S1 Methods

**Algorithm S1** This algorithm calculates $S^3$ of a network alignment. Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two networks and $f : V_1 \rightarrow V_2$ be an alignment between the two networks. We calculate $S^3$ by creating the alignment graph (described below) and using the alignment graph to calculate the number of conserved and non-conserved edges. The alignment graph, $G_A(V_A, E_A)$, is the graph in which each aligned pair is fused into a supernode and the number of edges between two supernodes is the number of edges connecting the nodes in one supernode to the nodes in the other supernode. The algorithm creates the alignment graph and counts the number of conserved edges (the number of supernode pairs in $G_A$ with double edges between them) and the number of non-conserved edges (the number of supernode pairs in $G_A$ with a single edge between them).

1: procedure $S^3(f, G_1, G_2)$
2: Let $G_2'(V_2', E_2')$ be the induced subgraph of $G_2$ induced by $V_2' = \{ f(u) \mid u \in V_1 \}$. $\triangleright$ Edge set of alignment graph, $G_A$
3: $E_T \leftarrow (|E_1| + |E_2|) \times 1$ vector
4: $i \leftarrow 1$
5: for $(u, v) \in E_2'$ do $\triangleright$ Add edges mapped from $G_2'$
6: $E_T[i] \leftarrow (\min(u, v), \max(u, v))$
7: $i \leftarrow i + 1$
8: for $(u, v) \in E_1$ do $\triangleright$ Add edges mapped from $G_1$
9: $u' \leftarrow f(u)$
10: $v' \leftarrow f(v)$
11: $E_T[i] \leftarrow (\min(u', v'), \max(u', v'))$
12: $i \leftarrow i + 1$
13: $E_A \leftarrow |V_1| \times |V_2|$ matrix of 0s $\triangleright$ Alignment graph, $G_A$, where $E_A[u, v]$ is the number of edges between supernode $u$ and supernode $v$. $G_A$ is stored as a sparse matrix data structure (e.g., hash table or compressed sparse column/row format), so that the following accumulation loop can be done with $O(|E_T|)$ time complexity.
14: for $(u, v) \in E_T$ do
15: $E_A[u, v] \leftarrow E_A[u, v] + 1$ $\triangleright$ Number of conserved edges
16: $N_c \leftarrow 0$
17: $N_n \leftarrow 0$ $\triangleright$ Number of non-conserved edges
18: for $(u, v, w) \in \text{nonzeros}(E_A)$ do $\triangleright \text{nonzeros}(E_A)$ is the set of index pairs such that $E_A[u, v] \neq 0$
19: if $E_A[u, v] = 2$ then
20: $N_c \leftarrow N_c + 1$
21: else if $E_A[u, v] = 1$ then
22: $N_n \leftarrow N_n + 1$
23: return $\frac{N_c}{N_c + N_n}$
Algorithm S2 This algorithm calculates DS$^3$ of a dynamic network alignment. Let $H_1(V_1, T_1)$ and $H_2(V_2, T_2)$ be two temporal networks and assume without loss of generality that for each node pair $(u, v) \in H_i, i = 1, 2$, there exists no two events associated with the node pair that overlap with each other. Let $f : V_1 \rightarrow V_2$ be an alignment between the two networks. We calculate DS$^3$ by creating the alignment graph (described below) and using the alignment graph to calculate the alignment conserved event time (alignment CET) and alignment non-conserved event time (alignment NCET). The alignment graph, $H_A(V_A, T_A)$, is the graph in which each aligned pair is fused into a supernode and the set of events between two supernodes in the union of the set of events between nodes in one supernode and nodes in the other supernode. The algorithm creates the alignment graph and calculates the sum of the CET and NCET of each supernode pair $H_A$ (using Algorithm S3) over all supernode pairs.

