Unsupervised Network Alignment

5

5.1 Overview

Identifying the common users shared by different online social sites is a very hard task even for humans. Manually labeling of the anchor links can be extremely challenging, expensive (in human efforts, time, and money costs), and tedious, and the scale of the real-world online social networks involving millions even billions of users also renders the training data labeling much more difficult. In this chapter, we will introduce several approaches to resolve the *network alignment* problem based on the unsupervised learning setting instead, where no labeled training data will be needed in model building.

Given two heterogeneous online social networks, which can be represented as $G^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)})$ and $G^{(2)} = (\mathcal{V}^{(2)}, \mathcal{E}^{(2)})$, respectively, the *unsupervised network alignment* problem aims at inferring the set of potential anchor links connecting the shared users between networks $G^{(1)}$ and $G^{(2)}$. Let $\mathcal{U}^{(1)} \subset \mathcal{V}^{(1)}$ and $\mathcal{U}^{(2)} \subset \mathcal{V}^{(2)}$ be the user sets in these two networks, respectively, we can represent the set of potential anchor links between networks $G^{(1)}$ and $G^{(2)}$ as $\mathcal{A} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$. In the *unsupervised network alignment* problem, among all the potential anchor links in set $\mathcal{A} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$, we want to infer which one in set $\mathcal{A}$ should exist in the real world.

In this chapter, we will study the network alignment problem based on the unsupervised learning setting with no training data at all. We will first introduce several unsupervised heuristics for measuring the similarities of user accounts across networks, which can serve as the basic predictors of anchor links across networks. After that we will introduce the unsupervised network alignment problem of homogeneous networks especially, and talk about the existing approaches proposed to address the problem. To handle the heterogeneous social networks involving both complex structures and different types of attribute information, a state-of-the-art unsupervised heterogeneous network alignment algorithm will be introduced afterwards. In the real-world social networks, besides the users, many other types of information entities can also be shared across different networks, like products, videos, and POIs (points of interest). Identifying multiple types of common information entities shared across social networks simultaneously is called the *network co-alignment* problem. A novel unsupervised network co-alignment model will be introduced to solve the problem. Besides the pairwise networks, aligning multiple (more than 2) networks simultaneously is called the *multiple network alignment* problem. In the *multiple network alignment* problem, preserving the consistency of the alignment results among all these networks is necessary, which will introduce more constraints on

J. Zhang, P. S. Yu, *Broad Learning Through Fusions*,
https://doi.org/10.1007/978-3-030-12528-8_5

the unsupervised alignment model at the same time. Finally, we will introduce a novel unsupervised *multiple network alignment* algorithm at the end of this chapter.

5.2 Heuristics Based Unsupervised Network Alignment

To infer the correspondence relationships of the shared users between different social networks, many different types of unsupervised heuristics can be applied. Besides the similarity measures (or features) calculated based on the social network structures, location check-ins, textual contents, and temporal activities of users introduced in Sect. 4.3.1, in this section we will introduce two other categories of unsupervised network alignment heuristics based on the user name and user profile information, respectively.

5.2.1 User Names Based Network Alignment Heuristics

Formally, given a user pair $u_i^{(1)}$ and $u_j^{(2)}$ from networks $G^{(1)}$ and $G^{(2)}$, respectively, we can represent their names in these two networks as $\text{name}(u_i^{(1)})$ and $\text{name}(u_j^{(2)})$. Generally, user names can be denoted as strings involving one or several words. To help their friends identify them, users tend to use their real names in many online social networks, like Facebook, Twitter, and LinkedIn. Several heuristics can be proposed to infer the user anchor links across networks with user names. Exact matching of user names is the most intuitive idea, which is based on the assumption that users tend to use exactly the same names in different sites. For instance, if the names of $u_i^{(1)}$ and $u_j^{(2)}$ are exactly the same (i.e., $\text{name}(u_i^{(1)}) = \text{name}(u_j^{(2)})$), then $u_i^{(1)}$ and $u_j^{(2)}$ are highly likely to be the same person.

However, in real-world social networks, such an assumption can be violated with certain degrees and this method may not perform well due to several reasons:

- *Abbreviated Name*: For some reasons, users tend to abbreviate their names in some sites. For instance, name "Anne-Marie Slaughter" can be written as "A.-M. Slaughter" by using the initials of the first name instead.
- *Order-Reversed Name*: Depending on the sites, user names can display in different ways. For instance, names "Anne-Marie Slaughter" and "Slaughter, Anne-Marie" may refer to the same user, but the string representations have flipped the first and last name, which may create many problems for exact name matching.
- *Alias Names*: Some people can have their alias names, like the nick name. For example, "Michael" and "Mike" may refer to the same person, but they are written in totally different ways.
- *Duplicated Names*: For many common names, lots of users may have exactly the same name, e.g., "James," "Michael," "Robert," "Mary," "Maria," and "David." Simply name matching may not be sufficient to identify the exact shared users accounts.

To overcome these aforementioned problems (except *duplicated names*), in this part, we will introduce the unsupervised network alignment heuristics based on the user names to compute the user similarities. For the *duplicated name* case, besides the user name information, we will introduce several other network alignment heuristics by using the user profile information in the following subsection.

5.2.1.1 Name Similarity Metrics

Instead of using the exact name matching, *name similarity metrics* are usually applied to calculate how likely the users are the same person based on their names. Currently, the *name similarity metrics* can be categorized into three groups: *character based similarity metrics*, *token based similarity metrics*, and *phonetic similarity metrics*.

Character Based Similarity Metrics *Character based similarity metrics* are usually used to handle typographical differences of user name strings, and the well-known *character based similarity metrics* include

- *Edit Distance*: Given two strings, the *edit distance* [31, 34, 41] denotes the minimal number of *insertion*, *deletion*, and *substitution* edit operations of characters needed to transform one string into the other one. Given two user names, if the *edit distance* between them is short, they will be similar to each other.
- *Damerau-Levenshtein Distance*: *Damerau-Levenshtein distance* [28] is a variation of *edit distance*. Besides the basic *insertion*, *deletion*, and *substitution* edit operations, it also takes the character *transposition* as another elementary edit operation.
- *Jaro Distance*: *Jaro distance* [22] is also a type of edit distance computed based on the number of shared characters between strings and the number of transpositions operations. Formally, given two strings s_1 and s_2, the *Jaro Distance* between them is defined as

$$d_j(s_1, s_2) = \begin{cases} 0, & \text{if } m = 0, \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right), & \text{otherwise,} \end{cases} \tag{5.1}$$

  where m denotes the number of matching characters between s_1 and s_2, t denotes half the number of transpositions to change one shared character sequence into the other, and $|s_1|$, $|s_2|$ denote the length of s_1 and s_2, respectively. Two characters from s_1 and s_2, respectively, are considered matching only if they are the same and their indexes are not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$ way. We will introduce more information about the *Jaro distance* in Example 5.2.
- *Jaro-Winkler Distance*: *Jaro-Winkler distance* [42] is a variant of *Jaro distance*, and it uses a prefix scale p to give more favorable ratings to strings that match from the beginning for a set prefix length. Given two strings s_1 and s_2, the *Jaro-Winkler distance* between them is defined as

$$d_w(s_1, s_2) = d_j(s_1, s_2) + l \cdot p \cdot (1 - d_j(s_1, s_2)), \tag{5.2}$$

  where l is the length of common prefix at the start of the string up to a maximum of four characters, and p is a constant scaling factor for how much the score is adjusted upward for having common prefixes (whose standard value is $p = 0.1$).
- *Longest Common Sub-string (LCS)*: *LCS* [21] repeatedly identifies and removes the longest common sub-string from two names, whose total length can denote the similarity between the user names. For instance, the total length of shared sub-string (involving more than 1 character) by input names "Michael Jordan" and "Yordan Michelin" is 11 (the shared sub-strings include "mich," "el," and "ordan").

Example 5.1 For instance, given two user name strings "Michael Jordan" and "Mike Jordan," we can convert "Mike Jordan" into "Michael Jordan" with 4 edits: 1 *substitution* and 3 *insertion*:

- *substitute "k" with "c"*: with this operation, we can convert "Mike Jordan" to "Mice Jordan";
- *insert "h" after "c"*: with this operation, we will be able to convert "Mice Jordan" to "Miche Jordan";
- *insert "a" after "h"*: with this operation, we can convert "Miche Jordan" to "Michae Jordan";
- *insert "l" after "e"*: with this operation, we will be able to successfully convert "Michae Jordan" into "Michael Jordan."

Example 5.2 Given two strings "TRACE" and "CRATE," we can identify the matching characters between them are "R," "A," and "E" and $m = 3$. Here, characters "T" and "C" are shared by these two strings, but they are not considered to be matching characters since their index distance is 3, which is father than $\left\lfloor \frac{\max(5,5)}{2} \right\rfloor - 1 = 1$ away. Considering that the matching characters "R-A-E" are already in the same order in both of these two strings, so no transpositions are needed, and we have $t = 0$.

Therefore, the *Jaro distance* between "TRACE" and "CRATE" is

$$\begin{aligned} &\frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \\ &= \frac{1}{3}\left(\frac{3}{5} + \frac{3}{5} + \frac{3-0}{3}\right) \\ &= \frac{11}{15} \end{aligned} \tag{5.3}$$

Token Based Similarity Metrics The *character based similarity metrics* may fail to handle the names with first and last name flipped. To handle such a case, a set of *token based similarity metrics* are proposed, which divide name strings into a set of tokens (i.e., words) instead.

- *Common Token*: *Common token* metric counts the number of common words shared by different user names. Given two user names s_1 and s_2, we can transform them into two sets of tokens in advance, which can be denoted as $\text{set}(s_1)$ and $\text{set}(s_2)$, respectively. The number of *common tokens* [44] shared by them can be denoted as

$$\text{CT}(\text{set}(s_1), \text{set}(s_2)) = |\text{set}(s_1) \cap \text{set}(s_2)|. \tag{5.4}$$

- *Token based Jaccard's coefficient*: In some cases, people can have very long names, like "Jose Arcadio Buendia" and "Jose Arcadio," which share two common tokens but actually refer to different people in "*One Hundred Years of Solitude*." *Token based Jaccard's coefficient* proposes to penalize the long names, and the score calculated for names $\text{set}(s_1)$ and $\text{set}(s_2)$ can be denoted as

$$\text{TJC}(\text{set}(s_1), \text{set}(s_2)) = \frac{|\text{set}(s_1) \cap \text{set}(s_2)|}{|\text{set}(s_1) \cup \text{set}(s_2)|}. \tag{5.5}$$

- *Token based Cosine Similarity*: *Cosine Similarity* [5] expresses strings as term vectors, where each word denotes a dimension in the vector representation. Formally, given two user names s_1 and s_2, we can represent their token vectors as $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively, and the *token based cosine similarity*

score of the user names can be represented as

$$\text{TSC}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\mathbf{s}_1^\top \cdot \mathbf{s}_2}{\|\mathbf{s}_1\| \cdot \|\mathbf{s}_2\|}. \tag{5.6}$$

- *TFIDF-TCS*: *TFIDF* denotes "*Term Frequency/Inverted Document Frequency*" [35], which counts frequency of a word but also penalizes it with its occurrence in other documents. *TFIDF* can be applied to weight the name's feature vectors before using them to compute the *Token based Cosine Similarity* measure. Formally, the *Token based Cosine Similarity* measure subject to the *TFIDF* weighted user name vectors can be denoted as *TFIDF-TCS*.

As to the partition of names into tokens, different techniques can be applied, including specific delimiter based partition, n-gram [8] based partition, etc. We will not introduce these name partition methods here since it will be out of the scope of this book.

Example 5.3 For instance, given two user names s_1 ="Michael Jordan" and s_2="Jordan, Michael," based on the character based similarity metrics, e.g., *edit distance*, we can compute their edit distance to be 12. Meanwhile, by dividing the user names into tokens by separator " " (i.e., the space between tokens) or ", ", we can transform the name strings into two unit token sets $set(s_1) = \{Michael, Jordan\}$ and $set(s_1) = \{Jordan, Michael\}$.

According to the *common token*, *token based Jaccard's coefficient* or *token based cosine similarity* metrics, we can compute their similarity scores all to be 1.0. Viewed in such a perspective, these *token based similarity metrics* can effectively amend the disadvantages of the *character based similarity metrics*.

Phonetic Similarity Metrics These metrics introduced before are mostly based on the string representation of the names. In many cases, people like to use abbreviated names based on the pronunciation instead of the textual representations. For instance, people like to use "Mike" as a nick name of "Michael," which is pronounced in a very similar way but is very different in their writings. Some *phonetic similarity metrics* [37] have been proposed to measure the phonetic similarity between strings, which can be applied to compute the user name similarities as well.

- *Soundex Matching*: *Soundex* [39] is a phonetic algorithm for indexing names by sound. The Soundex code [39] for a name consists of a letter followed by three numerical digits: the letter is the first letter of the name, and the digits encode the remaining consonants. Given two name strings s_1 and s_2, if their *Soundex codes* are the same, their *Soundex matching* will be 1; otherwise, it will be 0.
- *Soundex Similarity*: Instead of exact matching the *Soundex codes*, *Soundex similarity* proposes to count the shared characters in the *Soundex codes* of input strings, which is normalized by the returned *Soundex code* length.

5.2.2 Profile Based Network Alignment Heuristics

In the real-world online social networks, many users will share the same name (especially the regular names like "Mike," "David," etc.) and the above name based similarity metrics will fail to differentiate these users in the alignment process. To help improve the alignment results, a set of *profile based network alignment heuristics* will be introduced in this part based on the user profile information,

including *hometown*, *employment history*, *educational records*, *gender*, *birthday*, and other personal information. Generally, these information are represented as strings as well, and it seems the string similarity measures introduced before can also be applied here. Meanwhile, slightly different from names, the profile information usually consists of several components and is much longer than names, which renders the similarity metrics to be used here different from those introduced before.

- *Location Distance*: Given the hometown of two users from different online social networks, the hometown locations are usually represented as strings and we can apply the character/token based string similarity metrics introduced before to measure how close the users are in terms of geographical locations. Meanwhile, in many cases, there exist various different string representations for the same location, e.g., "PA," "Penn," and "Pennsylvania," all denote the Pennsylvania state in the USA, which will make the introduced metrics fail to work. A better way to handle the hometown information is using the online Map APIs to transform the hometown strings to the (latitude, longitude) coordinate pairs, and calculate the geographical distance between the coordinate pairs as the metric instead. Several different Map APIs are available online, like Google Maps,[1] Bing Maps,[2] Apple Maps.[3]
- *Birthday*: Online social networks may ask for users' birthday, and will offer some special services for users on their birthday by either sending notifications to friends online or generating some celebration posts on their homepage timeline. User's birthday is usually represented as a string containing the year, month, and day. Besides the string similarity metrics, these birthday strings can be transformed into a date object, based on which we can calculate the birthday closeness by computing how many days the users' birthdays are apart from each other. The open source toolkits that can handle the date related strings include Python datetime[4] and Java Date (java.util.Date[5]).
- *Vector Space Model*: For the remaining information represented in strings, they can be handled with either exact matching or *character based similarity metrics* for the *gender* information, *token based similarity metrics* and *phonetic similarity metrics* for the employment and educational records. To deal with such long string information together, we can treat user profile as a document and the *vector space model* [36] can be applied here. Vector space model or term vector model is an algebraic model for representing textual documents as vectors of identifiers. Given two users' profile information, one of user's profile can be treated as the query "profile document" and used to retrieve the similar "profile document" from another network. The similarity metric frequently used in the *vector space model* is *cosine similarity*, and TFIDF can also be applied to weight the feature vector representations prior to the similarity score computation.

5.3 Pairwise Homogeneous Network Alignment

Given two homogeneous networks $G^{(1)}$ and $G^{(2)}$, matching the nodes between them is an extremely challenging task, which is also called the *graph isomorphism* problem [12, 30]. By this context so far, no efficient algorithm exists that can address the problem in polynomial time. In this part, we will

[1] https://developers.google.com/maps/.
[2] https://www.bingmapsportal.com.
[3] https://developer.apple.com/maps/.
[4] https://docs.python.org/2/library/datetime.html.
[5] https://www.tutorialspoint.com/java/util/java_util_date.htm.

introduce several heuristics based network alignment models, and a mapping matrix inference model to solve the *pairwise homogeneous network alignment* problem.

5.3.1 Heuristics Based Network Alignment Model

The information generated by users' online social activities can indicate their personal characteristics. Besides the features extracted in Sect. 4.3.1 and user name/profile based heuristics introduced in Sect. 5.2, in this part, we will introduce a new category of measures, *Relative Centrality Difference* (RCD) [47, 48], which computes the similarity of user node centrality scores in different networks as the alignment results. Based on the assumption that "users with similar *centrality* scores are more likely to be the same user," the *relative centrality difference* can also be applied to solve the *unsupervised network alignment* problem.

The *centrality* concept introduced in Chap. 3.3.2 denotes the importance of nodes in the network structured data. Here, we assume that important users in one social network (like celebrities, movie stars and politicians) will be important as well in other networks. Viewed in such a perspective, the *centrality* of users in different networks can be an important signal for inferring the anchor links across networks.

