Data Min Knowl Disc (2017) 31:1331–1358
DOI 10.1007/s10618-017-0505-2

Lagrangian relaxations for multiple network alignment

Eric Malmi[1] · **Sanjay Chawla**[2] · **Aristides Gionis**[1]

Received: 30 March 2016 / Accepted: 24 March 2017 / Published online: 30 March 2017

Abstract We propose a principled approach for the problem of aligning multiple partially overlapping networks. The objective is to map *multiple* graphs into a single graph while preserving vertex and edge similarities. The problem is inspired by the task of integrating partial views of a family tree (genealogical network) into one unified network, but it also has applications, for example, in social and biological networks. Our approach, called FLAN, introduces the idea of generalizing the *facility location* problem by adding a non-linear term to capture edge similarities and to infer the underlying entity network. The problem is solved using an alternating optimization procedure with a Lagrangian relaxation. FLAN has the advantage of being able to leverage prior information on the number of entities, so that when this information is available, FLAN is shown to work robustly without the need to use any ground truth data for fine-tuning method parameters. Additionally, we present three multiple-network extensions to an existing state-of-the-art pairwise alignment method called NATALIE. Extensive experiments on synthetic, as well as real-world datasets on social networks and genealogical networks, attest to the effectiveness of the proposed

Responsible editors: Thomas Gärtner, Mirco Nanni, Andrea Passerini and Celine Robardet.

✉ Eric Malmi
eric.malmi@aalto.fi

Sanjay Chawla
schawla@qf.org.qa

Aristides Gionis
aristides.gionis@aalto.fi

1 HIIT, Aalto University, Espoo, Finland

2 Qatar Computing Research Institute, HBKU, Doha, Qatar

approaches which clearly outperform a popular multiple network alignment method called IsoRankN.

Keywords Multiple network alignment · Facility location · Lagrangian relaxation · Genealogical trees · Social networks

1 Introduction

The multiple network alignment problem encodes the task of de-duplicating vertices in a collection of graphs while preserving similarity between vertices and edges. Vertex similarity is typically modeled by comparing vertex attributes or feature vectors. Edge similarity encodes structural dependencies among the input graphs, in particular, a desirable network alignment should preserve the edges of the input graphs to the largest extent.

A need for data de-duplication often arises when interrelated datasets from different sources have to be integrated. Many datasets naturally have a network structure which is why the potential application areas of network alignment methods are plentiful. These methods have attracted a significant amount of attention in the area of biological networks (Clark and Kalita 2014), in particular for the problem of aligning protein–protein interaction networks in order to identify functional orthologs across different species (Elmsallati et al. 2015). Other applications include social network alignment (Goga et al. 2015; Zhang and Yu 2015), ontology alignment (Bayati et al. 2013), and image matching in computer vision (Conte et al. 2004).

However, our initial motivation to study this problem arises from the application of merging family trees (genealogical networks) which is a common problem for genealogists. Services like *Ancestry.com* and *MyHeritage* attract millions of paying subscribers who upload their own family tree to the service, trying to find new relatives to add to their network. Automatic alignment of individual family trees can help people to find new relatives and trace their ancestry further back in time. Consider for example the top half of Fig. 1, which shows three individuals, A, B and C, and the partial views they have on their ancestry. The bottom half shows the underlying family tree which is hidden and unknown. The trees contain two types of information: vertex attributes (not shown in the figure) and relationships between vertices. Due to the difficulty of interpreting historical documents and errors in these documents, the vertices and edges of individual trees may contain errors. Furthermore, family trees are not trees in the graph-theoretic sense as they contain cycles such as *Mother–FirstChild–Father–SecondChild–Mother*. Note that we draw an edge between every parent-child pair, whereas Fig. 1 follows the layout commonly used by genealogists. In conclusion, the problem of merging family trees is an instance of the multiple network alignment problem.

To address the multiple network alignment problem, we introduce a novel extension of the *facility location* problem (Vazirani 2001) to account for both vertex and edge similarity. In particular, we present a non-linear extension of facility location to specifically favor mappings where neighboring vertices are mapped to other neighbors. To the best of our knowledge, this extension is a novel problem in its own right and of

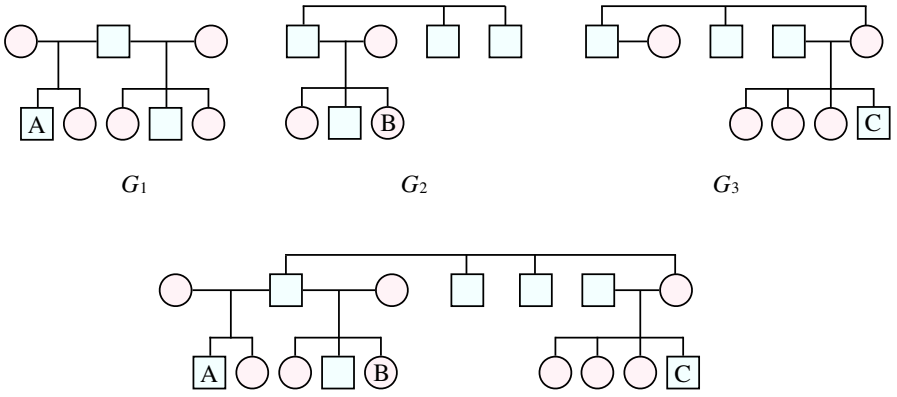

Fig. 1 The *bottom half* of the figure shows an underlying but unknown family tree, indicating the location of three individuals A, B and C. The *square vertices* represent males and the round vertices females. Each individual has a partial view of the tree. The objective is to reconstruct the underlying and hidden tree from the partial views shown in the *top half*

independent interest. We refer to the extension as the facility location formulation for aligning multiple networks (FLAN). Since FLAN is **NP**-hard, we provide an approximate solution using a Lagrangian relaxation approach. A practical benefit of FLAN is that it allows the user to fix the number of entities (i.e., vertices in the hidden graph) in cases when prior information on that is available. This can help with parameter selection which is a common problem when applying network alignment methods in practice.

Natalie, proposed by Klau (2009), El-Kebir et al. (2015), is one of the best performing existing network alignment methods (Clark and Kalita 2014). Similar to our approach, Natalie formulates the pairwise problem as a non-linear integer program and provides an approximate solution using a Lagrangian relaxation approach. However, Natalie only supports pairwise alignments. Therefore, we investigate three approaches to extending Natalie to multiple networks.

Finally, we present an experimental comparison between FLAN and the extensions of Natalie on synthetic data, social network data, and family tree data. We demonstrate that FLAN performs well in all of these experiments, especially in terms of precision, whereas the proposed Natalie extensions typically yield the highest recall. Furthermore, if prior information on the number of entities is available, FLAN works robustly without any ground truth data which is typically needed for fine-tuning method parameters.

To summarize, our main contributions are:

- We formalize the multiple network alignment problem using a novel non-linear extension of the facility location problem. This extension captures both the problem of inferring the vertices referring to the same entity and the problem of inferring the underlying entity network. We refer to the extension as the facility location formulation for aligning multiple networks (FLAN).
- We propose an alternating optimization approach to obtain an approximate solution of FLAN, using the technique Lagrangian relaxation. The advantage of the

Lagrangian relaxation is that we automatically obtain instance-level approximation bounds on the quality of the solution.
- If prior information on the number of entities is available, FLAN is shown to work robustly without the need to use any ground truth data for fine-tuning method parameters.
- We present three multiple-network extensions to Natalie, which is a state-of-the-art pairwise network alignment method. A progressive extension with edge updates (PROGNATALIE++) is shown to provide a good experimental performance.
- The code and the data for reproducing the experiments are publicly available at: https://github.com/ekQ/flan

The rest of the paper is structured as follows. In Sect. 2, we present the facility location formulation for the multiple network alignment problem, and in Sect. 3, we describe the Lagrangian relaxation approach for solving it, as well as the extensions of Natalie to multiple networks. In Sect. 4, we discuss the related work, and in Sect. 5, provide an experimental evaluation of the different methods. Finally, we draw conclusions in Sect. 6.

2 Problem formulation

The input to the network-alignment problem consists of k graphs $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$. We define $V = \bigcup_{i=1}^{k} V_i$ to be the set of all vertices in the k input graphs and we set $n = |V|$. As the correspondence between the graph vertices is unknown initially, we assume that all vertex sets are pairwise disjoint. The case that some correspondences among vertices of input graphs are known can be easily incorporated in our framework.

We assume that the input graphs are manifestations of an underlying *entity graph* $G_e = (U, E_e)$, where the vertex set U denotes the underlying entities and E_e the edges between them. The objective of the network-alignment problem is to infer the entity graph and find an assignment $\mathcal{X} : V \to U$ so that each vertex in the input graphs can be mapped to its underlying entity vertex. We further assume that the entities are represented by a subset of the vertices in the input graphs, that is, $U \subseteq V$.

The alignment of the input graphs is driven by the graph structure, so that to the largest extent possible, neighbors in the input graphs should map to neighbors in the entity graph, as well as by the similarity between vertices in different graphs. In particular, we assume that each vertex has a set of attributes. A distance function (dissimilarity) $d(i, j)$ is then defined between each pair of vertices i and j. The distance $d(i, j)$ is derived by comparing the attributes of i and j.

