# Finding a Team of Experts in Social Networks

Theodoros Lappas[*]
Computer Science Dept.
UC Riverside, CA, USA
tlappas@cs.ucr.edu

Kun Liu
IBM Almaden
San Jose, CA, USA
kun@us.ibm.com

Evimaria Terzi
IBM Almaden
San Jose, CA, USA
eterzi@us.ibm.com

## ABSTRACT

Given a task $T$, a pool of individuals $\mathcal{X}$ with different skills, and a social network $G$ that captures the compatibility among these individuals, we study the problem of finding $\mathcal{X}'$, a subset of $\mathcal{X}$, to perform the task. We call this the TEAM FORMATION problem. We require that members of $\mathcal{X}'$ not only meet the skill requirements of the task, but can also work effectively together as a team. We measure effectiveness using the *communication cost* incurred by the subgraph in $G$ that only involves $\mathcal{X}'$. We study two variants of the problem for two different communication-cost functions, and show that both variants are NP-hard. We explore their connections with existing combinatorial problems and give novel algorithms for their solution. To the best of our knowledge, this is the first work to consider the TEAM FORMATION problem in the presence of a social network of individuals. Experiments on the DBLP dataset show that our framework works well in practice and gives useful and intuitive results.

## Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

## General Terms

Algorithms, Experimentation, Theory

## Keywords

team formation, social networks, graph algorithms

## 1. INTRODUCTION

The success of a project depends not only on the expertise of the people who are involved, but also on how effectively they collaborate, communicate and work together as a team. Assume, for example, an IT project manager who wants to build a team of engineers skilled in the following areas: $T$={*algorithms, software engineering, distributed systems, web programming*}. Also assume there are five candidates, {*a, b, c, d, e*}, with the following backgrounds: $X_a$={*algorithms*}, $X_b$={*web programming*}, $X_c$={*software engineering, distributed systems*}, $X_d$={*software engineering*} and $X_e$={*software engineering, distributed systems, web programming*}. The relationships among these candidates are represented by the social network shown in Figure 1, where the existence of an edge between two nodes in $G$ indicates that the corresponding persons can collaborate effectively.

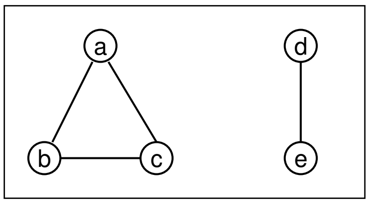


Figure 1: Network of connections between individuals in $\{a, b, c, d, e\}$.

Without considering how effectively these people can collaborate, the manager can select either $\mathcal{X}' = \{a, b, c\}$ or $\mathcal{X}'' = \{a, e\}$ – both these teams have the required skillset. However, the existence of graph $G$ makes $\mathcal{X}' = \{a, b, c\}$ a superior solution, since the structure of $G$ indicates that $a$ and $e$ cannot work together at all.

The existence of a social network between individuals is quite common in real scenarios. In a company, the network may capture the hierarchical organization of the employees. In this case, the graph encodes the fact that people in the same group or department can communicate easier than people working in different divisions. In a research community, the network captures previous successful collaborations among scientists. Other examples of social networks between professionals include LinkedIn (`www.linkedin.com`), Xing (`www.xing.com`) and others.

**The problem:** In this paper, we study the problem of finding a group of individuals who can function as a team to accomplish a specific task. We assume that there exists a pool of $n$ candidates $\mathcal{X} = \{1, \ldots, n\}$, where each candidate $i$ has a set of skills $X_i$. We also assume that these candidates are organized in a *weighted* and *undirected* social graph $G(\mathcal{X}, E)$. The weights on the edges of $G$ should

[*]Part of this work was done when the author was at IBM Almaden Research Center.

*KDD'09,* June 28–July 1, 2009, Paris, France.
Copyright 2009 ACM 978-1-60558-495-9/09/06 ...$10.00.

be interpreted as follows: a low-weight edge between nodes $i, j$ implies that candidate $i$ and $j$ can collaborate and/or communicate more easily than candidates connected with a high-weight edge. These weights can be instantiated in different ways in different application domains. For example, in a company, the weight between two employees may correlate to the length of the path from one employee to another through the organizational chart. In a scientific research community, the weight between two scientists is related to the total number of publications they have coauthored. Interpersonal relationships among individuals can also be used to calculate the weights.

Given a task $T$ that requires a set of skills, our goal is to find a set of individuals $\mathcal{X}' \subseteq \mathcal{X}$, such that every required skill in $T$ is exhibited by at least one individual in $\mathcal{X}'$. Additionally, the members of team $\mathcal{X}'$ should define a subgraph in $G$ with *low communication cost.* The communication cost measures how effectively the team members can collaborate: the lower the communication cost, the better the quality of the team.

**Our contributions:** To the best of our knowledge, we are the first to consider the Team Formation problem in the presence of a social network of individuals. We study two instances of this problem, analyze them rigorously and present algorithms for their solution. Our experiments illustrate that our problem definitions, as well as our algorithms, work well in practice and give useful and intuitive results.

**Roadmap:** The rest of the paper is organized as follows: in Section 2 we review the related work on team formation and task allocation. In Section 4 we formally define the Team Formation problem and identify the two variants that we are going to consider in this paper. In the same section, we also study their computational complexity. In Section 5 we give algorithms for the different variants of the Team Formation problem and in Section 6 we illustrate the usefulness of our methodology on a real collaboration dataset. We conclude the paper in Section 7.

## 2. RELATED WORK

There is a considerable amount of literature on Team Formation in the operations research (OR) community [3, 7, 17, 16]. A trend in this line of work is to formulate the Team Formation problem as an integer linear program (ILP), and then focus on finding an optimal match between people and the demanded functional requirements. The problem is often solved using techniques such as simulated annealing [3], branch-and-cut [17] or genetic algorithms [16]. Interested readers may also identify a high-level connection between this formulation and the assignment problem. The main difference between the studies above and our work is that we explicitly take into account the social graph structure of the individuals when deciding the right group. In most of the previous work, the organizational or social bonds among individuals are ignored and the focus is limited on their skills. Moreover, the problem formulations we provide, and the algorithmic approaches we take, are fundamentally different from those proposed in the OR literature.

The necessity of effective collaboration among individuals in a team has been considered in the past. Fitzpatrick and Askin [9] use the Kolbe Conative Index (KCI) to measure individuals' drive and temperament, which in turn reflects the quality of the team. Chen and Lin in [7] use the Myers-Briggs test to measure the candidates' personality and evaluate their interpersonal relationships as team members. Although these approaches are interesting from the antrhopological/pscychological point of view, they also ignore the existing graph structure among individuals. Therefore, these approaches should be considered complementary to ours.