Definition 5.1 (Relative Centrality Difference) Given two users $u_i^{(1)}, u_j^{(2)}$ from networks $G^{(1)}$ and $G^{(2)}$, respectively, let $C(u_i^{(1)})$ and $C(u_j^{(2)})$ denote the *centrality* scores of these two users, we can define the *relative centrality difference* (RCD) between them as

$$RCD\left(u_i^{(1)}, u_j^{(2)}\right) = \left(1 + \frac{|C\left(u_i^{(1)}\right) - C\left(u_j^{(2)}\right)|}{\left(C\left(u_i^{(1)}\right) + C\left(u_j^{(2)}\right)\right)/2}\right)^{-1}. \tag{5.7}$$

Depending on the *centrality* measures applied here, different types of *relative centrality difference* measures can be defined. For instance, if we use node degree [1, 6] as the *centrality* measure, the *relative degree difference* can be represented as

$$RDD\left(u_i^{(1)}, u_j^{(2)}\right) = \left(1 + \frac{|D\left(u_i^{(1)}\right) - D\left(u_j^{(2)}\right)|}{\left(D\left(u_i^{(1)}\right) + D\left(u_j^{(2)}\right)\right)/2}\right)^{-1}. \tag{5.8}$$

Meanwhile, if the *eigen-centrality* [6, 7] definition is adopted here, we can represent the nodes *relative eigen-centrality difference* measure as

$$RECD\left(u_i^{(1)}, u_j^{(2)}\right) = \left(1 + \frac{|C_{eigen}\left(u_i^{(1)}\right) - C_{eigen}\left(u_j^{(2)}\right)|}{\left(C_{eigen}\left(u_i^{(1)}\right) + C_{eigen}\left(u_j^{(2)}\right)\right)/2}\right)^{-1}. \tag{5.9}$$

Example 5.4 Based on the input homogeneous network structures shown in Fig. 5.1, if the *relative centrality difference* (with *degree* as the *centrality* measure) is adopted, we can compute the RDD

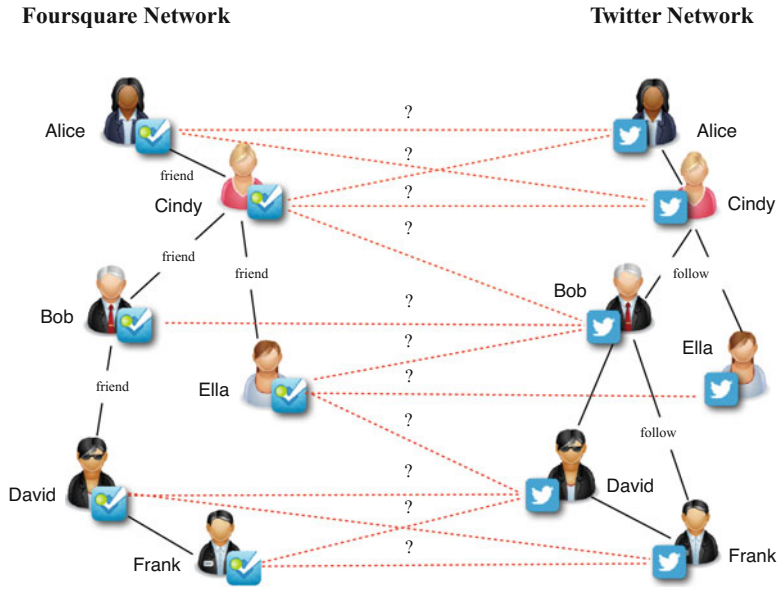

Fig. 5.1 An example of input pairwise homogeneous networks

scores for the three potential anchor links incident to $Ella_F$ as follows:

$$\begin{aligned} RDD(Ella_F, Bob_T) &= \left(1 + \frac{|D(Ella_F) - D(Bob_T)|}{(D(Ella_F) + D(Bob_T))/2}\right)^{-1} \\ &= \left(1 + \frac{|1-3|}{(1+3)/2}\right)^{-1} \\ &= \frac{1}{2} \end{aligned} \tag{5.10}$$

$$\begin{aligned} RDD(Ella_F, Ella_T) &= \left(1 + \frac{|D(Ella_F) - D(Ella_T)|}{(D(Ella_F) + D(Ella_T))/2}\right)^{-1} \\ &= \left(1 + \frac{|1-1|}{(1+1)/2}\right)^{-1} \\ &= 1 \end{aligned} \tag{5.11}$$

$$\begin{aligned} RDD(Ella_F, David_T) &= \left(1 + \frac{|D(Ella_F) - D(David_T)|}{(D(Ella_F) + D(David_T))/2}\right)^{-1} \\ &= \left(1 + \frac{|1-2|}{(1+2)/2}\right)^{-1} \\ &= \frac{3}{5} \end{aligned} \tag{5.12}$$

Therefore, compared with Bob_T and $David_T$, $Ella_F$ is more likely to be connected by the anchor link with $Ella_T$, since they have a closer node degree (both with degree 1) in the input network structures.

5.3.2 IsoRank

Another model called IsoRank [38] initially proposed to align the biomedical networks can also be applied to align the *homogeneous social networks*. The IsoRank algorithm has two stages. It first associates a score with each possible anchor link between user nodes across networks. For instance, we can denote $r(u_i^{(1)}, u_j^{(2)})$ as the reliability score of a potential anchor link $(u_i^{(1)}, u_j^{(2)})$ between the networks $G^{(1)}$ and $G^{(2)}$, and all such scores can be organized into a vector $\mathbf{r}$ of length $|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|$. The second stage of IsoRank is to construct the mapping for the networks by extracting information from the vector $\mathbf{r}$.

Definition 5.2 (Reliability Score) The *reliability score* $r(u_i^{(1)}, u_j^{(2)})$ of the anchor link $(u_i^{(1)}, u_j^{(2)})$ is highly correlated with the support provided by the neighbors of users $u_i^{(1)}$ and $u_j^{(2)}$. Therefore, we can define score $r(u_i^{(1)}, u_j^{(2)})$ as

$$r\left(u_i^{(1)}, u_j^{(2)}\right) = \sum_{u_m^{(1)} \in \Gamma\left(u_i^{(1)}\right)} \sum_{u_n^{(2)} \in \Gamma\left(u_i^{(2)}\right)} \frac{1}{|\Gamma\left(u_i^{(1)}\right)||\Gamma\left(u_j^{(2)}\right)|} r\left(u_m^{(1)}, u_n^{(2)}\right), \tag{5.13}$$

where sets $\Gamma(u_i^{(1)})$ and $\Gamma(u_i^{(2)})$ represent the neighbors of users $u_i^{(1)}$ and $u_i^{(1)}$, respectively, in networks $G^{(1)}$ and $G^{(2)}$.

The above definition is for the regular *unweighted networks*. Meanwhile, if the studied networks are weighted, and all the intra-network connections like $(u_i^{(1)}, u_m^{(1)})$ are associated with a weight $w(u_i^{(1)}, u_m^{(1)})$, we can represent the *reliability* measure of the weighted network as

$$r\left(u_i^{(1)}, u_j^{(2)}\right) = \sum_{u_m^{(1)} \in \Gamma\left(u_i^{(1)}\right)} \sum_{u_n^{(2)} \in \Gamma\left(u_j^{(2)}\right)} \frac{w\left(u_i^{(1)}, u_m^{(1)}\right) w\left(u_j^{(2)}, u_n^{(2)}\right) \cdot r\left(u_m^{(1)}, u_n^{(2)}\right)}{\left(\sum_{u_p^{(1)} \in \Gamma\left(u_i^{(1)}\right)} w\left(u_i^{(1)}, u_p^{(1)}\right)\right)\left(\sum_{u_q^{(2)} \in \Gamma\left(u_j^{(2)}\right)} w\left(u_j^{(2)}, u_q^{(2)}\right)\right)}, \tag{5.14}$$

As we can see, Eq. (5.13) is a special case of Eq. (5.14) with link weight $w(u_i^{(1)}, u_j^{(1)}) = 1$ for $u_i^{(1)} \in \mathcal{U}^{(1)}$ and $u_j^{(2)} \in \mathcal{U}^{(2)}$.

To simplify the problem settings, we will take the *unweighted social networks* as an example in the following analysis, and Eq. (5.13) can also be rewritten with a linear algebra as follows:

$$\mathbf{r} = \mathbf{A}\mathbf{r}, \tag{5.15}$$

where the matrix $\mathbf{A}$ is of dimension $(|\mathcal{U}^{(1)}||\mathcal{U}^{(2)}|) \times (|\mathcal{U}^{(1)}||\mathcal{U}^{(2)}|)$, and the row and column indexes correspond to different potential anchor links across the networks. The matrix entry

$$A(i,j)(p,q) = \begin{cases} \frac{1}{|\Gamma(u_i^{(1)})||\Gamma(u_j^{(2)})|}, & \text{if } (u_i^{(1)}, u_p^{(1)}) \in \mathcal{E}^{(1)}, (u_j^{(2)}, u_q^{(2)}) \in \mathcal{E}^{(2)}, \\ 0, & \text{otherwise}, \end{cases} \tag{5.16}$$

which corresponds the anchor links $(u_i^{(1)}, u_j^{(2)})$ and $(u_p^{(1)}, u_q^{(2)})$. As we can see, the above equation has a very similar representation to the stationary equation of *random walk*, whose solutions correspond to the principal eigenvector of the matrix $\mathbf{A}$ corresponding to the eigenvalue 1. For more information about the *random walk* model, please refer to Chap. 3.3.3.3.

Example 5.5 Here, we will use an example from [38] to illustrate the IsoRank algorithm. Formally, given the two input graphs as shown in Fig. 5.2, we can represent the *reliability* score of the potential anchor links across the networks with the following equations:

$$r_{aa'} = \frac{1}{4} r_{bb'}, \tag{5.17}$$

$$r_{bb'} = \frac{1}{3} r_{ac'} + \frac{1}{3} r_{a'c} + r_{aa'} + \frac{1}{9} r_{cc'}, \tag{5.18}$$

$$r_{cc'} = \frac{1}{4} r_{bb'} + \frac{1}{2} r_{be'} + \frac{1}{2} r_{bd'} + \frac{1}{2} r_{eb'} + \frac{1}{2} r_{db'} + r_{ee'} + r_{ed'} + r_{de'} + r_{dd'}, \tag{5.19}$$

$$r_{dd'} = \frac{1}{9} r_{cc'}, \tag{5.20}$$

$$r_{ee'} = \frac{1}{9} r_{cc'}. \tag{5.21}$$

By assigning the *reliability* scores of all these anchor links with random initial values and updating the scores according to the above equations, we can achieve the stationary score values of all the links as shown in the right matrix (the *reliability* score vector $\mathbf{r}$ is reshaped into a matrix for ease of viewing). For the entries without values, they are filled with the 0 by default.

IsoRank adopts a greedy strategy to select the anchor links from the *reliability* score matrix subject to the *one-to-one* cardinality constraint, where the anchor links corresponding to the largest *reliability* score will be selected first and no other anchor links incident to these selected nodes will be selected in the following rounds. For instance, among all the potential anchor links in this example, the

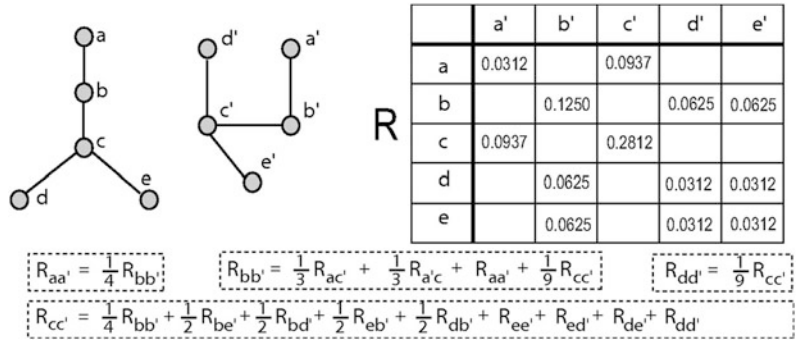

Fig. 5.2 An example of IsoRank

anchor links (c, c'), (b, b'), (a, a') will be selected first. As indicated in [38], by using sequence information, the ambiguities in the remaining anchor links (d, d'), (d, e'), (e, d'), and (e, e') with the same *reliability* scores can be resolved successfully.

5.3.3 IsoRankN

IsoRankN algorithm introduced in [29] is an extension to IsoRank, which uses a different method of spectral clustering on the induced graph of pairwise alignment scores. The new approach provides significant advantages not only over the original IsoRank algorithm but also over other methods. IsoRankN has four main steps: (1) initial network alignment with IsoRank, (2) star spread, (3) spectral partition, and (4) star merging. Next, we will introduce these four steps in detail.

5.3.3.1 Initial Network Alignment

Given k networks $G^{(1)}, G^{(2)}, \ldots, G^{(k)}$, IsoRankN computes the local alignment scores of node pairs across networks with the IsoRank algorithm. For instance, if the networks are unweighted, the alignment score between nodes $u_l^{(i)}$ and $u_m^{(j)}$ in networks $G^{(i)}$, $G^{(j)}$ can be denoted as:

$$r\left(u_l^{(i)}, u_m^{(j)}\right) = \sum_{u_p^{(i)} \in \Gamma\left(u_l^{(i)}\right)} \sum_{u_q^{(j)} \in \Gamma\left(u_m^{(j)}\right)} \frac{1}{|\Gamma\left(u_l^{(i)}\right)||\Gamma\left(u_m^{(j)}\right)|} r\left(u_p^{(i)}, u_q^{(j)}\right), \tag{5.22}$$

It will lead to a weighted k-partite graph, where the links denote the anchor links across networks weighted by the *reliability* scores calculated above. If the networks $G^{(1)}, \ldots G^{(k)}$ are all complete graphs, the alignment results will be the maximum weighted cliques. However, in the real world, such an assumption can hardly hold, and IsoRankN proposes to use a technique called "Star Spread" to select a subgraph with high weights.

5.3.3.2 Star Spread

For each node in a network, e.g., $u_l^{(i)}$ in network $G^{(i)}$, the set of nodes from all the other networks connected with $u_l^{(i)}$ via potential anchor links can be denoted as set $\Gamma(u_l^{(i)})$. The nodes in $\Gamma(u_l^{(i)})$ can be further pruned by removing those connected with *weak* anchor links. Here, the term "weak" denotes the anchor links with low *reliability* scores calculated with IsoRank. Formally, among all the nodes in $\Gamma(u_l^{(i)})$, we can denote the node connected to $u_l^{(i)}$ with the strongest link as $v^* = \arg_{v \in \Gamma(u_l^{(i)})} \max r(u_l^{(i)}, v)$. For all the nodes with weights lower than $\beta \cdot r(u_l^{(i)}, v^*)$ will be removed from $\Gamma(u_l^{(i)})$ (where β is a threshold parameter), and the remaining nodes together with $u_l^{(i)}$ will form a star structured graph $S_{u_l^{(i)}}$.

5.3.3.3 Spectral Partition

For each node $u_l^{(i)}$, IsoRankN aims at selecting a subgraph $S^*_{u_l^{(i)}}$ from the star-structured graph $S_{u_l^{(i)}}$, which contains the highly weighted neighbors of $u_l^{(i)}$. To achieve such an objective, IsoRankN proposes to identify a subgraph with a low *conductance* from $S_{u_l^{(i)}}$ instead.

Formally, given a network $G = (\mathcal{V}, \mathcal{E})$, let $\mathcal{S} \subset \mathcal{V}$ denote a node subset of G. The *conductance* of the subgraph involving $\mathcal{S}$ can be formally represented as

$$\phi(\mathcal{S}) = \frac{\sum_{u \in \mathcal{S}} \sum_{v \in \bar{\mathcal{S}}} r_{u,v}}{\min(\text{vol}(\mathcal{S}), \text{vol}(\bar{\mathcal{S}}))}, \tag{5.23}$$

where $\bar{\mathcal{S}} = \mathcal{V} \setminus \mathcal{S}$, and $\text{vol}(\mathcal{S}) = \sum_{u \in \mathcal{S}} \sum_{v \in \mathcal{V}} r_{u,v}$.

IsoRankN points out that a node subset $\mathcal{S}$ containing node $u_l^{(i)}$ can be computed effectively and efficiently with the personalized PageRank algorithm starting from node $u_l^{(i)}$, which will not be introduced here.

5.3.3.4 Star Merging

Considering that links in the selected star graph $S^*_{u_l^{(i)}}$ are all the anchor links across networks, there exist no intra-network links at all, e.g., the links in network $G^{(i)}$ only. However, in many cases, there may exist multiple nodes corresponding to the same entity inside the network as well. To solve such a problem, IsoRankN proposes a star merging step to combine several star graphs together, e.g., $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$.

Formally, given two star graphs $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$ if the following conditions both hold, $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$ can be merged into one star graph.

$$\forall v \in S^*_{u_m^{(j)}} \setminus \{u_m^{(j)}\}, r\left(v, u_l^{(i)}\right) \geq \beta \cdot \max_{v' \in \Gamma\left(u_l^{(i)}\right)} r\left(v', u_l^{(i)}\right), \tag{5.24}$$

$$\forall v \in S^*_{u_l^{(i)}} \setminus \{u_l^{(i)}\}, r\left(v, u_m^{(j)}\right) \geq \beta \cdot \max_{v' \in \Gamma\left(u_m^{(j)}\right)} r\left(v', u_m^{(j)}\right). \tag{5.25}$$

Via these aforementioned four steps, IsoRankN will be able to compute the alignment among multiple input networks, where steps (3) and (4) will repeat until all the nodes are assigned to a cluster.