For the relationship between the input graphs and the underlying graph we consider the following characteristics:

- The attributes of a vertex of an input graph may have been distorted from the attributes of the corresponding entity of the underlying graph.
- The vertices of an input graph may correspond to only a subset of the entities ($V_i \subseteq U$).
- The edges between entities are preserved with probability $p < 1$ (not necessarily fixed for all edges) so the edges of an input graph may correspond to only a

subset of the edges in the underlying entity graph. In other words, the entity graph contains the edges of the input graphs and potentially some missing edges, that is, $E_e = \left(\bigcup_{i=1}^{k} E_i\right) \cup E_m$, where E_m is a set of potentially missing edges.

From the above considerations it follows that the assignment $\mathcal{X} : V \to U$ we are searching for should satisfy the following properties:

(P1) Vertices are mapped to entities with as similar attributes as possible.
(P2) Adjacent vertices are assigned to adjacent entities.

To find a set of entities U, the edges between them E_e, and an assignment $\mathcal{X} : V \to U$ that respects properties (P1) and (P2) we formulate a non-linear integer programming (IP) problem. The IP formulation is an optimization problem over binary variables y_i and x_{ij} and an adjacency matrix B. The first two variables encode a solution in terms of the sought set U and the assignment $\mathcal{X}$, respectively. In particular, y_i indicates whether there is some vertex $j \in V$ that has been assigned to entity i, and x_{ij} indicates whether vertex i is assigned to entity j. The binary adjacency matrix B encodes the missing edges E_m to be optimized and the edges of the input graphs $E_I = \bigcup_{i=1}^{k} E_i$. The integer program is the following

$$\min_{x,y,B} \quad \sum_j f y_j + \sum_{i,j} d(i,j) x_{ij} - g \sum_{i,j,k,\ell} A_{ik} B_{j\ell} x_{ij} x_{k\ell} + \gamma \sum_{(i,j)\notin E_I} B_{ij}^2, \tag{1}$$

$$\text{such that} \quad x_{ij} \le y_j, \quad i, j = 1, \ldots n, \tag{2}$$

$$\sum_j x_{ij} = 1, \quad i = 1, \ldots, n, \tag{3}$$

$$\sum_{i \in V_m} x_{ij} \le 1, \quad j = 1, \ldots, n \text{ and } m = 1, \ldots, k, \tag{4}$$

$$x_{ij}, y_j \in \{0, 1\}, \quad i, j = 1, \ldots, n. \tag{5}$$

In the above formulation, f, g, and γ are scalar parameters, while A and B are adjacency matrices representing the graph structure of the problem instance. In particular, entry A_{ik} indicates whether the vertices i and k of the input graphs are neighbors, while entry $B_{j\ell}$ indicates whether entities j and ℓ are neighbors.

The integer program presented above can be seen as a non-linear extension of the *uncapacitated facility-location* problem (Hochbaum 1982). Selecting vertex $i \in V$ to be an entity so that other vertices can be mapped to it corresponds to setting $y_i = 1$, which can be seen as *opening* vertex i as a facility. The parameter f represents the cost of opening a facility, so the first term in the objective function (1) penalizes for every opened entity. Note that one extreme solution is to consider every vertex as an entity; setting $f = 0$ would make such a solution optimal.

The second term in the objective function (1) penalizes for assigning vertices to dissimilar entities. Recall that $d(i, j)$ is the distance between vertex i and entity j, computed using the attributes of i and j, and note that the cost is paid only when vertex

i is assigned to entity j, expressed by $x_{ij} = 1$. The third term uses the adjacency matrices A and B and gives a discount of g for each pair of adjacent vertices that are assigned to adjacent entities.

The fourth term adds L_2 regularization to the adjacency matrix B by introducing cost γ for every added missing edge. Parameter γ controls the amount of evidence needed for introducing new edges to the entity graph; if $\gamma = 0$, the complete graph becomes the optimal solution for B, whereas if $\gamma = \frac{g}{2}$, a new edge is added when at least one pair of adjacent vertices has been assigned to the corresponding entity pair, as shown later in Sect. 3.3.5.

Next we discuss the constraints of the integer program given in (2)–(4). The first set of constraints (2) ensures that vertices are only assigned to opened entities. The second set of constraints (3) ensures that each vertex is assigned to exactly one entity. Finally, the third set of constraints (4) prevents two vertices in the same input graph from being assigned to the same entity.

An alternative formulation results by having prior knowledge about the total number of entities N_e. In this case we can set a constraint for the number of opened entities, so the objective function (1) is replaced by

$$\min_{x,y,B} \quad \sum_{i,j} d(i,j)x_{ij} - g \sum_{i,j,k,\ell} A_{ik}B_{j\ell}x_{ij}x_{k\ell} + \gamma \sum_{(i,j)\notin E_I} B_{ij}^2, \tag{6}$$

and we need to include constraint

$$\sum_i y_i \leq N_e, \tag{7}$$

on top of constraints (2)–(5). Both problem formulations require three parameters as input: the first formulation requires f, g, and γ, while the second formulation requires N_e, g, and γ.

For both of these formulations we have the following result.

Proposition 1 *Multiple network alignment, as defined by optimizing objective function (1) subject to constraints (2)–(5), or optimizing objective function (6), subject to constraints (2)–(5) and (7), is* ***NP****-hard.*

Proof Following the proof for the **NP**-hardness of pairwise network alignment by El-Kebir et al. (2015), we show a reduction from the CLIQUE decision problem, which asks to determine whether a k-vertex clique $G_c = (V_c, E_c)$ exists in graph $G = (V, E)$. We consider an instance of the multiple network alignment problem with two networks, i.e., $k = 2$, which are set to $G_1 = G$ (the input graph for the CLIQUE problem) and $G_2 = G_c$ (the k-clique). Let $f = 0$ (or $N_e = \infty$ if we consider the problem with the fixed number of entities) and

$$d(u,v) = \begin{cases} 0, & \text{if } (u = v \text{ and } u \in V) \text{ or } (u \in V_c \text{ and } v \in V) \\ \infty, & \text{otherwise,} \end{cases}$$

so that the vertices of G are encouraged to be assigned to themselves, whereas the vertices in G_c are encouraged to be assigned to any vertices in V.

A k-vertex clique exists in G if and only if the cost of the optimal multiple network alignment for input graphs G and G_c is $-g\,(|E| + |E_c|)$. This cost corresponds to the sum of the discounts we get from aligning each neighbor pair in G to itself and being able to align each neighbor pair in G_c to another neighbor pair in G. Note that the same pairwise cost could be achieved by aligning the neighbor pairs in G_c to non-neighbor pairs in G and adding edges between the corresponding entities. However, this would result in a higher overall cost due to the regularization term $\gamma \sum_{(i,j)\notin E_I} B_{ij}^2$ when $\gamma > 0$. □

3 Methods

We now discuss our methods for solving the integer programs introduced in the previous section. Our algorithms (Sect. 3.3) use the Lagrangian relaxation framework, so we first give a brief overview of the framework (Sect. 3.1). The Lagrangian relaxation framework is also used by Natalie by Klau (2009), El-Kebir et al. (2015) for aligning two networks. We present Natalie and propose an extension of it to multiple networks in Sect. 3.2.

3.1 Background: Lagrangian relaxation framework

The Lagrangian relaxation approach (Fisher 1981) aims to obtain approximate solutions to constrained optimization problems, like the ones presented in the previous section. The method dualizes/relaxes some constraint(s) by adding them to the objective with multipliers λ. In practice, the constraint(s) to be relaxed are chosen so that the relaxed problem $Z_{LD}(\lambda)$ can be solved in polynomial time.

Next, the problem $Z_{LD}(\lambda^*) = \max_\lambda Z_{LD}(\lambda)$ is solved using the subgradient method (Shor 2012). Since the value of $Z_{LD}(\lambda)$ is a lower bound for the original problem for any λ, the value of $Z_{LD}(\lambda^*)$ yields a lower bound (ℓ^*) for the optimal solution. Every relaxed solution computed during the subgradient optimization is modified using some heuristics to construct feasible solutions for the original non-relaxed problem. The feasible solution with the lowest cost provides an upper bound (u^*) for the optimal solution. If $\ell^* = u^*$, then the optimal solution for the original problem has been found. Even for many **NP**-hard problems, the optimal solution is often found in a reasonable time with this approach (Fisher 1981).

3.2 NATALIE

NATALIE is a state-of-the-art pairwise (i.e., $k = 2$) network-alignment method introduced by Klau (2009). It formulates the two-network alignment problem as the following integer program

$$\max_{x} \quad \sum_{(i,j)\in V_1\times V_2} \sigma(i,j)x_{ij} + \sum_{(i,j)\in V_1\times V_2} \sum_{(k,\ell)\in V_1\times V_2} \tau(i,j,k,l)x_{ij}x_{k\ell},$$

$$\text{such that} \quad \sum_{j\in V_2} x_{ij} \leq 1, \quad \text{for all } i \in V_1,$$

$$\sum_{i\in V_1} x_{ij} \leq 1, \quad \text{for all } j \in V_2,$$

$$x_{ij} \in \{0,1\}, \quad \text{for all } (i,j) \in V_1 \times V_2,$$

where $\sigma(i, j)$ is a similarity score between vertices i and j. The parameter $\tau(i, j, k, \ell)$ is a similarity score between pairs of vertices (i, k) and (j, ℓ) which is typically set to

$$\tau(i,j,k,\ell) = \begin{cases} g, & \text{if } (i,k) \in E_1 \text{ and } (j,\ell) \in E_2 \\ 0, & \text{otherwise,} \end{cases}$$

where g is a positive constant.

