Online Team Formation in Social Networks*

Aris Anagnostopoulos[1]
aris@dis.uniroma1.it

Luca Becchetti[1]
becchett@dis.uniroma1.it

Carlos Castillo[2]
chato@acm.org

Aristides Gionis[2]
gionis@yahoo-inc.com

Stefano Leonardi[1]
leon@dis.uniroma1.it

[1]Sapienza University of Rome, Italy

[2]Yahoo! Research Barcelona, Spain

ABSTRACT

We study the problem of online team formation. We consider a setting in which people possess different skills and compatibility among potential team members is modeled by a social network. A sequence of tasks arrives in an online fashion, and each task requires a specific set of skills. The goal is to form a new team upon arrival of each task, so that (i) each team possesses all skills required by the task, (ii) each team has small communication overhead, and (iii) the workload of performing the tasks is balanced among people in the fairest possible way.

We propose efficient algorithms that address all these requirements: our algorithms form teams that always satisfy the required skills, provide approximation guarantees with respect to team communication overhead, and they are online-competitive with respect to load balancing. Experiments performed on collaboration networks among film actors and scientists, confirm that our algorithms are successful at balancing these conflicting requirements.

This is the first paper that simultaneously addresses all these aspects. Previous work has either focused on minimizing coordination for a single task or balancing the workload neglecting coordination costs.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – *Sequencing and scheduling*

General Terms: Algorithms

Keywords: Team formation, Task assignment, Scheduling

1. INTRODUCTION

We consider the problem of forming teams on a networked community of people with diverse skill sets. The goal is to assemble teams to deal with an incoming stream of tasks that are not known in advance. Each task requires a set of skills that must be covered by members of the assembled team. Each team should have small coordination cost, that is, it should be well-connected according to the underlying social network. Finally, this allocation should be fair, in the sense that no one should be overloaded with tasks or unfairly singled out.

In previous works (i) Lappas et al. [14] considered the team-formation problem for a single task with the objective of minimizing the coordination cost, but ignored the issue of balancing the workload, and (ii) Anagnostopoulos et al. [2] considered the formation of teams with the objective of balancing the workload, but ignored the coordination costs. According to our observations (detailed in Section 6) focusing exclusively on one of these aspects can lead to an unfair allocation of work or poorly connected teams.

Contributions. In this paper we propose the first online algorithms that assemble teams to deal with tasks, in a way that keeps coordination costs bounded and results in a fair allocation of the workload. Specifically:

– we introduce the Balanced Social Task Assignment problem and consider several variants, that include the problems studied by Lappas et al. [14] and Anagnostopoulos et al. [2] as special cases;

– we provide online competitive algorithms [21] with provable performance guarantees for these variants, noting that all of them are **NP**-hard;

– we demonstrate the effectiveness of our approach on two real-world datasets; compared with solutions that ignore user workloads, our algorithms achieve a significant decrease in the unbalance of the workload (60%–70%) with a small increase on the coordination overhead (5%–10%).

Roadmap. The next section discusses related work. Section 3 describes the general framework we use. Sections 4 and 5 describe our methods, which are tested experimentally in Section 6. The last section presents our conclusions.

2. RELATED WORK

Scheduling with load balancing: A well-studied problem since the work of Graham [12], is the scheduling of jobs on a set of machines with the goal of minimizing the maximum load on a machine. The setting has been extended to the restricted-assignment problem, in which the goal of balancing the workload is kept but additionally each job can only be processed by a subset of the available machines. The latter problem is **NP**-hard and has competitive ratio $O(\ln k)$ (where k is the number of jobs), shown to be asymptotically tight even for randomized online algorithms [4,5]. Our work deals with a problem harder than restricted assignment, since forming a team entails solving a set-cover prob-

*The research leading to these results has received funding from the EU FP7 Project N. 255403 SNAPS; the Spanish Ministry of Science and Innovation (project CEN-20101037 Social Media and Torres Quevedo programme); and the PRIN 2008 research projects COGENT.

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

lem, which has additional logarithmic lower-bound factors even in the offline case [22].

Matching people to tasks: Refereed conferences and journals require submissions to be reviewed by members of a program committee. This is a matching problem for which several systems have recently been proposed, for instance Easychair.org, Linklings.com and SoftConf.com. Easychair is currently based on a preliminary bidding process where reviewers rank papers into three classes. The problem of providing efficient solutions in the bidding model has recently been addressed by Mehlhorn [17]. Our work does not use explicit bids and is based on solving a *covering* problem instead of a *matching* problem. We also have the additional requirement of optimizing the coordination cost of formed teams.

Agent-organized team formation: Gaston et al. [11] study the team-formation problem in a setting where networked agents form teams in a decentralized manner. Agents are able to locally rewire their social network. We consider a setting where the social network is given, and also load balancing is taken into account.

Team formation with coordination costs: Lappas et al. [14] introduce the problem of team formation in social networks. The objective is to minimize the coordination cost, for example, in terms of diameter or weight of the minimum spanning tree for the team. This problem has been extended to cases in which potentially more than one member possessing each skill is required, and where density-based measures are used as objectives [9,15]. It has also been extended to allow partial coverage of the required skills, introducing a multi-objective optimization problem that is optimized using simulated annealing [8]. More recently, Kargar and An [13], consider a variation with a different cost model: when a user who participates in a task contributes with a variety of his skills (more on this in Section 3), the contribution to the cost is independent for each skill. Apart from modeling different scenarios with respect to the rest of previous (and our) work, technically this model simplifies the problem as it avoids the "set-covering" aspect of it.

Team formation with load balancing: In our previous work [2] we studied offline and online versions of the problem of allocating teams to tasks with the objective of balancing workload, but ignoring coordination costs.

3. PROBLEM DEFINITION

In this section, we present our framework and notation and formally define the problem that we study. Our notation is also summarized in Table 1.

3.1 Tasks, Skills, People and Teams

Tasks. We denote by $\mathcal{J} = \{\mathbf{J}^j; j = 1, 2, \ldots, k\}$ a set of *tasks* (or *jobs*). In the online setting that we consider, the jth task arrives at the jth time step and is assigned to a team of experts before the arrival of the $(j+1)$th task.

Skills. Each task requires a subset of the m available skills to be completed. We use the notion of *skill space* $\mathcal{S} = \{0,1\}^m$, which is the space that contains the possible ways that m skills combine to form a task, so, each task is a point in the skill space: $\mathbf{J}^j \in \mathcal{S}$. We use $\mathbf{J}^j_i = 1$ to denote that ith skill is required by the jth task, while $\mathbf{J}^j_i = 0$ otherwise. Thus, we have $\mathbf{J}^j = (\mathbf{J}^j_1, \mathbf{J}^j_2, \ldots, \mathbf{J}^j_m)$.

