IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 12 May 2023, accepted 10 June 2023, date of publication 20 June 2023, date of current version 12 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3287833

RESEARCH ARTICLE

Matrix Tri-Factorization Over the Tropical Semiring

AMRA OMANOVIĆ, POLONA OBLAK, AND TOMAŽ CURK
Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia
Corresponding author: Tomaž Curk (tomaz.curk@fri.uni-lj.si)

The work of Amra Omanović was supported by the Slovene Research Agency, Young Researcher under Grant 52096. The work of Polona Oblak was supported by the Research Program under Grant P1-0222. The work of Tomaž Curk was supported by the Research Program under Grant P2-0209.

ABSTRACT Tropical semiring has proven successful in several research areas, including optimal control, bioinformatics, discrete event systems, and decision problems. Previous studies have applied a matrix two-factorization algorithm based on the tropical semiring to investigate bipartite and tripartite networks. Tri-factorization algorithms based on standard linear algebra are used to solve tasks such as data fusion, co-clustering, matrix completion, community detection, and more. However, there is currently no tropical matrix tri-factorization approach that would allow for the analysis of multipartite networks with many parts. To address this, we propose the `triFastSTMF` algorithm, which performs tri-factorization over the tropical semiring. We applied it to analyze a four-partition network structure and recover the edge lengths of the network. We show that `triFastSTMF` performs similarly to `Fast-NMTF` in terms of approximation and prediction performance when fitted on the whole network. When trained on a specific subnetwork and used to predict the entire network, `triFastSTMF` outperforms `Fast-NMTF` by several orders of magnitude smaller error. The robustness of `triFastSTMF` is due to tropical operations, which are less prone to predict large values compared to standard operations.

INDEX TERMS Tropical semiring, tri-factorization, network structure analysis, four-partition network.

I. INTRODUCTION

Matrix factorization methods embed data into a latent space using a two-factorization or tri-factorization approach, depending on the number of low-dimensional factor matrices required for the specific task. Matrix factorization methods can help solve problems in recommender systems [1], pattern recognition [2], data fusion [3], network structure analysis [4], and similar. In many of these scenarios, two-factorization achieves state-of-the-art results. However, there are cases where tri-factorization outperforms two-factorization, such as in intermediate data fusion [3], where tri-factorization is used to fuse multiple data sources to improve the predictive power of the model.

Matrix factorization methods employ different types of operations to compute the factor matrices [5], [6], [7]. Most matrix factorization methods are based on standard linear algebra, such as non-negative matrix factorization [8] (`NMF`), binary matrix factorization [9] (`BMF`), probabilistic NMF [10] (`PMF`), while some novel approaches such as `STMF` [11] and `FastSTMF` [12] are based on the tropical semiring.

The $(\max, +)$ *semiring* or *tropical semiring* $\mathbb{R}_{\max}$ is the set $\mathbb{R} \cup \{-\infty\}$, equipped with max as addition ($\oplus$), and $+$ as multiplication ($\otimes$). For example, $2 \oplus 3 = 3$ and $1 \otimes 1 = 2$. Throughout the paper, the symbols "$+$" and "$-$" refer to standard operations of addition and subtraction. The renowned `NMF` method [8] is based on the element-wise sum, which results in the "parts-of-whole" interpretation of factor matrices. On the contrary, tropical or $(\max, +)$ factorization uses the maximum operator, which results in a "winner-takes-it-all" interpretation [13]. Matrix factorization approaches using tropical semiring demonstrated their robustness against overfitting and achieved predictive performance comparable to techniques that use standard linear algebra. Moreover, they also reveal different

The associate editor coordinating the review of this manuscript and approving it for publication was Yilun Shang.

patterns, as we have demonstrated in our previous studies [11], [12].

Tropical semirings have various applications in network structure analysis and other research areas [14], [15], [16]. Multiplication and addition of a similar $(\min, +)$ semiring enable mapping local edge information to global information on the *shortest paths*, while the $(\max, +)$ semiring describes the *longest path* problem. In our work, we are interested in an inverse problem that infers information about edges from potentially noisy or incomplete information [4]. To the best of our knowledge, there is no matrix tri-factorization method based on the tropical semiring. Thus, we propose the first tropical tri-factorization method, called `triFastSTMF`, which introduces a third factor matrix. The proposed `triFastSTMF` can be used for various tasks that involve a *single* data source. Our GitHub repository https://github.com/Ejmric/triFastSTMF provides the source code and data required to replicate our experiments. We demonstrate the applicability of `triFastSTMF` in edge approximation and prediction in a four-partition network. Moreover, this work sets the foundation for future research aimed at creating a tropical data fusion model capable of combining *multiple* data sources.

The paper is divided into the following sections. Section II describes the related methodology, while Section III introduces the proposed approach. In Section IV, we present the experimental evaluation. We conclude the work and discuss future opportunities in Section V.

II. RELATED WORK

Matrix factorization (MF) is one of the most popular methods for data embedding, which enables the discovery of interesting feature patterns by clustering and gaining additional knowledge from the resulting factor matrices. A well-known matrix two-factorization approach is non-negative matrix factorization (`NMF`), which imposes non-negativity on both the input and output factor matrices for a more straightforward interpretation of the results. The tri-factorization based `NMF` called `NMTF` is used to extract patterns from relational data [17], and is applied in various research areas from modeling topics in text data [18] to discovering disease-disease associations [19]. `Fast-NMTF` [20] is a version of `NMTF` that uses faster training algorithms based on projected gradients, coordinate descent, and alternating least squares optimization. One of the usual applications of `NMTF` is in data fusion methods. `DFMF` [3] is a variant of penalized matrix tri-factorization for data fusion, which simultaneously factorizes data matrices in standard linear algebra to reveal hidden associations.

In the field of tropical matrix factorization, De Schutter & De Moor in 1997 [21] presented a heuristic algorithm `TMF` to compute factorization of a matrix over the tropical semiring. The `STMF` method [11] is based on `TMF`, but it can perform matrix completion over the tropical semiring. With `STMF`, we have shown that tropical operations can discover patterns that cannot be revealed with standard linear algebra. `FastSTMF` [12] is an efficient version of `STMF`, where we introduce a faster way of updating factor matrices. The main advantage of `FastSTMF` over `STMF` is better computational performance since it achieves better results with less computation. Both `STMF` and `FastSTMF` showed the ability to outperform `NMF` in achieving higher distance correlation and smaller prediction error. However, `NMF` still achieves better results in terms of approximation error on the train set.

We can also use matrix factorization to solve different network optimization problems. The Floyd-Warshall algorithm [22] for shortest paths can be formulated as a computation over a $(\min, +)$ semiring. Hook [4], in his work of linear regression over the tropical semiring, showed how a $(\min, +)$ semiring can be used for the low-rank matrix approximation to analyze the structure of a network. The basis of this approach is a two-factorization algorithm that can recover the edge lengths of the shortest path distances for tripartite and bipartite networks. Network partitioning can be performed using the algorithm for community detection called the Louvain method [23]. Another interesting application of semirings is the fact that we can write the Viterbi algorithm [24] compactly in a $(\min, +)$ semiring over probabilities [25].

Currently, no method returns three factorized matrices computed over the tropical semiring. In our work, we propose a first tri-factorization algorithm over the tropical semiring called `triFastSTMF`, which is based on `FastSTMF`. To evaluate it empirically, we apply our `triFastSTMF` to approximate and predict the edge lengths of a four-partition network.