The network structure between individuals in a workforce pool has been taken into account by Gaston *et al.* [11]. The authors provide an experimental study of how different graph structures among the individuals affect the performance of a team. Although related, the work presented in [11] does not address the computational problem of finding a team of experts in a given network. Some work has also been devoted to the construction of the social network [5, 16], given a pool of skilled individuals.

The dynamics of group-formation processes and their impact on the formation of communities in networks have been recently addressed in [2]. The game-theoretic aspects of the same problem have been studied in [12]. These studies are complementary to ours and mostly focus on providing useful insights about social processes.

## 3. PRELIMINARIES

We assume a pool of candidates consisting of $n$ individuals, $\mathcal{X} = \{1, \ldots, n\}$. We also assume $\mathcal{A} = \{a_1, \ldots, a_m\}$ to be a universe of $m$ skills. Each individual $i$ is associated with a set of skills $X_i \subseteq \mathcal{A}$. If $\alpha_j \in X_i$ we say that individual $i$ *has* skill $a_j$; otherwise individual $i$ does not have skill $a_j$. We often use the set of skills an individual possesses to refer to him. Also, we say that a subset of individuals $\mathcal{X}' \subseteq \mathcal{X}$ possesses skill $a_j$ if there exists at least one individual in $\mathcal{X}'$ that has $a_j$.

A *task* $T$ is simply a subset of skills required to perform a job. That is, $T \subseteq \mathcal{A}$. If $a_j \in T$ we say that skill $a_j$ is *required* by task $T$. We can also define the *cover* of a set of individuals $\mathcal{X}'$ with respect to task $T$, denoted by $C(\mathcal{X}', T)$, to be the set of skills that are required by $T$ and for which there exists at least one individual in $\mathcal{X}'$ that has them. That is, $C(\mathcal{X}', T) = T \cap \left( \cup_{i \in \mathcal{X}'} X_i \right)$. Given a skill $a \in \mathcal{A}$, we define its *support set* (or simply *support*), denoted by $S(a)$, to be the set of individuals in $\mathcal{X}$ that has this skill. That is, $S(a) = \{i \mid i \in \mathcal{X} \text{ and } a \in X_i\}$.

As we have already discussed, we assume that individuals are organized in an *undirected* and *weighted* graph $G(\mathcal{X}, E)$. Every node of $G$ corresponds to an individual in $\mathcal{X}$; $E$ is the set of edges connecting the nodes. The edges of $G$ are weighted; edges of low (high) weight represent low (high) communication cost between the nodes they connect. Without loss of generality, we assume that the graph $G$ is connected; we can transform every disconnected subgraph to a connected one by simply adding very high-weight edges between every pair of nodes that belong to different connected components. Note that this very high weight is a number higher than the sum of all pairwise shorted paths in $G$.

For every two nodes $i, i' \in \mathcal{X}$ we define the *(graph) distance* function $d(i, i')$ to be the weight of the shortest path between $i$ and $i'$ in $G$. Note that this distance function between the nodes is a metric and thus satisfies the triangle inequality. For every pair of nodes we also use $\mathit{Path}(i, i')$ to represent the set of nodes that are along the shortest path from $i$ to $i'$. Apart from computing the distance between two nodes in $G$, we will often need the distance between a node $i \in \mathcal{X}$ and a set of nodes $\mathcal{X}' \subseteq \mathcal{X}$. We define this to be $d(i, \mathcal{X}') = \min_{i' \in \mathcal{X}'} d(i, i')$. In this case, we use $\mathit{Path}(i, \mathcal{X}')$

to represent the set of nodes that are along the shortest path from $i$ to the node $j = \arg\min_{i' \in \mathcal{X}'} d(i, i')$.

Finally, given graph $G$ and $\mathcal{X}' \subseteq \mathcal{X}$, we use $G[\mathcal{X}']$ to denote the subgraph of $G$ that contains only the nodes in $\mathcal{X}'$.

## 4. PROBLEMS

In this section, we formally define the TEAM FORMATION problem that we address in this paper. Our problem definitions reflect our belief that efficient communication among team members is an important factor for the successful completion of a task.

### 4.1 Problem Definition

PROBLEM 1. *[*TEAM FORMATION*] Given the set of $n$ individuals $\mathcal{X} = \{1, \ldots, n\}$, a graph $G(\mathcal{X}, E)$, and task $T$, find $\mathcal{X}' \subseteq \mathcal{X}$, so that $C(\mathcal{X}', T) = T$, and the* communication cost *$\text{Cc}(\mathcal{X}')$ is minimized.*

In order to stress the generality of the TEAM FORMATION problem, we have deliberately avoided defining the communication cost in the definition of Problem 1. In this paper, we focus on two instantiations of the communication-cost function. We chose these instantiations as we believe they are practical, simple and intuitive.

**Diameter** ($R$)**:** Given graph $G(\mathcal{X}, E)$ and a set of individuals $\mathcal{X}' \subseteq \mathcal{X}$, we define the *diameter communication cost* of $\mathcal{X}'$, denoted by $\text{Cc-R}(\mathcal{X}')$, to be the diameter of the subgraph $G[\mathcal{X}']$. Recall that the diameter of a graph is the largest shortest path between any two nodes in the graph.

**Minimum Spanning Tree** (MST)**:** Given graph $G(\mathcal{X}, E)$ and $\mathcal{X}' \subseteq \mathcal{X}$ we define the MST *communication cost* of $\mathcal{X}'$, denoted by $\text{Cc-Mst}(\mathcal{X}')$, to be the cost of the *minimum spanning tree* on the subgraph $G[\mathcal{X}']$. Recall that the cost of a spanning tree is simply the sum of the weights of its edges.

We call the TEAM FORMATION problem with communication function CC-R, the DIAMETER-TF problem. Similarly, we refer to the TEAM FORMATION problem with communication function CC-MST as the MST-TF problem.

PROPOSITION 1. *The* DIAMETER-TF *problem is NP-complete.*

PROOF. We prove the proposition by a reduction from the MULTIPLE-CHOICE COVER (MCC) problem [1]. An instance of the MCC problem consists of a universe $V = \{1, \ldots, N\}$ of $N$ elements, a $N \times N$ symmetric real matrix $D$ with non-negative entries, and a $\mathcal{S} = \{S_1, \ldots, S_k\}$ such that each $S_i \subseteq V$. Given constant $K$, the decision version of the MCC problem asks whether there exists $V' \subseteq V$ such that for every $i \in \{1, \ldots, k\}$, $|V' \cap S_i| > 0$ and $\max_{(u,v) \in V' \times V'} D(u, v) \leq K$.