Table 1: Notation

$\mathcal{P}$	Set of people	n	Number of people
$\mathcal{S}$	Set of skills	m	Number of skills
$\mathcal{J}$	Set of tasks	k	Number of tasks
$\mathbf{p}^j$	jth person		
$\mathbf{J}^j$	jth task		
$d(\mathbf{p}^j, \mathbf{p}^{j'})$	Distance between persons j and j'		
$d(j, j')$	$d(\mathbf{p}^j, \mathbf{p}^{j'})$		
$\mathbf{p}^j_i$	ith skill of jth person		
$\mathbf{J}^j_i$	ith skill of jth task		
Q^j	Team assigned to jth task		
$\mathbf{q}^j$	Skill profile of team j		
$\mathbf{q}^j_i$	ith skill of profile of team j		
$\mathbf{L}(\mathbf{p})$	Load of person $\mathbf{p}$		
$\mathbf{L}(Q)$	Load of team Q		
$c(Q)$	Coordination cost of team Q		
$c(T)$	Coordination cost of subgraph T		
$a(Q)$	Allocation cost of team Q		
$a(\mathbf{p}^j)$	Allocation cost of person j		

People. We consider a set of *people* (or *experts*) $\mathcal{P} = \{\mathbf{p}^j; j = 1, 2, \ldots, n\}$. Each expert has a subset of skills in her profile, so she is also represented by a point in the skill space: $\mathbf{p}^j \in \mathcal{S}$. We use $\mathbf{p}^j_i = 1$ to denote that the jth person has the ith skill, while $\mathbf{p}^j_i = 0$ otherwise. Thus, we have $\mathbf{p}^j = (\mathbf{p}^j_1, \mathbf{p}^j_2, \ldots, \mathbf{p}^j_m)$.

Teams. Each task needs to be assigned to a *team* of experts. We let $Q^j \subseteq \mathcal{P}$ denote the team assigned to the jth task. We use $\mathbf{q}^j_i = 1$ to denote that the ith skill is covered by the jth team, so we have $\mathbf{q}^j = (\mathbf{q}^j_1, \mathbf{q}^j_2, \ldots, \mathbf{q}^j_m)$. For each team Q^j we compute its *team profile* $\mathbf{q}^j \in \mathcal{S}$ in the *additive skill model* [2] that defines the expertise of the team as the (binary) sum of the skills of each individual:

$$\mathbf{q}^j_i = \min\{ \sum_{\mathbf{p}^\ell \in Q^j} \mathbf{p}^\ell_i, 1\}_{i=1,\ldots,m}.$$

In other words, a skill is covered by the team if there exists at least one member who has that skill. We refer to a team for task $\mathbf{J}^j$ by using the notation $\mathbf{q}^j$ and Q^j interchangeably.

Given a team Q with profile $\mathbf{q}$ assigned to task $\mathbf{J}$, we define $\mathbf{cov}(\mathbf{q}, \mathbf{J}) \in \{0, 1\}$ to be 1 if and only if $\mathbf{q}^j_i \geq \mathbf{J}^j_i, \forall i = 1, \ldots, m$ and we say that the team Q *covers* the task $\mathbf{J}$. Note that this happens if every skill required by the task is possessed by the team.

3.2 Optimization Goals

We aim to form teams that can accomplish the specified tasks while optimizing two, possibly conflicting, goals: achieving a fair allocation of the overall workload among people and forming teams that have the lowest possible coordination cost. We now make these notions more precise.

Load. The first quantity we want to optimize is the *load* $L(\mathbf{p})$ of an expert $\mathbf{p}$, which we define to be the number of tasks in which $\mathbf{p}$ participates: $L(\mathbf{p}) = \big|\{j; \mathbf{p} \in Q^j\}\big|$. In particular, as in our previous work [2], we are interested in minimizing the maximum load over all the experts.

Coordination cost. In practice, team work faces coordination costs. Such costs limit the size of organizations, as formalized in the seminal work of Coase [7]. In our setting, the coordination overhead in collaboration may nullify the potential advantages of a larger expertise coverage.

We model coordination costs by means of a social network over the set of people, $G = (\mathcal{P}, E)$, and assume a metric distance function $\mathbf{d} : E \to \mathbb{R}^+$ on the edges of the network. This function may model preferences based on past interactions, geographical proximity, compatibility in collaborating, distance in a company's hierarchy, and so on. We extend the definition of the distance function $\mathbf{d}(\cdot, \cdot)$ to any pair of people in the network, by considering the sum of the distances along a shortest path connecting the two people.

There are many ways of defining the coordination cost of a team as a function of the pairwise distances among the people in the social network. Before describing specific coordination-cost measures, we first consider two natural options with respect to defining the "underlying network" of the team.

Implicitly Connected Teams (ICT). In this model we do not require each team Q to form a connected graph; we only ask for communication paths between people in the team through other members of the social network (not necessarily present in Q). This model is realistic if the existence of a short chain of acquaintances is enough to declare people as "compatible."

Explicitly Connected Teams (ECT). This follows the approach of Lappas et al. [14] and requires that each team Q forms a connected graph using the set of edges $T = \{(\mathbf{p}^i, \mathbf{p}^j) : \mathbf{p}^i, \mathbf{p}^j \in Q\}$. We also denote by $\mathbf{d}$ the distances computed in subgraph (Q, T).

The difference between the ICT and ECT cases is the subgraph over which the coordination cost is computed. In the ICT case, it is computed over the whole social network; in the ECT case, over the subgraph induced by the team. In both cases we use the following measures of coordination cost, which are the ones considered by Lappas et al. [14]

1. Steiner tree: $\mathbf{Steiner}(Q)$ is the cost of the min cost Steiner tree T that has as terminal nodes the team members.
2. Diameter: $\mathbf{Diam}(Q) = \max_{\mathbf{p}^i, \mathbf{p}^{i'} \in Q} \mathbf{d}^j(i, i')$

In general, we denote by $c(Q)$ the coordination cost of team Q, which we also write as $c(T)$ if the set of edges T connecting team members is given. We have also studied the sum-of-distances cost,[1] however due to lack of space we omit the results from this version.

3.3 Problem Statement

Assembling teams while accounting for two objective functions, coordination cost and load, leads to a bi-criteria objective, which in general is hard to analyze, especially if we seek for theoretical guarantees. A principled approach that is usually adopted in the literature [19, 20] to address bi-criteria optimization problems is to bound the cost on one objective function and optimizing on the other. This allows to turn the bi-criteria optimization problem into an optimization problem for a single objective function. In our case, we put a bound B on the coordination cost for each job $\mathbf{J}^j$ and we optimize the maximum load of a person, with the interpretation that B is a bound on the coordination cost that we are willing to accept for the particular application.

In this spirit, we consider the following generic optimization problem, which we call Balanced Social Task Assignment. It consists in minimizing the maximum load of a person while assigning each job to a team that covers all the skills of the job, with a bounded coordination cost:

$$\begin{aligned} &\min \max_{i \in \mathcal{P}} L(\mathbf{p}^i) \\ &\mathbf{cov}(\mathbf{J}^j, \mathbf{q}^j) = 1 \qquad \forall j \in \mathcal{J} \\ &c(Q^j) \le B \qquad \forall j \in \mathcal{J}. \end{aligned}$$

In the next section we propose algorithms with theoretical guarantees for this problem under different coordination cost functions in the online setting.

Another option would be to attempt to bound the maximum load and minimize the coordination cost. Note though that such an attempt is more unnatural in our online setting. Since we do not know a priori the number of tasks it is hard to set a bound on the total maximum load. Instead it is more sensible to preset an acceptable level of coordination and attempt to minimize the load as new tasks keep arriving. However, despite the choice to set up a fixed bound B on the coordination cost for analysis reasons, in Sections 5 and 6 we observe that by weighing differently the load and the coordination costs we can tradeoff between those two objectives.

