarest skills and then they create a pool of expert who possesses the required skills and once this

pool is ready it greedily picks experts one by one such that the bi-objective criteria are minimized. The second algorithm finds Pareto optimal teams based on diameter and personnel cost given to it. It generates many teams that meet the required skills for the project, but they are distinguished based on their dominance.

Given a set of required skills to build teams of experts has been examined in many studies. [5] first introduced the discovery of a team of experts from a social network. Then, the authors of [17] tested a new function called the sum of the distance to find the best teams. Later, [18] introduced another cost function based on the density of the induced sub-graph. The contribution by [19], [16] and [2] are significant in order to have variant research of team formation problems.

In addition to this, the team formation problem was tackled by evolutionary computations in order to handle the complex expert network. The authors [20] applied Genetic Algorithms to discover teams of experts and considered the geographical location of each member of the team while optimizing the approach. Recently, the authors [21] considered the team formation problem in the health care setting and used Cultural algorithms to optimize multi-objectives.

The authors of [22] considered a set of experts where each expert is associated with a set of skills and a collection of projects arriving one at a time in an online form. [1] proposed Cluster Hire problem for the first time, and followed the similar concept of [22]. But, the difference was that they didn't choose projects from online and generated a single team that can perform many projects. Recently, the authors of [4] extended the work of [1] by considering the previous collaboration among experts and optimizing the communication cost among experts. Probably our work is most related to [4]. But the significant difference is that we didn't consider the communication cost between team members since we aim to get more productive experts which seem to be more important than previous collaboration in online freelancing work.

The authors [22] have addressed the Online Team Formation Problem in Social Networks. Team formation as we all know has been studied widely but in this, the author tackles the team formation problem in social networks where the tasks are received online. Here the problems that need to be solved involves finding a team for each task that is received such that the members of the team possess the required skills to complete the task undertaken. We also need to make sure that each member of the team has been given a similar workload and no member is overburden by the work. In other words, load balancing for the team should be kept into consideration while selecting the members for the team. Authors in [22] claims that until this paper they were the first one who has solved the problem of online team formation by considering all aspects such as workload balance, all skills required by the task can be covered by the selected members and the communication overhead is minimum. In past, many authors have tried to solve the team formation problem but due to the nature of the problem it has a lot of variations and it is very difficult to cover all the aspects involved in this area. Authors in [5] have considered communication cost and they have minimized the communication cost between the experts but the main challenge that they faced was to tackle the social network when it is not connected. Due to a large number of experts in the real world, there are very fewer chances that we find experts who have worked together in the past and to solve this problem they have used a reference expert who has worked in the past with the other two experts. There are cases when some expert leaves a team in the middle of the project and there must be a way in which the expert can be replaced.

The authors in [2] address the Team Formation problem in social networks. Team formation problem has been researched by many researchers and has been studied widely. Team formation problem is finding a group of experts from the social network who can cover all the skills required to complete a project. While discovering a team of experts there are various factors that need to be considered to get the best results. Some of the fac-

tors are communication cost, the capacity of experts, personal cost etc. [16]. In this paper, the author [2] has contributed a new factor Authority of the experts while selecting a team of experts. The authors [2] have formulated various function that combines communication cost and authority of experts. Authors in this paper are extending a team formation problem which involves discovering a team of expert that can cover all the required skills to complete a project. The selection criteria involve minimizing the communication cost. Communication cost is the cost spent for all team members to communicate amongst themselves. Authors [2] extend this problem by considering the authority of authors. Authority of an author is determined from a graph where the nodes represent an expert and the node labels represent their area of expertise. The authors propose a greedy algorithm that minimizes the communication cost and considers the authority of the experts while selecting a team of experts. Authors have proposed a function that takes Communication cost and connector authority into account. Its optimized value is then given to another function that takes this optimized value and skill holder authority into account and optimizes it. Using this function, the author discovers a team to solve this problem.

The authors [23] have addressed Online Search for the overlapping communities. A lot of research has been conducted in discovering and modeling communities in large complex networks. A community refers to a group of vertices that are densely connected to each other and sparsely connected to other vertices in the graph. Overlapping Community Detection (OCD) means finding overlapping community in the entire network whereas Overlapping Community Search (OCS) means finding an overlapping community where a specific vertex belongs to [23]. The authors have shown that OCD is a very long and slow process as compared to OCS which is fast and efficient. Authors have proposed various algorithms to find the Overlapping Community in the network.

The author [3] have addressed Profit-driven Team Grouping in Social Networks. We are given a graph which has information of experts who are socially connected. Here nodes

are experts and the edge between any two nodes indicates that the two experts are socially connected. The weights on these edges indicate the communication overhead between the experts [3]. Communication overhead is the cost that needs to be spent in order to make the two experts work together. We are given a set of tasks where each task needs a set of skills to be completed. To complete these tasks, we have a set of experts who need to be assigned to different task such that we are able to complete all tasks and on completion of each task company gets a profit value. Author solves this problem by proposing an LP-based approximation algorithm. The author also makes sure that the load balance within the team members is maintained and no expert is over-worked.

The authors [24] have addressed the Team Formation Problem for Participatory Tasks that considers Social Connections between experts. The performance of any collaborative task depends on the contribution made by each team member collectively. To achieve a participatory task viably and productively, the Team Formation Problem (TFP) outweighs all other considerations. It is even more complicated when social connections amongst the experts are taken into consideration. It is a challenging task to find a group of experts that can complete a task effectively and efficiently as each task has its requirements and uniqueness [24]. Moreover, factors such as diversity of skills possessed by each individual and social connection between them also play an important part in the selection of a candidate. In this paper, the author has proposed two algorithms: Team Formation- Strong Ties (TFP-ST) and Team Formation-Weak Ties (TFP - WT). Here the strong and weak ties are the social bonds between any two candidates [24]. Socially strong candidates will have a Strong-Tie whereas socially weak bond between any two candidates indicates Weak Tie. To solve this problem author has proposed two algorithms: TFP-ST (Team Formation Problem Strong-Tie) and TFP-WT (Team Formation Problem Weak Tie) to solve the team formation problem in the social network. The TFP ST problem needs to find a team which meets the requirement of the success ratio for every task. This problem is

proved to be NP-Hard [24] there does not exist any polynomial approximation solution, so the authors of this paper provide a solution based on Group Steiner Tree Problem. The algorithm assumes that to complete a Project P there has to be a set of task T that needs to be completed [24]. The algorithm adds virtual candidates and virtual edges to the graph. The weight of the virtual edges should be larger than the sum of weights of all edges in the graph. This will prevent it to get selected by the shortest path criteria. The first candidate is chosen based on the shortest path criteria and the selected candidate should meet the requirement of the task undertaken. This candidate should possess the rarest skill as we do not want to miss the candidates who have rarest skills. It then chooses the candidates that meet the requirements of the task and they should have a success ratio greater than the set value which in the paper is taken as 80 percent. Once all candidates are chosen a clean up process is initiated to remove those candidates who have redundant skills. This is done to make sure we do not have more than one candidate doing the same task as it will affect the efficiency of the task. The time complexity of this algorithm is proved to be cubic in the paper.

Authors in [1] were the first one to address the Cluster Hire Problem. Given a pool of Projects P where each project requires a set of skills for its completion, given a pool of expert E where each expert posses a set of skills and charges a cost C to provide his/her expertise, a budget B is given to hire a group of experts. Hiring a group of experts who can work together to complete all projects within the given budget B is known as Cluster Hire Problem [1]. Authors have presented two algorithms to solve this problem, Expert Greedy Algorithm and Project Greedy Algorithm. The problem takes a set of projects, set of experts, a budget to hire experts and profit the organization will achieve after completion of each project. Each expert posses a set of skills and each project requires a set of skills to be completed. Authors [1] have a profit function and they are maximizing it in their algorithm. Their aim is to greedily choose one expert at a time such that all projects are covered by the

experts and the least amount of money is spent on hiring a team of experts. This makes it a bi-objective function where they are minimizing budget and maximizing profit. The algorithm greedily picks one expert at a time and assigns him/her to all those projects where the required skill set matches with the experts skills. This process is continued until it reaches a stage where all projects are completed. The second algorithm greedily picks one project at a time. Here the approach changes, the algorithm chooses one project at a time and then it looks for the best expert who can work on this project using a score function. Expert with a high score are considered more effective and are expected to produce better results. This is done until all projects are covered or the given budget is exhausted.

The Expert Greedy Algorithm [1] starts with an empty team and it selects one expert at a time according to required skills keeping budget constraint in mind. It selects one expert and performs various more iteration to have enough experts that can complete the project in an assigned budget. In cases where budget clashes the algorithm chooses an expert randomly from the ones that are clashing. The Project Greedy Algorithm [1] starts with an already formed team in its iteration and this selects project greedily. It then checks for missing skills in a team to cover the project. According to the missing skills it then hires few more experts that can work with the existing team keeping the total cost in the budget. The Clique Greedy Algorithm [1] is an extension of project greedy algorithm which chooses one project in one iteration. This was designed to overcome limitations of project greedy algorithm. It forms a group of a number of projects based on the skillsets required. Project with similar kind of skill set requirement is grouped together. Two projects can be under one group only if it satisfies the compatibility factor which is aimed to maximize profit. It then forms a graphical structure where each project is a node and each edge show that those two projects are compatible. Based on the nodes further decision can be taken. As we mentioned earlier, Cluster Hire problem was introduced by [1]. Authors in [4] extended the work done by them. Given a set of projects and a set of available experts, the

authors found a subset of projects along with a subset of experts to perform the projects while maximizing the profit of projects and not violating the given budget constraint. For performing a subset of given projects, we are given a predetermined budget (also in dollar value) denoted by B. This budget is spent on hiring experts. Our goal is to hire as many experts as possible while the sum of the hiring costs (i.e., salary) is under the given budget B. Since we are interested in maximizing the profit and minimizing the communication cost, our problem is a bi-objective optimization problem. A common approach to solving a bi-objective optimization problem is to convert it into a single objective problem. This can be done by introducing a trade-off parameter that varies between 0 and 1 and determines whether we want to put more weight towards profit or communication cost. Furthermore, since one of the objectives is a maximization problem (maximizing the profit) and the other one is a minimization problem (minimizing the communication cost), they have modified one of them and made both of them of the same type. Therefore, they maximize the reverse of the communication cost. Our work can be most related to the work done by [4] but the significant difference is that we have considered the productivity of the expert over communication cost as it is more important to have a productive expert as compared to the previous collaboration of the experts.