III. METHODS

A. OUR CONTRIBUTION

1) SEMIRINGS $(\max, +)$ AND $(\min, +)$

In a matrix semiring, the operations on the matrices are based on the main operations in the underlying semiring. We denote by $\mathbb{R}_{\max}^{t \times s}$ the set of all matrices with t rows and s columns over $\mathbb{R}_{\max}$ and for a matrix $X \in \mathbb{R}_{\max}^{t \times s}$ we denote its element in the ith row and the jth column by X_{ij}. Moreover, $\mathbb{R}_{\max}^{t} = \mathbb{R}_{\max}^{t \times 1}$ is the set of all vectors with t components over $\mathbb{R}_{\max}$. We define the matrix addition over $\mathbb{R}_{\max}$ as

$$(A \oplus B)_{ij} = A_{ij} \oplus B_{ij} = \max\{A_{ij}, B_{ij}\},$$

for all $A, B \in \mathbb{R}_{\max}^{m \times n}$, $i = 1, \ldots, m$ and $j = 1, \ldots, n$, and the matrix multiplication as

$$(A \otimes B)_{ij} = \bigoplus_{k=1}^{p} A_{ik} \otimes B_{kj} = \max_{1 \le k \le p} \{A_{ik} + B_{kj}\},$$

for $A \in \mathbb{R}_{\max}^{m \times p}$ and $B \in \mathbb{R}_{\max}^{p \times n}$. Similarly, in the $(\min, +)$ semiring, the matrix addition is defined as

$$(A \oplus^{*} B)_{ij} = A_{ij} \oplus^{*} B_{ij} = \min\{A_{ij}, B_{ij}\}$$

for all $A, B \in \mathbb{R}_{\min}^{m\times n}$, $i = 1, \dots, m$ and $j = 1, \dots, n$, and the matrix multiplication is defined as

$$(A \otimes^* B)_{ij} = \bigoplus_{k=1}^{p} A_{ik} \otimes^* B_{kj} = \min_{1\le k\le p} \{A_{ik} + B_{kj}\},$$

for $A \in \mathbb{R}_{\min}^{m\times p}$ and $B \in \mathbb{R}_{\min}^{p\times n}$ for $i = 1, \dots, m$ and $j = 1, \dots, n$.

We say that matrix A is less than or equal to matrix B, denoted as $A \preceq B$, if every element in A is less than or equal to its corresponding element in B. For given matrices $A \in \mathbb{R}_{\max}^{m\times n}$ and $B \in \mathbb{R}_{\max}^{m\times p}$, the solutions of matrix equation

$$A \otimes X = B \tag{1}$$

do not need to exist. However, there might exist some matrices $X' \in \mathbb{R}_{\max}^{n\times p}$, such that $A \otimes X' \preceq B$. Such X' is called a *subsolution* of the equation (1). The *greatest subsolution* of (1) is a matrix $X_0 \in \mathbb{R}_{\max}^{n\times p}$, such that $A \otimes X_0 \preceq B$ and for any matrix X', satisfying $A \otimes X' \preceq B$ we have $X' \preceq X_0$.

It is well known (see, *e.g.* [26]) that for $A \in \mathbb{R}_{\max}^{m\times n}$ and $b = [b_1\, b_2 \dots b_m]^T \in \mathbb{R}_{\max}^{m}$, the greatest subsolution $x = [x_1\, x_2 \dots x_n]^T \in \mathbb{R}_{\max}^{n}$ of

$$A \otimes x = b$$

exists and is given by

$$x_k = -\max_{1\le \ell\le m} \{-b_\ell + A_{\ell k}\} = \min_{1\le \ell\le m} \{-A_{k\ell}^T + b_\ell\},$$

for $k = 1, \dots, n$, or equivalently

$$x = -A^T \otimes^* b.$$

More generally, for matrix equations, the greatest subsolution is given by the following theorem.

Theorem 1 (Described by Gaubert and Plus [26]): For any $A \in \mathbb{R}_{\max}^{m\times n}$ and $B \in \mathbb{R}_{\max}^{m\times p}$ the greatest subsolution of the equation $A \otimes X = B$ is

$$X = (-A)^T \otimes^* B.$$

In what follows, we need to include both operators $\otimes$ and $\otimes^*$ in our computations. First, we prove the following technical lemma.

Lemma 1: For any $A \in \mathbb{R}_{\max}^{m\times n}$, $B \in \mathbb{R}_{\max}^{n\times p}$ and $C \in \mathbb{R}_{\max}^{p\times q}$ we have

$$(A \otimes B) \otimes^* C = A \otimes (B \otimes^* C)$$

and

$$(A \otimes^* B) \otimes C = A \otimes^* (B \otimes C).$$

Proof: For any $k \in \{1, 2, \dots, m\}$ and $\ell \in \{1, 2, \dots, q\}$ we have

$$\begin{aligned}((A \otimes B) \otimes^* C)_{k\ell} &= \min_{1\le i\le p} \{(A \otimes B)_{ki} + C_{i\ell}\} = \\ &= \min_{1\le i\le p} \max_{1\le j\le n} \{A_{kj} + B_{ji} + C_{i\ell}\} = \\ &= \max_{1\le j\le n} \min_{1\le i\le p} \{A_{kj} + B_{ji} + C_{i\ell}\} = \\ &= \max_{1\le j\le n} \{A_{kj} + (B \otimes^* C)_{j\ell}\} = \\ &= (A \otimes (B \otimes^* C))_{k\ell},\end{aligned}$$

which proves the first equality. Similarly, we prove the second one. □

To implement a tropical matrix tri-factorization algorithm, we need to know how to solve tropical linear systems. In particular, we need to find the greatest subsolution of the linear system $A \otimes X \otimes B = C$.

Theorem 2: For any $A \in \mathbb{R}_{\max}^{m\times n}$, $B \in \mathbb{R}_{\max}^{p\times q}$ and $C \in \mathbb{R}_{\max}^{m\times q}$ the $n \times p$ matrix

$$X = (-A)^T \otimes^* C \otimes^* (-B)^T$$

is the greatest subsolution of the equation

$$A \otimes X \otimes B = C. \tag{2}$$

Proof: Observing the equation $A \otimes Y = C$, its greatest subsolution is by Theorem 1 equal to $Y' = (-A)^T \otimes^* C$, implying

$$A \otimes ((-A)^T \otimes^* C) = A \otimes Y' \preceq C. \tag{3}$$

Moreover, if any matrix Y'' satisfies the inequality $A \otimes Y'' \preceq C$, this implies that $Y'' \preceq (-A)^T \otimes^* C$. Similarly, the greatest subsolution of the equality $Z \otimes B = C$ is by Theorem 1 equal to $Z' = C \otimes^* (-B)^T$, thus

$$(C \otimes^* (-B)^T) \otimes B = Z' \otimes B \preceq C, \tag{4}$$

and if any matrix Z'' satisfies the inequality $Z'' \otimes B \preceq C$, this implies that $Z'' \preceq C \otimes^* (-B)^T$.

Define $X_0 = (-A)^T \otimes^* C \otimes^* (-B)^T$. Using equations (3), (4) and Lemma 1 observe that

$$\begin{aligned}A \otimes X_0 \otimes B &= A \otimes ((-A)^T \otimes^* C \otimes^* (-B)^T) \otimes B \\ &= (A \otimes (-A)^T \otimes^* C) \otimes^* (-B)^T \otimes B \\ &\preceq C \otimes^* (-B)^T \otimes B \preceq C,\end{aligned}$$

which implies that $X_0 = (-A)^T \otimes^* C \otimes^* (-B)^T$ is the subsolution of equation (2).

Assume now there exists a subsolution X' of (1), *i.e.*,

$$A \otimes X' \otimes B \preceq C.$$

Let us prove that $X' \preceq X_0$, which will imply that X_0 is the greatest subsolution of equation (1). Since $X' \otimes B$ is the subsolution of the equation $A \otimes Y = C$, it follows that $X' \otimes B \preceq (-A)^T \otimes^* C$. This implies X' is the subsolution of the equation $Z \otimes B = (-A)^T \otimes C$, which assures that

$$X' \preceq (-A)^T \otimes^* C \otimes^* (-B)^T = X_0.$$

□

2) TRI-FACTORIZATION OVER THE TROPICAL SEMIRING

We propose a tri-factorization algorithm `triFastSTMF` over the tropical semiring, which returns three factorized matrices that we later use for the analysis of the structure of four-partition networks.