1: procedure DS$^3$(f, $H_1$, $H_2$)
2: Let $H'_2(V'_2, T'_2)$ be the induced subgraph of $H_2$ induced by $V'_2 = \{ f(u) \mid u \in V_1 \}$.
3: $T_T \leftarrow (|T_1| + |T_2|) \times 1$ vector \(\triangleright\) Event set of alignment graph, $H_A$
4: $i \leftarrow 1$
5: for $(u, v, t_s, t_e) \in T'_2$ do
6: \hspace{1em} $I \leftarrow \{(t_s, t_e)\}$ \(\triangleright\) Add events mapped from $H'_2$
7: \hspace{1em} $T_T[i] \leftarrow (\min(u, v), \max(u, v), I)$
8: \hspace{1em} $i \leftarrow i + 1$
9: for $(u, v, t_s, t_e) \in T_1$ do \(\triangleright\) Add events mapped from $H_1$
10: \hspace{1em} $u' \leftarrow f(u)$
11: \hspace{1em} $v' \leftarrow f(v)$
12: \hspace{1em} $I \leftarrow \{(t_s, t_e)\}$
13: \hspace{1em} $T_T[i] \leftarrow (\min(u', v'), \max(u', v'), I)$
14: \hspace{1em} $i \leftarrow i + 1$
15: $T_A \leftarrow |V_1| \times |V_2|$ matrix of \(\varnothing\)s \(\triangleright\) Alignment graph, $H_A$, where $T_A[u, v]$ is the set of edges between supernode $u$ and supernode $v$. $H_A$ is stored as a sparse matrix data structure (e.g., hash table or compressed sparse column/row format), so that the following accumulation loop can be done with $O(|T_T|)$ time complexity.
16: for $(u, v, I) \in T_T$ do \(\triangleright\) Add $I$ using sorted merge (where you sort w.r.t. the start time $t_s$), so that we will not have to sort $T_A[u, v]$ for the CET\_NCET function below
17: \hspace{1em} $T_A[u, v] \leftarrow T_A[u, v] \cup I$
18: $T_c \leftarrow 0$ \(\triangleright\) Total conserved event activity
19: $T_n \leftarrow 0$ \(\triangleright\) Total non-conserved event activity
20: for $(u, v) \in \text{nonzeros}(T_A)$ do \(\triangleright\) nonzeros($T_A$) is the set of index pairs such that $T_A[u, v] \neq \varnothing$
21: \hspace{1em} $(T'_c, T'_n) \leftarrow \text{CET\_NCET}(T_A[u, v])$
22: \hspace{1em} $T_c \leftarrow T_c + T'_c$
23: \hspace{1em} $T_n \leftarrow T_n + T'_n$
24: return $\frac{T_n}{T_c + T_n}$
Algorithm S3 Given a set of events times (stored as time intervals), \( I = \{ (t_s, t_e) \} \), sorted from smallest to largest by start time \( t_s \), this algorithm calculates the CET and NCET of the set. The CET is the total amount of time during which two events are active (i.e. where the time intervals overlap), and the NCET is the total amount of time during which only one event is active (i.e. one time interval exists and the other does not).

1: procedure CET,NCFET(I)
2: if \( I = \emptyset \) then
3: return (0, 0)
4: \((T_c, T_n)\) ← (0, 0)
5: \((a, b)\) ← pop \( I \)
6: while \( T \neq \emptyset \) do
7: \((c, d)\) ← pop \( I \)
8: if \( b \leq c \) then
9: \( T_n \) ← \( T_n + (b - a) \)
10: \((a, b)\) ← \((c, d)\)
11: else if \( b > d \) then
12: \( T_n \) ← \( T_n + (c - a) \)
13: \( T_c \) ← \( T_c + (d - c) \)
14: \((a, b)\) ← \((d, b)\)
15: else
16: \( T_n \) ← \( T_n + (c - a) \)
17: \( T_c \) ← \( T_c + (b - c) \)
18: \((a, b)\) ← \((b, d)\)
19: \( T_n \) ← \( T_n + (b - a) \)
20: return \((T_c, T_n)\)

S1.1 Other methods we evaluate against

In Section 3.1 of the main paper, we test whether optimizing both dynamic edge (event) conservation and dynamic node conservation in DynaMAGNA++ is better than optimizing either dynamic edge conservation alone or dynamic node conservation alone. That is, we compare three different versions of DynaMAGNA++ that differ in their optimization functions. The three versions optimize: (i) a combination of dynamic edge conservation and dynamic node conservation (corresponding to \( \alpha = 0.5 \), named DynaMAGNA++ (E+N)), (ii) dynamic edge conservation only (corresponding to \( \alpha = 1 \), named DynaMAGNA++ (E)), and (iii) dynamic node conservation only (corresponding to \( \alpha = 0 \), named DynaMAGNA++ (N)).