5.3.4 Matrix Inference Based Network Alignment

For homogeneous network alignment, we will introduce another algorithm based on matrix inference at the end of this section. Formally, the set of *anchor links* actually define a mapping of nodes across networks, which is subject to the *one-to-one* constraint.

Formally, given a homogeneous network $G^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)})$, its structure can be organized as the adjacency matrix $\mathbf{A}_{G^{(1)}} \in \mathbb{R}^{|\mathcal{V}^{(1)}| \times |\mathcal{V}^{(1)}|}$. If network $G^{(1)}$ is unweighted, then matrix $\mathbf{A}_{G^{(1)}}$ will be a binary matrix and entry $A_{G^{(1)}}(i, p) = 1$ (or $A_{G^{(1)}}(u_i^{(1)}, u_p^{(1)}) = 1$) iff the correspond social link $(u_i^{(1)}, u_p^{(1)})$ exists in the link set $\mathcal{E}^{(1)}$. In the case that the network is weighted, the entries, e.g., $A_{G^{(1)}}(i, p)$, will denote the weight of link $(u_i^{(1)}, u_p^{(1)})$ and the entry will be 0 if the link $(u_i^{(1)}, u_p^{(1)})$ doesn't exist. The concept of *adjacency matrix* has been introduced when talking about the network representations in Sect. 3.2.1. In a similar way, we can also represent the social adjacency matrix $\mathbf{A}_{G^{(2)}}$ for network $G^{(2)}$ as well.

The network alignment problem actually aims at inferring a *one-to-one* node mappings, that can project nodes from one network to the other network. For instance, we can denote the mapping

between networks $G^{(1)}$ to $G^{(2)}$ as $f : \mathcal{V}^{(1)} \to \mathcal{V}^{(2)}$. Via the mapping f, besides the nodes, the network structure can be projected across networks as well. For instance, given a social connection $(u_i^{(1)}, u_p^{(1)})$ in $G^{(1)}$, we can represent its corresponding projected connection in $G^{(2)}$ as $(f(u_i^{(1)}), f(u_p^{(1)}))$.

Based on the assumption that the mapped nodes should have similar network connection patterns across different networks, via the mapping f, we can denote the projection error as the network structure differences introduced by the projection between $G^{(1)}$ and $G^{(2)}$, i.e.,

$$L(G^{(1)}, G^{(2)}, f) = \sum_{u_i^{(1)} \in \mathcal{V}^{(1)}} \sum_{u_p^{(1)} \in \mathcal{V}^{(1)}} \left(A_{G^{(2)}}\left(u_i^{(1)}, u_p^{(1)}\right) - A_{G^{(1)}}\left(f\left(u_i^{(1)}\right), f\left(u_p^{(1)}\right)\right)\right)^2. \quad (5.26)$$

Formally, such a node projection can be represented as a matrix $\mathbf{P} \in \{0, 1\}^{|\mathcal{V}^{(1)}| \times |\mathcal{V}^{(2)}|}$ as well, where entry $P(i, j) = 1$ iff anchor link $(u_i^{(1)}, u_j^{(2)})$ exists between networks $G^{(1)}$ and $G^{(2)}$. Via the matrix $\mathbf{P}$, we can represent the above loss term as

$$L(\mathbf{A}_{G^{(1)}}, \mathbf{A}_{G^{(2)}}, \mathbf{P}) = \left\| \mathbf{P}^\top \mathbf{A}_{G^{(1)}} \mathbf{P} - \mathbf{A}_{G^{(2)}} \right\|^2. \quad (5.27)$$

If there exists a perfect mapping of users across networks, we can obtain a mapping matrix $\mathbf{P}$ introducing zero loss in the above function, i.e., $L(\mathbf{A}_{G^{(1)}}, \mathbf{A}_{G^{(2)}}, \mathbf{P}) = 0$. However, in the real-world scenarios, few networks can be perfectly aligned together. Inferring the optimal mapping matrix $\mathbf{P}$ which can introduce the minimum loss can be represented as the following objective function

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \left\| \mathbf{P}^\top \mathbf{A}_{G^{(1)}} \mathbf{P} - \mathbf{A}_{G^{(2)}} \right\|^2, \quad (5.28)$$

where the matrix $\mathbf{P}$ is usually subject to some constraints, like $\mathbf{P}$ is binary and each row and column should contain at most one entry being filled with value 1.

In general, it is not easy to find the optimal solution to the above objective function, as it is a purely combinatorial optimization problem. Identifying the optimal solution requires the enumeration of all the potential user mapping across different networks. In [40], Umeyama provides an algorithm that can solve the function with a nearly optimal solution based on eigen-decomposition. If the readers are interested in the solution, please refer to [40] for more detailed information. In the following sections, we will introduce an approximation method to address a similar optimization problem by relaxing the hard binary constraint to real values instead.

5.4 Multiple Homogeneous Network Alignment with Transitivity Penalty

The works introduced in the previous section are all about pairwise network alignment, which focus on the alignment of two networks only. However, in the real world, people are normally involved in multiple (usually more than two) social networks simultaneously. In this section, we will focus on the simultaneous alignment problem of multiple (more than two) networks, which is called the M-NASA problem formally [47].

5.4.1 Multiple Network Alignment Problem Description

Example 5.6 To help illustrate the M-NASA problem more clearly, we also give an example in Fig. 5.3, which involves three different social networks (i.e., networks I, II, and III). Users in these three networks are all anonymized and their names are replaced with randomly generated meaningless identifiers. Each pair of these three anonymized networks can actually share some common users, e.g., "David" participates in both networks I and II simultaneously, "Bob" is using networks I and III concurrently, and "Charles" is involved in all these three networks at the same time. Besides these shared anchor users, in these three partially aligned networks, some users are involved in one single network only (i.e., the non-anchor users [49]), e.g., "Alice" in network I, "Eva" in network II, and "Frank" in network III. The M-NASA problem studied in this section aims at discovering the anchor links (i.e., the dashed bi-directional red lines) connecting anchor users across these three social networks.

The significant difference of M-NASA from existing *pairwise* network alignment problems is due to the "*transitivity law*" that anchor links follow. In traditional set theory [26], a relation $\mathcal{R}$ is defined to be a *transitive relation* in domain $\mathcal{X}$ iff $\forall a, b, c \in \mathcal{X}, (a, b) \in \mathcal{R} \wedge (b, c) \in \mathcal{R} \rightarrow (a, c) \in \mathcal{R}$. If we treat the union of user account sets of all these social networks as the target domain $\mathcal{X}$ and treat anchor links as the relation $\mathcal{R}$, then anchor links depict a "*transitive relation*" among users across networks. We can take the networks shown in Fig. 5.3 as an example. Let u be a user involved in networks I, II, and III simultaneously, whose accounts in these networks are u^I, u^{II}, and u^{III}, respectively. If anchor links (u^I, u^{II}) and (u^{II}, u^{III}) are identified in aligning networks (I, II) and networks (II, III), respectively (i.e., u^I, u^{II}, and u^{III} are discovered to be the same user), then anchor link (u^I, u^{III}) should also exist in the alignment result of networks (I, III) as well. In the M-NASA problem, we need to guarantee the inferred anchor links can meet the *transitivity law*.

Formally, the M-NASA problem can be formally defined as follows. Given the n isolated anonymized social networks $\{G^{(1)}, G^{(2)}, \ldots, G^{(n)}\}$, the M-NASA problem aims at discovering the anchor links among these n networks, i.e., the undirected anchor link sets $\mathcal{A}^{(1,2)}, \mathcal{A}^{(1,3)}, \ldots, \mathcal{A}^{(n-1,n)}$. Networks $G^{(1)}, G^{(2)}, \ldots, G^{(n)}$ are partially aligned and the constraint on anchor links in $\mathcal{A}^{(1,2)}, \mathcal{A}^{(1,3)}, \ldots, \mathcal{A}^{(n-1,n)}$ is *one-to-one*, which also follow the *transitivity law*.

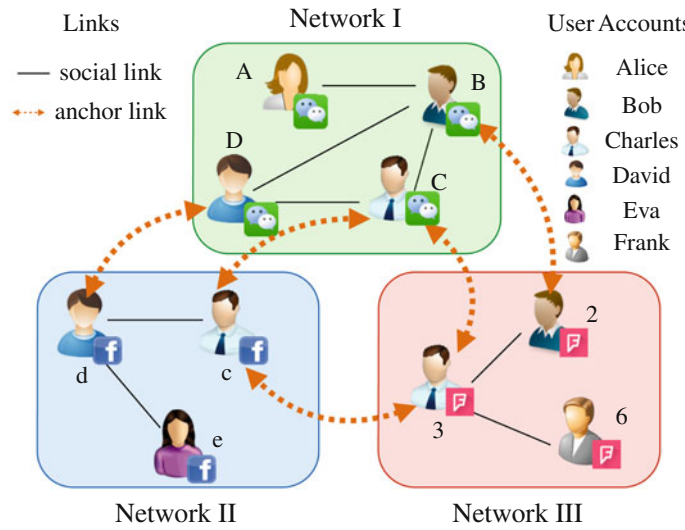

Fig. 5.3 An example of multiple anonymized partially aligned social networks

To solve the M-NASA problem, a novel network alignment framework UMA (Unsupervised Multi-network Alignment) proposed in [47] will be introduced in this section. UMA addresses the M-NASA problem with two steps: (1) unsupervised transitive anchor link inference across multi-networks, and (2) transitive multi-network matching to maintain the *one-to-one constraint*. In step (1), UMA infers a set of potential anchor links with unsupervised learning techniques by minimizing the *friendship inconsistency* and preserving the *alignment transitivity* property across networks. In step (2), UMA keeps the one-to-one constraint on anchor links by selecting those which can maximize the overall existence probabilities while maintaining the *matching transitivity* property at the same time. Next, we will introduce these two steps in great detail.

5.4.2 Unsupervised Multiple Network Alignment

In this part, we will introduce the first step of the UMA framework. We will first introduce the problem setting and objective function for pairwise network alignment based on matrix inference, which will be extended to the *multiple network* settings subject to the *transitivity law*. The integrated objective function can be addressed by relaxing the hard binary constraints, and the approximated solution will serve as the input for the second step of UMA to be introduced in Sect. 5.4.3.

5.4.2.1 Unsupervised Pairwise Network Alignment

Anchor links between any two given networks $G^{(i)}$ and $G^{(j)}$ actually define a *one-to-one* mapping (of users and social links) between $G^{(i)}$ and $G^{(j)}$. To evaluate the quality of different inferred mappings (i.e., the inferred anchor links), two new concepts of cross-network *Friendship Consistency/Inconsistency* will be introduced here. The optimal inferred anchor links are those which can maximize the *Friendship Consistency* (or minimize the *Friendship Inconsistency*) across networks.

As introduced in Sect. 5.3.4, given two partially aligned social networks $G^{(i)} = (\mathcal{U}^{(i)}, \mathcal{E}^{(i)})$ and $G^{(j)} = (\mathcal{U}^{(j)}, \mathcal{E}^{(j)})$, we can represent their corresponding *social adjacency* matrices to be $\mathbf{S}^{(i)} \in \mathbb{R}^{|\mathcal{U}^{(i)}|\times|\mathcal{U}^{(i)}|}$ and $\mathbf{S}^{(j)} \in \mathbb{R}^{|\mathcal{U}^{(j)}|\times|\mathcal{U}^{(j)}|}$, respectively.

Meanwhile, given the anchor link set $\mathcal{A}^{(i,j)} \subset \mathcal{U}^{(i)} \times \mathcal{U}^{(j)}$ between networks $G^{(i)}$ and $G^{(j)}$, the *binary transitional matrix* defined based on $\mathcal{A}^{(i,j)}$ can be represented as $\mathbf{T}^{(i,j)} \in \{0,1\}^{|\mathcal{U}^{(i)}|\times|\mathcal{U}^{(j)}|}$, where $\mathbf{T}^{(i,j)}(l,m) = 1$ iff link $(u_l^{(i)}, u_m^{(j)}) \in \mathcal{A}^{(i,j)}$, $u_l^{(i)} \in \mathcal{U}^{(i)}$, $u_m^{(j)} \in \mathcal{U}^{(j)}$. The *binary transitional matrix* from $G^{(j)}$ to $G^{(i)}$ can be defined in a similar way, which can be represented as $\mathbf{T}^{(j,i)} \in \{0,1\}^{|\mathcal{U}^{(j)}|\times|\mathcal{U}^{(i)}|}$, where $(\mathbf{T}^{(i,j)})^\top = \mathbf{T}^{(j,i)}$ as the anchor links between $G^{(i)}$ and $G^{(j)}$ are undirected. Considering that anchor links have an inherent *one-to-one* constraint, each row and each column of the *binary transitional matrices* $\mathbf{T}^{(i,j)}$ and $\mathbf{T}^{(j,i)}$ should have at most one entry filled with 1, which will pose a constraint on the inference space of potential *binary transitional matrices* $\mathbf{T}^{(i,j)}$ and $\mathbf{T}^{(j,i)}$.

Meanwhile, the *friendship inconsistency* can be defined as the number of non-shared social links between those mapped from $G^{(i)}$ and the original links in $G^{(j)}$. Based on the inferred *anchor transitional matrix* $\mathbf{T}^{(i,j)}$, the introduced *friendship inconsistency* between networks $G^{(i)}$ and $G^{(j)}$ can be represented as:

$$\left\| (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} - \mathbf{S}^{(j)} \right\|_F^2, \tag{5.29}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. And the optimal *binary transitional matrix* $\bar{\mathbf{T}}^{(i,j)}$, which can lead to the minimum *friendship inconsistency* can be represented as follows:

$$\begin{aligned}
&\bar{\mathbf{T}}^{(i,j)} = \arg\min_{\mathbf{T}^{(i,j)}} \left\| (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} - \mathbf{S}^{(j)} \right\|_F^2, \\
&s.t. \quad \mathbf{T}^{(i,j)} \in \{0,1\}^{|\mathcal{U}^{(i)}| \times |\mathcal{U}^{(j)}|}, \\
&\qquad \mathbf{T}^{(i,j)} \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1}, \\
&\qquad (\mathbf{T}^{(i,j)})^\top \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1},
\end{aligned} \tag{5.30}$$

where the last two equations are added to maintain the *one-to-one* constraint on anchor links. The inequality $\mathbf{X} \preccurlyeq \mathbf{Y}$ holds iff $\mathbf{X}$ is of the same dimensions as $\mathbf{Y}$ and every entry in $\mathbf{X}$ is no greater than the corresponding entry in $\mathbf{Y}$.

5.4.2.2 Multiple Network Alignment with Transitivity Constraint

Isolated network alignment can work well in addressing the alignment problem of two social networks. However, in the M-NASA problem studied in this section, multiple social networks (more than two) social networks are to be aligned simultaneously. Besides minimizing the *friendship inconsistency* between each pair of networks, the *transitivity* property of anchor links also needs to be preserved in the transitional matrices inference.