Matrix tri-factorization over a tropical semiring is a decomposition of a form $R = G_1 \otimes S \otimes G_2$, where $R \in \mathbb{R}_{\max}^{m\times n}$, $G_1 \in \mathbb{R}_{\max}^{m\times r_1}$, $S \in \mathbb{R}_{\max}^{r_1\times r_2}$, $G_2 \in \mathbb{R}_{\max}^{r_2\times n}$, $r_1 \in \mathbb{N}_0$ and $r_2 \in \mathbb{N}_0$. Since for small values of r_1 and r_2 such

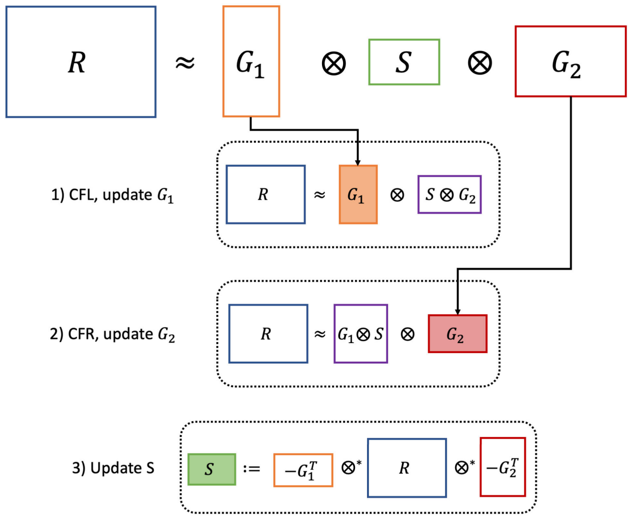


FIGURE 1. Schematic diagram of one iteration of the proposed `triFastSTMF` method for updating factor matrices G_1, S and G_2 of the data matrix $R \approx G_1 \otimes S \otimes G_2$. Step 1) updates the factor matrix G_1 through `CFL`, while step 2) uses the new G_1 to update G_2 through `CFR`. The last step, 3) updates S using Theorem 2 and newly-computed factor matrices G_1 and G_2. The procedure repeats until convergence.

decomposition may not exist, we define the tropical matrix tri-factorization problem as: Given a matrix R and factorization ranks r_1 and r_2, find matrices G_1, S and G_2 such that

$$R \cong G_1 \otimes S \otimes G_2. \tag{5}$$

Theoretically, the lower bound of ranks for which tri-factorization exist is 1. The approximation quality depends on the presence of latent structure in the data [3].

Because the solution of equation (5) does not exist in general, we will evaluate the computed tri-factorization by *b-norm* following the results from [11], [12], and [21], defined as $\|W\|_b = \sum_{i,j} |W_{ij}|$. In particular, we want to minimize the cost function

$$J(G; S) = \|R - G_1 \otimes S \otimes G_2\|_b \,.$$

In Algorithm 1, we present the pseudocode of the algorithm `triFastSTMF` illustrated in Figure 1. The convergence of the proposed algorithm `triFastSTMF`, defined in Algorithm 1, is checked similarly to that of `STMF` [11] and `FastSTMF` [12]. The factor matrices are updated only if the b-norm decreases, ensuring that the approximation error is monotonically reduced.

The `triFastSTMF` method consists of the following steps:

1) We follow the results obtained in [12] to preprocess a data matrix into a suitable shape using transformations, like matrix transposition and random permutation of rows. Wide matrices are shown to achieve smaller errors compared to tall matrices [12].
2) The default initialization of factor matrices G_1, S and G_2 uses the Random Acol strategy [11], which computes the element-wise average of randomly selected columns from matrix R. Fixed initialization for matrices G_1, S, and G_2 can be used straight from the data, see Section IV-B.
3) Until converged, each iteration of the algorithm first updates G_1 and G_2 using `CFL` and `CFR`, presented in Algorithms 2 and 3, respectively, and described below. Then we compute the middle factor S as the greatest subsolution of equation $G_1 \otimes S \otimes G_2 = R$ by Theorem 2 as
$$S = (-G_1)^T \otimes^* R \otimes^* (-G_2)^T.$$
4) As the last step of `triFastSTMF`, we reshape the factor matrices G_1, S, and G_2 into appropriate forms depending on the initial transformation of the data matrix R. If some of the elements of the data matrix R are not given, we apply the operations proposed in [11] to skip all the missing values in the calculation.

Algorithm 1 Tri-Factorization Over the Tropical Semiring (`triFastSTMF`)

Input: data matrix $R \in \mathbb{R}_{\max}^{m \times n}$, approximation ranks r_1, r_2
Output: factorization $G_1 \in \mathbb{R}_{\max}^{m \times r_1}$, $S \in \mathbb{R}_{\max}^{r_1 \times r_2}$, $G_2 \in \mathbb{R}_{\max}^{r_2 \times n}$

if R not wide **then** transpose R
$perm \leftarrow$ random permutation of indices $1 \ldots m$
$R \leftarrow R[perm, :]$
initialize G_1, G_2
$S \leftarrow (-G_1)^T \otimes^* R \otimes^* (-G_2)^T$
while not converged **do**
  $G_1 \leftarrow CFL(R, G_1, S, G_2)$
  $G_2 \leftarrow CFR(R, G_1, S, G_2)$
  $S \leftarrow (-G_1)^T \otimes^* R \otimes^* (-G_2)^T$
end while
if R transposed **then**
$(G_1, S, G_2) \leftarrow (G_2^T, S^T, G_1[perm^{-1}, :]^T)$
else $(G_1, S, G_2) \leftarrow (G_1[perm^{-1}, :], S, G_2)$
return G_1, S, G_2

Note that `triFastSTMF` updates one factor matrix at a time using `CFL` and `CFR`, presented in Algorithms 2 and 3, respectively. They are both based on `FastSTMF` and represent the two-factorization with `FastSTMF` core [12] that contains minor changes:

- In `CFL`/`CFR`, we remove the initialization of the factor matrices, as they are already initialized at the beginning of `triFastSTMF`. In `CFL`, we update only the left factor matrix G_1, and declare $Q = S \otimes G_2$ to be the second factor matrix. Similarly, in `CFR`, we update only the right factor matrix G_2 and $Q = G_1 \otimes S$ is the first factor matrix. This approach prevents overfitting factor matrices since the optimization iterates over the left and right factorization. Such a process gives equal importance to both factor matrices, allowing patterns to spread in multiple factor matrices instead of being consolidated in one of them.

Algorithm 2 Compute Factorization to Update the Left Factor Matrix G_1 (`CFL`)

Input: data matrix $R \in \mathbb{R}_{\max}^{m\times n}$, factor matrices: left $G_1 \in \mathbb{R}_{\max}^{m\times r_1}$, middle $S \in \mathbb{R}_{\max}^{r_1\times r_2}$, and right $G_2 \in \mathbb{R}_{\max}^{r_2\times n}$
Output: left factor matrix $G_1 \in \mathbb{R}_{\max}^{m\times r_1}$
$Q = S \otimes G_2$
while not converged **do**
  for each *row* i of R
    err, row_inds, $col\_inds \leftarrow$ TD_A(R, G_1, Q, i)
    for each j **in** argsort(err) in decreasing order
      $k \leftarrow \text{argmax}_\ell\,(count_\ell\,(row\_inds \cup col\_inds[j]))$
      $(G_1, Q, G'_{1(\cdot k)}, Q'_{k\cdot}) \leftarrow$ F-ULF(R, G_1, Q, i, j, k)
      if $\|R - G_1 \otimes S \otimes G_2\|_b$ decreases **then break**
      else $(G_{1(\cdot k)}, Q_{k\cdot}) \leftarrow (G'_{1(\cdot k)}, Q'_{k\cdot})$
      $(G_1, Q, G'_{1(\cdot k)}, Q'_{k\cdot}) \leftarrow$ F-URF(R, G_1, Q, i, j, k)
      if $\|R - G_1 \otimes S \otimes G_2\|_b$ decreases **then break**
      else $(G_{1(\cdot k)}, Q_{k\cdot}) \leftarrow (G'_{1(\cdot k)}, Q'_{k\cdot})$
end while
return G_1

