Subjectively Interesting Connecting Trees

Florian Adriaens(✉), Jefrey Lijffijt, and Tijl De Bie

IDLab, Department of Electronics and Information Systems,
Ghent University – Imec, Ghent, Belgium
{florian.adriaens,jefrey.lijffijt,tijl.debie}@ugent.be

Abstract. Consider a large network, and a user-provided set of *query* nodes between which the user wishes to explore relations. For example, a researcher may want to connect research papers in a citation network, an analyst may wish to connect organized crime suspects in a communication network, or an internet user may want to organize their bookmarks given their location in the world wide web. A natural way to show how query nodes are related is in the form of a tree in the network that connects them. However, in sufficiently dense networks, most such trees will be large or somehow trivial (e.g. involving high degree nodes) and thus not insightful. In this paper, we define and investigate the new problem of mining *subjectively interesting trees* connecting a set of query nodes in a network, i.e., trees that are highly surprising to the specific user at hand. Using information theoretic principles, we formalize the notion of interestingness of such trees mathematically, taking in account any prior beliefs the user has specified about the network. We then propose heuristic algorithms to find the best trees efficiently, given a specified prior belief model. Modeling the user's prior belief state is however not necessarily computationally tractable. Yet, we show how a highly generic class of prior beliefs, namely about individual node degrees in combination with the density of particular sub-networks, can be dealt with in a tractable manner. Such types of beliefs can be used to model knowledge of a partial or total order of the network nodes, e.g. where the nodes represent events in time (such as papers in a citation network). An empirical validation of our methods on a large real network evaluates the different heuristics and validates the interestingness of the given trees.

Keywords: Exploratory Data Mining · Subjective interestingness
Information theory · Graphs · Graph pattern mining

1 Introduction

Given a graph and a set of query nodes, we are interested in connecting these query nodes in a minimal but highly informative manner. *Minimal* in the sense that we are looking for a preferably small subgraph to which the query nodes belong to. *Informative* meaning that our aim is to show a user a subgraph that is highly insightful to them, i.e., the subgraph contains relationships between nodes that are unexpected and surprising to the user. In this paper we consider the case of connecting the query nodes through a subgraph that has a tree structure.

M. Ceci et al. (Eds.): ECML PKDD 2017, Part II, LNAI 10535, pp. 53–69, 2017.
https://doi.org/10.1007/978-3-319-71246-8_4

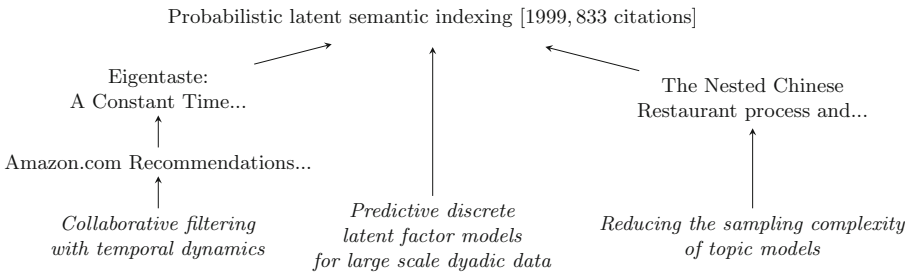

Fig. 1. Tree connecting the three most recent KDD best paper award winners listed at the official ACM SIGKDD webpage (http://www.kdd.org/awards/sigkdd-best-research-paper-awards) that are also present in the Aminer ACM-Citation-network v8 (https://aminer.org/citation, [10]). The result of our algorithm with heuristic s-IR given no background knowledge about the graph. See Sect. 5 for more details.

An example: suppose we have a scientific paper citation network, where edges denote that one paper references another. Given a set of query papers, a directed tree containing these query papers is one possible way to represent interesting citation relationships between these papers. The root of the tree could represent a paper that was (perhaps indirectly) highly influential to all the papers in the query set. Connections between nodes are subjectively interesting if they are surprising. E.g., if a user knows certain papers are widely cited (have high degree), those papers would be less interesting to find in the connecting tree: the user already expects this connection to exist and hence does not learn much.

An example of an informative tree connecting three recent KDD best paper award winners where no prior knowledge about degrees was assumed is given in Fig. 1. Another example application would be to organize your bookmarks by constructing a tree where the bookmarks are the query nodes and the network is the WWW. With a prior containing the degrees of the nodes in the network, an informative tree would partition the bookmarks according to links that are surprising and hence specific to a sub-network (they have low degree). Our method find such trees without doing community detection as an intermediate step.

The main question here is: what makes a certain tree interesting to a given user? We believe that the goal of *Exploratory Data Mining* (EDM) is to increase a user's understanding of his or her data in an efficient way. However, we have to consider that every user is different. It is in this regard that the notion of *subjective interestingness* was formalised [9] and more particularly the creation of the data mining framework FORSIED that we build upon [4,5].

The FORSIED framework specifies in general terms how to model prior beliefs the user has about the data. Given a *background model* representing these prior beliefs, we may find patterns that are highly surprising to the particular user. Hence in our setting, a tree will generally be more interesting if it contains, according to the user's beliefs, more unexpected relationships between the nodes.