We transform an instance of the MCC problem to an instance of the DIAMETER-TF problem as follows: for every set $S_i$ in the MCC problem we create a skill $a_i$. The task $T$ to be performed requires all the $k$ skills. That is, $T = \{a_1, \ldots, a_k\}$. For every element $v \in V$ of the MCC instance, we create an individual $i_v$ with skills $X_v = \{a_i \mid v \in S_i\}$. Two individuals $i_v$ and $i'_v$ are connected in the graph $G$ by an undirected edge with weight equal to $D(v, v')$. Given this mapping it is easy to show that there exists a solution to the MCC problem with cost at most $K$ if and only if there exists a solution to the DIAMETER-TF problem with CC-R cost at most $K$. The problem is trivially in NP. □

Note that the above reduction does not assume anything about the distance function between the nodes in $G$. However, from [1], we know that the MCC problem is NP-hard even when the distance matrix $D$ corresponds to a metric. Therefore, the DIAMETER-TF problem is NP-hard when the distance function $d$ between the individuals in $G$ is a metric. Observe that the above reduction is *approximation preserving*. Therefore, the approximation properties of the MCC problem described in [1] carry over to the DIAMETER-TF problem as well.

For the MST-TF problem, we have the following hardness result:

PROPOSITION 2. *The* MST-TF *problem is NP-complete.*

PROOF. We prove the proposition by a reduction from the GROUP STEINER TREE (GST) problem [13]. An instance of the GST problem consists of an undirected graph $G(V, E)$, cost function $c : E \rightarrow \mathbf{R}$ and $k$ subsets of vertices (called groups) $\{g_1, \ldots, g_k\}$ with $g_i \subseteq V, i \in \{1, \ldots, k\}$.

Given constant $K$, the decision version of the GST problem asks whether there exists a subtree $T(V', E')$ of $G(V, E)$ (i.e., $V' \subseteq V$ and $E' \subseteq E$) such that $|V' \cap g_i| > 0$ for every $i \in \{1, \ldots, k\}$ and cost $\sum_{e \in E'} c(e) \leq K$.

We transform an instance of the GST problem to an instance of the MST-TF problem as follows: for every group $g_i$ in the GST problem we create a skill $a_i$. The task $T$ to be performed requires all the $k$ skills. That is, $T = \{a_1, \ldots, a_k\}$. For every node $v \in V$ of the GST problem we create an individual $i_v$ with skills $X_v = \{a_i \mid v \in g_i\}$. The graph $G'$ of the MST-TF problem is identical to the graph $G$ of the GST problem, where the cost function $c$ determines the weights of the edges in the MST-TF instance of the problem. Given this mapping it is easy to show that there exists a tree solution to the GST problem with cost at most $C$ if and only if there exists a solution to the MST-TF problem with CC-MST cost at most $C$. The problem is trivially in NP. □

As before, note that the proofs above do not assume anything about the distance function between individuals in $G$. However, since the GST problem remains NP-hard even when the graph edge weights satisfy the triangle inequality, so does the MST-TF. As in the case of the DIAMETER-TF problem, the above reduction is approximation preserving. Therefore, the approximation properties of the GST problem ([6] and references therein) carry over to the MST-TF problem as well.

### 4.2 Discussion

In the definition of the TEAM FORMATION problem and its specializations, we focused on minimizing the communication cost among team members. Other notions of the "effectiveness" of a team can lead to different optimization functions. For example, if the communication cost was not a concern, we could define as our goal to find $\mathcal{X}' \subseteq \mathcal{X}$, such that $C(\mathcal{X}', T) = T$ and $|\mathcal{X}'|$ are minimized. Such a problem definition ignores the existence of the underlying graph $G(\mathcal{X}, E)$, and is actually an instance of the classic SET COVER problem, which can be solved by the standard `GreedyCover` algorithm. Details are presented in Sections 5.2 and 6.

Optimizing both the cardinality of the team and the communication cost between its members would require the minimization of a function of the form $\alpha\cdot|\mathcal{X}'|+(1-\alpha)\cdot\text{Cc}(\mathcal{X}',G)$, where $\alpha \in [0,1]$. For $\alpha = 1$ the problem seeks for teams with the minimum cardinality. For $\alpha = 0$ this problem is the Team Formation problem. However, for values of $\alpha$ in $(0,1)$ it is not clear that optimizing this alternative function makes sense; this is mostly because the two terms in the sum are in different scales and there is no knowledge on how these scales relate.

Alternatively, these two objectives (team size and communication cost) could be taken into account simultaneously by defining the problem as a bi-objective optimization problem. In such cases the goal is to find *Pareto-optimal solutions* [4]. Note that a solution is called Pareto-optimal if there does not exist another solution that is better in both objectives. For many problems, the set of Pareto-optimal solutions is exponential to the size of the input and thus cannot be found in polynomial time. Although we do not study this bi-objective version of the problem in this paper, we note that a solution with *minimum communication cost* implicitly requires a small team, since larger teams typically result in higher communication costs.

In our setting, we assume that individuals either have a skill or not; we do not allow for a scaling of the nodes' abilities. Similarly for the tasks; we assume that a task requires a certain set of skills, without considering the special importance that different skills might have for the completion of the task. Therefore, a straightforward generalization of the Team Formation problem would be its *graded* variant. In such a variant, the degree of skillfulness of individuals and the extent to which a skill is required for the completion of a task can be modelled by means of an integer weight in some interval, e.g., $\{0,1,\ldots,\delta\}$. In this case, the task specification explicitly states for every required skill $a_j \in T$ the minimum level requirement $\delta_j$. Similarly, for every individual $i$ with skill $a_j$, the level of his competence with respect to $a_j$ is specified. Then, all individuals with competence level higher or equal to the minimum required level are capable of contributing in covering this skill for the given task. Conceptually, we assume that an individual has a skill, only if his respective competence level is equal or higher to the required level. In this way, this "graded" version of the problem becomes identical to the basic version of the Team Formation problem, studied in this paper.

## 5. ALGORITHMS

In this section, we present algorithms for the Diameter-Tf and Mst-Tf problems. Our algorithmic solutions exploit the relationship of these two problems with the Mcc and Gst problems, respectively.

### 5.1 Algorithms for the Diameter-Tf problem

Algorithm 1 shows the pseudocode of the `RarestFirst` algorithm for the Diameter-Tf problem. The algorithm is a variation of the *Multichoice* algorithm presented in [1]. First, for every skill $a$ required by the task $T$, we compute $S(a)$, the support of $a$. Then, the algorithm picks the skill $a_{\text{rare}} \in T$ with the lowest-cardinality support $S(a_{\text{rare}})$. Note that at least one individual from the set $S(a_{\text{rare}})$ needs to be included in the solution. Among all candidates from the set $S(a_{\text{rare}})$, the algorithm picks the one that leads to the smallest diameter subgraph, when connected to its closest individual in all other support groups $S(a)$ ($a \in T$ and $a \neq a_{\text{rare}}$).