Algorithm 3 Compute Factorization to Update the Right Factor Matrix G_2 (`CFR`)

Input: data matrix $R \in \mathbb{R}_{\max}^{m\times n}$, factor matrices: left $G_1 \in \mathbb{R}_{\max}^{m\times r_1}$, middle $S \in \mathbb{R}_{\max}^{r_1\times r_2}$, and right $G_2 \in \mathbb{R}_{\max}^{r_2\times n}$
Output: right factor matrix $G_2 \in \mathbb{R}_{\max}^{r_2\times n}$
$Q = G_1 \otimes S$
while not converged **do**
  for each *row* i of R
    err, row_inds, $col\_inds \leftarrow$ TD_A(R, Q, G_2, i)
    for each j **in** argsort(err) in decreasing order
      $k \leftarrow \text{argmax}_\ell\,(count_\ell\,(row\_inds \cup col\_inds[j]))$
      $(Q, G_2, Q'_{\cdot k}, G'_{2(k\cdot)}) \leftarrow$ F-ULF(R, Q, G_2, i, j, k)
      if $\|R - G_1 \otimes S \otimes G_2\|_b$ decreases **then break**
      else $(Q_{\cdot k}, G_{2(k\cdot)}) \leftarrow (Q'_{\cdot k}, G'_{2(k\cdot)})$
      $(Q, G_2, Q'_{\cdot k}, G'_{2(k\cdot)}) \leftarrow$ F-URF(R, Q, G_2, i, j, k)
      if $\|R - G_1 \otimes S \otimes G_2\|_b$ decreases **then break**
      else $(Q_{\cdot k}, G_{2(k\cdot)}) \leftarrow (Q'_{\cdot k}, G'_{2(k\cdot)})$
end while
return G_2

- We change the computation of the approximation error. `FastSTMF` computes the error of two-factorization, while `CFL`/`CFR` computes the tri-factorization error using the current factor matrices G_1, S, and G_2.
- We do not transpose the matrices nor permute the rows of matrices in `CFL`/`CFR` since this is performed as part of `triFastSTMF`.

The functions `F-ULF`, `F-URF` and `TD-A` used in `CFL` and `CFR` are the same as in the `FastSTMF` algorithm [12]. We present the pseudocode of `TD-A` in Algorithm 4, where the notation of functions used is given in [12].

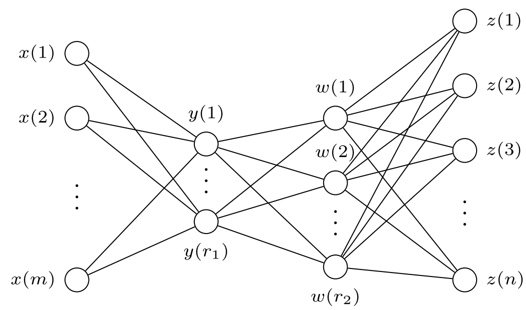


FIGURE 2. Example of a four-partition network.

Algorithm 4 `TD_A`

Input: data matrix $R \in \mathbb{R}_{\max}^{m\times n}$, left factor matrix U, right factor matrix V, row i of R
Output: *errors*, *row_indices*, *column_indices*
  $row\_indices \leftarrow \{f(i, t) : t = 1, \ldots, n\}$
  $errors$, $columns\_indices \leftarrow [\,], [\,]$
  for each *column* j of R
    $e \leftarrow \text{td}_{col}(R, U, V, j)$
    append e **to** *errors*
    $col\_indices \leftarrow \{f(t, j) : t = 1, \ldots, m\}$
    append *col_indices* **to** *columns_indices*
  return *errors*, *row_indices*, *column_indices*

3) DIFFERENT ASPECTS OF THE TRI-FACTORIZATION ON NETWORKS

The four-partition network shown in Figure 2 is an illustrative example of where we can apply tri-factorization for network structure analysis. We represent the four-partition network with three factor matrices which is the basis of tri-factorization methods. Further, different approaches to four-partition networks can be used depending on the nature of the data and the task that needs to be solved.

For a network Γ with a vertex set

$$V(\Gamma) = \{x(i) : i = 1, \ldots, m\} \cup \{y(j) : j = 1, \ldots, r_1\} \cup \{w(k) : k = 1, \ldots, r_2\} \cup \{z(\ell) : \ell = 1, \ldots, n\}$$

and an edge set $E(\Gamma)$, we define a matrix $G_1 \in \mathbb{R}_{\max}^{m\times r_1}$ such that $G_{1(ij)}$ represents the weight on the edge from $x(i)$ to $y(j)$, a matrix $S \in \mathbb{R}_{\max}^{r_1\times r_2}$ where S_{jk} represents the weight on the edge from $y(j)$ to $w(k)$ and a matrix $G_2 \in \mathbb{R}_{\max}^{r_2\times n}$ where $G_{2(k\ell)}$ represents the weights of the edges from $w(k)$ to $z(\ell)$. Then $R = G_1 \otimes S \otimes G_2$ is the $m \times n$ matrix such that

$$R_{i\ell} = \max_{1\le j\le r_1, 1\le k\le r_2} ((G_1)_{ij} + S_{jk} + (G_2)_{k\ell})$$

is the length of the longest path from $x(i)$ to $z(\ell)$, see Figure 2. If a matrix R is given, we can estimate G_1, S and G_2 with `triFastSTMF`.

The main question is how to present an arbitrary network as a four-partition network. The two main approaches are:

- **All nodes in the four-partition network are real nodes**. The matrices G_1, S, and G_2 represent weights of the real edges from the original network, which preserves the interpretability of the network since the relations are only between real nodes. Moreover, the size of the four-partition network remains the same size as the original network. This approach is suitable when the original network's structure already has four partitions.
- **Some nodes in the four-partition network are latent nodes**. The real nodes are only outer nodes (x, z), while latent nodes are inner nodes (y, w). In this case, the matrices G_1, S and G_2 represent latent features of the outer nodes and not real weights from the original network, leading to more difficult interpretability of the network since now the relations are also between real and latent nodes. The size of the four-partition network is larger than the size of the original network, which increases the complexity of the task using this approach.

We focus on the first approach, where all nodes in the network are real nodes since we want to use the patterns from the data to initialize the factor matrices, maintain network interpretability, demonstrate how to work with real four-partition networks, and consequently obtain a better approximation of matrices R, G_1, S, G_2. In this way, we fully present the power of tri-factorization over two-factorization and its primary purpose.

4) COMPARISON WITH OTHER STRATEGIES

In our work, we developed different tropical tri-factorization strategies, `triSTMF` and `Consecutive`, that are based on two-factorizations [11], [12]. We compare their effectiveness with proposed `triFastSTMF` in Section IV-A1.

The **triSTMF strategy** is based on the `TD_A` method from `FastSTMF`, and we implement `triSTMF` tri-factorization as two different two-factorizations:

i) Left factor matrix is $G_1 \otimes S$, right factor matrix is G_2.
ii) Left factor matrix is G_1, right factor matrix is $S \otimes G_2$.

We denote errors obtained from `TD_A` in the *i)* case as ε_L and errors in the *ii)* case as ε_R. We developed two versions called `triSTMF-BothTD` and `triSTMF-RandomTD`, which differ in the order of how the error is computed. In `triSTMF-BothTD`, the computation is performed using both ε_L and ε_R. The smaller error between ε_L and ε_R is selected to perform optimization. In contrast, `triSTMF-RandomTD` randomly computes ε_L or ε_R and continues with the optimization. Also, `triSTMF` uses `ULF` and `URF` from `STMF` as the basis for updating factor matrices. Note that we cannot use `F-ULF` and `F-URF` directly in the case of tri-factorization since the third factor matrix S introduces additional complexity to `F-ULF` and `F-URF`, resulting in incompatible operations. This results in a slow optimization process of both versions of `triSTMF`.