This paper contributes the following:

- We define the new problem of finding subjectively interesting trees connecting a set of query nodes in a network. (Sect. 2)
- We show how to formalize a user's knowledge that the graph has a 'DAG'-like structure, for example because the nodes represent events in time. (Sect. 3)
- We propose heuristics for mining the most interesting trees efficiently in the case of directed graphs. (Sect. 4)
- We evaluate and compare the effectiveness of these heuristics on real data and study the utility of the resulting trees, showing that the results are truly and usefully dependent on the assumed prior beliefs of the user. (Sect. 5)

2 Subjectively Interesting Trees in Graphs

We denote a network (aka graph) G as $G = (V, E)$, where V is the set of nodes (aka vertices) and $E \subseteq V \times V$ is the edge set. We denote the adjacency matrix of a graph as $\mathbf{A}$, where $\mathbf{A}_{ij} = 1$ iff there is an edge connecting node i to j, i.e., iff $(i, j) \in E$. The main focus of this paper will be on directed networks. However, our methods directly apply to undirected networks, when considered as a special case where $\mathbf{A}$ is symmetric. We assume that the set of nodes V is fixed and known, and the user is interested the network's connectivity, i.e., the edge set E, especially in relation to a set of so-called *query nodes* $Q \subseteq V$.

2.1 Trees Connecting Query Nodes as Data Mining Patterns

The data mining process we consider is query-driven: the user provides a set of query nodes $Q \subseteq V$ between which they suspect connections exist in the graph that might be of interest to them. In response to this query, the methods proposed in this paper will thus provide the user with a tree-structured sub-network connecting the query nodes. We consider trees because they are easy to interpret. We refer to the presence of a tree as a *pattern* found in the network.

Formally, a tree $T = (V_T, E_T)$ is a network over the nodes $V_T \subseteq V$ with edges $E_T = \{e_1, \ldots, e_{|V_T|-1}\} \subseteq V_T \times V_T$, where $e_i(2) \neq e_j(2)$ for $i \neq j$ (i.e., each node has only one parent). The tree $T = (V_T, E_T)$ is said to be present in the network $G = (V, E)$ iff $V_T \subseteq V$ and $E_T \subseteq E$. The methods proposed below search for interesting trees $T = (V_T, E_T)$ present in the network $G = (V, E)$ with $Q \subseteq V_T$.

Remark 1. *The above description is a special type of tree: a* rooted arborescence. *This is a tree-structured directed sub-network with a unique directed path between the root and each of the leaves. The edges all point away from the root (*out-arborescence*), but by reversing all edge directions also* in-arborescences *can be considered. We will simply refer to the considered patterns as trees.*

2.2 Subjective Interestingness

The FORSIED framework aims to quantify interestingness of a pattern in a subjective manner, dependent on prior beliefs the user holds about the data. To

model the user's belief state about the data, the framework proposes to use a so-called *background distribution*, which is a probability distribution P over the data space (in our setting, the set of all possible edge sets E). It was argued that a good choice for the background distribution is the maximum entropy distribution subject to the prior beliefs as constraints [4,5].

The FORSIED framework then prefers patterns that achieve a trade-off between how much information the pattern conveys to the user (considering their belief state), versus the effort required of the user to assimilate the pattern. Specifically, De Bie [4] argued that the *Subjective Interestingness (SI)* of a pattern can be quantified as *the ratio of the Information Content (IC) and the Description Length (DL)* of a pattern. The IC is defined as the negative log probability of the pattern w.r.t. the background distribution P. The DL is quantified as the length of the code needed to communicate the pattern to the user.

The IC of a tree. The background distributions P for all prior belief types discussed in this paper have the property that P factorizes as a product of independent Bernoulli distributions[1], one for each possible edge $e \in V \times V$. Hence the IC of a tree T with edges E_T decomposes as

$$\mathrm{IC}(T) = -\log\Big(\prod_{e \in E_T} \Pr(e)\Big) = \sum_{e \in E_T} \mathrm{IC}(e), \tag{1}$$

where we defined the IC of an edge e to be $\mathrm{IC}(e) = -\log(\Pr(e))$.

The DL of a tree. A tree can be described by first describing the set of nodes V_T and then the set of edges E_T over this set of nodes. To describe the set $V_T \subseteq V$ efficiently, note that $Q \subseteq V_T$ such that only $V_T \backslash Q$ needs to be described. This can be done using a sequence of $|V_T| - |Q| + 1$ symbols from $V \backslash Q \cup \{\text{'stop'}\}$, where the last one is a stop symbol. This results in a description length of $(|V_T| - |Q| + 1)\log(|V| - |Q| + 1)$ bits for V_T. Given V_T, E_T can be described by listing the parents of all nodes from within $V_T \cup \{\text{'none'}\}$, where the 'none' symbol is used for the root. This requires $|V_T|\log(|V_T| + 1)$ bits. Thus:

$$\mathrm{DL}(T) = (|V_T| - |Q| + 1)\log(|V| - |Q| + 1) + |V_T|\log(|V_T| + 1). \tag{2}$$

2.3 Finding Subjectively Interesting Trees

The methods presented in this paper aim to solve the following problem:

Problem 1. *Given a graph $G = (V, E)$ and set of query nodes $Q \subseteq V$, we want to find a root $r \in V$ and an out-arborescence rooted at r, such that the arborescence is maximally subjectively interesting. We additionally require that all leaf nodes are query nodes, and we constrain the height of the tree not to be larger than a user-defined parameter k.*

[1] This just happens to be true for the studied prior beliefs. This may indeed reduce computational complexity and it surely reduces the complexity of exposition.

Since the SI depends on the background distribution and thus on the user's prior beliefs, the optimal solution to Problem 1 does as well. As stated in Remark 1, by transposing the adjecency matrix $\mathbf{A}$, we can equivalently consider in-arborescences in exactly the same manner.