**Algorithm 1** The `RarestFirst` algorithm for the Diameter-Tf problem.

```
    Input: Graph $G(\mathcal{X},E)$; individuals' skill vectors $\{X_1,\ldots,X_n\}$ and task $T$.
    Output: Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
1: for every $a \in T$ do
2:     $S(a) = \{i \mid a \in X_i\}$
3: $a_{rare} \leftarrow \arg\min_{a\in T}|S(a)|$
4: for every $i \in S(a_{rare})$ do
5:     for $a \in T$ and $a \neq a_{rare}$ do
6:         $R_{ia} \leftarrow d\big(i, S(a)\big)$
7:     $R_i \leftarrow \max_a R_{ia}$
8: $i^* \leftarrow \arg\min R_i$
9: $\mathcal{X}' = i^* \cup \{Path(i^*, S(a)) \mid a \in T\}$
```

Recall that in line 6 of Algorithm 1, $d\big(i, S(a)\big)$ is simply $\min_{i'\in S(a)} d(i,i')$. Also recall that $Path(i^*, S(a))$ in line 9 is the set of nodes in the graph that are along the shortest path from $i^*$ to $i'$, where $i'$ is such that $i' \in S(a)$ and $d\big(i^*, S(a)\big) = d(i^*, i')$. We assume that all pairs shortest path have been pre-computed, and we use hash tables for storing the attributes of every individual and a different set of hashtables for storing the individuals that posses a specific attribute. Then, the running time of the `RarestFirst` algorithm is $\mathcal{O}(|S(a_{rare})| \times n)$. A worst-case analysis suggests that $|S(a_{rare})| = \mathcal{O}(n)$. Thus the worst-case running time of the `RarestFirst` is $\mathcal{O}(n^2)$. However, in practice, the running time of the algorithm is much less that this worst-case analysis suggests.

Since the employed distance function $d$ is a metric, we can state the following for the approximation factor of the `RarestFirst` algorithm:

Proposition 3. *For any graph-distance function $d$ that satisfies the triangle inequality, the* Cc-R *cost of the solution $\mathcal{X}'$, given by* `RarestFirst` *for a given task, is at most twice the* Cc-R *cost of the optimal solution $\mathcal{X}^*$. That is,* $\text{Cc-R}(\mathcal{X}') \leq 2 \cdot \text{Cc-R}(\mathcal{X}^*)$.

Proof. The analysis we present here is similar to the analysis of the *Multichoice* algorithm presented in [1]. First, consider the solution $\mathcal{X}'$ output by the `RarestFirst` algorithm, and let $a_{rare} \in T$ be the skill possessed by the least number of individuals in $\mathcal{X}$. Also, let $i^*$ be the individual picked from set $S(a_{rare})$ to be included in the solution $\mathcal{X}'$. Now consider two other skills $a \neq a' \neq a_{rare}$ and individuals $i, i' \in \mathcal{X}'$ such that $i \in S(a), i \notin S(a')$ and $i' \in S(a'), i' \notin S(a)$. If $i, i'$ are part of the team reported by the `RarestFirst` algorithm, it means that $i = \arg\min_{j\in S(a)} d(i^*, j)$ and $i' = \arg\min_{j\in S(a')} d(i^*, j)$. Due to the way the algorithm operates, we can lowerbound the Cc-R cost of the optimal solution as follows:

$$\text{Cc-R}(\mathcal{X}^*) \geq d(i^*, i) \text{ and } \text{Cc-R}(\mathcal{X}^*) \geq d(i^*, i')\,. \qquad (1)$$

Since we have assumed that the distance function $d$ satisfies the triangle inequality we also have that $d(i,i') \leq d(i^*,i)+$

$d(i^*, i')$. By applying the bounds given in (1) in the triangle inequality we get the proposed approximation factor.

$$\begin{aligned} d(i, i') &\leq d(i^*, i) + d(i^*, i') \\ &\leq \textsc{Cc-R}(\mathcal{X}^*) + \textsc{Cc-R}(\mathcal{X}^*) \\ &= 2 \cdot \textsc{Cc-R}(\mathcal{X}^*). \end{aligned}$$

□

## 5.2 Algorithms for the Mst-Tf problem

In this section we describe two algorithms for solving the Mst-Tf problem: the `CoverSteiner` and `EnhancedSteiner` algorithms. Both algorithms are motivated by the resemblance of Mst-Tf to Steiner tree problems.

### 5.2.1 *The* `CoverSteiner` *algorithm*

**Algorithm 2** The `CoverSteiner` algorithm for the Mst-Tf problem.

```
    Input: Graph $G(\mathcal{X}, E)$; individuals' skill vectors $\{X_1, \ldots, X_n\}$ and task $T$.
    Output: Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
 1: $\mathcal{X}_0 \leftarrow$ GreedyCover($\mathcal{X}, T$)
 2: $\mathcal{X}' \leftarrow$ SteinerTree($G, \mathcal{X}_0$)
```

The first heuristic we present for the Mst-Tf problem proceeds in two steps. In the first step, the social network is ignored and the algorithm focuses on finding a set of individuals $\mathcal{X}_0 \subseteq \mathcal{X}$ such that $\cup_{i \in \mathcal{X}_0} X_i \supseteq T$. In the second step, the algorithm finds the minimum cost tree that spans all the nodes in $\mathcal{X}_0$, and possibly other nodes in $\mathcal{X} \setminus \mathcal{X}_0$. In that way, a set of nodes $\mathcal{X}'$ such that $\mathcal{X}_0 \subseteq \mathcal{X}' \subseteq \mathcal{X}$ is reported. We call this two-step algorithm the `CoverSteiner` algorithm.

The pseudocode of this algorithm is given in Algorithm 2. The goal of the first step is to solve an instance of the classical Set Cover problem: the universe of elements to be covered are the requirements of task $T$ and each individual in $\mathcal{X}$ is a subset of the universe. To solve this, we use the standard `GreedyCover` algorithm for the Set Cover problem. The `GreedyCover` algorithm is an iterative greedy procedure, adding at each step $t$ the individual $X_t$ that possesses the most yet uncovered required skills in $T$. For details on this algorithm see [15].