Table 2: Notation for analysis

(Q^*, T^*)	Optimal solution for the Social Task Assignment problem
$(Q^{\mathtt{opt}}, T^{\mathtt{opt}}) = (Q^{\mathtt{opt}}_\lambda, T^{\mathtt{opt}}_\lambda)$	Optimal solution for modified problem instance with ratio λ
$f^{\mathtt{opt}}(\lambda) = \lambda a(Q^{\mathtt{opt}}) + c(T^{\mathtt{opt}})$	Optimal cost for modified problem instance with ratio λ
$(Q^{\mathtt{apx}}, T^{\mathtt{apx}}) = (Q^{\mathtt{apx}}_\lambda, T^{\mathtt{apx}}_\lambda)$	γ-approximate solution for modified problem instance with ratio λ
$f^{\mathtt{apx}}(\lambda) = \lambda a(Q^{\mathtt{apx}}) + c(T^{\mathtt{apx}})$	γ-approximate cost for modified problem instance with ratio λ
$\lambda^* = c(T^*)/a(Q^*)$	

4. ALGORITHMS

In this section, we introduce and analyze the performance of a general online algorithm for Balanced Social Task Assignment that can be applied to a variety of coordination cost functions. For easy reference, we summarize our notation in Table 2.

4.1 The General Online Algorithm

Our online algorithm works as follows: upon arrival of a new task $\mathbf{J}$, a team is formed for task $\mathbf{J}$ by solving a suitably-defined instance of the (offline) Social Task Assignment problem. The Social Task Assignment subproblem for task $\mathbf{J}$ consists in selecting the team Q that minimizes a specific *cost allocation function* $a(Q)$ subject to a constraint on the coordination cost $c(Q)$. The problem for task $\mathbf{J}$ is as follows:

$$\begin{aligned} &\min_Q a(Q) \\ &\mathbf{cov}(\mathbf{J}, \mathbf{q}) = 1 \\ &c(Q) \le B. \end{aligned}$$

The Social Task Assignment problem depends on the specific coordination cost $c(Q)$ and allocation cost $a(Q)$ that we adopt. As we mentioned in the previous section for the coordination cost we consider two measures for Q: $\mathbf{Steiner}(Q)$ and $\mathbf{Diam}(Q)$.

[1] $\mathbf{Sum}(Q) = \sum_{\mathbf{p}^i, \mathbf{p}^{i'} \in Q} \mathbf{d}^j(i, i')$

Allocation cost function. The allocation function $a(\cdot)$ used in the SOCIAL TASK ASSIGNMENT instance depends on the individual loads on the persons, as they have resulted from the previous allocation of the first $j-1$ tasks. Different choices of the allocation cost function affect the performance of the overall online algorithm. There exist several natural allocation cost functions one can consider, such as minimizing the team size, the maximum load, or the total load; in our previous work [2] we analyze such functions explicitly, and we have shown that their performance, in terms of balancing workload, is not as good as the `ExpLoad` allocation cost function that we describe below. The `ExpLoad` function is motivated by research in online scheduling, and is proven to be effective when the coordination cost is not taken into account [2]. Here we extend the results by showing that `ExpLoad` is an effective allocation cost function also when coordination cost is taken into account.

Let us denote by $L(\mathbf{p}^i)$ the load of person i when task $\mathbf{J}$ is presented, and let Λ be an appropriately chosen value that depends on the cost of the optimal solution. Then, the `ExpLoad` function is defined as follows:

$$a(Q) = \sum_{\mathbf{p}^i \in Q} (2n)^{\frac{L(\mathbf{p}^i)}{\Lambda}}.$$

Details on how to adaptively estimate Λ during the execution of the algorithm can be found in our previous work [2]. Using `ExpLoad` guarantees effective algorithms both in theory and in practice.

4.2 Competitive Analysis

For evaluating the performance of online algorithms we resort to the notion of competitive analysis [21]. Competitive analysis compares an online algorithm against an offline adversary who knows the entire input sequence in advance and serves it optimally. An online algorithm is c-competitive if for any input sequence it provides a solution with cost bounded by c times the optimal cost. Observe that even an online algorithm that optimally solves the SOCIAL TASK ASSIGNMENT problem for each task may not provide an optimal solution for the overall problem, given the lack of information about future tasks. The following result, proven similarly to [3], shows that using the `ExpLoad` allocation cost function guarantees the optimal asymptotic performance possible. The full proof will be included in an extended version of this work.

THEOREM 1. *Consider the* BALANCED SOCIAL TASK ASSIGNMENT *problem and assume the corresponding* SOCIAL TASK ASSIGNMENT *problem using the* `ExpLoad` *allocation cost function can be solved optimally. Then this algorithm is* $\Theta(\ln k)$*-competitive. In addition this is the best competitive ratio possible.*

However, the SOCIAL TASK ASSIGNMENT problem can not be solved optimally efficiently unless $\mathbf{P} = \mathbf{NP}$, since it entails the set-cover and the Steiner-tree problems.

Bi-criteria approximation guarantee. Every algorithm A for SOCIAL TASK ASSIGNMENT returns for a task $\mathbf{J}$ a pair (Q^A, T^A), where T^A is a subset of edges used to connect the members of team Q^A and $c(T^A)$ is the coordination cost. Let $(Q^{\mathtt{opt}}, T^{\mathtt{opt}})$ be the optimal solution to the problem of minimizing $a(Q)$ subject to coordination cost being bounded by B. The performance of the algorithm on a single instance of SOCIAL TASK ASSIGNMENT will be measured through a bi-criteria approximation guarantee:

DEFINITION 1. *An* (α, β) *bi-criteria approximation algorithm for the* SOCIAL TASK ASSIGNMENT *problem for a task* $\mathbf{J}$ *will return in polynomial time a pair* (Q, T) *such that: (i)* $a(Q) \leq \alpha \cdot a(Q^{opt})$ *and (ii)* $c(T) \leq \beta B$.

The values of the approximation factors α and β depend on the specific coordination cost $c(T)$ that we consider. Next we show how a bi-criteria algorithm for the SOCIAL TASK ASSIGNMENT problem, together with the `ExpLoad` cost allocation function, can yield an online algorithm for the BALANCED SOCIAL TASK ASSIGNMENT problem. The proof is similar to the proof of Theorem 3 in [2] and will be included in an extended version of this paper.

THEOREM 2. *Assume that there exists an* (α, β) *bi-criteria approximation algorithm for the* SOCIAL TASK ASSIGNMENT *problem with the* `ExpLoad` *allocation cost function and with a coordination cost function* $c(T)$*. Then there exists a polynomial time algorithm for the* BALANCED SOCIAL TASK ASSIGNMENT *problem that is* $\Theta(\alpha \ln k)$*-competitive and for each task* $\mathbf{J}$ *attains a coordination cost* $c(T) \leq \beta B$.

Remarks. Theorem 2 shows that the competitive ratio of the overall online algorithm depends on how closely we approximate the optimum of the SOCIAL TASK ASSIGNMENT instances. In the remainder of this section we provide algorithms with provable performance for the SOCIAL TASK ASSIGNMENT problem under the **Steiner** and **Diam** coordination cost functions in the ICT model. Heuristics to solve this problem (and, thus, the overall scheduling problem) in the ECT model are described in Section 5.

4.3 The Steiner-Tree Coordination Cost

In this section we describe an approach that guarantees a solution for the BALANCED SOCIAL TASK ASSIGNMENT problem with an optimal competitive ratio in the case of **Steiner** cost and implicitly connected teams. Given Theorem 2, we need to provide a solution to the following SOCIAL TASK ASSIGNMENT problem:

$$\begin{aligned} \min_{Q} \sum_{\mathbf{p}^i \in Q} a(\mathbf{p}^i) \\ \mathbf{cov}(\mathbf{J}, \mathbf{q}) = 1 \\ c(Q) \leq B, \end{aligned}$$

where we define $a(\mathbf{p}^i) = (2n)^{\frac{L(\mathbf{p}^i)}{\Lambda}}$, so that $a(Q) = \sum_{\mathbf{p}^i \in Q} a(\mathbf{p}^i)$. The constraint $c(Q) \leq B$ specifies that the Steiner-tree cost of the nodes in the selected team Q has to be at most B. This constraint, along with the coverage constraint $\mathbf{cov}(\mathbf{J}, \mathbf{q}) = 1$, define a *group Steiner tree* problem [6, 10]. Note that the connection to the group Steiner tree was also considered by Lappas et al. [14] for the algorithms they developed.