3 The Background Distribution to Model the User Beliefs

As mentioned, the background distribution is computed as the maximum entropy distribution subject to the prior beliefs as constraints. Here we discuss how this is done in detail for three types of prior beliefs: (1) on the overall edge density; (2) on the individual node degrees; and () for networks with nodes that correspond to timed events, on the tendency of nodes to be connected to nodes *at a specified time difference* (as well as generalization thereof). These prior beliefs can be combined as well. Note that (1) and (2) were introduced before in [6].

3.1 Prior Beliefs on Overall Density, and on Individual Node Degrees

As shown in [6], given prior beliefs on the degrees of the nodes, the maximum entropy distribution factorizes as:

$$P(\mathbf{A}) = \prod_{i,j} \frac{\exp((\lambda_i^r + \lambda_j^c)\mathbf{A}_{ij})}{1 + \exp(\lambda_i^r + \lambda_j^c)},$$

where λ_i^r and λ_j^c are parameters from the resulting optimization problem. [3] showed how these can be computed efficiently. For a prior belief on the overall graph density, every edge probability in the model equals the assumed density.

3.2 Prior Beliefs When Nodes Represent Timed Events

If the nodes in G correspond to events in time, we can partition the nodes into bins according to a time-based criterion. For example, if the nodes are scientific papers in a citation network, we can partition them by publication year. Given these bins, it is possible to express prior beliefs on the number of edges between two bins. This would allow one to express e.g. beliefs on how often papers from year x cite papers from year y. This is useful e.g. if one believes that papers cite recent papers more often than older ones.

We consider the case when our beliefs are in line with a *stationarity* property, i.e. when the beliefs regarding two bins are independent of the absolute value of the time-based criterion of these two bins, but rather only depend on the time difference. Given an adjacency matrix $\mathbf{A}$, this amounts to expressing prior beliefs on the total number of ones in each of the block-diagonals of the resulting block matrix (formed by partitioning the elements into bins), see Fig. 2 for clarification.

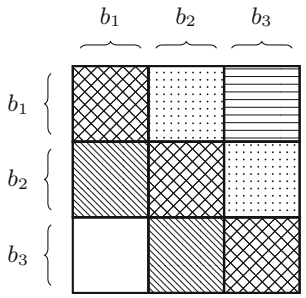

Fig. 2. A resulting block matrix with 3 bins b_1, b_2 and b_3. There are 5 block-diagonals D_k (indicated by the same fill). For each D_k, we express prior beliefs on the sum of all elements in D_k.

We consider the problem of finding the maximum entropy distribution over the set of rectangular binary matrices $\mathcal{A} = \{0,1\}^{n\times n}$, while constraining the expectation of the sum of the elements in each of the block diagonals, as well each of the row and column sums. It is found by solving:

$$\begin{aligned}
&\arg\max_{P(\mathbf{A})} -\sum_{\mathbf{A}\in\mathcal{A}} P(\mathbf{A})\log P(\mathbf{A}),\\
&\quad\text{s.t.}\sum_{\mathbf{A}\in\mathcal{A}} P(\mathbf{A})\sum_{j=1}^{n}\mathbf{A}_{ij} = d_i^r, \quad \sum_{\mathbf{A}\in\mathcal{A}} P(\mathbf{A})\sum_{i=1}^{n}\mathbf{A}_{ij} = d_j^c,\\
&\quad\sum_{\mathbf{A}\in\mathcal{A}} P(\mathbf{A})\sum_{(i,j)\in D_k}\mathbf{A}_{ij} = B_k,\\
&\quad\sum_{\mathbf{A}\in\mathcal{A}} P(\mathbf{A}) = 1,
\end{aligned}$$

with $i,j \in \{1,\ldots,n\}$ and $k \in \{1,\ldots,2\#\text{bins}-1\}$, and with d_i^r the expected sum of the i'th row, d_j^c the expected sum of the j'th column, and B_k the expected sum of the k'th block diagonal D_k. The resulting maximum entropy distribution factorizes as a product of independent Bernoulli distributions, one for each random variable $\mathbf{A}_{ij} \in \{0,1\}$:

$$P(\mathbf{A}) = \prod_{i,j}\frac{\exp((\lambda_i^r+\lambda_j^c+\alpha_k)\mathbf{A}_{ij})}{1+\exp(\lambda_i^r+\lambda_j^c+\alpha_k)}, \tag{3}$$

where λ_i^r, λ_j^c and α_k are the Lagrange multipliers for the corresponding row, column and block-diagonal constraints. These Lagrange multipliers are found by minimizing the Lagrange dual function, as given by:

$$L(\lambda^r,\lambda^c,\alpha) = \sum_{i,j}\log(1+\exp(\lambda_i^r+\lambda_j^c+\alpha_k)) - \sum_i \lambda_i^r d_i^r - \sum_j \lambda_j^c d_j^c - \sum_k \alpha_k B_k.$$

Standard methods for unconstrained convex optimization such as Newton's method can be used to infer the optimal values. The number of variables to be optimized over is equal to $2(\mathrm{n} + \#\mathrm{bins}) - 1$, where $1 \leq \#\mathrm{bins} \leq \mathrm{n}$. Using Newton's method then requires solving a linear system of $O(n)$ equations, with computational complexity $O(n^3)$. For practical problems involving large networks, this quickly becomes infeasable. However, with a similar argument as in [3], we can dramatically reduce the number of variables. Observe that if $d_k^r = d_l^r$ and k and l belong to the same bin, then we have $L(\ldots, \lambda_k^r, \lambda_l^r, \ldots) = L(\ldots, \lambda_l^r, \lambda_k^r, \ldots)$. The convexity of L implies $\lambda_k^r = \lambda_l^r$ at the optimum. A similar argument holds for the λ^c parameters. Thus the number of free variables per bin to be optimized over, is bounded by the number of distinct row and column sums *per bin*.

