To appear in the *Journal of Experimental & Theoretical Artificial Intelligence*
Vol. 00, No. 00, Month 20XX, 1–34

RESEARCH ARTICLE

Summarization of Weighted Networks

Fang Zhou[a], Qiang Qu[b] and Hannu Toivonen[c*]

[a] *Center for Data Analytics and Biomedical Informatics, Temple University, USA.*
[b] *Department of Computer Science, Innopolis University, Russia.*
[c] *Department of Computer Science and HIIT, University of Helsinki, Finland.*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

Networks often contain implicit structure. We introduce novel problems and methods that look for structure in networks, by grouping nodes into supernodes and edges to superedges, and then make this structure visible to the user in a smaller generalized network. This task of finding generalizations of nodes and edges is formulated as 'network summarization'. We propose models and algorithms for networks that have weights on edges, on nodes, or on both, and study three new variants of the network summarization problem. In *edge-based weighted network summarization*, the summarized network should preserve edge weights as well as possible. A wider class of settings is considered in *path-based weighted network summarization*, where the resulting summarized network should preserve longer-range connectivities between nodes. *Node-based weighted network summarization* in turn allows weights also on nodes and summarization aims to preserve more information related to high weight nodes. We study theoretical properties of these problems and show them to be NP-hard. We propose a range of heuristic generalization algorithms with different trade-offs between complexity and quality of the result. Comprehensive experiments on real data show that weighted networks can be summarized efficiently with relatively little error.

Keywords: Weighted Networks, Network Mining, Generalization, Network Summarization.

1. Introduction

Networks are widely used to model various types of interactions or relationships between entities in a myriad of applications, such as social interactions between persons (e.g. Kimura, Saito, Nakano, and Motoda (2010)), hyperlinks between web pages (Lin, Yu, Han, and Liu (2010)), or interactions between proteins (e.g. Mamitsuka (2012)). In many of them, relationships have weights that are central to any use or analysis of networks: how frequently do two persons communicate, how much does web traffic flow from one page to another, or how strongly does one protein regulate the other one? In addition, each node in a graph or network is typically associated with additional information, such as profiles and tweets of people in social networks (e.g. Gonalves, Perra, and Vespignani (2011)), publications of authors in authorship networks (e.g. Viana, Amancio, and Costa (2013)), functions of a protein in biological networks (e.g. Ota, Gonja, Koike, and Fukuchi (2016)), etc. These may reflect the importances of different nodes: how much does a popular twitter user influence others' opinions, how often are the works of an author cited, or how important is a protein for the organism?

In this paper, we propose novel models and methods for mining such weighted net-

*Corresponding author. Email: hannu.toivonen@cs.helsinki.fi

works (or graphs). The essential knowledge discovery task is to automatically discover generalizations of nodes and edges into supernodes and superedges, respectively, so that the network can be represented using a relatively small number of supernodes and superedges instead of the original components, with little loss of information. The problem was originally introduced only for unweighted networks (Navlakha, Rastogi, and Shrivastava (2008); Tian, Hankins, and Patel (2008)). Here we propose models and methods for weighted networks.

As a small example, consider the co-authorship network in Figure 1(a). It contains an excerpt from the DBLP Computer Science Bibliography[1], a subgraph containing *Jiawei Han* and *Philip S. Yu* and a dozen related authors. Nodes in this graph represent authors and edges represent co-authorships. Edges are weighted by the number of co-authored articles.

A summary of this network highlights some of the inherent structure or roles in the original network (Figure 1(b)). For instance, *Ke Wang* and *Jianyong Wang* have identical sets of co-authors (in this excerpt from DBLP) and have been generalized into a single supernode together. (This group of nodes would not be found by traditional network clustering methods, since the two nodes are not directly connected.) *Daxin Jiang* and *Aidong Zhang* have been generalized into one supernode, too, but additionally the self-edge of their supernode indicates that they have also authored papers together.

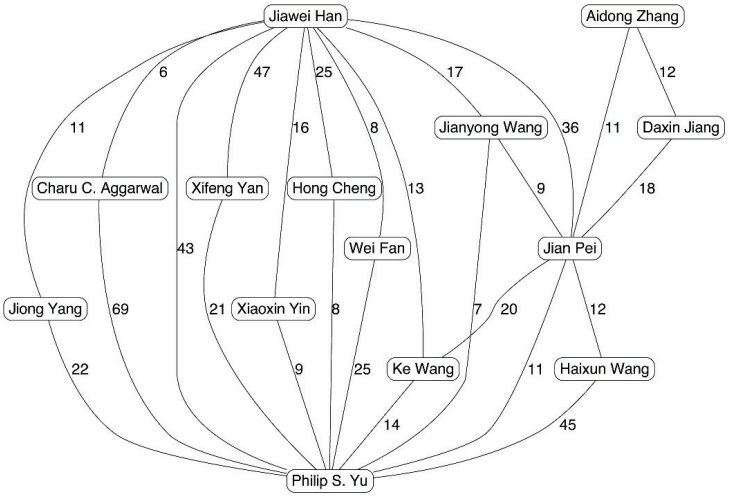
Generalizations that could not be obtained by unweighted summarization algorithms (Navlakha et al. (2008); Tian et al. (2008)) can be observed among the six authors that only connect to *Jiawei Han* and *Philip S. Yu*. Instead of being all grouped together as structurally equivalent nodes, there are three groups that have different edge weight profiles in Figure 1(b). *Charu C. Aggarwal* is a group by himself, strongly connected with *Philip S. Yu*. A second group includes *Jiong Yang*, *Wei Fan*, and *Xifeng Yan*, who are roughly equally strongly connected to both *Jiawei Han* and *Philip S. Yu*. The third group, *Hong Cheng* and *Xiaoxin Yin*, are more strongly connected to *Jiawei Han*. Such groups are not found with methods for unweighted networks.

Next, assume that the nodes in Figure 1(a) are weighted by the impact of the published work of scientists, such as their citation counts[2]. In contrast to Figure 1(b), a different result could be obtained, in which mostly high-weighted nodes and edges are maintained. Such a result is shown in Figure 1(c). The most striking difference is that the nodes of *Daxin Jiang* and *Aidong Zhang* (with node weights 521 and 880, respectively, not shown in the figure), are not present in the summarized network. Instead, more information is maintained about the heavier nodes by making *Xifeng Yan* an individual group, allowing more accurate edge weights not only for him but also for *Jiong Yang* and *Wei Fan*, all in relations to *Jiawei Han* and *Philip S. Yu*, the two heaviest nodes in the network. In this version the edge between *Jiawei Han* and *Charu C. Aggarwal* is also missing, even though both are heavy nodes; the weight of the missing edge is relatively low, however.

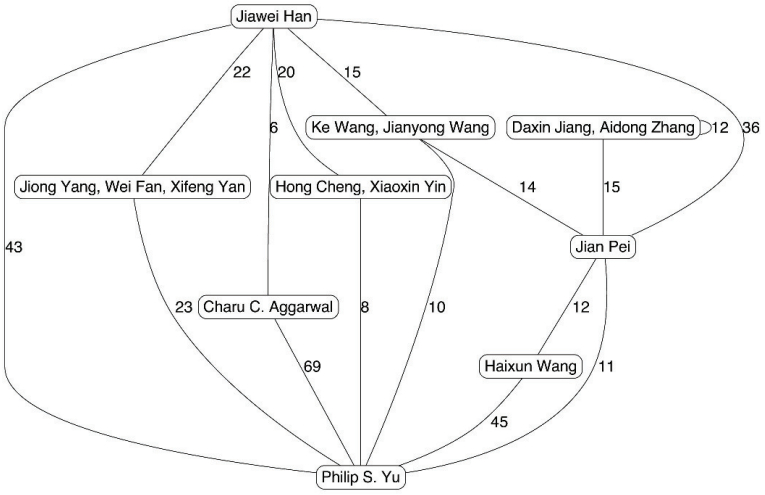
In this paper, the task of finding generalizations of nodes and edges is formulated as three novel variants of the summarization problem. In what we define as the *edge-based weighted network summarization problem*, the goal is to preserve the weights on individual edges as much as possible. However, for many applications on weighted networks it is important to preserve relationships between faraway nodes, too, not just individual edge weights. Motivated by this, we then introduce the *path-based weighted network summarization problem* where the goal is to produce a generalized network that maintains connectivities across the network. In this setting, the quality of the best path between

[1] http://dblp.uni-trier.de/

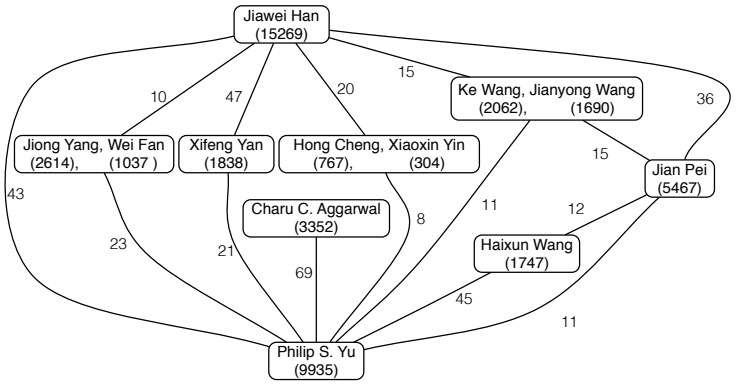
[2] http://dl.acm.org



(a) A neighborhood network of Jiawei Han in the DBLP bibliography.



(b) Generalized co-authorship structure found by edge-based summarization.



(c) Generalized co-authorship structure found by node-based summarization.

Figure 1. (a) A neighborhood network of Jiawei Han in the DBLP bibliography. (b, c) Generalized co-authorship structure found by summarizing the network.

any two nodes in the summarized network should be similar to the one in the original network, but the path does not have to be the same. Furthermore, nodes have varying weights or importances in many applications. Motivated by this, we also present the *node-based weighted network summarization problem* where we preferably preserve more information related to nodes with high weights.

The main contributions of the paper are the following: (1) The edge-based, the path-based, and the node-based weighted network summarization problems are introduced. (2) We analyze the problems and prove their computational hardness. For efficient computation, bounds for the distances between networks are derived. (3) Several algorithms for the weighted network summarization problems are presented, with different time/quality trade-offs. (4) Extensive experimental results on real weighted networks show that generalization of nodes and edges can summarize weighted networks effectively and efficiently. We also find that summarization can have surprisingly little effect on further network operations such as clustering even though some information is lost in the summary.

A preliminary report of parts of this study was published as a conference paper (Toivonen, Zhou, Hartikainen, and Hinkka (2011)). The current paper significantly extends the work and results in all of the points above.

Road-map. The rest of this paper is organized as follows. The weighted network summarization problems are formulated in Section 2. The bounds for network distance are derived in Section 3, and summarization operations that utilize these bounds are presented in Section 4. The proposed algorithms for the weighted network summarization are described in Section 5, and are experimentally evaluated in Section 6. Related work is reviewed in Section 7. Section 8 contains our concluding remarks.

2. Problem Statement

The aim of weighted network summarization is to discover hidden structure in a given network and use it to generalize nodes and edges so as to summarize a given weighted network into a smaller one with little loss of information. Three variants of this problem are defined. We start by defining concepts and notations common to the three variants, and then formalize the problems. This is followed by a subsection on computational complexity analysis. Since the problem is conceptually related to clustering of nodes, and since our approach to solving it is similar to hierarchical agglomerative clustering, we will draw parallels between these problems throughout the text.

2.1. *Preliminaries for Weighted Network Summarization*

Definition 2.1: A *weighted network* is a tuple $G = (V, E, w, \mathcal{I})$, where V is a set of vertices (nodes), $E \subset V \times V$ is a set of edges, $w : E \to \mathbb{R}^+$ assigns a non-negative weight to each edge $e \in E$, and $\mathcal{I} : V \to \mathbb{R}^+$ assigns a non-negative importance to each node $v \in V$. We use notation $G = (V, E, w)$ if all nodes have equal importances, and $G = (V, E)$ if all edge weights are also equal. For further notational convenience, we define $w(u, v) = 0$ if $(u, v) \notin E$.

In this paper, edges are assumed undirected, and in the sequel we use notations such as $\{u, v\} \in V \times V$ in the obvious way. The definitions and algorithms can, however, be easily adapted for the directed setting.

The following definition of a summarized network largely follows the definition of network summarization for the unweighted case (Navlakha et al. (2008)).

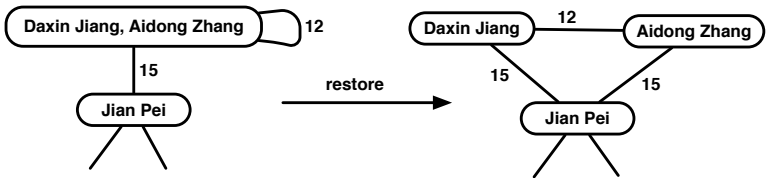


Figure 2. An example of restoring a part of a summarized network.