Chapter 3

Proposed Algorithm

In this chapter, we propose two different algorithms based on different strategies to find a group of experts while maximizing the profit and productivity. The first strategy picks an expert in each iteration and assigns her to some of the projects. The second strategy picks a project in each iteration and finds the best group of experts to finish that specific project.

3.1 Cluster Hire with Expert Pick Strategy

Since finding the best group of experts to cover a subset of projects while maximizing the profit and productivity is an NP-hard problem, here we propose the first greedy algorithm to find a group of experts while covering a subset of projects with high profit. In the first algorithm and in each iteration, we greedily pick one expert and add her to the pool of existing experts. We also check to see if adding her will cover any of the remaining projects. One important challenge is to make sure adding a new expert (with her salary) does not exceed the given budget B. In each iteration, we assign a score to each pair of expert/projects and choose the expert with the highest score. The score is designed based on the following intuitions:

- We want to choose a *cheap expert* so that we do not overspend the budget on a single expensive expert.
- We want to choose an expert that covers many required skills for a *high profitable* project.
- We want to choose an expert with *high productivity*.

Algorithm 1 Cluster Hire with Expert Pick Strategy

Input: set of n experts $E = \{e_1, e_2, \ldots, e_n\}$, set of m skills $S = \{s_1, s_2, \ldots, s_m\}$, set of k projects $P = \{p_1, p_2, \ldots, p_k\}$, tradeoff parameter λ, Productivity, Capacity of each expert, each Project's Skillset, each Expert's skillset and budget B.
Output: subset of projects $\mathscr{P} \subseteq P$ and a group of experts $\mathscr{E} \subseteq E$ that maximize $PP(\mathscr{P}, \mathscr{E})$ under the given budget B.

$\mathscr{E} \leftarrow \emptyset$, $\mathscr{P} \leftarrow \emptyset$, $b \leftarrow 0$
while $b < B$ and $E/\mathscr{E} \neq \emptyset$ **do**
  $\mathscr{R} \leftarrow \{e \mid e \in E \text{ and } e \notin \mathscr{E} \text{ and } C(e) + b \leq B\}$
  for all $e \in \mathscr{R}$ **do**
    if e does not cover any required skills in $P/\mathscr{P}$ **then**
      remove e from $\mathscr{R}$
  if $\mathscr{R} = \emptyset$ **then**
    return $\mathscr{E}$ and $\mathscr{P}$
  for all $p \in P/\mathscr{P}$ **do**
    for all $e \in \mathscr{R}$ **do**
      if e covers at least one skill in p **then**
        $sc_e^p \leftarrow \lambda . \frac{PF(p).\min\{Skill(e,p),\ Cap(e)\}}{C(e)} + (1-\lambda).PR(e)$
      else
        $sc_e^p \leftarrow 0$
  $(e, p) \leftarrow \arg\max_{e \in \mathscr{R},\ p \in P/\mathscr{P}}\ sc_e^p$
  add e to $\mathscr{E}$
  assign skills of e to p based on rarest skill strategy
  update $Cap(e)$
  update $PS(p)$
  $b \leftarrow b + C(e)$
  if $|PS(p)| = 0$ **then**
    add p to $\mathscr{P}$
  while $Cap(e) > 0$ **do**
    $p' \leftarrow \arg\max_{p \in P/\mathscr{P}}\ score_e^p$
    assign skills of e to p' based on rarest skill strategy
    update $PS(p')$ according to $ES(e)$
    update $Cap(e)$
    if $|PS(p')| = 0$ **then**
      add p' to $\mathscr{P}$
return $\mathscr{E}$ and $\mathscr{P}$

As one might notice, these intuitions are not necessarily compatible. A cheap expert may not be able to cover many skills from a profitable project or she may not have high productivity. In order to take into account all these objectives, we design the following score function for each pair of projects (i.e., p) and experts (i.e., e).

$$sc_e^p \leftarrow \lambda.\frac{PF(p).\min\{Skill(e,p),\ Cap(e)\}}{C(e)} + (1 - \lambda).PR(e) \quad (3.1)$$

Recall that λ is the tradeoff parameter between profit and productivity (see Problem Statement). Note that $Skill(e,p)$ determines the number of skills in p that could be covered by expert e. Note that after assigning an expert to the pool of existing experts, her capacity is updated based on the projects she participates in. The first part of the above equation chooses a pair of the expert/project in which the project p has high profit and the expert e covers as many skills as possible in p. This number is divided by the cost of expert e to ensure we take into account the salary of expert e. Between the number of skills that expert e can cover in p and the capacity of e, we choose the minimum value. This is because we do not want to violate the capacity of expert e and overload her with many tasks. For example, if an expert is able to cover 5 skills in a project, but her capacity is only 3, we use 3 in the above equation to make sure if she is the selected expert with that project, she is only assigned to 3 skills and not 5. The next part of this equation maximizes the productivity of the expert.

Algorithm 1 is our solution to our Problem to find a group of experts while maximizing the profit of covered projects. This algorithm receives the set of n experts, set of m skills, a set of k projects, the tradeoff parameter λ, and the available budget B as input. The output of this algorithm is a subset of projects $\mathscr{P}$ and a group of experts $\mathscr{E}$ that covers all required skills in $\mathscr{P}$ while maximizing the objective of Problem. Furthermore, the sum of the salary of the experts in $\mathscr{E}$ is not more than the given budget B.

In line 1, $\mathscr{E}$ and $\mathscr{P}$ are initialized to $\emptyset$. Also, b is set to 0. We store the amount of

money that we have spent so far in b. We can keep adding experts to $\mathscr{E}$ as long as $b < B$. Line 2 starts the iteration of the greedy algorithm. As long as we have experts in E that have not been added to $\mathscr{E}$ and we have not overspent the budget, we can consider adding more experts to $\mathscr{E}$. This is the condition of the while loop in line 2. In line 3, we store all unassigned experts in E in which their salary is within the remaining budget to $\mathscr{R}$. The for loop in line 4 checks to see each expert in $\mathscr{R}$ satisfy at least one required skill in an uncovered project. If this is not the case, the expert is removed from $\mathscr{R}$ as these experts are useless for the remaining projects. In line 7, we check to see if $\mathscr{R}$ is empty or not. If it is empty, we return the current $\mathscr{E}$ and $\mathscr{P}$ and terminate the algorithm. The reason is, if $\mathscr{R}$ is empty, we have no other experts to add to $\mathscr{E}$. In lines 9 to 14, we assign a score to each pair of uncovered project p (projects in $P/\mathscr{P}$) and expert e in $\mathscr{R}$. Later, we choose the highest score and add the associated expert to $\mathscr{E}$. If e does not cover any of the required skills in p (line 11), the score is set to 0 (line 14) as expert e is useless for project p. If e covers at least one of the required skills in p, then we calculate the score of e and p according to Equation 3.1. In line 15, we choose pair (e, p) in which their score sc_e^p is maximized among all other pairs. e is added to $\mathscr{E}$ in line 19. Then, the skills of e are assigned to p in line 17. Note that if the capacity of e is smaller than the required number of skills in p, we assign the rarest skills in e first. We then update the capacity of e, the required skills in p (i.e., $PS(p)$), and the value of b. If all of the required skills in p are covered (line 24), p is added to $\mathscr{P}$ (line 22).

One advantage of our proposed algorithm is that if an expert e is added to the group of experts, we try to use her maximum capacity as she will get paid the same amount of salary regardless of the number of skills she covers in different projects. Based on this strategy, after expert e is selected to be added to $\mathscr{E}$ in the current iteration, we assign her remaining capacity to other projects in lines 23 to 29. As long as her capacity is larger than zero (line 23), we find a project p' that maximizes the expert/project score when the expert e is fixed

(line 24). We then assign the skills of e to p' and update $PS(p')$ and the capacity of e. If all required skills of p' are covered, it will be added to $\mathscr{P}$. As we are considering a lot of factors such as Productivity, maximizing profit and minimizing the cost our algorithm's worst case running time is $O(nmk)$. In other words, the worst case run time of our algorithm is cubic.

Let's understand the algorithm using a working example. We are given a set of Projects, Skills required to complete the project and the profit gained after completing it in Table 3.1. In Table 3.2, we are given the details of experts with their possessed skills, cost they will charge, expert's capacity and the productivity of the experts. Our task is to find a team from Table 3.2 who can cover all the projects in Table 3.1 within the given budget 150.

Project	**Skills**	**Profit**
P1	S1, S3, S6	180
P2	S2, S3	240
P3	S1, S4, S5	120

Table 3.1: List of Project with required Skills and Profit

Expert	**Cost**	**Skills**	**Capacity**	**Productivity**
E1	50	S1, S4	2	4
E2	40	S6, S3	1	3
E3	60	S4, S5	4	4
E4	30	S1, S2, S3	6	3
E5	30	S3, S5	3	2

Table 3.2: Profile of Experts

We can clearly see that Cost, Profit and Productivity are all on a different scale and if we use these values to calculate score it will not give equal weight to each factor we are considering. To bring them on the same scale we have used min-max normalization that will help us to convert Profit, Cost, and Productivity in a range of 0 and 1.

Min-Max normalization is used to re-scale the features to a range between 0 and 1 inclusive.The figure 3.1 explains the process of calculating normalized value. In our case, we use it to convert the values of Cost, Profit, and Productivity. The normalized value is used

only for score calculation and selection process.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Figure 3.1: Min-Max Normalization [25]

3.1.1 Illustration of Expert Pick Strategy using Normalized Values

In this section, we will see the working example of our algorithm. The Table 3.3 and Table 3.4 shows the information of Projects and Experts with Normalized Value. The values are converted using the information in Table 3.1 and Table 3.2.