Let $\widetilde{m}$ be the total number of free row variables, and $\widetilde{n}$ be the total number of free column variables. The following Lemma provides an upper bound on $\widetilde{m} + \widetilde{n}$ in terms of the number of non-zero elements of $\mathbf{A}$ and the number of bins k:

Lemma 1. *Let $\boldsymbol{A}$ be a binary rectangular matrix and denote $s = \sum_{i,j} \boldsymbol{A}$. Then it holds that $\widetilde{m} + \widetilde{n} \leq 2\sqrt{2ks}$.*

Proof. Let $\widetilde{m}_i$ be the number of distinct row variables in the i-th bin and similarly for $\widetilde{n}_i$ with $i \in \{1, \ldots, k\}$. Let s_i (s_i') be the total number of ones in all the rows (columns) of the elements in bin i. Then the following inequalities hold [3]:

$$\widetilde{m}_i \leq \sqrt{2s_i}, \quad \text{and} \quad \widetilde{n}_i \leq \sqrt{2s_i'}.$$

Hence $\widetilde{m} + \widetilde{n} \leq \sqrt{2}(\sqrt{s_1} + \ldots + \sqrt{s_k'})$. Clearly also $\sum_i s_i + s_i' = 2s$ and thus by Jensen's inequality $\sqrt{s_1} + \ldots + \sqrt{s_k'} \leq 2\sqrt{ks}$, which proves the lemma. □

Denote $\widetilde{\lambda}_{k,l}^r$ as the l-th unique row parameter in the k-th bin. Denote the corresponding row sum constraint as $\widetilde{d}_{k,l}^r$ having $\widetilde{m}_l^k$ occurences in that bin. Similarly for $\widetilde{\lambda}_{k,l}^c$, $\widetilde{d}_{k,l}^c$ and $\widetilde{n}_l^k$. Denote $\alpha_{kk'}$ as the α parameter of the $\mathbf{A}_{ij}$ elements with $i \in$ bin k and $j \in$ bin k$'$. The reduced Lagrange dual function then becomes

$$L(\widetilde{\lambda^r}, \widetilde{\lambda^c}, \alpha) = \sum_k \sum_{k'} \sum_l \sum_{l'} \widetilde{m}_l^k \widetilde{n}_{l'}^{k'} \log(1 + \exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_{kk'}))$$
$$- \sum_k \sum_l \widetilde{m}_l^k \widetilde{d}_{k,l}^r \widetilde{\lambda}_{k,l}^r - \sum_{k'} \sum_{l'} \widetilde{n}_{l'}^{k'} \widetilde{d}_{k',l'}^c \widetilde{\lambda}_{k',l'}^c - \sum_m \alpha_m B_m.$$

The gradient is computed as

$$\frac{\partial L}{\partial \widetilde{\lambda}_{k,l}^r} = \sum_{k'} \sum_{l'} \widetilde{m}_l^k \widetilde{n}_{l'}^{k'} \frac{\exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_{kk'})}{1 + \exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_{kk'})} - \widetilde{m}_l^k \widetilde{d}_{k,l}^r, \tag{4}$$

$$\frac{\partial L}{\partial \widetilde{\lambda}_{k',l'}^c} = \sum_k \sum_l \widetilde{m}_l^k \widetilde{n}_{l'}^{k'} \frac{\exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_{kk'})}{1 + \exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_{kk'})} - \widetilde{n}_{l'}^{k'} \widetilde{d}_{k',l'}^c, \tag{5}$$

$$\frac{\partial L}{\partial \alpha_k} = \sum_{D_k} \frac{\exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_k)}{1 + \exp(\widetilde{\lambda}_{k,l}^r + \widetilde{\lambda}_{k',l'}^c + \alpha_k)} - B_k. \tag{6}$$

and a similar expression for the Hessian. In all cases (rows, columns and block diagonal) the corresponding gradient is simply the difference between the expected number of ones and the corresponding parameter as given by the constraints. When applying Newton's method to the reduced model, we need $O(\widetilde{m}\widetilde{n})$ calculations to compute both the gradient and Hessian. After that we need to solve a linear system with $\widetilde{m} + \widetilde{n} + 2k - 1$ equations, with cubic complexity. By Lemma 1, this is $O(\sqrt{k^3 s^3} + k^3)$, making it very efficient in many real life applications (sparse networks and a small number of bins).

Remark 2. *Note that we are not limited to the case of stationarity, nor is it necessary that nodes correspond to timed events. Expressing prior beliefs on the density of* any particular subset of edges *is possible in a similar manner. We tackled this specific case because it directly applies to the data used in this paper.*

4 Algorithms for Finding the Most Interesting Tree

The problem of finding a directed Steiner arborescence (spanning all the query nodes) with maximum SI is NP-hard in general, as can be seen from the case of constant edge weights (e.g., if the prior belief is the overall graph density). In this case the SI of a tree will be a decreasing function of the number of nodes in the tree. Hence the problem is equivalent to the minimum Steiner arborescence problem, with constant edge weights, which is NP-hard. For nonconstant background models it will be a trade-off between the IC and the DL of a tree. In most cases, we are looking for small trees with highly informative edges.

There are a number of algorithms that provide good approximation bounds for the directed Steiner problem [2,7,11], and this problem has also been studied recently in the data mining community, e.g., [1,8]. However, Problem 1 is equivalent to the Steiner problem in the case of a uniform background distribution, i.e., when the IC of the edges is constant and hence irrelevant. In general, we aim to solve a *maximization* problem, while Steiner tree problems aim to *minimize* the cost of the tree. For this reason we propose fast heuristics for large graphs, that perform well on different kinds of background distributions.