Definition 2.2: A weighted network $S = (V', E', w')$ is a *summarized representation of* $G = (V, E, w)$ or its *weighted summarized network* if $V' = \{v'_1, \ldots, v'_n\}$, $v'_i \subset V$, and for all $i \neq j$, $v'_i \cap v'_j = \emptyset$. I.e., V' is a partition of the full original set V of nodes or of its subset. The nodes $v' \in V'$ are called *supernodes*. The edges $e' \in E'$ between supernodes v'_i and v'_j are called *superedges*.

While Definition 2.2 essentially talks just about clustering of nodes into supernodes $v'_1, \ldots, v'_n$, Definition 2.3 specifies the role of edges in summarized networks.

Definition 2.3: Given a weighted summarized network $S = (V', E', w')$ as above, the *restored network* $res(S)$ *of* S is a weighted network $res(S) = (V'', E'', w'')$ such that $V'' = \bigcup_{v' \in V'} v'$, $E'' = \bigcup_{\{u', v'\} \in E'} u' \times v'$, and $w''(\{u, v\}) = w'(u', v')$ s.t. $u \in u'$ and $v \in v'$.

In a summarized network, a supernode v' represents the subset $v' \subset V$ of vertices in the original network, and all these nodes are present in the restored network again. For instance, in Figure 2, extracted from the summarized network of Figure 1(b), *Daxin Jiang* and *Aidong Zhang* become individual nodes again in the restored network. A superedge, in turn, represents the set of all possible edges between all pairs of the respective nodes. In the restored network of Figure 2, *Daxin Jiang* and *Aidong Zhang* are both connected to *Jian Pei* since their supernode was connected to him. Further on, *Daxin Jiang* and *Aidong Zhang* are mutually connected with an edge since the supernode has a self-edge. The weight of a restored edge equals the weight of the corresponding superedge. The edges connecting *Daxin Jiang* and *Aidong Zhang* to *Jian Pei* both have weight 15 like the superedge did.

The restored network of Figure 2 is not identical to the corresponding original network fragment in Figure 1(a). Because the edges connecting *Jian Pei* to *Daxin Jiang* and *Aidong Zhang* were generalized into one superedge, and their weights also were generalized into a single value, 15, while the original weights were 11 and 18.

The actual summarization problem will be to find a good summarization (clusterings of nodes and edges), in the sense that it is small (*i.e.*, the number of clusters is small) and that its restoration does not deviate much from the original network.

Summarization and restoration may actually cause four types of errors (Figure 3). (1) Edge weights may change. (2) New edges may be introduced. (3) Edges may be removed, and (4) nodes may be removed. This highlights again the complexity of this problem over more traditional node clustering. Both superfluous and missing edges are usually introduced by the generalization of nodes to supernodes. When a set of nodes with similar but not identical sets of neighbors are generalized into a single supernode, the generalization also assigns them an identical set of neighbors. For some nodes, this may add neighbors (that are typical for other nodes in the supernode) or remove neighbors (that are atypical for other nodes in the supernode). In a similar way, when generalization of nodes leads to generalization of edges, edge weights are also generalized to a single value.

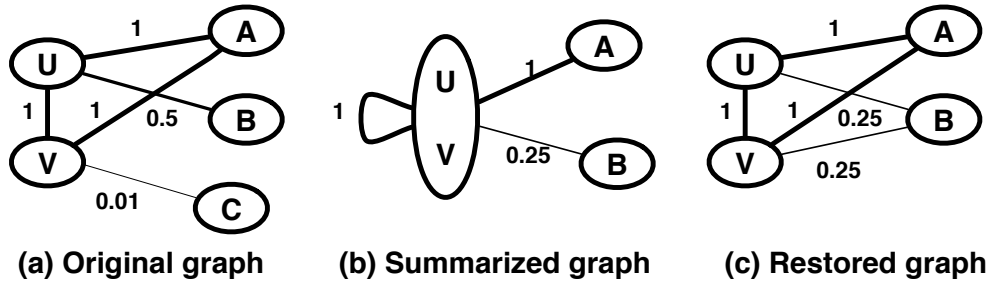


Figure 3. Illustration of four types of summarization errors: (1) The edge weight on $\{U, B\}$ changes from 0.5 to 0.25. (2) $\{V, B\}$ is a new edge in the summarized network. (3) Edge $\{V, C\}$ does not exist in the summarized network. (4) Node C does not exist in the summarized network.

The goal of summarization thus is to produce a small but accurate representation of the original network. We will formally address what is 'small' in the next subsection, and different ways of addressing what is 'accurate' in Subsections 2.3–2.5.

2.2. *Compression Ratio*

The compression ratio, denoted by cr, measures how small the summarized network is in comparison to the original network. Two variants of the compression ratio are used, each scaled to match the maximum summarization possible in different settings so that the ratio theoretically ranges from one (no space saved) towards zero (all potentially removable information removed).

In the edge-based and path-based variants (to be presented in Sections 2.3 and 2.4), all node identities but not all edge identities are preserved. Therefore the compression ratio is measured as the relative number of edges in the summary:

$$cr_1(S) = \frac{|E'|}{|E|}. \tag{1}$$

In the node-based variant (to be described in Section 2.5) nodes are additionally allowed to be removed in the summarization process. An appropriate measure of compression ratio is then based on the cardinalities of both nodes and edges:

$$cr_2(S) = \frac{|V'| + |E'|}{|V| + |E|}. \tag{2}$$

In contrast to conventional clustering, we thus measure the size of the result primarily by the number of superedges, and only in the node-based case also by the number of supernodes (node clusters).

2.3. *Edge-based Weighted Network Summarization*

While a compression ratio such as cr_1 provides a measurement of the size of the summary, it does not tell us how much information is lost in the summarization. The loss is measured by (conceptually) restoring the summarized network and then calculating the dissimilarity between the original and the restored network. In clustering terminology, this dissimilarity will constitute the objective function for clustering. For any pair

of nodes u and v, the dissimilarity between the original network and the corresponding restored network where u and v have been merged, can be thought of as the distance between u and v.

The goal of edge-based weighted network summarization is to generalize a network into a smaller one while maintaining the original structure, including edge weights, as well as possible. Since nodes are here assumed to have equal weights and equal importance, all nodes are required to be preserved in the restored network. Therefore, only the differences in the existence of edges and in their weights are calculated. The measure of dissimilarity between the original and the restored networks with identical node sets is presented below.

Definition 2.4: The *edge-based distance* between the original network $G = (V, E, w)$ and the restored network $res(S) = (V'', E'', w'')$, with an identical set of nodes $V = V''$, is

$$dist_e(G, res(S)) = \sqrt{\sum_{\{u,v\} \in V \times V} (w(\{u, v\}) - w''(\{u, v\}))^2}. \quad (3)$$

Recall that by definition a non-existent edge has weight 0, i.e., $w(\{u, v\}) = 0$ if edge (u, v) does not exist in the network, so the distance function $dist_e$ is defined for any set of nodes.

The distance measure has an interpretation as the Euclidean distance between G and $res(S)$ in a space where each pair of nodes $\{u, v\} \in V \times V$ has its own dimension. The distance can be seen as the cost of summarization, whereas the compression ratio, calculated by $cr_1(S)$, represents the savings. Since there is no unique way to balance these conflicting goals, the following form of the problem is considered.

Definition 2.5: Given a weighted network G and a compression ratio cr $(0 < cr < 1)$, the *edge-based weighted network summarization problem* is to produce a summarized representation S of G with $cr_1(S) \leq cr$ such that $dist_e(G, res(S))$ is minimized.

2.4. *Path-based Weighted Network Summarization*

For many applications, general connectivities between nodes are more important than individual edge weights. For example, in a social network setting, influence spreads beyond immediate neighbors; or in a biological network, regulatory functions of proteins can have long-ranging effects. The path-based weighted network summarization approach is also motived as a pre-processing step for computationally complex network analysis algorithms that rely more on strengths of connections than individual edge weights. The model is based on measuring the shortest path (or, in more general terms, the best paths) between all pairs of nodes. The summarization then aims to preserve the qualities of best paths, while not necessarily preserving the exact paths.

The definition of how good a path is and which one is the best depends on the kind of network and its application. For the sake of generality, our formulation is parameterized by a path quality function q. Without loss of generality, we assume that the value of the path quality function is positive, and that a larger value of q indicates better quality. For practical reasons, we also parameterize the generalized definition by a maximum path length λ.

Definition 2.6: The *path-based distance* between G and $res(S)$ with an identical node

set V, with respect to a maximum path length λ and a path quality function q, is

$$dist_\lambda(G, res(S)) = \sqrt{\sum_{\{u,v\}\in V\times V} (Q_\lambda(u, v; G) - Q_\lambda(u, v; res(S)))^2}, \quad (4)$$

where $Q_\lambda(u, v; G)$ represents the quality of the best path of length at most λ between u and v, i.e., $Q_\lambda(u, v; G) = \max_{P\in\mathcal{P}} q(P)$, where $\mathcal{P}$ is the set of all paths between u and v in G, of length at most λ. If there are no such paths then $Q_\lambda(u, v; G) = 0$.

Definition 2.7: Given a weighted network G and a compression ratio cr $(0 < cr < 1)$, the *path-based weighted network summarization problem* is to produce a summarized representation S of G with $cr_1(S) \leq cr$ such that $dist_\lambda(G, res(S))$ is minimized.

Obviously, the edge-based weighted network summarization problem defined earlier in Section 2.3 is a special case of the path-based weighted network summarization problem with $\lambda = 1$ and $q(\{e\}) = w(e)$. In this paper, only the two extreme cases are considered: $\lambda = 1$ and $\lambda = \infty$.

2.5. *Node-based Weighted Network Summarization*

In many applications, networks have weights on nodes to distinguish their importances. For example, the weight of a node can be the number of publications of an author in coauthorship networks (e.g. Alonso, Cabrerizo, Herrera-Viedma, and Herrera (2009); Zhu et al. (2011)), or the textual relevance of an online document to a query in web networks (e.g. Haveliwala, Kamvar, Kamvar, and Jeh (2003); Salton, Wong, and Yang (1975)). In this work we assume that the weight of a node is positive, $\mathcal{I} : V \to \mathbb{R}^+$.

The proposed model takes both node and edge weights into account. In contrast to the problems mentioned above, the model will delete some nodes, especially those with low weights. The goal is to maintain the most information related to nodes and edges with high weights. The respective measure of dissimilarity between the original and the restored networks is given next.

Definition 2.8: Given the original network $G = (V, E, w, \mathcal{I})$ and the restored network $res(S) = (V'', E'', w'', \mathcal{I})$ where $V'' \subseteq V$, the *node-based distance* between G and $res(S)$ (with respect to $\mathcal{I}$) is

$$dist_n(G, res(S)) = \sqrt{\sum_{\{u,v\}\in V\times V} \mathcal{I}(u)\mathcal{I}(v)(w(u, v) - w''(u, v))^2}. \quad (5)$$

Note that the node weight function $\mathcal{I}$ is shared between G and $res(S)$ since, in our model, node importances do not change in the summarization process. Again, for any node u or v not in V'', we define $w''(u, v) = 0$.

In Definition 2.8, the dissimilarity between two networks is measured by computing the change of edge weights weighted by their node weights. There is no separate cost for missing nodes, they are simply penalized by the corresponding missing edges. Since some node identities may be missing, the compression ratio cr_2 (Equation 2) is applied to measure the savings. The problem is formalized in Definition 2.9.

Definition 2.9: Given a weighted network $G = (V, E, w, \mathcal{I})$ and a compression ratio

cr $(0 < cr < 1)$, the *node-based weighted network summarization problem* is to produce a summarized representation S of G with $cr_2(S) \leq cr$ such that $dist_n(G, res(S))$ is minimized.

The edge-based weighted network summarization problem (Section 2.3) is a special case of node-based weighted network summarization problem with $\mathcal{I}(u) = 1$ for all nodes in the network; additionally, in the edge-based version we require that all nodes are preserved.

2.6. *Problem Complexity*

The weighted network summarization problem can be shown to be NP-hard, by a reduction from the max-cut problem. Formally, we present it as Theorem 2.10. The proof can be found in Appendix A.

Theorem 2.10: *The edge-based weighted network summarization problem, the path-based weighted network summarization problem, and the node-based weighted network summarization problem are NP-hard.*

Recent studies show that partitioning nodes to minimize costs is a special version of the Set Partitioning Problem (SPP) as a Cartesian product of two complete combination sets of nodes or edges of a given network (Lamarche-Perrin, Demazeau, and Vincent (2014)). The problem is thus NP-complete due to the hardness of SPP in the general case (Chakravarty, Orlin, and Rothblum (1982)). It is worth mentioning that information-theoretic measures, which are often non-monotonic, are used as cost objectives in the study of Lamarche-Perrin et al. (2014) to deduce a set of exponential solutions. The network summarization problem of this paper employs three distance-based measures designed to reveal implicit generalization structures.

3. Bounds of Distances between Networks

The weighted network summarization problems are defined through the network distance functions of Equations 3–5. Unfortunately, they can be complicated to compute, and the complexity is amplified for iterative algorithms such as hierarchical agglomerative clustering, where the distance function has to be recomputed several times. In this section, bounds for distances between networks are derived, which allow more efficient algorithms for all instances of the summarization problem. The bounds derived here will then be used by the merge operations introduced in the next section to reduce the complexity of distance calculations.