Project	**Skills**	**Profit**
P1	S1, S3, S6	0.6
P2	S2, S3	1
P3	S1, S4, S5	0.2

Table 3.3: List of Project with required Skills and Profit

Expert	**Cost**	**Skills**	**Capacity**	**Productivity**
E1	0.75	S1, S4	2	1
E2	0.5	S6, S3	1	0.66
E3	1	S4, S5	4	1
E4	0.25	S1, S2, S3	6	0.66
E5	0.25	S3, S5	3	0.33

Table 3.4: Profile of Experts

Process: We calculate score for each project and expert pair and select the pair that has the highest score. The Table 3.5 shows how the score is calculated. Here's our score

function that we use to calculate scores:

$$sc_e^p \leftarrow \lambda.\frac{PF(p).\min\{Skill(e,p),\ Cap(e)\}}{C(e)} + (1 - \lambda).PR(e)3.1 \quad (3.2)$$

	P1	**P2**	**P3**
E1	((0.5) * (0.6 * min{1,2}) / 0.75 + (0.5 * 1)) = 0.9	0	((0.5) * (0.2 * min{2,2}) / 0.75 + (0.5 * 1)) = 0.766
E2	((0.5) * (0.6 * min{2,1}) / 0.5 + (0.5 * 0.66)) = 1.53	((0.5) * (1 * min{1,1}) / 0.5 + (0.5 * 0.66)) = 1.33	0
E3	0	0	((0.5) * (0.2 * min{2,4}) / 1 + (0.5 * 1)) = 0.7
E4	((0.5) * (0.6 * min{2,6}) / 0.25 + (0.5 * 0.66)) = 2.73	**((0.5) * (1 * min{2,6}) / 0.25 + (0.5 * 0.66)) = 4.33**	((0.5) * (0.2 * min{1,6}) / 0.25 + (0.5 * 0.66)) = 0.73
E5	((0.5) * (0.6 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 1.365	((0.5) * (1 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 2.165	((0.5) * (0.2 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.5: Iteration 1 Score Calculation

After calculation of the score, we select the expert who has the highest score. In Table 3.5, Expert E4 has the highest score for Project P2. We select Expert E4 and assign to all the Projects that need to be completed until they reach their maximum capacity. Table 3.6 shows the remaining skills that need to be covered after Iteration 1 and Table 3.7 shows the skills that are covered in Iteration 1.

Project	Remaining Skills
P1	S6
P2	–
P3	S4, S5

Table 3.6: Status after Iteration 1

Team	Project	Skills Covered
E4	P1	S1, S3
	P2	S2, S3
	P3	S1

Table 3.7: Skills Covered after Iteration 1

Remaining Budget after Iteration 1 = 150 - Cost of Expert E4 = 150 - 30 = 120

In Iteration 2 we follow the same process. Table 3.8 shows the score calculation for Iteration 2.

	P1	**P3**
E1	0	((0.5) * (0.2 * min{1,2}) / 0.75 + (0.5 * 1)) = 0.63
E2	**((0.5) * (0.6 * min{1,1}) / 0.5 + (0.5 * 0.66)) = 0.93**	0
E3	0	((0.5) * (0.2 * min{2,4}) / 1 + (0.5 * 1)) = 0.7
E5	0	((0.5) * (0.2 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.8: Iteration 2 Score Calculation

Table 3.9 shows the skills that remains to be covered after Iteration 2 and Table 3.10

shows skills covered in Iteration 2.

Project	Remaining Skills
P1	–
P2	–
P3	S4, S5

Table 3.9: Status after Iteration 2

Team	Project	Skills Covered
E4	P1	S1, S3
	P2	S2, S3
	P3	S1
E2	P1	S6

Table 3.10: Skills Covered after Iteration 2

Remaining Budget after Iteration 2 = 120 - Cost of Expert E2 = 120 - 40 = 80

In Iteration 3 we again follow the same procedure. Table 3.11 shows the score calculation in Iteration 3.

	P3
E1	((0.5) * (0.2 * min{1,2}) / 0.75 + (0.5 * 1)) = 0.63
E3	((0.5) * (0.2 * min{2,4}) / 1 + (0.5 * 1)) = 0.7
E5	((0.5) * (0.2 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.11: Iteration 3 Score Calculation

Table 3.12 shows the remaining skills that need to be covered and Table 3.13 shows the skills covered after Iteration 3.

Project	Remaining Skills
P1	–
P2	–
P3	–

Table 3.12: Status after Iteration 3

Final Team	Project	Skills Covered
E4	P1	S1, S3
	P2	S2, S3
	P3	S1
E2	P1	S6
E3	P3	S4, S5

Table 3.13: Skills Covered after Iteration 3

Remaining Budget after Iteration 3 = 80 - Cost of Expert E3 = 80 - 60 = 20

As we can see in Table 3.12 there are no more skills that need to be covered. We now have the Final Team that can cover all the Projects in Table 3.1. Table 3.13 shows the Final Team of Experts and the Skills they cover. Total Budget spent on hiring experts is 120.

3.2 Cluster Hire with Project Pick Strategy

The second algorithm to find a group of experts to cover the most profitable set of projects is designed based on the idea of selecting a project in each iteration. In each iteration, we assign a score to each uncovered project and choose the one with the highest score to be added to the pool of projects. The score of each project is designed based on the following intuitions:

- We want to choose a project with *high profit*.

- The set of experts responsible to cover the required skills in the project should be *cheap*.

- The set of experts that cover the skills of the project should be *productive*.

The same as the first strategy, these intuitions are not necessarily compatible. A high profitable project might need an expensive set of experts and/or non-productive set. We design the scoring function that takes into account a combination of all these objectives. In each iteration and for each uncovered project p, we find a set of experts E_p to cover the required skills of p. In order to do that, we use a modified version of the greedy weighted set cover algorithm. Recall that in greedy set cover, we are given a collection of sets (corresponding to the set of skills of each expert) in which each set is associated with a cost (corresponding to the salary of the expert in our problem). The goal is to choose a subset of sets to cover a given union set (corresponding to the set of skills required for a given project in our problem). In the greedy weighted set cover algorithm, in each iteration, a set that maximizes the number of covered elements divided by the cost of the set is selected. In other words, the algorithm selects a set, in which the price of covering a single element is minimized. We also add the productivity to the price per skill when selecting the next expert to cover a given project. Formally, in each iteration, and for any remaining project, we find a set of experts that are able to cover that project. To find this set for project p, we start with an empty set E_p. Then, we select an expert to be added to E_p that maximizes the following equation:

$$sc_e \leftarrow \lambda.\frac{\min\{Skill(e,p),\ Cap(e)\}}{C(e)} + (1 - \lambda).PR(e) \quad (3.3)$$

Recall that λ is the tradeoff parameter. The first part of this equation is taken from the greedy set cover algorithm with a slight modification that takes the capacity of the expert into account. The second part of it evaluates the productivity of the expert. After finding

Algorithm 2 Cluster Hire with Project Pick Strategy

Input: set of n experts $E = \{e_1, e_2, \ldots, e_n\}$, set of m skills $S = \{s_1, s_2, \ldots, s_m\}$, set of k projects $P = \{p_1, p_2, \ldots, p_k\}$, tradeoff parameter λ, Productivity, Capacity of each expert, each Project's Skillset, each Expert's skillset and budget B.
Output: subset of projects $\mathscr{P} \subseteq P$ and a group of experts $\mathscr{E} \subseteq E$ that maximize $PP(\mathscr{P}, \mathscr{E})$ under the given budget B.

$\mathscr{E} \leftarrow \emptyset$, $\mathscr{P} \leftarrow \emptyset$, $b \leftarrow 0$
while $b < B$ and $P/\mathscr{P} \neq \emptyset$ **do**
  $P' \leftarrow P/\mathscr{P}$, $\mathscr{R} \leftarrow \{e \mid e \in E$ and $e \notin \mathscr{E}$ and $C(e) + b \leq B\}$
  if $\mathscr{R} = \emptyset$ **then**
    return $\mathscr{E}$ and $\mathscr{P}$
  for all $p \in P'$ **do**
    $E_p \leftarrow \emptyset$, $S_p \leftarrow PS(p)$, $\mathscr{R}' \leftarrow \mathscr{R}$
    while $S_p \neq \emptyset$ **do**
      for all $e \in \mathscr{R}'$ **do**
        if e covers at least one skill in p **then**
          $sc_e \leftarrow \lambda . \frac{\min\{Skill(e,p),\ Cap(e)\}}{C(e)} + (1-\lambda).PR(e)$
        else
          $sc_e \leftarrow 0$
      $e \leftarrow \arg\max_{e \in \mathscr{R}'} sc_e$
      add e to E_p, **update** S_p
  for all $p \in P'$ **do**
    if $(\sum_{e \in E_p} C(e)) + b > B$ **then**
      remove p from P'
  if $P' = \emptyset$ **then**
    return $\mathscr{E}$ and $\mathscr{P}$
  $(p, E_p) \leftarrow \arg\max_{p \in P'} \lambda . \frac{PF(p)}{\sum_{e \in E_p} C(e)} + (1-\lambda) . \sum_{e \in E_p} PR(e)$
  add p to $\mathscr{P}$, **assign** skills of experts in E_p to p
  for all $e \in E_p$ **do**
    add e to $\mathscr{E}$, **update** $Cap(e)$, $b \leftarrow b + C(e)$
    while $Cap(e) > 0$ **do**
      $s \leftarrow$ rarest skill in e which is required by a p in $P/\mathscr{P}$
      assign skill s to the most expensive p in $P/\mathscr{P}$
      update $Cap(e)$
return $\mathscr{E}$ and $\mathscr{P}$

the set of experts E_p for all uncovered projects, we select one of the projects as the winner and add it to the pool of already selected projects. In order to do that, we select a project that maximizes the following equation.

$$\lambda . \frac{PF(p)}{\sum_{e \in E_p} C(e)} + (1-\lambda) . \sum_{e \in E_p} PR(e) 3.5 \tag{3.4}$$