This formulation is equivalent to the formulation presented in Sect. 2 applied to two networks. The main difference is that we have introduced an entity-opening cost f (or alternatively a budget N_e) to control the number of entities when aligning partially overlapping networks. NATALIE also supports partial alignment since a vertex is not required to be matched to another vertex. Instead it can get mapped to a gap if some of the scores $\sigma(i, j)$ are negative. The number of aligned vertices can be controlled by shifting the scores towards more negative or more positive values. In our implementation of NATALIE,[1] which is used for the experiments of this paper, we set a threshold score for mapping a vertex to a gap instead of shifting the scores. In our experiments, this threshold is also denoted by f for consistency.

To solve the integer program, NATALIE first linearizes it and then employs a Lagrangian relaxation approach. The derivation is similar to the one presented in Sect. 3.3.2 but due to the additional term in the objective (or alternatively an additional constraint) in our formulation, we need to relax two constraints instead of one and develop feasibility heuristics (Sect. 3.3.3) whereas NATALIE's formulation allows to directly extract a feasible solution from a relaxed one.

NATALIE 2.0 (El-Kebir et al. 2015) is an extension of the original algorithm (Klau 2009). The original method adopts a subgradient method for updating the Lagrangian multipliers, whereas NATALIE 2.0 employs both the subgradient method and a dual descent method to obtain stronger upper and lower bounds. Nevertheless, in this paper we only consider the subgradient approach for updating the multipliers.

In summary, the main differences between our method and NATALIE are the following: Our method directly supports multiple network alignment (as well as pairwise alignment) and it also optimizes the underlying entity graph, but these improvements come with the cost of having to solve a more complex optimization problem. Both methods require parameters f and g, but our method also supports specifying the num-

[1] The code is available at: https://github.com/ekQ/flan.

ber of vertices N_e instead of specifying f. In addition, our method requires parameter γ which is set to $\frac{g}{2}$ as discussed later in Sect. 3.3.5.

3.2.1 Adaptation to multiple networks

Klau (2009) suggests that NATALIE can be extended to multiple networks or alternatively it can be used to progressively align multiple networks. In this section, we present one possible extension to multiple networks and propose an improvement for the straightforward progressive version of NATALIE.

To find a multiple network alignment, we assume an ordering of graphs and consider aligning vertex i with any vertex from graphs $g' = 1, \ldots, g-1$. Thus we obtain the following problem.

$$
\begin{aligned}
\max_{x} \quad & \sum_{1 \le g' < g \le k} \sum_{(i,j) \in V_g \times V_{g'}} \sigma(i,j) x_{ij} \\
& + \sum_{1 \le g' < g \le k} \sum_{(i,j) \in V_1 \times V_2} \sum_{(k,\ell) \in V_1 \times V_2} \tau(i,j,k,\ell) x_{ij} x_{k\ell}, \\
\text{such that} \quad & \sum_{j \in V_{g'}} x_{ij} \le 1, \quad \text{for all } 1 \le g' < g \le k, i \in V_g, \\
& \sum_{i \in V_g} x_{ij} \le 1, \quad \text{for all } 1 \le g' < g \le k, j \in V_{g'}, \\
& x_{ij} \in \{0,1\}, \quad \text{for all } 1 \le g' < g \le k, (i,j) \in V_g \times V_{g'},
\end{aligned}
$$

NATALIE can be used to solve this problem with the following modifications: (i) if a vertex is mapped to a gap, we define it to be mapped to itself, and (ii) the last bipartite matching step (for details, see Klau 2009) must be done for each input graph separately since two vertices from different graphs can be mapped to the same vertex even though two vertices from the same graph cannot.

To avoid obtaining too many entities, we assume transitivity and define that if $x_{ab} = x_{bc} = 1$, then vertices a, b, and c should all belong to the same entity. Hence, entities can be extracted by finding connected components of the alignment graph. However, the transitivity assumption comes with the drawback that we cannot guarantee anymore that vertices a and a' from the same input graph are mapped to a distinct entity since we might have that $x_{ab} = 1$ and $x_{a'c} = x_{cb} = 1$. Avoiding such injectivity violations does not seem trivial, so in the experiments, we simply ignore the injectivity constraint in case the aforementioned situation occurs.

This method is called NATALIE in the experiments.

To solve the multiple networks alignment problem progressively, we pick a target graph and solve $k-1$ pairwise network alignment problems using NATALIE to align other graphs to it. For every vertex mapped to a gap, we create a new vertex in the target graph so that the vertices of subsequent graphs can be mapped to it. This method is called PROGNATALIE.

One limitation of PROGNATALIE is that the edges between the original vertices and the newly created vertices in the target graph are entirely absent. Furthermore, since the edge sets are noisy, it would be useful to be able to aggregate edge information across the graphs. Therefore, we propose PROGNATALIE++ which updates target graph edges after every aligned input graph: for each pair of neighboring input graph vertices mapped to vertices j and l in the target graph, we create an edge (j, ℓ) if it is not already present.

3.3 Our approach: FLAN

Before discussing our method in detail, we present a high-level overview. We adopt an alternating optimization procedure (Bezdek and Hathaway 2003) which splits the variables into two subsets $\{x, y\}$ and $\{B\}$, and iteratively solves the alternating restricted minimization problems over the two subsets. The method consists of the following steps.

1. Initialize the entity graph $B = A$.
2. Keeping B fixed, solve the optimal alignment x, y as follows
   - Linearize the problem to obtain an integer linear programming (ILP) problem (Sect. 3.3.1).
   - Solve the integer linear program using a Lagrangian relaxation approach (Sects. 3.3.2, 3.3.3, and 3.3.4).
3. Optimize the adjacency matrix B of the entity graph, keeping x and y fixed (Sect. 3.3.5).
4. Go back to step 2 unless the iteration has converged.

3.3.1 Linearizing the problem

The first step is to eliminate quadratic terms, and turn the quadratic integer program into an integer linear program. This is achieved by introducing new variables $w_{ijk\ell} = x_{ij}x_{k\ell}$. Note that we only need to create a variable $w_{ijk\ell}$ for index quadruplets that "form a square" in the input graphs and the entity graph (i.e., vertices i and k are neighbors and entities j and ℓ are neighbors) since in all other cases $A_{ik}B_{j\ell} = 0$ and $w_{ijk\ell}$ does not play a role. We denote by $\mathcal{S}$ the set of quadruplet indices for which a variable $w_{ijk\ell}$ is introduced.

To ensure that the definition of variables $w_{ijk\ell}$ is consistent, we introduce the following constraints

$$\sum_{\ell:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} = \sum_{\ell:(i,j,k,\ell)\in\mathcal{S}} x_{ij}x_{k\ell} \leq \sum_{\ell} x_{ij}x_{k\ell} = x_{ij}, \quad \text{for all } i, j, k \tag{8}$$

$$\sum_{k:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} = \sum_{k:(i,j,k,\ell)\in\mathcal{S}} x_{ij}x_{k\ell} \leq \sum_{k\in V(i)} x_{ij}x_{k\ell} \leq x_{ij}, \quad \text{for all } i, j, \ell \tag{9}$$

$$w_{ijk\ell} = w_{k\ell ij}, \quad \text{for all } i, j, k, \ell \tag{10}$$

where $V(i) = \{v : i \in V_j \text{ and } v \in V_j\}$, that is, the vertices of the input graph to which vertex i belongs.

After the linearization step and dropping the regularization term which does not affect the minimum when B is fixed, we obtain the following integer linear program, where B is fixed

$$\min_{x,y,w} \quad \sum_i f y_i + \sum_{i,j} d(i,j) x_{ij} - g \sum_{(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} \tag{11}$$

$$\text{such that} \quad x_{ij} \le y_j, \quad i,j = 1,\ldots n \tag{12}$$

$$\sum_j x_{ij} = 1, \quad i = 1,\ldots,n \tag{13}$$

$$\sum_{i\in V_m} x_{ij} \le 1, \quad j = 1,\ldots,n \text{ and } m = 1,\ldots,k \tag{14}$$

$$\sum_{\ell:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} \le x_{ij}, \quad \text{for all } i,j,k \tag{15}$$

$$\sum_{k:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} \le x_{ij}, \quad \text{for all } i,j,\ell \tag{16}$$

$$w_{ijk\ell} = w_{k\ell ij}, \quad \text{for all } i,j,k,\ell \tag{17}$$

$$x_{ij}, y_i, w_{ijk\ell} \in \{0,1\}, \quad \text{for all } i,j,k,\ell. \tag{18}$$

Note that the number variables in (11) is potentially very high, which could prevent us from solving problem instances of a realistic size. However, this shortcoming can be overcome by a technique known as *blocking* in the entity-resolution literature (Christen 2012). The idea is to consider that each vertex can be mapped not to every other vertex but only to a set of *candidate entities*. These candidates are typically determined by selecting entities above a similarity threshold or by taking the c most similar entities.[2] This is also known as *sparse* network alignment (El-Kebir et al. 2015; Bayati et al. 2013).