In the group Steiner tree problem, we are given a network $G = (V, E)$, where $|V| = n$, with a distance function $d : E \to \mathbb{R}^+$, and a family $\mathcal{G} = \{g_1, \ldots, g_\ell\}$ of *groups* of nodes, with $g_i \subset V$. The objective is to find a minimum-cost subtree T of G that contains at least one node from each group g_i. This problem has a $\gamma = O(\ln^2 n \ln \ln n \ln \ell)$ approximation [6].

In fact, our problem is more complicated as we need to find an algorithm A that returns a pair (Q^A, T^A) that not

Algorithm 1 The **Steiner** coordination-cost algorithm.

1. **Function** `findGroup`($\mathbf{J}$)
2. /* *Find the best group for task* $\mathbf{J}$. *For upperBound see the text.* */
3. $\lambda_A \leftarrow$ `binarySearch`($\mathbf{J}, 0, upperBound$)
4. /* *Estimate the threshold; we find the largest value below the threshold with an accuracy of* ϵ. */
5. $(Q^A, T^A) \leftarrow$ `findGroupForGivenLambda`($\mathbf{J}, \lambda_A$)
6. **return** (Q^A, T^A)

1. **Function** `binarySearch`($\mathbf{J}, \lambda_1, \lambda_2$)
2. /* *Return the highest value between* λ_1 *and* λ_2 *for which* $c(Q)$ *is below the threshold, with an accuracy gap of* ϵ. */
3. **if** $(\lambda_2 - \lambda_1 < \epsilon \cdot \lambda_2)$
4. /* *The two endpoint are very close; value found* */
5. **return** λ_1
6. **end if**
7. $\lambda \leftarrow (\lambda_1 + \lambda_2)/2$
8. $(Q_\lambda^{\mathtt{apx}}, T_\lambda^{\mathtt{apx}}) \leftarrow$ `findGroupForGivenLambda`($\mathbf{J}, \lambda$)
9. **if** $(c(Q_\lambda^{\mathtt{apx}}) < 2\gamma B)$
10. **return** `binarySearch`($\mathbf{J}, \lambda, \lambda_2$)
11. **else**
12. **return** `binarySearch`($\mathbf{J}, \lambda_1, \lambda$)
13. **end if**

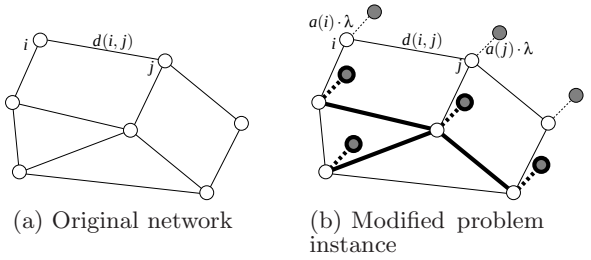
1. **Function** `findGroupForGivenLambda`($\mathbf{J}, \lambda$)
2. /* *Find the best group on the modified graph for the given value of* λ */
3. $G' \leftarrow$ `createModifiedGraph`(λ)
4. $(Q_\lambda, T_\lambda) \leftarrow$ `solveGroupSteinerTree`($G', \mathbf{J}$)
5. **return** (Q_λ, T_λ)

1. **Function** `createModifiedGraph`(λ)
2. /* *Return the modified graph instance using the value of* λ. */
3. $V' \leftarrow$ A copy of V
4. $E' = \{e_i = (i, i'); i \in V, i' \in V', w(e_i) = c(i) \cdot \lambda\}$
5. $i' \in V'$ belongs to group j iff user i possess skill j
6. $G' = (V \cup V, E \cup E', \mathbf{w})$
7. **return** G'

only satisfies the group Steiner tree constraints, but that also minimizes the cost allocation function $a(Q)$ over the nodes returned. Let (Q^*, T^*) be the optimal solution to our Social Task Assignment problem.

The description of the algorithm `findGroup`($\mathbf{J}$), which solves the Social Task Assignment problem for task $\mathbf{J}$, is in Algorithm 1. Our high-level strategy to solve the Social Task Assignment problem is the following: (i) First we define a family of problem instances, which we call the *modified problem instances*, parametrized by a value $\lambda \in \mathbb{R}^+$. Each of the modified problem instances will be an instance of the group Steiner tree problem. We can also show that there is a value $\lambda = \lambda^*$ such that the solution to the modified problem instance is an approximate solution to the original Social Task Assignment problem. (ii) Next we show that although it is hard to compute the value λ^* exactly, we are able to compute efficiently an approximate value that yields an approximate solution to the original problem. We now proceed with the details.

(i) Build modified problem instance. For a given value of parameter λ we construct the following modified problem instance (see Figure 1). If the initial graph is $G = (V = \mathcal{P}, E)$ we create a new graph $G' = (V \cup V', E \cup E')$ with a new fictitious node $i' \in V'$ for each node $i \in V$, and with a new edge $e_i = (i, i') \in E'$ of cost $\lambda a(i)$, for each node



(a) Original network (b) Modified problem instance

Figure 1: The original network on which we want to solve the Social Task Assignment problem and the modified problem instance on which we solve the group Steiner tree problem. With bold we can see an example solution (Q, T). Notice that the cost of the solution is the coordination cost $c(T)$ (of the original network) and the cost of the new edges, which is equal to the cost allocation $a(Q)$.

$i \in V$. We then consider the group Steiner tree problem on this modified instance, in which each skill required by the arrived task corresponds to a group and a node i' belongs to the rth group if the corresponding expert i possesses the rth skill. A solution to the group Steiner tree on the modified problem instance is a pair (Q, T) where Q is the set of nodes selected that cover all the groups (skills) and T a subtree that connects the nodes in Q. Note that the cost of the solution (Q, T) on the modified instance is given by

$$f(Q, T, \lambda) = \lambda a(Q) + c(T).$$

For a given algorithm A, let us define the cost of the solution (Q^A, T^A) that it outputs as

$$f^A(\lambda) = \lambda a(Q^A) + c(T^A),$$

with the understanding that if A is a randomized algorithm then $f^A(\lambda)$ is the expected cost of its solution. Let also $\mathtt{opt} = \mathtt{opt}_\lambda$ be the optimal algorithm for the given modified problem instance so that $f^{\mathtt{opt}}(\lambda)$ is the cost of its optimal solution $(Q^{\mathtt{opt}}, T^{\mathtt{opt}})$. Similarly, for a γ-approximation algorithm $\mathtt{apx} = \mathtt{apx}_\lambda$, we have that $f^{\mathtt{apx}}(\lambda)$ is the cost of the solution $(Q^{\mathtt{apx}}, T^{\mathtt{apx}})$ produced by `apx`. When we want to explicitly specify the value of λ for which the solution is optimal or approximate we will write $(Q_\lambda^{\mathtt{opt}}, T_\lambda^{\mathtt{opt}})$ and $(Q_\lambda^{\mathtt{apx}}, T_\lambda^{\mathtt{apx}})$.

