Applied Network Science

RESEARCH Open Access

# Using machine learning to predict links and improve Steiner tree solutions to team formation problems - a cross company study

Peter Keane[1]*, Faisal Ghaffar[2] and David Malone[1]

*Correspondence:
peter.keane.2014@mumail.ie
[1]Maynooth University, Maynooth, Kildare, Ireland
Full list of author information is available at the end of the article

**Abstract**
The team formation problem has existed for many years in various guises. One important challenge in the team formation problem is to produce small teams that have a required set of skills. We propose a framework that incorporates machine learning to augment a collaboration graph with latent links between collaborators. This is combined with the solution of Steiner tree problems to form small teams that cover a specified set of tasks. Our framework not only considers the size of the team but also the likelihood that team members are going to collaborate with each other. We demonstrate our results using data from the US Patent office covering two different companies' inventor networks. The results show that this technique can reduce the size of suggested teams.

**Keywords:** Team formation, Link prediction, Steiner tree

## Introduction

Online team formation and collaboration within enterprises are vast and widely studied fields, not just with regards to patents (Wang et al. 2015; Lappas et al. 2009; Wu et al. 2013; Guimera et al. 2005; Zeng et al. 2016). Network science and complex systems have also emerged as useful tools in management science (Amaral and Uzzi 2007). Team formation is the problem of identifying a set of individuals with skills that are required by a task for its completion. One can imagine a setting where there is a stream of incoming tasks and a system is applied automatically to find groups of individuals who can complete those tasks. The individuals would typically be drawn from a pool of interconnected community members. Online social networks (e.g., Facebook, Linkedin, Enterprise networking tools etc.) in our personal and professional lives provide us opportunity to connect with each other and work together on common tasks. Despite these networks, it is quite challenging to find a team of experts to address particular tasks. In general, the optimal solution of this problem has been shown to be NP-hard (Lappas et al. 2009). However, there have been approximate solutions to the problem and many of these rely on social or collaboration networks among individuals. The proposed solutions leverage techniques from graph theory, utilising topological features of social and business networks, combined with graph theory to examine the existing structures of teams, interactions, and collaborations within the enterprise (Zhang et al. 2015; Tang et al. 2012; Newman 2001; Spadon et al. 2019). Other works have made use of machine learning, to see what individual's characteristics are likely to lead to the formation of a link between colleagues and suggest people with these features collaborate (Al Hasan et al. 2006; Lü and Zhou 2011).

A concern in online team formation research (e.g (Lappas et al. 2009; Wang et al. 2015)) is the reduction of communicative cost among selected team members. The *communicative cost* in a network is a measure of how effectively team members can collaborate with each other (Lappas et al. 2009). Naturally, there is a correlation between team size and costs in both financial and communication overhead (Gorla and Lam 2004; Pendharkar and Rodger 2009). The problem that we concentrate on in this paper is producing small, ideally minimal teams, that have the required skills to complete a task. In producing a small team, we aim to keep costs low.

In this paper, we propose a framework that incorporates both an individual's attributes as well as topological features from individual's network into a machine learning link prediction task. Our aim is to improve the performance of the Enhanced Steiner Tree Algorithm for team formation, as proposed in Lappas et al. (2009). Our aim is to use machine learning to produce an augmented graph, containing both current and potential links, before using the Enhanced Steiner Tree Algorithm (Lappas et al. 2009) to return a minimal team which covers all skills required to complete a given task. In doing so, we are combining previous work on link prediction using machine learning (Al Hasan et al. 2006) and work carried out using minimal spanning tree solutions (Lappas et al. 2009). We hope that by combining the two approaches we will overcome possible shortcomings of the individual techniques. The Enhanced Steiner Tree Algorithm provides good solutions to the team formation problem, but one possible shortcoming we point to is that it may sometimes neglect to consider isolated nodes or nodes that sit on unconnected components of the network. A bipartite structure, where vertices represent patents and inventors, was considered before settling on the Enhanced Steiner Tree. It is worth noting that the Enhanced Steiner Tree Algorithm creates another set of vertices representing expertise in certain topics, which are then connected to the vertices representing the inventors who hold that expertise. Since inventors are still directly connected, however, it is not a true bipartite graph.

This scheme, under some circumstances, will consider recommending collaboration between two inventors who aren't closely connected. There may be an appropriate inventor who has all the required skills, and also has parameters that indicate the potential for a good collaborative relationship with at least one member of the potential team. Thus, It may be expeditious to include this person in the recommended team instead of adding many more inventors to cover the required skills. This will help in terms of finding small teams, and so keep cost down.

To illustrate our scheme, we will show how we implemented it on a collaboration graph consisting of IBM Patent inventors. Our data is drawn from the US Patent Office (USPTO) data set. We evaluate the scheme in terms of the team sizes produced. To show robustness of the scheme, we also extract analogous data for Samsung and show that the same technique can be applied there.

We now give an outline of our paper. In "Preliminaries" section we discuss preliminaries, including notation and previous work carried out on the team formation problem. We then give a high-level summary of our framework in "Proposed framework" section and explain the details of our implementation in "Scheme implementation" section. The results of the applying the framework to both IBM and Samsung datasets are shown in "Results" section. We discuss our results and future work in "Discussion" section and conclude the paper in "Conclusion" section.

## Preliminaries

In this section we present our notation and summarise the work which we build on most directly. We note that this paper is an extended version of Keane et al. (2019). The original results were produced using data from 1976 to 2011 for IBM alone, however this paper uses a broader dataset, ranging from 1976 to 2019. We also present more details on the structure of the graph datasets, and investigate how it influences the team formation problem. For comparison, we also apply our technique to data for Samsung during the same timeframe.

### Mathematical notation

We denote by $G = (V, E)$ a weighted graph, with a node/vertex set $V$ and edge set $E$ and weights $w_e$. In our case, the graph will be a collaboration graph where the nodes will represent patent inventors and the edges collaborations. The weights, $w_e$, are assigned to each edge and represent the strength of collaborations, where the higher the weight, the less strong the collaboration is. We will also work with an induced subgraph of $G$ based on the skills required for the task. This is a subgraph $G'(V', E')$ where $V' \subset V$ and $E' \subset E$ are the edges with both nodes in $V'$. Full details of the definitions can be found in "Modern Graph Theory (Bollobás 2013)" or "Graph Theory" (Diestel 2017).

We also make use of Steiner Tree problems, named after Jakob Steiner. These are combinatorial optimisation problems and involve finding a minimum weighted tree that includes a particular set of nodes. Thus, they combine finding a minimal spanning trees with finding the shortest path problem. For example, in Hwang et al. (1992) it is used to find "*a shortest network which spans a given set of points*."

As input to the machine learning part of our scheme (see "Machine learning model training" section), we will use a number of simple graph-theory metrics, such as clustering coefficient (Watts and Strogatz 1998).