The intuition is the same, we are interested to choose the project that has a high profit, needs experts with low salary, and the set of experts responsible to perform the project's tasks have high productivity. Now, we are ready to present Algorithm 2 that returns a group of experts for performing a set of profitable projects with project pick strategy. The input and output of this algorithm are similar to Algorithm 1. In the first line, we initialize three variables $\mathscr{E}$, $\mathscr{P}$, and b, which are responsible to store the final group of experts, the selected projects, and amount of budget spent so far, respectively. The while loop of line 2 iterates until we run out of budget or no project is left to be covered. In line 3, we first assign all uncovered projects to set P'. We then put all of the experts in which adding them to $\mathscr{E}$ will not violate the budget to set $\mathscr{R}$. If $\mathscr{R}$ is empty, we terminate the algorithm in line 5 as we cannot proceed further and cover any more projects. The for loop of line 6 starts the process of assigning a score to each project p in P'. As we discussed before, the score function is a modification of the weighted greedy set cover algorithm that also takes into account the capacity of experts and communication cost. Sets E_p, S_p, and $\mathscr{R}'$ are initialized in line 7. E_p stores the set of experts to perform p. S_p is a duplicate of the set of required skills in p, in which we try to cover them by adding experts to E_p. $\mathscr{R}'$ is a duplicate of $\mathscr{R}$. The while loop of line 8 is executed until no more skill is required by p (i.e., S_p becomes $\emptyset$). Line 9 iterates over all experts in $\mathscr{R}'$ and each iteration chooses the one that maximizes the modified weighted greedy set cover score. This expert is selected in line 14 and is added to E_p in line 15. We also update S_p in line 15. After finding the set of experts for all projects, in lines 16 to 18, we remove the projects in which adding their associated expert set to $\mathscr{E}$

violate the budget constraint. If set P' becomes empty after this operation, we terminate the algorithm in line 20. If P' is not empty, we select the best project p in P' in line 21 according to equation 3.5. In line 22, we add the best project to $\mathscr{P}$, and cover the skills of p. The for loop of line 23 iterates through all experts in E_p. These are the set of experts that are responsible to cover the best-selected project in line 21. In line 24, each expert in E_p is added to $\mathscr{E}$ and its capacity is updated. If the expert has some unassigned capacity, we assign her rarest skill to the most profitable project in lines 25 to 28 until her capacity is full. The motivation for doing it the same as the last part of Algorithm 1, as soon as we hire an expert, we prefer to use her maximum capacity. As we are considering a lot of factors the worst case running time of our algorithm is $O(nmk)$. In other words, the worst case run time of our algorithm is cubic.

3.2.1 Illustration of Project Pick Strategy using normalized values

In this section, we will see a working example of our Project Greedy approach. The input data is the same as the one we used in the Expert Greedy Approach. Table 3.4 and Table 3.3 are the input data for this example. We start with an initial budget of 150.

In each iteration, we will calculate the expert score for each project. We try to form a team for each project. Once we have the team for each project, we calculate the score for the project-expert pair using the equation 3.5. Table 3.14 shows the expert score calculation for Project 1.Table 3.15 shows the skills covered by the highest expert. It also shows the remaining skills that need to be covered to complete the project.

	P1
E1	((0.5) * (0.6 * min{1,2}) / 0.75 + (0.5 * 1)) = 0.9
E2	((0.5) * (0.6 * min{2,1}) / 0.5 + (0.5 * 0.66)) = 1.53
E3	0
E4	**((0.5) * (0.6 * min{2,6}) / 0.25 + (0.5 * 0.66)) = 2.73**
E5	((0.5) * (0.6 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 1.365

Table 3.14: Expert Score Calculation for Project 1 (Iteration 1)

Project 1		
S1	S3	S6
E4	E4	

Table 3.15: Skills covered after 1st Iteration

Table 3.16 shows the score calculation in internal Iteration 2. This iteration tries to look for the expert to complete the remaining skills. Table 3.17 shows the skills covered in internal Iteration 2.

	P1
E1	0
E2	**((0.5) * (0.6 * min{1,1}) / 0.5 + (0.5 * 0.66)) = 0.93**
E3	0
E5	0

Table 3.16: Expert Score Calculation for Project 1 (Iteration 2)

P1		
S1	S3	S6
E4	E4	E2

Table 3.17: Skills covered after 2nd Iteration

Team for Project 1: (E4, E2) Similarly, Table 3.18 shows score calculation for Project 2 and Table 3.19 shows the skills covered in internal Iteration 1.

	P2
E1	0
E2	((0.5) * (1 * min{1,1}) / 0.5 + (0.5 * 0.66)) = 1.33
E3	0
E4	**((0.5) * (1 * min{2,6}) / 0.25 + (0.5 * 0.66)) = 4.33**
E5	((0.5) * (1 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 2.165

Table 3.18: Expert Score Calculation for Project 2 (Iteration 1)

P2	
S2	S3
E4	E4

Table 3.19: Skills covered after 1st Iteration

Team for Project 2: (E4)

Table 3.20 and Table 3.22 shows the score calculation for internal Iteration 1 and internal Iteration for Project 3 and Table 3.21 and Table 3.23 shows the skills covered in internal Iteration 1 and internal Iteration 2.

	P3
E1	**((0.5) * (0.2 * min{2,2}) / 0.75 + (0.5 * 1)) = 0.766**
E2	0
E3	((0.5) * (0.2 * min{2,4}) / 1 + (0.5 * 1)) = 0.7
E4	((0.5) * (0.2 * min{1,6}) / 0.25 + (0.5 * 0.66)) = 0.73
E5	((0.5) * (0.2 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.20: Expert Score Calculation for Project 3 (Iteration 1)

P3		
S1	S4	S5
E1	E1	

Table 3.21: Skills covered after 1st Iteration

	P3
E2	0
E3	((0.5) * (0.2 * min{1, 4}) / 1 + (0.5 * 1)) = 0.6
E4	0
E5	((0.5) * (0.2 * min{1,3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.22: Expert Score Calculation for Project 3 (Iteration 2)

P3		
S1	S4	S5
E1	E1	E5

Table 3.23: Skills covered after 2nd Iteration

Team for Project 3: (E1, E5)

Once we calculate Expert Score for each Project we use equation 3.5 to calculate Project Score and after calculating project score we select the project with the highest score and finalize the team and assign it to that project. Once we hire these experts we allocate them to remaining projects based on their capacity. After assigning experts to remaining projects we continue the same process for remaining projects until we find a team for all remaining projects.

$$\lambda . \frac{PF(p)}{\sum_{e \in E_p} C(e)} + (1-\lambda) . \sum_{e \in E_p} PR(e) \tag{3.5}$$

Project	Score
P1	(((0.5 * 0.6) / (0.5 + 0.25)) + (0.5 * (0.66 + 0.66))) = 1.06
P2	**(((0.5 * 1) / 0.25) + (0.5 * (0.66))) = 2.33**
P3	(((0.5 * 0.2) / (0.75 + 0.25)) + (0.5 * (1 + 0.33))) = 0.765

Table 3.24: Project Score after External Iteration 1

Table 3.24 shows the score calculation of projects. We select the Project-Expert pair that has the highest score and adds it to the final team. Table 3.25 shows the skills that are covered in Iteration 1. It also shows the remaining skills that need to be covered to complete all projects.

After External Iteration 1							
P1			**P2**		**P3**		
S1	S3	S6	S2	S3	S1	S4	S5
E4	E4		E4	E4	E4		

Table 3.25: Skills Covered in Iteration 1

Remaining Budget after External Iteration 1 = 150 - 30 = 120

After one external iteration, we will be able to complete one Project. We will now start another external iteration and try to select another project which has the highest score among the remaining projects. Table 3.26 shows score calculation for Project 1 and Table 3.27 shows the skills covered by the highest expert in internal iteration 1.

	P1
E1	0
E2	**((0.5) * (0.6 * min{1, 1}) / 0.5 + (0.5 * 0.66)) = 0.93**
E3	0
E5	0

Table 3.26: Expert Score Calculation for Project 1 (Iteration 1)

P1
S6
E2

Table 3.27: Skills covered after 1st Iteration

Team for Project 1: (E2)

Table 3.28 shows the score calculation for Project 3 and Table 3.29 shows the skills covered in internal iteration.

	P3
E1	((0.5) * (0.2* min{1, 2}) / 0.75 + (0.5 * 1)) = 0.63
E2	0
E3	**((0.5) * (0.2 * min{2, 4}) / 1 + (0.5 * 0.1)) = 0.7**
E5	((0.5) * (0.2 * min{1, 3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.28: Expert Score Calculation for Project 3 (Iteration 1)

P3	
S4	S5
E3	E3

Table 3.29: Skills Covered after 1st Iteration

Table 3.30 shows the score calculation for expert-project pair discovered in internal iterations of external iteration 2. Table 3.31 shows the skills covered in external iteration 2 and it also shows the remaining skills required to complete other projects.

Project	**Score**
P1	**(((0.5 * 0.6) / (0.5)) + (0.5 * (0.66))) = 0.93**
P3	(((0.5 * 0.2) / 1) + (0.5 * (1))) = 0.6

Table 3.30: Project Score after External Iteration 2

After External Iteration 2							
P1			**P2**		**P3**		
S1	S3	S6	S2	S3	S1	S4	S5
E4	E4	E2	E4	E4	E4		

Table 3.31: Skills Covered in Iteration 2

Remaining Budget after External Iteration 2 = 120 - 40 = 80

Team for Project 3: (E3)

Table 3.32 shows the score calculation of Project 3 and selects the expert with highest score. Table 3.33 shows the skills covered in this iteration.

	P3
E1	((0.5) * (0.2* min{1, 2}) / 0.75 + (0.5 * 1)) = 0.63
E3	**((0.5) * (0.2 * min{2, 4}) / 1 + (0.5 * 0.1)) = 0.7**
E5	((0.5) * (0.2 * min{1, 3}) / 0.25 + (0.5 * 0.33)) = 0.565

Table 3.32: Iteration 1 Score Calculation

P3	
S4	S5
E3	E3

Table 3.33: Skills Covered after 1st Iteration

Table 3.34 shows the skills covered in external iteration 3 and we have now completed all the projects. It shows which expert covers which skill of which project. We are now ready with the final team so we stop our iterations.

After External Iteration 3							
P1			P2		P3		
S1	S3	S6	S2	S3	S1	S4	S5
E4	E4	E2	E4	E4	E4	E3	E3

Table 3.34: Skills Covered after External Iteration 3

Remaining Budget after External Iteration 3 = 80 - 60 = 20

Final Team: (E2, E3, E4)

Chapter 4

Experiments

This chapter elaborates the performance of our proposed algorithms over the synthetic dataset and a real DBLP dataset.