Recall that (Q^*, T^*) is the optimal solution to the Social Task Assignment problem. Let us define

$$\lambda^* = \frac{c(T^*)}{a(Q^*)}.$$

It turns out that the value λ^* is the value of λ for which we would like to solve our problem, as the solution to the modified problem for $\lambda = \lambda^*$ provides an approximation solution to the STA problem, as one can see in the following proposition, whose proof is straightforward and omitted for the sake of space.

Proposition 3. *The following are true.*

1. $c(T_{\lambda^*}^{opt}) \leq 2B$.
2. $a(Q_{\lambda^*}^{opt}) \leq 2a(Q^*)$.

(ii) Estimating λ^*. Until now we have shown that if we know the value λ^* then we can define the appropriate modified problem instance to solve, which in turn can give us an approximate solution to the original SOCIAL TASK ASSIGNMENT problem. The second claim that we need to show is that we can find the value λ^* efficiently. Actually, since we cannot solve the group Steiner tree on the modified problem instances exactly, we cannot compute the value λ^* exactly. However, we can estimate it and we will show that our estimate will provide a bi-criteria approximate solution to the original problem. To estimate the value λ^* we exploit the fact that the function $f^{\text{opt}}(\lambda)$ is a monotone function of λ, as we show in the next key lemma (whose proof we defer to an extended version of this paper). This will allow us to estimate λ^* using a binary search. Also it will allow us to show that having only an estimate of λ^* can provide a bi-criteria approximation.

LEMMA 4. *The following hold:*

1. *The function $f^{opt}(\lambda)$ is nondecreasing in λ.*
2. *Assume that $\lambda_1 < \lambda_2$. Then we have that $a(Q^{opt}_{\lambda_1}) \geq a(Q^{opt}_{\lambda_2})$ and $c(T^{opt}_{\lambda_1}) \leq c(T^{opt}_{\lambda_2})$.*

In essence, this lemma allows us to estimate λ^* with the required accuracy, using a binary search, each iteration of which involves solving an instance of the modified problem for some value λ. The exact description is in Algorithm 1. Note that our technique is an example of the parametrized-search approach for the solution of constrained optimization problems, originally proposed by Megiddo [16] and used in subsequent work (see for example [19] for its application to bi-criteria spanning tree problems). Care needs to be taken because we cannot solve each problem instance optimally when performing the binary search, but only approximately.

Recall that $(Q^{\text{opt}}_{\lambda}, T^{\text{opt}}_{\lambda})$ is the optimal solution of the modified problem instance for parameter λ (i.e., the value that minimizes $f(Q, T, \lambda)$). To obtain an understanding of our algorithm, it is easier to describe a version where we could solve optimally the modified problem instance for any value of λ. Then we estimate the value λ^* by performing a binary search. In every step of the binary search we pick the value of λ that is the average of the two endpoints λ_1 and λ_2 and we solve the modified problem instance. Throughout the entire execution we maintain the invariant that $c(T^{\text{opt}}_{\lambda_1}) < B$ and $c(T^{\text{opt}}_{\lambda_2}) \geq B$. We stop when λ_1 and λ_2 are close to each other, and the second claim of Lemma 4 guarantees that $\lambda_1 \leq \lambda^* \leq \lambda_2$.

Since we can only solve the modified problem with an approximation of γ, we cannot compute the value λ^*. Instead we modify the search to find a value that satisfies the constraint allowing for room because of the approximation ratio. For the sake of brevity, let A denote Algorithm `findGroup` so that the solution it returns is (Q^A, T^A), and let λ_A the estimate of λ^* computed by A. The above considerations are at the basis of the following theorem, whose proof is omitted for the sake of space. The result establishes that our algorithm is an $(O(\gamma), O(\gamma))$ bi-criteria approximation algorithm for the SOCIAL TASK ASSIGNMENT problem.

THEOREM 5. *For any task $\mathbf{J}$, Algorithm `findGroup`($\mathbf{J}$) returns a solution (Q^A, T^A) such that:*

1. *$c(T^A) \leq 2\gamma B$.*
2. *$a(Q^A) \leq \frac{2\gamma}{1-\epsilon} a(Q^*)$.*

Algorithm 2 The diameter algorithm.

```
1. Function findGroup(J)
2. /* Find the best group for task J */
3.    r ← arg min_{i∈S} Σ_{j∈P} p_i^j
4.    /* r now is (one of) the rarest skill */
5.    Q^A ← ∅
6.    foreach (ℓ such that p_r^ℓ = 1)
7.       U ← {p^j ; d(p^ℓ, p^j) ≤ B}
8.       Q ← greedyWeightedSetCover(U)
9.       if (a(Q) < a(Q^A))
10.         Q^A ← Q
11.      end if
12.   end foreach
13.   return Q^A
```

Combining Theorems 1, 2, and 5 it follows that there exists a bi-criteria approximation algorithm for the BALANCED SOCIAL TASK ASSIGNMENT problem (recall that n is the number of people, m the number of skills, and k the number of tasks):

THEOREM 6. *There exists an $O(\ln^2 n \ln\ln n \ln m \ln k)$-competitive algorithm for the* BALANCED SOCIAL TASK ASSIGNMENT *problem, which for each task attains a Steiner-tree coordination cost bounded by $O(B \ln^2 n \ln\ln n \ln m)$.*

Running time. It is easy to see that the complexity of the algorithm 1 is polynomial. This follows because (i) every invocation of `findGroupForGivenLambda` runs in polynomial time [6] and (ii) the function `binarySearch`) is invoked a polynomial number of times.

To see why (ii) holds, note that the parameter *upperBound* in line 3 of the algorithm is set to the largest possible value of the ratio $c(T)/a(Q)$. Note that $c(T) \leq \sum_{\mathbf{p}^j, \mathbf{p}^\ell \in \mathcal{P}} d(\mathbf{p}^j, \mathbf{p}^\ell)$. Moreover, a trivial lower bound to $a(Q)$ is 1 (the load on every person is 0 when the first task is allocated and the team size is 1). The claim then follows since `binarySearch` is invoked a number of times that is at most logarithmic in *upperBound*, for a constant ϵ.

Notice however that, although polynomial, the cost of the algorithm can be high in practice, due to the super-linear cost of the group Steiner tree algorithm [6]. For this reason, in Section 5, we consider computationally less demanding heuristics to implement `findGroupForGivenLambda`.

4.4 The Diameter Coordination Costs

For the diameter coordination cost (**Diam**) we follow a different, simpler approach, which we present in Algorithm 2. Similar to the previous coordination cost functions, we try to find a team of users Q with diameter at most γB (it turns out that for the diameter we can obtain $\gamma = 2$) that minimizes $a(Q)$. However, in this case we do not build a new problem instance to combine the load with the social cost function. Instead we solve the problem directly.

Our results are summarized in the following theorem, whose proof will be given in the full version of the paper.

THEOREM 7. *There exists an $O(\ln m \ln k)$-competitive algorithm for the* BALANCED SOCIAL TASK ASSIGNMENT *problem, which for each task attains a diameter coordination cost bounded by $2B$.*

Running time. The running time of the algorithm is obviously polynomial: the main loop (lines 6–12 of Algorithm 2)

is executed at most n times, while each execution requires a run of the (polynomial) greedy heuristics for weighted set cover.

4.5 The Sum-of-Distances Coordination Cost

For the sum-of-distances coordination cost (**Sum**) we can apply a similar technique, and define an appropriate modified problem instance. Due to lack of space we will include the details in an extended version of this work. We only mention that to solve it we define an appropriate class of modified problem instances that we solve by formulating them as a convex program and rounding it performing randomized rounding. This allows us to obtain results along the lines of Theorems 6 and 7.