In its second step, the `CoverSteiner` algorithm solves an instance of the Steiner Tree problem on graph $G$. Recall that in the standard Steiner Tree problem, we are given an undirected graph with non-negative edge costs. The vertices of this graph are partitioned into two sets: the *required* and the *Steiner* vertices. The Steiner Tree problem then asks for the minimum-cost tree in the input graph that contains all required vertices and any subset of the Steiner vertices. In our case, the set of nodes $\mathcal{X}_0$ reported by the `GreedyCover` algorithm corresponds to the set of required vertices, while the vertices in $\mathcal{X} \setminus \mathcal{X}_0$ represent the Steiner vertices. Given graph $G(\mathcal{X}, E)$, the goal of line 2 of Algorithm 2 is to find the solution $\mathcal{X}'$ that minimizes Cc-Mst$(\mathcal{X}')$, under the constraint that $\mathcal{X}' \supseteq \mathcal{X}_0$.

There exist many algorithms for solving the classical Steiner Tree problem. The pseudocode of the algorithm we use for our experiments is given in Algorithm 3. We call this algorithm the `SteinerTree`. The algorithm is due to [14], and is in fact a greedy heuristic for the Steiner Tree. The algorithm incrementally adds to the current solution $\mathcal{X}'$ nodes from the required set $\mathcal{X}_0$. At every step, a single node from $\mathcal{X}_0$ is added; this is the node $v^*$ that has the minimum distance to the set of nodes $\mathcal{X}'$ already added to the solution (line 3). If such node exists $v^*$ along with all the nodes in the shortest path from it to $\mathcal{X}'$ are added to the solution set. Otherwise, failure is reported.

**Algorithm 3** The `SteinerTree` algorithm.

```
    Input: Graph $G(\mathcal{X}, E)$; required nodes $\mathcal{X}_0$ and Steiner nodes $\mathcal{X} \setminus \mathcal{X}_0$.
    Output: Team $\mathcal{X}_0 \subseteq \mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
 1: $\mathcal{X}' \leftarrow v$, where $v$ is a random node from $\mathcal{X}_0$.
 2: while $(\mathcal{X}_0 \setminus \mathcal{X}') \neq \emptyset$ do
 3:     $v* \leftarrow \arg\min_{u \in \mathcal{X}_0 \setminus \mathcal{X}'} d(u, \mathcal{X}')$
 4:     if $Path(v^*, \mathcal{X}') \neq \emptyset$ then
 5:         $\mathcal{X}' \leftarrow \mathcal{X}' \cup \{Path(v^*, \mathcal{X}')\}$
 6:     else
 7:         Return Failure
```

The running time of the `CoverSteiner` algorithm is the summation of the running times of `GreedyCover` and `SteinerTree`. The time required for the execution of the `GreedyCover` algorithm is $\mathcal{O}(|T| \times |\mathcal{X}|)$ or $\mathcal{O}(mn)$. The time required for the execution of `SteinerTree` shown in Algorithm 3 is $\mathcal{O}(|\mathcal{X}_0| \times |E|)$. Thus, in the worst case, the running time of `CoverSteiner` is $\mathcal{O}(n^3)$ (this is because $|\mathcal{X}_0| = \mathcal{O}(n)$ and $|E| = \mathcal{O}(n^2)$). However, in practice the cardinalities of sets $\mathcal{X}_0$ and $E$ are much less than their worst-case upper bounds.

The main disadvantage of the `CoverSteiner` algorithm is that, in the first step, it completely ignores the underlying graph structure. This can lead to teams with a high commnication cost, or may even lead to failure, even in cases where a solution to the Mst-Tf problem actually exists.

### 5.2.2 *The* `EnhancedSteiner` *algorithm*

The inadequacies of the `CoverSteiner` algorithm can be alleviated by the `EnhancedSteiner` algorithm that we describe in this section.

The `EnhancedSteiner` algorithm starts by first enhancing graph $G$ with additional nodes and edges to form the *enhanced graph* $H$. Then, `SteinerTree` is evoked to solve the Steiner Tree problem on the enhanced graph $H$ (for similar applications of Steiner tree algorithms see [8]). The pseudocode that corresponds to these two steps of the `EnhancedSteiner` algorithm is shown in Algorithm 4.

**Algorithm 4** The `EnhancedSteiner` algorithm for the Mst-Tf problem.

```
    Input: Graph $G(\mathcal{X}, E)$; individuals' skill vectors $\{X_1, \ldots, X_n\}$ and task $T$.
    Output: Team $\mathcal{X}' \subseteq \mathcal{X}$ and subgraph $G[\mathcal{X}']$.
 1: $H \leftarrow$ EnhanceGraph$(G, T)$
 2: $\mathcal{X}_H \leftarrow$ SteinerTree($H, \{Y_1, \ldots, Y_k\}$)
 3: $\mathcal{X}' \leftarrow \mathcal{X}_H \setminus \{Y_1, \ldots, Y_k\}$
```

Let the task to be performed require $k$ skills, i.e., $T = \{a_1, \ldots, a_k\}$. The routine `Enhance` (line 1 of Algorithm 4) makes a linear pass over the graph $G$ and enhances it as follows: an additional node $Y_j$ is created for every skill $a_j \in$

$T$. Each such new vertex $Y_j$ is connected to a node $i \in \mathcal{X}$ if and only if $a_j \in X_i$. The distance between node $Y_j$ and nodes $i \in S(a_j)$ are set to be $d(Y_j, i) = D$ where $D$ is a large real number, larger than the sum of all the pairwise distances of the nodes in the graph $G$. Finally, every node $i \in \mathcal{X}$ that has abilities $X_i$ is replaced by a clique $C_i$ of size $|X_i|$. Each node in the clique $C_i$ should be considered as a copy of individual $i$ that has only a single distinct skill from the set $X_i$. The distance between every two nodes in the clique $C_i$ is set to zero. Each node in the clique $C_i$ maintains all the existing connections of node $i$ to the rest of the graph – including the connections to nodes $\{Y_1, \ldots, Y_k\}$.

The set of nodes $\mathcal{X}_H$ that participate in the Steiner tree of the enhanced graph $H$ are found by calling the `SteinerTree` algorithm with required nodes $Y_1, \ldots, Y_k$. In a final step, the algorithm removes from set $\mathcal{X}_H$ the artificially added nodes $Y_1, \ldots, Y_k$ (and their incident edges) to obtain the final solution $\mathcal{X}'$.

The following claim can be made with respect to this algorithm. Let $\mathcal{X}_H^*$ be the set of nodes in the *optimal Steiner tree* of the enhanced graph $H$, and $\mathcal{X}^*$ be the optimal team for the MST-TF problem. Then, we have that $\text{Cc-Mst}(\mathcal{X}^*) = \text{Cc-Mst}(\mathcal{X}_H^* \setminus \{Y_1, \ldots, Y_k\})$. That is, if we remove nodes $Y_1, \ldots, Y_k$ (and their incident edges) from the optimal solution of the Steiner tree problem on the enhanced graph $H$, then the remaining nodes form the optimal solution to the MST-TF problem.