The **Consecutive strategy** has two versions: `lrConsecutive` and `rlConsecutive`. The goal of this strategy is to achieve tri-factorization by first applying `FastSTMF` to the data matrix R, resulting in factor matrices U and V. In the second step, `lrConsecutive` obtains the third factor matrix by applying `FastSTMF` to the matrix V to obtain S and G_2, while $G_1 = U$. In contrast, `rlConsecutive` applies `FastSTMF` to the matrix U to obtain G_1 and S, while $G_2 = V$. The drawback of a consecutive strategy is the consolidation of the patterns in one of the factor matrices during the first step.

B. SYNTHETIC DATA

We created a *synthetic data matrix* of size 200×100 using the $(\max, +)$ multiplication of three random non-negative matrices sampled from a uniform distribution over $[0, 1)$. Since the purpose of synthetic data is to present the perfect scenario in which the proposed method works best, we created our synthetic data using three random factor matrices of *sufficiently* large ranks $r_1 = 25$ and $r_2 = 20$. We use a synthetic data matrix to compare different tropical matrix factorization methods in Section IV-A1. We also created a *synthetic network* with four partitions of sizes $(m, r_1, r_2, n) = (45, 10, 15, 30)$ and used it to analyze the four-partition network in Section IV-A2.

C. REAL DATA

We downloaded the real-world interaction dataset of an ant colony [27] from the Network Data Repository [28]. The nodes represent 160 ants, the edges represent physical contact (interaction), and the edge weight is the frequency of interaction during 41 days in total. We preprocessed the network to the appropriate format for evaluation as explained in Section IV-B. In Figure 3, we show the daily average frequency of interactions between ants. The distance between the nodes indicates the strength of interactions, *i.e.*, nodes are closer when the interaction is stronger; contrary, nodes are farther apart when the interaction is weaker. The outer nodes interact less frequently with the nodes in the center of the network. We depict the individual frequency of interactions with the transparency of the edge color in Figure 3.

D. EVALUATION METRICS

In our work, we use the following metrics:

- *Root-mean-square error* or RMSE is a commonly used metric for comparing matrix factorization methods [12]. We use the RMSE in our experiments to evaluate the approximation error RMSE-A on the train data, and prediction error RMSE-P on the test data.
- *b-norm* is defined as $||W||_b = \sum_{i,j} |W_{ij}|$, and it is used in [11] and [12] as objective function. We also use the b-norm to minimize the approximation error of `triFastSTMF`.
- *Rand score* is a similarity measure between two clusterings that considers all pairs of samples and counts pairs assigned in the same or different clusters in the predicted and actual clusterings [29]. We use the Rand score to

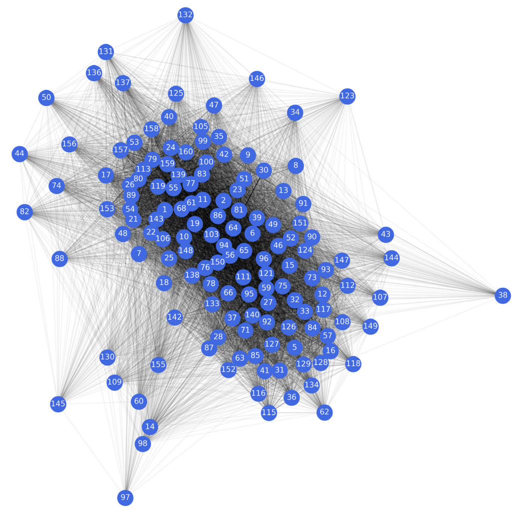


FIGURE 3. A real-world network of the daily average frequency of interactions in an ant colony. The strength of the interaction is visualized with the distance between nodes and edge transparency.

compare different partitioning strategies of the synthetic network.

E. EVALUATION

We conducted experiments on synthetic data matrices with true ranks $r_1 = 25$ and $r_2 = 20$. The experiments were repeated 25 times for 300 seconds using Random Acol initialization.

For the synthetic four-partition network reconstruction, we repeat the experiments 25 times using fixed initialization with different random and partially-random partitionings. Due to the smaller matrices, these experiments run for 100 seconds.

For real data, we used the Louvain method [23] to obtain r_1 and r_2. Furthermore, we randomly removed at most 20% of the edges. We use fixed initialization and run the experiments for 300 seconds.

IV. RESULTS

We perform experiments on synthetic and real data. First, we compare different tropical matrix factorization methods on the synthetic data matrix and show that `triFastSTMF` achieves the best results of all tropical approaches. Next, we analyze the effect of different partitioning strategies on the performance of `triFastSTMF`. Finally, we evaluate the proposed `triFastSTMF` on real data and compare it with `Fast-NMTF`.

A. SYNTHETIC DATA

1) COMPARISON BETWEEN THE TROPICAL MATRIX FACTORIZATION METHODS

We experiment with different two-factorization and tri-factorization tropical methods. The set of all tri-factorizations

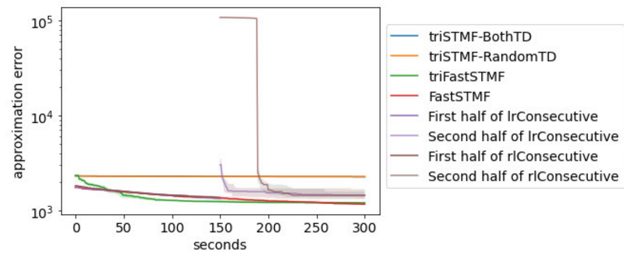


FIGURE 4. Comparison of different tropical tri-factorization methods. The median, first and third quartiles of the approximation error in 25 runs on the synthetic random tropical 200 × 100 matrix are shown.

represent a subset of all two-factorizations. Specifically, each tri-factorization is also a two-factorization, meaning that, in general, we cannot obtain better *approximation* results with tri-factorization compared to two-factorization. In Figure 4, we see that the first half of `lrConsecutive` is better than the second half of `lrConsecutive`. Namely, in the first half, we perform two-factorization, while in the second half, we factorize one of the factor matrices to obtain three factor matrices as the final result. This second approximation introduces uncertainty and larger errors than in the first half. We see similar behavior in `rlConsecutive`. In this scenario, we show that the two-factorization is better than the tri-factorization. We see that the results of `triSTMF-BothTD` and `triSTMF-RandomTD` overlap and do not make any updates during the limited running time since they use slow algorithms to update factor matrices.

Comparing the two-factorization method `FastSTMF` and the tri-factorization method `triFastSTMF`, we obtain a similar approximation error in Figure 4. We see that our proposed `triFastSTMF` achieves the lowest approximation error on the synthetic data matrix of all tested tropical tri-factorization methods. Tri-factorization may outperform two-factorization in a *limited running time* because of the nature of the data and the initialization of factor matrices. Theoretically, we expect that two-factorization and tri-factorization would achieve the same results when evaluated across a large number of datasets. Tri-factorization has demonstrated its superiority over two-factorization in many examples. An important application of tri-factorization is the fusion of data from different sources [3]. In our work, we show that tri-factorization can be applied to approximate and predict weights in four-partition networks.