The *transitivity* property should hold for the alignment of any n networks, where the minimum of n is 3. To help illustrate the *transitivity property* more clearly, we will use three network alignment as an example to introduce the M-NASA problem, which can be easily generalized to the case of n networks alignment. Let $G^{(i)}$, $G^{(j)}$, and $G^{(k)}$ be three social networks to be aligned concurrently. To accommodate the alignment results and preserve the *transitivity* property, a new *alignment transitivity penalty* is introduced as follows:

Definition 5.3 (Alignment Transitivity Penalty) Let $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(i,k)}$ be the inferred binary transitional matrices from $G^{(i)}$ to $G^{(j)}$, from $G^{(j)}$ to $G^{(k)}$ and from $G^{(i)}$ to $G^{(k)}$, respectively, among these three networks. The *alignment transitivity penalty* $C(\{G^{(i)}, G^{(j)}, G^{(k)}\})$ introduced by the inferred transitional matrices can be quantified as the number of inconsistent social links being mapped from $G^{(i)}$ to $G^{(k)}$ via two different alignment paths $G^{(i)} \to G^{(j)} \to G^{(k)}$ and $G^{(i)} \to G^{(k)}$, i.e.,

$$\begin{aligned}
&C(\{G^{(i)}, G^{(j)}, G^{(k)}\}) \\
&\quad = \left\| (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} - (\mathbf{T}^{(i,k)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,k)} \right\|_F^2.
\end{aligned} \tag{5.31}$$

Alignment transitivity penalty is a general penalty concept and can be applied to n networks $\{G^{(1)}, G^{(2)}, \ldots, G^{(n)}\}, n \geq 3$ as well, which can be defined as the summation of penalty introduced by any three networks in the set, i.e.,

$$\begin{aligned}
&C(\{G^{(1)}, G^{(2)}, \ldots, G^{(n)}\}) \\
&\quad = \sum_{\forall \{G^{(i)}, G^{(j)}, G^{(k)}\} \subset \{G^{(1)}, G^{(2)}, \ldots, G^{(n)}\}} C(\{G^{(i)}, G^{(j)}, G^{(k)}\}).
\end{aligned} \tag{5.32}$$

The optimal *binary transitional matrices* $\bar{\mathbf{T}}^{(i,j)}$, $\bar{\mathbf{T}}^{(j,k)}$, and $\bar{\mathbf{T}}^{(k,i)}$ which can minimize friendship inconsistency and the *alignment transitivity penalty* at the same time will be

$$\begin{aligned}
&\bar{\mathbf{T}}^{(i,j)}, \bar{\mathbf{T}}^{(j,k)}, \bar{\mathbf{T}}^{(k,i)} \\
&\quad = \arg\min_{\mathbf{T}^{(i,j)},\mathbf{T}^{(j,k)},\mathbf{T}^{(k,i)}} \left\| (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} - \mathbf{S}^{(j)} \right\|_F^2 \\
&\qquad + \left\| (\mathbf{T}^{(j,k)})^\top \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} - \mathbf{S}^{(k)} \right\|_F^2 + \left\| (\mathbf{T}^{(k,i)})^\top \mathbf{S}^{(k)} \mathbf{T}^{(k,i)} - \mathbf{S}^{(i)} \right\|_F^2 \\
&\qquad + \alpha \cdot \left\| (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} - \mathbf{T}^{(k,i)} \mathbf{S}^{(i)} (\mathbf{T}^{(k,i)})^\top \right\|_F^2 \\
&s.t.\ \mathbf{T}^{(i,j)} \in \{0,1\}^{|\mathcal{U}^{(i)}| \times |\mathcal{U}^{(j)}|}, \mathbf{T}^{(j,k)} \in \{0,1\}^{|\mathcal{U}^{(j)}| \times |\mathcal{U}^{(k)}|} \\
&\quad \mathbf{T}^{(k,i)} \in \{0,1\}^{|\mathcal{U}^{(k)}| \times |\mathcal{U}^{(i)}|} \\
&\quad \mathbf{T}^{(i,j)} \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1}, (\mathbf{T}^{(i,j)})^\top \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1}, \\
&\quad \mathbf{T}^{(j,k)} \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1}, (\mathbf{T}^{(j,k)})^\top \mathbf{1}^{|\mathcal{U}^{(j)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1}, \\
&\quad \mathbf{T}^{(k,i)} \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1}, (\mathbf{T}^{(k,i)})^\top \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1},
\end{aligned} \tag{5.33}$$

where parameter α denotes the weight of the alignment transitivity penalty term.

5.4.2.3 Relaxation of the Optimization Problem

The objective function introduced in the previous subsection aims at obtaining the *hard* mappings among users across different networks and entries in all these *transitional matrices* are binary, which can lead to a fatal drawback: *hard assignment* can be neither possible nor realistic for networks with star structures as proposed in [25] and the hard subgraph isomorphism [27] is NP-hard.

To overcome such a problem, the UMA framework proposes to relax the binary constraint of entries in transitional matrices to allow them to be real values within range [0, 1]. Each entry in the transitional matrix represents a probability, denoting the confidence of certain user-user mapping across networks. Such a relaxation can make the *one-to-one* constraint no longer hold (which will be addressed with transitive network matching in the next subsection) as multiple entries in rows/columns of the transitional matrix can have non-zero values. To limit the existence of non-zero entries in the transitional matrices, we replace the one-to-one constraint, e.g.,

$$\mathbf{T}^{(k,i)} \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1}, (\mathbf{T}^{(k,i)})^\top \mathbf{1}^{|\mathcal{U}^{(k)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(i)}| \times 1} \tag{5.34}$$

with *sparsity constraints*

$$\left\| \mathbf{T}^{(k,i)} \right\|_0 \leq t \tag{5.35}$$

instead, where term $\|\mathbf{T}\|_0$ denotes the L_0 norm of matrix $\mathbf{T}$, i.e., the number of non-zero entries in $\mathbf{T}$, and t is a small positive number to limit the non-zero entries in the matrix (i.e., the sparsity). Furthermore, the framework UMA adds term $\|\mathbf{T}\|_0$ to the minimization objective function, as it can be hard to determine the value of t in the constraint.

Based on the above relaxations, we can obtain the updated objective function to be

$$\begin{aligned}
&\bar{\mathbf{T}}^{(i,j)}, \bar{\mathbf{T}}^{(j,k)}, \bar{\mathbf{T}}^{(k,i)} \\
&= \arg\min_{\mathbf{T}^{(i,j)},\mathbf{T}^{(j,k)},\mathbf{T}^{(k,i)}} \left\| (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} - \mathbf{S}^{(j)} \right\|_F^2 \\
&\quad + \left\| (\mathbf{T}^{(j,k)})^\top \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} - \mathbf{S}^{(k)} \right\|_F^2 + \left\| (\mathbf{T}^{(k,i)})^\top \mathbf{S}^{(k)} \mathbf{T}^{(k,i)} - \mathbf{S}^{(i)} \right\|_F^2 \\
&\quad + \alpha \cdot \left\| (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} - \mathbf{T}^{(k,i)} \mathbf{S}^{(i)} (\mathbf{T}^{(k,i)})^\top \right\|_F^2 \\
&\quad + \beta \cdot \left\| \mathbf{T}^{(i,j)} \right\|_0 + \gamma \cdot \left\| \mathbf{T}^{(j,k)} \right\|_0 + \theta \cdot \left\| \mathbf{T}^{(k,i)} \right\|_0 \\
&s.t.\ \mathbf{0}^{|\mathcal{U}^{(i)}| \times |\mathcal{U}^{(j)}|} \preccurlyeq \mathbf{T}^{(i,j)} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(i)}| \times |\mathcal{U}^{(j)}|}, \\
&\quad \mathbf{0}^{|\mathcal{U}^{(j)}| \times |\mathcal{U}^{(k)}|} \preccurlyeq \mathbf{T}^{(j,k)} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(j)}| \times |\mathcal{U}^{(k)}|}, \\
&\quad \mathbf{0}^{|\mathcal{U}^{(k)}| \times |\mathcal{U}^{(i)}|} \preccurlyeq \mathbf{T}^{(k,i)} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(k)}| \times |\mathcal{U}^{(i)}|},
\end{aligned} \tag{5.36}$$

which involves three variables $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(k,i)}$ simultaneously, obtaining the joint optimal solution for which at the same time is very hard and time consuming. The UMA framework proposes to address the above objective function by fixing two variables and updating the other variable alternatively with the gradient descent method [3]. As proposed in [25], if during the alternating updating steps, the entries of the transitional matrices become invalid (i.e., values less than 0 or greater than 1), UMA applies the projection technique introduced in [25] to adjust negative entries to 0 and change entries greater than 1 to 1 instead. With such a process, the updating equations of matrices $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, $\mathbf{T}^{(k,i)}$ at step $t+1$ are given as follows:

$$\begin{aligned}
&\mathbf{T}^{(i,j)}(t+1) \\
&\quad = \mathbf{T}^{(i,j)}(t) - \eta^{(i,j)} \frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}(t), \mathbf{T}^{(j,k)}(t), \mathbf{T}^{(k,i)}(t), \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(i,j)}},
\end{aligned} \tag{5.37}$$

$$\begin{aligned}
&\mathbf{T}^{(j,k)}(t+1) \\
&\quad = \mathbf{T}^{(j,k)}(t) - \eta^{(j,k)} \frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}(t+1), \mathbf{T}^{(j,k)}(t), \mathbf{T}^{(k,i)}(t), \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(j,k)}},
\end{aligned} \tag{5.38}$$

$$\begin{aligned}
&\mathbf{T}^{(k,i)}(t+1) \\
&\quad = \mathbf{T}^{(k,i)}(t) - \eta^{(k,i)} \frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}(t+1), \mathbf{T}^{(j,k)}(t+1), \mathbf{T}^{(k,i)}(t), \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(k,i)}}.
\end{aligned} \tag{5.39}$$

In the updating equations, $\eta^{(i,j)}$, $\eta^{(j,k)}$, and $\eta^{(k,i)}$ are the learning steps in updating $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(k,i)}$, respectively. The Lagrangian function of the objective function can be represented as

$$\begin{aligned}\mathcal{L}(\mathbf{T}^{(i,j)}, \mathbf{T}^{(j,k)}, \mathbf{T}^{(k,i)}, \beta, \gamma, \theta) &= \left\| (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} - \mathbf{S}^{(j)} \right\|_F^2 \\ &+ \left\| (\mathbf{T}^{(j,k)})^\top \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} - \mathbf{S}^{(k)} \right\|_F^2 + \left\| (\mathbf{T}^{(k,i)})^\top \mathbf{S}^{(k)} \mathbf{T}^{(k,i)} - \mathbf{S}^{(i)} \right\|_F^2 \\ &+ \alpha \cdot \left\| (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} - \mathbf{T}^{(k,i)} \mathbf{S}^{(i)} (\mathbf{T}^{(k,i)})^\top \right\|_F^2 \\ &+ \beta \cdot \left\| \mathbf{T}^{(i,j)} \right\|_0 + \gamma \cdot \left\| \mathbf{T}^{(j,k)} \right\|_0 + \theta \cdot \left\| \mathbf{T}^{(k,i)} \right\|_0 . \end{aligned} \tag{5.40}$$

Meanwhile, considering that $\|\cdot\|_0$ is not differentiable because of its discrete values [43], we will replace the $\|\cdot\|_0$ with the $\|\cdot\|_1$ instead (i.e., the sum of absolute values of all entries). Furthermore, as all the negative entries will be projected to 0, the L_1 norm of transitional matrix $\mathbf{T}$ can be represented as $\left\| \mathbf{T}^{(k,i)} \right\|_1 = \mathbf{1}^\top \mathbf{T}^{(k,i)} \mathbf{1}$ (i.e., the sum of all entries in the matrix). In addition, the Frobenius norm $\|\mathbf{X}\|_F^2$ can be represented with trace $\mathrm{Tr}(\mathbf{X}\mathbf{X}^\top)$.

The partial derivatives of function $\mathcal{L}$ with regard to $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(k,i)}$ will be:

$$\begin{aligned}(1) &\frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}, \mathbf{T}^{(j,k)}, \mathbf{T}^{(k,i)}, \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(i,j)}} \\ &= 2 \cdot \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} (\mathbf{T}^{(i,j)})^\top (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \\ &\quad + 2 \cdot (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \\ &\quad + 2\alpha \cdot \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top \\ &\quad + 2\alpha \cdot (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top \\ &\quad - 2 \cdot \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} (\mathbf{S}^{(j)})^\top - 2 \cdot (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{S}^{(j)} \\ &\quad - 2\alpha \cdot (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \mathbf{T}^{(k,i)} \mathbf{S}^{(i)} (\mathbf{T}^{(k,i)})^\top (\mathbf{T}^{(j,k)})^\top \\ &\quad - 2\alpha \cdot \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \mathbf{T}^{(k,i)} (\mathbf{S}^{(i)})^\top (\mathbf{T}^{(k,i)})^\top (\mathbf{T}^{(j,k)})^\top - \beta \cdot \mathbf{1}\mathbf{1}^\top . \end{aligned} \tag{5.41}$$

$$\begin{aligned}(2) &\frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}, \mathbf{T}^{(j,k)}, \mathbf{T}^{(k,i)}, \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(j,k)}} \\ &= 2 \cdot \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top (\mathbf{S}^{(j)})^\top \mathbf{T}^{(j,k)} \\ &\quad + 2 \cdot (\mathbf{S}^{(j)})^\top \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} \\ &\quad + 2\alpha \cdot (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \\ &\quad + 2\alpha \cdot (\mathbf{T}^{(i,j)})^\top (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \\ &\quad - 2 \cdot \mathbf{S}^{(j)} \mathbf{T}^{(j,k)} (\mathbf{S}^{(k)})^\top - 2 \cdot (\mathbf{S}^{(j)})^\top \mathbf{T}^{(j,k)} \mathbf{S}^{(k)} \\ &\quad - 2\alpha \cdot (\mathbf{T}^{(i,j)})^\top (\mathbf{S}^{(i)})^\top \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \mathbf{T}^{(k,i)} \mathbf{S}^{(i)} (\mathbf{T}^{(k,i)})^\top \\ &\quad - 2\alpha \cdot (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} \mathbf{T}^{(k,i)} (\mathbf{S}^{(i)})^\top (\mathbf{T}^{(k,i)})^\top - \gamma \cdot \mathbf{1}\mathbf{1}^\top . \end{aligned} \tag{5.42}$$

$$
\begin{aligned}
&(3)\frac{\partial \mathcal{L}\left(\mathbf{T}^{(i,j)}, \mathbf{T}^{(j,k)}, \mathbf{T}^{(k,i)}, \beta, \gamma, \theta\right)}{\partial \mathbf{T}^{(k,i)}} \\
&= 2 \cdot \mathbf{S}^{(k)}\mathbf{T}^{(k,i)}(\mathbf{T}^{(k,i)})^{\top}(\mathbf{S}^{(k)})^{\top}\mathbf{T}^{(k,i)} \\
&\quad + 2 \cdot (\mathbf{S}^{(k)})^{\top}\mathbf{T}^{(k,i)}(\mathbf{T}^{(k,i)})^{\top}\mathbf{S}^{(k)}\mathbf{T}^{(k,i)} \\
&\quad + 2\alpha\mathbf{T}^{(k,i)}(\mathbf{S}^{(i)})^{\top}(\mathbf{T}^{(k,i)})^{\top}\mathbf{T}^{(k,i)}\mathbf{S}^{(i)} \\
&\quad + 2\alpha\mathbf{T}^{(k,i)}\mathbf{S}^{(i)}(\mathbf{T}^{(k,i)})^{\top}\mathbf{T}^{(k,i)}(\mathbf{S}^{(i)})^{\top} \\
&\quad - 2 \cdot \mathbf{S}^{(k)}\mathbf{T}^{(k,i)}(\mathbf{S}^{(i)})^{\top} - 2 \cdot (\mathbf{S}^{(k)})^{\top}\mathbf{T}^{(k,i)}\mathbf{S}^{(i)} \\
&\quad - 2\alpha \cdot (\mathbf{T}^{(j,k)})^{\top}(\mathbf{T}^{(i,j)})^{\top}(\mathbf{S}^{(i)})^{\top}\mathbf{T}^{(i,j)}\mathbf{T}^{(j,k)}\mathbf{T}^{(k,i)}\mathbf{S}^{(i)} \\
&\quad - 2\alpha \cdot (\mathbf{T}^{(j,k)})^{\top}(\mathbf{T}^{(i,j)})^{\top}\mathbf{S}^{(i)}\mathbf{T}^{(i,j)}\mathbf{T}^{(j,k)}\mathbf{T}^{(k,i)}(\mathbf{S}^{(i)})^{\top} - \theta \cdot \mathbf{1}\mathbf{1}^{\top}.
\end{aligned} \tag{5.43}
$$

Such an iterative updating process will stop when all *transitional matrices* converge. The achieved variable matrices $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(k,i)}$ will serve as the input for the *transitive network matching* to be introduced in the following subsection to maintain both the *one-to-one* constraint and the *transitivity law* constraint.

5.4.3 Transitive Network Matching

The constraint relaxation in the previous section actually violates the *one-to-one* property on anchor links seriously, since each node in a network can be connected by multiple anchor links with different inferred scores in matrices $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$, and $\mathbf{T}^{(k,i)}$. To resolve such a problem, UMA adopts the *transitive network matching* step as proposed in [47] to prune the introduced redundant anchor links. The matching results (i.e., selected anchor links) need to meet both the *one-to-one* constraint and the *transitivity property*.

Formally, given two networks $G^{(i)}$ and $G^{(j)}$, each potential anchor link, e.g., $(u_l^{(i)}, u_m^{(j)})$, between $G^{(i)}$ and $G^{(j)}$ is associated with a binary variable $x_{l,m}^{(i,j)} \in \{0, 1\}$ to denote whether anchor link $(u_l^{(i)}, u_m^{(j)})$ is selected or not in the matching, where

$$
x_{l,m}^{(i,j)} = \begin{cases} 1 & \text{if } \left(u_l^{(i)}, u_m^{(j)}\right) \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.44}
$$

For each user in network $G^{(i)}$, e.g., $u_l^{(i)} \in \mathcal{U}^{(i)}$, at most one potential anchor link attached to him/her will be selected in the final alignment result with another network, e.g., $G^{(j)}$ (or $G^{(k)}$). So, based on the introduced binary variables, the *one-to-one* constraint on anchor links between networks $G^{(i)}$ and $G^{(j)}$ as well as between networks $G^{(i)}$ and $G^{(k)}$ can be represented as follows:

$$
\sum_{u_m^{(j)} \in \mathcal{U}^{(j)}} x_{l,m}^{(i,j)} \leq 1, \sum_{u_o^{(k)} \in \mathcal{U}^{(k)}} x_{l,o}^{(i,k)} \leq 1, \forall u_l^{(i)} \in \mathcal{U}^{(i)}. \tag{5.45}
$$

Similarly, we can also define the binary variables $x_{m,o}^{(j,k)}, x_{o,l}^{(k,i)} \in \{0, 1\}$ and the corresponding *one-to-one* constraints for potential anchor links $(u_m^{(j)}, u_o^{(k)})$ and $(u_o^{(k)}, u_l^{(i)})$ between networks $G^{(j)}$, $G^{(k)}$ and between networks $G^{(k)}$, $G^{(i)}$, respectively, to represent whether these links are selected or not.