4.1 Synthetic Data

We generate synthetic data sets for our experiment. Our program (i.e. the coding) has been implemented in order to change the required numerical values to have the different type of dataset. For the expert details, we set the value for the number of experts; each expert randomly gets a specific number of skills and this skillset will be assigned from a set of all skills (65 skills). The capacity and productivity take values between a min and max value randomly. The min and max value is decided by the user. For the project details, we set the values for the number of projects with the profit of completing the project. Each project randomly gets a specific number of required skills from the set of all skills (65 skills).

We randomly generated the dataset with the following values: the number of expert's skills randomly from 5 to 8, the productivity value is randomly from 1 to 10, the capacity of the experts is randomly from 3 to 6, and the salary of an expert is randomly from 500 to 550. We set the number of projects from 5 to 60. The profit of the project is generated randomly between 500 and 600. We run the experiments 10 times and record the average values. The default value of λ is assigned to 0.5 since we need to give priority to both the profit and productivity equally. Our experiments use the various range of values for the budget to see the total profit returned by each algorithm.

4.1.1 Performance Analysis

For the baseline comparison, we use the random algorithm. It selects a group of experts that can cover all required skills to complete the given projects without considering capacity, productivity, and profit. It only considers the Budget constraint and makes sure that the overall cost of hiring experts is less than or equal to given Budget B. We also compare the proposed algorithm with the exact algorithm for obtaining the results using an exhaustive search. We used Intel Core i7 2.6 GHz computer with 8 GB of RAM to implement our algorithms in Java.

We check the effect of the budget on the total profit of the projects as shown in figure 4.1, figure 4.2, 4.3 and figure 4.4. Each experiment is evaluated with a k number of projects, in which k = $\{5, 15, 25, 40\}$. The graphs are plotted for total profit against various budget from 2000 to 30,000. The results indicate that Project Greedy achieves a higher overall profit than Expert Greedy when the budget is low. However, when the budget is high, both Project greedy and Expert greedy perform similarly. Both the project greedy and expert greedy outperformed the random algorithm.

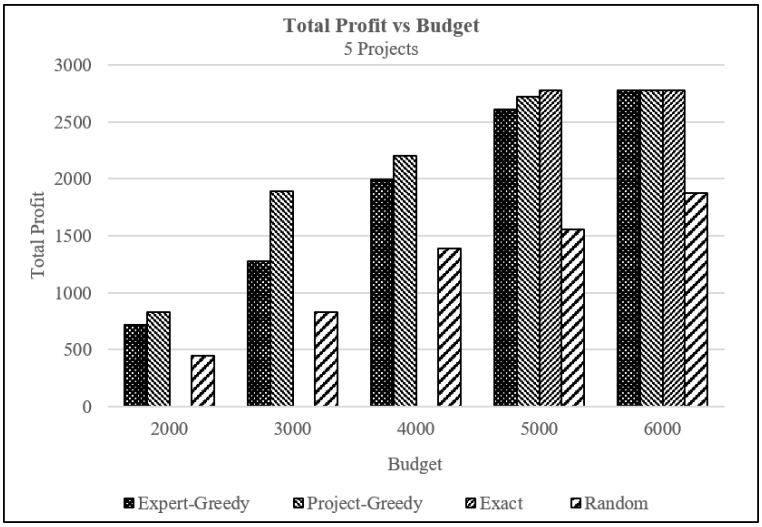

Figure 4.1: Comparison for Total profit vs. a budget of Project Greedy, Expert Greedy with Exact and Random Algorithm

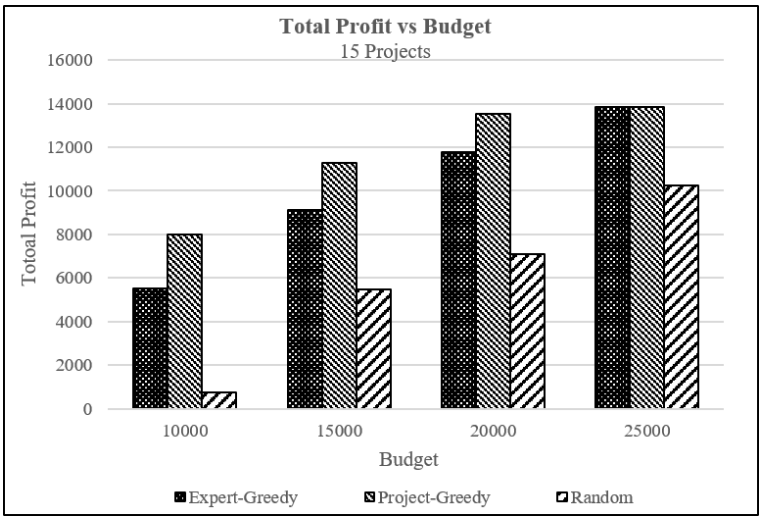

Figure 4.2: Comparison for Total profit vs. a budget of Project Greedy, Expert Greedy with Random Algorithm

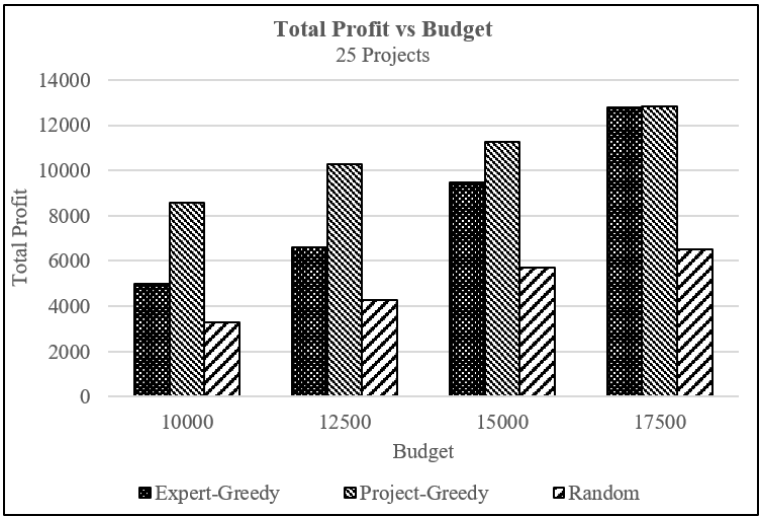

Figure 4.3: Comparison for Total profit vs. a budget of Project Greedy, Expert Greedy Random Algorithm

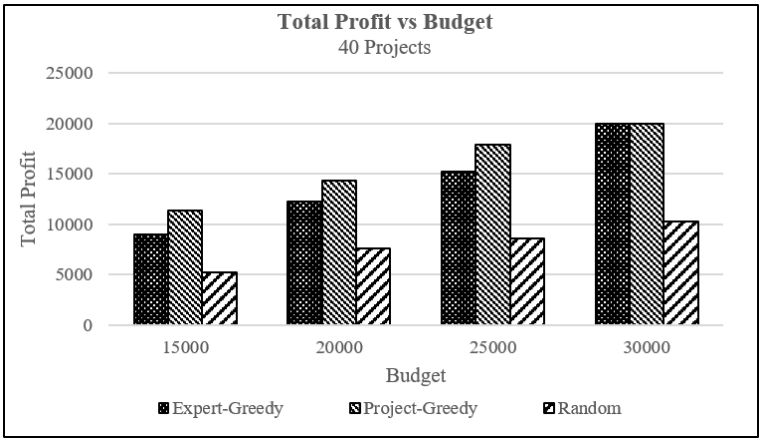

Figure 4.4: Comparison for Total profit vs. a budget of Project Greedy, Expert Greedy with Random Algorithm

The exhaustive search takes a very long time to produce results because the problem is NP-hard and the search space is exponential. We are able to get results for 5 projects with a specific budget B in a considerable time. When we have less budget, the exact algorithm needs to check the entire subsets of 5 projects with 1000 experts(let's say). The number of possible teams of experts for all these subsets will be very high. Therefore, the exact

algorithm has been executed without considering the subsets of the projects. It executed for 5 projects with the various budget in figure4.1.

Then, we tested the number of completed project vs budget as shown in figure 4.5, figure 4.6, figure 4.7 and 4.8 with default $\lambda = 0.5$. The result shows that both the project greedy and expert greedy behaves similarly as in the result from total profit vs. budget. The Project greedy completes more projects than Expert greedy when the budget is limited. For higher values of the budget, both algorithms complete the same number of projects or almost all projects are completed. At the same time, both Project greedy and Expert greedy outperform the Random algorithm. The exact algorithm couldn't perform with the subset of the project as we explained above.

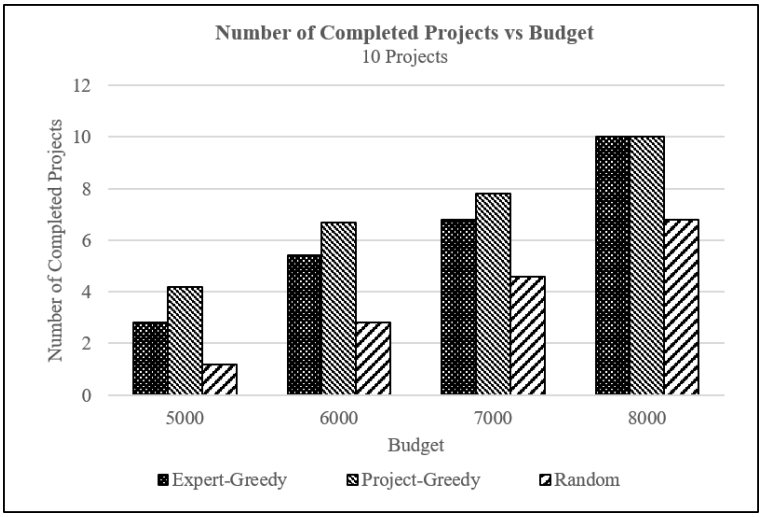

Figure 4.5: Comparison for the completed project vs. budget of Project Greedy, Expert Greedy with Random Algorithm