Observe that the replacement of every individual $i$ with a clique $C_i$ of size $|X_i|$ is only conceptual. In practice, the implementation of the algorithm does not require this. Therefore, the enhanced graph $H$ contains only $k$ more nodes than the input graph $G$, namely the nodes $Y_1, \ldots, Y_k$. Therefore, following the analysis of the `SteinerTree` done in the previous section, we have that the running time of the `EnhancedSteiner` algorithm is $\mathcal{O}(k \times |E|)$.

The `EnhancedSteiner` algorithm is in fact motivated by the obvious similarity between the MST-TF problem and the GROUP STEINER TREE (GST) problem; the connection was already highlighted in the proof of Proposition 2. In general, instead of the `EnhancedSteiner` algorithm any other (approximation) algorithm for the GST problem can also be used to solve the MST-TF problem. We have picked the `EnhancedSteiner` algorithm because it is simple, intuitive and works well in practice. The best approximation ratio achieved by an algorithm is $O(\log^3 n \log k)$ [10]. For a review of some recent approximation algorithms for the GST problem see [6, 8, 10] and references therein.

## 6. EXPERIMENTAL EVALUATION

In this section we evaluate the proposed algorithms for the TEAM FORMATION problem using the scientific-collaboration graph extracted from the DBLP bibliography server. We show that our algorithms for both the DIAMETER-TF and MST-TF problems give high-quality results in terms of the *communication cost*, *the cardinality of the team*, and *the connectivity of the team.* Examples of teams reported by our methods illustrate the effectiveness of our framework in real scenarios.

### 6.1 Other algorithms

In addition to the algorithms we described in Section 5, we also experiment with some straightforward greedy heuristics that would be natural alternatives for solving the TEAM FORMATION problem. The rationale of these algorithms is to form a solution iteratively. At round $t$, team $\mathcal{X}_t$ is formed by adding to the team $\mathcal{X}_{t-1}$ a node $i \in \mathcal{X} \setminus \mathcal{X}_{t-1}$. The node $i$ is selected so that it maximizes the ratio

$$i = \underset{i' \in \mathcal{X} \setminus \mathcal{X}_{t-1}}{\arg\max} \frac{\left|C\big(\mathcal{X}_{t-1} \cup Path(\mathcal{X}_{t-1}, i'), T\big) - C\big(\mathcal{X}_{t-1}, T\big)\right|}{\text{Cc}\big(\mathcal{X}_{t-1} \cup Path(\mathcal{X}_{t-1}, i')\big)}.$$

That is, the node $i$ that achieves the best ratio of newly covered skills in $T$ divided by the corresponding communication cost is picked. We refer to the variation of the greedy algorithm that uses the CC-R (resp. CC-MST) communication-cost function, as `GreedyDiameter` (resp. `GreedyMST`).

### 6.2 The DBLP dataset

We use a snapshot of the DBLP data taken on April 12, 2006 to create a benchmark dataset for our experiments. We only keep entries of the snapshot that correspond to papers published in the areas of *Database* (DB), *Data mining* (DM), *Artificial intelligence* (AI) and *Theory* (T) conferences. For each paper, we have information about its authors (names), title, the forum where it was published and the year of publication. We end up with a total of 19 venues categorized as follows: DB = {SIGMOD, VLDB, ICDE, ICDT, EDBT, PODS}, DM = {WWW, KDD, SDM, PKDD, ICDM}, AI = {ICML, ECML, COLT, UAI} and T = {SODA, FOCS, STOC, STACS}. We refer to the set of selected papers as the **DBLP** dataset.

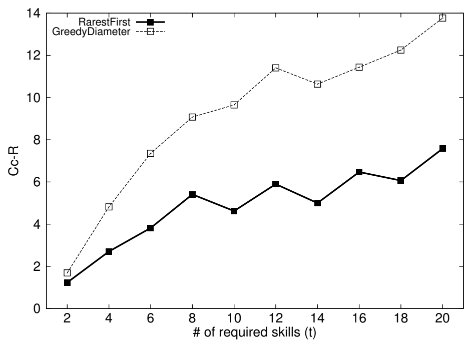
We now proceed to generate the input to the TEAM FORMATION Problem as follows. The set of skilled individuals $\mathcal{X}_{\text{dblp}}$ consists of the set of authors that have at least three papers in the **DBLP** dataset. The skillset $X_i$ of each such author $i$ consists of the set of terms that appear in *at least two titles* of papers in **DBLP** that he has co-authored. The above procedure creates a set $\mathcal{X}_{\text{authors}}$ consisting of 5508 individuals and 1792 distinct skills. Two authors $i, i'$ are connected in the graph $G_{\text{dblp}}(\mathcal{X}_{\text{dblp}}, E)$ if they appear as co-authors in *at least two papers* in **DBLP**. This threshold leads to a graph $G_{\text{dblp}}$ that has 5588 total edges. The weight of an edge connecting nodes $i, i'$ is $w(i, i') = 1 - \frac{|P_i \cap P_{i'}|}{|P_i \cup P_{i'}|}$; $P_i$ (resp., $P_{i'}$) is the set of papers authored by $i$ (resp., $i'$). In other words, the weights on the edges represent pairwise Jaccard distances between all pairs of connected nodes. We compute the graph distance between two nodes in graph $G_{\text{dblp}}$ using the shortest path distance as we described in Section 3.

### 6.3 Performance Evaluation

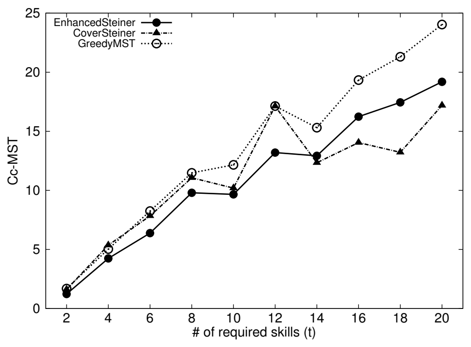
This section evaluates the TEAM FORMATION algorithms on the *communication cost*, the *cardinality of the team* and the *connectivity of the team.*

**Task generation:** Every generated task is characterized by two parameters: 1) $t$ – the number of required skills in the task; and 2) $s$ – the diversity of the required skills in terms of their corresponding areas. We use $T(t, s)$ to refer to a task generated for a specific configuration of these parameters.