### Related work

As described in the introduction, we aim to form small collaboration teams. This is motivated by work including (Pendharkar and Rodger 2009), where the authors note that smaller teams should lead to lower communicative and co-ordination costs. However, they also observe that *"Most of the software development cost is related to the programmers' salaries"*. Consequently, recommending smaller teams should reduce costs both due to reducing the team size and lowering communication/co-ordination costs. Naturally, we still need to ensure that the requisite skills are available and that there is good potential for collaboration between individuals.

One of the primary algorithms we will use in this paper is the Enhanced Steiner Tree Algorithm (Lappas et al. 2009). We build on their problem definition for the *Team Formation Problem*: given a task $T$, individuals $V$ with skills, and a graph $G$ representing the network between those individuals, we aim to return a minimal team which contains individuals that not only cover the skills required for *T*, but can also work effectively together. It is also important that the skills of individual inventors contribute effectively to the completion of the task (Duch et al. 2010).

We noted above that we would use standard graph metrics as an input to our machine learning scheme. In addition, we will also make use of a number of metrics specifically designed, including bonding capital and bridging capital, intended for use in collaboration networks. Such metrics include social capital (Sharma et al. 2018) and collaborative ratio measurement (Wu et al. 2013). The metrics we use are described in more detail in "Machine learning model training" section.

## Proposed framework

The steps involved in our proposed model are outlined in the high level block diagram shown in Fig. 1.

The scheme begins with *Raw Data*, where all of the raw patent data is downloaded and summarised. We identified the number of inventors, to assess our graph size, and also how many different patent classes there, as these will represent our skills and determine the induced subgraphs.

At the *Pre-Processing* we select a subset of patents that we will be working with. In our case we will examine patents from IBM and Samsung, as two separate studies.

We next generate the *Skills & Inventor Network*. We consider the distribution of skills, (i.e. patent classes) across inventors, and created a cross-domain graph where each node represents an inventor. An edge between inventors represents a collaboration between them. We can also extract a subgraph where certain inventors have expertise in particular skills. This graph will be generated for each company, so there will be an IBM collaboration graph, and a Samsung collaboration graph.

In order to implement the scheme as seen in Fig. 1, we must examine the distribution of patent classes among inventors so that determine where particular inventors have specific expertise. As seen in Fig. 2 (which shows the top 20 classes in which IBM holds patents), one class, G06F (electric digital data processing), dominates the landscape of IBM patents. As such, an inventor who holds three or four patents in this class may not be considered an expert, because there may be many inventors who have multiples of that in this class.

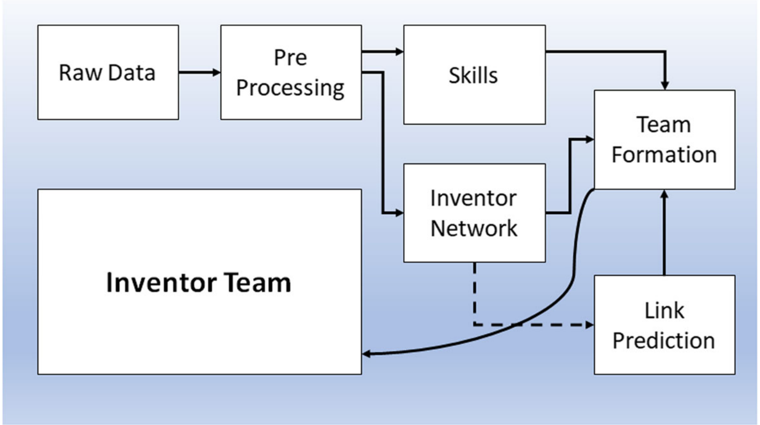


**Fig. 1** Block Diagram of Proposed Model

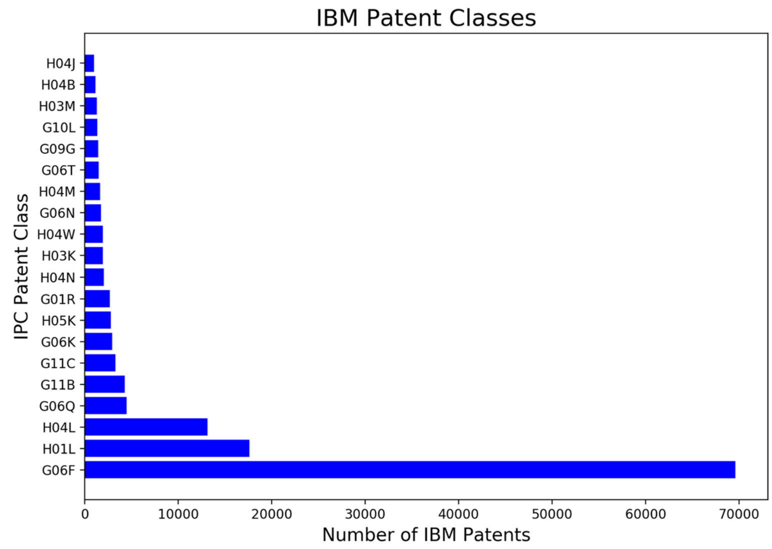


**Fig. 2** Distribution of IBM Patent Classes (Cooperative Patent Classifications)

However, an inventor who holds three or four patents in a class where IBM has fewer patents might be considered an expert within IBM, as they may be the most experienced inventor available to IBM in that particular class.

Using this data, a Python list was created for each patent class. This list consisted of the number of patents held by each inventor in that class. The list was sorted and zeros were removed. After these zeroes were removed a cutoff figure for finding the top quartile of numbers of patents held was established.

*Link Prediction* is the stage where machine learning is used to predict potential links based on certain topological and individual parameters of the inventors. Using these predictions, the original inventor network is augmented with the false positives from the machine learning model to produce an augmented network.

Finally, the *Enhanced Steiner Tree* (Lappas et al. 2009) is the algorithm that we use to form a small team which possesses the necessary skills to cover the skill requirement.

## Scheme implementation

In this section, we will lay out the steps involved in producing the graphs used for analysis, and how we processed the CSVs used for machine learning.

### The USPTO database

Data was downloaded in TSV format from the USPTO PatentsView website[1], specifically the *patent, ipcr, patent_inventor* tables and tables relating to company assignments.

Using the company assignee data for IBM, all the IBM patents were extracted. They were cross referenced with the other tables and Python's Pandas package was used to merge the relevant data to produce a final working table which consisted of the following headings: *patent_id, inventor_id,*
*full_class_id*. The full *full_class_id* used is the International Patent Classification[2].

[1] http://www.patentsview.org/download/
[2] https://www.wipo.int/classifications/ipc/en/

Our previous application of this method focused on the years from 1976–2011 (Keane et al. 2019). However, the USPTO now provide an updated databases and the records we use now run until August 2019. This has allowed us to test the method on an updated data set including more recent patents.