2) ANALYSIS OF FOUR-PARTITION NETWORK CONSTRUCTION

We construct a random *tropical* network K of total 100 nodes with a four-partition $A \cup B \cup C \cup D$. We denote the sizes of the sets A, B, C and D as m, r_1, r_2 and n, respectively, and choose $(m, r_1, r_2, n) = (45, 10, 15, 30)$, see Figure 5. We want to check the robustness of the proposed `triFastSTMF` to the partitioning process and answer the following question: *Is*

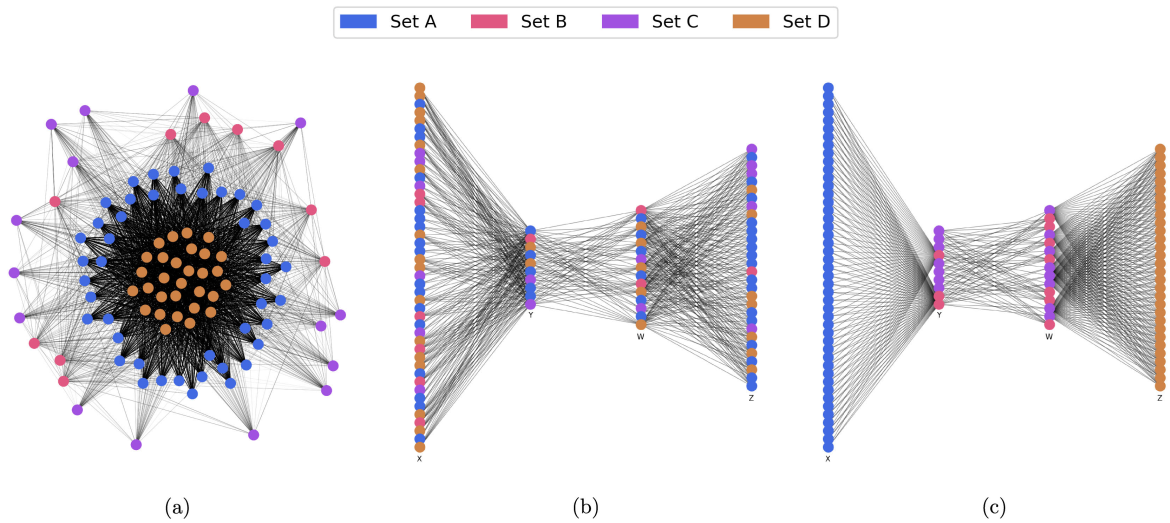


FIGURE 5. **(a) A synthetic random *tropical* network *K* of 100 nodes created by applying the tropical semiring on four sets *A*, *B*, *C* and *D*. Sets *A* and *D* are densely connected, following the network construction process. In contrast, sets *B* and *C* are less connected. Example of partitioning network *K*, using b) random and c) partially-random partitioning.**

approximation error stable among different choices of partitioning?

Network K contains the following edges:

- edges from A to B, denoted as $A - B$, have weights represented by a random matrix $M_1^{m \times r_1}$,
- edges from B to C, denoted as $B - C$, have weights represented by a random matrix $T^{r_1 \times r_2}$,
- edges from C to D, denoted as $C - D$, have weights represented by a random matrix $M_2^{r_2 \times n}$,
- edges from A to D, denoted as $A - D$, have weights represented by matrix $E = M_1 \otimes T \otimes M_2$.

Matrices M_1, T, and M_2 are sampled from a uniform distribution over $[0, 1)$.

We propose the following general algorithm to convert the input network K into a suitable form for tri-factorization. First, partition all network nodes into four sets, X, Y, W, and Z, with fixed sizes m, r_1, r_2 and n, respectively, in two ways:

- *random partitioning*: $X \cup Y \cup W \cup Z$ is a random four-partition of the chosen size. Random partitioning is a valid choice when all network nodes represent only one type of object. For example, in a social network, a node represents a person.
- *partially-random partitioning*: Y, W are random subsets of nodes of K of sizes r_1 and r_2, while $X = A$ and $Z = D$, where A, D are given. Partially-random partitioning is applicable when there are two types of objects represented in the network. For example, in the movie recommendation system, *users* belong to the set X and *movies* to Z. In this case, sets Y and W represent the latent features of X and Z.

See examples of random and partially-random partitioning in Figure 5, where we show only the edges $X - Y$, $Y - W$ and $W - Z$ to achieve easier readability of the network. Given the (pseudo)random partitioning, construct matrix R as the edges $X - Z$. The matrices G_1, S and G_2 are constructed as explained in III-A3 and can be used for the initialization of tri-factorization of R (fixed initialization). For the missing edges, we set the corresponding values in `triFastSTMF` to be a random number from elements of G_1, S and G_2. Tri-factorization on R will return updated R, G_1, S, G_2 with approximated/predicted weights on edges.

We show that partially-random partitioning achieves higher Rand scores, but approximation errors are similar to the ones obtained by random partitioning, see Figure 6. We conclude that the partitioning process does not significantly affect the approximation error of `triFastSTMF`. Still, if there is some additional knowledge about the sets of partition, it is better to use partially-random partitioning. When we do not know the real partition, random partitioning or advanced algorithms, such as the Louvain method, can be used.

B. REAL DATA

We test our method on a real-world interaction dataset of the ant colony introduced in Section III-C. We describe the data on the interaction between pairs of ants using a weighted adjacency matrix of size 160×160, where diagonal elements are equal to 0. The adjacency matrix is symmetric, and we use the data from the upper triangular part to construct the matrix H, where each row describes one pair of ants, and columns represent a specific day. Since H is large, we use *k-means* clustering to obtain 50 clusters and analyze the behavioral patterns of the ants on each day, shown in Figure 7.

There are three groups of days with different dynamics of ant interaction: D_1 represents days $1 - 19$, D_2 are days $20-31$, and D_3 are days $32-41$. We preprocessed the data for each group of days D_1, D_2 and D_3 such that the corresponding weight between two ants represents the daily average of all interactions for the specific days, see Figure 8.

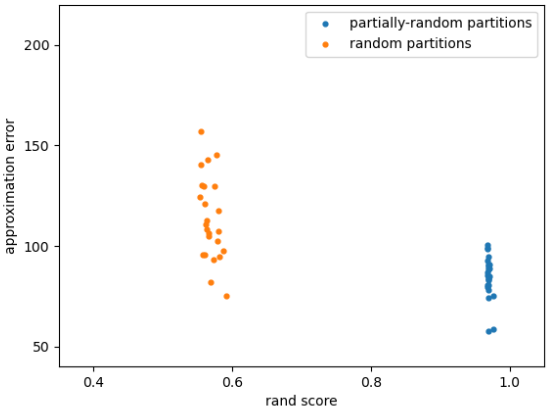


FIGURE 6. Rand score and approximation error of `triFastSTMF` on 25 random and 25 partially-random partitionings of synthetic data. We performed one run of 100 seconds for each matrix R and used true ranks r_1 and r_2 as factorization parameters.

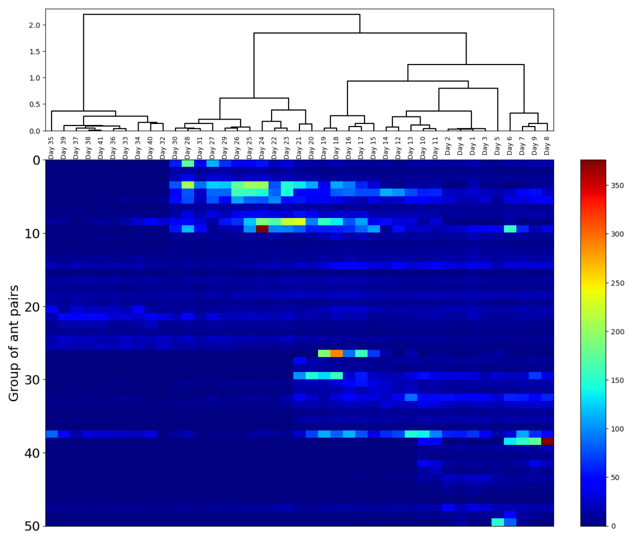


FIGURE 7. Analysis of ants' behavioral patterns over 41 days. The rows represent centroids of clustered ant pairs with k-means using $k = 50$, and the columns denote daily interactions. Rows and columns are ordered using optimal leaf ordering for hierarchical clustering [30] using cosine distance and Ward linkage.

The group D_2, which contains days 20−31 and 140 pairs of ants with positive weights, is the most dynamic of the three groups and has local communities. We construct a network N from the group D_2, where the nodes represent individual ants, and the weight of the edges represents the strength of the interactions between ants. The network density of N is 88%. The weighted adjacency matrix of N is denoted as A.