In all three variants of the problem, the network distances are defined as a function over all pairs of nodes for the sake of consistency. Obviously, in the edge-based and node-based variants (Equations 3 and 5), only the changes related to the union of the edges in the original and the restored networks need to be computed. In contrast, in the path-based variant, changes of connectivities between all pairs of nodes need to be checked (especially if maximum path length $\lambda = \infty$). Therefore, it will be more time consuming to calculate the distances in the path-based variant.

A summarized network can be obtained, in a manner similar to hierarchical agglomerative clustering, by executing a series of merge operations (to be described in Section 4) resulting in a sequence $G = S_0, S_1, \ldots, S_n = S$ of increasingly summarized networks. The distance functions $dist(\cdot)$ defined above are metrics and satisfy the triangle inequal-

ity (recall the interpretation as Euclidean distance), so the exact distance $dist(G, res(S))$ between the original and the summarized networks is upper bounded by the sum over distances between increasingly summarized networks, denoted by $dist(res(S_{j-1}), res(S_j))$. That is,

$$dist(G, res(S)) \quad \leq \quad \sum_{i=1}^{n} dist(res(S_{i-1}), res(S_i)). \tag{6}$$

This observation applies to all three variants of the problem, and it is tight in the sense that actual cases exist where the distance meets the bound.

For the edge-based and the node-based variants, $dist(res(S_{i-1}), res(S_i))$ is relatively easy to compute since the summarization operations only have local effects. However, for the path-based variant, computing $dist(res(S_{i-1}), res(S_i))$ is expensive since summarizing a part of the network may have a global effect, that is, summarization can change the connectivities between arbitrary pairs of nodes. To simplify computation in the path-based variant, we use the following upper bound. Let S_i be the result of merging nodes u and v in network S_{i-1}. As a result of the merger, new superedges are produced as mergers of previous edges. Let $d_{\max}(u, v; S_{i-1})$ denote the maximum difference of weights between any two edges merged together. The following bound now holds for many natural path quality functions q:

$$dist(res(S_{i-1}), res(S_i)) \leq \sqrt{n_e} \cdot d_{\max}(u, v; S_{i-1}), \tag{7}$$

where $n_e = |V|(|V| - 1)/2$ is the number of unordered node pairs (cf. the sum over all node pairs in Equation 4; more details can be found elsewhere (Toivonen et al. (2011))).

4. Merge Operations and Optimal Superedge Weights

In this section, merge operations are introduced to generalize a network in a manner similar to agglomerative clustering. A merge operation groups a pair of (super)nodes into a new supernode, and links the new supernode with the neighbors of the merged nodes, and then assigns weights to the new superedges. Thereby it possibly generalizes the weights of the corresponding original edges between the new generalized supernode and the neighbors of the merged nodes. To minimize the distance caused by merge operations, methods for the calculation of optimal edge weights in a merge operation are discussed.

Two kinds of merge operations are presented, *node-pair merge* and *node-pair merge with deletion*. Both operations may introduce new edges (whenever the two nodes merged to the supernode do not have equal sets of neighbors). However, the second operation has the additional possibility of removing edges and nodes. The qualities of merge operations are assessed differently, as will be explained below.

4.1. *Node-Pair Merge*

This type of operation is applied in the edge-based and the path-based network summarization problems. The reason is that by Definitions 2.5 and 2.7, the two problem variants preserve all the node identities in the summarized networks.

4.1.1. Edge-Based Summarization

Assume (super)nodes u' and v' are being merged, and let $\{x'_1, \ldots, x'_k\}$ be the union of their neighbors. (We use notation u' and v' for nodes here to remind the reader that they are in general supernodes, not nodes of the original network.) Let u' and v' in network S_{i-1} be generalized into supernode z' in the resulting network S_i. The weight of the (super)edge between u' and each neighbor x'_j is changed from $w'_{i-1}(u', x'_j)$ to $w'_i(z', x'_j)$. Similar changes also happen to other (super)edges between u' and v' and their neighbors, but not to other edges. Therefore, the dissimilarity between sequential networks S_{i-1} and S_i results from these changes alone.

Remember, however, that a single superedge represents all the edges between the corresponding original nodes. Thus, the distance between S_{i-1} and S_i is obtained as follows:

$$dist_e(res(S_{i-1}), res(S_i)) = \left(\sum_{j=1}^{k} \{ \underbrace{|u'||x'_j|(w'_{i-1}(u', x'_j) - w'_i(z', x_j))^2}_{\text{first term}} + \underbrace{|v'||x'_j|(w'_{i-1}(v', x'_j) - w'_i(z', x'_j))^2}_{\text{second term}} \} \right)^{1/2}. \quad (8)$$

The first term in (8) denotes the change related to the $|u'| \cdot |x'_j|$ edges represented by (super)edge $\{u', x'_j\}$, and the second term denotes the change related to the $|v'| \cdot |x'_j|$ edges represented by superedge $\{v', x'_j\}$. Equation 8 can be used within Equation 6 to compute an upper bound for the distance between the original and a summarized network.

We now move on to see how to assign edge weights optimally. In the edge-based network summarization setting the weight of the new superedge is independent of other new superedge weights. In order to minimize $dist_e(res(S_{i-1}), res(S_i))$ of Equation 8, the optimal superedge weight of $\{z', x'_j\}$ is then obtained simply as the mean weight of the original edges between u', v' and x'_j, that is,

$$w'_i(z', x'_j) = \frac{|u'| w'_{i-1}(u', x'_j) + |v'| w'_{i-1}(v', x'_j)}{|u'| + |v'|}. \quad (9)$$

Obviously, according to Equation 9, $w'_i(z', x'_j)$ is always positive, as edge weights are non-negative. When edge weights are optimized this way, supernode z' has links with all neighbors of u' and v'. All original edges are preserved in the summarized network and some new ones are potentially added by the generalization. The pseudocode of the node-pair merge is described in Appendix B.

4.1.2. Path-Based Summarization

The node-pair merge operation used in the path-based variant is much more complicated, because the effect of a single merge operation is in general not local anymore: edge weights contribute to best paths and therefore distances up to λ hops away.

This yields two consequences. (1) Optimal assignment of weights might in general require adjusting also other weights than the ones of the new superedges, e.g., to compensate an increase in edge weight in a superedge by decrease of some other weights elsewhere on the network. This would be computationally a hard problem. To avoid the expensive computation, the node-pair merge operation outlined above (and detailed in Appendix B) is used as an efficient, approximate solution in the path-based variant, where we only set the weights of the new superedges. The exploration of better solutions

is left for future work. (2) Computing the effect of a merge operation on the network distance in general requires computation of all-pairs best paths (of length up to λ), even if only the superedge weight is adjusted. The bounds of Equations 6 and 7 are used as efficient approximations between the summarized and the original networks.

4.2. *Node-pair Merge with Deletion*

The second merge operation, where both nodes and edges are possibly removed during the merge operation, is applied in the node-based variant. Compared with the operation discussed in the previous section, there are two major differences. The first is that the operation may remove nodes and edges, therefore, its effect could not be assessed by plain network dissimilarity. Instead, a new evaluation criterion, standard normalized error ('error over size reduction'), is used, which calculates the exact distance between the original and current summarized network and divides it by the total saved space. The second is that instead of computing the distance to the previous state, the distance to the original network is calculated.

The standard normalized error thus is

$$ne_{std} = \frac{dist_n(G, res(S_i))}{sr}, \tag{10}$$

where sr denotes the total amount of space saved by generalizing G to S_i.

The total saved space sr can be computed as follows. Assume that (super)nodes u' and v' of network S_{i-1} are generalized into supernode z' in the resulting network S_i. Let $|nei_{S_{i-1}}[u']|$ be the number of neighbors of u' in S_{i-1}, including u' itself if a self-edge exists on u'. Hence $|nei_{S_i}[z']| = |nei_{S_{i-1}}[u'] \cup nei_{S_{i-1}}[v']|$. Merging u' and v' into z', the number of nodes is reduced by 1, and the number of edges is reduced by the number of shared neighbors, which is $|nei_{S_{i-1}}[u']| + |nei_{S_{i-1}}[v']| - |nei_{S_i}[z']|$. The amount of space saved by one merge when generalizing S_{i-1} to S_i, denoted by sr_i, is $sr_i = 1 + |nei_{S_{i-1}}[u']| + |nei_{S_{i-1}}[v']| - |nei_{S_i}[z']|$. The total amount of space saved by generalizing G to S_i (through $S_1, S_2, \ldots, S_{i-1}$) is $sr = \sum_{m=1}^{i} sr_m$.

We next discuss the essential part, deleting superedges and supernodes, in merging u' and v', to minimize the standard normalized error. Clearly, the network dissimilarity is minimized when no new superedge is removed. However, it is not equivalent to the standard normalized error being minimized. The standard normalized error could be minimized by saving more space with a little extra network dissimilarity, that is, through dropping new superedges and possibly also nodes.

In our model, nodes are removed as a result of getting isolated, i.e., a node with no edges is considered deleted. Assume that superedge $\{x'_j, z'\}$ is removed, and let $\Delta sr((u', v'), x'_j)$ represent the resulting savings in space. By default, $\Delta sr((u', v'), x'_j) = 1$, corresponding to the single deleted edge. If removing $\{z', x'_j\}$ makes x'_j isolated, then x'_j will be removed and $\Delta sr((u', v'), x'_j) = 2$. If x'_j has a self-edge, then it will be removed too and $\Delta sr((u', v'), x'_j) = 3$. The total saved space then is $sr + \Delta sr((u', v'), x'_j)$.

The distance $dist_n(G, res(S_i))$ between the current state S_i and the original network G can be calculated incrementally and efficiently from $dist_n(G, res(S_{i-1}))$ since the changes are local to the merged nodes. Recall that $dist_n(G, res(S_i))$ is computed as a squareroot of a sum of squares. Let $d_n(u', v')$ denote the change to this sum, due to merging nodes u' and v' in S_{i-1}. Then $dist_n(G, res(S_i)) = (dist_n(G, res(S_{i-1}))^2 + d_n(u', v'))^{1/2}$. Note that $d_n(u', v')$ can be negative in some cases, as the new merger can actually shorten the distance to the original network. Additionally, in case a superedge $\{x'_j, z'\}$ is re-

moved, the resulting change to the sum of squares must be added; we denote this difference by $\Delta d_n((u', v'), x'_j)$. The updated network distance then is $(dist_n(G, res(S_i))^2 + \Delta d_n((u', v'), x'_j))^{1/2}$.

The standard normalized error of merging u', v' in S_{i-1} with deletion then is

$$ne_{std-del} = \frac{\sqrt{dist_n(G, res(S_i))^2 + \Delta d_n((u', v'), x'_j)}}{sr + \Delta sr((u', v'), x'_j)}.$$

If $ne_{std-del} \leq ne_{std}$, then removing superedge $\{x'_j, z'\}$ results in a smaller standard normalized error and superedge $\{x'_j, z'\}$ can be deleted. The merge operation only terminates when no edge (or node) can be removed to obtain a smaller standard normalized error.

Finally, an approximately optimal new superedge weight is the weighted mean of the weights in S_{i-1}, similarly to Equation 9,

$$w_i(z', x'_j) = \frac{\mathcal{I}(u')w_{i-1}(u', x'_j) + \mathcal{I}(v')w_{i-1}(v', x'_j)}{\mathcal{I}(u') + \mathcal{I}(v')}, \tag{11}$$

where the weight $\mathcal{I}(u')$ is the sum of importances of nodes inside u', that is, $\mathcal{I}(u') = \sum_{a \in T(u')} \mathcal{I}(a)$. Note that $w_i(z', x'_j)$ is always positive in Equation 11, so z' connects with all neighbors of u' and v'.

The merge operation with deletion, using standard normalized error as criterion, is given in Algorithm 1. It takes a network and two nodes as input, and returns a network where the given nodes are merged into one. The new superedge weights are set according to Equation 11 (line 7). In order to obtain the optimal standard normalized error, it first calculates the extra error $\Delta d_n((u', v'), x'_j)$ for each new superedge (line 9). Then, the algorithm iteratively tries to remove superedges to improve the standard normalized error (lines 13-22).

5. Algorithms

A series of algorithms are next proposed for the weighted network summarization problem. All of the proposed algorithms work in a greedy fashion, similar to hierarchical agglomerative clustering, generalizing two (super)nodes and their edges at a time until the specified compression ratio is achieved. We start with the most computationally complex algorithm, brute-force method, and progress to faster methods.

Brute-force algorithm. The brute-force method iteratively evaluates the effects of all possible pairwise merges and performs the best merge, and then repeats this until the requested compression ratio is achieved. This is the most direct counterpart to hierarchical agglomerative clustering. It suffers, however, from the need to evaluate the effects of merges in each iteration.