In an analogous manner, we also extracted Samsung's patents for the same timeframe from 1976–2019. This allowed us to compare the structures of the collaboration graphs of each company, and to see if our proposed framework, which was originally tested using IBM data, would also work on another company's collaboration graph.

### Graph creation and investigation

The IBM graph is a collaboration graph $G = (V, E)$, where $V$ contains the inventors of IBM patents, and $E$ contains edges representing any collaboration between the inventors on a patent. The extracted patent collaboration graph has 61,868 nodes and 285,262 edges with average degree of 9.22 and clustering coefficient of 0.62. The weights were assigned to each edge to correspond to the strength of the collaboration. It was calculated as inverse of the number of co-inventions to indicate the cost between two inventors. Note, these original weights were used mainly for exploring the graph before analysis and are not used in our prediction scheme.

The Samsung collaboration graph is slightly smaller than the IBM graph and had 46,268 nodes, and 252,548 edges. It had an average degree of 10.92, and an average clustering coefficient of 0.55. This compares quite closely to the IBM collaboration graph figures for average degree and average clustering coefficient; 9.22 and 0.62 respectively.

These graphs were created from the table generated by Algorithm 1. Python's Pandas package was used for grouping the data. As noted above, patent classes are used to represent the skills required on a team. A very small scale example of such a graph is shown in Fig. 3, where the weight is the cost of traversing the edge.

```
Algorithm 1: Creating the Collaboration Network G
Result: A Graph G(V,E), where V is the set of patent holders and E represents
        collaborations
Initialisation: Group the dataframe by unique patent ids, resulting in a Group with a
title of patent id, and within the group are all of the inventor ids associated with that
patent;
for Each Group do
    if Patent ID Group Has Only One Unique Inventor then
        Add node with Inventor ID;
    else
        for Each Pair of Inventors do
            if Pair is Not Connected then
                Create nodes for inventors and add edge of weight 1 between
                Inventors;
            else
                Add 1 to current edge weight;
Invert all weight values;
```

| patent id | inventor | patent id | inventor |
|---|---|---|---|
| 1 | Peter | 6 | Peter |
| 1 | Siobhàn | 6 | David |
| 2 | Peter | 6 | Faisal |
| 2 | Siobhàn | 7 | Peter |
| 3 | Peter | 7 | Faisal |
| 3 | Siobhàn | 8 | Peter |
| 4 | Peter | 8 | Faisal |
| 4 | Siobhàn | 9 | David |
| 5 | Peter | 9 | Hazel |
| 5 | David | 10 | John |

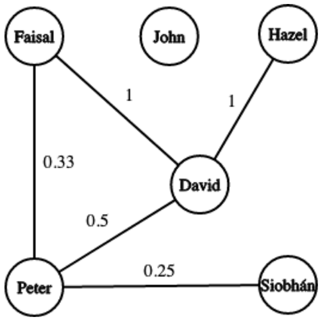


**Fig. 3** Example Dataset and Resulting Graph Created by Algorithm 1

The initial IBM graph has 61,868 nodes and 285,262 edges. As such it would be computationally expensive to perform many calculations with it. Consequently, for efficiency, the graph was reduced by selecting a subgraph relevant to specific patent class ids. This was achieved by inducing the subgraph on all nodes/inventors holding patents in those classes. Again, this procedure was also performed on the Samsung collaboration graph, which is also large enough for computation to be potentially costly. Pseudocode for the method which was used to split the graph is outlined in Algorithm 2.

```
Algorithm 2: Creating Patent Class Split Graphs
Result: A Graph H(V,E) where V is the set of patent holders with patents in the
        specified classes, and E is the set of edges representing collaboration between
        those patent holders
Initialisation: Given the collaboration graph, G, described in Algorithm 1, a set of
patent classes, and the dictionary with inventors as key, and all classes in which they
hold patents as value;
Create empty list, L;
for Each Patent Class in Set do
    for Each Inventor Key in Dictionary do
        if Patent Class is in Inventor's Set then
            Add node to list, L;
Induce subgraph of G on all nodes in L
```

The left of Fig. 4 shows the distribution of degrees among IBM inventors width a bin width of five. It is clear that the majority of nodes have a small degree, somewhere between 0 and 5. It is worth looking at the whole set of numbers on a log scale, so that we can see what is going on at the tail of the histogram. This is shown on the right of Fig. 4, also with a bin width of five.

Interestingly, the majority of occurrences are of degrees between zero and five, which is significantly below the average degree of 9.2. This suggests that there are, indeed, some IBM inventors with large collaborative networks that are increasing the average figure and pulling it up. A cursory look at the degree data shows us that the minimum degree is 0, predictably, as there will be inventors who work alone, and the maximum is 332. For

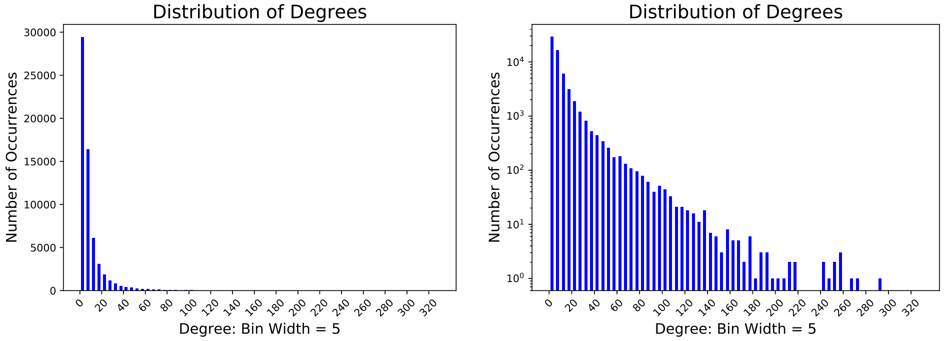


**Fig. 4** Degree Distribution — IBM. Linear scale (left), log scale (right)

comparative purposes, the maximum degree for the Samsung graph is 464, and of course, the minimum degree is 0.

It is worth taking a look at the degree distributions of the graph after we have trimmed it by class, as per Algorithm 2.

### Machine learning model training

Next, using the collaboration graphs $G$ created in the previous section, new datasets were created for the purpose of training machine learning models. The aim is to produce a set of features that might predict if two inventors are likely to collaborate, even if that is not reflected in the present patent data sets.

Specifically, datasets were created with the columns: *vi, vq, bonding_vi, bridging_vi, bonding_vq, bridging_vq, vi_collab_ratio, vq_collab_ratio, vi_cluster, vq_cluster, vi_patents, vq_patents, common_neighbors, expert_jaccard, resource_allocation, connected*. We will explain each of these in turn, but note that each was selected to help with the prediction of potential collaborations while being relatively inexpensive to calculate from the graph. For each pair of inventors, *vi* and *vq* refer to each inventor of the pair. Some of the measurements relate to each inventor individually while others depend on both *vi* and *vq*. The individual metrics are indicated by having *vi* or *vq* in their name.