Next, we construct ten different networks, $N_1, \ldots, N_{10}$ by sampling with replacement the edges from N. Each sampled network has at most 20% of missing edges from N, which are used for evaluation. For each network N_i, $i \in \{1, \ldots, 10\}$, we construct the weighted adjacency matrix A_i with the exact same size and ordering of the nodes in rows and columns as in matrix A. Now, to apply tri-factorization on networks, we need to perform Louvain partitioning [23] for each N_i to obtain a four-partition of its nodes: $X_i \cup Y_i \cup W_i \cup Z_i$.

TABLE 1. Louvain partitioning of N_i where $i \in [1, 10]$, containing 140 nodes from days 20 − 31.

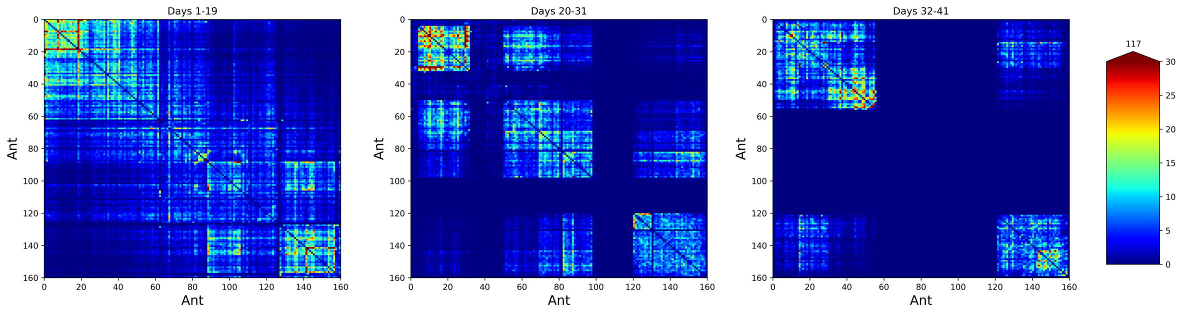
Network	m	r_1	r_2	n	μ
N_1	65	21	2	52	84%
N_2	57	22	5	56	81%
N_3	57	24	2	57	81%
N_4	60	21	2	57	84%
N_5	60	20	4	56	83%
N_6	61	19	2	58	85%
N_7	57	18	15	50	76%
N_8	52	23	14	51	74%
N_9	67	4	2	67	96%
N_{10}	65	15	2	58	88%

Louvain method assigns sets of a four-partition and enables favoring larger communities using parameter γ. Different partitions are obtained for different values of γ, from which we select a connected four-partition network. We prefer the outer sets X_i and Z_i of the corresponding sizes m and n, respectively, to have a larger size than the inner sets Y_i and W_i of sizes r_1 and r_2, respectively. This ensures that the matrix factorization methods embed data into low-dimensional space using rank values $r_1, r_2 \ll \min\{m, n\}$. Louvain algorithm results in different parameters m, r_1, r_2 and n for each N_i, $i \in \{1, \ldots, 10\}$, see Table 1. We define μ to represent a percentage of nodes in outer sets. Table 1 shows that $\mu \geq 74\%$ for all N_i. We construct R_i matrices of corresponding sizes $m \times n$ using the edges from X_i to Z_i, and the corresponding matrices G_1, S and G_2 of sizes $m \times r_2$, $r_1 \times r_2$ and $r_2 \times n$, respectively, using all four sets. In R_i, we mask all values equal to 0.

We run matrix factorization methods on each R_i matrix using the corresponding factor matrices G_1, S, and G_2 for fixed initialization and obtain updated matrices G_1, S, and G_2. Since we use fixed initialization, we evaluate each method only once because there is no presence of randomness. In Table 2, we present the comparison between our proposed `triFastSTMF` and `Fast-NMTF`. The results show that `Fast-NMTF` achieves a smaller approximation error RMSE-A, while `triFastSTMF` outperforms `Fast-NMTF` in a better prediction error RMSE-P. This result is consistent with previous research in [11] and [12], where we have shown that matrix factorization over the tropical semiring is more robust to overfitting compared to methods using standard linear algebra.

The matrix R_i contains only edges $X_i - Z_i$. All other edges $X_i - Y_i$, $Y_i - W_i$ and $W_i - Z_i$ are hidden in the corresponding factor matrices G_1, S and G_2. If we want to obtain predictions for all edges of the network N using different partitions of N_i, we need to also consider factor matrices, not just matrix R_i. To achieve this, we take into account the corresponding G_1, S and G_2, including their products $G_1 \otimes S$, $S \otimes G_2$ and $G_1 \otimes S \otimes G_2$. The edges that were removed from N during the sampling process to obtain N_i are used to measure the prediction error, while the edges in N_i are used for approximation.

In Table 3, we present the comparison between our proposed `triFastSTMF` and `Fast-NMTF` on network N using different partitions of N_i. The results show that



(a) (b) (c)

FIGURE 8. Comparison between the daily average of all interactions between ant pairs for different groups of days: (a) days 1-19, (b) days 20-31, and (c) days 32-41. Rows and columns are ordered using optimal leaf ordering for hierarchical clustering [30] using cosine distance and Ward linkage.

TABLE 2. RMSE-A and RMSE-P on data matrices R_i. The result of the best method in the comparison between `triFastSTMF` and `Fast-NMTF` is shown in bold.

Metric	Method	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
RMSE-P	`triFastSTMF`	1.90	**1.07**	2.32	**1.09**	**1.15**	1.78	**0.88**	**0.88**	**1.38**	2.05
	`Fast-NMTF`	**0.93**	1.25	**1.42**	1.11	1.21	**0.82**	1.17	1.10	1.48	**1.59**
RMSE-A	`triFastSTMF`	1.90	0.90	2.34	1.23	1.02	1.64	0.92	0.69	1.33	2.12
	`Fast-NMTF`	**0.54**	**0.37**	**0.49**	**0.52**	**0.43**	**0.53**	**0.17**	**0.17**	**0.80**	**0.58**

TABLE 3. RMSE-A and RMSE-P on network N using different partitions of N_i. The result of the best method in the comparison between `triFastSTMF` and `Fast-NMTF` is shown in bold.

Metric	Method	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}
RMSE-P	`triFastSTMF`	9.43	**11.40**	12.76	10.35	**10.99**	**8.56**	**10.63**	**12.85**	7.56	9.28
	`Fast-NMTF`	**7.24**	3602725.48	**6.21**	**7.09**	289129.41	116821.55	12733965.01	763401.17	**6.24**	**6.18**
RMSE-A	`triFastSTMF`	8.43	**9.95**	12.37	9.63	**10.10**	**8.50**	**10.20**	**11.90**	7.35	8.75
	`Fast-NMTF`	**6.07**	1034154.82	**6.22**	**6.02**	339449.43	118555.72	13423203.65	691714.60	**6.02**	**6.15**

`triFastSTMF` and `Fast-NMTF` have the same number of wins regarding the RMSE-A and RMSE-P. However, the main difference between `triFastSTMF` and `Fast-NMTF` is in the fact that `Fast-NMTF` achieves an enormous error compared to `triFastSTMF` in half of the cases. This is because now we are also predicting edges $X_i - Y_i$, $Y_i - W_i$, $W_i - Z_i$ and $X_i - W_i$, $Y_i - Z_i$, which we obtain by multiplying the corresponding factor matrices G_1, S and G_2 properly. There is no guarantee that the factor matrices G_1, S, and G_2 and their products are on the same scale as the data matrix R_i on which the matrix factorization methods were trained. Since `Fast-NMTF` uses standard linear algebra, one more matrix multiplication is needed to get to the original data scale. Using standard operations $+$ and $\times$ results in significant error, since the predicted values expand in magnitude quickly. `triFastSTMF` does not have this problem because it is based on tropical semiring, and the operators max and $+$ are more averse to predicting large values.