A Python implementation of the algorithms and the experiments is available at http://www.interesting-patterns.net/forsied/sict/.

4.1 Proposed Heuristics

Our proposed methods all work in a similar way. We apply a preprocessing step, resulting in a set of candidate roots. Given a candidate root r, we build the tree by iteratively adding edges (parents) to the *frontier*—initialized as $Q \backslash \{r\}$—, until *frontier* is empty. We exhaustively search over all candidate roots and select the best resulting tree. The heuristics differ in the way they select allowable edges. The outline of SteinerBestEdge is given in Algorithm 1.

Preprocessing. All of the proposed heuristics have two common preprocessing steps. First we find the common roots of the nodes in Q up to a certain level k,

meaning we look for nodes r, s.t. $\forall q \in Q : \text{SPL}(q, r) \leq k$, with $\text{SPL}(\cdot)$ denoting the shortest path length. This can be done using a BFS expansion on the nodes in Q until the threshold level k is reached. Note that query nodes are also potential candidates for being the root, if they satisfy the above requirement.

Secondly, for each r we create a subgraph $H \subset G$, consisting of all simple paths $q \rightsquigarrow r$ with $\text{SPL}(q, r) \leq k$, for all $q \in Q$. This can be done using a modified DFS-search. The number of simple paths can be large. However, we can prune the search space by only visiting nodes that we encountered in the BFS expansion, making the construction of H quite efficient for small k.

SteinerBestEdge. Given the subgraph H, we construct the arborescence working from the query nodes up to the root. We initialize the frontier as $Q \setminus \{r\}$, and iteratively add the best feasible edge to a partial solution, denoted as *Steiner*, according to a greedy criterion. The greedy criterion is based on the ratio of the IC of that edge to the DL[2] of the partial *Steiner* that would result from adding that edge. This heuristic prefers to pick edges from a parent node that is already in *Steiner*, yielding a more compressed tree and thus a smaller DL.

Algorithm 2 checks if an edge is *feasible* by propagating its potential influence to all the other nodes in H. The check can fail in two ways. First, the addition of an edge could yield a *Steiner* tree with height $> k$, see Fig. 3 for an example. Secondly, the addition of an edge may lead to cycles in *Steiner*. Cycles are avoided by only considering edges (s, t) that do not potentially change $\text{SPL}(t, r)$. If $\text{SPL}(t, r)$ would change, the shortest path –given the current *Steiner*– from s to r is not along the edge (s, t) and hence for all $f \in frontier$ we always have 1 feasible edge to pick (i.e. an edge that is part of a shortest path $f \rightsquigarrow r$). One way to select the best feasible edge is to first sort the edges according to the greedy criterion. Then try the check from Algorithm 2 on this sorted list (starting with the best edge(s)), until the first success, and add the resulting edge to *Steiner*. Algorithm 2 will also return an updated shortest path function *NewSP*, containing all the changes in $\text{SPL}(n, r)$ for $n \in H$ due to the addition of that edge to *Steiner*. After performing the necessary updates on the *SP* function, and the *frontier*, *parents* and *level* sets, we continue to iterate until *frontier* is empty.

SteinerBestIC. Instead of adding 1 edge at a time, this heuristic adds multiple edges at once. We look for the parent node that (potentially) adds the most total information content of allowable edges to the current *Steiner*. However, given such a parent node, it not always possible to add multiple edges, see Fig. 4. Instead we sort the edges coming from such a parent node according to their IC, and iteratively try to add the next best edge to *Steiner*.

SteinerBestIR. A natural extension of SteinerBestIC is to actually take in account the DL of the partial *Steiner* solution, as we did in SteinerBestEdge.

[2] Note that during construction, the partial solution *Steiner* is often a forest. However, we compute the DL as if it was an equally sized tree. This makes sense because the end result will in fact be a tree, and we are optimizing towards the IR of that tree.

Algorithm 1. SteinerBestEdge(subgraph H, root r, queryset Q, maxlevel k)

```
1: Steiner ← Q ∪ {r}
2: ∀x ∈ H : SP(x) ← ShortestPathLength(r, x)
3: frontier ← Q.remove(r)
4: level(frontier) ← 0
5: parents ← parents(frontier)
6: while frontier do
7:     n ← length(Steiner), ranking ← ∅
8:     for all edges from parents to frontier do
9:         if parent ∈ Steiner then
10:            ranking.add({edge, IC(edge)/DLTree(n)})
11:        else
12:            ranking.add({edge, IC(edge)/DLTree(n + 1)})
13:    ranking ← sort(ranking)
14:    for edge ∈ ranking do
15:        if CheckChildren(H, edge, Steiner, k, SP) is True then
16:            update(Steiner, frontier, parents, level, SP)
17:            Quit loop
18: return Steiner
```

SteinerBestIR favors parent nodes that are already in *Steiner*, steering towards an even more compressed tree.

SteinerBestEdgeBestIR. Our last method simply picks the single best edge coming from the best parent, where the best parent is determined by the same criteria as in SteinerBestIR. In general this will pick a locally less optimal edge than SteinerBestEdge, but it will pick edges from a parent node that has lots of potential to the current *Steiner* solution.

Correctness of the solutions. The following theorem states that all the heuristics indeed result in a tree with maximal height $\leq k$.