The bonding and bridging capital (Sharma et al. 2018) of each inventor are designed to measure homogenity and heterogenity of ties of inventors. They are calculated as follows. First, the communities of the graph $G$ were extracted using NetworkX's best partition function (in the community module). We then considered how many neighbors of each inventor are in the same community. The bonding capital is then given by:

$$\text{Bonding Capital} = \frac{\left|\text{Neighbors in same community}\right|}{\left|\text{Total number of neighbors}\right|}. \tag{1}$$

Similarly, the bridging capital is given by:

$$\text{Bridging Capital} = \frac{\left|\text{Neighbors } \textbf{not} \text{ in same community}\right|}{\left|\text{Total number of neighbors}\right|}. \tag{2}$$

The collaborative ratio (Wu et al. 2013) of each inventor is given by:

$$\text{Collaborative Ratio} = \frac{|\text{Collaborative patents of inventor}|}{|\text{Patents of inventor}|}, \tag{3}$$

where a *collaborative patent* is any patent which has more than one inventor. This measure is included to capture inventors who are likely to work collaboratively.

The values *vi_cluster* and *vq_cluster* are simply the clustering co-efficient of each inventor. The ***clustering coefficient*** of a node or vertex $v$ is defined as;

$$c_v = \frac{2T_v}{d(v)(d(v)-1)} \tag{4}$$

where $T_v$ is the number of triangles formed through node $v$. These measures were included as a measure of collaboration between an inventor's immediate neighbours. Likewise, *vi_patents* and *vq_patents* are the total number of patents held by each inventor, and give an indication of inventors current success in patenting. The *common_neighbors* are the number of neighbours common to both *vi* and *vq*, and gives a simple measure of how overlapped the inventors' collaborators are.

The Jaccard Index of two sets is given by:

$$J(A,B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid}. \tag{5}$$

To find the *expert_jaccard* of *vi* and *vq*, we take the Jaccard Index of the set of patent classes in which each inventor held expertise. We define expertise by an inventor being in the top quartile of inventors in a patent class, as measured by number of patents held in that class. We consider this a simplistic measure of the common interests of the inventors.

Resource allocation is the *resource allocation index* of a pair of nodes, $u$ and $v$ as given by Python's NetworkX module and is defined as:

$$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\mid \Gamma(w) \mid} \tag{6}$$

where $\Gamma(u)$ denotes the set of neighbors of $u$. The resource allocation index has previously been identified as useful in missing link prediction (Zhou et al. 2009).

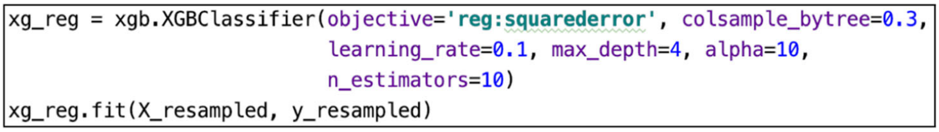
Finally, the connected value is 1 if there is an edge between *vi* and *vq* in the collaborator graph, and 0 if there is no edge.

We now have a dataset for IBM and Samsung which we can use to predict if two nodes are connected based on the features described above.

## Machine learning model

We now use Python's XGBoost module to create models, with the *connected* column providing the binary classification target for prediction. Figure 5 shows the parameters used with XGBoost to create the model. We experimented with various machine learning models, such as LR, SVM, and CART. We used XGBoost as it provided the most accurate model.

As there is no link between most pairs of inventors, the connection column exhibits substantial skew that might impact predictions. To counteract this, SMOTE oversampling (Chawla et al. 2002) was employed to ensure a more accurate model was trained. SMOTE



```
xg_reg = xgb.XGBClassifier(objective='reg:squarederror', colsample_bytree=0.3,
                           learning_rate=0.1, max_depth=4, alpha=10,
                           n_estimators=10)
xg_reg.fit(X_resampled, y_resampled)
```

**Fig. 5** Snippet of Code Showing XGBoost Parameters

works by creating synthetic points on lines between points in the minority class, and using this to provide more training points.

Following the training of this model, a second graph split between random domains was created as before, and a dataset of the same format was created. The model was tested on this new, unseen, dataset and the predicted *connected* column was compared against the actual one. From this, the accuracy of the model was measured. This was repeated for the Samsung data.

Figure 6 (left) shows that the model provides good accuracy across all classifications for the IBM data. The model is predicting an edge between inventors, with a 92% accuracy. Conversely, it is incorrectly predicting that there is a zero, or no edge between inventors, 8% of the time. Of particular interest to us is the top right quadrant. Our model predicts that there is no edge between inventors, 99% of the time, and predicting incorrectly that there is an edge between inventors just 1% of the time. We will use this 1% of *false positives* to create additional links in our graph. The results show similar accuracy for the Samsung graph, as shown in Fig. 6 (right). These results are similarly promising, with 8% of instances where the model incorrectly predicted an edge provides us with the edges we will use to augment our Samsung graph.

### Creating new links

Having built a model to predict links, we are now ready to introduce the graph that is augmented with extra edges that are predicted by that model. This is to overcome a limitation of using the Enhanced Steiner Tree Algorithm (Lappas et al. 2009) directly on the collaboration graph, in that it will not consider collaborations that sit in unconnected components.

As such, we propose that artificial links are created on the graph between pairs of inventors for which our machine learning model "incorrectly" predicted there was a link, corresponding to the top right quadrant of the confusion matricies shown in Fig. 6. The machine learning model predicted links, based on the parameters discussed in "Machine learning model training" section, that these two inventors had attributes that indicated a potential collaborative relationship. Consequently, we will add these potential links using the procedure in Algorithm 3.

The graph was augmented by creating an edge between every pair of inventors for which the model predicted an edge, but there didn't already exist such an edge. We also need to

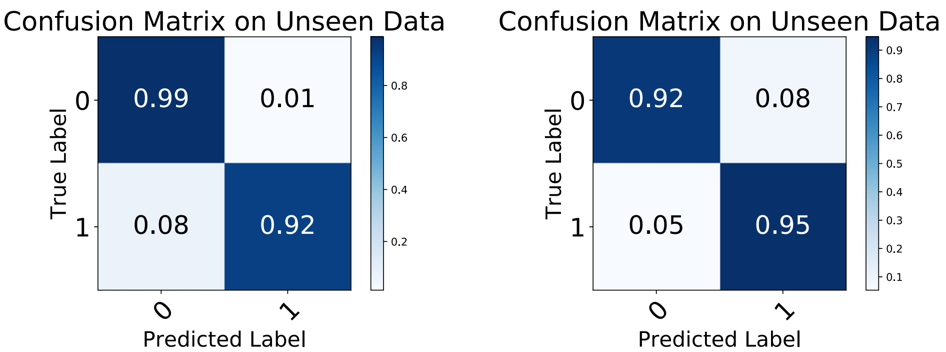


**Fig. 6** Confusion Matrix of Machine Learning Model (unseen data). IBM (left) and Samsung (right)

```
Algorithm 3: Creating Machine Learning Augmented Graph
Result: An Augmented Graph with Machine Learning False Positive Edges added
Initialisation: Gather all pairs of nodes between which there was an edge predicted
by the machine learning model but no edge in the collaboration graph;
for Each Pair of Nodes do
    Add edge between them;
```

provide a weight for the edge. This was set as $1-P$ where $P$ was the probability of an edge existing as predicted by the machine learning model. Note that this weight is applied to *all* edges, including those included in the original collaboration graph. This ensures that all weights are comparable.