According to the definition of "transitivity law," if anchor links $(u_l^{(i)}, u_m^{(j)})$ and $(u_m^{(j)}, u_o^{(k)})$ are selected $\forall l \in \{1, 2, \ldots, |\mathcal{U}^{(i)}|\}, m \in \{1, 2, \ldots, |\mathcal{U}^{(j)}|\}, o \in \{1, 2, \ldots, \mathcal{U}^{(k)}|\}$ in matching networks $G^{(i)}$, $G^{(j)}$ and networks $G^{(j)}$, $G^{(k)}$, then anchor link $(u_o^{(k)}, u_l^{(i)})$ should be selected as well in the matching of networks $G^{(k)}$, $G^{(i)}$, i.e., $x_{o,l}^{(k,i)} = 1$. In other words, in three networks matching, the case that only two variables in $\{x_{l,m}^{(i,j)}, x_{m,o}^{(j,k)}, x_{o,l}^{(k,i)}\}$ are assigned with value 1 while the remaining one is 0 cannot hold in the final matching results, i.e.,

$$\begin{aligned} x_{l,m}^{(i,j)} + x_{m,o}^{(j,k)} + x_{o,l}^{(k,i)} \neq 2, \forall l \in \{1, 2, \ldots, |\mathcal{U}^{(i)}|\}, \\ \forall m \in \{1, 2, \ldots, |\mathcal{U}^{(j)}|\}, \forall o \in \{1, 2, \ldots, \mathcal{U}^{(k)}|, \end{aligned} \tag{5.46}$$

which is called the *matching transitivity constraint*. The *matching transitivity constraint* can be easily generalized to the case of matching$n$ $(n \geq 3)$ networks.

Definition 5.4 (Matching Transitivity Constraint) Let $\mathcal{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(n)}\}$ be a set of n networks, the *matching transitivity constraint* (MTC) for matching these n networks in $\mathcal{G}$ can be defined recursively as follows:

$$\text{MTC}(\mathcal{G}) = \left\{ \sum x_{\mathcal{G}} \neq |\mathcal{G}| - 1 \right\} \cup \left\{ \bigcup_{\mathcal{G}' \subset \mathcal{G}, |\mathcal{G}'| = |\mathcal{G}| - 1} \text{MTC}(\mathcal{G}') \right\}. \tag{5.47}$$

In the above equation, the variable summation term $\sum x_{\mathcal{G}} = x_{l,m}^{(1,2)} + x_{m,o}^{(2,3)} + \cdots + x_{p,l}^{(n,1)}, \forall l \in \{1, 2, \ldots, |\mathcal{U}^{(1)}|\}, \forall m \in \{1, 2, \ldots, |\mathcal{U}^{(2)}|\}, \ldots, \forall p \in \{1, 2, \ldots, |\mathcal{U}^{(n)}|\}$ represents the transitivity constraint involving all these n networks.

The final selected anchor links should be those with high confidence scores in the inferred *transitional matrices* but also can meet the *one-to-one matching* constraint and *matching transitivity* constraint simultaneously. The *transitive network matching* can be formulated as the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{x}^{(i,j)}, \mathbf{x}^{(j,k)}, \mathbf{x}^{(k,i)}} \sum_{l,m} x_{l,m}^{(i,j)} \mathbf{T}^{(i,j)}(l, m) + \sum_{l,m} x_{l,m}^{(i,j)} \mathbf{T}^{(i,j)}(l, m) \\ &\quad + \sum_{l,m} x_{l,m}^{(i,j)} \mathbf{T}^{(i,j)}(l, m), \\ &s.t. \sum_{u_m^{(j)} \in \mathcal{U}^{(j)}} x_{l,m}^{(i,j)} \leq 1, \sum_{u_o^{(k)} \in \mathcal{U}^{(k)}} x_{l,o}^{(i,k)} \leq 1, \forall u_l^{(i)} \in \mathcal{U}^{(i)}, \\ &\quad \sum_{u_l^{(i)} \in \mathcal{U}^{(i)}} x_{m,l}^{(j,i)} \leq 1, \sum_{u_o^{(k)} \in \mathcal{U}^{(k)}} x_{m,o}^{(j,k)} \leq 1, \forall u_m^{(j)} \in \mathcal{U}^{(j)}, \end{aligned}$$

$$\sum_{u_l^{(i)} \in \mathcal{U}^{(i)}} x_{o,l}^{(k,i)} \leq 1, \sum_{u_m^{(j)} \in \mathcal{U}^{(j)}} x_{o,m}^{(k,j)} \leq 1, \forall u_o^{(k)} \in \mathcal{U}^{(k)},$$
$$x_{l,m}^{(i,j)} + x_{m,o}^{(j,k)} + x_{o,l}^{(k,i)} \neq 2, \forall l \in \{1, 2, \ldots, |\mathcal{U}^{(i)}|\},$$
$$\forall m \in \{1, 2, \ldots, |\mathcal{U}^{(j)}|\}, \forall o \in \{1, 2, \ldots, \mathcal{U}^{(k)}|,$$
$$x_{l,m}^{(i,j)} \in \{0, 1\}, \forall u_l^{(i)} \in \mathcal{U}^{(i)}, u_m^{(j)} \in \mathcal{U}^{(j)}.$$
$$x_{m,o}^{(j,k)} \in \{0, 1\}, \forall u_m^{(j)} \in \mathcal{U}^{(j)}, u_o^{(k)} \in \mathcal{U}^{(k)}.$$
$$x_{o,l}^{(k,i)} \in \{0, 1\}, \forall u_o^{(k)} \in \mathcal{U}^{(k)}, u_l^{(i)} \in \mathcal{U}^{(i)}. \tag{5.48}$$

In the above objective function, the matching transitivity constraint $x_{l,m}^{(i,j)} + x_{m,o}^{(j,k)} + x_{o,l}^{(k,i)} \neq 2$ is actually non-convex, which can be another challenge in addressing the function. In [47], framework UMA proposes to (1) remove the matching transitivity constraint from the objective function, and (2) apply the matching transitivity constraint to post-process the solution (obtained from the objective function without the constraint).

The objective function (with the matching transitivity constraint removed) can be solved with open source optimization toolkit, e.g., Scipy.Optimization[6] and GLPK,[7] and we will not describe how to solve in detail due to the limited space. Among all the obtained solutions, we can check all the links whose corresponding variables meeting $x_{l,m}^{(i,j)} + x_{m,o}^{(j,k)} + x_{o,l}^{(k,i)} = 2$ and assign the variable with value 0 with 1 instead. For example, for three given variables $x_{l,m}^{(i,j)}$, $x_{m,o}^{(j,k)}$ and $x_{o,l}^{(k,i)}$, if $x_{l,m}^{(i,j)} = x_{m,o}^{(j,k)} = 1$ but $x_{o,l}^{(k,i)} = 0$, we will assign $x_{o,l}^{(k,i)}$ with new value 1 and $x_{o,x}^{(k,i)} = 0, \forall x \neq l$, $x_{x,l}^{(k,i)} = 0, \forall x \neq o$ to preserve the matching transitivity constraint.

5.5 Heterogeneous Network Co-alignment

The real-world social networks usually contain heterogeneous information, including both very complex network structures and different categories of attribute information, which can all provide extra signals for inferring the potential anchor links across networks. Besides these common users, social networks offering similar services can also share other common information entities, e.g., locations shared between Foursquare and Yelp, and products sold in both Amazon and Ebay. To distinguish the anchor links between different types of information entities, those aligning the common users are called the *user anchor links*, while those connecting locations (or products) are called the *location anchor links* (or *product anchor links*).

In this section, we will study the problem to infer different categories of anchor links connecting various anchor instances across social networks simultaneously, which is formally defined as the network "Partial Co-alignmenT" (PCT) problem. PCT is a general research problem and can be applied to different types of social networks, like Foursquare and Yelp, Amazon and eBay. In this section, we will take online social networks as an example, and will mainly focus on the partial co-alignment of location based social networks via shared users and locations with the various connection and attribute information available in the networks.

[6] http://docs.scipy.org/doc/scipy/reference/optimize.html.
[7] http://www.gnu.org/software/glpk/.

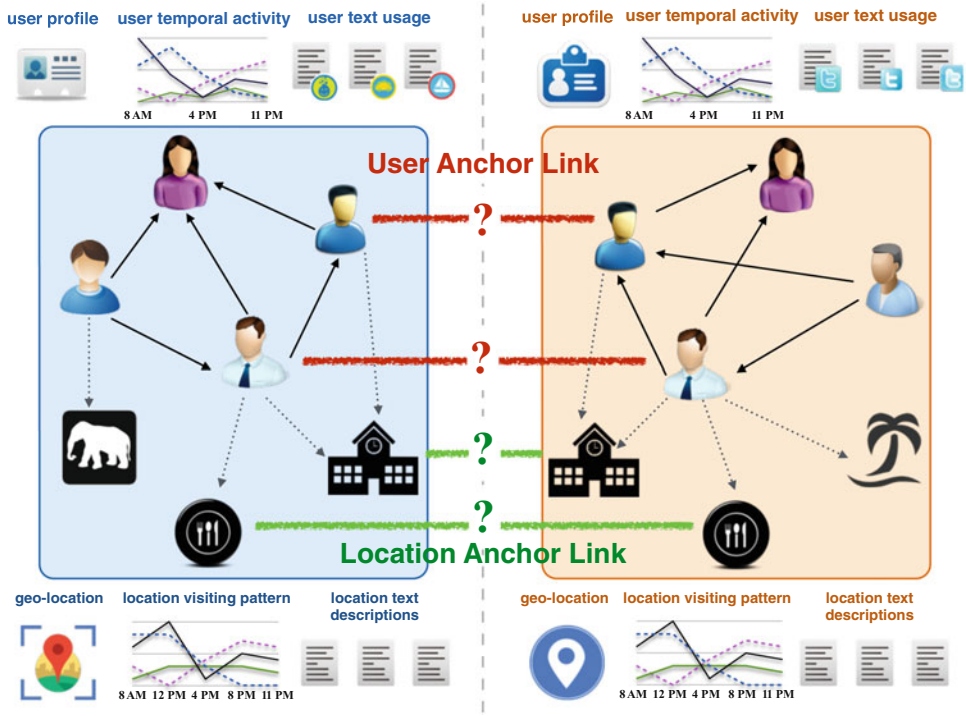

Fig. 5.4 An example of the PCT problem

5.5.1 Network Co-alignment Problem Description

Example 5.7 As shown in Fig. 5.4, we have two location based online social networks. In these social networks, users can check-in at the locations of their interest. Besides the connections among users and those between users and locations, both the users and locations are associated with different types of attribute information as well. For instance, for the users, we can obtain their profile information, the temporal activities information, and the posts written by them. As introduced before, these different types of attribute information can reveal the personal characteristics and is helpful for identifying the shared common users. Similarly, for the locations, we can have the profile information of them, and can also accumulate the timestamps and words from the posts attaching check-ins at these locations.

The timestamps and words of posts written at these places can reveal the characteristics of the locations as well. For instance, as illustrated in Fig. 5.5, we have two totally different locations: the Lincoln Park Zoo[8] and Scarlet Bar.[9] The Lincoln Park Zoo is the largest free zoo in Chicago and is open during 10:00 AM–5:00 PM. The Scarlet Bar is one of the most famous bars in Chicago, where people can drink with friends, dance to enjoy their night life, and it is open during 8:00 PM–2:00 AM.

We also have four online posts published by people at these two places in either Foursquare or Twitter. From the content of these posts, we find that people usually publish words about animals, pictures, and the scene at the Lincoln Park Zoo. However, people who visit the Scarlet Bar mainly talk about the atmosphere in the bar, the drinks, the dance floor, and the music there. Meanwhile, we can also accumulate the timestamps of posts published at these two places. The timestamps of posts

[8] http://www.lpzoo.org.

[9] http://www.scarletbarchicago.com.

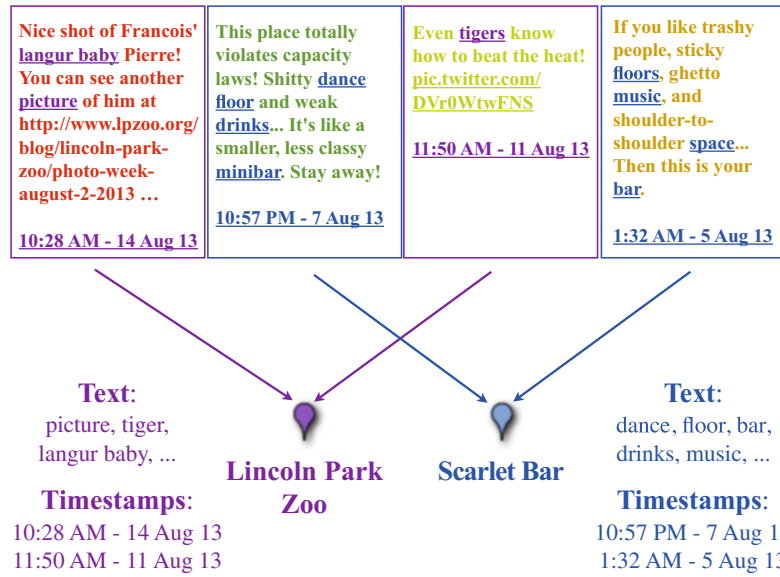

Fig. 5.5 An example of information accumulation for locations

published at the Lincoln Park Zoo are mostly during the daytime, while those of posts published at the Scarlet Bar are at night. Such accumulated information can serve as the profile information of these locations, respectively.

Based on the above example and descriptions, we can define the PCT problem as follows. For any two given social networks $G^{(1)}$ and $G^{(2)}$, with the link and attribute information in both $G^{(1)}$ and $G^{(2)}$, the PCT problem aims at inferring the potential anchor links between users and locations across $G^{(1)}$ and $G^{(2)}$, respectively. To differentiate the user anchor links and location anchor links, we use sets $\mathcal{A}_u^{(1,2)}$ and $\mathcal{A}_l^{(1,2)}$ to represent these two types of anchor links between networks $G^{(1)}$ and $G^{(2)}$, respectively. To address the problem, in this section, we will introduce UNICOAT model proposed in [48].

5.5.2 Anchor Link Co-inference

As introduced in the problem definition, anchor set $\mathcal{A}_u^{(1,2)}$ between networks $G^{(1)}$ and $G^{(2)}$ actually maps users between networks $G^{(1)}$ and $G^{(2)}$. Considering that users in different social networks are associated with both links and attribute information, the quality of the inferred anchor links $\mathcal{A}_u^{(1,2)}$ can be measured by the costs introduced by such mappings calculated with users' link and attribute information, i.e.,

$$cost(\mathcal{A}_u^{(1,2)}) = \text{ cost in links } (\mathcal{A}_u^{(1,2)}) + \alpha \cdot \text{ cost in attributes}(\mathcal{A}_u^{(1,2)}), \tag{5.49}$$

where α denotes the weight of the cost obtained from the attribute information (α is set as 1 in the experiments for simplicity, i.e., the link and attribute information is treated to be of the same importance). Considering that locations are also attached with link and attributes, similar cost function can be defined for the inferred location anchor links in $\mathcal{A}_l^{(1,2)}$:

$$cost(\mathcal{A}_l^{(1,2)}) = \text{ cost in links } (\mathcal{A}_l^{(1,2)}) + \alpha \cdot \text{ cost in attributes}(\mathcal{A}_l^{(1,2)}). \tag{5.50}$$

The optimal user and location anchor links $(\mathcal{A}_u^{(1,2)})^*$ and $(\mathcal{A}_l^{(1,2)})^*$ to be inferred in the PCT problem that minimize the cost functions can be represented as

$$(\mathcal{A}_u^{(1,2)})^*, (\mathcal{A}_l^{(1,2)})^* = \arg \min_{\mathcal{A}_u^{(1,2)}, \mathcal{A}_l^{(1,2)}} cost(\mathcal{A}_u^{(1,2)}) + cost(\mathcal{A}_l^{(1,2)}). \tag{5.51}$$

To resolve the objective function, in the following parts of this section, we will introduce the isolated user anchor link inference in Sect. 5.5.2.1, the isolated location anchor link inference in Sect. 5.5.2.2, and the joint co-inference framework of user and location anchor links in Sect. 5.5.3.

5.5.2.1 User Anchor Links Inference

Social connections among users clearly illustrate the social community structures of users in online social networks. Meanwhile, attribute information (e.g., profile information, text usage patterns, temporal activities) can reveal users' unique personal characteristics. Common users in different networks tend to form similar community structures [46] and have very close personal characteristics [45]. As a result, link and attribute information about the users both play very important roles in inferring potential user anchor links across networks. In this part, we will introduce how to use such information to improve the user anchor link inference results.

