ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Bled Winter Seminar
February 2024

Talk - Progressive Graph Alignment 2: Ambiguous Edges, Mutants and Heuristics

Speaker: PhD Student - Marcos Laffitte

Advisor: Prof. Dr. Peter F. Stadler

- Institut Für Informatik, Bioinformatik, Uni Leipzig
- ScaDS.AI Leipzig

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

SACHSEN

Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÄT LEIPZIG

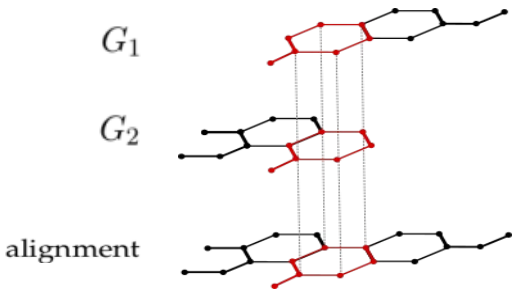
Introduction - pairwise graph alignment

The alignment of two graphs is another graph satisfying certain conditions. For a formal definition see **[1, 2]**

This alignment can be characterized as follows:

Proposition ([1, 2])

A graph G is an alignment of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, if and only if, the set of columns $\{Q \cap V_1 \neq \varnothing \text{ and } Q \cap V_2 \neq \varnothing\}$ defines a common induced subgraph of G_1 and G_2.

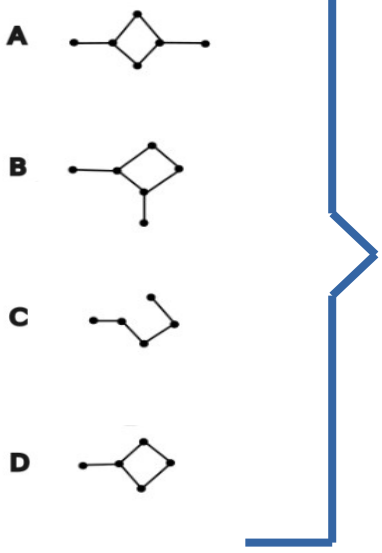


***** But in practice we would like the induced subgraph to be optimal in some way, e.g. a **maximum common induced subgraph** (MCS)

[1] Stadler Peter F. 2021. Alignments of biomolecular contact maps. Interface Focus 11. **http://doi.org/10.1098/rsfs.2020.0066**
[2] Berkemer, S.J., et. al. Compositional Properties of Alignments. Math.Comput.Sci. 15, 609–630 (2021). **https://doi.org/10.1007/s11786-020-00496-8**

Introduction - progressive graph alignment

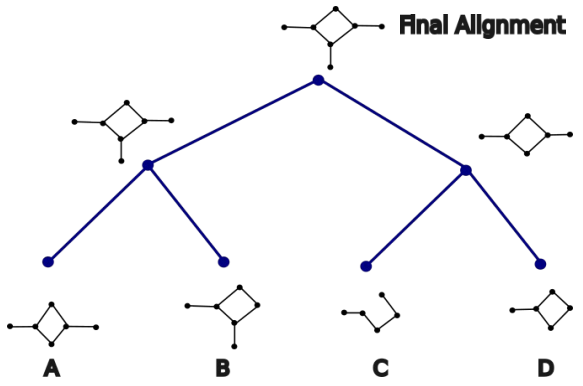
Based on pairwise alignments we can produce a **heuristic** alignment of multiple graphs ...