3.3.2 Solving an instance of the relaxed problem

To solve the integer linear program (11)–(18), we adopt the Lagrangian relaxation approach. We start by dualizing constraints (13) and (17), which yields the following problem

$$Z_{LD}(\lambda) = \min_{x,y,w} \quad \sum_i f y_i + \sum_{i,j} d(i,j) x_{ij} - g \sum_{(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} + \sum_i \lambda_i \left(1 - \sum_j x_{ij}\right) + \sum_{(i,j,k,\ell)\in\mathcal{S}} \lambda_{ijk\ell} \left(w_{ijk\ell} - w_{k\ell ij}\right)$$

[2] Like in the case of NATALIE, we assume an ordering of graphs and consider aligning vertex i with itself or any vertex from graphs $g' = 1,\ldots,g-1$. In other words, we avoid considering simultaneously vertex i as an entity for vertex j and j as an entity for i, which we have observed to result in larger duality gaps.

$$= \min_{x,y,w} \sum_i \lambda_i + \sum_i f y_i + \sum_{i,j} (d(i,j) - \lambda_i) x_{ij} + \sum_{(i,j,k,\ell)\in\mathcal{S}} (2\lambda_{ijk\ell} - g) w_{ijk\ell} \tag{19}$$

subject to constraints (12), (14)–(16), and (18).

Despite the fact that the relaxed problem has integral variables (as we have not relaxed constraint (18)), as we show next, the problem can be solved in polynomial time for any given λ.

Theorem 1 *The relaxed problem* (19) *can be solved in polynomial time.*

Proof The relaxed problem can be decomposed into two problems, so that the first one is over variables x and y, and the second one is over variables w. In particular, the relaxed problem can be written as

$$\begin{aligned} Z_{LD}(\lambda) \quad = \quad & \min_{x,y} \sum_i \lambda_i + \sum_i f y_i + \sum_{i,j} [d(i,j) - \lambda_i + v_{ij}(\lambda)] x_{ij} \\ & \text{such that} \quad x_{ij} \le y_j, \quad i,j = 1,\dots n \\ & \qquad\qquad x_{ij}, y_i \in \{0,1\} \quad i,j = 1,\dots n, \end{aligned} \tag{20}$$

where

$$\begin{aligned} v_{ij}(\lambda) \quad = \quad & \min_{w} \sum_{k,\ell:(i,j,k,\ell)\in\mathcal{S}} (2\lambda_{ijk\ell} - g) w_{ijk\ell} \\ & \text{such that} \quad \sum_{\ell:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} \le 1, \quad \text{for all } k \\ & \qquad\qquad \sum_{k:(i,j,k,\ell)\in\mathcal{S}} w_{ijk\ell} \le x_{ij}, \quad \text{for all } \ell \\ & \qquad\qquad w_{ijk\ell} \in \{0,1\}, \quad \text{for all } k,\ell. \end{aligned} \tag{21}$$

Notice that problems (20) and (21) are equivalent to the relaxed problem (19) despite the fact that in (20) the terms $v_{ij}(\lambda)$ are multiplied by x_{ij}. The reason is that all variables x_{ij} and $w_{ijk\ell}$ are either 0 and 1, and whenever a x_{ij} is 0, then $w_{ijk\ell}$ are also 0, for all k and ℓ.

Observe that a variable v_{ij} is defined for each x_{ij}, and the value of v_{ij} is given as a solution to the minimization problem (21). Given that the variables x_{ij} in the second constraint are either 0 and 1, it is not hard to see that the minimization problem (21), together with the corresponding constraints, is a *minimum-cost matching* problem. Thus the value of each v_{ij} can be computed using the *Hungarian algorithm*. Furthermore, given λ and x, all these minimization problems are independent, and thus, the value of each v_{ij} can be computed separately.

Once we have computed v_{ij}, we can solve Z_{LD} easily after observing that the optimal value of x_{ij} is given by

$$x_{ij} = \begin{cases} y_j, & \text{if } d(i,j) - \lambda_i + v_{ij}(\lambda) \leq 0 \\ 0, & \text{otherwise,} \end{cases}$$

enabling us to write

$$\begin{aligned} Z_{LD}(\lambda) &= \min_y \sum_i \lambda_i + \sum_i f y_i + \sum_j \left(\sum_i \min\{0, d(i,j) - \lambda_i + v_{ij}(\lambda)\} \right) y_j \\ &= \min_y \sum_i \lambda_i + \sum_i (f + C_i) y_i \\ &\text{such that} \quad y_i \in \{0,1\}, \end{aligned}$$

where $C_i = \sum_i \min\{0, d(i,j) - \lambda_i + v_{ij}(\lambda)\}$. Hence we get

$$y_i = \begin{cases} 1, & \text{if } f + \sum_j \min\{0, d(j,i) - \lambda_j + v_{ji}(\lambda)\} < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

If instead of setting cost f for opening an entity we want to set a constraint for the number of opened entities, the final minimization problem would take the form

$$\begin{aligned} Z_{LD}(\lambda) &= \min_y \sum_i C_i y_i \\ &\text{such that} \quad \sum_i y_i = N_e \quad \text{and} \quad y_i \in \{0,1\}. \end{aligned}$$

This problem can be solved by sorting y_i based on the coefficients C_i and setting $y_i = 1$ for the N_e smallest coefficients. □

In Sect. 3.3.4 we discuss how a solution to the problem $Z_{LD}(\lambda)$, for a given vector λ, is used within the Lagrangian relaxation method. Before that, we present our methods for finding a feasible solution to the multiple network-alignment problem from a solution to the relaxed problem (Sect. 3.3.3).

3.3.3 Finding a feasible network alignment

Once we have obtained the optimal solution x^*, y^*, w^* for the relaxed problem, we still need to obtain a feasible solution for the original network-alignment problem. Note that although the solution x^*, y^* of the relaxed problem is integral, it is not necessarily

feasible, since we have relaxed the constraint $\sum_j x_{ij} = 1$ and hence, some vertices may be mapped to zero or more than one entities.

We will next show that transforming the optimal solution x^*, y^* of the relaxed problem to a feasible solution for the original non-relaxed and non-linearized network-alignment problem with minimal additional cost in $Z_{LD}(\lambda)$ is an **NP**-hard problem. In fact, the problem is related to *set cover*, and thus, we propose an algorithm that is based on a *greedy* approach.

We will show that both of the versions we consider lead to an **NP**-hard problem: (i) when parameter f is given as input (and thus, the number of open entities depends on f); and (ii) when the number of opened entities N_e is given as input. Recall that in the former case, y^* is obtained by opening the entities with negative coefficients (according to (22)), while in the latter case y^* is obtained by opening the entities with the N_e smallest coefficients. As already mentioned, the optimal y^* may lead to some vertices not being assigned to any entities. The hardness of the problem of finding a feasible solution based on y^* stems from the fact that more (or different) entities need to be opened in order to ensure feasibility. Both versions of the problem (for fixed f and fixed N_e) can be compactly formulated as follows:

$$\min_y \quad \sum_i C'_i y_i, \tag{23}$$

$$\text{such that} \quad \text{a matching between the vertices and opened entities exists,} \tag{24}$$

$$\left(\sum_i y_i = N_e, \quad i = 1, \ldots, n, \right) \tag{25}$$

$$y_i \in \{0, 1\}, \quad i = 1, \ldots, n, \tag{26}$$

where $C'_i = f + C_i$ if f is fixed, and $C'_i = C_i$ if the number of entities N_e is fixed, and $C_i = \sum_i \min\{0, d(i, j) - \lambda_i + v_{ij}(\lambda)\}$ as defined in the previous section. Constraint (25) is only included in the case of fixing the number of entities N_e. The hardness result is stated and proven next.

Proposition 2 *Finding the optimal set of open entities, as defined in (23)–(26), is an **NP**-hard problem both when the entity-opening cost f is fixed, and when the number of entities N_e is fixed.*

Proof We give a reduction from the optimization version of the weighted set cover problem. We consider a special case where the number of sets is equal to the number of items to be covered, which still keeps the problem **NP**-hard. Let $\{1, 2, \ldots, n\}$ be a set of items to be covered, $\mathcal{S}$ a collection of n sets, and $\mathcal{M}(j)$ the sub-collection of sets that contain item j, for $j = 1, \ldots, n$. Each set S_i is associated with a cost C'_i.

By using variable y_i to denote whether set i is included in the cover or not, the weighted set cover problem can be written as

$$\min_y \quad \sum_i C'_i y_i, \tag{27}$$

$$\text{such that} \quad \sum_{c \in \mathcal{M}(j)} y_c \geq 1, \quad j = 1, \ldots, n, \tag{28}$$
$$y_i \in \{0, 1\}, \quad i = 1, \ldots, n.$$

Now we observe that this is a special case of problem (23) where each vertex is considered to be its own graph and $\mathcal{M}(i)$ denotes the set of candidate entities for vertex i, thus making constraints (24) and (28) equivalent. Furthermore, we could add constraint (25), without loss of generality, by setting $N_e = n$, which proves the result for both cases of fixed f and fixed N_e. □

The Lagrangian framework itself does not require us to find the optimum y, which we have shown to be **NP**-hard, but the further the feasible y we obtain is from the optimum, the larger the duality gap will be, and thus we want to come up with a reasonable way of finding an approximate solution.