User Anchor Link Inference with Link Information

Similar to the matrix inference based network alignment approach introduced in Sect. 5.3.4, based on the social links among users in both $G^{(1)}$ and $G^{(2)}$ (i.e., $\mathcal{E}_{u,u}^{(1)}$ and $\mathcal{E}_{u,u}^{(2)}$, respectively), we can construct the binary *social adjacency matrices* [32] $\mathbf{S}^{(1)} \in \mathbb{R}^{|\mathcal{U}^{(1)}|\times|\mathcal{U}^{(1)}|}$ and $\mathbf{S}^{(2)} \in \mathbb{R}^{|\mathcal{U}^{(2)}|\times|\mathcal{U}^{(2)}|}$ for networks $G^{(1)}$ and $G^{(2)}$, respectively. Meanwhile, via the inferred user anchor links $\mathcal{A}_u^{(1,2)}$, users as well as their social connections can be mapped between networks $G^{(1)}$ and $G^{(2)}$. We can represent the inferred user anchor links $\mathcal{A}_u^{(1,2)}$ with binary *user transitional matrix* $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}^{(1)}|\times|\mathcal{U}^{(2)}|}$, where the ($i$th, lth) entry $P(i,l) = 1$ iff link $(u_i^{(1)}, u_l^{(2)}) \in \mathcal{A}_u^{(1,2)}$. Considering that the constraint on user anchor links is *one-to-one*, each column and each row of $\mathbf{P}$ can contain at most one entry being assigned with value 1, i.e.,

$$\mathbf{P}\mathbf{1}^{|\mathcal{U}^{(2)}|\times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(1)}|\times 1},\ \mathbf{P}^\top \mathbf{1}^{|\mathcal{U}^{(1)}|\times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(2)}|\times 1}. \tag{5.52}$$

The optimal user anchor links are those which can minimize the inconsistency of mapped social links across networks and the cost introduced by the inferred user anchor link set $\mathcal{A}_u^{(1,2)}$ with the link information can be represented as

$$\text{cost in link}(\mathcal{A}_u^{(1,2)}) = \text{cost in link}(\mathbf{P}) = \left\| \mathbf{P}^\top \mathbf{S}^{(1)} \mathbf{P} - \mathbf{S}^{(2)} \right\|_F^2, \tag{5.53}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of the corresponding matrix.

User Anchor Link Inference with Attribute Information

Besides social links, users in social networks can be associated with a set of attributes, which can provide extra hints for identifying the correspondence relationships about users across networks. In this part, we will introduce the method to infer the user anchor links with attribute information, which includes *username information*, *text usage patterns*, and *temporal activity information*.

Username that can differentiate users from each other in online social networks is like their online ID, which is an important factor in inferring potential anchor links. Let $(u_i^{(1)}, u_l^{(2)})$ be a potential

anchor link between $G^{(1)}$ and $G^{(2)}$, the usernames of $u_i^{(1)}$ and $u_l^{(2)}$ can be represented as two sets of characters $n(u_i^{(1)})$ and $n(u_l^{(2)})$, respectively, based on which various metrics introduced in Sect. 5.2.1 can be applied to measure the similarity between $u_i^{(1)}$ and $u_l^{(2)}$. For instance, by adopting the *token based Jaccard's coefficient* measure, we can compute the user similarity score between users $u_i^{(1)}$ and $u_j^{(2)}$ as follows:

$$sim\left(n\left(u_i^{(1)}\right), n\left(u_l^{(2)}\right)\right) = \frac{|n\left(u_i^{(1)}\right) \cap n\left(u_l^{(2)}\right)|}{|n\left(u_i^{(1)}\right) \cup n\left(u_l^{(2)}\right)|}. \tag{5.54}$$

Users usually have their unique active temporal patterns in online social networks [24]. For example, some users like to socialize with their online friends in the early morning, but some may prefer to do so in the evening after work. Users' online active time can be extracted based on their post publishing timestamps effectively. Let $\mathbf{t}(u_i^{(1)})$ and $\mathbf{t}(u_l^{(2)})$ be the normalized temporal activity distribution vectors of users $u_i^{(1)}$ and $u_l^{(2)}$, which are both of length 24. Entries of $\mathbf{t}(u_i^{(1)})$ and $\mathbf{t}(u_l^{(2)})$ contain the ratios of posts being published at the corresponding hour in a day. For example, $\mathbf{t}(u_i^{(1)})(3)$ denotes the ratio of all posts written by $u_1^{(1)}$ at 3 AM. Based on vectors $\mathbf{t}(u_i^{(1)})$ and $\mathbf{t}(u_l^{(2)})$, we can calculate the inner product of the temporal distribution vectors [24] as the similarity scores between $u_i^{(1)}$ and $u_l^{(2)}$ in their temporal activity patterns, i.e.,

$$sim\left(\mathbf{t}\left(u_i^{(1)}\right), \mathbf{t}\left(u_l^{(2)}\right)\right) = \mathbf{t}\left(u_i^{(1)}\right)^\top \mathbf{t}\left(u_l^{(2)}\right). \tag{5.55}$$

Besides profile and online activity temporal distribution information, people normally have very different text usage habits online [45], which can reveal personal unique characteristics and can be applied in inferring the user anchor links across networks. We represent the text content used by users $u_i^{(1)}$ and $u_l^{(2)}$ as bag-of-words vectors [24], $\mathbf{w}(u_i^{(1)})$ and $\mathbf{w}(u_l^{(2)})$, weighted by TF-IDF [23], respectively. Commonly used text similarity measure: *Cosine similarity* [10] can be applied to measure the similarities in text usage patterns between $u_i^{(1)}$ and $u_l^{(2)}$, i.e.,

$$sim\left(\mathbf{w}\left(u_i^{(1)}\right), \mathbf{w}\left(u_l^{(2)}\right)\right) = \frac{\mathbf{w}\left(u_i^{(1)}\right)^\top \cdot \mathbf{w}\left(u_l^{(2)}\right)}{\left\|\mathbf{w}\left(u_i^{(1)}\right)\right\| \cdot \left\|\mathbf{w}\left(u_l^{(2)}\right)\right\|}. \tag{5.56}$$

With these different attribute information (i.e., username, temporal activity, and text content), we can calculate the similarities between users across networks $G^{(1)}$ and $G^{(2)}$. We represent such similarity matrix as $\mathbf{\Lambda} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|}$, where entry $\mathbf{\Lambda}(i,l)$ is the similarity between $u_i^{(1)}$ and $u_l^{(2)}$. $\mathbf{\Lambda}(i,l)$ can be represented as a combination of $sim(n(u_i^{(1)}), n(u_l^{(2)}))$, $sim(\mathbf{t}(u_i^{(1)}), \mathbf{t}(u_l^{(2)}))$, and $sim(\mathbf{w}(u_i^{(1)}), \mathbf{w}(u_l^{(2)}))$ and linear combination is used here due to its simplicity and wide usages. The optimal weights of similarity scores calculated with different attribute information can be learned from the data theoretically, but it will make the model too complicated. To focus on the co-alignment problem itself, we can assume they are all of the same importance and propose to assign them with the same weight for simplicity. In other words, we have

$$\begin{aligned}\mathbf{\Lambda}(i,l) = \frac{1}{3}\Big(&sim\left(n\left(u_i^{(1)}\right), n\left(u_l^{(2)}\right)\right) + sim\left(\mathbf{t}\left(u_i^{(1)}\right), \mathbf{t}\left(u_l^{(2)}\right)\right) \\ &+ sim\left(\mathbf{w}\left(u_i^{(1)}\right), \mathbf{w}\left(u_l^{(2)}\right)\right)\Big).\end{aligned} \tag{5.57}$$

Similar users across social networks are more likely to be the same user and user anchor links in $\mathcal{A}_u^{(1,2)}$ that align similar users together should lead to lower cost. The cost function introduced by the inferred user anchor links $\mathcal{A}_u^{(1,2)}$ with attribute information can be represented as

$$\text{cost in attribute}(\mathcal{A}_u^{(1,2)}) = \text{cost in attribute}(\mathbf{P}) = -\|\mathbf{P} \circ \mathbf{\Lambda}\|_1, \tag{5.58}$$

where $\|\cdot\|_1$ is the L_1 norm [33] of the corresponding matrix, entry $(\mathbf{P} \circ \mathbf{\Lambda})(i, l)$ can be represented as $\mathbf{P}(i, l) \cdot \mathbf{\Lambda}(i, l)$ and $\mathbf{P} \circ \mathbf{\Lambda}$ denotes the Hadamard product [9] of matrices $\mathbf{P}$ and $\mathbf{\Lambda}$.

User Anchor Link Inference with Link and Attribute Information
Both link and attribute information can be very important for user anchor link inference. By taking these two categories of information into considerations simultaneously, the cost introduced by the inferred user anchor link set $\mathcal{A}_u^{(1,2)}$ can be represented as

$$\begin{aligned} cost(\mathcal{A}_u^{(1,2)}) &= \text{cost in link}(\mathcal{A}_u^{(1,2)}) + \alpha \cdot \text{cost in attribute}(\mathcal{A}_u^{(1,2)}) \\ &= \left\|\mathbf{P}^\top \mathbf{S}^{(1)} \mathbf{P} - \mathbf{S}^{(2)}\right\|_F^2 - \alpha \cdot \|\mathbf{P} \circ \mathbf{\Lambda}\|_1. \end{aligned} \tag{5.59}$$

The optimal *user transitional matrix* $\mathbf{P}^*$ which can lead to the minimum cost will be achieved by addressing the following objective function:

$$\begin{aligned} \mathbf{P}^* &= \arg\min_{\mathbf{P}} cost(\mathcal{A}_u^{(1,2)}) \\ &= \arg\min_{\mathbf{P}} \left\|\mathbf{P}^\top \mathbf{S}^{(1)} \mathbf{P} - \mathbf{S}^{(2)}\right\|_F^2 - \alpha \cdot \|\mathbf{P} \circ \mathbf{\Lambda}\|_1 \\ s.t. \quad & \mathbf{P} \in \{0, 1\}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|}, \\ & \mathbf{P}\mathbf{1}^{|\mathcal{U}^{(2)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1}, \mathbf{P}^\top \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(2)}| \times 1}. \end{aligned} \tag{5.60}$$

5.5.2.2 Location Anchor Links Inference

Similar to users, locations in online social networks are also associated with both link and attribute information (like the location links between users and locations, profile information and text descriptions about the locations, as well as the (longitude, latitude) coordinate information). The (longitude, latitude) pairs of the same location in different networks are usually not identical and various nearby locations can have very close coordinates, which pose great challenges in differentiating the locations from each other.

Location Anchor Link Inference with Link Information
Let $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$ be the sets of locations in networks $G^{(1)}$ and $G^{(2)}$, respectively. Based on the location links between users and locations in networks $G^{(1)}$ and $G^{(2)}$ (i.e., $\mathcal{E}_{u,l}^{(1)}$ and $\mathcal{E}_{u,l}^{(2)}$), we can construct the binary *location adjacency matrices* $\mathbf{L}^{(1)} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{L}^{(1)}|}$ and $\mathbf{L}^{(2)} \in \mathbb{R}^{|\mathcal{U}^{(2)}| \times |\mathcal{L}^{(2)}|}$ for networks $G^{(1)}$ and $G^{(2)}$, respectively. Entries in $\mathbf{L}^{(1)}$ and $\mathbf{L}^{(1)}$ $\left(\text{e.g., } \mathbf{L}^{(1)}(i, j) \text{ and } \mathbf{L}^{(2)}(l, m)\right)$ are filled with value 1 iff user $u_i^{(1)}$ has visited location $l_j^{(1)}$ in $G^{(1)}$ and user $u_l^{(2)}$ has visited location $l_m^{(2)}$ in $G^{(2)}$.

Besides the *user transitional matrix* $\mathbf{P}$ which maps users between $G^{(1)}$ and $G^{(2)}$, we can also construct the binary *location transitional matrix* $\mathbf{Q} \in \{0, 1\}^{|\mathcal{L}^{(1)}| \times |\mathcal{L}^{(2)}|}$ based on the inferred location anchor link set $\mathcal{A}_l^{(1,2)}$, which maps locations between $G^{(1)}$ and $G^{(2)}$. The cost introduced by the

inferred location anchor link set $\mathcal{A}_l^{(1,2)}$ can be defined as the number of mis-mapped location links across networks, i.e.,

$$\text{cost in link}(\mathcal{A}_l^{(1,2)}) = \left\| \mathbf{P}^\top \mathbf{L}^{(1)} \mathbf{Q} - \mathbf{L}^{(2)} \right\|_F^2. \tag{5.61}$$

Location Anchor Link Inference with Attribute Information

In location-based social networks, each location has its own profile page, which shows the name and all the review comments about the location. Similar to the similarity scores for user anchor links, for any two locations $l_i \in \mathcal{L}^{(1)}$ and $l_m \in \mathcal{L}^{(2)}$, based on the names of locations l_i and l_m, we can calculate the similarity scores between l_i and l_m to be

$$sim(n(l_i), n(l_m)) = \frac{|n(l_i) \cap n(l_m)|}{|n(l_i) \cup n(l_m)|}. \tag{5.62}$$

Users' review comments can summarize the unique features about locations, which are also very important hints for inferring potential location anchor links. Similarly, we represent users' review comments posted at locations l_i and l_m as bag-of-word vectors [19] weighted TF-IDF [35], $\mathbf{w}(l_i)$ and $\mathbf{w}(l_m)$. And the similarity between l_i and l_m based on the review comments can be represented as

$$sim(\mathbf{w}(l_i), \mathbf{w}(l_m)) = \mathbf{w}(l_i)^\top \cdot \mathbf{w}(l_i). \tag{5.63}$$

Closer locations are more likely to the same site than the ones which are far away. Based on the (latitude, longitude) information, the similarity score between locations l_i and l_m can be defined as follows:

$$\begin{aligned} &sim(lat\text{-}long(l_i), lat\text{-}long(l_m)) \\ &= 1.0 - \frac{\sqrt{(lat(l_i) - lat(l_m))^2 + (long(l_i) - long(l_m))^2}}{\sqrt{(180 - (-180))^2 + (90 - (-90))^2}}. \end{aligned} \tag{5.64}$$

Furthermore, we can also construct the similarity matrix between locations in $G^{(1)}$ and $G^{(2)}$ as $\mathbf{\Theta} \in \mathbb{R}^{|\mathcal{L}^{(1)}| \times |\mathcal{L}^{(2)}|}$, where we have entry

$$\begin{aligned} \mathbf{\Theta}(j, m) = \frac{1}{3}\Big(&sim(n(l_i), n(l_m)) + sim(\mathbf{w}(l_i), \mathbf{w}(l_m)) \\ &+ sim(lat\text{-}long(l_i), lat\text{-}long(l_m)) \Big). \end{aligned} \tag{5.65}$$

The optimal *location transitional matrix* $\mathbf{Q}$ which can minimize the cost in attribute information can be represented as

$$\text{cost in attribute}(\mathcal{A}_l^{(1,2)}) = -\left\| \mathbf{Q} \circ \mathbf{\Theta} \right\|_1. \tag{5.66}$$

Location Anchor Link Inference with Link and Attribute Information

By considering the location links and attributes attached to locations simultaneously, the cost function of inferred location anchor links $\mathcal{A}_l^{(1,2)}$ can be represented as

$$\begin{aligned} \text{cost}(\mathcal{A}_l^{(1,2)}) &= \text{cost in link}(\mathcal{A}_l^{(1,2)}) + \alpha \cdot \text{cost in attribute}(\mathcal{A}_l^{(1,2)}) \\ &= \left\| \mathbf{P}^\top \mathbf{L}^{(1)} \mathbf{Q} - \mathbf{L}^{(2)} \right\|_F^2 - \alpha \cdot \left\| \mathbf{Q} \circ \mathbf{\Theta} \right\|_1. \end{aligned} \tag{5.67}$$

The optimal user and location transitional matrices $\mathbf{P}^*$ and $\mathbf{Q}^*$ that can minimize the mapping cost will be

$$\begin{aligned}
\mathbf{P}^*, \mathbf{Q}^* &= \arg\min_{\mathbf{P},\mathbf{Q}} \text{cost}(\mathcal{A}_l^{(1,2)}) \\
&= \arg\min_{\mathbf{P},\mathbf{Q}} \left\| \mathbf{P}^\top \mathbf{L}^{(1)} \mathbf{Q} - \mathbf{L}^{(2)} \right\|_F^2 - \alpha \cdot \|\mathbf{Q} \circ \mathbf{\Theta}\|_1 , \\
s.t. \quad & \mathbf{Q} \in \{0,1\}^{|\mathcal{L}^{(1)}| \times |\mathcal{L}^{(2)}|}, \mathbf{P} \in \{0,1\}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|}, \\
& \mathbf{P}\mathbf{1}^{|\mathcal{U}^{(2)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1}, \mathbf{P}^\top \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(2)}| \times 1}. \\
& \mathbf{Q}\mathbf{1}^{|\mathcal{L}^{(2)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{L}^{(1)}| \times 1}, \mathbf{Q}^\top \mathbf{1}^{|\mathcal{L}^{(1)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{L}^{(2)}| \times 1},
\end{aligned} \tag{5.68}$$

where *location anchor links* also have *one-to-one* constraint, and the last two equations are added to maintain such a constraint.