The brute-force and other proposed methods can be improved by the 2-hop optimization. Instead of considering arbitrary pairs of nodes, the 2-hop optimized version only considers u' and v' for a potential merger if they are exactly two hops from each other. Since 2-hop neighbors have a shared neighbor that can be linked to the merged supernode with a single superedge, some space can be saved. The time saved by the 2-hop optimization is significant: for the brute-force method, for instance, only approximately

Algorithm 1 mergeDel(u', v', S_{i-1})

Input: Nodes u' and v', and $S_{i-1} = (V_{i-1}, E_{i-1}, w_{i-1})$

Output: $S_i = (V_i, E_i, w_i)$ obtained by merging u' and v' in S_{i-1} and possibly removing nodes and edges.

$V_i \leftarrow V_{i-1}$; $E_i \leftarrow E_{i-1}$; $w_i \leftarrow w_{i-1}$;
$z' \leftarrow \{u', v'\}$;
$V_i \leftarrow V_i \setminus \{u', v'\} \cup \{z'\}$;
{/* replace old edges by new superedges */}
for $x'_j \in V_{i-1}$ such that $\{u', x'_j\}$ or $\{v', x'_j\} \in E_{i-1}$ **do**
  $E_i \leftarrow E_i \setminus \{\{u', x'_j\}, \{v', x'_j\}\} \cup \{\{z', x'_j\}\}$;
  $w_i(u', x'_j) = 0; w_i(v', x'_j) = 0$;
  $w_i(z', x'_j) = \frac{\mathcal{I}(u')w_{i-1}(u', x'_j) + \mathcal{I}(v')w_{i-1}(v', x'_j)}{\mathcal{I}(u') + \mathcal{I}(v')}$;
end for
{/* calculate the extra network dissimilarity from removing each new superedge */}
for $x'_j \in V_i$ such that $\{z', x'_j\} \in E_i$ **do**
  calculate $\Delta d_n((u', v'), x'_j)$;
end for
{/* Iteratively remove superedges $\{z', x'_j\}$ */}
repeat
  compute $ne_{std-del} = \frac{\sqrt{dist_n(G, res(S_i))^2 + \Delta d_n((u', v'), x'_j)}}{sr + \Delta sr((u', v'), x'_j)}$ for each new superedge $\{z', x'_j\}$;
  let $\{z', x'\}$ be the superedge that produces the smallest normalized error $ne_{std-del}$;
  if $ne_{std-del} \leq ne_{std}$ **then**
    $E_i \leftarrow E_i \setminus \{z', x'\}$;
    $w_i(z', x') = 0$;
    if x' is isolated **then**
      $V_i \leftarrow V_i \setminus \{x'\}$;
      $E_i \leftarrow E_i \setminus \{x', x'\}$;
    end if
    update $dist_n(G, res(S_i))$, sr and ne_{std};
  end if
until no further removal can reduce the normalized error;
return $S_i = (V_i, E_i, w_i)$

$O(|V|d^2)$ feasible node pairs need to be considered with the optimization, where d is the maximum node degree, instead of the $O(|V|^2)$ pairs in the unoptimized algorithm.

In the edge-based variant, the time complexity of the brute-force method is then $O(d^3|V|^2)$. In the path-based variant, the time complexity is $O(d^2|V|^2|E|\log|V|)$. The time complexity for the node-based variant is $O(d^3|V|^3)$. More detailed complexity analyses of the algorithms can be found in Appendix C.

Thresholded algorithm. The next algorithm is a more practical alternative, a thresholded method (Algorithm 2). It iterates over all pairs of nodes and executes a merge operation once the effect of merging (u', v') is below the current threshold, that is, $ASSESS(u', v'; S) \leq T$. The threshold value T is increased iteratively in a heuristic manner until no merge can be done with the current threshold.

Different schemes for setting the threshold would give different results and time complexity. The heuristic we have used has $K = 20$ exponentially growing steps. The aim is to produce relatively high-quality results faster than the brute-force method. Increasing

Algorithm 2 Thresholded Algorithm

Input: Network G, compression ratio cr, and problem parameters.
Output: Summarization S

```
while CR(S) > cr do
  for all pairs {u', v'} ∈ V' × V' do
    if ASSESS(u', v'; S) ≤ T and CR(S) > cr then
      S ← MERGE(u', v', S);
    end if
  end for
  if CR(S) ≤ cr then
    return S;
  end if
  update T;
end while
```

the threshold in larger steps would give a faster method, but eventually it will equal random summarization (cf. *Random algorithm* below).

In the worst case, the thresholded algorithm reduces to the brute-force greedy method. The time complexity of the thresholded algorithm for the edge-based variant is $O(d^3|V|^2)$. The time complexity for the path-based variant is $O(d^2|V|^2|E|\log|V|)$. In the node-based variant, the time complexity is $O(d^3|V|^3)$. These are theoretical upper bounds for highly improbable worst cases, and in practice the algorithms are faster.

Semi-greedy algorithm. This method is half random, half greedy. In each iteration, it first picks a node v' at random, and then chooses node u' so that the merger of u' and v' is optimal with respect to network dissmilarity caused by the merger. With 2-hop optimization, this algorithm is a generalized version of the algorithm proposed by Navlakha et al. (2008).

The resulting complexity in the edge-based variant is then $O(d^3|V|)$. In the path-based weighted variant, the total time complexity is $O(d^2|V||E|\log|V|)$. The time complexity for the node-based weighted variant is $O(d^3|V|^2)$.

Random algorithm. This random method simply merges pairs of nodes randomly without any aim to produce a good summarization. (The random method provides a baseline for the quality of other methods that make informed decisions about mergers.)

For the randomized methods, a straight-forward implementation of 2-hop optimization by random walk has a nice property. Assume that one node (u') has been chosen. Then a random pair (u', v') is found by taking two consecutive random hops starting from the first node u'. A 2-hop neighbor v' that has more shared neighbors with u' is more likely to get picked, since there are several 2-hop paths from u' to v'. Such pairs, with more shared neighbors, lead to better summarization. A uniform selection among all 2-hop neighbors does not have this property.

The total complexity of the random method is $O(d|V|)$ in the edge-based variant, $O(|V||E|\log|V|)$ in the path-based variant, and $O(d|V|^2)$ in the node-based variant.

A conclusion from the complexity analysis is that the methods possess a wide range of time complexities. The fastest non-random methods (semi-greedy) have complexities starting from $O(d^3|V|)$, i.e., linear in the number of nodes but cubic in the degree of nodes. The more accurate methods (the thresholded algorithm) have complexities starting from $O(d^3|V|^2)$, i.e., quadratic in the number of nodes and cubic in the degree of nodes.

Given the NP-hardness of the problem (Theorem 2.10), it is no surprise that methods

Table 1. Experimental datasets.

	$\|V\|$	$\|E\|$
30 medium networks	1,000 — 1,220	2,411 — 3,802
5 biological networks	1,000 — 5,000	2,295 — 16,538
5 large biological networks	10,001 50,000 103,741 179,180 215,412	12,195 103,422 232,994 323,039 406,895
4 Erdös-Rényi random networks	10,000	40,064 199,500 1,000,448 4,996,173

trying to solve it even heuristically are also computationally complex. We do not consider parallelized versions of the methods in this paper, but the experiments in the next section show that already the serial algorithms perform well on networks with tens and hundreds of thousands of edges.

6. Experiments

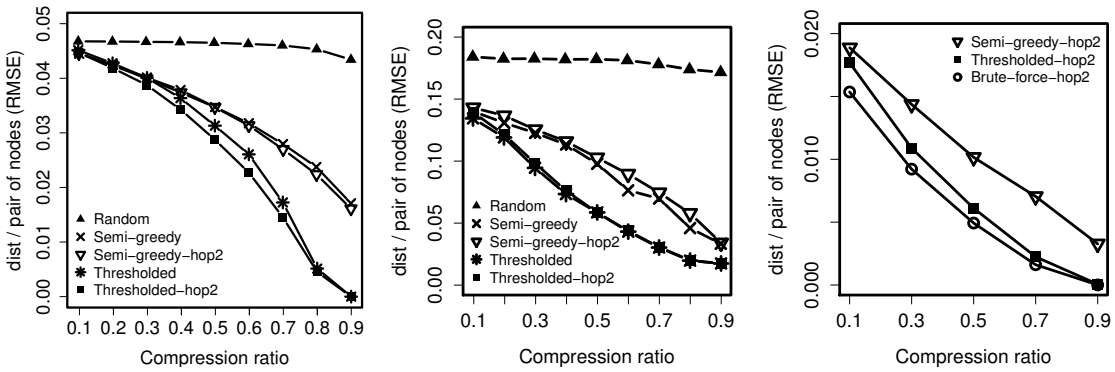
In this section we describe and discuss experimental results using the proposed algorithms on real databases. With these experiments we aim to address the following questions. (1) How well can nodes and edges be generalized: what is the trade-off between the amount of space saved and the distance to the original network? (2) How do the different algorithms fare in this task: how good are the results they produce? (3) What are the running times of the algorithms? (4) How do node weights affect the summarization? (5) How good are the merge operations? Furthermore, we present examples to show how weighted network summarization can also be applied in other graph-mining tasks and applications, e.g. graph clustering.

6.1. *Experimental Setup*

Test networks were extracted from the biological Biomine database (Eronen and Toivonen (2012)). Edges correspond to known or predicted relations between entities. Edge weights are in the range $[0, 1]$, viewed in Biomine as the probability that the edges exist. The natural path quality function then is the probability that the whole path exists, i.e., the product of weights of the edges in the path. Nodes in the Biomine database do not have weights. Node weights were assigned randomly and uniformly distributed in (0,1]. A set of 4 Erdös-Rényi random networks was used to test the performance of the algorithms in networks of varying density. These datasets are a pathological case for network summarization since they only have random structure. Datasets used in the experiments are summarized in Table 1.

6.2. *Summarization of Weighted Networks in the Three Variants of the Problem*

We start by looking at how well the test networks can be summarized. Figures 4(a)–4(c) show the distance between the summarized and original networks as a function of the



(a) Edge-based (b) Path-based (c) Node-based

Figure 4. Distance between the summarized and the original network as a function of compression ratio. A smaller compression ratio indicates a higher reduction in network size.

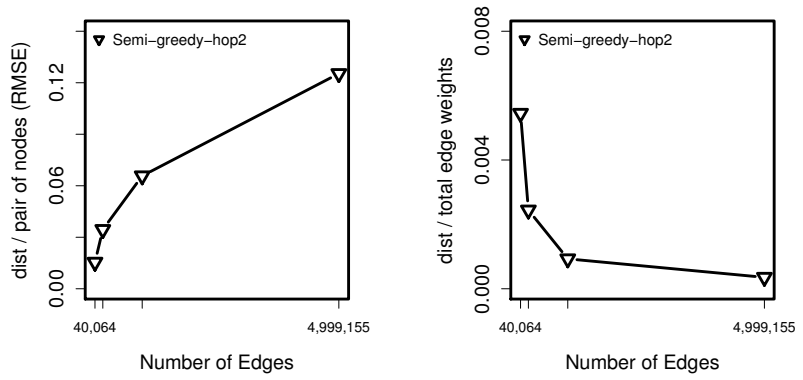
compression ratio; for now it suffices to look at the lowest curves in each figure. For better interpretability in the experimental results, we represent the distance as the root mean squared error (RMSE) over all pairs of nodes, obtained directly from the distance as $dist(G, res(S))/\sqrt{|V_G| \cdot (|V_G| - 1)/2}$. This is the y-axis in the figures. The experiments were performed on the 30 medium networks.

The findings show that our methods can produce small but accurate summaries. Summarizing a network to half of its original size can be achieved with average errors of 0.03 (in the edge-based variant), 0.06 (in the path-based variant), and 0.006 (in the node-based variant) per node pair. These results indicate that these weighted networks can be summarized to a small fraction of their original size with little error. More importantly, this implies that the summarization methods have been able to discover structure in the original networks.

6.3. *Effectiveness of Algorithms*

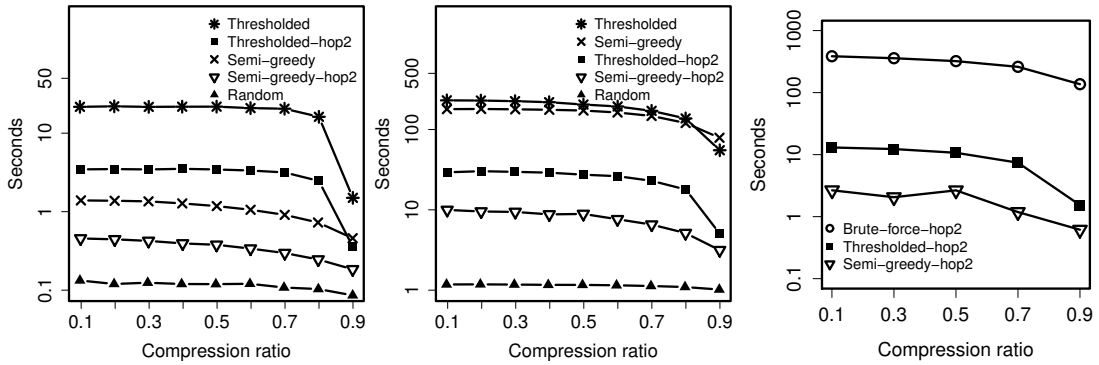
Next we compare the different algorithms in terms of their effectiveness (accuracy). The brute-force method clearly produces the best results (Figure 4(c)) (but is very slow as we will see shortly). The thresholded method produces the second best results, especially for larger compression ratios. The results are as good as with the brute-force method for compression ratios 0.7-0.9 but the gap grows a bit for smaller compression ratios. The semi-greedy algorithm, in turn, is not as good with the larger compression ratios but the performance is similar to the thresholded method when the compression ratio is small. The random pairwise summarization algorithm performs consistently the worst, with a clear margin to the other methods (Figures 4(a)-4(b)). Apparently, a few early bad merge operations raise the distance even for big compression ratios. A final observation from Figure 4(a) is that 2-hop optimization produces similar or sometimes even better results than without it.