Greedy methods are known to give good approximations for set cover problems and hence we adopt a greedy approach for finding y. Next we present a high-level overview of this approach—the details can be found from the publicly available source code.[3]

In the case of fixed N_e, we start opening entities one-by-one after ranking them primarily by how many vertices from different input graphs can be assigned to each entity and secondarily by the coefficient of each entity, until a matching between vertices and opened entities exists. Note that this greedy strategy is different than the usual "normalized cost" strategy that is used for weighted set cover, however, we prefer to prioritize the selection of entities based on the number of matching vertices, as we have a budget N_e on the number of entities that we can open. When all vertices are matched by at least one opened entity, we can still open extra entities based on the coefficients until the number of open entities is N_e. Finally, we find a feasible assignment x by matching the input graphs to the open entities with the Hungarian algorithm one graph at a time—note that all these matching problems are independent, so the order we process the graphs does not matter. The edge weights for the matching problem are given by $d(i, j) - \lambda_i + v_{ij}(\lambda)$ from (20).

In the case of fixed f, we initially set $y = y^*$. Then we start again matching input graphs one at a time and open extra entities along the way if a matching cannot be found otherwise.

3.3.4 Solving the full Lagrangian relaxation

Recall from Sect. 3.1 that for the full Lagrangian relaxation algorithm we wish to find λ as close to the optimal vector λ^*, that is, $Z_{LD}(\lambda^*) = \max_\lambda Z_{LD}(\lambda)$, as possible. Such a vector is found by the subgradient method (Shor 2012). In particular, we start with $\lambda = 0$, as well as a trivial upper bound $u^* = \infty$ and lower bound $\ell^* = -\infty$ to the optimal cost of the original problem.

Then we start an iterative process: in each iteration, we solve $Z_{LD}(\lambda)$, as described by Theorem 1, for the current value of λ. This solution is transformed to a feasible

[3] The implementation of the feasibility heuristics is available at: https://github.com/ekQ/flan.

solution for the network-alignment problem, as discussed in Sect. 3.3.3, and the two solutions are used for updating the bounds ℓ^* and u^*. Next, a new vector λ is computed by the subgradient update

$$\lambda_i^t = \lambda_i^{t-1} + \theta^t(1 - \sum_j x_{ij}),$$
$$\lambda_{ijk\ell}^t = \lambda_{ijk\ell}^{t-1} + \theta^t(w_{ijk\ell} - w_{k\ell ij}),$$

where x_{ij} and $w_{ijk\ell}$ are part of the relaxed solution. For θ^t, we adopt the same update rules used in Natalie 2.0 (El-Kebir et al. 2015).

The iterative process continues by solving $Z_{LD}(\lambda)$ for the new vector λ and repeating the aforementioned steps until a convergence criterion $|u^* - \ell^*| \leq \epsilon$ is satisfied or the maximum number of iterations (300 in our experiments) is satisfied.

3.3.5 Inferring the edges of the entity graph

The edges of the underlying entity graph are unknown. A reasonable initial guess can be obtained by setting $B = A$, but some edges may be missing from this initial guess as (i) A does not contain any edges between vertices from different graphs, and thus, there will not be any edges between entities that correspond to vertices in different input graphs, and (ii) the vertices of an input graph may correspond to only a subset of the entities.

To infer the missing edges when x and y are fixed, we need to minimize the objective function (1) with respect to the elements of B which correspond to the potentially missing edges.[4] We decompose the third term of the objective function and denote all the terms which are constant with respect to the optimized elements of B by C. This allows us to write the optimization problem as

$$\begin{aligned}
\min_B \quad & \sum_j f y_j + \sum_{i,j} d(i,j) x_{ij} - g \sum_{(j,\ell)\in E_I} \sum_{i,k} A_{ik} B_{j\ell} x_{ij} x_{k\ell} \\
& - g \sum_{(j,\ell)\notin E_I} \sum_{i,k} A_{ik} B_{j\ell} x_{ij} x_{k\ell} + \gamma \sum_{(j,\ell)\notin E_I} B_{j\ell}^2 \\
= \min_B \quad & C + \sum_{(j,\ell)\notin E_I} \left[\left(-g \sum_{i,k} A_{ik} B_{j\ell} x_{ij} x_{k\ell} \right) + \gamma B_{j\ell}^2 \right] \\
= \min_B \quad & C + \sum_{(j,\ell)\in E_m} B_{j\ell} \left(\gamma B_{j\ell} - g \sum_{i,k} A_{ik} x_{ij} x_{k\ell} \right).
\end{aligned}$$

[4] For simplicity, we write "$\min_B$ objective" although the objective is being minimized only w.r.t. elements B_{jl}, where $(j, \ell) \notin E_I$.

This is solved by setting $B_{j\ell} = 1$ whenever term $\gamma B_{j\ell} - g \sum_{i,k} A_{ik} x_{ij} x_{k\ell}$ is negative, which happens when

$$\sum_{i,k} A_{ik} x_{ij} x_{k\ell} > \frac{\gamma}{g}.$$

In all of our experiments, we set $\gamma = \frac{g}{2}$, which means that we add an edge between entities j and ℓ if there is at least one pair of neighboring vertices that is aligned to entities j and ℓ.

4 Related work

The Lagrangian relaxation framework has been successfully applied to many **NP**-hard optimization problems (Fisher 1981). Cornuejols et al. (1977) show that it is well suited for the uncapacitated facility location problem where the one-to-one constraint on sites and facilities is relaxed. Klau (2009) later shows that it is also applicable to the pairwise network alignment problem where a symmetry constraint is relaxed. Our multiple network alignment method combines these two ideas and relaxes both the one-to-one constraint (13) and the symmetry constraint (17) to make the relaxed problem feasible. Another related application of the Lagrangian relaxation approach is by Althaus and Canzar (2008) who adopt it for multiple sequence alignment.

A recent survey by Elmsallati et al. (2015) provides an overview of thirteen different network alignment methods. While the application focus of the survey is on protein–protein interaction networks, the techniques described are general and can be applied to different domains. Out of the thirteen methods, only three support multiple network alignment, namely, IsoRankN (Liao et al. 2009), SMETANA (Sahraeian and Yoon 2013), and NetCoffee (Hu et al. 2013). IsoRankN is a multiple network extension of the earlier IsoRank method (Singh et al. 2008) which is inspired by the PageRank algorithm, SMETANA is a greedy method based on a semi-Markov random walk model, and NetCoffee employs simulated annealing to optimize an objective function developed for multiple network alignment.

Clark and Kalita (2014) present an experimental survey on ten different pairwise alignment methods. In many of the experiments presented in the survey, Natalie (Klau 2009; El-Kebir et al. 2015) yields the highest accuracy and it is also reported to have a fast running time (El-Kebir et al. 2015). Therefore, one of our objectives is to extend Natalie to support multiple networks. This extension and the differences between Natalie and our method FLAN have been discussed in Sect. 3.2.

Apart from the biological problems, network alignment has been previously applied at least to ontology alignment by Bayati et al. (2013). The authors present a novel approach for solving the pairwise graph alignment problem based on the use of belief propagation (BP): given two networks the BP algorithm begins by defining a probability distribution on all matchings between the networks and then using a message passing algorithm to approximately infer a matching which gives the maximum a posterior (MAP) assignment. However, again the BP algorithm is restricted to pairwise

alignment and it is not clear how a generalization to the multiple alignment case can be derived.

Multiple network alignment can also be seen as a *collective entity resolution* problem (Bhattacharya and Getoor 2007; Singla and Domingos 2006). In the future work, it would be interesting to study the applicability of multiple network alignment methods to typical collective entity resolution problems, such as author disambiguation.

The problem of merging family trees has been recently studied by Kouki et al. (2016), who employ a greedy method, and Malmi et al. (2016), who study an active learning setting. Furthermore, entity resolution for genealogical data has been previously studied, for example, by Efremova et al. (2015) and Christen et al. (2015). There are also methods developed for a related problem of tree alignment, for example, in the context of web data extraction (Zhai and Liu 2005). However, these methods are not applicable to family trees since the latter contain cycles.

5 Experimental evaluation

In this section, we present experiments on synthetic and real-world datasets. Our real-world data are social networks and genealogical trees. In each of these scenarios, the input graphs are only partially overlapping and the number of graphs is more than two.

A common challenge when applying network alignment methods in practice is the tuning of the method parameters. In the methods studied in this paper, the parameters are: f, which controls the dissimilarity of vertices we are willing to align instead of keeping them separate, and g, which controls the balance between the importance of aligning neighbors to neighbors vs. aligning vertices to similar vertices. If ground truth data is available, the parameter values can be tuned via cross-validation but otherwise, it is common to resort to using some default parameter values.

Motivated by the challenge of parameter selection, the focus of the following experiments is on studying the sensitivity of the methods to the selection of f which is crucial when aligning partially overlapping networks. We study both the performance of different methods for a range of f values and the performance of FLAN when prior knowledge on the number of entities is available and thus the selection of f is not required.

Method naming conventions The following methods are compared in the experiments: NATALIE, PROGNATALIE, and PROGNATALIE++ (Sect. 3.2) are our multiple-network extensions of the pairwise method originally presented by Klau (2009). FLAN is our facility-location-based method for multiple network alignment, whereas FLAN0 is a baseline method which only solves the Lagrangian relaxation once and does not update B, the adjacency matrix for entities. CFLAN_N_e refers to the version of FLAN with N_e as the fixed number of entities. Both FLAN and CFLAN_N_e run the alternating optimization procedure for at most five iterations since the solution typically does not improve anymore after that. ISORANKN (Liao et al. 2009) is a popular multiple network alignment method. Instead of f, it has parameter *alpha* which controls the relative weight of network and sequence data, taking values between 0 and 1. The other parameters of ISORANKN are set to $K = 30$, $thresh = 10^{-4}$, and $maxveclen = 10^6$, based on the recommendations in the README file of the pro-

gram. Finally, method UNARY, which is only used in Sect. 5.3, refers to PROGNATALIE where discount g for the quadratic term is set to zero so the method aligns the vertices only based on their attribute similarity.