5. IMPLEMENTATION AND HEURISTICS

In the previous section we developed algorithms that provably approximate the optimal solution in the online case and also run in polynomial time. These algorithms are specifically suited for the implicitly connected team (ICT) model. For the explicitly connected team (ECT) model, we note that the problem is as hard: for example, the explicitly connected version for the **Steiner** coordination cost is as hard as the node-weighted group Steiner tree problem that is strictly harder than the edge-weighted group Steiner tree problem. Moreover, it still relies on an approximation algorithm for the group Steiner tree problem.

In this section we address several issues regarding the practicality of our approach. In particular, to be able to formulate an optimization problem that addresses the multi-criteria objectives, we attempt to minimize the maximum load, while keeping the coordination cost bounded by a constant B (see Section 3.3). Second, to provide approximation guarantees we need to employ theoretical results for the group Steiner tree problem. For the **Diam** coordination cost this is not an issue, since the approach that we describe in Section 4.4 is efficient both in theory and in practice.

The theoretical algorithm used for the group Steiner tree problem [6], while it provides asymptotic guarantees, is not practical; it involves a probabilistic embedding of the graph to a tree, solving a linear program and appropriate rounding. Therefore, in our experiments we apply some simpler heuristics. One heuristic is the approach used by Lappas et al. [14], which is based on the creation of (yet) another graph instance, on which the Steiner tree problem is solved. Briefly, for each skill (group), we create a new node that is connected to all the experts that possess it with a large cost D. Then we compute the Steiner tree that connects the required skills. For the latter task we employ the minimum spanning-tree heuristic.

We also develop a second algorithm, which we call the *set-cover heuristic* and we present in detail in Algorithm 3. We will represent the allocation costs (appropriately scaled by λ) on the nodes of the social network and social costs on the edges of the social network. Since we want to produce solutions that are close to the optimum, we assume that our algorithm's solution shares at least one expert with the team of the optimal solution. This is obtained by trying as roots all the experts that possess the rarest skill of the the task (i.e., the skill owned by the least number of experts). Let Q and T be the team and Steiner tree constructed so far in the execution of the algorithm. In line 8, gain($\mathbf{p}^j$) is set to the number of required skills owned by expert $\mathbf{p}^j$ that

Algorithm 3 The set-cover **Steiner** algorithm.

```
1. Function findGroupForGivenLambda-setCover(J, λ)
2. /* Find the best group for task J using the set-cover
   heuristic. */
3.    Q ← ∅
4.    T ← ∅
5.    while (∃i s.t. (J_i − q_i)^+ > 0)
6.       mostCostEffective ← 0
7.       foreach (p^j ∈ P \ Q)
8.          gain(p^j) ← Σ_{i=1}^{m} 1_{{p_i^j=1 ∧ J_i=1 ∧ q_i=0}}
9.          loss(p^j) ← cost in G for adding p^j in Q
10.         costEffectiveness ← gain(p^j)/ loss(p^j)
11.         if (costEffectiveness > maxCostEffectiveness)
12.            maxCostEffectiveness ← costEffectiveness
13.            mostCostEffective ← j
14.         end if
15.      end foreach
16.      T ← T ∪ "shortest path from p^{mostCostEffective} to T"
17.      Q ← Q ∪ {p^{mostCostEffective}} /* This is for the ICT
          model. For the ECT model we add all the experts
          of the path. See the text for more details. */
18.   end while
19.   return (Q, T)
```

are not covered yet by Q. (The notation $\mathbf{1}_{\mathcal{A}}$ denotes the indicator function for $\mathcal{A}$.) To see why this is the case, note that expert $\mathbf{p}^j$ contributes to the ith skill if and only if he possesses the skill ($\mathbf{p}_i^j = 1$), the skill is required ($\mathbf{J}_i = 1$), but not yet covered ($\mathbf{q}_i = 0$). In line 9 we set loss($\mathbf{p}^j$) to the cost of adding expert $\mathbf{p}^j$ to team Q as we discuss in the next paragraph. Then, at each step of the algorithm we add to Q the most *cost-effective* expert, where cost-effectiveness is defined by the ratio $\frac{\text{gain}(\mathbf{p}^j)}{\text{loss}(\mathbf{p}^j)}$. The algorithm terminates when all skills of the task are covered.

The algorithms for the different notions of coordination cost and for the models ICT and ECT differ only in the definition of the costs loss($\mathbf{p}^j$). Denote by $\ell^s(\mathbf{p}^j, \mathbf{p}^{j'})$ the length of the min cost path $p^s(\mathbf{p}^j, \mathbf{p}^{j'})$ in social network N that connects $\mathbf{p}^j$ to $\mathbf{p}^{j'}$ plus the (appropriately scaled) allocation costs of all the people on the path, excluding $\mathbf{p}^{j'}$.

We define $\text{loss}(\mathbf{p}^j) = \min_{\mathbf{p}^{j'} \in Q} \ell^s(\mathbf{p}^j, \mathbf{p}^{j'})$. In the ICT model once we find the most cost-effective people $\mathbf{p}^j$, we add path $p^s(\mathbf{p}^j, \mathbf{p}^{j'})$ to T and expert $\mathbf{p}^j$ is included in team Q. In the ECT model, once we find the most cost-effective expert $\mathbf{p}^j$, we add path $p^s(\mathbf{p}^j, \mathbf{p}^{j'})$ to T and team Q is extended to include all experts on the path, to ensure the team resulting team is a connected subgraph. Therefore, in this case, $\ell^s(\mathbf{p}^j, \mathbf{p}^{j'})$ takes into account the cost of including these experts as well. We note that in the ECT model, experts on path $p^s(\mathbf{p}^j, \mathbf{p}^{j'})$ may contribute not only by connecting others but often provide additional skills for the task, that are considered covered by those experts given that they will be part of the generated team.

Now we address the first issue that we raised in the beginning of the section, which is that in our modeling we choose to optimize the cost allocation subject to an upper bound B in the coordination cost for the reasons explained in Section 3.3. While for the theoretical analysis we wrote a principled version of the problem, in many practical applications, the upper bound B on the coordination cost is neither explicitly given nor clearly defined. Another approach to bi-criteria optimization problems is to compute the set of

Table 3: Summary statistics of datasets.

Dataset	IMDB	Bibsonomy
n (people)	725	816
m (skills)	21	793
k (tasks)	1 000	2 000
Average skills per expert	2.96	7.64
Average skills per task	11.10	4.44
Average degree soc. net.	17.10	5.53
Power-law parameter α	0.99	1.60
Average edge distance	0.66	0.70
St. dev. of edge distance	0.11	0.28

Pareto optimal solutions [18], which are those solutions that are not outperformed by other solutions on both allocation and coordination costs. The set of all Pareto-optimal solutions usually form a convex curve that defines a clear trade-off between allocation and social cost. It is of course highly desirable that the Pareto curve is not far from the point in the space with coordinates equal to the minimum coordination cost and the minimum allocation cost of any solution. The two approaches are tightly related and algorithms for one problem can be used for solving the other problem. If the underlying optimization problems are **NP**-hard we can only hope to find in polynomial time Pareto-nearly-optimal solutions, which are solutions that are provably not outperformed from other solutions in all objectives for more than a limited amount [18].

