



Multiple Graph Alignment for the Structural Analysis of Protein Active Sites



Three-dimensional protein structures modeled by graphs

- Inexact graph matching
 - Consensus graphs characterizing a protein family are allowed to have varying sizes
 - size of consensus graph depends on size of the graph alignment
 - Weakly conserved patterns that are not necessarily found in each member of a family can also be identified
 - Graphs describing a binding pocket are dense (e.g. 85 nodes, 20% of all pairs of nodes are connected)
- How to align graphs?



Graph representation

$\mathcal{G} = \{G_1(V_1, E_1), \dots, G_n(V_n, E_n)\}$ a set of connected, node-labeled, edge-weighted graphs, each representing a protein binding pocket

$v \in V_i \quad l(v) \in \{1, 2, 3, 4, 5\}$

Nodes: predefined pseudo-centers

Node labels: H-bond donor, acceptor, mixed donor/acceptor, hydrophobic aliphatic, aromatic

Edge and edge labels: Euclidean distance below 12.0 Å
Each edge is labeled with the respective distance



Edit distances

Three types of edit operations distinguishing $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$

1. Insertion or deletion of a node $v \hat{\in} V_1$ (“InDel”)
2. Change of the label $l(v_1)$ of a node $v \hat{\in} V_1$ (“Node Mismatch”)
3. Change of the weight $w(e_1)$ of an edge $e \hat{\in} E_1$ (“Edge Mismatch”)



Graph alignment

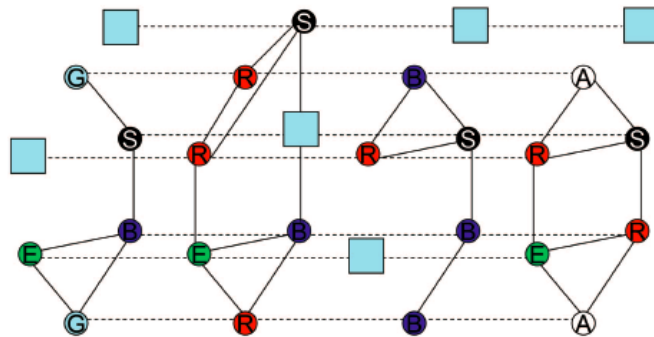


Fig. 1. An alignment of four similar but not identical graphs. The node labels are indicated by the letters assigned to the nodes (shown as circles). The edge labels are omitted for simplification. The assignments among the different graph nodes are indicated by the dashed lines. Large boxes in gray represent dummy nodes that have been introduced in the alignment to represent “missing” nodes.

```
. GSBGE .  
SR . BRER  
. BSBB . R  
. ASRAER
```

Fig. 2. Textual representation of the graph alignment in Fig. 1. Each row contains the labels of the nodes of one of the graphs. The first row represents the leftmost graph, the last row the rightmost graph. The columns represent nodes that are assigned to each other. Dummy nodes $\perp$ are represented as dots. A major difference compared to sequence alignments is that the order of the columns is arbitrary (i.e., it is possible to swap complete columns without affecting the meaning of the alignment). From this representation, it is obvious that columns 3, 4, 6, and 7 contain centers which are highly conserved and in which graph-conserved nodes are missing or are labeled differently.



Graph alignment

Introduce dummy nodes $\wedge$ analogous to “gaps”

Let $\mathcal{G} = \{G_1(V_1, E_1), \dots, G_n(V_n, E_n)\}$ be a set of graphs

$\mathcal{A} \subseteq (V_1 \cup \{\wedge\}) \times \dots \times (V_n \cup \{\wedge\})$ is an alignment of the graphs if and only if

1. for all $i = 1, \dots, n$ and for each $v \in V_n$ there exists exactly one $a = (a_1, \dots, a_n) \in \mathcal{A}$ such that $v = a_i$ (i.e. each node of each graph occurs exactly once in the alignment)
2. for each $a = (a_1, \dots, a_n) \in \mathcal{A}$ there exists at least one $1 \leq i \leq n$ such that $a_i \neq \wedge$ (i.e. each tuple of the alignment contains at least one nondummy node)



Scoring alignment

node alignment score

$$\underline{nodeScore} \begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix} = \sum_{j,k=1, i < j}^n \begin{cases} \underline{nScore}_{match} & l(a_j^i) = l(a_k^i) \\ \underline{nScore}_{mismatch} & l(a_j^i) \neq l(a_k^i) \\ \underline{nScore}_{dummy} & a_x^i = \perp, a_y^i \neq \perp; x, y = j, k \end{cases}$$



Scoring alignment

edge alignment score

$$\underline{edgeScore} \left(\begin{pmatrix} a_1^i \\ \vdots \\ a_n^i \end{pmatrix}, \begin{pmatrix} a_1^j \\ \vdots \\ a_n^j \end{pmatrix} \right) = \sum_{k,l=1, k < l}^n \begin{cases} \underline{eScore}_{match} & |w(a_k^i, a_k^j) - w(a_l^i, a_l^j)| \leq \epsilon \\ \underline{eScore}_{mismatch} & |w(a_k^i, a_k^j) - w(a_l^i, a_l^j)| > \epsilon \\ \underline{eScore}_{dummy} & (a_x^i, a_x^j) \in E_x, (a_y^i, a_y^j) \notin E_y; x, y = k, l. \end{cases}$$



Scoring alignment

Sum of node score and edge score

$$s(\mathcal{A}) = \sum_{i=1}^m \underline{nodeScore}(a^i) + \sum_{i,j=1, i < j}^m \underline{edgeScore}(a^i, a^j),$$



Pairwise Graph Alignment

Algorithm 1: Algorithm for the calculation of pairwise graph alignments using a greedy strategy.