5.1 Aligning synthetic networks

We start by describing the data generation process and then present the results.

5.1.1 Data

First, we generate an underlying entity graph using the preferential attachment model (Barabási and Albert 1999) with 100 vertices and 2 as the number of edges new vertices are attached with. Then we sample a label for each vertex from a set of 33 unique labels so that there will be 3 vertices with a duplicate label on average. The labels are treated as attributes and the distance between two vertices is set to 0 if their labels are the same and 1 otherwise. Only the vertices with the same label are considered as candidate entities.

Second, we generate 10 manifestations of the entity graph which serve as the input graphs. The manifestations are generated by picking a random seed vertex and then doing random walk until 30 distinct vertices have been discovered.

Third, we corrupt the edges of the input graphs to make the alignment problem more challenging. We first discard 20% of the graph edges, chosen at random, and then we add 10% more edges, again selected at random. This process is done independently for each input graph.

The optimization problem corresponding to the alignment of these graphs contains 1500 variables x_{ij} and 1300 variables $w_{ijk\ell}$ on average, depending on the initialization.

5.1.2 Results

We set $g = 0.5$ and vary f. Parameter g controls the balance between vertex attribute distances and the pairwise discounts, but since in this case the attribute distances are 0 for each candidate entity, only the proportion f/g matters. The proportion can be varied merely by varying f.

The results are shown in Fig. 2. Precision and recall are computed based on the set of vertex pairs which are predicted to correspond to the same entity ($\mathcal{P}$) and the set of vertex pairs that truly correspond to the same entity ($\mathcal{T}$) as follows

$$\text{precision} = \frac{|\mathcal{P} \cap \mathcal{T}|}{|\mathcal{P}|}, \quad \text{recall} = \frac{|\mathcal{P} \cap \mathcal{T}|}{|\mathcal{T}|}.$$

In terms of precision, PROGNATALIE++ and FLAN yield the best overall performance. Both of these methods clearly outperform their counterparts FLAN0 and PROGNATALIE that do not infer the underlying network. In terms of recall, NATALIE and PROGNATALIE++ obtain the best performance, particularly with higher f valuer. All of these methods clearly outperform ISORANKN when f is sufficiently large.

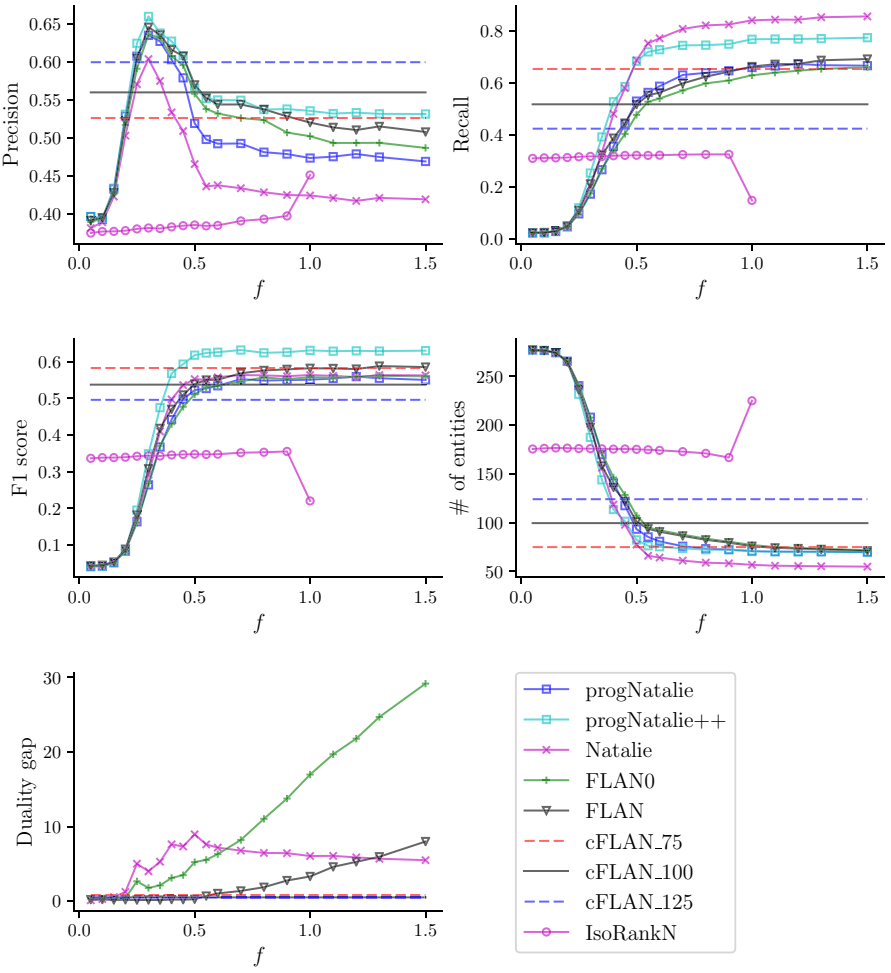

Fig. 2 Aligning partial random graphs, varying the cost f of opening an entity

We also study the scenario that we have some prior knowledge about the number of entities, 100, allowing us to employ CFLAN. The results show that by setting the constraint on the number of entities above or below the true number, we can either improve precision or improve recall, respectively. This observation can be leveraged in the case that one of the measures is more important than the other. Although, it is possible to obtain a higher precision or recall compared to CFLAN by carefully selecting f, in overall, CFLAN works rather robustly, providing good alignments without having to fine-tune f.

Finally, in the bottom-middle plot of Fig. 2, we show the duality gaps for all the methods except for the progressive ones since they solve several alignment problems and hence do not provide a single duality gap value. The gaps are relatively small compared to the total number of variables, 2800, but we can notice that especially for

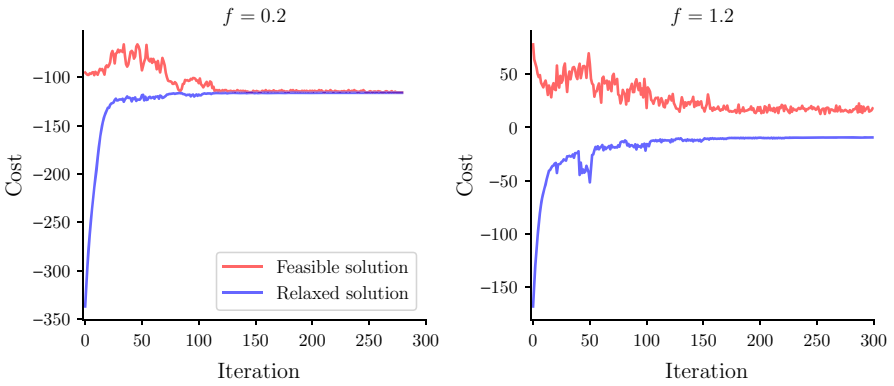

Fig. 3 Evolution of the cost of the feasible solution, which yields an upper bound, and the relaxed solution, which yields a lower bound. For $f = 0.2$ the algorithm finds the global minimum and converges in 280 steps

FLAN0, the duality gap increases with f. Figure 3 shows two examples on how the duality gap evolves when using FLAN0. For FLAN, the gaps are smaller, which shows that by updating the adjacency matrix B and solving the optimization problem again the problem becomes easier and the Lagrangian relaxation tighter.

5.2 Aligning social networks

The social network alignment problem refers to the problem of finding matching users across different social networks, such as Facebook and Twitter. This problem points out privacy concerns as identifying a person's user handles across multiple service can lead to highly increased user profiling accuracy. However, it also comes with useful applications—for instance, a social networking service can provide helpful friend suggestions for a new user if it can identify the user's profile in another service.

5.2.1 Data

We study the multiplex dataset from Aarhus University (Magnani et al. 2013). The dataset contains five networks between the employees of the Department of Computer Science. Table 1 shows statistics about these networks.

All attributes of the vertices have been anonymized but the true alignment of users is known through user IDs. We make two modifications to the dataset. First, we select at most 40 users from each network in order to make the alignment problem more challenging by having only partially overlapping networks. Second, we sample a name label for each user ID so that each label is shared by three user IDs on average. Candidate entities are formed by taking the vertices with the same label.

An example of a social network alignment problem is presented in Fig. 4 where each name label is assigned a distinct color.