The idea of the algorithms developed in Section 4 is that the right trade-off between allocation and coordination cost is given by the ratio λ^* between coordination and allocation costs in the optimal solution. The problem is therefore reduced to solving the Social Task Assignment problem with allocation cost scaled by λ^*. An algorithm A will return a solution (Q^A, T^A) that minimizes $f^A(\lambda) = \lambda a(Q^A) + c(T^A)$. Given that we do not assume in practice a bound on the coordination cost, we do not compute the profile of the Pareto solutions. Rather, we proceed with the optimization of $f^A(\lambda)$ for different scaling factors λ. This will allow us to identify the best trade-off between social and allocation cost. Nevertheless, note that to solve (approximately) the optimization problem as formulated in Section 3.3 it suffices to observe only the values of λ that are requested by the algorithms in Section 4.

6. EXPERIMENTS

Finding datasets that reflect the application scenarios of team formation is challenging, as such systems are not yet in wide-spread use, and most data from them is not publicly available. For this reason, we follow the approach of previous work [2,13,14], and we use bibliography and movie datasets that can demonstrate the effectiveness of our approach. Summary statistics from these datasets are included in Table 3. We expect that emerging crowdsourcing and labor marketplaces will provide in the future more realistic datasets for this line of research.

6.1 Datasets

`IMDB`. Our first dataset is extracted from the Internet Movie Database. We focus on two types of movie personnel, *directors* and *actors*. For skills we use the movie genres. We assume that the set of genres of the movies that a person has participated make the set of skills for that person. For example, *Alfred Hitchcock* has the skills {*comedy*, *crime*, *film-noir*, *mystery*, *romance*, *thriller*}. We run our algorithms so that directors represent experts and actors represent tasks, where a random subset of 1000 actors was sampled. Our setting simulates an imaginary scenario in which people who have directed a movie create small committees to audition actors. We do not consider actors who have only one skill, because those can be assigned to a single director, which is not an interesting case.

Next, we define a social graph among the directors. Since, typically movies are directed by only one director, we "connect" directors using actors as intermediaries: we form an edge between two directors if they have directed at least two distinct actors in common. The cost of the edge is set to e^{-rD}, where D is the number of distinct actors directed by the two directors. The distance function e^{-rD} takes values between 0 and 1, and it approaches very fast the value 0 as the number of commonly directed actors D between two directors increases. We select the value of parameter $r = \frac{1}{10}$, for which we experimentally verify that it yields an intuitive range of values for the distance values on the edge weights.

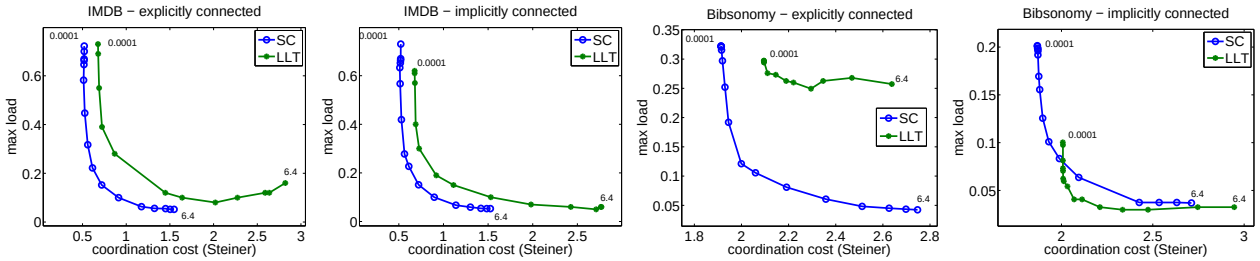
`Bibsonomy`. Our second dataset is extracted from bibsonomy [1], a social-bookmarking and publication-sharing system. The dataset contains a large number of computer-science related publications. Each publication is written by a set of *authors*. The bibsonomy website is visited by a large community of users who use *tags* to annotate the publications, for example, *theory*, *software*, or *ontology*. We use the set of tags associated with the papers of an author to represent the set of skills for that author. We partition the set of authors into two sets: one set representing the experts and another set representing the tasks. This setting simulates a scenario where committees of scientists interview other scientists, say, for a job position. We consider the experts to be the most prolific authors in the dataset, and this explains why in Table 3 the number of skills per expert is higher than the number of skills per task. For the set of authors who represent the tasks, we sample 2000 authors, and as before, we do not consider authors who have only one skill.

For creating a social graph among the experts we use co-authorship: two authors are connected with an edge if they have written at least one paper together. The cost of an edge is set to e^{-rD}, where D is the number of common papers coauthored by the two authors, and we set again $r = \frac{1}{10}$.

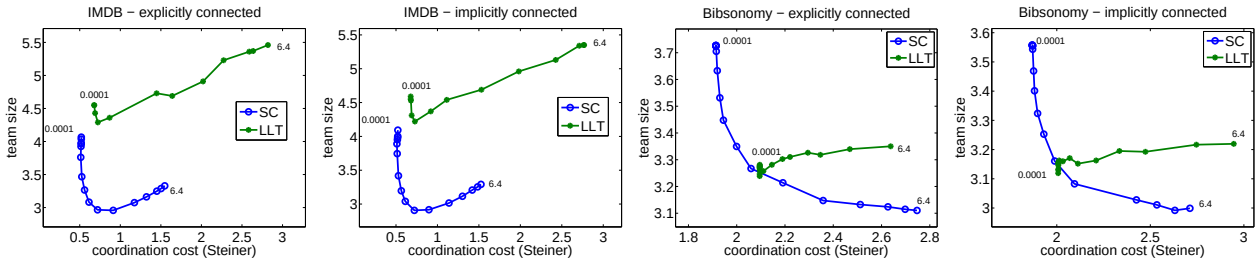
6.2 Results

We implement the algorithms discussed in Section 5 in the cases where the coordination cost is the Steiner-tree cost and the diameter. Tasks arrive sequentially and at each point the state of the system is described by the allocation cost $a(\mathbf{p})$ for each person $\mathbf{p} \in \mathcal{P}$, which is set according to the `ExpLoad` strategy based on the number of teams in which $\mathbf{p}$ has participated so far. The allocation costs reflect the load of people due to the previous tasks. Given a new task $\mathbf{J}$ our heuristic returns a sub-graph having nodes Q and edges T with the goal of minimizing $f(\lambda) = \lambda a(Q) + c(T)$.

Following the discussion of Section 5, we study the behavior of our algorithm as we vary the parameter λ. We start with a very small value of λ, which down-weights the load-balancing factor and emphasizes the coordination cost, and we increase λ exponentially, so that load balancing becomes more important. The results are shown in Figure 2 for the

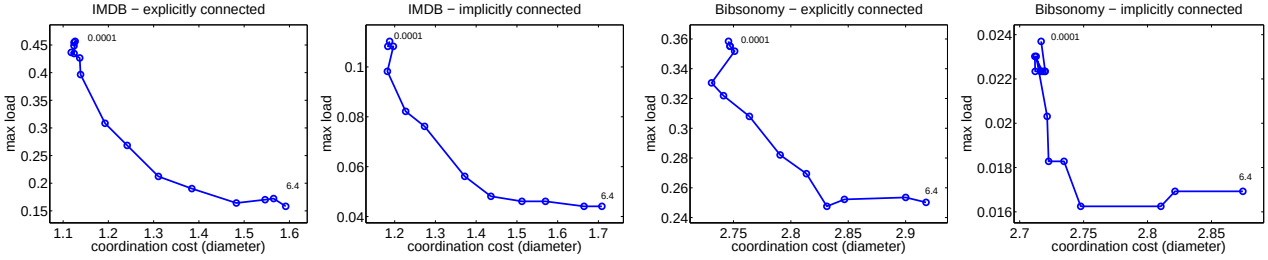


(a) Steiner cost vs. load on IMDB dataset (b) Steiner cost vs. load on Bibsonomy dataset



(c) Steiner cost vs. average team size on IMDB dataset (d) Steiner cost vs. average team size on Bibsonomy dataset

Figure 2: Trade-offs between Steiner coordination cost vs. max load and average team size. For computing the group Steiner tree on a task instance, two algorithms are used, the set-cover heuristic proposed in this paper (SC) and the heuristic proposed by Lappas et al. (LLT). The label next to the first and last data points shows the value of λ. For the in-between points shown in the curve the value of λ increases exponentially.