Since we show that optimizing both dynamic edge (event) conservation and dynamic node conservation, i.e., DynaMAGNA++ (E+N), performs best, we mainly report results using the \( \alpha \) parameter value of 0.5 for both DynaMAGNA++ and MAGNA++ in the main paper. Here in the Supplement, we report results for all three versions for both DynaMAGNA++ and MAGNA++, i.e., we report results using \( \alpha = 0.5 \), \( \alpha = 1 \), and \( \alpha = 0 \) for both DynaMAGNA++ and MAGNA++.

S1.2 Evaluation using synthetic networks

We follow our discussion on evaluation of synthetic networks in Section 3.1 of the main paper. In the main paper, we evaluate DynaMAGNA++ using synthetic networks generated using a biologically inspired network evolution model. In the Supplement, we also report results for networks generated using a social network evolution model (Leskovec et al., 2008). Specifically, we generate 15 dynamic networks using three network evolution models (or versions of the same model with different parameter values) that simulate the evolution of social networks, resulting in five networks per model (Hulovatyy et al., 2015). The three models we use are the i) exponential arrival function with 20 edges initiated using preferential attachment, (ii) quadratic arrival function with edge initiation parameters \( \lambda_1 = 0.032, \alpha_1 = 0.8, \beta_1 = 0.002 \), and (iii) linear arrival function with edge initiation parameters \( \lambda_1 = 0.02, \alpha_1 = 0.9, \beta_1 = 0.004 \), each having approximately 1000 nodes. Hulovatyy et al. (2015) generalized the static versions of these models to their dynamic counterparts, and we rely on the same model networks as those used by Hulovatyy et al. (2015) (see their paper for details). Detailed descriptions of the synthetic networks are shown in Supplementary Table S2 and Supplementary Figures S4 and S3.
S1.3 Evaluation using real-world networks

We follow our discussion on evaluation of real-world networks in Section 3.2 of the main paper. In the main paper, we evaluate DynaMAGNA++ using the Grevy’s zebra proximity network. We report in the Supplement the results for the onager proximity network, which contains information on interactions between 28 onagers in India over 81 days. Onagers are a species closely related to Grevy’s zebras. The data was collected by driving a predetermined route over the study area each day while searching for herds. There are 522 events in the onager network.