Table 1 The CS-Aarhus multiplex dataset before and after removing some people to have only partially overlapping networks

Method	# of vertices	# of edges	Used # of vertices	Used # of edges
Lunch	60	193	40	85
Facebook	32	124	32	124
Co-author	25	21	25	21
Leisure	47	88	40	64
Work	60	194	40	85
Total	**61**	**620**	**60**	**379**

The last row shows the number of distinct vertices after combining the datasets

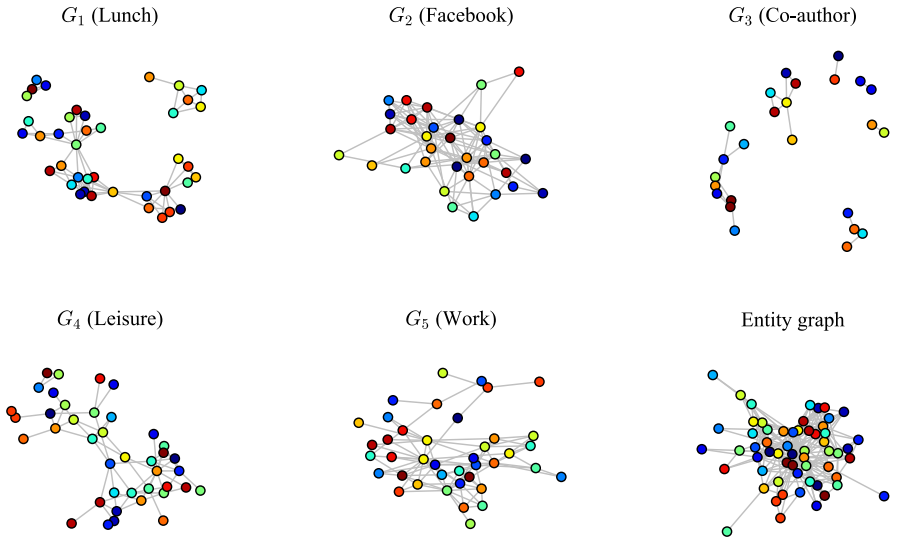

Fig. 4 An instance of the social network alignment problem. Vertices can only be aligned to other vertices with the same color. Entity graph depicts the set of underlying entities and edges between them

5.2.2 Results

Figure 5 shows the results for the social network experiment. The relative performance of the methods is similar to the previous experiment. However, this time the precision difference between PROGNATALIE++ and FLAN is higher. Precision of ISORANKN is not shown in the figure to make the differences between the other methods more visible but it is always between 0.34 and 0.40.

5.3 Aligning family trees

Services like *Ancestry.com* and *MyHeritage* attract millions of paying subscribers who upload their ancestry information to the service, trying to find new relatives to add to

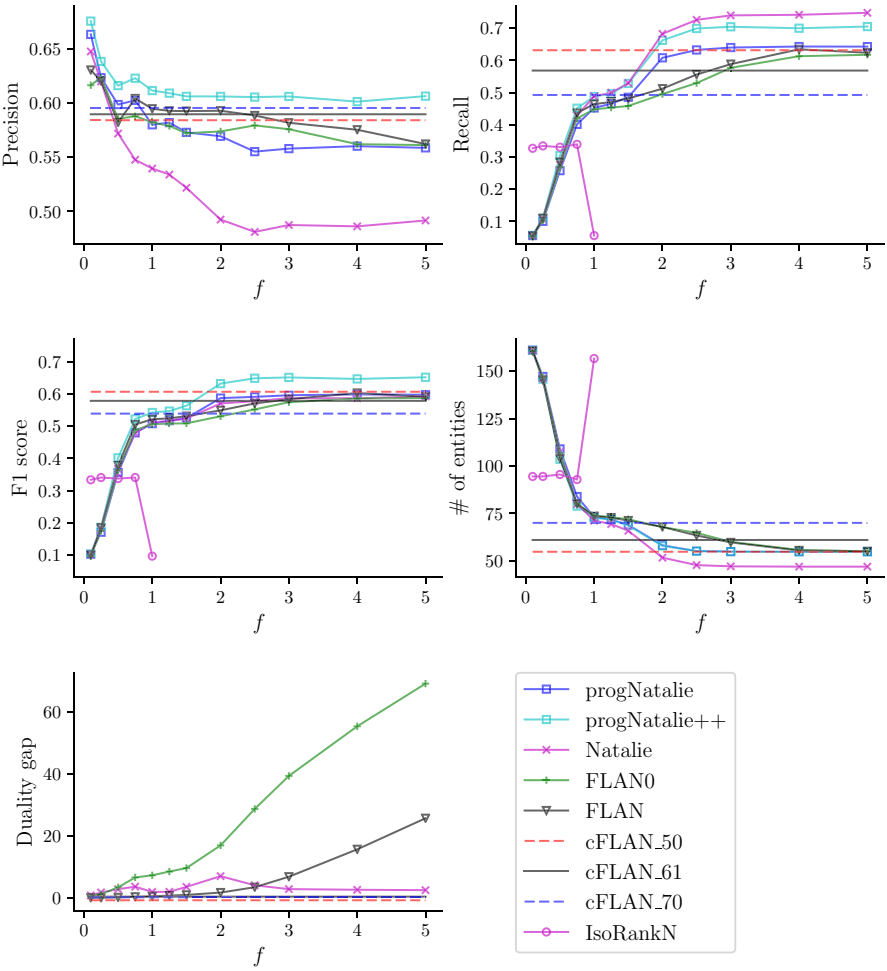

Fig. 5 Aligning the layers of a multiplex graph with randomized usernames. On average, there are three persons with a duplicate name

their *family tree*. These family trees (or directed acyclic graphs to be more precise) are prone to error since they are constructed based on noisy historical records, which are often challenging to interpret. Furthermore, they overlap only partially since each genealogist typically starts to expand their own tree so that it is connected to others only after going back a sufficient number of generations. Therefore, network alignment methods seem to be ideal tools for aggregating family trees from different users. However, apart from two recent works (Kouki et al. 2016; Malmi et al. 2016), we are not aware of previous published work on applying network alignment methods on genealogical data, which is the aim of this experiment.

5.3.1 Data

We have obtained a family tree containing 64,208 people constructed by an individual genealogical researcher in Finland. To be able to have a ground truth to compare against, we take this tree as the underlying entity graph and sample subgraphs of it to be aligned.

First, we sample 10 subgraphs as follows. We start by picking a seed person and then doing random walk until 100 distinct people have been discovered.

Second, from the different attributes associated with each person, we consider only the first name, last name, and birth year in this experiment. Birth year is corrupted by rounding it to the nearest ten. For the first and the last name, we use lists of alternative spellings of names obtained from the Genealogical Society of Finland. The name of each person found on these lists is randomly replaced by one of its alternative spellings. For instance, the alternative spellings for name *Jean* are *Jan*, *Jans*, *Janne*, and *Jannes*.

When selecting candidate entities for each individual, we find people from other trees born in the same decade and select up to 5 people with the most similar names. Name similarity is computed as the average of the Jaro–Winkler string similarity (Winkler 1990) of the first names and last names. The Jaro–Winkler similarity is a popular choice for de-duplicating name records.

5.3.2 Results

Now that the vertex similarities are not constant for each candidate match, varying g could potentially change the results. However, for simplicity, we again set $g = 0.5$ and focus on studying parameter f. The alignment results are shown in Fig. 6.

This time also FLAN0 and PROGNATALIE yield a good performance since the edges of the input graphs have not been corrupted so updating B does not provide any significant improvements. Comparing CFLAN with other methods, we observe that unless the user is able to set f between 0.5 and 1.5, CFLAN obtains the highest F1 score. The parameter N_e has been set by counting the true number of entities using the ground truth data. In practice, when the ground truth is not available, we could be able to estimate N_e, for example, based on census records by counting how many people used to live in the area that the family trees cover.

In this experiment, we also included method UNARY which aligns vertices only based on attribute similarities and cost f for leaving a vertex unaligned. The performance of this method is in overall the lowest after ISORANKN which suggests that it is, indeed, important to consider also the structure of the networks when aligning family trees.

5.4 Scalability

Finally, we study the running times of the different alignment methods. We use family trees as the dataset, varying the number of graphs and the number of people per graph. The results averaged over 10–30 random initializations of the graphs are shown in Fig. 7.