We next evaluate the effect of network density using four Erdös-Rényi random networks in the edge-based variant. The networks are summarized to half of their original size. The result (Figure 5(a)) shows an apparent correlation between the average error and network density. The reason is that the number of edges increases and thereby the total volume of distances increases. A normalization of the distance by the total volume of edge weights (Figure 5(b)) shows that relative distances actually become smaller with denser networks. The reason is again the larger search space that denser networks have.



(a) Edge-based (b) Edge-based

Figure 5. (a) Dissimilarity (per pair of nodes) for the edge-based variant on networks of varying density. (b) Dissimilarity (per total edge weight) for the edge-based variant on networks of varying density. The compression ratio is 0.5. The number of nodes in all networks are 10,000.



(a) Edge-based (b) Path-based (c) Node-based

Figure 6. Running times of weighted network summarization algorithms (on logarithmic scales) as a function of compression ratio.

6.4. *Efficiency of Algorithms*

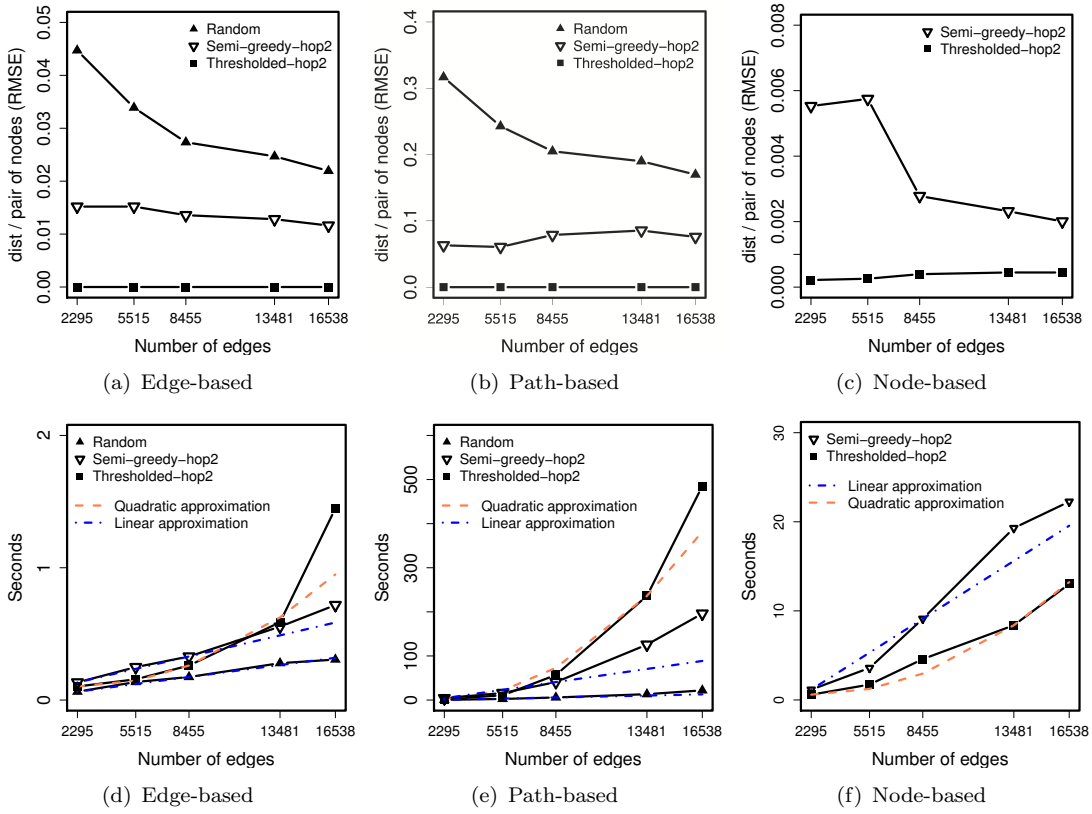
6.4.1. Efficiency as a Function of Compression Ratio

We now compare the efficiency of algorithms in each of the three variants of the problem. The mean running times of weighted network summarization algorithms as a function of compression ratio, over the 30 medium networks, are shown in Figures 6(a)–6(c).

In the edge-based variant (Figure 6(a)), the thresholded algorithm takes less than 50 seconds to summarize the network, the semi-greedy algorithm takes around 1 second, whereas the random algorithm takes 0.1 seconds. The differences are big between the methods, more than two orders of magnitude between the extremes. Running time comparisons of the methods show similar patterns in the other problem variants, even if the scales are different.

The 2-hop-optimized versions are an order of magnitude faster than the unoptimized versions while the results are equally good. 2-hop optimization thus very clearly pays off. In the path-based variant, the relative impact of the 2-hop optimization is even larger.

The brute-force method is 1 to 3 orders of magnitude slower than other methods (Figure 6(c)). This implies that the brute-force algorithm in practice only works for small networks.



(a) Edge-based (b) Path-based (c) Node-based

(d) Edge-based (e) Path-based (f) Node-based

Figure 7. Network dissimilarities and running times of weighted network summarization algorithms on 5 biological networks, as a function of the network size. The compression ratio is 0.8.

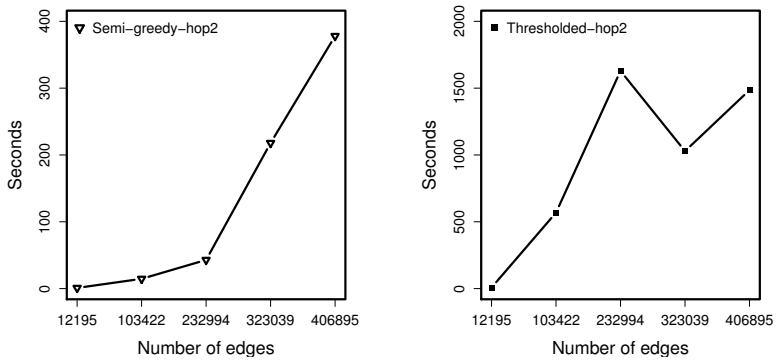
Next, the running times for different network summarization problem variants are compared. The running times of the path-based variants (Figure 6(b)) are larger than those of the edge-based variants (Figure 6(a)) by an order of magnitude; and for the semi-greedy versions, the difference is two orders of magnitude. This is not surprising given the higher complexity of the path-based version.

The running times of the node-based variants (Figure 6(c)) are a bit longer than those of the edge-based variants (Figure 6(a)). The reason is that the node-based methods have a larger search space as they also consider the options of deleting edges and nodes in addition to just merging nodes. Furthermore, the node-based method computes the exact distance between the original and current summarized networks when using standard normalized error as criterion, whereas in the edge-based variant, the distance between the previous and current summarized networks is computed instead.

6.4.2. Scalability with respect to Network Size

Figures 7(a)–7(f) show how the algorithms behave as a function of the network size, at a fixed compression ratio of 0.8, using a series of 5 biological networks. The distance between the original and summarized networks are shown in Figures 7(a)–7(c). The relative performances of the algorithms change only little. The thresholded algorithm consistently produces very good summarized networks, while the semi-greedy method improves its relative performance a bit for larger networks.

The respective running times are given in Figures 7(d)–7(f). The edge-based and path-based variants exhibit similar behavior. The random method is close to linear (and



(a) Edge-based (b) Node-based

Figure 8. Running times of weighted network summarization algorithms on 5 large biological networks, as a function of the network size. The compression ratio is 0.8.

deviations are likely due to random effects). The thresholded method is slightly super-quadratic; when the compression ratio is big and the threshold T is small, the thresholded method will iterate over almost all pairs of nodes. The semi-greedy algorithm has a much more graceful behavior, even if slightly super-linear. However, in the node-based variant (Figure 7(f)), the thresholded method is faster than the semi-greedy method. The reason is that the semi-greedy method chooses one node pair from d^2 node pairs (d is average node degree), and it considers node and edge deletion for each node pair candidate. In contrast, in the thresholded method, two nodes are randomly selected and they can be directly merged if the distance is within the threshold, so the node and edge deletion is just considered for one node pair.

Figures 7(a), 7(b), 7(d) and 7(e) together show that when the network size is large, the semi-greedy algorithm becomes relatively attractive in the edge-based and path-based settings: while the thresholded method remains clearly more accurate, the semi-greedy method scales better to large networks. On the other hand, Figures 7(c) and 7(f) together imply that the thresholded algorithm remains a good choice in the node-based variant also when the network is large, having both a better accuracy and better scalability than the semi-greedy algorithm.

We next consider scalability experiments with 5 larger biological networks, with 10,000 to 200,000 nodes and 12,000 to 400,000 edges. Based on the above results, we use the semi-greedy algorithm in the edge-based variant and the thresholded algorithm in the node-based variant, both with 2-hop optimization. Using a constant compression ratio $cr = 0.8$, the summarization times for up to 400,000 edges are less than 7 minutes (in the edge-based variant, Figure 8(a)) or half an hour (in the node-based variant, Figure 8(b); with some fluctuation caused by different properties of the graphs).

6.5. *Node-Based Summarization*

6.5.1. Effect of Node Weights on Summarization Results

We now carry out experiments specific to the node-based summarization problem, and start by assessing whether taking node weights into account during the generalization can maintain more information related to high-weighted nodes. For this purpose, the node-pair merge with deletion operation is applied on the 30 medium networks. Summarization is performed in two alternative ways: (1) with node weights, and (2) without node weights.

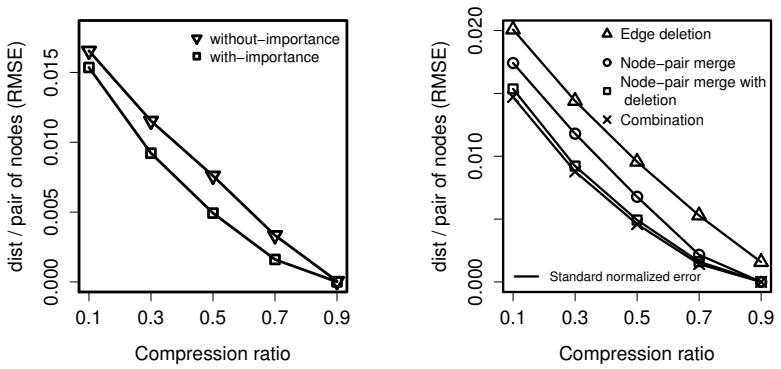


Figure 9. (a) Assessment of the effect of node weights on the quality of the summarization. (b) Effectiveness of operations.

The distance between the original and the resulting summarized networks is computed using the node weights.

A comparison of the results (Figure 9(a)) to each other, and across the whole range of summarization ratios tested, shows that results are clearly better when node weights are considered during summarization. This indicates that taking node weights into account can guide the process to a better summarization.

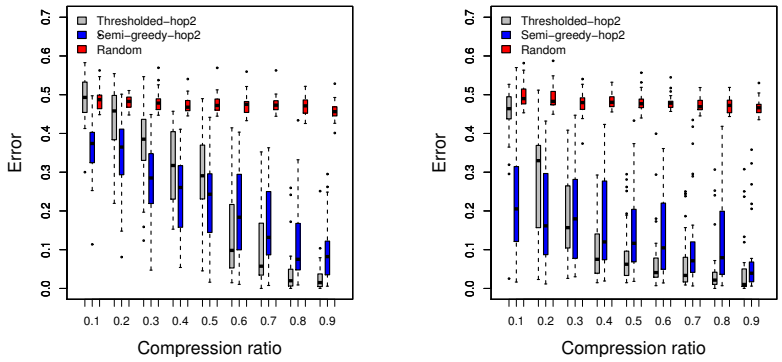
6.5.2. Comparison of Merge Operations

We next compare the two merge operations of the node-based variant, node-pair merge and node-pair merge with deletion. In addition, two other operations serve as reference points. The first reference operation is plain edge deletion. While it does not generalize the network, it helps to make it simpler. Our aim here is to see how much generalization helps over simple edge deletion. The second reference operation, called 'combination', is actually a hybrid: in each iteration, this combination operation can either perform the standard node-pair merge with deletion, or it can just delete a single edge (if there are no useful merges).