This provides us with our augmented graph on which we will run the Enhanced Steiner Tree Algorithm to identify a team.

To show how this changes the graph, we looked at the degree distribution of the collaboration graph, as we did for the original graph in "Graph creation and investigation" section. We will look at the degree of the graph trimmed for a subset of classes, both before and after augmentation. This is shown in Fig. 7. It can be seen that the average degree value increases, which is obvious. However, in the augmented graph, the base of the histogram is much wider, i.e., there are more occurrences of degrees across a wide range.

### Testing and evaluation

Given the relation between team size and cost described above, our main criterion for testing will be the team size required to cover a particular set of skills.

This raised an interesting point that, having added in edges predicted by the machine learning model, the cardinality of the team would almost certainly be smaller, as the augmented collaboration graph would now be better connected. In fact, even if we added edges at random, we would expect teams to be smaller. We take advantage of this intuition to evaluate our scheme. For comparison, we use teams generated from a randomly augmented graph. To generate this randomly augmented graph, we add edges at random between pairs of nodes until the number of edges of this randomly augmented graph

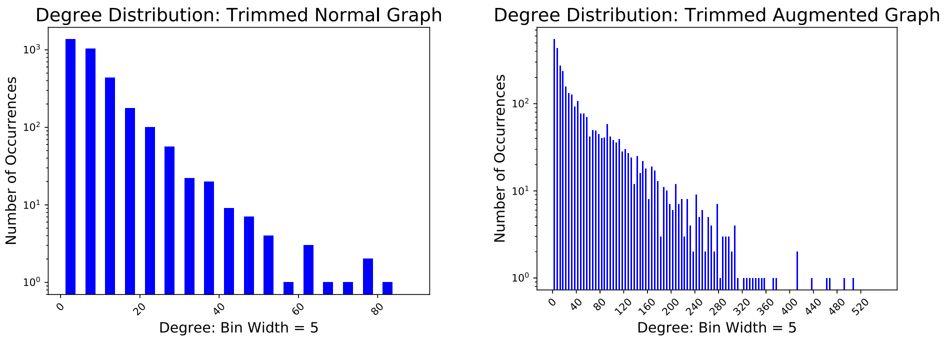


**Fig. 7** Degree Distribution — IBM graph (log scale). Original Trimmed Graph (left), Augmented Trimmed Graph (right)

equals the number of edges of the machine learning augmented graph. This is outlined in Algorithm 4.

```
Algorithm 4: Creating Randomly Augmented Graph
Result: A Graph with Random edges added
Initialisation: Gather all pairs of nodes between which there while #Edges Random Graph <#Edges ML Augmented Graph do
    Choose random pair of unconnected nodes and add edge between them;
```

We note that this produces a substantially different distribution of edges to that produced by our machine learning scheme. Figure 8 shows the degree distribution for a randomly augmented graph. When compared to the distributions shown in Fig. 7, we see that the randomly augmented graph has a larger number of high degree nodes. The edge weights of all graphs, original, ML augmented and randomly augmented are set to $1 - P$ where $P$ is the probability of there being an edge as per our machine learning model. This ensures an even comparison between all three graphs where edge weights are considered.

In our evaluation, we will compare the average team size required for different skillsets (i.e. sets of required skills). Since there are many skillsets, we compare the average team size as a function of the number of skills.

Specifically, a task $T$ is a set of patent classes that could be interpreted as the skillset required for the task at hand. For this task, the Enhanced Steiner Tree Algorithm (Lappas et al. 2009) can be run on (1) the original graph, (2) the machine learning augmented graph, and (3) the randomly augmented graph. For completeness, we restate the Enhanced Steiner Tree Algorithm from (Lappas et al. 2009) in Algorithms 5 and 6. The line, *"EnhanceGraph(G,T)"* in Algorithm 6 makes a pass over the graph G. An additional node $Y_j$ is created for every skill in $T$ and these nodes are edge connected to an inventor if and

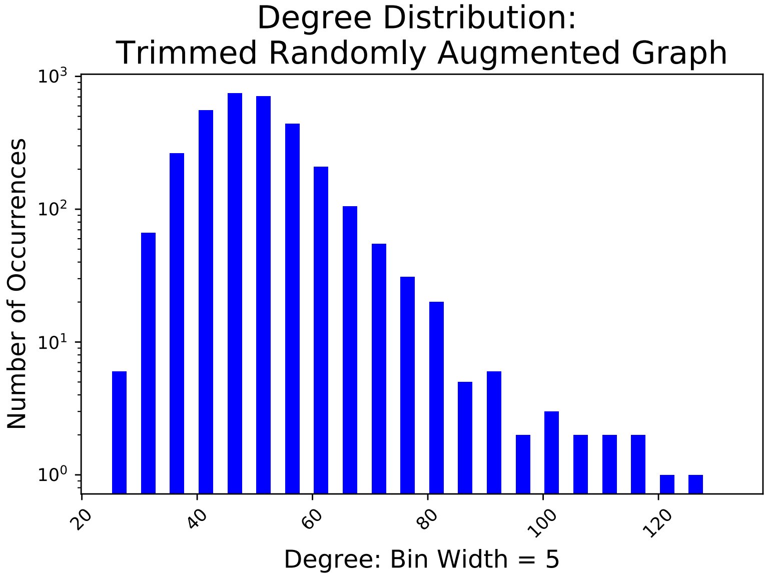


**Fig. 8** Degree Distribution — IBM Randomly Augmented Trimmed Graph (log scale)

only if that inventor has expertise in that skill. The distance between inventors and these new nodes is set to be greater than the sum of pairwise distances of nodes in G.

```
Algorithm 5: Steiner Tree Algorithm
Result: Team $X_0 \subseteq X' \subseteq X$ and subgraph $G[X']$
Input: Graph $G(V, E)$;
required nodes $X_0$ and Steiner nodes $X \setminus X_0$;
$X' \leftarrow v$ where $v$ is a random node from $X_0$;
while $X_o \setminus X' \neq \emptyset$ do
    $v* \leftarrow arg, min_{u \in X_0 \setminus X'} d(u, X')$
    if $Path(v*, X') \neq \emptyset$ then
        $X' \leftarrow X' \cup \{Path(v*, X')\}$
    else
        Return Failure
```

```
Algorithm 6: Enhanced Steiner Tree Algorithm
Result: Team $X' \subseteq X$ and subgraph $G[X']$
Input: Graph $G(V, E)$;
Individuals' expertise set $\{X_1, ..., X_n\}$ and task $T$;
$H \leftarrow EnhanceGraph(G, T)$
$X_H \leftarrow SteinerTree(H, \{Y_1, ..., Y_k\})$
$X' \leftarrow X_H \setminus \{Y_1, ..., Y_k\}$
```