V. CONCLUSION

Matrix factorization is a popular data embedding approach used in various machine learning applications. Most factorization methods use standard linear algebra. Recent research introduced tropical semiring to matrix factorization, which enables the modeling of nonlinear relations. Two-factorization approaches are often applied to study bipartite and tripartite networks. However, tri-factorization is suitable for application on four-partition networks, and to the best of our knowledge, our work is the first to explore this option.

In this study, we evaluate different strategies based on two-factorization, called `triSTMF` and `Consecutive`. Both strategies have different drawbacks, such as a slow optimization process in `triSTMF` and the overfitting of one of the factor matrices in `Consecutive`. These limitations have motivated us to develop a novel tri-factorization approach that addresses the limitations of `triSTMF` and `Consecutive`. We propose `triFastSTMF`, a tri-factorization algorithm over the tropical semiring that can be used for a single data source. Our proposed algorithm is based on `FastSTMF`, a two-factorization method, with the necessary modifications for tri-factorization. We also provide a detailed theoretical analysis for solving the linear system and computing the third factor matrix. The obtained solution is used for the optimization in the proposed `triFastSTMF`.

We tested the method on synthetic and real data, applied it to the edge approximation and prediction task in four-partition networks, and demonstrated that `triFastSTMF` achieves close approximation and prediction results as `Fast-NMTF`. Furthermore, `triFastSTMF` is more robust than `Fast-NMTF` in cases when methods are fitted on a part of the network and then used to approximate and predict the entire network.

Although in this study we presented the proposed method on a *single* data source, we established the basis for creating a model capable of combining *multiple* data sources. Our future work involves the application and modification of the proposed `triFastSTMF` to the data fusion problem, which often uses tri-factorization.

SUPPORTING INFORMATION

The supporting Python notebooks and data are available on GitHub (https://github.com/Ejmric/triFastSTMF) and Zenodo (https://doi.org/10.5281/zenodo.7928148). Real-world interaction dataset of an ant colony named insecta-ant-colony3 was taken from *Animal Social Networks* data collection on http://networkrepository.com.

REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[2] W. Liu, N. Zheng, and Q. You, "Nonnegative matrix factorization and its applications in pattern recognition," *Chin. Sci. Bull.*, vol. 51, no. 1, pp. 7–18, Jan. 2006.

[3] M. Žitnik and B. Zupan, "Data fusion by matrix factorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 41–53, Jan. 2015.

[4] J. Hook, "Linear regression over the max-plus semiring: Algorithms and applications," 2017, *arXiv:1712.03499*.

[5] S. Karaev, J. Hook, and P. Miettinen, "Latitude: A model for mixed linear-tropical matrix factorization," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2018, pp. 360–368.

[6] S. Karaev and P. Miettinen, "Capricorn: An algorithm for subtropical matrix factorization," in *Proc. SIAM Int. Conf. Data Mining*, Jun. 2016, pp. 702–710.

[7] S. Karaev and P. Miettinen, "Algorithms for approximate subtropical matrix factorization," 2017, *arXiv:1707.08872*.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[9] Z. Zhang, T. Li, C. Ding, and X. Zhang, "Binary matrix factorization with applications," in *Proc. 7th IEEE Int. Conf. Data Mining (ICDM)*, Oct. 2007, pp. 391–400.

[10] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[11] A. Omanović, H. Kazan, P. Oblak, and T. Curk, "Sparse data embedding and prediction by tropical matrix factorization," *BMC Bioinf.*, vol. 22, no. 1, pp. 1–18, Dec. 2021.

[12] A. Omanović, P. Oblak, and T. Curk, "FastSTMF: Efficient tropical matrix factorization algorithm for sparse data," 2022, *arXiv:2205.06619*.

[13] S. Karaev and P. Miettinen, "Cancer: Another algorithm for subtropical matrix factorization," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2016, pp. 576–592.

[14] J. Weston, R. J. Weiss, and H. Yee, "Nonlinear latent factorization by embedding multiple user interests," in *Proc. 7th ACM Conf. Recommender Syst.*, Oct. 2013, pp. 65–68.

[15] T. Le Van, S. Nijssen, M. van Leeuwen, and L. De Raedt, "Semiring rank matrix factorization," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1737–1750, Aug. 2017.

[16] L. Zhang, G. Naitzat, and L.-H. Lim, "Tropical geometry of deep neural networks," in *Proc. 35th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 80, J. Dy and A. Krause, Eds. Stockholm Sweden, Jul. 2018, pp. 5824–5832. [Online]. Available: http://proceedings.mlr.press/v80/zhang18i.html

[17] G. Fu, J. Wang, C. Domeniconi, and G. Yu, "Matrix factorization-based data fusion for the prediction of lncRNA–disease associations," *Bioinformatics*, vol. 34, no. 9, pp. 1529–1537, May 2018.

[18] H. Wang, F. Nie, H. Huang, and F. Makedon, "Fast nonnegative matrix tri-factorization for large-scale data co-clustering," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1–15.

[19] M. Žitnik, V. Janjić, C. Larminie, B. Zupan, and N. Pržulj, "Discovering disease-disease associations by fusing systems-level molecular data," *Sci. Rep.*, vol. 3, no. 1, pp. 1–9, Nov. 2013.

[20] A. Čopar, B. Zupan, and M. Zitnik, "Fast optimization of non-negative matrix tri-factorization," *PLoS ONE*, vol. 14, no. 6, Jun. 2019, Art. no. e0217994.

[21] B. De Schutter and B. De Moor, "Matrix factorization and minimal state space realization in the max-plus algebra," in *Proc. Amer. Control Conf.*, 1997, pp. 3136–3140.

[22] S. Hougardy, "The Floyd–Warshall algorithm on graphs with negative cycles," *Inf. Process. Lett.*, vol. 110, nos. 8–9, pp. 279–281, Apr. 2010.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[24] G. D. J. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 5, pp. 268–278, Mar. 1993.

[25] E. Theodosis and P. Maragos, "Analysis of the Viterbi algorithm using tropical algebra and geometry," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2018, pp. 1–5.

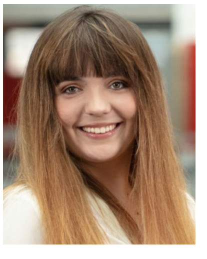
[26] S. Gaubert and M. Plus, "Methods and applications of (max,+) linear algebra," in *Proc. Annu. Symp. Theor. Aspects Comput. Sci.* Cham, Switzerland: Springer, 1997, pp. 261–282.

[27] D. P. Mersch, A. Crespi, and L. Keller, "Tracking individuals shows spatial fidelity is a key regulator of ant social organization," *Science*, vol. 340, no. 6136, pp. 1090–1093, May 2013.

[28] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, 2015, pp. 1–12. [Online]. Available: http://networkrepository.com

[29] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.

[30] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," *Bioinformatics*, vol. 17, no. 1, pp. S22–S29, Jun. 2001.



AMRA OMANOVIĆ received the master's degree from the Faculty of Computer and Information Science, University of Ljubljana, in 2018. She is currently a Junior Researcher with the Faculty of Computer and Information Science, University of Ljubljana. Her research interests include the application of linear algebra over semirings in data embedding and fusion methods.

POLONA OBLAK received the Ph.D. degree from the Faculty of Mathematics and Physics, University of Ljubljana, in 2008. She is currently an Associate Professor with the Faculty of Computer and Information Science, University of Ljubljana. Her research interests include inverse eigenvalue problems for graphs, linear algebra over semirings, and combinatorial matrix theory.

TOMAŽ CURK received the Ph.D. degree from the Faculty of Computer and Information Science, University of Ljubljana, in 2007. He is currently an Associate Professor and the Vice-Dean for Research with the Faculty of Computer and Information Science, University of Ljubljana. His research interests include the application of machine learning and data integration methods in bioinformatics.