Similarity matrix obtained from graph-kernels, as, for example, the **Structural-Shortest-Path** kernel from python's graphkit-learn

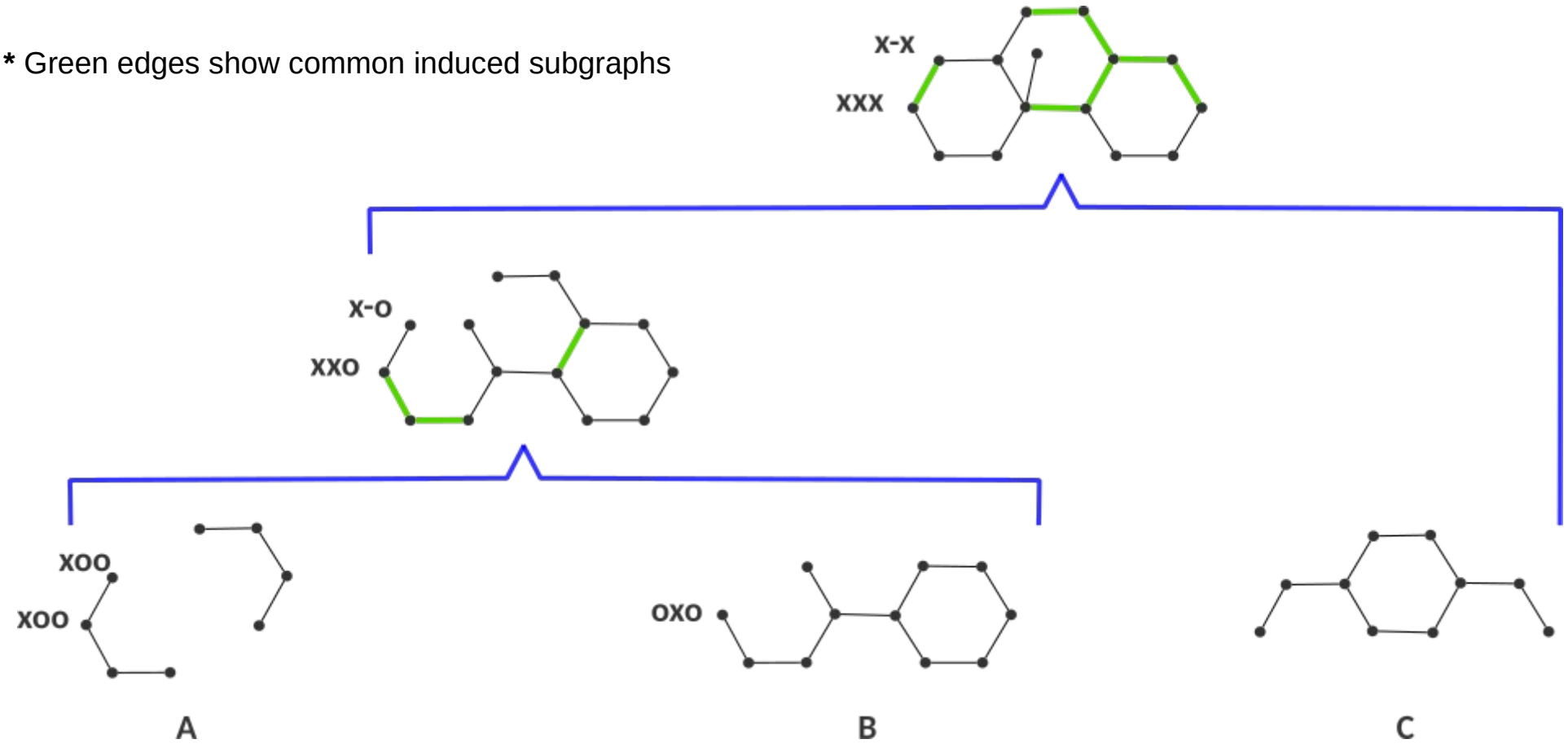
	A	B	C	D
A	-	80	30	30
B	80	-	30	30
C	30	30	-	80
D	30	30	80	-

The guide tree can be obtained with Weighted Pair Group Method with Arithmetic Mean **(WPGMA)**.