The experiments are performed on the 30 medium networks by applying the brute-force algorithm with the standard normalized error. The reference operation 'combination' performs best (Figure 9(b)), which is natural since it has the largest space of possible operations at its disposal. However, the node-pair merge with deletion is virtually equally good. They both perform better than the node-pair merge operation, especially when the compression ratio is small. The results show that deleting some weak nodes and edges in the merge actually produces a smaller error. Edge deletion alone performs relatively poorly, as can be expected.

6.6. *Case studies*

We next describe three different case studies of weighted network summarization: summarization as pre-processing for node clustering, automated morpheme discovery as a summarization task, and summarization as a tool to analyze evolution of prokaryotic species.



(a) Edge-based (b) Path-based

Figure 10. Effect of summarization on node clustering. Y-axis is the fraction of node pairs clustered inconsistently in the original and the summarized network.

6.6.1. Case Study I: A Pre-processing Step for Clustering

This section shows an application of weighted network summarization as a pre-processing step to reduce the network size for other computationally complex tasks on large networks. In the experiment, a node clustering algorithm is applied to both the original and the summarized networks and we then compare the resulting clusterings. (To be exact, in this experiment a clustering algorithm is run on the restored versions of the summarized networks. Algorithms operating directly on the summarized networks are outside the scope of this paper.) The focus of this test is on the practical effect of summarization on the results of the subsequent clustering phase.

The experimental data is the 30 medium networks, as each of them potentially contains some cluster-like structure arising from the connections between three sets of genes. Since the number of clusters in each network is already known, k-medoids clustering algorithm with $k = 3$ is applied. The proximity between two nodes was computed as the product of weights (probabilities) of edges on the best path between them. The difference between clusterings is measured by the fraction of node pairs that are clustered inconsistently in the clusterings, i.e., assigned to the same cluster in one network and to different clusters in the other network.

According to the results, the thresholded and semi-greedy summarization methods can generalize a weighted network with little effect on node clustering (Figures 10(a) and 10(b)). The effect is small especially in the path-based variant (Figure 10(b)), where the inconsistency (or error) is less than 0.3 (the thresholded method) or 0.2 (the semi-greedy method) for a wide range of compression ratios. The effects of the thresholded and semi-greedy versions are larger for the edge-based variant (Figure 10(a)), especially when the compression ratio cr becomes smaller. This indicates that the path-based variant has clear value over the plain edge-based one and motivates the additional complexity of path-based summarization.

Surprisingly, the semi-greedy method performs best in this comparison with compression ratio $cr \leq 0.2$. In the path-based variant even an aggressive summarization using the semi-greedy method introduced relatively little changes to node clustering. In the other extreme, clusters found in randomly summarized networks are quite different from the clusters found in the original network: close to 50% of pairs are clustered inconsistently, whereas a random clustering would have about 66% inconsistency.

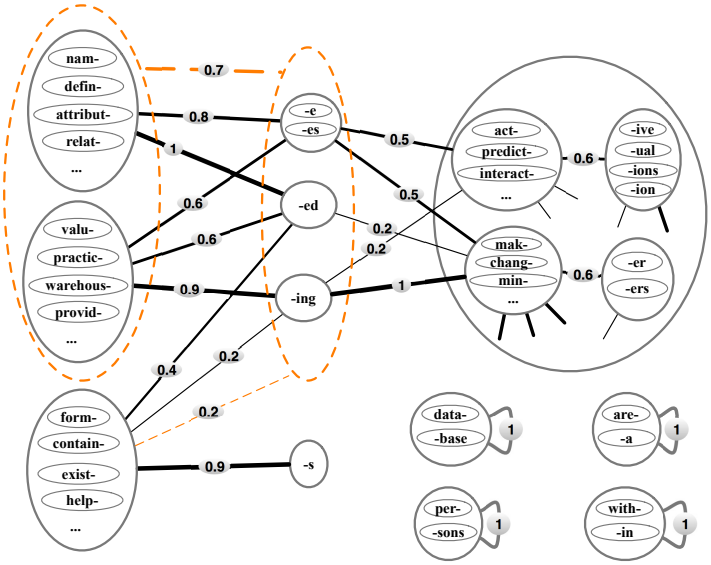


Figure 11. Partial generalization hierarchy of morphemes discovered in the the Wikipedia page for Data Mining. Edges with weight ≤ 0.1 are not shown.

6.6.2. Case Study II: Linguistic Analysis Using Network Summarization

We now look at an application of network summarization in discovery and generalization of morphological structures. The goal is to use network summarization to discover different types of morphemes in an English document. We illustrate this application with a simple example, focusing only on word-starting segments (linguistically typically stems or roots) and word-final segments (typically suffixes). For simplicity, in this example we call them collectively stems and suffixes, respectively.

In our example, the Wikipedia page for Data Mining[3] is used as the source document. A set of potential stems and suffixes is obtained heuristically by splitting white-space separated words of the document into a possible stem and suffix. Each stem and each suffix is then considered a node. A stem and a suffix are connected with an undirected edge if their concatenation is a word in the document. A single word may be represented by several alternative splits into a stem and a suffix.

Now, the edge-based network summarization finds generalizations of stems and suffixes (Figure 11). With compression ratio $cr = 0.1$ (clusters with solid lines), supernodes correspond in particular to three different types of verb/noun stems. (1) One supernode consists of stems such as *practic-, valu-, warehous-, provid-, includ-, chang-, mak-* (and 16 others), all strongly connected with inflectional endings *-ing, -ed, -es, -e*. Not only does summarization find a natural generalization, the generalization also helps one predict that *practicing, valued, included, marketed* etc. are possible words even though they do not occur in the document. (Errors are made only with *market-* and four other stems that cannot be paired with *-es, -e*). (2) Another supernode contains those verb/noun stems that are not used as present participles or gerunds, i.e., they do not occur with *-ing* in the document, but are strongly connected with inflectional endings *-ed, -e, -es*. This supernode consists of stems *defin-, nam-, attribut-, rul-, relat-, generat-, bas-, automat-*. (3) The third supernode contains stems of verbs/nouns that are typically used just with the inflectional ending *-s* and that are dictionary forms of the word: *form-, contain-, exist-, help-, part-, decision-, require-, integrate-, mine-, use-*. The supernode also has

[3] `http://en.wikipedia.org/wiki/Data_mining`

weak edges with *-ed, -ing*.

In addition, there are separate supernodes for four words that the preprocessing step split to parts: *data-, -base*; *with-, -in*; *per-, -sons*; and *are-, -a*; each with a very strong self-edge and no links elsewhere. There is also one larger supernode consisting of less representative stems and suffixes (see below).

At this level of generalization, suffixes *-ed* and *-ing* are not grouped together even though that would seem linguistically appropriate. The reason is that some stems do not occur with *-ing* in the document, as was described above, so keeping *-ed, -ing* separate helps identify different groups of verbs/nouns based on their usage patterns. When summarizing the network more, for example $cr = 0.06$ (clusters with dashed lines), the verb/noun inflectional endings *-ed, -ing, -e, -es* are all grouped into one supernode, reflecting their similar linguistic roles. This generalization suggests that *defin-, nam-,* etc. could occur with *-ing* even though they never did in the document, and this obviously is a correct prediction.

A more fine-grained analysis is obtained, in turn, by using a higher compression ratio. At $cr = 0.2$, the larger and somewhat mixed supernode mentioned above is split to linguistically motivated supernodes. In particular, there are two groups of derivative endings: *-ers, -er* and *-ive, -ual, -ions, -ion*, and the supernodes of verb/noun stems have been refined accordingly to match their use with these endings. Stems *chang-, mak-, min-, market-, driv-* are used to derive actors with *-ers, -er* (including a number of novel, valid generalizations), and *predict-, interact-, act-, collect-, associat-, mill-* to derive adjectives and nouns with endings *-ive, -ual, -ions, -ion*. Here, *mill-ion* is an obvious mistake, and some of the combinations with *-ual* are not good, but otherwise there are again many valid, novel generalizations.

These observations illustrate how network summarization can discover natural generalizations of morphemes based on their linguistic roles and usage patterns, and that these generalizations help predict several unseen words. Summarizing morpheme structures is just one example of many potential applications involving text. A large number of methods exist for producing graphs from text corpora and using them for various natural language processing tasks (Nastase, Mihalcea, and Radev (2015)). A particularly interesting task for network summarization is extractive document summarization, more specifically, graph-based summarization (e.g.,Amancio, Nunes, Oliveira, and Costa (2012); Baralis, Cagliero, Mahoto, and Fiori (2013); Gross, Doucet, and Toivonen (2014)), where a summary of the graphical representation of a document potentially could be used to extract a useful textual summary of the document.

6.6.3. Case Study III: A Tool for Understanding the Contingency of Evolution of Prokaryotic Species

We next describe an already published biological application of node-based weighted network summarization (Zhou, Toivonen, and King (2014)). The goal is to compare thousands of inferred metabolisms in *Archaea* and *Eubacteria* to gain understanding of how different pathways have evolved since their divergence.

To simplify computation, the metabolisms of each domain are integrated into one weighted network, weighing nodes by their importance in the metabolic networks of that domain. The weighted metabolic networks are analyzed using the node-based network summarization technique, and our hypothesis is that comparing summarizations across and between the two biological domains may help us understand the biodiversity and evolution of prokaryote metabolisms. We calculated the average weight of the nodes remaining in the summarized network and investigated the similarity of those summarized

pathways. All these results provide evidence for the conservation of evolution. A brief overview of the application is in Appendix D.

7. Related Work

Network summarization: Network summarization as presented in this paper is based on merging nodes that have similar relationships to other entities, i.e., that are structurally most equivalent — a classic concept in social network analysis (Lorrain and White (1971)). There are other alternative definitions for structural similarity of nodes (e.g. Jeh and Widom (2002); Leicht, Holme, and Newman (2006)). Each one makes their own assumptions and works best when those assumptions hold in data. For instance, the structural similarity measure of Leicht et al. (2006) is based on the assumption that similar nodes tend to be linked. We specifically avoid that assumption since we are interested in structural similarities also in cases where similar nodes are not linked. The case study in morphology is a good example of this: inflectional endings *-ing, -ed, -es, -e* are grouped together even though they are never linked to each other.

Structural equivalence and many other types of relations between (super)nodes have been considered in social networks under block modeling (see, e.g., Borgatti and Everett (1992)), where the goal is both to identify supernodes and to choose among the different possible types of connections between them. Our approach (as well as the ones by Navlakha et al. (2008) and Tian et al. (2008), see below) uses only two types of block model relations: 'null' (no edges) and 'complete' (all pairs are connected), since they are the only types that are directly represented by the existence or non-existence of a superedge, without any other additional information.

Due to the increasing interest on network data, many algorithms have been proposed for network summarization problems, including MDL-based algorithms (LeFevre and Terzi (2010); Navlakha et al. (2008)), OLAP methods (Chen, Yan, Zhu, Han, and Yu (2008)), SNAP (Tian et al. (2008)), and SCMiner (Feng, He, Konte, Böhm, and Plant (2012)) for bipartite networks. All these methods aim to produce a highly compact representation that is useful for data mining tasks, such as reducing the number of embeddings when searching frequent subgraphs in a large graph (e.g. Chen et al. (2009); Zhu, Zhang, and Qu (2013)), revealing biological modules (e.g. Navlakha, Schatz, and Kingsford (2009)), identifying magnet communities (e.g. G. Wang, Zhao, Shi, and Yu (2012)), facilitating queries (e.g. Fan, Li, Wang, and Wu (2012)), protecting privacy (e.g. Hay, Miklau, Jensen, Towsley, and Weis (2008); Skarkala et al. (2012)), analyzing node roles in social networks (e.g. Kate and Ravindran (2009)), understanding the structure of large networks (e.g. Koutra, Kang, Vreeken, and Faloutsos (2014)), and online network analysis (e.g. Qu et al. (2011)). However, none of the above works considers summarizing *weighted* networks.

Many of the algorithmic ideas we use are similar to those of Navlakha et al. (2008) and LeFevre and Terzi (2010). Theirs and our methods are conceptually based on hierarchical, agglomerative clustering and different techniques to implement it efficiently. However, we generalize all these approaches in three important and related directions: to weighted networks, to long-range, indirect (weighted) connections between nodes, and to take node weights into account.

Other related work includes compression of Web networks (e.g. Adler and Mitzenmacher (2001); Apostolico and Drovandi (2009); Boldi and Vigna (2004); Randall, Stata, Wiener, and Wickremesinghe (2002)): the goal is to produce as compact a representation of a network as possible, in whatever format and usually not as a network, and use the bit/edge rate as evaluation criterion. These studies focus on minimizing the cost per link

in any representation rather than discovering generalizations of nodes and edges and constructing a summarized network. We produce a representation that is a network, too, and that lends itself e.g. to graphical visualization of the generalized structure of the network.