5.5.2.3 Co-inference of Anchor Links

User transitional matrix $\mathbf{P}$ is involved in the objective functions of inferring both *user anchor links* and *location anchor links*, and these two different anchor link inference tasks are strongly correlated (due to $\mathbf{P}$) and can be inferred simultaneously. By integrating the objective equations of anchor link inference for both users and locations, the optimal transitional matrices $\mathbf{P}^*$ and $\mathbf{Q}^*$ can be obtained simultaneously by solving the following objective function:

$$\begin{aligned}
\mathbf{P}^*, \mathbf{Q}^* &= \arg\min_{\mathbf{P},\mathbf{Q}} \text{cost}(\mathcal{A}_u^{(1,2)}) + \text{cost}(\mathcal{A}_l^{(1,2)}) \\
&= \arg\min_{\mathbf{P},\mathbf{Q}} \left\| \mathbf{P}^\top \mathbf{S}^{(1)} \mathbf{P} - \mathbf{S}^{(2)} \right\|_F^2 + \left\| \mathbf{P}^\top \mathbf{L}^{(1)} \mathbf{Q} - \mathbf{L}^{(2)} \right\|_F^2 \\
&\quad - \alpha \cdot \|\mathbf{P} \circ \mathbf{\Lambda}\|_1 - \alpha \cdot \|\mathbf{Q} \circ \mathbf{\Theta}\|_1 , \\
s.t. \quad & \mathbf{P} \in \{0,1\}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|}, \mathbf{Q} \in \{0,1\}^{|\mathcal{L}^{(1)}| \times |\mathcal{L}^{(2)}|}, \\
& \mathbf{P}\mathbf{1}^{|\mathcal{U}^{(2)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1}, \mathbf{P}^\top \mathbf{1}^{|\mathcal{U}^{(1)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(2)}| \times 1}, \\
& \mathbf{Q}\mathbf{1}^{|\mathcal{L}^{(2)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{L}^{(1)}| \times 1}, \mathbf{Q}^\top \mathbf{1}^{|\mathcal{L}^{(1)}| \times 1} \preccurlyeq \mathbf{1}^{|\mathcal{L}^{(2)}| \times 1}.
\end{aligned} \tag{5.69}$$

The objective function is a constrained nonlinear integer programming problem, which is hard to address mathematically. Many relaxation algorithms have been proposed so far [2]. To solve the problem, as proposed in [48], the binary constraint of matrices $\mathbf{P}$ and $\mathbf{Q}$ can be relaxed to the real numbers in range $[0, 1]$ and entries in $\mathbf{P}$ and $\mathbf{Q}$ will denote the existence probabilities/confidence scores of the corresponding anchor links. Redundant anchor links introduced by such a relaxation will be pruned with the co-matching algorithm to be introduced in the next section.

Meanwhile, the Hadamard product terms $\mathbf{P} \circ \mathbf{\Lambda}$ and $\mathbf{Q} \circ \mathbf{\Theta}$ can be very hard to deal with when solving the optimization problem. Considering that matrices $\mathbf{P}$, $\mathbf{\Lambda}$, $\mathbf{Q}$, and $\mathbf{\Theta}$ are all positive matrices, the L_1 norm of Hadamard product terms can be replaced according to the following Lemmas.

Lemma 5.1 *For any given matrix* $\mathbf{A}$*, the square of its Frobenius norm equals to the trace of* $\mathbf{A}\mathbf{A}^\top$*, i.e.,* $\|\mathbf{A}\|_F^2 = tr(\mathbf{A}\mathbf{A}^\top)$.

The proof of the above lemma will be left as an exercise for the readers.

Lemma 5.2 *For two given positive matrices* $\mathbf{A}$ *and* $\mathbf{B}$ *of the same dimensions, the* L_1 *norm of the Hadamard product about* $\mathbf{A}$ *and* $\mathbf{B}$ *equals to the trace of* $\mathbf{A}^\top\mathbf{B}$ *or* $\mathbf{A}\mathbf{B}^\top$*, i.e.,* $\|\mathbf{A} \circ \mathbf{B}\|_1 = tr(\mathbf{A}^\top\mathbf{B}) = tr(\mathbf{A}\mathbf{B}^\top)$.

Proof According to the definitions of matrix trace, terms $tr(\mathbf{A}^\top\mathbf{B})$ and $tr(\mathbf{A}\mathbf{B}^\top)$ equal to the Frobenius product [33] of matrices $\mathbf{A}$ and $\mathbf{B}$, i.e.,

$$tr(\mathbf{A}^\top\mathbf{B}) = tr(\mathbf{A}\mathbf{B}^\top) = \sum_{i,j} \mathbf{A}(i,j)\mathbf{B}(i,j). \tag{5.70}$$

Meanwhile,

$$\|\mathbf{A} \circ \mathbf{B}\|_1 = \sum_{i,j} |(\mathbf{A} \circ \mathbf{B})(i,j)| = \sum_{i,j} |\mathbf{A}(i,j) \cdot \mathbf{B}(i,j)|\,. \tag{5.71}$$

Considering that both $\mathbf{A}$ and $\mathbf{B}$ are positive matrices, so the following equation can always hold:

$$\|\mathbf{A} \circ \mathbf{B}\|_1 = \sum_{i,j} \mathbf{A}(i,j) \cdot \mathbf{B}(i,j) = tr(\mathbf{A}^\top\mathbf{B}) = tr(\mathbf{A}\mathbf{B}^\top). \tag{5.72}$$

To solve the objective function, the alternating projected gradient descent (APGD) method introduced in [25] can be applied here and the *one-to-one* constraint will be relaxed. The constraints $\mathbf{P}\mathbf{1} \preccurlyeq \mathbf{1}, \mathbf{P}^\top\mathbf{1} \preccurlyeq \mathbf{1}$ are replaced with $\|\mathbf{P}\|_1 \leq t$ instead, where t is a small constant. Similarly, the *one-to-one* constraint on $\mathbf{Q}$ is also relaxed and replaced with $\|\mathbf{Q}\|_1 \leq t$. Furthermore, by incorporating terms $\|\mathbf{P}\|_1$ and $\|\mathbf{Q}\|_1$ into the minimization objective function, based on the relaxed constraints as well as Lemmas 5.1–5.2, the new objective function can be represented to be

$$\begin{aligned}
\arg\min_{\mathbf{P},\mathbf{Q}} \; & f(\mathbf{P},\mathbf{Q}) = tr\Big((\mathbf{P}^\top\mathbf{S}^{(1)}\mathbf{P} - \mathbf{S}^{(2)})(\mathbf{P}^\top\mathbf{S}^{(1)}\mathbf{P} - \mathbf{S}^{(2)})^\top\Big) \\
& + tr\Big((\mathbf{P}^\top\mathbf{L}^{(1)}\mathbf{Q} - \mathbf{L}^{(2)})(\mathbf{P}^\top\mathbf{L}^{(1)}\mathbf{Q} - \mathbf{L}^{(2)})^\top\Big) \\
& - \alpha \cdot tr(\mathbf{P}\mathbf{\Lambda}^\top) - \alpha \cdot tr(\mathbf{Q}\mathbf{\Theta}^\top) + \gamma \cdot \|\mathbf{P}\|_1 + \mu \cdot \|\mathbf{Q}\|_1 \\
s.t. \; & \mathbf{0}^{|\mathcal{U}^{(1)}|\times|\mathcal{U}^{(2)}|} \preccurlyeq \mathbf{P} \preccurlyeq \mathbf{1}^{|\mathcal{U}^{(1)}|\times|\mathcal{U}^{(2)}|}, \\
& \mathbf{0}^{|\mathcal{L}^{(1)}|\times|\mathcal{L}^{(2)}|} \preccurlyeq \mathbf{Q} \preccurlyeq \mathbf{1}^{|\mathcal{L}^{(1)}|\times|\mathcal{L}^{(2)}|},
\end{aligned} \tag{5.73}$$

where γ and μ denote the weights on $\|\mathbf{P}\|_1$ and $\|\mathbf{Q}\|_1$, respectively.

As we can see, the objective function is with respect to $\mathbf{P}$ and $\mathbf{Q}$ and we cannot give a closed-form solution for the objective function. In [48], the optimal $\mathbf{P}$ and $\mathbf{Q}$ are learned with an alternative updating procedure based on the gradient descent algorithm: (1) fix $\mathbf{Q}$ and minimize the objective function *w.r.t.* $\mathbf{P}$; and (2) fix $\mathbf{P}$ and minimize the objective function *w.r.t.* $\mathbf{Q}$. If during these two updating procedures, entries in $\mathbf{P}$ or $\mathbf{Q}$ become invalid, we use a projection to guarantee the [0, 1] constraint: (1) if $\mathbf{P}(i,j) > 1$ or $\mathbf{Q}(i,j) > 1$, we project it to 1; and (2) if $\mathbf{P}(i,j) < 0$ or $\mathbf{Q}(i,j) < 0$, we project it to 0 [25]. The alternative updating equations of these two matrices are available as follows:

$$\begin{aligned}\mathbf{P}^{\tau} &= \mathbf{P}^{\tau-1} - \eta_1 \cdot \frac{\partial \Gamma(\mathbf{P}^{\tau-1}, \mathbf{Q}^{\tau-1}, \gamma, \mu)}{\partial \mathbf{P}} \\ &= \mathbf{P}^{\tau-1} - 2\eta_1 \cdot \Big(\mathbf{S}^{(1)}\mathbf{P}\mathbf{P}^{\top}(\mathbf{S}^{(1)})^{\top}\mathbf{P} + (\mathbf{S}^{(1)})^{\top}\mathbf{P}\mathbf{P}^{\top}\mathbf{S}^{(1)}\mathbf{P} \\ &\quad + \mathbf{L}^{(1)}\mathbf{Q}\mathbf{Q}^{\top}(\mathbf{L}^{(1)})^{\top}\mathbf{P} - \mathbf{S}^{(1)}\mathbf{P}(\mathbf{S}^{(2)})^{\top} - (\mathbf{S}^{(1)})^{\top}\mathbf{P}\mathbf{S}^{(2)} \\ &\quad - \mathbf{L}^{(1)}\mathbf{Q}(\mathbf{L}^{(2)})^{\top} - \frac{1}{2}\alpha\mathbf{\Lambda} + \frac{1}{2}\gamma\mathbf{1}\mathbf{1}^{\top}\Big),\end{aligned} \tag{5.74}$$

$$\begin{aligned}\mathbf{Q}^{\tau} &= \mathbf{Q}^{\tau-1} - \eta_2 \cdot \frac{\partial \Gamma(\mathbf{P}^{\tau}, \mathbf{Q}^{\tau-1}, \gamma, \mu)}{\partial \mathbf{Q}} \\ &= \mathbf{Q}^{\tau-1} - 2\eta_2 \cdot \Big((\mathbf{L}^{(1)})^{\top}\mathbf{P}\mathbf{P}^{\top}\mathbf{L}^{(1)}\mathbf{Q} - (\mathbf{L}^{(1)})^{\top}\mathbf{P}\mathbf{L}^{(2)} \\ &\quad - \frac{1}{2}\alpha\mathbf{\Theta} + \frac{1}{2}\mu\mathbf{1}\mathbf{1}^{\top}\Big),\end{aligned} \tag{5.75}$$

where η_1 and η_2 are the *learning rate* in updating $\mathbf{P}$ and $\mathbf{Q}$, respectively. Such an updating process will continue until both $\mathbf{P}$ and $\mathbf{Q}$ converge.

5.5.3 Network Co-matching

To solve the objective function, the *one-to-one* constraint on both *user anchor links* and *location anchor links* are relaxed, which can take values in range [0, 1]. As a result, users and locations in each network can be connected by multiple user/location anchor links of various confidence scores across networks simultaneously and the *one-to-one* constraint can no longer hold any more. To maintain such a constraint on both user and location anchor links, the redundant ones introduced due to the relaxation with *network flow* will be pruned based on a network co-matching algorithm in this subsection.

Based on user sets $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$, location sets $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$, as well as the existence confidence scores of potential user and location anchor links between networks $G^{(1)}$ and $G^{(1)}$ (i.e., entries of $\mathbf{P}$ and $\mathbf{Q}$), we can construct the user and location preference bipartite graphs as shown in the left plots of Fig. 5.6.

User Preference Bipartite Graph

The *user preference bipartite graph* can be represented as $BG_{\mathcal{U}} = (\mathcal{U}^{(1)} \cup \mathcal{U}^{(2)}, \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}, \mathcal{W}_{\mathcal{U}})$, where $\mathcal{U}^{(1)} \cup \mathcal{U}^{(2)}$ denotes the user nodes in $G^{(1)}$ and $G^{(2)}$, $\mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$ contains all the potential user anchor links between $G^{(1)}$ and $G^{(2)}$, and $\mathcal{W}_{\mathcal{U}}$ will map links in $\mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$ to their confidence scores (i.e., entries in $\mathbf{P}$) inferred in the previous section.

Location Preference Bipartite Graph

Similarly, we can also represent the *location preference bipartite graph* to be $BG_{\mathcal{L}} = (\mathcal{L}^{(1)} \cup \mathcal{L}^{(2)}, \mathcal{L}^{(1)} \times \mathcal{L}^{(2)}, \mathcal{W}_{\mathcal{L}})$, where the weight mapping of potential *location anchor links* (i.e., $\mathcal{W}_{\mathcal{L}}$) can be obtained from *location transitional matrix* $\mathbf{Q}$ in a similar way as introduced before.

Co-matching Network Flow Graph

To prune the non-existing anchor links, the traditional network flow algorithm can be employed to match users and locations across networks $G^{(1)}$ and $G^{(2)}$ simultaneously, which are grouped together in an integrated network flow model, named "co-matching network flow." As shown in the right plot

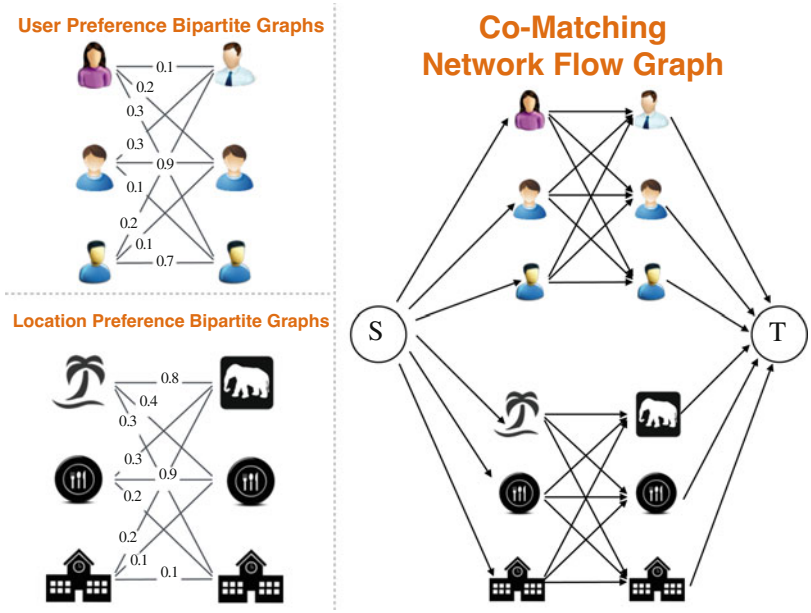

Fig. 5.6 User and location preference bipartite graphs and co-matching network flow graph

of Fig. 5.6, based on the *user preference bipartite graphs* and *location preference bipartite graphs*, the *co-matching network flow graph* can be constructed by adding (1) a source node S, (2) a sink node T, (3) links connecting node S and links in $\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)}$ (i.e., $\{S\} \times (\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)})$), and (4) links connecting nodes in $\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}$ and node T (i.e., $(\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}) \times \{T\}$).

Bound Constraint

In the network flow model, each link in the *co-matching network flow graph* is associated with an *upper bound* and *lower bound* to control the amount of flow going through it. For example, the upper and lower bounds of potential user anchor link $(u, v) \in \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$ in the *co-matching network flow graph* can be represented as

$$\underline{B}(u, v) \leq F(u, v) \leq \overline{B}(u, v), \tag{5.76}$$

where $F(u, v)$ denotes the flow amount going through link (u, v), $\underline{B}(u, v)$ and $\overline{B}(u, v)$ represent the *lower bound* and *upper bound* associated with link (u, v), respectively.

Considering that the constraint on both user and location anchor links is *one-to-one* and networks studied in this section are partially aligned, users in online social networks include both anchor and non-anchor users; so is the case for locations. In other words, each user and location in online social networks can be connected by at most one anchor links across networks, which can be achieved by adding the following upper and lower bound constraint on links $\{S\} \times (\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)})$ and $(\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}) \times \{T\}$:

$$0 \leq F(u, v) \leq 1, \forall (u, v) \in \{S\} \times (\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)}) \cup (\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}) \times \{T\}. \tag{5.77}$$

Among all the potential user anchor links in $\mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$ and location anchor links in $\mathcal{L}^{(1)} \times \mathcal{L}^{(2)}$, only part of these links will be selected finally due to the *one-to-one* constraint. To represent whether a link (u, v) is selected or not, the flow amount going through links $\mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \cup \mathcal{L}^{(1)} \times \mathcal{L}^{(2)}$ will be set

as integers with upper and lower bounds to be 0 a 1 (1 denotes the link is selected, and 0 otherwise), respectively, i.e.,

$$F(u, v) \in \{0, 1\}, \forall (u, v) \in \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \cup \mathcal{L}^{(1)} \times \mathcal{L}^{(2)}. \tag{5.78}$$

Mass Balance Constraint

In addition, in network flow model, for each node in the graph (except the source and sink node), the amount of flow going through it should meet the *mass balance constraint*, i.e., for each node in the network, the amount of network flow going into it should equal to that going out from it:

$$\sum_{w \in \mathcal{N}_F, (w,u) \in \mathcal{L}_F} F(w, u) = \sum_{v \in \mathcal{N}_F, (u,v) \in \mathcal{L}_F} F(u, v), \tag{5.79}$$

where $\mathcal{N}_F = \{S\} \cup \mathcal{U}^{(1)} \cup \mathcal{U}^{(2)} \cup \mathcal{L}^{(1)} \cup \mathcal{L}^{(2)} \cup \{T\}$ denotes all the nodes in the *co-matching network flow graph* and $\mathcal{L}_F = \{S\} \times (\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)}) \cup \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \cup \mathcal{L}^{(1)} \times \mathcal{L}^{(2)} \cup (\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}) \times \{T\}$ represents all the links in graph.