## Results

This evaluation was initially performed by selecting 10 patent classes (at random) as the task, $T_i$. The framework was run 100 times, and the average size of the resulting team was recorded. This was then repeated for a total of four different random tasks, $T_i$, each of size 10.

Table 1 shows the average team size for each of these ten tests. Results are shown using the original IBM graph $G$, the randomly-augmented graph and the graph augmented using our machine learning scheme. Our augmented graph returns a smaller team than the original collaboration graph, as expected, but also consistently returns a smaller team than the graph with randomly created links, even though the random graph and augmented graph both have the same number of edges.

To see how the scheme scaled, a random $T$ was selected as before, but with a size of 8. The same tests were again run, in this case 50 times, and the average number of members

**Table 1** Average Size of Teams for Task of size 10 — IBM Graph

| Task | G | G Random-Augmented | G ML-Augmented |
|---|---|---|---|
| $T_1$ | 7.84 | 7.82 | **6.52** |
| $T_2$ | 8.79 | 6.41 | **6.11** |
| $T_3$ | 6.99 | 5.50 | **4.48** |
| $T_4$ | 13.33 | 11.23 | **8.40** |

in each team recorded. $T$ was increased by 1 random patent class after each iteration until $T$ contained 24 patent classes. A similar testing method was employed in Lappas et al. (2009), where the authors compared the Enhanced Steiner Tree algorithm against other algorithms, including a variation of the cover set algorithm, greedy cover, rarest first, etc. Since the cardinality of teams returned by the Enhanced Steiner Tree algorithm in Lappas et al. (2009) was consistently lower than the algorithms against which it was compared, the decision was made that the Enhanced Steiner Tree algorithm, as it appears in Lappas et al. (2009) would be a good benchmark against which to test our scheme.

The results for the IBM graph are shown in Fig. 9 as the number of patent classes in task $T$ varies. The number of elements in a task $T$ is shown on the x-axis, and the average number team members over 50 runs is shown on the y-axis.

The average cardinality of the team returned for the augmented graph is lower than that of the original collaboration graph, intuitively and as we discussed in "Testing and evaluation" section. More interestingly, the average cardinality of the team returned when the Enhanced Steiner Tree Algorithm is run on the machine learning augmented graph is consistently lower than the graph which was augmented with random links, even though the random graph has the same number of edges as the machine learning augmented graph.

This result is replicated for nine other seed random tasks $T$ as shown in Fig. 10. In this table, each graph follows the same layout and axes as Fig. 9, although some of the y-axis scales differ. We see that the machine learning augmented graph consistently returns smaller teams than those returned by either the randomly augmented graph or the regular graph. This is largely consistent regardless of the exact skills in the skillsets, which differ between subgraphs. Note that the details of each subgraph does depend on the set of skills in the set $T$, which explains the different structures observed between the subgraphs.