(a) Diameter vs. max load on IMDB dataset (b) Diameter vs. max load on Bibsonomy dataset

Figure 3: Trade-off between diameter and max load. The label next to the first and last data points shows the value of λ. For the in-between points shown in the curve the value of λ increases exponentially.

Steiner cost, and in Figure 3 for the diameter cost. The plots show the trade-offs among coordination cost, team size, and maximum load obtained in different runs of the algorithm parametrized by λ. In these plots one can see clearly that our heuristic exploits nicely the trade-off between load and coordination cost, making it possible to choose this parameter according to application-dependent objectives.

The results also indicate that ignoring the coordination costs ($\lambda \to \infty$) as in [2] yields solutions that have high coordination costs. Conversely, ignoring the workload ($\lambda \to 0$) as in [14] yields solutions that generate imbalanced schedules. Improvements from these extremes in one of the components of the cost (coordination cost or load balance) can be obtained by paying a moderate price in the other component.

Steiner cost. For the Steiner coordination cost, in addition to the set-cover heuristic discussed in Section 5, we also implement and compare with the group Steiner heuristic proposed by Lappas, Liu and Terzi [14], which we consequently refer to as the LLT heuristic. Figures 2(a) and 2(b) show the trade-off between Steiner cost and max load, as obtained by our online algorithm with call to the two different Steiner heuristics. Similarly, Figures 2(c) and 2(d) show the trade-off between Steiner cost and average team size. We see that in almost all cases the set-cover heuristic is better that the LLT heuristic. For the IMDB dataset, the set-cover heuristic produces solutions that dominate the LLT heuristic in terms of coordination cost and load. Also the LLT heuristic gives teams of significantly larger size.

For the Bibsonomy dataset the difference between the two heuristics is not so clear. First we see that in the case of implicitly-connected teams, the LLT heuristic outperforms slightly the set-cover heuristic in the coordination-cost vs. max-load space. On the other hand, in the case of explicitly-connected teams, the LLT heuristic performs significantly worse than the set-cover heuristic. In terms for average team size, for the smaller values of λ the two heuristics produce solutions that do not dominate each other, while the LLT heuristic produces very similar solutions to each other.

However, for the larger vales of λ, for the same coordination cost the set-cover heuristic gives smaller teams than the LLT heuristic.

Overall, the set-cover heuristic explores better the coordination-cost vs. max-load trade-off as a function of λ, it gives smaller teams, and produces solutions that in most cases dominate the solutions produced by the LLT heuristic.

Diameter. Our results for the diameter coordination cost on our two datasets (Figure 3) show the trade-off of diameter vs. maximum load for the `IMDB` dataset. We see that the dependency of the solution variables to the parameter λ is similar as in the case of Steiner cost, however the range of values is smaller than the corresponding values in the Steiner case, and the curves are less smooth, especially on the `Bibsonomy` dataset. One characteristic of the solutions we obtain for the diameter is that the average team size remains constant for all values of λ. For the `IMDB` dataset it is about 4.5 for the explicitly-connected teams, and about 3.3 for the implicitly-connected teams. For the `Bibsonomy` dataset it is about 7.5 for the explicitly-connected teams, and about 2.5 for the implicitly-connected teams. Note also the strange artifact that in some few cases (e.g., in Figure 3(b)) we obtain solutions that have both higher cost and load by increasing the λ. This is due to the limitation that we can obtain only an approximate solution to the Social Task Assignment problem, and while rare, it demonstrates how the hardness of the problem can lead to suboptimal solutions.

Alternative load measure. Measuring the load of the single most-loaded person is a measure that may be sensitive to outliers. For this reason we computed an alternative, most robust, measure, which is the average load of the 10% most-loaded people. We report here that the behavior of this measure is identical to the max-load measure, however, we omit the plots for lack of space.

Running time. Overall, our method is also efficient in practice: we observe running times of 4-5 seconds per task for the Steiner cost, and 9-10 seconds per task for the diameter. These times are recorded on a 4GB RAM, 2.5 GHz laptop.

7. CONCLUSIONS AND FUTURE WORK

We consider the problem creating teams of experts and assigning tasks to these teams, taking into account the requirements of every task, the work load of the experts, and their potential to collaborate or coordinate with each other. We develop approximation algorithms with provable theoretical guarantees, and our experiments demonstrate the ability of our solutions to keep low coordination costs and to balance the workload simultaneously also in practice.

The potential applications of the framework are numerous, resulting to a large number of different future directions. In this work we have assumed that we know the social network as well as the skill profiles of people. In practice, this information needs to be learned, for example, from observed data. It is also worth exploring how well teams can be created across different social networks exhibiting varying degrees of homophily. For instance, if the only connections are among experts with similar skills, covering a large set of skills with a tightly connected group may be difficult.

We have used two different measures of coordination cost; further results, not included here, indicate that similar ideas can be applied to the case where coordination cost is measured as the diameter of a team. We have also assumed that we want teams of people that are close to each other. The converse can be true in other settings, for instance if we want to create a panel or jury of experts from which we want independent opinions—here we probably want them to be far apart in the social network.

Finally, in this paper we have presented a static social network, while in practice people arrive and leave continually while the strength of their connections changes over time. In addition, we have assumed that the strength of the connections are independent of the individual task. Note that our algorithms and our results hold for all these settings as well. In future work we would like to evaluate the performance of our algorithms in such dynamic settings.

Key references: [2, 14]

8. REFERENCES

[1] Knowledge and Data Engineering Group, University of Kassel, Benchmark Folksonomy Data from BibSonomy. 2007.

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

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.

[4] Y. Azar. On-line load balancing. In *Theoret. Comp. Sci.*, 1992.

[5] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. In *SODA*, 1992.

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

[7] R. H. Coase. The Nature of the Firm. *Economica*, 4(16):386–405, November 1937.

[8] C. Dorn and S. Dustdar. Composing near-optimal expert teams: a trade-off between skills and connectivity. In *CoopIS*, 2010.

[9] A. Gajewar and A. D. Sarma. Multi-skill Collaborative Teams based on Densest Subgraphs. In *SDM*, 2012.

[10] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In *SODA*, 1998.

[11] M. E. Gaston and M. des Jardins. Agent-organized networks for dynamic team formation. In *AAMAS*, 2005.

[12] R. L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In *AFIPS*, 1972.

[13] M. Kargar and A. An. Discovering top-k teams of experts with/without a leader in social networks. In *CIKM*, 2011.

[14] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD*, 2009.

[15] C.-T. Li and M.-K. Shan. Team Formation for Generalized Tasks in Expertise Social Networks. In *SocialCom*, 2010.

[16] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *JACM*, 30, 1983.

[17] K. Mehlhorn. Assigning papers to referees. In *ICALP*, 2009.

[18] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, 2000.

[19] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem (extended abstract). In *SWAT*, 1996.

[20] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Many birds with one stone: multi-objective approximation algorithms. In *STOC*, 1993.

[21] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *CACM*, 28(2), 1985.

[22] V. V. Vazirani. *Approximation algorithms*. 2001.