Theorem 1. *Given a non-empty query set Q, a candidate root r and a height $k \geq 1$. In all cases all four heuristics will return a tree with height $\leq k$.*

Proof. In all cases the proposed heuristics return a tree rooted at r with height $\leq k$. We call a partial forest solution *Steiner* valid, if for all leaf nodes $l \in Steiner : \text{SPL}(l, r|Steiner) \leq k$, where $SPL(\cdot|Steiner)$ denotes a shortest path length given the partial *Steiner* solution. Note that the initial *Steiner* is valid, due to the way the subgraph H was constructed. It is always possible to go from one valid *Steiner* solution to another valid one, by selecting an edge (incident to a frontier node) along a shortest path—given we have *Steiner*—from r that frontier node. This will result in an unchanged SPL for all other nodes (in particular the leaf nodes), and hence remains a valid *Steiner*. If we have n frontier nodes, we have at least n such valid edges to pick from. Hence, all of the heuristics have at least $n \geq 1$ valid edges to pick from. The process of adding edges is finite, and will eventually result in an arborescence rooted at r with height $\leq k$.

Algorithm 2. CheckChildren($H, edge, Steiner, k, SP$)

```
1: frontier ← frontier(Steiner), level ← level(Steiner), (source, target) ← edge
2: NewSP ← {source : SP(target) + 1}
3: if NewSP(source) = SP(target) then return True
4: else if NewSP(source) + level(source) > k then return False
5: else
6:     children ← children(source)
7: while children do
8:     nextChildren ← ∅
9:     for c ∈ children do
10:        if c ∉ Steiner \ frontier then
11:            updatedP = parents(c) ∩ NewSP
12:            otherP = parents(c) \ updatedP
13:            cand ← min(NewSP(p) : p ∈ updatedP) + min(SP(p) : p ∈ otherP) + 1
14:            if cand > SP(c) then
15:                NewSP(c) = cand
16:                if c ∈ query and NewSP(c) > k then return False
17:                if c is target then return False          ▷ Possible cycle avoided
18:                nextChildren.add(children(c))
19:        else
20:            NewSP(c) = NewSP(SteinerParent(c)) + 1
21:            if c is target then return False              ▷ Possible cycle avoided
22:            nextChildren.add(children(c))
23:    children ← nextChildren
24: return True, NewSP
```

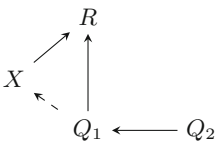

Fig. 3. Example of why look-ahead is needed to ensure the returned tree has depth as most k. If $k = 2$, the only valid tree is $(Q_1, R)(Q_2, Q_1)$. Initially, the *frontier* is $\{Q_1, Q_2\}$ and X is a candidate parent for Q_1 because there is a path $Q \rightsquigarrow R$ of at most length 2. Yet, adding the dashed edge violates the shortest path constraint for Q_2.

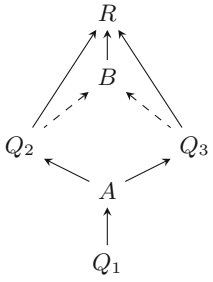

Fig. 4. Example of why look-ahead is needed for *sets of edges*. For $k = 3$, neither of the two dashed edges violate the depth constraint—they are part of a valid tree—, but together they indirectly violate the shortest path constraint for Q_1. Regardless of which parent is chosen for A, the path from Q_1 to R has length 4.

5 Experiments

In this section we empirically evaluate our proposed methods on real data. All experiments are based on the ACM-Citation-network v8[3], a scientific paper citation network. This (directed) network contains 2,381,688 papers and 10,476,564 citations. The oldest paper is a seminal paper of C.E. Shannon from 1938. The most recent papers are from 2016. We will use the acronyms s-E, s-IC, s-IR and s-EIR for resp. SteinerBestEdge, SteinerBestIC, SteinerBestIR and SteinerBestEdgeBestIR. First, we evaluate and compare the performance of the heuristics.

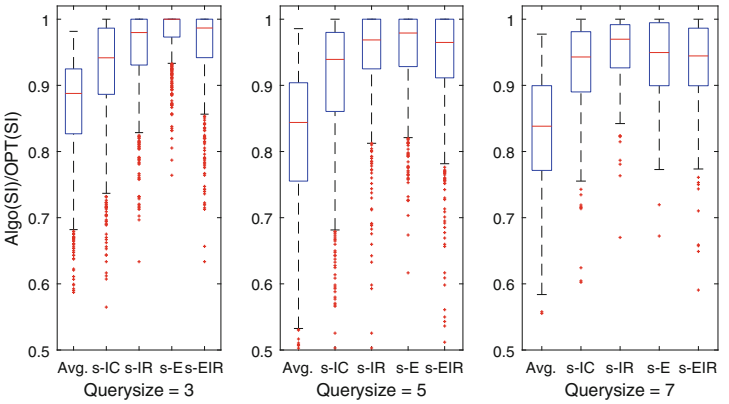

Fig. 5. The interestingness of the heuristics (relative to the optimal interestingness) versus querysize. We also compare with the average interestingness over all trees. Note the decrease in performance of s-E for larger querysizes.

5.1 Comparing the Heuristics

To compare the performance of the heuristics we set up an experiment similar to [1]. We fitted the background model with prior beliefs on the degrees of the network. To generate a set of n query nodes we used a snowball-like sampling scheme. We randomly selected an initial node in the graph. Then, we explore $n' < n$ of its neighbors, each selected with probability s. For each of these nodes we continue to test n' of its neighbors until we have n selected nodes. From this query set we randomly select a valid common root within a maximum distance k. To have a baseline, we find the arborescence with maximal SI using exhaustive search. To keep this comparison feasible, we only consider cases where the number of trees is <200,000. For querysizes $= \{3, 5\}$ we generated 1000 query sets, for querysize $= 7$ we have done 250 sets. In all cases k was limited to 3, the beamwidth n' was chosen to be 2 and sampling rates $s \in \{0.1, \ldots, 0.9\}$.