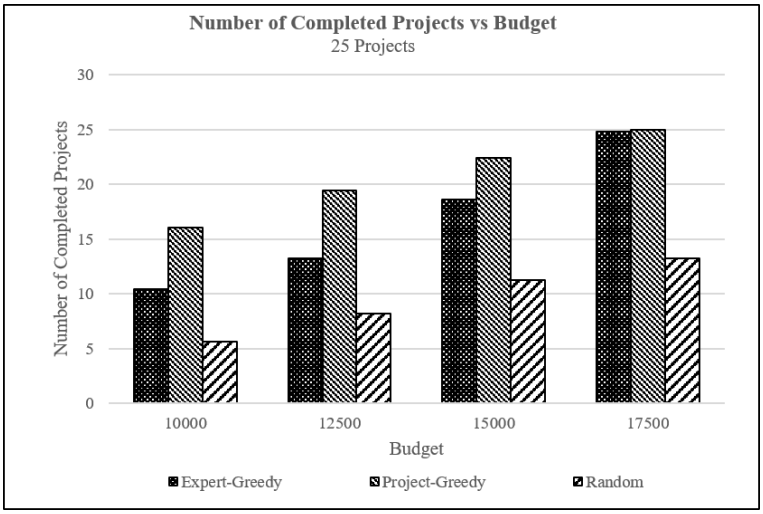

Figure 4.6: Comparison for the completed project vs. budget of Project Greedy, Expert Greedy with Random Algorithm

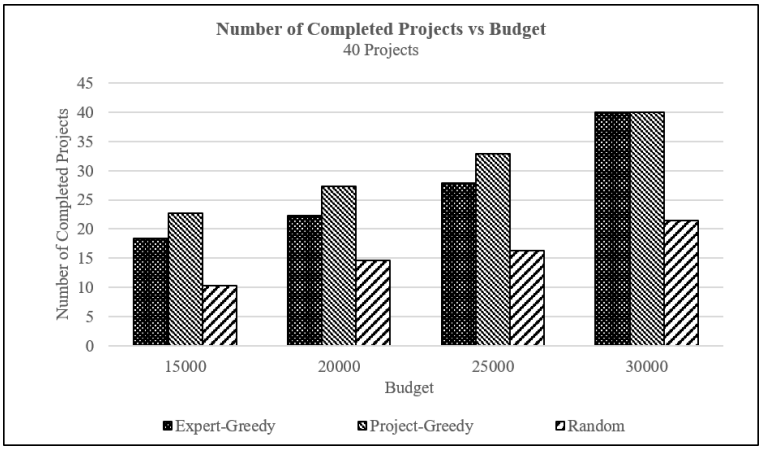

Figure 4.7: Comparison for the completed project vs. budget of Project Greedy, Expert Greedy with Random Algorithm

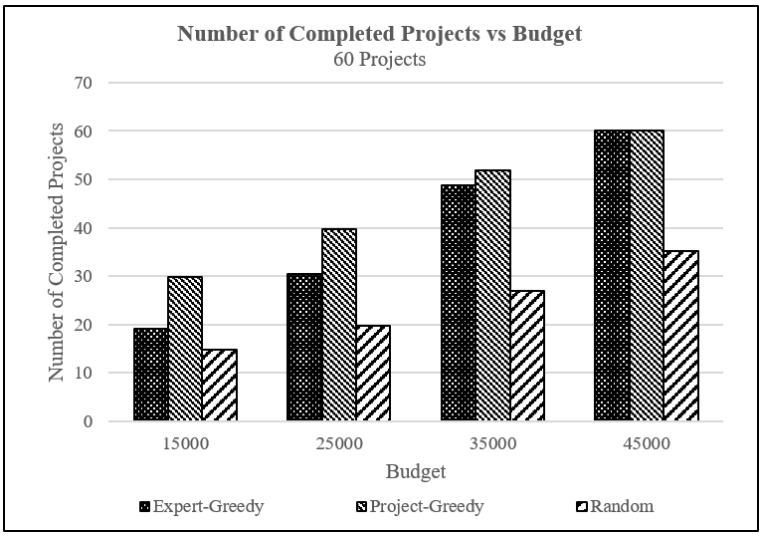

Figure 4.8: Comparison for the completed project vs. budget of Project Greedy, Expert Greedy with Random Algorithm

Moreover, we checked the run time of the proposed two algorithms and random algorithms by varying the number of projects as shown in figure 4.9. Random algorithm took less time than the other two since it selects experts based on their skills. It did not minimize or maximize any objective. The project greedy took little more time than expert greedy algorithms as in the project greedy algorithm we have two iterations one internal and one external. External iteration selects the best project and the best expert associated with it. The internal iteration then finds remaining experts to complete the selected project in the external iteration. On the other hand, in expert greedy, we do not iterate internally.

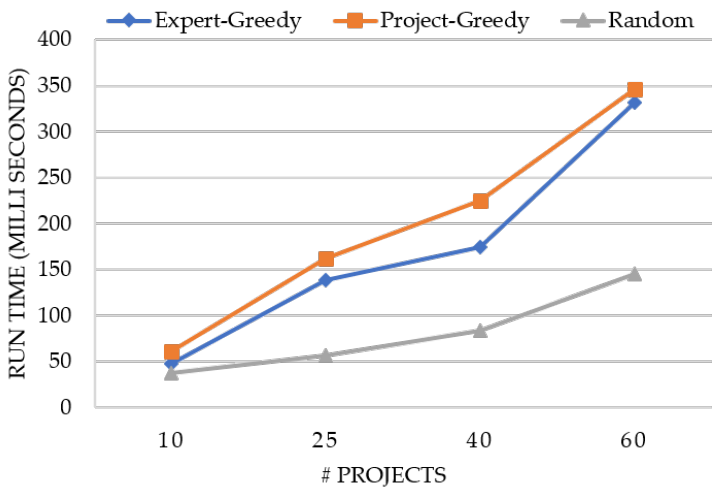

Figure 4.9: The run time of Project Greedy, Expert Greedy, and Random Algorithm when we have various number of projects.

Since our objective is to maximize profit and productivity we now check our results to compare productivity with the number of projects. Figure 4.10 shows the average productivity per expert. We need to compare it with productivity per expert because total productivity depends on the number of experts in the team. If the team is bigger than the total productivity has a greater value. If the team is smaller than the total productivity is less. In order to compare productivity, we need to consider the average productivity. In other words, we need to consider productivity per expert. The results show that the average productivity of the team generated by Expert Greedy and Project Greedy algorithm is between 7 and 9. Whereas the average productivity of random algorithm has not shown any fixed pattern and it is lower than the productivity generated by our algorithms. This justifies that we meet our objective of maximizing Productivity.

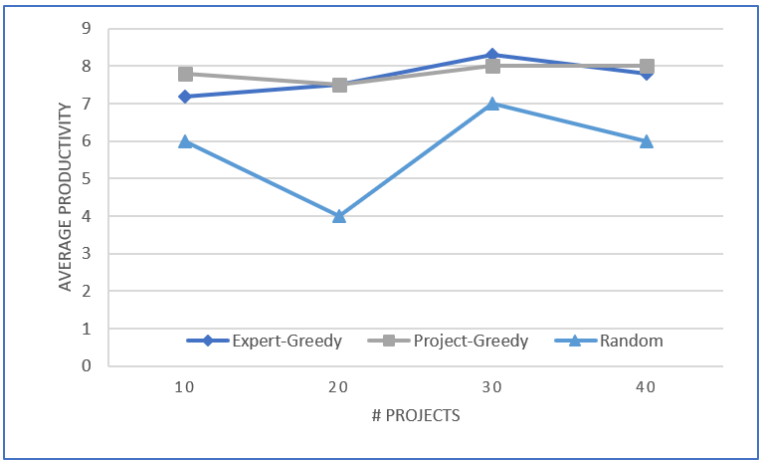

Figure 4.10: Average Productivity of Project Greedy, Expert Greedy, and Random Algorithm when we have various number of projects.

4.2 DBLP Dataset

DBLP dataset contains information of 200k authors. Each expert in the dataset is associated with a set of skills. The cost of an expert is determined by assigning a random cost value to each expert. The productivity of an expert is already present in the dataset and it is determined by the number of papers published by them. Capacity, Productivity and Project details are generated in the same way as in synthetic data. The only difference is in the skill set which is used to generate random data. Here we use the skill set formed by scanning through all the experts in the dataset.

4.2.1 Performance Evaluation

In this section, we will check the performance of our proposed algorithms. Since there exists no work that is exactly similar to our work we would compare our work with the work done by [1] as we are extending their work. They proposed three algorithms Expert Greedy, Project Greedy and Clique Greedy algorithms. Amongst the three of them, Project

Greedy has performed the best. Therefore we are comparing our work with their project greedy algorithm.

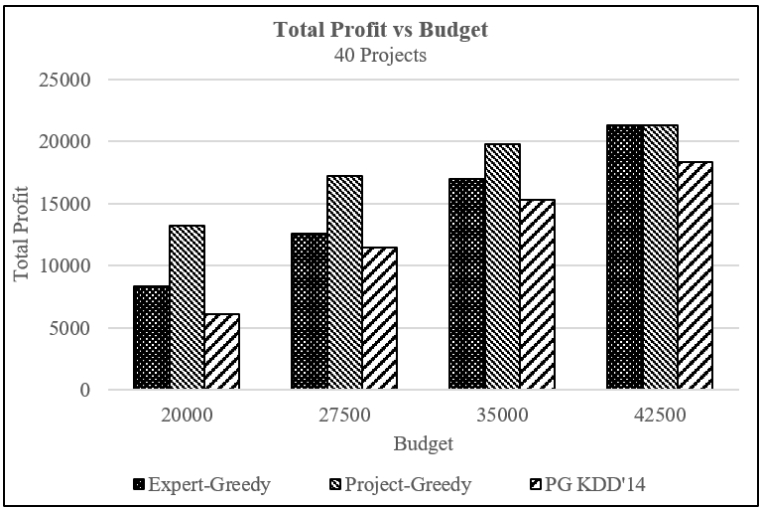

Figure 4.11: Comparison of Total Profit vs Budget of Project Greedy, Expert Greedy, and PG KDD'14

We compared our Expert Greedy and Project Greedy with [1] for k number of projects where the value of k is 40 and 60. We have compared total profit with the budget by varying the budget. As seen in the figure 4.11 and 4.12 our Project Greedy approach has outperformed all other algorithms. When the budget is low total profit is less as profit is directly dependent on the number of completed projects. Expert Greedy performs a bit lower than Project Greedy because Project Greedy concentrates on completing one project at a time while Expert Greedy hires one expert in each iteration and cover as many projects as possible. So at the end of one iteration Project Greedy guarantees one complete project while expert greedy does not guarantee it. Hence, Project Greedy performs better when the budget is low. When the budget is high both Project Greedy and Expert Greedy performs same as they are both able to complete all the projects.