Supplementary Tables

<table>
<thead>
<tr>
<th>Network model</th>
<th>Random instance</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEO-GD $p = 0.3$</td>
<td>Instance 1</td>
<td>1000</td>
<td>792</td>
</tr>
<tr>
<td>GEO-GD $p = 0.3$</td>
<td>Instance 2</td>
<td>1000</td>
<td>966</td>
</tr>
<tr>
<td>GEO-GD $p = 0.3$</td>
<td>Instance 3</td>
<td>1002</td>
<td>923</td>
</tr>
<tr>
<td>GEO-GD $p = 0.3$</td>
<td>Instance 4</td>
<td>1000</td>
<td>874</td>
</tr>
<tr>
<td>GEO-GD $p = 0.3$</td>
<td>Instance 5</td>
<td>1001</td>
<td>1058</td>
</tr>
<tr>
<td>GEO-GD $p = 0.7$</td>
<td>Instance 1</td>
<td>1002</td>
<td>5407</td>
</tr>
<tr>
<td>GEO-GD $p = 0.7$</td>
<td>Instance 2</td>
<td>1000</td>
<td>4789</td>
</tr>
<tr>
<td>GEO-GD $p = 0.7$</td>
<td>Instance 3</td>
<td>1000</td>
<td>5390</td>
</tr>
<tr>
<td>GEO-GD $p = 0.7$</td>
<td>Instance 4</td>
<td>1002</td>
<td>3188</td>
</tr>
<tr>
<td>GEO-GD $p = 0.7$</td>
<td>Instance 5</td>
<td>1000</td>
<td>4547</td>
</tr>
<tr>
<td>SF-GD $p = 0.3, q = 0.7$</td>
<td>Instance 1</td>
<td>779</td>
<td>1415</td>
</tr>
<tr>
<td>SF-GD $p = 0.3, q = 0.7$</td>
<td>Instance 2</td>
<td>733</td>
<td>1054</td>
</tr>
<tr>
<td>SF-GD $p = 0.3, q = 0.7$</td>
<td>Instance 3</td>
<td>742</td>
<td>1155</td>
</tr>
<tr>
<td>SF-GD $p = 0.3, q = 0.7$</td>
<td>Instance 4</td>
<td>708</td>
<td>936</td>
</tr>
<tr>
<td>SF-GD $p = 0.3, q = 0.7$</td>
<td>Instance 5</td>
<td>751</td>
<td>1216</td>
</tr>
<tr>
<td>SF-GD $p = 0.7, q = 0.6$</td>
<td>Instance 1</td>
<td>988</td>
<td>3419</td>
</tr>
<tr>
<td>SF-GD $p = 0.7, q = 0.6$</td>
<td>Instance 2</td>
<td>980</td>
<td>3371</td>
</tr>
<tr>
<td>SF-GD $p = 0.7, q = 0.6$</td>
<td>Instance 3</td>
<td>974</td>
<td>2974</td>
</tr>
<tr>
<td>SF-GD $p = 0.7, q = 0.6$</td>
<td>Instance 4</td>
<td>981</td>
<td>3227</td>
</tr>
<tr>
<td>SF-GD $p = 0.7, q = 0.6$</td>
<td>Instance 5</td>
<td>980</td>
<td>2931</td>
</tr>
</tbody>
</table>

Table S1: The number of nodes and edges in the flattened network, and the number of snapshots, of each of the synthetic dynamic network instances produced by the biologically inspired network models. Five random instances are generated for each each model. The model parameters are discussed in Section 3.1 in the main paper.
<table>
<thead>
<tr>
<th>Network model</th>
<th>Random instance</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social (i)</td>
<td>Instance 1</td>
<td>981</td>
<td>18613</td>
</tr>
<tr>
<td>Social (i)</td>
<td>Instance 2</td>
<td>981</td>
<td>18575</td>
</tr>
<tr>
<td>Social (i)</td>
<td>Instance 3</td>
<td>981</td>
<td>18643</td>
</tr>
<tr>
<td>Social (i)</td>
<td>Instance 4</td>
<td>981</td>
<td>18614</td>
</tr>
<tr>
<td>Social (i)</td>
<td>Instance 5</td>
<td>981</td>
<td>18619</td>
</tr>
<tr>
<td>Social (ii)</td>
<td>Instance 1</td>
<td>1000</td>
<td>1935</td>
</tr>
<tr>
<td>Social (ii)</td>
<td>Instance 2</td>
<td>1000</td>
<td>2005</td>
</tr>
<tr>
<td>Social (ii)</td>
<td>Instance 3</td>
<td>1000</td>
<td>2006</td>
</tr>
<tr>
<td>Social (ii)</td>
<td>Instance 4</td>
<td>1000</td>
<td>2021</td>
</tr>
<tr>
<td>Social (ii)</td>
<td>Instance 5</td>
<td>1000</td>
<td>1953</td>
</tr>
<tr>
<td>Social (iii)</td>
<td>Instance 1</td>
<td>1000</td>
<td>2148</td>
</tr>
<tr>
<td>Social (iii)</td>
<td>Instance 2</td>
<td>1000</td>
<td>2102</td>
</tr>
<tr>
<td>Social (iii)</td>
<td>Instance 3</td>
<td>1000</td>
<td>2119</td>
</tr>
<tr>
<td>Social (iii)</td>
<td>Instance 4</td>
<td>1000</td>
<td>2094</td>
</tr>
<tr>
<td>Social (iii)</td>
<td>Instance 5</td>
<td>1000</td>
<td>2108</td>
</tr>
</tbody>
</table>

Table S2: The number of nodes and edges in the flattened network, and the number of snapshots, of each of the synthetic dynamic network instances produced by the social network evolution models. Five random instances are generated for each model. The model parameters are discussed in Supplementary Section S1.2.