Input: Graphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$

Output: Alignment $\mathcal{A}$ for the graphs G_1, G_2

//construct association graph $A(V, E)$

```
foreach  $v_1 \in V_1$  do
  foreach  $v_2 \in V_2$  do
    if  $l(v_1) = l(v_2)$  then
      insert node  $(v_1, v_2)$  into  $V$ 
    end
  end
end
foreach  $v = (v_1, v_2) \in V$  do
  foreach  $v' = (v'_1, v'_2) \in V$  do
    if  $|w(v_1, v'_1) - w(v_2, v'_2)| \leq \epsilon$  then
      insert edge  $(v, v')$  into  $E$ 
    end
  end
end
end
```

- Association graph A contains a node for each pair of nodes from the two input graphs G_1 and G_2 that have the same label
- Two nodes in A are connected if the corresponding nodes in G_1 and G_2 are connected by edges with compatible labels or they are connected in neither graphs
- A maximal clique in A corresponds to a maximal number of trivial common subgraphs that can be merged into one large nontrivial common subgraph of G_1 and G_2

```

// find seed matches and extend greedily
for up to  $n$  cliques  $C$  in  $A$  do
  insert each  $(v_1, v_2) \in C$  into  $\mathcal{A}$ 
  foreach  $v_1 \in V_1$  with  $v_1 \notin C$  do
    collect in  $W_1$  edge weights to all nodes contained in  $C$ 
    adjacent to  $v_1$ 
    foreach  $v_2 \in V_2$  with  $v_2 \notin C$  do
      collect in  $W_2$  edge weights to all nodes contained in  $C$ 
      adjacent to  $v_2$ 
      if  $l(v_1) = l(v_2)$  then
         $score := \underline{nScore}_{match}$ 
      else
         $score := \underline{nScore}_{mismatch}$ 
      end
      foreach  $w_1 \in W_1$  do
        look up corresponding  $w_2 \in W_2$ 
        if  $|w_1 - w_2| \leq \epsilon$  then
           $score := score + \underline{eScore}_{match}$ 
        else
           $score := score + \underline{eScore}_{mismatch}$ 
        end
      end
    end
  end
  select  $v'_2$  with maximal  $score$ 
  if  $score > \underline{nScore}_{dummy}$  then
    insert  $(v_1, v'_2)$  into  $\mathcal{A}$ 
  else
    insert  $(v_1, \perp)$  into  $\mathcal{A}$ 
  end
end
end
select and return  $\mathcal{A}$  with maximum overall score

```

- Use greedy approach: start with the large common subgraph (“seed”)
- Find a node v_1 in G_1 that is not part of the common subgraph and has the highest number of edges to those nodes that are already in the common subgraph.
- Find a match partner v'_2 in G_2 that is not part of the common subgraph and that result in best match score (both node and edge match scores)
- Map v_1 to v'_2 in the solution



Multiple Graph Alignment

Algorithm 2: Algorithm for the star-like merging of graph alignments using a fixed reference graph G_i as center.

Input: Alignments $\mathcal{A}_1, \mathcal{A}_2$, both containing pivot graph $G_i(V_i, E_i)$

Output: Merged Alignment from $\mathcal{A}_1$ and $\mathcal{A}_2$

```
foreach  $a = (a.1, \dots, a.n) \in \mathcal{A}_2$  do
  if  $\exists a' = (a'.1, \dots, a'.m) \in \mathcal{A}_1$  s.t.  $a.i = a'.i$  then
    replace  $a' \in \mathcal{A}_1$  by  $a'' = (a'.1, \dots, a'.m, a.1, \dots, a.n)$ 
  else
    insert  $a'' = (\perp, \dots, \perp, a.1, \dots, a.n) \in \mathcal{A}_1$ 
  end
end
foreach  $a' = (a'.1, \dots, a'.n) \in \mathcal{A}_1$  do
  if  $\nexists a = (a.1, \dots, a.n) \in \mathcal{A}_2$  s.t.  $a.i = a'.i$  then
    insert  $a'' = (a'.1, \dots, a'.n, \perp, \dots, \perp) \in \mathcal{A}_1$ 
  end
end
return  $\mathcal{A}_1$ 
```

- Incrementally merging precalculated alignments
- One of the graph instances is used as a pivot/center, all other instances are aligned to this pivot.