Specifically, a task $T(t, s)$ is generated as follows: first, we select a subset of the research areas $S \subseteq \{$ DB, DM, AI ,T $\}$ with $|S| = s$. Then, we randomly pick $t$ required skills from the terms appearing in papers published in conferences belonging to these areas. For the results we report in this section we use $t \in \{2, 4, \ldots, 20\}$ and $s = 1$. For every $(s, t)$ configuration we generate 100 random tasks for this con-

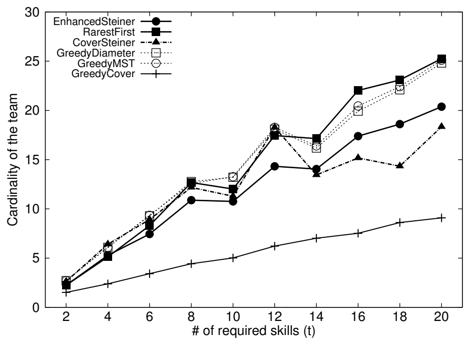


(a) Cc-R cost

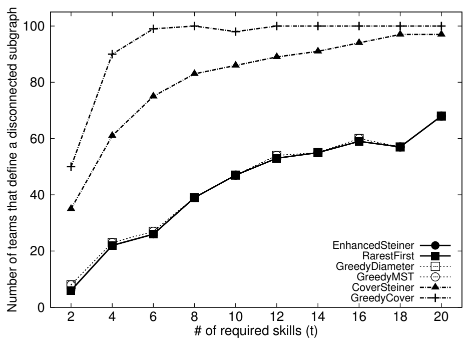


(b) Cc-Mst cost

Figure 2: Average communication cost of the teams produced by each Team Formation algorithm for tasks $T(t,1)$ with $t \in \{2,4,\ldots,20\}$. Figure 2(a): Average Cc-R cost of `RarestFirst` and `GreedyDiameter` algorithms. Figure 2(b): Average Cc-Mst cost of `EnhancedSteiner`, `CoverSteiner` and `GreedyMST` algorithms.



(a) Cardinality of the team.



(b) Number of disconnected teams.

Figure 3: Figure 3(a): Average cardinality of the teams reported by `RarestFirst`, `EnhancedSteiner`, `CoverSteiner`, `GreedyDiameter`, `GreedyMST`, `GreedyCover`. Figure 3(b): Number of reported teams that define a subgraph with disconnected components. The count is taken over 100 independent tasks generated for every $T(t,1)$ where $t \in \{2,4,\ldots,20\}$.

figuration and report the average results obtained by the different methods. Experiments for $s = 2,3,4$ exhibit similar trends as those for $s = 1$ and thus are not presented due to space constraints.

**Communication cost:** Figure 2(a) shows the average Cc-R costs of the solutions achieved by `RarestFirst` and `Greedy-Diameter` on tasks $T(t,1)$ with $t \in \{2,4,\ldots,20\}$. Figure 2(b) shows the average Cc-Mst costs of the `EnhancedSteiner`, `CoverSteiner` and `GreedyMST` algorithms on the same set of tasks. Note that the average is calculated for the solutions $\mathcal{X}'$ that result in a connected graph $G[\mathcal{X}']$. If, for a specific task, the solution produced by a specific algorithm does not lead to a connected graph, we simply ignore it.

It can be observed that, in terms of the diameter cost, `RarestFirst` significantly outperforms `GreedyDiameter`. Similarly, in terms of the MST cost, `EnhancedSteiner` generally gives better results than `CoverSteiner` and `GreedyMST`. The conclusion is that our proposed algorithms can form teams that are able to accomplish a given task with low communication efforts.

**Cardinality of the team:** Since the size of the team often has a positive correlation with the expenses of a project, we evaluate the cardinality of the teams formed by every Team Formation algorithm. The results in Figure 3(a) show that the `RarestFirst` algorithm tends to report relatively large teams, especially for large values of $t$. On the other hand, the `EnhancedSteiner` algorithm generally finds teams of small size. This can be explained by the fact that the `RarestFirst` algorithm aims to minimize the diameter of the graph, which is less likely to be affected by the introduction of new nodes. On the other hand, the `EnhancedSteiner` algorithm tries to minimize the Mst cost, which is always increased when a new node is added to the team.

For comparison purposes, we also include the cardinality of the teams reported by the `GreedyCover` algorithm. Recall that `GreedyCover` ignores the existence of the graph and only reports a set of individuals who can perform the task by simply looking at their skillsets. Therefore, the cardinality of this solution is a lower bound on the cardinality of the solutions produced by all the five aforementioned algorithms. However, since `GreedyCover` ignores the graph structure, it often forms teams of members that cannot communicate. That is, the subgraph of the original graph defined by the members of such teams is not connected. The following experiment illustrates the validity of this claim.

**Connectivity of the team:** Given a task $T$, it might be the case that there does not exist a team $\mathcal{X}'$ such that the members of $\mathcal{X}'$ simultaneously have all the skills required by

$T$ and define a connected subgraph. Further, even if such a team exists, it might be the case that some algorithms fail to find it. In this experiment, we evaluate the effectiveness of the different algorithms in finding teams that correspond to connected subgraphs of the original graph. Recall that connected subgraphs have significantly lower communication costs (both Cc-R and Cc-Mst) than disconnected ones.

Figure 3(b) shows, for every algorithm and every $t \in \{2, 4, \ldots, 20\}$, the number of times a team formed by an algorithm defines a disconnected subgraph. The count is taken over the 100 independent tasks generated for every $T(t, 1)$. We can observe that `RarestFirst`, `GreedyDiameter`, `Enhanced-Steiner` and `GreedyMST` produce approximately the same number of disconnected teams. We conjecture that the tasks for which these algorithms fail to report a connected subgraph are in fact those that have no connected team as a solution. On the other hand, `CoverSteiner` and `GreedyCover` often fail to find a connected team, even in cases where such a team actually exists. The results indicate that, although `GreedyCover` produces teams of small size, the members of this team cannot communicate efficiently.

## 6.4 Qualitative evidence

The goal of this experiment is show that our problem definitions and their corresponding algorithms produce reasonable and intuitive results in practical settings. As input to our problem, we again consider the individual authors in $\mathcal{X}_{\text{dblp}}$ and the corresponding co-authorship graph $G_{\text{dblp}}$, that we described in Section 6.2. We test our framework on 10 distinct tasks. The required skills for each task are defined by the words appearing in the title of an already published paper. The papers were chosen from the "Most Cited Computer Science Articles" list, maintained by CiteSeerX (`citeseerx.ist.psu.edu/stats/articles`). We thus form 10 tasks by selecting the top-10 cited papers from the list, which were also published in one of the 19 conferences covered by the **DBLP** Dataset. Table 1 shows the titles of the these papers.

Table 2 shows the ten teams of authors obtained by the `RarestFirst` and `EnhancedSteiner` algorithms. The set of original authors for every paper is also reported. The names highlighted in bold in the last two columns of the table indicate authors that have been selected because they covered some required skill of the input task. The names appearing not in bold correspond to authors that were included in the team as mediators, i.e., communication nodes that ensure the connectivity of the graph.