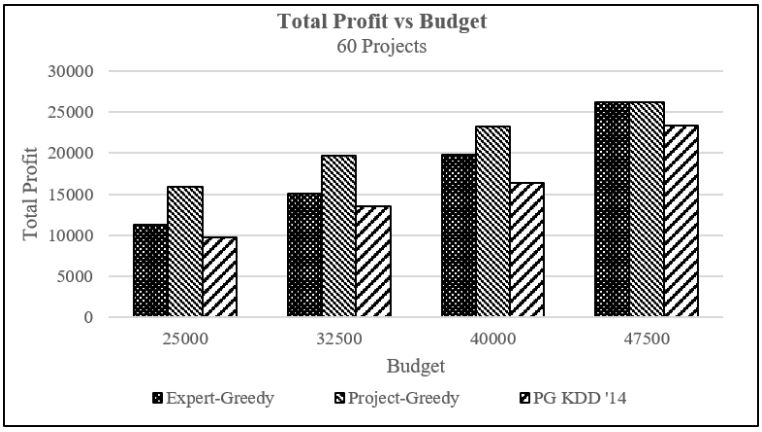

Figure 4.12: Comparison of Total Profit vs Budget of Project Greedy, Expert Greedy, and PG KDD'14

We further compare Project Greedy and Expert Greedy with PG KDD'14 in terms of a number of projects completed with the budget. Figure 4.13 and figure 4.14 shows that when the budget is low the number of projects completed is less and as the budget increases the number of completed projects increases. Amongst the three algorithms Project Greedy performs the best whereas Expert Greedy is comparatively below Project Greedy. Our both algorithms perform better than the PG KDD'14.

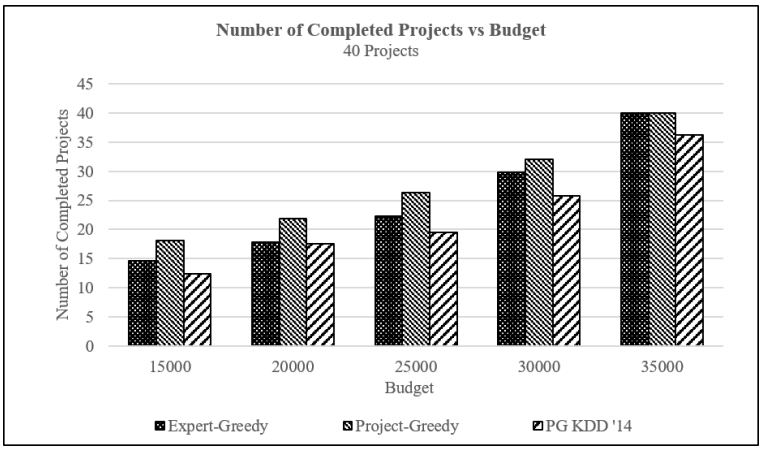

Figure 4.13: Comparison of Number of Completed Projects vs Budget of Project Greedy, Expert Greedy, and PG KDD'14

We have performed the experiments for 40 and 60 Projects respectively. Figure 4.13 shows the results for 40 Projects while figure 4.14 shows the results for 60 Projects. It is seen that when the budget is high the number of completed projects for both Expert Greedy and Project Greedy are almost similar.

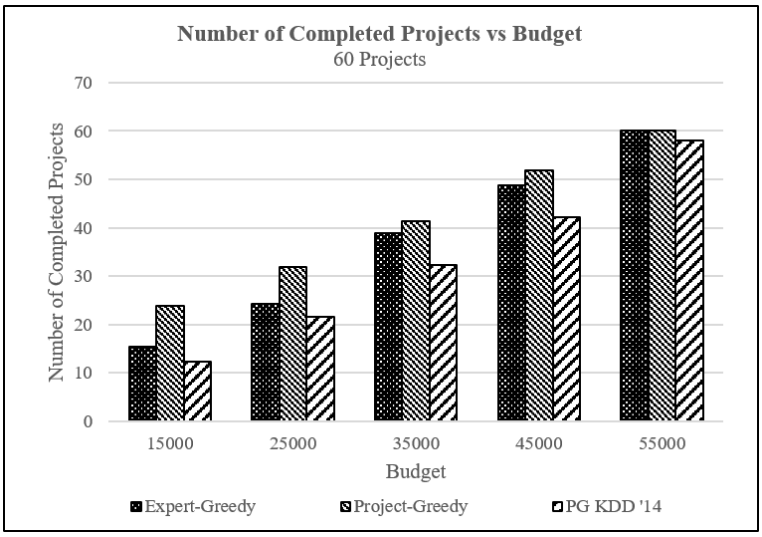

Figure 4.14: Comparison of Number of Completed Projects vs Budget of Project Greedy, Expert Greedy, and PG KDD'14

We now compare the run time of our algorithms. Figure 4.15 shows the runtime of Expert Greedy, Project Greedy and PG KDD '14. Figure 4.15 shows that PG KDD'14 has a lower run time than our Project Greedy and Expert Greedy algorithm. The reason for high runtime is the number of iterations in our algorithms. In our algorithms, we have a higher number of iterations due to productivity. We maximize profit and productivity and minimize budget. Project Greedy has the highest runtime as it calculates the score for each expert and then it runs another internal iteration to calculate project score. Hence, the number of iterations in Project Greedy algorithm is highest. Due to a high number of iteration Project Greedy has the highest runtime.

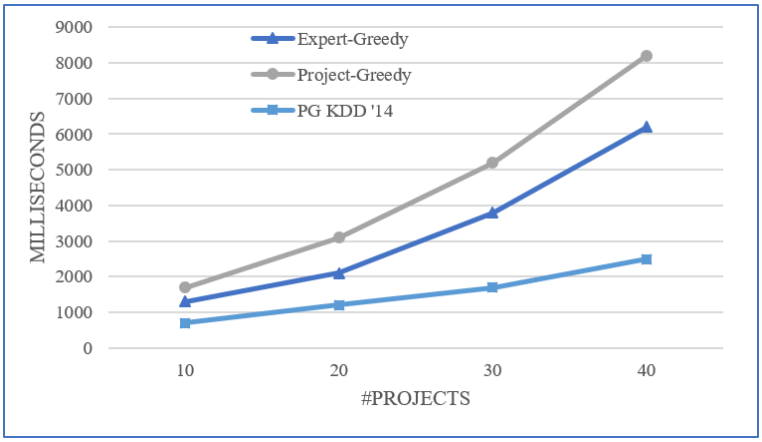

Figure 4.15: The run time of Project Greedy, Expert Greedy, and PG KDD '14 when we have various number of projects.

4.3 Discussion

In this section of the thesis, we will discuss the performance of our proposed algorithms to solve the Cluster Hire problem by maximizing profit and Productivity and by minimizing the Budget used to hire experts. Cluster Hire problem is an NP-hard problem and in order to solve this problem, we have presented two greedy algorithms: Expert Greedy and Project Greedy algorithm. We have performed various experiments and compared the results with the exact algorithm, random algorithm and existing method to evaluate the performance of our algorithms. To check the performance of our algorithms we have used two type of dataset: Synthetic dataset and DBLP dataset. Let's discuss them one by one.

Using synthetic data we have performed experiments that compare total profit with the budget, number of completed projects with the budget, runtime with the number of projects, and average productivity with a number of projects. When considering the experiment Total Profit vs Budget we have compared Expert Greedy and Project Greedy with the Exact algorithm and Random algorithm. We have performed this experiment by varying number of projects from 5 to 40. As shown in figure 4.1 total profit generated using a random

algorithm is slightly more than our algorithms. It can also be seen that our approach produces more profit than the random algorithm. In addition to this, it is seen that the Project Greedy algorithm produces more profit than the Expert Greedy algorithm and as the budget increases it slowly matches with the project greedy algorithm. The reason behind this difference is that when the budget is low Project Greedy algorithm completes more projects than expert greedy because it focuses on completing one project in each iteration while expert greedy algorithm focuses on hiring one expert in each iteration. Hence, project greedy algorithm produces more profit. The same trend is observed when the same experiment is performed by varying number of projects. However, we have considered the exact algorithm for 5 projects only as we were not able to find the solution for more than 5 projects inconsiderable amount of time due to a large number of possible combinations.

When considering an experiment which compares the number of completed projects with the budget we have compared Expert Greedy and Project Greedy with the Random algorithm. Through the experiments, it is observed that both expert greedy and project greedy were able to complete more projects than the random algorithm. Moreover, when the budget was low project greedy completed more projects than expert greedy algorithm. As the budget increases the number of completed projects for both algorithm increases and when there is sufficient budget both project greedy and expert greedy are able to complete all the projects. We have performed this experiment for 10, 25, 40 and 60 Projects.

The next experiment we performed was to see the runtime of these algorithms against the number of projects. In this experiment, the project greedy algorithm has the highest runtime amongst the three algorithms considered. Expert greedy is slightly low than the project greedy in terms of runtime and the random algorithm has the least runtime amongst the three of them. We have performed this experiment for 10, 25, 40 and 60 projects and runtime has varied between the range of approximately 30 ms to 350 ms.

We further considered average productivity against the number of projects and compared

project greedy and expert greedy against random algorithm and we have observed that the average productivity for project greedy and expert greedy algorithm lies between 7 and 9 whereas the average productivity for random algorithm does not follow a fixed pattern. It is seen to be up and down with a value approximately between 4 and 7.