In preliminary work on this topic (Toivonen et al. (2011)), the weighted network summarization problem and its edge-based and path-based variants were introduced, and some initial solutions were proposed. In the current paper, we introduce the node-based network summarization variant, a novel problem motivated by real-world applications (e.g. Zhou et al. (2014)), and integrate the three problem variants into one self-contained paper. We additionally give the computational hardness of the problem, elaborate on the problem definitions and derivations of bounds, propose a larger range of merge operations, perform more extensive experiments, present more applications, and discuss the related research to a greater extent.

Node clustering/graph partitioning Closely related to our work but conceptually subtly different are problems associated with node clustering and graph partitioning (e.g. Elsner (1997); Fjällström (1998)). The goal is to find groups of nodes that are more strongly connected to each other than to nodes in other groups. Graph partitioning techniques can be used to summarize graphs, too, but in a limited way: clusters of highly interlinked nodes can be treated as supernodes, connected by superedges (e.g. Mueller, Haegler, Shao, Plant, and Böhm (2011)). In our terminology, such operations specifically look for supernodes with strong self-edges (since nodes within clusters are relatively strongly connected) and weak superedges between supernodes (since clusters are relatively weakly connected). In contrast, we group together nodes that have similar relations to other nodes, regardless of how nodes grouped together connect mutually to each other.

Co-clustering: There is a host of work on bi-clustering or co-clustering (e.g. Dhillon, Mallela, and Modha (2003); Shan and Banerjee (2008); X. Wang, Tang, Gao, and Liu (2010)), which is a problem of simultaneously clustering rows (e.g., objects) and columns (e.g., attributes) of a data matrix. Unlike traditional clustering algorithms that focus on clustering one dimension of the matrix (the objects), co-clustering seeks blocks of rows and columns that are inter-related, and the output is a set of row clusters and a set of column clusters. On an abstract level, such simultaneous generalization of rows and columns bears similarities to simultaneous generalization of nodes and edges in a network. However, viewing the data matrix as a network, co-clustering methods essentially work on bipartite networks where the nodes are divided into rows and columns. Our methods work on networks in general, not only on bipartite networks. For more related work on this topic, we refer to reviews of biclustering algorithms (Madeira and Oliveira (2004); Tanay, Sharan, and Shamir (2005)).

Subgraph extraction and graph reduction: The goal of subgraph extraction is to extract a subnetwork, of some limited size, that maximally connects given nodes. Usually, a subgraph is extracted for nodes specified by users (e.g. Gilbert and Levchenko (2004); Hintsanen and Toivonen (2008)), or it can be done for all pairs of nodes (e.g., Hauguel, Zhai, and Han (2009); Toivonen, Mahler, and Zhou (2010)) by discarding edges (and nodes). Subgraph extraction does not generalize nodes or edges, it only picks out a subset of them. This is in contrast with the present paper where the goal is to discover generalizations of nodes and edges.

8. Conclusion

We have studied the knowledge discovery task of finding generalizations of nodes and edges in weighted networks, formalized as weighted network summarization. Three variants of the problem are studied: one is edge-oriented, aiming to preserve edge weights; another one is path-oriented, trying to preserve strengths of connections of up to λ hops; and the last one is node-oriented, with the goal of preserving more information related to high-weight nodes. The problems were shown to be NP-hard. We derived bounds that are useful in heuristic solutions, proposed such algorithms to solve the problem, and gave experimental results. The use of weighted network summarization as a knowledge discovery method was illustrated with three different case studies.

The following findings are concluded from the experimental studies. (1) Good generalizations of nodes and edges can be found in a data-oriented manner, in the sense that they can be used to summarize weighted networks quite a lot with little loss of information. (2) Node weights can guide the summarization process to better preserve information related to nodes with high weights. (3) Weighted networks can be summarized efficiently. (4) The path-based weighted network summarization approach is promising as a preprocessing step for computationally complex network analysis algorithms that rely more on strengths of connections.

A useful property of the proposed methods, built into the problem definition, is that the results are networks themselves. This gives two benefits. First, representing networks as networks is user-friendly. Users can easily tune the abstraction level by adjusting the compression ratio. This could also be done interactively to support visual inspection of a network. Second, some graph algorithms can be applied directly on the summarized network with reduced running time.

Network summarization is based on structural properties of the network; it follows that networks of different topologies behave differently when summarized. It is an interesting question then to what extent different network types and models (Newman (2010)) could be recognized and analyzed based on their summaries, compression ratios, number of self-edges or supernode size distribution. For instance, dense communities tend to be summarized into supernodes with strong self-edges. Erdös-Rényi random graphs tend to be summarized poorly due to their lack of non-random structure. Investigation into the applicability of network summarization to this kind of network analysis is a promising topic for future work.

Several obvious avenues exist for future work. More efficient algorithms deserve further research, as the computational complexities of the algorithms are polynomial in $|V|$, which limits their use for very large and dense networks. An interesting question regarding the path-based variant is how to better optimize superedge weights for maximum path lengths λ larger than one. Another one is the effect of λ on results and running time; we only considered values 1 and ∞. Finally, it would be interesting to study the use of network summarization as a preprocessing step for graph mining algorithms, extending the work from our Case Study I in Section 6. It appears that algorithms working directly on summarized representations could potentially be very efficient.

Acknowledgements

The authors would like to thank the anonymous referees for several constructive comments on the manuscript. The authors also thank Aleksi Hartikainen for his contributions to the code.

Disclosure statement

The authors have no conflict of interest to declare.

Funding

This work has been supported by the Academy of Finland (grants 118653, Algodan Centre of Excellence, and 276897, CLiC), by the European Commission (FET grant 611733, ConCreTe), and by the Russian Science Foundation (grant No. 15-11-10032).

References

Adler, M., & Mitzenmacher, M. (2001). Towards Compressing Web Graphs. In *Data Compression Conference* (pp. 203–212).

Alonso, S., Cabrerizo, F., Herrera-Viedma, E., & Herrera, F. (2009). h-Index: A Review Focused in its Variants, Computation and Standardization for Different Scientific Fields. *Journal of Informetrics*, *3*(4), 273–289.

Amancio, D. R., Nunes, M. G., Oliveira, O. N., & Costa, L. d. F. (2012). Extractive summarization using complex networks and syntactic dependency. *Physica A: Statistical Mechanics and its Applications*, *391*(4), 1855–1864.

Apostolico, A., & Drovandi, G. (2009). Graph compression by BFS. *Algorithms*, *2*(3), 1031–1044.

Baralis, E., Cagliero, L., Mahoto, N., & Fiori, A. (2013). GRAPHSUM: Discovering correlations among multiple terms for graph-based summarization. *Information Sciences*, *249*, 96–109.

Boldi, P., & Vigna, S. (2004). The Webgraph Framework I: Compression Techniques. In *Proceedings of the 13th International Conference on World Wide Web* (pp. 595–602). ACM.

Borgatti, S. P., & Everett, M. G. (1992). Regular blockmodels of multiway, multimode matrices. *Social Networks*, *14*, 91–120.

Chakravarty, A. K., Orlin, J. B., & Rothblum, U. G. (1982). A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations Research*, *30*(5), 1018–1022.

Chen, C., Lin, C. X., Fredrikson, M., Christodorescu, M., Yan, X., & Han, J. (2009). Mining Graph Patterns Efficiently via Randomized Summaries. *Proceedings of the VLDB Endowment*, *2*(1), 742–753.

Chen, C., Yan, X., Zhu, F., Han, J., & Yu, P. (2008). Graph OLAP: Towards Online Analytical Processing on Graphs. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 103–112).

Dhillon, I. S., Mallela, S., & Modha, D. S. (2003). Information-Theoretic Co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 89–98).

Elsner, U. (1997). *Graph Partitioning A survey* (Technical Report SFB393/97-27). Technische Universität Chemnitz.

Eronen, L., & Toivonen, H. (2012). Biomine: predicting links between biological entities using network models of heterogeneous databases. *BMC Bioinformatics*, *13*, 119.

Fan, W., Li, J., Wang, X., & Wu, Y. (2012). Query Preserving Graph Compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (pp. 157–168).

Feng, J., He, X., Konte, B., Böhm, C., & Plant, C. (2012). Summarization-based Mining Bipartite Graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1249–1257).

Fjällström, P.-O. (1998). Algorithms for graph partitioning: A survey. *Linköping Electronic Articles in Computer and Information Science*, *3*(10).

Gilbert, A., & Levchenko, K. (2004). Compressing Network Graphs. In *Proceedings of the LinkKDD workshop at the 10th ACM Conference on SIGKDD.*

Gonalves, B., Perra, N., & Vespignani, A. (2011, 08). Modeling Users' Activity on Twitter Networks: Validation of Dunbar's Number. *PLoS ONE*, *6*(8), 1-5.

Gross, O., Doucet, A., & Toivonen, H. (2014). Document summarization based on word associations. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (pp. 1023–1026).

Hauguel, S., Zhai, C., & Han, J. (2009). Parallel PathFinder Algorithms for Mining Structures from Graphs. In *Proceedings of the 9th IEEE International Conference on Data Mining* (pp. 812–817).

Haveliwala, T., Kamvar, S., Kamvar, A., & Jeh, G. (2003). *An Analytical Comparison of Approaches to Personalizing PageRank* (Technical Report No. 2003-35). Stanford InfoLab.

Hay, M., Miklau, G., Jensen, D., Towsley, D., & Weis, P. (2008). Resisting Structural Reidentification in Anonymized Social Networks. *Proceedings of the VLDB Endowment*, *1*(1), 102–114.

Hintsanen, P., & Toivonen, H. (2008). Finding reliable subgraphs from large probabilistic graphs. *Data Mining and Knowledge Discovery*, *17*(1), 3–23.

Jeh, G., & Widom, J. (2002). SimRank: A Measure of Structural-context Similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 538–543). ACM.

Karp, R. M. (1972). Reducibility Among Combinatorial Problems. In *Complexity of computer computations* (pp. 85–103).

Kate, K., & Ravindran, B. (2009). Epsilon Equitable Partition: A positional analysis method for large social networks. In *Proceedings of the 15th International Conference on Management of Data.*

Kimura, M., Saito, K., Nakano, R., & Motoda, H. (2010). Extracting influential nodes on a social network for information diffusion. *Data Mining and Knowledge Discovery*, *20*(1), 70–97.

Koutra, D., Kang, U., Vreeken, J., & Faloutsos, C. (2014). VoG: Summarizing and Understanding Large Graphs. In *Proceedings of the 2014 SIAM International Conference on Data Mining* (pp. 91–99).

Lamarche-Perrin, R., Demazeau, Y., & Vincent, J.-M. (2014). A Generic Algorithmic Framework to Solve Special Versions of the Set Partitioning Problem. In *IEEE 26th International Conference on Tools with Artificial Intelligence, ICTAI 2014* (pp. 891–897).

LeFevre, K., & Terzi, E. (2010). GraSS: Graph Structure Summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining* (pp. 454–465).

Leicht, E. A., Holme, P., & Newman, M. E. (2006). Vertex similarity in networks. *Physical Review E*, *73*(2), 026120.

Lin, C. X., Yu, Y., Han, J., & Liu, B. (2010). Hierarchical Web-Page Clustering via In-Page and Cross-Page Link Structures. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (pp. 222–229).

Lorrain, F., & White, H. C. (1971). Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, *1*(1), 49–80.

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *1*(1), 24–45.

Mamitsuka, H. (2012). Mining from protein–protein interactions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(5), 400–410.

Mueller, N., Haegler, K., Shao, J., Plant, C., & Böhm, C. (2011). Weighted Graph Compression for Parameter-free Clustering With PaCCo. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (pp. 932–943).

Nastase, V., Mihalcea, R., & Radev, D. R. (2015). A survey of graphs in natural language processing. *Natural Language Engineering*, *21*(05), 665–698.