Maximum Confidence Objective Function

All the potential links connecting users and locations across networks are associated with certain costs in network flow model, where links with lower costs are more likely to be selected. The model can be modified a little to select the links introducing the maximum confidence scores instead from $\mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$ and $\mathcal{L}^{(1)} \times \mathcal{L}^{(2)}$, respectively, which can be obtained with the following objective functions:

$$\max \sum_{(u,v) \in (\mathcal{U}^{(1)} \times \mathcal{U}^{(2)})} F(u, v) \cdot \mathcal{W}_{\mathcal{U}}(u, v), \tag{5.80}$$

$$\max \sum_{(m,n) \in (\mathcal{L}^{(1)} \times \mathcal{L}^{(2)})} F(m, n) \cdot \mathcal{W}_{\mathcal{L}}(m, n). \tag{5.81}$$

The final objective equation of simultaneous *co-matching* of users and locations across networks can be represented to be

$$\begin{aligned} \max & \sum_{(u,v) \in (\mathcal{U}^{(1)} \times \mathcal{U}^{(2)})} F(u, v) \cdot \mathcal{W}_{\mathcal{U}}(u, v) + \sum_{(m,n) \in (\mathcal{L}^{(1)} \times \mathcal{L}^{(2)})} F(m, n) \cdot \mathcal{W}_{\mathcal{L}}(m, n), \\ s.t. \ & 0 \le F(u, v) \le 1, \forall (u, v) \in \{S\} \times (\mathcal{U}^{(1)} \cup \mathcal{L}^{(1)}) \cup (\mathcal{U}^{(2)} \cup \mathcal{L}^{(2)}) \times \{T\}, \\ & F(u, v) \in \{0, 1\}, \forall (u, v) \in \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \cup \mathcal{L}^{(1)} \times \mathcal{L}^{(2)}, \\ & \sum_{w \in \mathcal{N}_F, (w,u) \in \mathcal{L}_F} F(w, u) = \sum_{v \in \mathcal{N}_F, (u,v) \in \mathcal{L}_F} F(u, v). \end{aligned} \tag{5.82}$$

The above network flow objective function can be solved with open-source toolkits (e.g., Scipy.Optimization[10] and GLPK[11]). In the obtained solution, the flow amount variable of potential

[10] http://docs.scipy.org/doc/scipy/reference/optimize.html.

[11] http://www.gnu.org/software/glpk/.

user and location anchor links achieving value 1 are the selected ones which will be assigned with label $+1$, while the remaining (i.e., those achieving value 0) are not selected which are assigned with label -1. The matching results obtained from the above objective function will be outputted as the final network co-alignment result.

5.6 Summary

In this chapter, we introduced the network alignment problem based on the unsupervised learning setting, where no training data (i.e., labeled anchor links) is available or necessarily needed. Technically speaking, the unsupervised network alignment problem is very challenging to address, which can be identically modeled as the graph isomorphism problem. To resolve the problem, we talked about several heuristics based network alignment approaches and several matrix inference approaches at first.

Based on the users' names and profile information, we provided a detailed description about how to utilize such information to compute the similarity scores among users. Based on the assumption that similar users are more likely to be the same user, several different similarity metrics have been introduced in this chapter. With the user names, we can compute the similarity scores among users based on the characters, tokens, and phonetic representations. Meanwhile, with the profile information, we can compute the similarity scores of users in their hometown locations, birthday, and textual information.

The anchor links actually define a mapping of users across networks, and we also introduce several network alignment approaches via inferring the mapping matrix about anchor links. Via the anchor links, both the user nodes and the social connections can be mapped from one network to the other network. The good mappings should be able to minimize the projection inconsistency about the network structures. Furthermore, there also exists a hard binary and one-to-one cardinality constraint on the mapping matrix, which renders the inference process to be extremely challenging.

We introduced an approach to apply the matrix inference based network alignment method to infer the anchor links across multiple (more than two) networks, where the alignment results should also preserve the transitivity law. To resolve the objective function, we talked about a two-phase solution: matrix inference via constraint relaxation, and post-processing of the alignment results via transitive matching.

For the alignment of networks via multiple types of shared information entities simultaneously, we also introduced a network co-alignment approach, which learns the mapping matrices of multiple types of anchor links simultaneously. By extending the anchor link mapping matrix inference approach to the scenario with both network structure and diverse attribute information, the introduced method is able to infer the user anchor links and location anchor links across heterogeneous networks at the same time.

5.7 Bibliography Notes

Graph isomorphism problem is an extremely challenging research problem, which is one of few standard problems in computational complexity theory belonging to NP. But by this context so far, it is still not known whether it belongs to P or NP-complete. It is one of only two problems whose

complexity remains unresolved as listed in [15], the other being integer factorization. There have been several research works proposing efficient algorithms to address the graph isomorphism problem in the past century [12, 30]. In 2015, László Babai claimed to have proven that the graph isomorphism problem is solvable in quasi-polynomial time [4], but the proof has not been vetted yet.

Entity resolution is a common problem in many areas, e.g., database, statistics, and artificial intelligence. Based on the entity names, [11] provides an introduction about several string distance metrics for name-matching tasks, including the edit distance like metrics, token based distance metrics, and hybrid distance metrics. Meanwhile, for a comprehensive survey about the text similarity approaches, the readers may refer to [18], which covers the similarity metrics between words, sentences, paragraphs, and documents.

The multiple network simultaneously alignment method is based on [47], where the transitivity law property on anchor links in alignment was initially pointed out in that work. Meanwhile, the heterogeneous network co-alignment approach via multiple types of shared information entities was introduced in [48]. The network matching procedure in [47, 48] are both formulated as the maximum network flow problem [20]. Over the years, many improved solutions to the problem have been discovered, e.g., the shortest augmenting path algorithm [14], the blocking flow algorithm [13], the push-relabel algorithm [17], and the binary blocking flow algorithm [16].

5.8 Exercises

1. (Easy) Please compute the *edit distance*, *Jaro distance*, and *Jaro-Winkler distance* of a pair of input strings "DIXON" and "DICKSONX."
2. (Easy) Please compute the *common token*, *token based Jaccard's coefficient*, and *token based Cosine Similarity* for the user names "Mike Jordan" and "Michael Jordan."
3. (Medium) Please identify the alignment results of the input network shown in Fig. 5.7 with the *relative degree difference* measure.

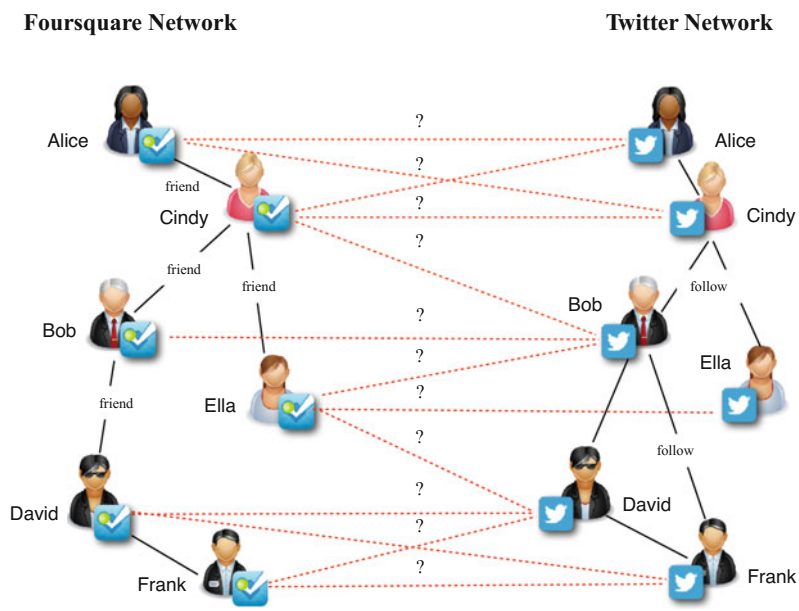

Fig. 5.7 A pair of aligned input networks

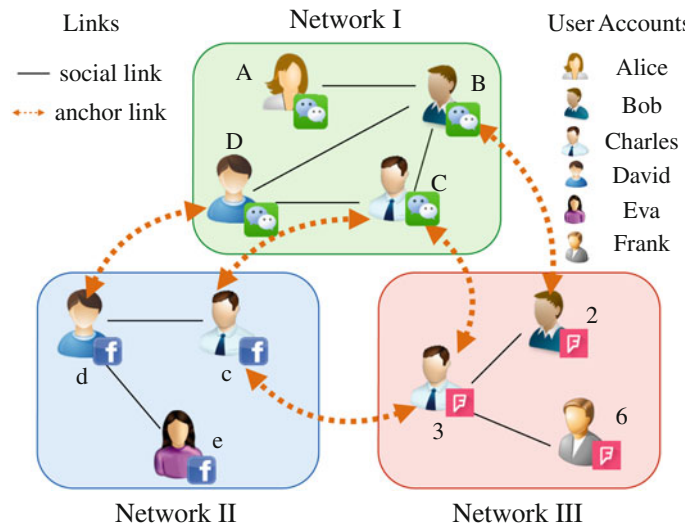

Fig. 5.8 Multiple aligned input networks

4. (Medium) Please identify the alignment results of the input network shown in Fig. 5.7 with the IsoRank algorithm.
5. (Medium) Please identify the alignment results of the input network shown in Fig. 5.8 with the IsoRankN algorithm.
6. (Medium) Please try to prove the Lemma 5.1.
7. (Hard) Please try to implement an algorithm with your preferred programming language to compute the *edit distance* of two input strings (Hint: You may consider to use the dynamic programming algorithm).
8. (Hard) Please try to implement the matrix inference based network alignment algorithm introduced in Sect. 5.3.4, and test it based on a small-sized synthetic aligned homogeneous network dataset.
9. (Hard) Please implement the UMA algorithm in your preferred programming language, and test it based on a small-sized synthetic aligned network dataset.
10. (Hard) Please implement the *network co-alignment* algorithm introduced in Sect. 5.5.2.2 your preferred programming language, and test it with a small-sized synthetic co-aligned network dataset.

References

1. L. Adamic, R. Lukose, A. Puniyani, B. Huberman, Search in power-law networks. Phys. Rev. E **64**, 046135 (2001). cs.NI/0103016
2. Y. Aflaloa, A. Bronsteinb, R. Kimmel, On convex relaxation of graph isomorphism. Proc. Natl. Acad. Sci. U S A **112**(10), 2942–2947 (2015)
3. M. Avriel, *Nonlinear Programming: Analysis and Methods* (Prentice-Hall, Englewood Cliffs, 1976)
4. L. Babai, Graph isomorphism in quasipolynomial time. CoRR, abs/1512.03547 (2015)
5. R. Baeza-Yates, B. Ribeiro-Neto, *Modern Information Retrieval* (Addison-Wesley Longman Publishing Co., Inc., Boston, 1999)
6. C. Bettstetter, On the minimum node degree and connectivity of a wireless multihop network, in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '02)* (ACM, New York, 2002)
7. S. Borgatti, M. Everett, A graph-theoretic perspective on centrality. Soc. Net. **28**(4), 466–484 (2006)

8. W. Cavnar, J. Trenkle, N-gram-based text categorization, in *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (1994)
9. D. Chandler, The norm of the Schur product operation. Numer. Math. **4**(1), 343–344 (1962)
10. M. Charikar, Similarity estimation techniques from rounding algorithms, in *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC '02)* (ACM, New York, 2002)
11. W. Cohen, P. Ravikumar, S. Fienberg, A comparison of string distance metrics for name-matching tasks, in *Proceedings of the 2003 International Conference on Information Integration on the Web (IIWEB'03)* (AAAI Press, Palo Alto, 2003)
12. D. Corneil, C. Gotlieb, An efficient algorithm for graph isomorphism. J. ACM **17**(1), 51–64 (1970)
13. E. Dinic, Algorithm for solution of a problem of maximum flow in a network with power estimation. Sov. Math. Dokl. **11**, 1277–1280 (1970)
14. J. Edmonds, R. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM **19**(2), 248–264 (1972)
15. M. Garey, D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, 1990)
16. A.A. Goldberg, S. Rao, Beyond the flow decomposition barrier. J. ACM **45**(5), 783–797 (1998)
17. A. Goldberg, R. Tarjan, A new approach to the maximum-flow problem. J. ACM **35**(4), 921–940 (1988)
18. W. Gomaa, A. Fahmy, Article: a survey of text similarity approaches. Int. J. Comput. Appl. **68**(3), 13–18 (2013)
19. Z. Harris, Distributional structure. Word **10**(23), 146–162 (1954)
20. T. Harris, F. Ross, *Fundamentals of a Method for Evaluating Rail Net Capacities*. Research Memorandum (The RAND Corporation, Santa Monica, 1955)
21. D. Hirschberg, Algorithms for the longest common subsequence problem. J. ACM **24**(4), 664–675 (1977)
22. M. Jaro, Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J. Am. Stat. Assoc. **84**(406), 414–420 (1989)
23. T. Joachims, A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)* (Morgan Kaufmann Publishers Inc., San Francisco, 1997)
24. X. Kong, J. Zhang, P. Yu, Inferring anchor links across multiple heterogeneous social networks, in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)* (ACM, New York, 2013)
25. D. Koutra, H. Tong, D. Lubensky, Big-align: fast bipartite graph alignment, in *2013 IEEE 13th International Conference on Data Mining* (IEEE, Piscataway, 2013)
26. K. Kunen, *Set Theory* (Elsevier Science Publishers, Amsterdam, 1980)
27. J. Lee, W. Han, R. Kasperovics, J. Lee, An in-depth comparison of subgraph isomorphism algorithms in graph databases, in *Proceedings of the VLDB Endowment*. VLDB Endowment (2012)
28. V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals. Sov. Phy. Dok. **10**, 707 (1966)
29. C. Liao, K. Lu, M. Baym, R. Singh, B. Berger. IsoRankN: spectral methods for global alignment of multiple protein networks. Bioinformatics **25**(12), i253–i258 (2009)
30. B. McKay, Practical graph isomorphism. Congr. Numer. **30**, 45–87 (1981)
31. G. Navarro, A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
32. M. Newman, Analysis of weighted networks. Phy. Rev. E **70**, 056131 (2004)
33. K. Petersen, M. Pedersen, *The Matrix Cookbook*. Technical report. Technical University of Denmark, Lyngby (2012)
34. E. Ristad, P. Yianilos, Learning string-edit distance. IEEE Trans. Pattern Anal. Mach. Intell. **20**(5), 522–532 (1998)
35. G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)
36. G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
37. T. Shi, S. Kasahara, T. Pongkittiphan, N. Minematsu, D. Saito, K. Hirose, A measure of phonetic similarity to quantify pronunciation variation by using ASR technology, in *18th International Congress of Phonetic Sciences (ICPhS 2015)* (University of Glasgow, Glasgow, 2015)
38. R. Singh, J. Xu, B. Berger, Global alignment of multiple protein interaction networks with application to functional orthology detection. Natl. Acad. Sci. **105**(35), 12763–12768 (2008)
39. The National Archives, The Soundex indexing system (2007)
40. S. Umeyama, An eigendecomposition approach to weighted graph matching problems. IEEE Trans. Pattern Anal. Mach. Intell. **10**(5), 695–703 (1988)
41. R. Wagner, M. Fischer, The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974)
42. W. Winkler, String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage, in *Proceedings of the Section on Survey Research* (1990)

43. D. Wipf, B. Rao, L0-norm minimization for basis selection, in *Proceedings of the 17th International Conference on Neural Information Processing Systems (NIPS'04)* (MIT Press, Cambridge, 2005)
44. M. Yu, G. Li, D. Deng, J. Feng, String similarity search and join: a survey. Front. Comput. Sci. **10**(3), 399–417 (2016)
45. R. Zafarani, H. Liu, Connecting users across social media sites: a behavioral-modeling approach, in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)* (ACM, New York, 2013)
46. J. Zhang, P. Yu, MCD: mutual clustering across multiple social networks, in *2015 IEEE International Congress on Big Data* (IEEE, Piscataway, 2015)
47. J. Zhang, P. Yu, Multiple anonymized social networks alignment, in *2015 IEEE International Conference on Data Mining* (IEEE, Piscataway, 2015)
48. J. Zhang, P. Yu, PCT: partial co-alignment of social networks, in *Proceedings of the 25th International Conference on World Wide Web (WWW '16)* (International World Wide Web Conferences Steering Committee Republic and Canton of Geneva, Geneva, 2016)
49. J. Zhang, P. Yu, Z. Zhou, Meta-path based multi-network collective link prediction, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)* (ACM, New York, 2014)