We can observe that for papers 3, 6, and 9, `RarestFirst` finds a single-node solution, whereas `EnhancedSteiner` fails to do so. This is due to the fact that `EnhancedSteiner` starts with a random node from $\mathcal{X}_0$, so it may be the case that none of the nodes in the final team possesses all the required skills. On the other hand, `RarestFirst` examines every node who has the skill with the lowest-cardinality support. If a node of them happens to have all other required skills, the process simply reports that node and terminates.

In general, both algorithms produce teams of reasonable size; note that not too many mediator nodes (nodes without skill contribution) are introduced. In many cases, the actual authors of a paper were included in the formed team. This is reasonable, since the real teams are more likely to combine skill coverage with a low communication cost. This attests not only to the effectiveness of the algorithms, but also to the validity of the problem definitions.

Table 1: Titles of the top-10 most cited papers from the **DBLP** dataset according to CiteSeerX citation counting. The keywords appearing in the tiles define the required skills of 10 distinct tasks.

| Rank | Paper title |
|---|---|
| **1** | The anatomy of a large-scale hypertextual Web search engine |
| **2** | Fast algorithms for mining association rules |
| **3** | Mining association rules between sets of items in large databases |
| **4** | Text categorization with support vector machines: Learning with many relevant features |
| **5** | Conditional random fields: Probabilistic models for segmenting and labeling sequence data |
| **6** | Mining frequent patterns without candidate generation |
| **7** | A survey of approaches to automatic schema matching |
| **8** | Automatic subspace clustering of high dimensional data for data mining applications |
| **9** | Models and issues in data stream systems |
| **10** | NiagaraCQ: A Scalable Continuous Query System for Internet Databases |

## 7. CONCLUSIONS

In this paper, we addressed the problem of forming a team of skilled individuals to perform a given task, while minimizing the communication cost among the members of the team. We explored two alternative formulations for the communication cost, which we believe are practical and intuitive. We proved that the Team Formation problem is NP-Hard for both formulations and proposed appropriate approximation algorithms. In a thorough experimental evaluation, we evaluated the performance of our algorithms, and compared them against reasonable baseline approaches. We concluded with a qualitative evaluation, reporting the teams formed by our algorithms on a set of real tasks.

## 8. REFERENCES

[1] E. M. Arkin and R. Hassin. Minimum-diameter covering problems. *Networks*, 36(3):147–155, 2000.

[2] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, New York, NY, USA, 2006. ACM.

[3] A. Baykasoglu, T. Dereli, and S. Das. Project team selection using fuzzy optimization approach. *Cybern. Syst.*, 38(2):155–185, 2007.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge UP, 2004.

[5] M. Cheatham and K. Cleereman. Application of social network analysis to collaborative team formation. In *CTS '06: Proceedings of the International Symposium on Collaborative Technologies and Systems*, pages 306–311.

Table 2: Authors of the top-10 most cited papers from the **DBLP** dataset; **column 1**: paper ranking in the top-10 list; **column 2**: the actual authors of the papers; **column 3**: authors suggested by the `RarestFirst` algorithm; **column 4**: authors suggested by the `EnhancedSteiner` algorithm

| Rank | Actual authors | `RarestFirst` result | `EnhancedSteiner` result |
|---|---|---|---|
| **1** | S. Brin, L. Page | **Paolo Ferragina, Patrick Valduriez, H. V. Jagadish, Alon Y. Levy, Daniela Florescu** Divesh Srivastava, S. Muthukrishnan | **P. Ferragina ,J. Han, H. V. Jagadish, Kevin Chen-Chuan Chang, A. Gulli**, S. Muthukrishnan, Laks V. S. Lakshmanan |
| **2** | R. Agrawal, R. Srikant | **R. Agrawal** | **Philip S. Yu** |
| **3** | R. Agrawal, T. Imielinski, A. N. Swami | **Philip S. Yu** | **Wei Wang, Philip S. Yu** |
| **4** | T. Joachims | **Wei-Ying Ma, Gui-Rong Xue, H. Liu, J. Han, H. Lu, Z. Chen**, Q.Yang, H. Cheng | **J. Han, H. Lu, Wei-Ying Ma, Z. Chen, H. Liu, Gui-Rong Xue**, Q. Yang |
| **5** | J. Lafferty, F. Pereira, A. McCallum | **A. McCallum** | **A. McCallum** |
| **6** | J. Han, J. Pei, Y. Yin | **F. Bonchi** | **A. Gionis, H. Mannila, R. Motwani** |
| **7** | E. Rahm, P. A. Bernstein | **C. Bettini, R. Agrawal, Kevin Chen-Chuan Chang**, T. Imielinski, H. Garcia-Molina, D. Barbara, S. Jajodia | **C. Bettini, P. A. Bernstein**, H. Garcia-Molina, S. Jajodia, D. Maier, D. Barbara |
| **8** | R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan | **D. Gunopulos, R. Agrawal** | **R. Agrawal, D. Gunopulos** |
| **9** | B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom | **M. T. Ozsu** | **H. V. Jagadish, D. Srivastava** |
| **10** | J. Chen, D. J. DeWitt, F. Tian, Y. Wang | **Donald Kossmann, David J. DeWitt, Michael J. Franklin, Michael J. Carey** | **M. J. Carey, M. J. Franklin, D. Kossmann, D. J. DeWitt** |

[6] C. Chekuri, G. Even, and G. Kortsarz. A greedy approximation algorithm for the group steiner problem. *Discrete Appl. Math.*, 154(1):15–34, 2006.

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

[8] C. W. Duin, A. Volgenant, and S. Vo[ss]. Solving group Steiner problems as Steiner problems. *European Journal of Operational Research*, 154(1):323–329, 2004.

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

[10] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

[11] M. Gaston, J. Simmons, and M. desJardins. Adapting network structures for efficient team formation. In *In Proceedings of the AAAI Fall Symposium on Artificial Multi-agent Learning*, 2004.

[12] M. Jackson. Network formation. *The New Palgrave Dictionary of Economics and the Law*, 2008.

[13] G. Reich and P. Widmayer. Beyond steiner's problem: a VLSI oriented generalization. In *Proceedings of the fifteenth international workshop on Graph-theoretic concepts in computer science*, pages 196–210, 1990.

[14] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonica*, 24:573–577, 1980.

[15] V. Vazirani. *Approximation Algorithms*. Springer, 2003.

[16] H. Wi, S. Oh, J. Mun, and M. Jung. A team formation model based on knowledge and collaboration. *Expert Syst. Appl.*, 36(5):9121–9134, 2009.

[17] A. Zzkarian and A. Kusiak. Forming teams: an analytical approach. *IIE Transactions*, 31:85–97, 2004.