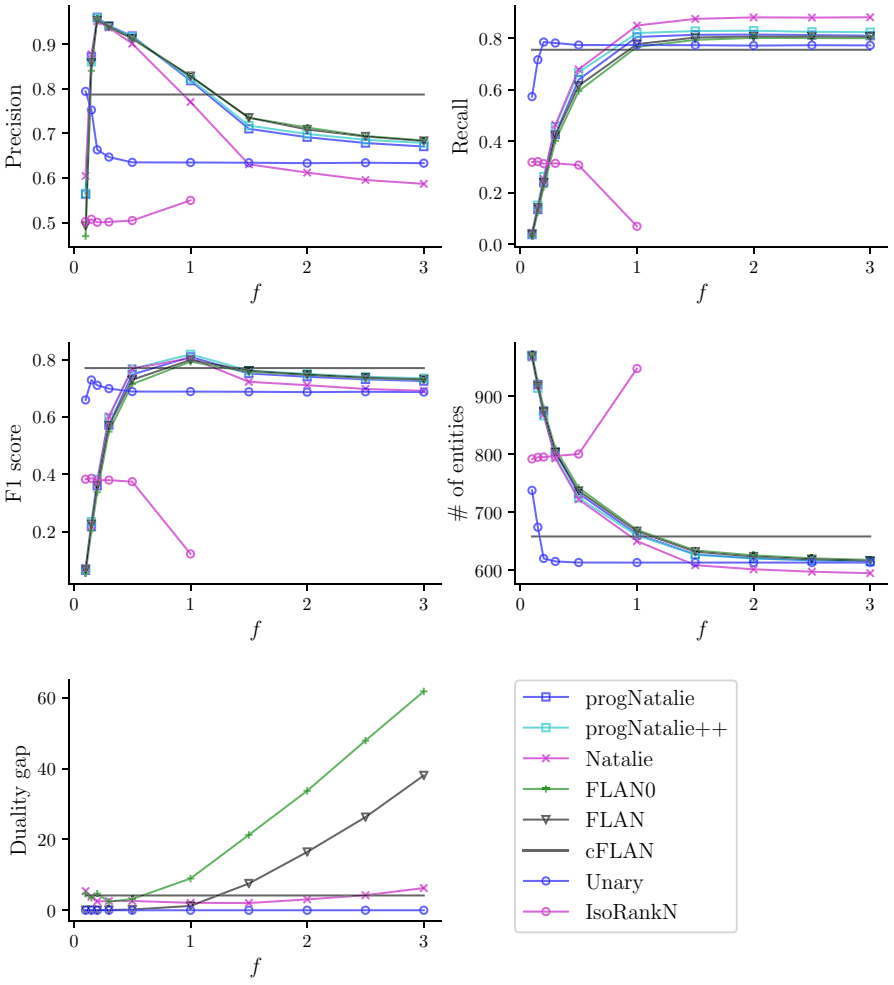

Fig. 6 Aligning family trees, varying the cost f of opening an entity

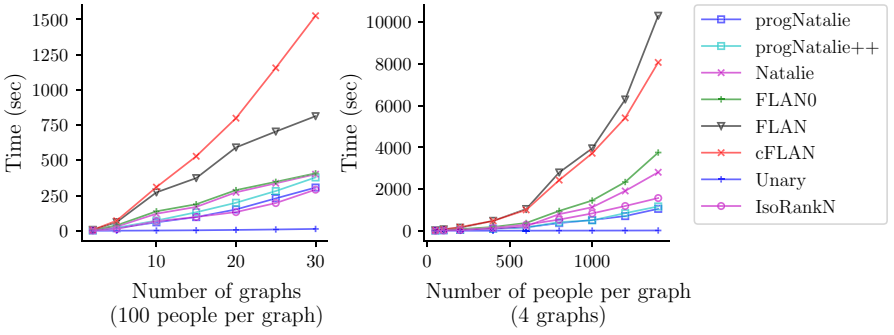

Fig. 7 Running times of the studied algorithms on family tree data

First, we notice that FLAN and CFLAN have the highest running times which is natural as they both solve the alignment problem up to five times, updating B between the runs. Their running times are not, however, five times higher compared to FLAN0, which can be explained by (i) the alternating optimization sometimes converging (i.e. returning the exact same solution compared to the previous iteration) in less than five iterations, and (ii) the problem getting easier with the updated and presumably more accurate B matrices, thus leading to a faster convergence of the subgradient optimization.

Second, when aligning four graphs with 1400 vertices each, the running time is almost three hours with FLAN, suggesting that the current Python implementation of FLAN would not be scalable enough for input graphs with 10^4 vertices or more. However, with a more optimized implementation one could expect significant speedups. For instance, a Matlab implementation of the pairwise NATALIE,[5] which heavily uses matrix operations and is written partly in C++, solves three pairwise family tree alignment problems in 5.1 s, whereas our implementation of PROGNATALIE, which also solves three pairwise problem when the number of input graphs is four, runs in 1050 s for family trees with 1400 vertices. Thus our implementation is over 200 times slower.

Third, we notice that the running times grow superlinearly when increasing the number of people per graph. Although the number of vertices grows linearly, the complexity of the algorithm is superlinear since it requires solving a bipartite matching problem which takes $\mathcal{O}\left(n^3\right)$ using the Hungarian algorithm. Nevertheless, the running times do not appear to grow exponentially despite the problem being **NP**-hard.

6 Conclusions

We formalized the multiple network alignment problem using a novel extension of the facility location problem. The problem is **NP**-hard, but we were able to obtain good approximate solutions using a Lagrangian relaxation approach, called FLAN, which also provides bounds on the quality of a solution.

A practical advantage of FLAN is that it has an option to specify the number of entities. This allows FLAN to work robustly without any other ground truth data which would typically be needed to fine-tune the parameters of a network alignment method via cross-validation before applying the method. This can be a significant advantage when solving a new problem instance where ground truth data is unavailable. Furthermore, in addition to aligning the input networks, FLAN infers the underlying entity network.

We also presented and evaluated three multiple-network extensions to Natalie, which is a state-of-the-art pairwise network alignment method. A progressive extension with edge updates (PROGNATALIE++) was shown to provide a good experimental performance.

As a practical guideline for solving multiple network alignment problems, we recommend first trying out PROGNATALIE++, which is computationally less expensive

[5] The implementation is available at https://www.cs.purdue.edu/homes/dgleich/codes/netalign/ and has been used in Bayati et al. (2013) and Malmi et al. (2016).

than FLAN since it does not consider a single large problem but decomposes it into multiple, smaller pairwise alignment problems. However, if prior information on the number of entities, i.e., the number of distinct elements in all the input graphs together, is available, then FLAN is recommended. For instance, in the case of family trees, such prior information could be possible to extract from separate census records. Finally, if the main goal is to achieve a high recall, it is recommended to use NATALIE with a high value of f parameter, which yields lower F1 scores compared to PROGNATALIE++ and FLAN but often achieves the highest recall.

Acknowledgements The authors are grateful to Pekka Valta and the Genealogical Society of Finland for providing the family tree dataset, to Jukka Suomela for useful discussions on FLAN, to Gunnar W. Klau for his advice on extending NATALIE to multiple networks, and to the anonymous reviewers for their constructive comments. This work was supported by Academy of Finland Project "Nestor" (286211).

References

Althaus E, Canzar S (2008) A Lagrangian relaxation approach for the multiple sequence alignment problem. J Comb Optim 16(2):127–154

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Bayati M, Gleich DF, Saberi A, Wang Y (2013) Message-passing algorithms for sparse network alignment. ACM Trans Knowl Discov Data 7(1):3

Bezdek JC, Hathaway RJ (2003) Convergence of alternating optimization. Neural Parallel Sci Comput 11(4):351–368

Bhattacharya I, Getoor L (2007) Collective entity resolution in relational data. ACM Trans Knowl Discov Data 1(1):5

Christen P (2012) Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer, Berlin

Christen P, Vatsalan D, Fu Z (2015) Advanced record linkage methods and privacy aspects for population reconstruction—a survey and case studies. In: Population reconstruction. Springer, pp 87–110

Clark C, Kalita J (2014) A comparison of algorithms for the pairwise alignment of biological networks. Bioinformatics 30(16):2351–2359

Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. IJPRAI 18(3):265–298

Cornuejols G, Fisher ML, Nemhauser GL (1977) Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manag Sci 23(8):789–810

Efremova J, Ranjbar-Sahraei B, Rahmani H, Oliehoek FA, Calders T, Tuyls K, Weiss G (2015) Multi-source entity resolution for genealogical data. In: Population reconstruction. Springer, pp 129–154

El-Kebir M, Heringa J, Klau GW (2015) Natalie 2.0: sparse global network alignment as a special case of quadratic assignment. Algorithms 8(4):1035–1051

Elmsallati A, Clark C, Kalita J (2015) Global alignment of protein–protein interaction networks: a survey. IEEE/ACM Trans Comput Biol Bioinform PP(99):1-1. doi:10.1109/TCBB.2015.2474391

Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Manag Sci 27:1–18

Goga O, Loiseau P, Sommer R, Teixeira R, Gummadi KP (2015) On the reliability of profile matching across large online social networks. In: Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1799–1808

Hochbaum DS (1982) Heuristics for the fixed cost median problem. Math Program 22(1):148–162

Hu J, Kehr B, Reinert K (2013) NetCoffee: a fast and accurate global alignment approach to identify functionally conserved proteins in multiple networks. Bioinformatics 30(4):540–548

Klau GW (2009) A new graph-based method for pairwise global network alignment. BMC Bioinform 10(Suppl 1):S59

Kouki P, Marcum C, Koehly L, Getoor L (2016) Entity resolution in familial networks. In: Proceedings of the 12th workshop on mining and learning with graphs

Liao CS, Lu K, Baym M, Singh R, Berger B (2009) IsoRankN: spectral methods for global alignment of multiple protein networks. Bioinformatics 25(12):i253–i258. doi:10.1093/bioinformatics/btp203

Magnani M, Micenkova B, Rossi L (2013) Combinatorial analysis of multiple networks. arXiv:1303.4986

Malmi E, Terzi E, Gionis A (2016) Active network alignment: a matching-based approach. arXiv:1610.05516

Sahraeian SME, Yoon BJ (2013) SMETANA: accurate and scalable algorithm for probabilistic alignment of large-scale biological networks. PLOS ONE 8(7):e67,995

Shor NZ (2012) Minimization methods for non-differentiable functions, vol 3. Springer, New York

Singh R, Xu J, Berger B (2008) Global alignment of multiple protein interaction networks with application to functional orthology detection. Proc Natl Acad Sci 105(35):12763–12768

Singla P, Domingos P (2006) Entity resolution with markov logic. In: Proceedings of the sixth international conference on data mining, ICDM'06. IEEE, pp 572–582

Vazirani VV (2001) Approximation algorithms. Springer, New York

Winkler WE (1990) String comparator metrics and enhanced decision rules in the fellegi–sunter model of record linkage. In: Proceedings of the section on survey research methods. American Statistical Association, pp 354–359

Zhai Y, Liu B (2005) Web data extraction based on partial tree alignment. In: Proceedings of the 14th international conference on world wide web. ACM, pp 76–85

Zhang J, Yu PS (2015) Multiple anonymized social networks alignment. In: Proceedings of the IEEE international conference on data mining, ICDM'15. IEEE