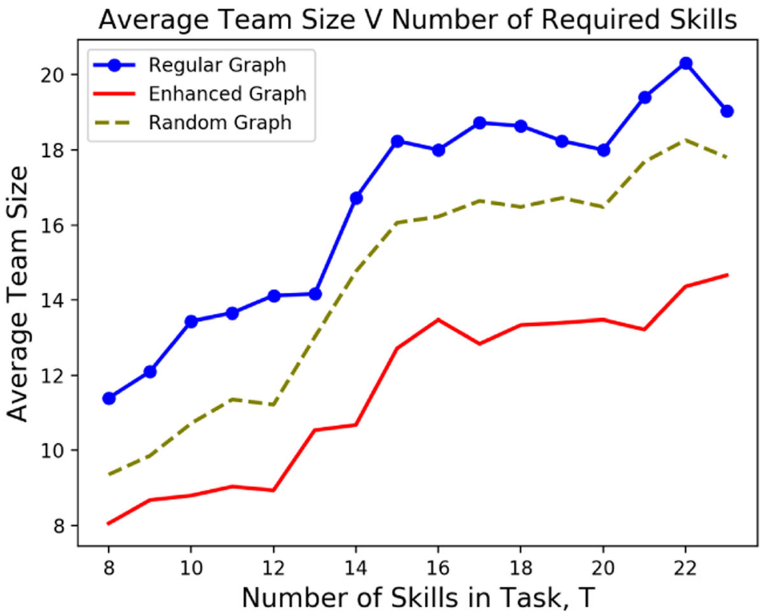


**Fig. 9** Average Cardinality of IBM Team V Number of Skills in task, $T$

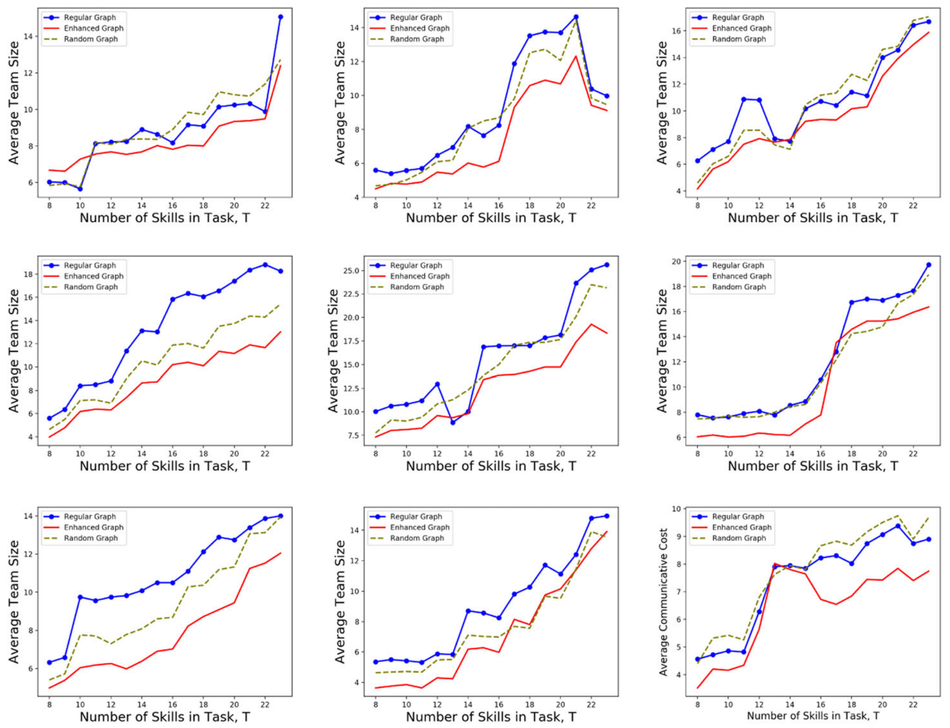


**Fig. 10** Average Cardinality of Teams for Differing Task Sizes — IBM

We repeated these experiments with the Samsung data using the same testing procedure. The results of these tests are shown in Table 2 and Fig. 11. As for the IBM collaboration graph, we seen that our machine learning augmented graph consistently returns smaller teams for the Samsung data, just as for the IBM graph. We saw earlier that the Samsung graph is smaller than the IBM graph, but with broadly similar average degrees. Interestingly, the teams formed for comparable skill sets seem slightly larger in the case of the Samsung graph, possibly representing distributions of skills within the companies.

So far, we have focused on team size as the primary factor. Alternatively, communicative cost might be a concern. As we have weighted the edges to represent the ease of collaboration between inventors, one way to measure communicative cost would be to consider the sum of weights edges in the team. The results of this communicative cost metric are shown in Fig. 12 for IBM (left) and Samsung (right). Interestingly, though the average cardinality of the team suggested by the machine learning graph is lower for both organisations, the sum of weights tends to be higher at larger numbers of skills. The physical or financial communicative cost is, of course, determined by the circumstances. Consequently, it may be interesting to explore other, more meaningful measurements of the

**Table 2** Average Cardinality of Teams for Task of size 10 — Samsung Graph

| Task | G | G Random-Augmented | G ML-Augmented |
|---|---|---|---|
| $T_1$ | 12.29 | 9.79 | **8.93** |
| $T_2$ | 5.73 | 4.62 | **3.82** |
| $T_3$ | 7.39 | 6.92 | **5.79** |
| $T_4$ | 7.01 | 6.51 | **5.65** |

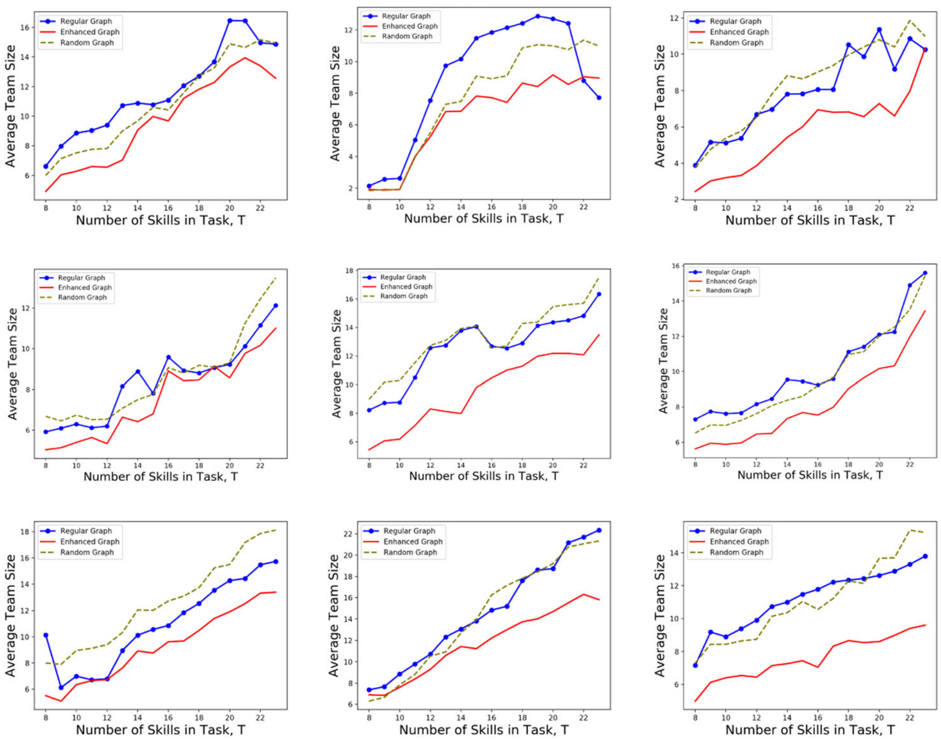


**Fig. 11** Average Cardinality of Teams for Differing Task Sizes — Samsung

communicative cost including real life aspects, such as location, time zone, access to high speed internet, colaboration tools, etc.

## Discussion

We first note that our results are consistent with our previous evaluation on a more limited IBM data set (Keane et al. 2019), where we observed smaller teams using the graph augmented using machine learning.

One particularly interesting new observation we've made is that when we augment the graph with machine learning we end up with more isolated nodes than if we randomly augment the graph. For example, the machine learning augmented graph for IBM has 69

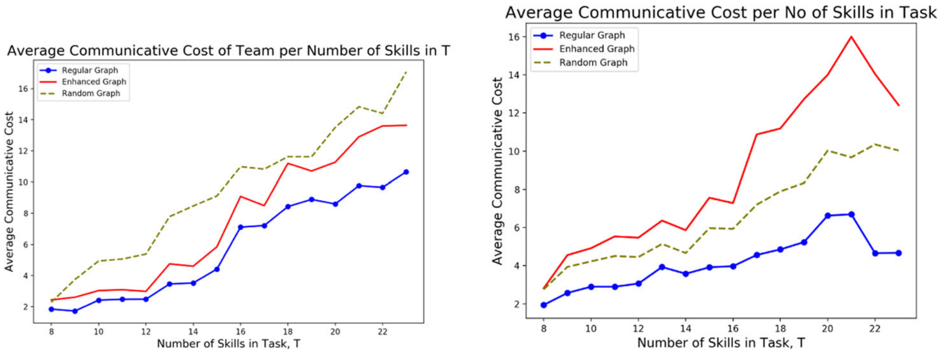


**Fig. 12** Communicative Cost based on Machine Learning Probabilities. IBM (left) and Samsung (right)

isolated nodes, but the randomly augmented IBM graph has no isolated nodes, i.e. it is one large connected component. This was the same for the Samsung collaboration graph; the machine learning augmented graph has 102 isolates and the randomly augmented graph has no isolated nodes. However, as seen in the results, our machine learning augmented graph consistently returns smaller teams. So although the average degree of both the machine learning augmented graph and the randomly augmented graph is the same, the machine learning augmented graph is clearly having edges added in a manner which is conducive to finding smaller teams.