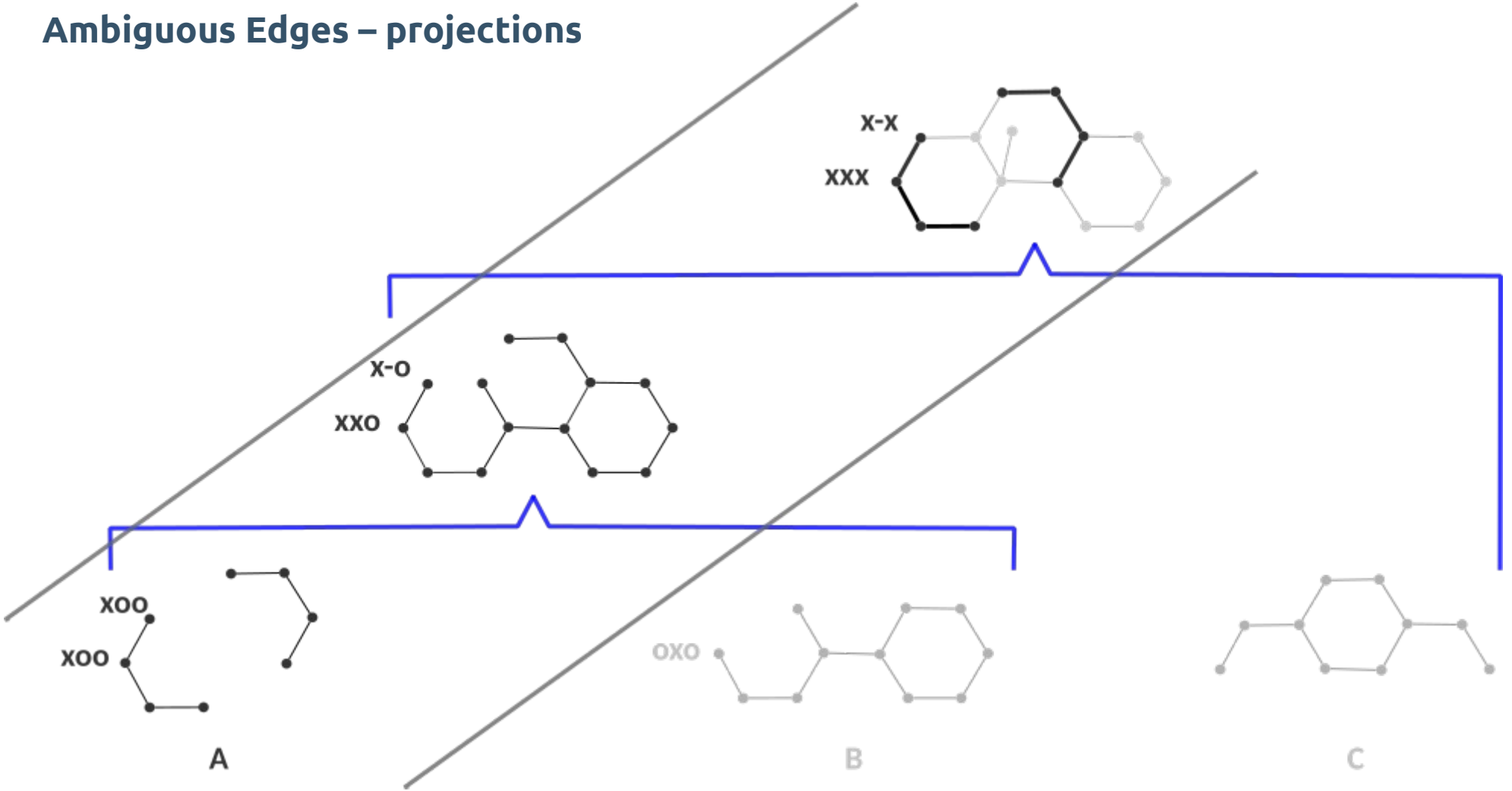


Ambiguous Edges – consider the following alignment ...