Figure 5 shows a boxplot of the interestingness scores of the tree-building heuristics (relative scores to the optimal arborescence interestingness) versus

[3] https://aminer.org/citation, [10].

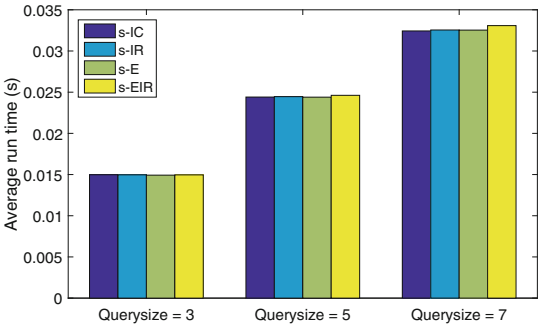

Fig. 6. Average run time of the heuristics. The main bottleneck is finding all the simple paths, which is included here. Hence, the run time differences are small.

query size. All four heuristics clearly are better strategies than randomly selecting an arborescence (the Avg. case). s-IR outperforms s-IC in all cases, which makes sense because s-IC has no regard for the DL of the tree. s-E performs comparatively worse for larger query sizes, and s-IR seems to be the best option for larger querysizes. This result is not definite, it could be due to the fact that we fixed the height at $k = 3$. However, while not reported here, we observed that s-IR is also the best option for larger query sizes and larger k. We also tested the effect of the sampling rate (not shown), and this appeared to affect neither the SI of the resulting trees, nor the ranking of the algorithms.

Figure 6 shows the average run time of our methods. The run time of the heuristics are all negligable compared to the time needed to find all simple paths from the root to the query nodes (see Sect. 4), which in all cases takes up more than 90% of the total running time. We conclude that with prior beliefs on the individual node degrees, s-IR seems to be best option for larger queries, while for smaller queries s-E seems to give a more interesting tree.

5.2 The Effect of Different Prior Beliefs, and a Subjective Evaluation

Here we evaluate the outcome of our heuristics w.r.t. three different kinds of prior beliefs on the ACM citation network. The first prior belief is on the overall graph density. In this case, every edge has the same probability in the background model and hence same information content. The optimal arborescence then is the smallest Steiner arborescence. The second set of prior beliefs is on the individual degree of each node. As for a citation network the number of citations a paper has is easier to estimate than the number of references of that paper (without reading it), we only constrained the expected in-degree of each node. As a result, edges to highly cited nodes are more probable, and a tree will be more interesting if it is not only small, but has a preference for less frequently cited papers. The final type of prior belief is on both the individual in-degree of each node, as well as the dependency of citation probabilities on the difference in publication date.

Table 1. Average number of common authors per edge in the tree from algorithm s-IR for different types of prior beliefs and query sizes. p-values for the Wilcoxon signed-rank test (pairwise comparison) of each type of prior with the prior on individual degrees, shown between brackets. The second column lists the time to fit the background model on the full data.

Prior beliefs	Time (s)	No. of authors, $\|Q\| = 5$	No. of authors, $\|Q\| = 7$	No. of authors, $\|Q\| = 9$
Overall density	-	2.71 (0.0035)	2.94 (0.005)	3.66 (0.0044)
Indiv. degrees	22	3.17	3.34	4.09
Bin every 3 years	380	2.92 (0.007)	3.12 (0.0417)	3.9 (0.0996)
Bin every 5 years	254	2.80 (5.63e−04)	2.84 (9.05e−06)	3.71 (0.0049)

In Sect. 3.2 we showed how to formalize prior beliefs on diagonal block sums. Here it is natural to group papers together according to their publication year (or per 2 years, 5 years, ...). In this way, it is possible to incorporate prior beliefs such as: "The number of papers from year X citing a paper from year $X - 3$ is high". In general, an edge will have a high probability if the corresponding expected block diagonal sum is high, see Eqs. (3) and (6). Note that the citation network should (in theory) be a directed acyclic graph, since no paper can cite a paper with a higher publication year. Yet the data contains 66,772 (<0.01%) violating edges, which our method handles gracefully.

Common authors as external validation. In many scientific fields, self-citations are common practice. We expect the trees to reflect this to differing degrees, depending on the prior beliefs taken into account. To test this, we set up an experiment similar as in Sect. 5.1. The queries are generated in the same way, but with a preference for queries that have some authors in common. If a paper has an author in common with the current query set, it is automatically chosen instead of being sampled with probability s. We generated 200 random queries for each querysize $\{5, 7, 9\}$, with max. height $k = 4$. For each query, we look at the tree generated by s-IR, computed for 4 different types of prior beliefs. Our measure is the total number of common authors per edge in the tree.

Table 1 shows the results for 4 types of prior beliefs. There is a substantial difference between the first and second prior. This makes sense because with a constant background model, s-IR is indifferent to the number of citations of papers. With the second prior, s-IR prefers nodes with fewer citations, penalizing highly cited papers. This means we are also favoring self-citations a bit more, since chances are high that nodes encountered in our experiment do not have authors in common for references to seminal papers. Secondly, there are differences in the self-citation rate between a prior on the time relations (priors 3 and 4) versus prior 2. Most people stop publishing after their PhD, but during that time they will have some references to their own papers. Hence with a background model of type 3 and 4, s-IR will prefer citations between papers with a high difference in publication year, making self-citations less common.

Subjective evaluation. We queried three recent KDD best paper award winners that were present in the network, see Figs. 1, 7 and 8 for results for different prior beliefs. We used $k = 3$, resulting in 33 candidate roots. Notice the number of citations and the publication year of the root in each of the resulting trees, confirming our expectations of the influence of the prior beliefs on the SI of trees.