<table>
<thead>
<tr>
<th>NA method</th>
<th>AUPR</th>
<th>F-score&lt;sub&gt;cross&lt;/sub&gt;</th>
<th>F-score&lt;sub&gt;max&lt;/sub&gt;</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynaMAGNA++ (E)</td>
<td>0.742</td>
<td>0.550</td>
<td>0.762</td>
<td>0.919</td>
</tr>
<tr>
<td>DynaMAGNA++ (E+N)</td>
<td>0.865</td>
<td>0.700</td>
<td>0.771</td>
<td>0.950</td>
</tr>
<tr>
<td>DynaMAGNA++ (N)</td>
<td>0.994</td>
<td>0.950</td>
<td>0.962</td>
<td>0.998</td>
</tr>
<tr>
<td>MAGNA++ (E)</td>
<td>0.681</td>
<td>0.600</td>
<td>0.608</td>
<td>0.844</td>
</tr>
<tr>
<td>MAGNA++ (E+N)</td>
<td>0.711</td>
<td>0.550</td>
<td>0.645</td>
<td>0.863</td>
</tr>
<tr>
<td>MAGNA++ (N)</td>
<td>0.492</td>
<td>0.425</td>
<td>0.716</td>
<td>0.871</td>
</tr>
</tbody>
</table>

Table S3: Network discrimination performance of DynaMAGNA++ and MAGNA++, while optimizing both edge and node conservation (E+N), edge conservation alone (E), and node conservation alone (N), for biological synthetic networks, with respect to the area under the precision-recall curve (AUPR), F-score at which precision and recall cross and are thus equal (F-score<sub>cross</sub>), maximum F-score (F-score<sub>max</sub>), and the area under the ROC curve (AUROC).

<table>
<thead>
<tr>
<th>NA method</th>
<th>AUPR</th>
<th>F-score&lt;sub&gt;cross&lt;/sub&gt;</th>
<th>F-score&lt;sub&gt;max&lt;/sub&gt;</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynaMAGNA++ (E)</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>DynaMAGNA++ (E+N)</td>
<td>0.908</td>
<td>0.800</td>
<td>0.839</td>
<td>0.959</td>
</tr>
<tr>
<td>DynaMAGNA++ (N)</td>
<td>0.751</td>
<td>0.600</td>
<td>0.723</td>
<td>0.883</td>
</tr>
<tr>
<td>MAGNA++ (E)</td>
<td>0.586</td>
<td>0.467</td>
<td>0.706</td>
<td>0.804</td>
</tr>
<tr>
<td>MAGNA++ (E+N)</td>
<td>0.786</td>
<td>0.667</td>
<td>0.741</td>
<td>0.878</td>
</tr>
<tr>
<td>MAGNA++ (N)</td>
<td>0.990</td>
<td>0.933</td>
<td>0.951</td>
<td>0.996</td>
</tr>
</tbody>
</table>

Table S4: Network discrimination performance of DynaMAGNA++ and MAGNA++, while optimizing both edge and node conservation (E+N), edge conservation alone (E), and node conservation alone (N), for social synthetic networks, with respect to the area under the precision-recall curve (AUPR), F-score at which precision and recall cross and are thus equal (F-score<sub>cross</sub>), maximum F-score (F-score<sub>max</sub>), and the area under the ROC curve (AUROC).
Supplementary Figures

Figure S1: The number of nodes vs. the number of edges for each snapshot, of each of the synthetic dynamic network instances produced by the biologically inspired network models. For each dynamic network, the number of nodes and edges grow over time. Five random instances are generated for each model. The model parameters are discussed in Section 3.1 in the main paper. The number of snapshots in each network is shown in Supplementary Table S1.
Figure S2: Degree distribution of the flattened network of each of the synthetic dynamic network instances produced by the biologically inspired network models. Five random instances are generated for each each model. The model parameters are discussed in Section 3.1 in the main paper.
Figure S3: The number of nodes vs. the number of edges for each snapshot, of each of the synthetic dynamic network instances produced by the social network evolution models. For each dynamic network, the number of nodes and edges grow over time. Five random instances are generated for each each model. The model parameters are discussed in Supplementary Section S1.2. The number of snapshots in each network is shown in Supplementary Table S2.