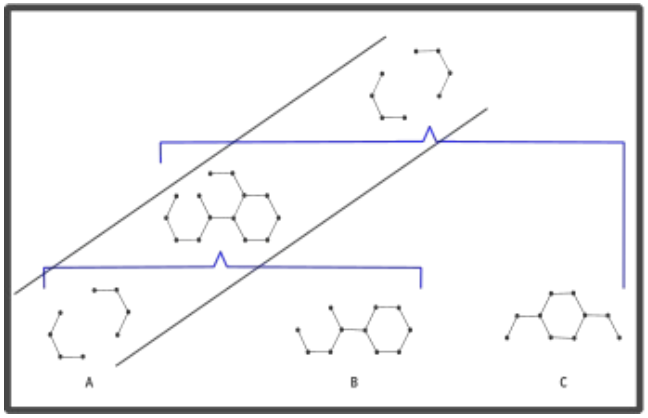
* Green edges show common induced subgraphs

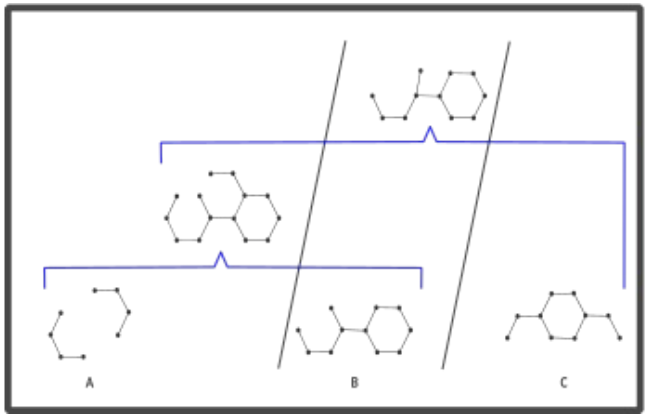


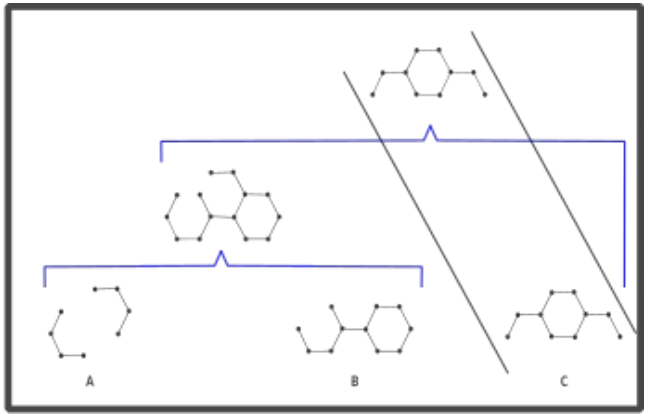