When we consider why we see these differences in team size, Figs. 7 and 8 suggest that the machine learning augmentation is connecting a number of potentially collaborative inventors quite highly, and these nodes may be providing short paths through the graph, facilitating small teams. This raises a possible concern that these highly-connected inventors might be frequently included in teams, leading to them being overloaded. One possible solution to this problem would be to rerun the algorithm if a suggested team contains an overloaded inventor.

We have touched on collaborative cost, but left a more detailed study for future work. While we used edge weights based on the machine learning probabilities an indication of communicative cost, this shows certain limitations, for example we saw that the relative communicative costs seemed to show different behaviour for the IBM and Samsung datasets. Consequently, we believe would be more interesting to use a collaborative cost based on actual financial or work-practice based metrics. These could potentially be predicted for the augmented graph based on a second machine learning model.

If these more direct methods were available to measure communicative cost they could be combined with the team members in an expanded model. This would allow the model to trade team size against communicative costs. Combined, this could provide a new angle on the team formation problem.

For simplicity, this work has also largely ignored the temporal nature of the data. Several avenues present themselves for considering the temporal aspects of the graph. For example, one could consider windowing the data to consider a series of graphs. Another possibility would be to include temporal information in the data presented to the machine learning algorithm and then to include the time that the team should be formed as an input to the problem.

Taking temporal aspects of the data into account might also provide new ways to evaluate suggested teams. For example, one might window the data and ask for teams for 2019 and then compare observed teams with suggested teams. However, this would not test our aim of suggesting potential good collaborations that might not arise without some proactive intervention.

It may also be possible to represent the data as a graph in other useful ways, for example as a bipartite networks, where one set of nodes is the expertise, and the other is the inventors. We note that the Enhanced Steiner Tree Algorithm actually takes an approach like this internally by creating nodes to represent the skills, but it is not a true bipartite graph, as the inventors are still connected directly.

Another potential application of such a method could be in aiding a company in acquisitions or new hires. For example, if IBM were to include another smaller company in a collaboration graph, the smaller company's component might not initially be directly connected to IBM's component of the graph. However, after augmentation, the two components would likely become connected if the machine learning algorithm spotted potential collaborative relationships between inventors from both companies. This could, in turn, lead to inventors from other companies being recommended as potential collaborators, suggesting cross-company collaboration.

More broadly, we believe that this technique should generalise to other types of collaboration that require team formation. We have restricted our machine learning inputs to relatively straight forward graph metrics, allowing the techniques application on most graphs. Of course, the scheme could also include domain-specific metrics when predicting collaborations.

## Conclusion

In this paper we have addressed the team formation problem with a view to identifying small teams requiring particular skills by using the structure of the collaboration graph. To avoid the problem of only recommending currently collaborating team members, we use both machine learning and graph-based methods. The machine learning is used to augment the graph by adding edges based on the potential for collaborative relationships.

Using this technique, for both IBM and Samsung datasets, we see smaller team across a range of task sizes and particular skills required using this augmented graph. These smaller teams appear both with respect to the original graph and a randomly augmented graph of comparable size. This suggests our augmented graph produces compact teams with good collaborative potential.

### Datasets

The datasets generated and analysed during the current study are available in the USPTO PatentsView repository, https://www.patentsview.org/download/.

### Acknowledgements

The authors would like to thank the members of the IBM Ireland Innovation Exchange Team (IIX Team) for their technical help.

### Authors' contributions

PK analysed the datasets, created the graphs, coded the algorithms, and produced the plots and results. PK, FG and DM were involved in conception and design of the work and writing and editing of the manuscript. All authors read and approved the final manuscript.

### Funding

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

### Competing interests

The authors declare that they have no competing interests.

### Author details

[1]Maynooth University, Maynooth, Kildare, Ireland. [2]Innovation Exchange, IBM, Dublin, Ireland.

Received: 3 March 2020 Accepted: 12 August 2020

Published online: 27 August 2020

### References

Al Hasan M, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning. In: SDM06: Workshop on Link Analysis, Counter-terrorism and Security Vol. 30. pp 798–805

Amaral LAN, Uzzi B (2007) Complex systems—A new paradigm for the integrative study of management, physical, and technological systems. Manag Sci 53(7):1033–1035

Bollobás B (2013) Modern Graph Theory. Graduate Texts in Mathematics, Vol. 184. Springer, New York

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Diestel R (2017) Graph Theory. Graduate Texts in Mathematics, Vol. 173. Springer, Heidelberger

Duch J, Waitzman JS, Amaral LAN (2010) Quantifying the performance of individual players in a team activity. PloS ONE 5(6):e10937

Gorla N, Lam YW (2004) Who should work with whom? Building effective software project teams. Commun ACM 47(6):79–82
Guimera R, Uzzi B, Spiro J, Amaral LAN (2005) Team assembly mechanisms determine collaboration network structure and team performance. Science 308(5722):697–702
Hwang F, Richards D, Winter P (1992) The Steiner Tree Problem. Annals of Discrete Mathematics, Vol. 53. Elsevier, Amsterdam
Keane P, Ghaffar F, Malone D (2019) Using machine learning to predict links and improve Steiner tree solutions to team formation problems. In: International Conference on Complex Networks and Their Applications. Springer. pp 995–1006
Lappas T, Liu K, Terzi E (2009) Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. pp 467–476
Lü L, Zhou T (2011) Link prediction in complex networks: A survey. Phys A: Stat Mech Appl 390(6):1150–1170
Newman ME (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64(2):025102
Pendharkar PC, Rodger JA (2009) The relationship between software development team size and software development cost. Commun ACM 52(1):141–144
Sharma R, McAreavey K, Hong J, Ghaffar F (2018) Individual-level social capital in weighted and attributed social networks. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE. pp 1032–1037
Spadon G, de Carvalho AC, Rodrigues-Jr JF, Alves LG (2019) Reconstructing commuters network using machine learning and urban indicators. Sci Rep 9(1):1–13
Tang J, Wu S, Sun J, Su H (2012) Cross-domain collaboration recommendation. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. pp 1285–1293
Wang X, Zhao Z, Ng W (2015) A comparative study of team formation in social networks. In: International Conference on Database Systems for Advanced Applications. Springer. pp 389–404
Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world'networks. Nature 393(6684):440–442
Wu S, Sun J, Tang J (2013) Patent partner recommendation in enterprise social networks. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. ACM. pp 43–52
Zeng XHT, Duch J, Sales-Pardo M, Moreira JA, Radicchi F, Ribeiro HV, Woodruff TK, Amaral LAN (2016) Differences in collaboration patterns across discipline, career stage, and gender. PLoS Biol 14(11):e1002573
Zhang J, Lv Y, Yu P (2015) Enterprise social link recommendation. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM. pp 841–850
Zhou T, Lü L, Zhang YC (2009) Predicting missing links via local information. Eur Phys J B 71(4):623–630

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.