After discussing the experiments with the synthetic dataset, we will now discuss the experiments performed using DBLP dataset. Using DBLP dataset we have performed experiments that compare total profit against the budget, number of completed projects against budget and runtime against the number of projects. All the values in the experiment are calculated by taking an average value of 10 iterations. While considering total profit against budget we have compared the results of expert greedy and project greedy with the existing method. It is found that the total profit generated by project greedy and expert greedy is higher than the previous method and amongst project greedy and expert greedy algorithm, project greedy algorithm generates higher profit. However, with an increase in the budget the profit gap between project greedy and expert greedy decreases and when there is sufficient budget the total profit is almost similar for both the algorithms. We have performed this experiment for 40 and 60 projects and a similar trend is observed for both the experiments. When the budget is low expert greedy algorithm produces approximately 37 percent better results than the previous approach and when the budget is high it produces approximately 16 percentage better result than the previous approach. Whereas project greedy algorithm generates approximately 90 percent better results when the budget is low and approximately 16 percent better results when the budget is high than the previous approach.

Another experiment that we have considered is the number of completed projects against the budget. After looking at the results we can say that project greedy algorithm completes the most number of projects when the budget is low. Expert greedy is slightly below than project greedy and previous approach completes less number of projects as compared to both project greedy and expert greedy algorithm. When the budget is high both expert

greedy and project greedy generates almost similar profit. When the budget is low expert greedy algorithm produces approximately 25 percent better results than the previous approach and when the budget is high it produces approximately 34 percentage better result than the previous approach. Whereas project greedy algorithm generates approximately 94 percent better results when the budget is low and approximately 34 percent better results when the budget is high than the previous approach.

We further considered the run time of the three algorithms we have and it is found that the previous approach has the least runtime when compared to expert greedy and project greedy and amongst project greedy and expert greedy expert greedy has less runtime as compared to project greedy algorithm. Through experiments, it is found that for less number of projects (10) runtime for expert greedy is 85 percent higher than that of the previous approach and when we consider a higher number of projects (40) run time is approximately 1.48 times higher than the previous approach. For project greedy approach the run time is approximately 1.42 times previous approach when the number of projects is low (10) and it is approximately 2.28 times when the number of projects is high (40).

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Few researchers in the past have addressed the Cluster Hire problem which is to hire a group of experts to complete multiple projects by selecting a group of experts within a given budget. This study examines the productivity of the experts for the projects which have maximum profit as an extension of previous work. In this thesis, we extend the work of [1] by introducing a significant factor Productivity. It is a significant concept since it considers the most efficient members to complete the project within budget. The study optimizes both profit and productivity. Therefore, it is a bi-objective problem and gives equal priority to both objectives by assigning 0.5 as a tradeoff value. To handle this NP-hard problem, we proposed two greedy algorithms in order to hire the best group of experts. We propose an Expert Greedy algorithm and a Project Greedy algorithm.

Expert Greedy algorithm focuses on hiring one expert in each iteration. This selection is done by using a score function that maximizes productivity and profit and minimizes the budget. Once an expert is hired the algorithm utilizes the expert by assigning him/her to other projects depending on the expert's capacity. The loop iterates until all the projects are completed or the budget is exhausted. It returns a team that can cover all the projects within the given budget.

Project Greedy algorithm, on the other hand, focuses on completing one project in each iteration. In this approach, we select one expert-project pair. The selection of this pair is done using the score function. The pair with maximum score is selected. Once we select an expert-project pair, we try to complete this project by hiring experts who possess the

required skills. We also assign all these experts to the remaining projects based on their capacity and the skills they possess.

Both algorithms operate in their unique ways and return a team that can complete all the projects within the given budget while maximizing productivity and profit. Experiments suggest that when the budget is low project greedy algorithm generates more profit as compared to expert greedy algorithm. It also suggests that project greedy algorithm completes more projects than the expert greedy algorithm when the budget is low. However, when the budget is high both project greedy and expert greedy algorithm generates similar results. Experiments prove that both project greedy and expert greedy algorithm performs better than previous approach and random algorithm in terms of total profit generated, average productivity and number of projects completed. However. previous approach and random algorithm have better run time than both our algorithms because the number of iterations in our approach is higher than that of the previous approach as it considers both productivity and profit while selecting an expert.

5.2 Future Work

In this section, we will discuss the possible future work for this thesis.

- Our algorithms have successfully maximized profit and productivity by finding a team of experts who can complete all projects undertaken. As an imminent future work, we can consider finding a backup expert for each expert in the team. There are situations in the real world where an expert leaves a project in the middle of the term. To tackle such situation we can consider finding a backup team that can replace each expert on the team.

- We can consider extending this problem by including communication cost in our problem and convert this problem into a multi-objective optimization prob-

lem. Communication cost is the communication overhead that is spent in order to get all experts together. We can minimize this factor while maximizing profit and productivity.

- In our approach, we assume that an expert can provide his/her expertise to multiple projects simultaneously. In other words, an expert works on more than one project at the same time. We do not have a count of the number of hours an expert spends on a project. As a potential future work, we can have a constraint that decides the number of hours an expert needs to spend on a particular project. This will allow predicting the timeline for various projects accurately. For example, an expert A is assigned to 3 projects. The expert does not know how much time needs to be spent on each project. If an expert finds one project interesting he/she might consider spending more time on that project which will make the expert biased to one project. As an improvement, if we introduce a parameter that specifies the number of hours that needs to be spent on one project based on its complexity it will make the expert unbiased and we will be able to complete all projects in expected time.

- Another aspect that can be considered in future work is to solve this problem by using an approximation algorithm instead of the greedy algorithm. The challenging part of using this approach is the complexity of the problem, if handled properly we might have chances of better results than the greedy algorithm.

- In this thesis, we proposed two algorithms expert greedy and project greedy algorithm. As future work we can consider designing a hybrid model that solves this problem by selecting which approach will best solve this problem under the given circumstances at that point in time.

BIBLIOGRAPHY

[1] B. Golshan, T. Lappas, and E. Terzi, "Profit-maximizing Cluster Hires," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: ACM, 2014, pp. 1196–1205. [Online]. Available: http://doi.acm.org/10.1145/2623330.2623690

[2] M. Zihayat, A. An, L. Golab, M. Kargar, and J. Szlichta, "Authority-based Team Discovery in Social Networks," in *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, V. Markl, S. Orlando, B. Mitschang, P. Andritsos, K.-U. Sattler, and S. B. s, Eds. OpenProceedings.org, 2017, pp. 498–501.

[3] "Tang." [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14791

[4] M. S. Patel and M. Kargar, "Cluster Hire in a Network of Experts," 2017.

[5] T. Lappas, K. Liu, and E. Terzi, "Finding a Team of Experts in Social Networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 467–476. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557074

[6] "Data mining," Dec. 2018, page Version ID: 871751288. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Data_mining&oldid=871751288

[7] P. Alpar, "What Data Is Necessary to Data Mine for Knowledge?" *AI Magazine*, vol. 17, pp. 37–54, 1996.

[8] "Data Mining | KDD process," Jun. 2018. [Online]. Available: https://www.geeksforgeeks.org/data-mining-kdd-process/

[9] "Team building," Nov. 2018, page Version ID: 870217171. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Team_building&oldid=870217171

[10] "Heuristic (computer science)," Nov. 2018, page Version ID: 871267953. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Heuristic_(computer_science)&oldid=871267953

[11] "Exact algorithm," Feb. 2018, page Version ID: 824685329. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Exact_algorithm&oldid=824685329

[12] "Greedy algorithm," Oct. 2018, page Version ID: 863293262. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Greedy_algorithm&oldid=863293262

[13] "Greedy_description," Oct. 2018, page Version ID: 863293262. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Greedy_algorithm&oldid=863293262

[14] "Greedy_example," Nov. 2018. [Online]. Available: /cpt-lessons/algorithms/greedy/

[15] "Multi-objective optimization," Nov. 2018, page Version ID: 869845354. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multi-objective_optimization&oldid=869845354

[16] M. Kargar, M. Zihayat, and A. An, "Finding Affordable and Collaborative Teams from a Network of Experts," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, ser. Proceedings. Society for Industrial and Applied Mathematics, May 2013, pp. 587–595. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972832.65

[17] M. Kargar and A. 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*, ser. CIKM '11.

New York, NY, USA: ACM, 2011, pp. 985–994. [Online]. Available: http://doi.acm.org/10.1145/2063576.2063718

[18] A. Gajewar and A. D. Sarma, "Multi-skill Collaborative Teams based on Densest Subgraphs," *arXiv:1102.3340 [physics]*, Feb. 2011, arXiv: 1102.3340. [Online]. Available: http://arxiv.org/abs/1102.3340

[19] M. Kargar, A. An, and M. Zihayat, "Efficient Bi-objective Team Formation in Social Networks," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, P. A. Flach, T. De Bie, and N. Cristianini, Eds. Springer Berlin Heidelberg, 2012, pp. 483–498.

[20] Y. Han, Y. Wan, L. Chen, G. Xu, and J. Wu, "Exploiting Geographical Location for Team Formation in Social Coding Sites," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, and Y.-S. Moon, Eds. Springer International Publishing, 2017, pp. 499–510.

[21] K. Selvarajah, P. Moradian Zadeh, Z. Kobti, M. Kargar, M. T. Ishraque, and K. Pfaff, "Team Formation in Community-Based Palliative Care," 2018, pp. 1–7.

[22] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Power in unity: forming teams in large-scale community systems," in *CIKM*, 2010.

[23] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang, "Online Search of Overlapping Communities," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: ACM, 2013, pp. 277–288. [Online]. Available: http://doi.acm.org/10.1145/2463676.2463722

[24] X. Yin, C. Qu, Q. Wang, F. Wu, B. Liu, F. Chen, X. Chen, and D. Fang, "Social Connection Aware Team Formation for Participatory Tasks," *IEEE Access*, vol. 6, pp.

20 309–20 319, 2018.

[25] “Feature scaling,” Nov. 2018, page Version ID: 870599160. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Feature_scaling&oldid=870599160

VITA AUCTORIS

NAME:	Parth Patel
PLACE OF BIRTH:	Gujarat, India
YEAR OF BIRTH:	1994
EDUCATION:	Emerald Heights International School
	Madhya Pradesh, India, 2012
	A. D. Patel Institute of Technology
	Gujarat, India, 2016
	University of Windsor, M.Sc
	Windsor, ON, 2018