Ambiguous Edges – projections



Ambiguous Edges – valid alignments



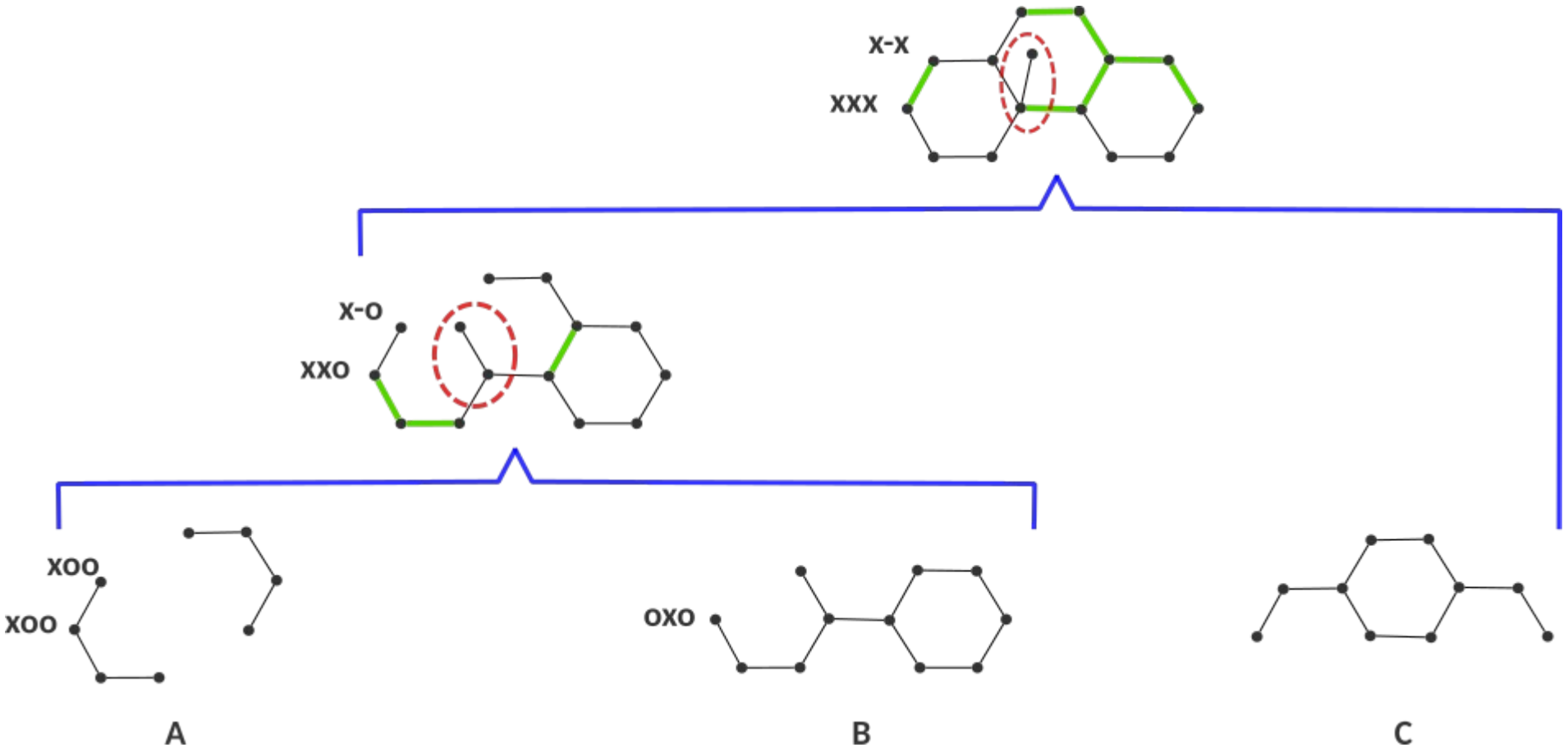




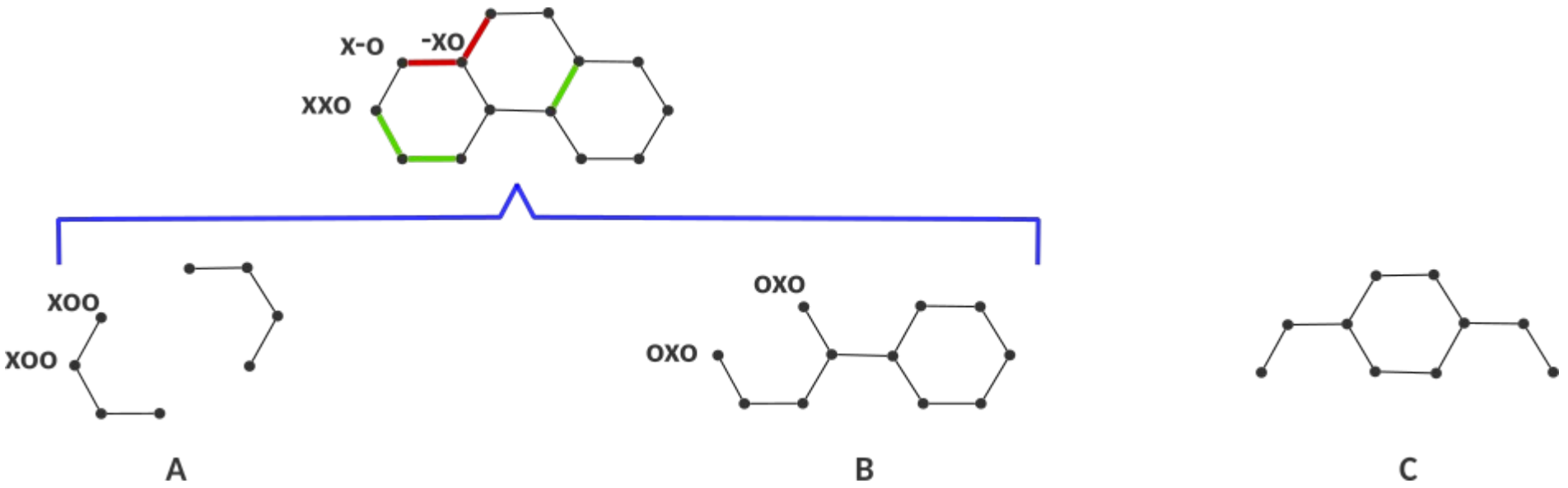