Figure S4: Degree distribution of the flattened network of each of the synthetic dynamic network instances produced by the social network evolution models. Five random instances are generated for each each model. The model parameters are discussed in Supplementary Section S1.2.
Figure S5: Network discrimination performance of DynaMAGNA++ and MAGNA++, while optimizing both edge and node conservation (E+N), edge conservation alone (E), and node conservation alone (N), for biological synthetic networks with respect to (a) F-score, (b) precision-recall curve, and (c) ROC curve.
Figure S6: Network discrimination performance of DynaMAGNA++ and MAGNA++, while optimizing both edge and node conservation (E+N), edge conservation alone (E), and node conservation alone (N), for social synthetic networks with respect to (a) F-score, (b) precision-recall curve, and (c) ROC curve.
Figure S7: Alignment quality of DynaMAGNA++ and MAGNA++ as a function of noise level when aligning the original Grevy’s zebra network to randomized (noisy) versions of the original network. In this figure, the randomization is as strict as possible, as it conserves all structure of the flattened version of the original dynamic network and only randomly “shuffles” the given percentage (noise level) of its event time stamps. Alignment quality is shown with respect to (a) each method’s objective function, and (b) node correctness. Red points in panel (a) shows the quality of perfect alignments, with respect to DynaMAGNA++’s objective function.

Figure S8: Alignment quality of DynaMAGNA++ and MAGNA++ as a function of noise level when aligning the original onager network to randomized (noisy) versions of the original network. In this figure, the randomization is as strict as possible, as it conserves all structure of the flattened version of the original dynamic network and only randomly “shuffles” the given percentage (noise level) of its event time stamps. Alignment quality is shown with respect to (a) each method’s objective function, and (b) node correctness. Red points in panel (a) shows the quality of perfect alignments, with respect to DynaMAGNA++’s objective function.
Figure S9: Alignment quality of DynaMAGNA++ and MAGNA++ as a function of noise level when aligning the original Grevy’s zebra network to randomized (noisy) versions of the original network. In this figure, the randomization used to create the noisy networks does not conserve the structure of the flattened version of the original dynamic network. Alignment quality is shown with respect to (a) each method’s objective function, and (b) node correctness. Red points in panel (a) shows the quality of perfect alignments, with respect to DynaMAGNA++’s objective function.

Figure S10: Alignment quality of DynaMAGNA++ and MAGNA++ as a function of noise level when aligning the original yeast network to randomized (noisy) versions of the original network. In this figure, the randomization is as strict as possible, as it conserves all structure of the flattened version of the original dynamic network and only randomly “shuffles” the given percentage (noise level) of its event time stamps. Alignment quality is shown with respect to (a) each method’s objective function, and (b) node correctness. Red points in panel (a) shows the quality of perfect alignments, with respect to DynaMAGNA++’s objective function.
Figure S11: Alignment quality of DynaMAGNA++ and MAGNA++ as a function of noise level when aligning the original Enron network to randomized (noisy) versions of the original network. In this figure, the randomization is as strict as possible, as it conserves all structure of the flattened version of the original dynamic network and only randomly “shuffles” the given percentage (noise level) of its event time stamps. Alignment quality is shown with respect to (a) each method’s objective function, and (b) node correctness. Red points in panel (a) shows the quality of perfect alignments, with respect to DynaMAGNA++’s objective function.

Funding
This work was supported by the Air Force Office of Scientific Research (AFOSR) [YIP FA9550-16-1-0147], the National Science Foundation (NSF) [CCF-1319469], and the Notre Dame’s Center for Research Computing.

References