Navlakha, S., Rastogi, R., & Shrivastava, N. (2008). Graph Summarization with Bounded Error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of*

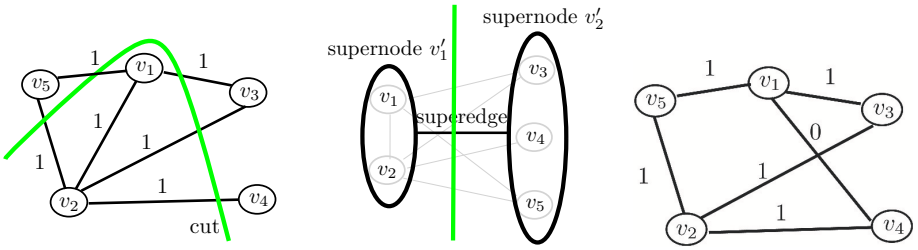
Data (pp. 419–432).
Navlakha, S., Schatz, M., & Kingsford, C. (2009). Revealing Biological Modules via Graph Summarization. *Journal of Computational Biology*, *16*(2), 253–264.
Newman, M. (2010). *Networks: An introduction*. Oxford university press.
Ota, M., Gonja, H., Koike, R., & Fukuchi, S. (2016, 06). Multiple-Localization and Hub Proteins. *PLoS ONE*, *11*(6), 1-18.
Qu, Q., Zhu, F., Yan, X., Han, J., Yu, P. S., & Li, H. (2011). Efficient Topological OLAP on Information Networks. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications* (pp. 389–403).
Randall, K. H., Stata, R., Wiener, J. L., & Wickremesinghe, R. G. (2002). The Link Database: Fast Access to Graphs of the Web. In *Data Compression Conference* (pp. 122–131).
Salton, G., Wong, A., & Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM*, *18*(11), 613–620.
Shan, H., & Banerjee, A. (2008). Bayesian Co-clustering. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 530–539).
Skarkala, M. E., Maragoudakis, M., Gritzalis, S., Mitrou, L., Toivonen, H., & Moen, P. (2012). Privacy Preservation by k-Anonymization of Weighted Social Networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining* (pp. 423–428).
Tanay, A., Sharan, R., & Shamir, R. (2005). Biclustering Algorithms: A Survey. *Handbook of Computational Molecular Biology*, *9*, 1–20.
Tian, Y., Hankins, R. A., & Patel, J. M. (2008). Efficient Aggregation for Graph Summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (pp. 567–580).
Toivonen, H., Mahler, S., & Zhou, F. (2010). A Framework for Path-Oriented Network Simplification. In *International Symposium on Intelligent Data Analysis* (pp. 220–231).
Toivonen, H., Zhou, F., Hartikainen, A., & Hinkka, A. (2011). Compression of Weighted Graphs. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 965–973).
Viana, M. P., Amancio, D. R., & Costa, L. d. F. (2013). On time-varying collaboration networks. *Journal of Informetrics*, *7*(2), 371–378.
Wang, G., Zhao, Y., Shi, X., & Yu, P. S. (2012). Magnet Community Identification on Social Networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 588–596).
Wang, X., Tang, L., Gao, H., & Liu, H. (2010). Discovering Overlapping Groups in Social Media. In *Proceedings of the 2010 IEEE International Conference on Data Mining* (pp. 569–578).
Zhou, F., Toivonen, H., & King, D. R. (2014). The Use of Weighted Graphs for Large-Scale Genome Analysis. *PLoS ONE*, *9(3)*, e89618.
Zhu, F., Qu, Q., Lo, D., Yan, X., Han, J., & Yu, P. S. (2011). Mining Top-K Large Structural Patterns in a Massive Network. *Proceedings of the VLDB Endowment*, *4*(11), 807-818.
Zhu, F., Zhang, Z., & Qu, Q. (2013). A Direct Mining Approach to Efficient Constrained Graph Pattern Discovery. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (pp. 821–832).

9. Appendices

Appendix A. Proof of the problem complexity.

We show the weighted network summarization problem to be NP-hard by a reduction from the max-cut problem.

Theorem A.1: *The edge-based weighted network summarization problem, the path-based*



(a) The original network G and its max-cut

(b) A summarized network S that corresponds to the max cut

(c) The restored network $res(S)$

Figure A1. The correspondence between a cut and a summarization to two supernodes.

weighted network summarization problem, and the node-based weighted network summarization problem are NP-hard.

Proof. In general, a problem is NP-hard if a special case of the problem is NP-hard. Because the edge-based summarization problem is a special case of both the path-based problem (when maximum path length $\lambda = 1$) and the node-based problem (when $\mathcal{I}(u) = 1$ for all nodes $u \in V$ and no nodes are deleted), by proving the NP-hardness of the edge-based problem we also prove the two other problem variants. We prove the complexity by reducing the max-cut problem, a well-known NP-complete problem (Karp (1972)), to the edge-based weighted network summarization problem.

A *cut* partitions the nodes of a given network into two disjoint sets. All edges that connect the two parts are cut, and the number of such edges is called the *size* of the cut. A *max-cut* is a cut whose size is at least the size of any other cut. Given a network $G = (V, E)$, the *max-cut problem* is to find a max-cut of the network.

The reduction to the network summarization problem consist of two steps. (1) Establishing an identity between a cut and a summarized network that consists of two supernodes. (2) Setting edge weights so that the distance between such a summarized network and the original one corresponds to the size of the cut. Under the conditions to be specified below, finding an optimal summarization solves the NP-complete max-cut problem.

(1) A summarized network consisting of exactly two supernodes directly identifies a cut into the two supernodes. For instance, the summarization in Figure A1(b) implies the cut shown in Figure A1(a).

The size of the cut is determined by the number of edges between the two supernodes in the original network. Therefore, we require that a superedge exists between the supernodes in the summarized network and will use it to compute the cut size in the next step. Edges within supernodes are not relevant for the cut size, so we do not allow self-edges in the summarized network.

In short, the summarized network we are looking for consists of two supernodes and a superedge between them. The compression ratio of the network is then $1/|E|$ (cf. Equation 1), so the same value can also be used as the maximum compression parameter cr: finding a summarized network with compression ratio at most $cr = 1/|E|$ and with no self-edges corresponds to finding a cut.

(2) Given such a summarized network, the size of the corresponding cut equals the number of original edges between the two supernodes. We obtain the cut size as the distance between the summarized and the original network as follows.

(a) We assign all edge weights in the original network to one, i.e., $w(u, v) = 1$ for

all $\{u, v\} \in E$ (cf. Figure A1(a)).

(b) We assign an edge weight in the restored network to one if the edge is in the cut (it is an original edge represented by the superedge), and to zero otherwise: $w'(\{u', v'\}) = 1$ if and only if both $\{u', v'\} \in E$ and $\{u', v'\} \in E''$ hold, otherwise $w'(\{u', v'\}) = 0$ (cf. Figure A1(c)).

As a result, bigger cuts result in distances closer to zero while smaller cuts give distances closer to $\sqrt{|E|}$ (cf. Equation 3). Since the distance is monotone increasing in the size of the cut, a summarization with the smallest distance corresponds to the largest cut size.

As a summary, the max-cut problem can be reduced to the edge-based weighted network summarization problem by setting $cr = 1/|E|$, disallowing self-edges in the summarized network, and by setting the weights in the restored network as defined above. This way of setting the weights is not identical to the operations we proposed in the paper, but this choice has no impact on the hardness of the problem.

The reduction is in polynomial time, which implies that if we had a polynomial time algorithm for the edge-based weighted network summarization problem, then we would have a polynomial time algorithm for the max-cut problem and P=NP. We therefore have that the edge-based summarization problem is NP-hard. As stated in the beginning of the proof, this implies that the path-based network summarization problem and the node-based network summarization problem also are NP-hard.

□

Appendix B. The algorithm of the node-pair merge operation.

The node-pair merge operation is specified in Algorithm 3. It takes a network and its two nodes as parameters, and it returns a network where the given nodes are generalized into one and the edge weights of the new supernode are set according to Equation 9. Line 9 of the merge operation sets the weight of the self-edge for the supernode. In edge-based network summarization, $\lambda = 1$, and function $W(x, y)$ returns the sum of weights of all original edges between x and y (as presented in the summarized network) using their mean weight $Q_1(\{x, y\}; S_{i-1})$. The weight of the self-edge is then zero and the edge non-existent if neither u nor v has a self-edge and if there is no edge between u and v. In the path-based variant, $\lambda = \infty$, and function $W(x, y)$ returns the connectivities between x and y using $Q_\infty(\{x, y\}; S_{i-1})$.

Appendix C. Complexity analysis of the algorithms.

We analyze the time complexity of the proposed algorithms, considering 2-hop optimization. Let d be the maximum node degree in the original network.

Brute-force algorithm. The method considers $O(d^2|V|)$ node pairs in each of $O(|V|)$ iterations. Thus there are $O(d^2|V|^2)$ merge operations and distance evaluations that have to be done.

In the edge-based variant, the merge operation has complexity $O(d)$ and distance re-evaluation can be done in $O(d)$ time. The total time complexity is then $O(d^3|V|^2)$.

In the path-based variant, the complexity of the merge operation is also $O(d)$. Computing the single-source best paths takes $O(|E| \log |V|)$ if we use Dijkstra's algorithm. Thus, the total time complexity $O(d^2|V|^2|E| \log |V|)$.

Algorithm 3 merge(u', v', S_{i-1})

Input: Two nodes u' and v', a (summarized) network $S_{i-1} = (V_{i-1}, E_{i-1}, w_{i-1})$, and a path quality function $Q_\lambda(\cdot)$.

Output: A summarized network $S_i = (V_i, E_i, w_i)$ obtained by merging u' and v' in S_{i-1}.

$V_i \leftarrow V_{i-1}$; $E_i \leftarrow E_{i-1}$; $w_i \leftarrow w_{i-1}$;
$z' \leftarrow \{u' \cup v'\}$;
$V_i \leftarrow V_i \setminus \{u', v'\} \cup \{z'\}$;
for all $x' \in V_i$ s.t. $u' \neq x' \neq v'$, and $\{u', x'\} \in E_{i-1}$ or $\{v', x'\} \in E_{i-1}$ **do**
  $w_i(z', x') = \frac{|u'|Q_\lambda(\{u',x'\};S_{i-1})+|v'|Q_\lambda(\{v',x'\};S_{i-1})}{|u'|+|v'|}$;
  $w_i(u', x') = 0; w_i(v', x') = 0$;
  $E_i \leftarrow E_i \setminus \{\{u', x'\}, \{v', x'\}\} \cup \{\{z', x'\}\}$;
end for
$w_i(\{z', z'\}) = \frac{W(u',u')+W(v',v')+W(u',v')}{|z'|(|z'|-1)/2}$
return $S_i = (V_i, E_i, w_i)$

function $W(x, y)$:
if $x \neq y$ **then**
  return $Q_\lambda(\{x, y\}; S_{i-1})|x||y|$;
else
  return $Q_\lambda(\{x, x\}; S_{i-1})|x|(|x| - 1)/2$;
end if

In the node-based variant, the merge operation has complexity $O(d)$, distance re-evaluation in the worst case can be done in $O(|V|)$ time, and the time complexity of edge deletion is $O(d|V|)$. The total time complexity thus is $O(d^3|V|^3)$.

Thresholded algorithm. In the worst case, the thresholded algorithm reduces to the brute-force greedy method. To find a pair with $ASSESS$-value below the threshold, the algorithm may consider at most $O(d^2|V|)$ pairs. For every pair considered the algorithm computes $ASSESS$. Since we make less than $|V|$ merges, we compute $ASSESS$ at most $O(d^2|V|^2)$ times.

The time complexity of the thresholded algorithm for the edge-based variant is $O(d^3|V|^2)$. The time complexity for the path-based variant is $O(d^2|V|^2|E| \log |V|)$. In the node-based variant, the time complexity is $O(d^3|V|^3)$.

Semi-greedy algorithm. This semi-greedy algorithm is $O(|V|)$ times faster than the brute-force greedy method, since the optimization is performed in each iteration over a set of nodes, not over all pairs of nodes. The resulting worst-case complexity in the edge-based variant is then $O(d^3|V|)$. In the path-based weighted variant, the total time complexity is $O(d^2|V||E| \log |V|)$. In the node-based weighted variant, the time complexity is $O(d^3|V|^2)$.

Random algorithm. The random algorithm only performs $O(|V|)$ merge operations, so time is spent on that instead of $d_{\max}(\cdot)$-value computations. The total complexity is $O(d|V|)$ in the edge-based variant, and is $O(|V||E| \log |V|)$ in the path-based variant. The time complexity for the node-based variant is $O(d|V|^2)$.

Appendix D. A Tool for Understanding the Contingency of Evolution of Prokaryotic Species.

We here describe an already published biological application of node-based weighted network summarization (Zhou et al. (2014)). The goal of this work was to compare the large-scale metabolisms in *Archaea* and *Eubacteria*, the two most fundamental branches (domains) of life, to gain understanding of how different pathways have evolved since their divergence.

Thousands of complete genomes have been sequenced. The aim was to use their inferred metabolic networks to better understand evolution, by comparing the networks across the two domains. In order to simplify the comparison of metabolisms of thousands of species, we proposed to integrate the metabolisms of each domain into one weighted network, weighing nodes by their importance in the metabolic networks of that domain. Three proposed node weights are: *taxonomic* weights, summarizing the phylogenetic importance of enzymes; *isoenzymatic* weights, summarizing the enzymatic variety of metabolism; and *sequence similarity* weights, summarizing the sequence conservation of metabolism.

The weighted metabolic networks were analyzed by using the node-based network summarization technique. Our hypothesis is that comparing summarizatios across and between the two biological domains may help us understand the biodiversity and evolution of prokaryote metabolisms.

The average taxonomic weight of the nodes remaining in the summarized network were calculated, over different compression ratios, in both domains. As expected, the average enzyme weight increases with more summarization, which implies that the enzymes left in the summarized network are more important to the domain. Then, the average taxonomic weight in the summarized part shared by both domains was computed. The results show that the average weight is higher than the average weight in the whole summarized network, which implies that the shared enzymes are more common in both domains. We finally investigated the similarity of those summarized pathways that are sub-parts of the metabolism to the original ones. The results show that there are highly significant and positive correlations of summarization results of pathways between domains in all three types of weights. This means that the pathways that are important in one domain are also important in the other domain. All these results provide evidence for the conservation of evolution.