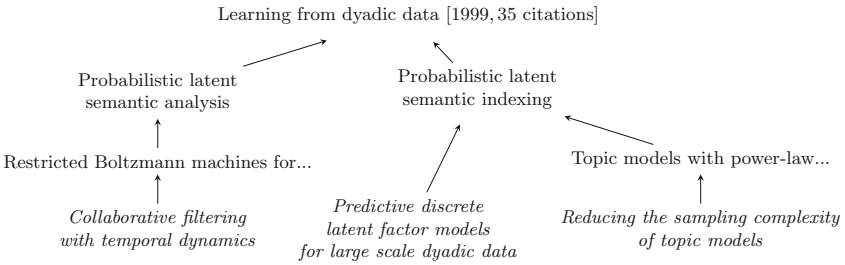

Fig. 7. Like Fig. 1, with as prior knowledge the degree of each node.

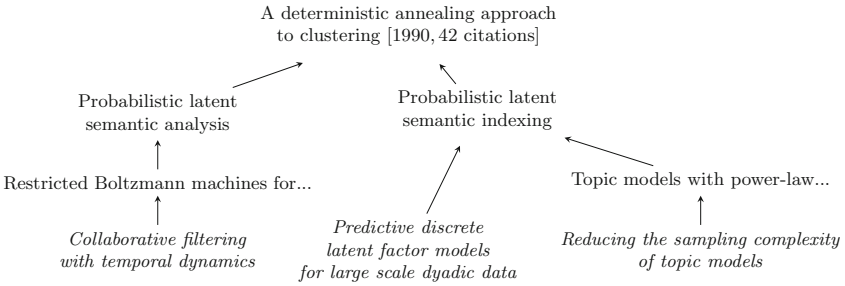

Fig. 8. Like Fig. 1, with as prior knowledge degrees and time constraints.

6 Discussion and Related Work

We studied the problem of finding interesting trees that connect a user-provided set of query nodes in a large network. This is useful for example to, based on citation data, find papers that (indirectly) influenced a set of query papers, perhaps to understand the structure of an organization from communication records, and in many other settings. We defined the problem of finding such trees as an optimization problem to find an optimal balance between the informativeness (the Information Content) and conciseness (the Description Length) of a tree. Additionally, by encoding the prior beliefs of a user, we propose how to find results that are surprising and interesting to a specific user.

We have introduced a general algorithmic strategy to construct such trees along with four heuristics of varying complexity. We have introduced a tractable model to include prior knowledge about the density of sub-networks and more specifically for the case where the nodes appear in time blocks and the probability of edges is expected to be a function of time. Finally, we evaluated the interestingness of the results in several experiments, both subjectively and using external criteria, plus we empirically compared the quality and computational efficiency of the four heuristics.

The computational problem solved in this paper is related to the problem of constructing a minimal Steiner arborescence (aka directed Steiner tree). There is a long development of approximation algorithms, e.g., [2,7,11]. Faster special-purpose approximations have also been studied in the data mining community, e.g., for temporal networks [8]. The most related algorithmic results are those of Akoglu et al. [1], who study the problem of finding a good partitioning and connection structure within each part on undirected graphs for a given set of query nodes. Although their purpose is to explore an undirected graph, they map the problem to graph partitioning plus finding Steiner arborescences.

It should be noted that Problem 1 is *not* equivalent to the Steiner arborescence problem, because in general the subjective interestingness of a tree does not factorize as a sum over the edges. Hence, we do not expect any existing algorithm to solve this problem well.

We are currently working on applications in biology as well as social media.

Acknowledgements. This work has been supported by the European Research Council under the EU's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement no. 615517, the FWO (project nos. G091017N, G0F9816N), and the European Union's Horizon 2020 research and innovation programme and the FWO under the Marie Sklodowska-Curie Grant Agreement no. 665501.

References

1. Akoglu, L., Chau, D.H., Faloutsos, C., Tatti, N., Tong, H., Vreeken, J.: Mining connection pathways for marked nodes in large graphs. In: Proceedings of SDM, pp. 37–45 (2013)
2. Charikar, M., Chekuri, C., Cheung, T.Y., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed Steiner problems. In: Proceedings of SODA, pp. 192–200 (1998)
3. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min. Knowl. Disc. **23**(3), 407–446 (2011)
4. De Bie, T.: An information theoretic framework for data mining. In: Proceedings of KDD, pp. 564–572 (2011)
5. De Bie, T.: Subjective interestingness in exploratory data mining. In: Proceedings of IDA, pp. 19–31 (2013)
6. van Leeuwen, M., De Bie, T., Spyropoulou, E., Mesnage, C.: Subjective interestingness of subgraph patterns. Mach. Learn. **105**(1), 41–75 (2016)
7. Melkonian, V.: New primal-dual algorithms for Steiner tree problems. Comput. Oper. Res. **34**(7), 2147–2167 (2007)

8. Rozenshtein, P., Gionis, A., Prakash, B.A., Vreeken, J.: Reconstructing an epidemic over time. In: Proceedings of KDD, pp. 1835–1844 (2016)
9. Silberschatz, A., Tuzhilin, A.: On subjective measures of interestingness in knowledge discovery. In: Proceedings of KDD, pp. 275–281 (1996)
10. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: Proceedings of KDD, pp. 990–998 (2008)
11. Watel, D., Weisser, M.A.: A practical greedy approximation for the directed Steiner tree problem. In: Proceedings of COCOA, pp. 200–215 (2014)