Keep vertices with vector-labels of the form:

X## #**X**# ##**X**

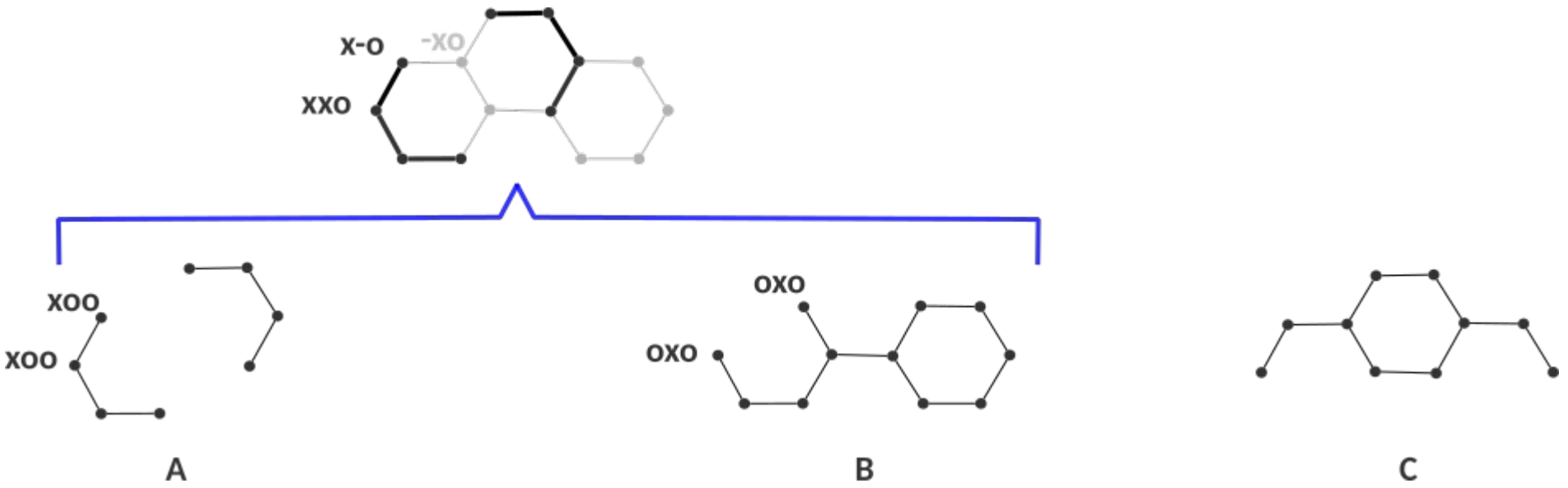
Ambiguous Edges – motivation



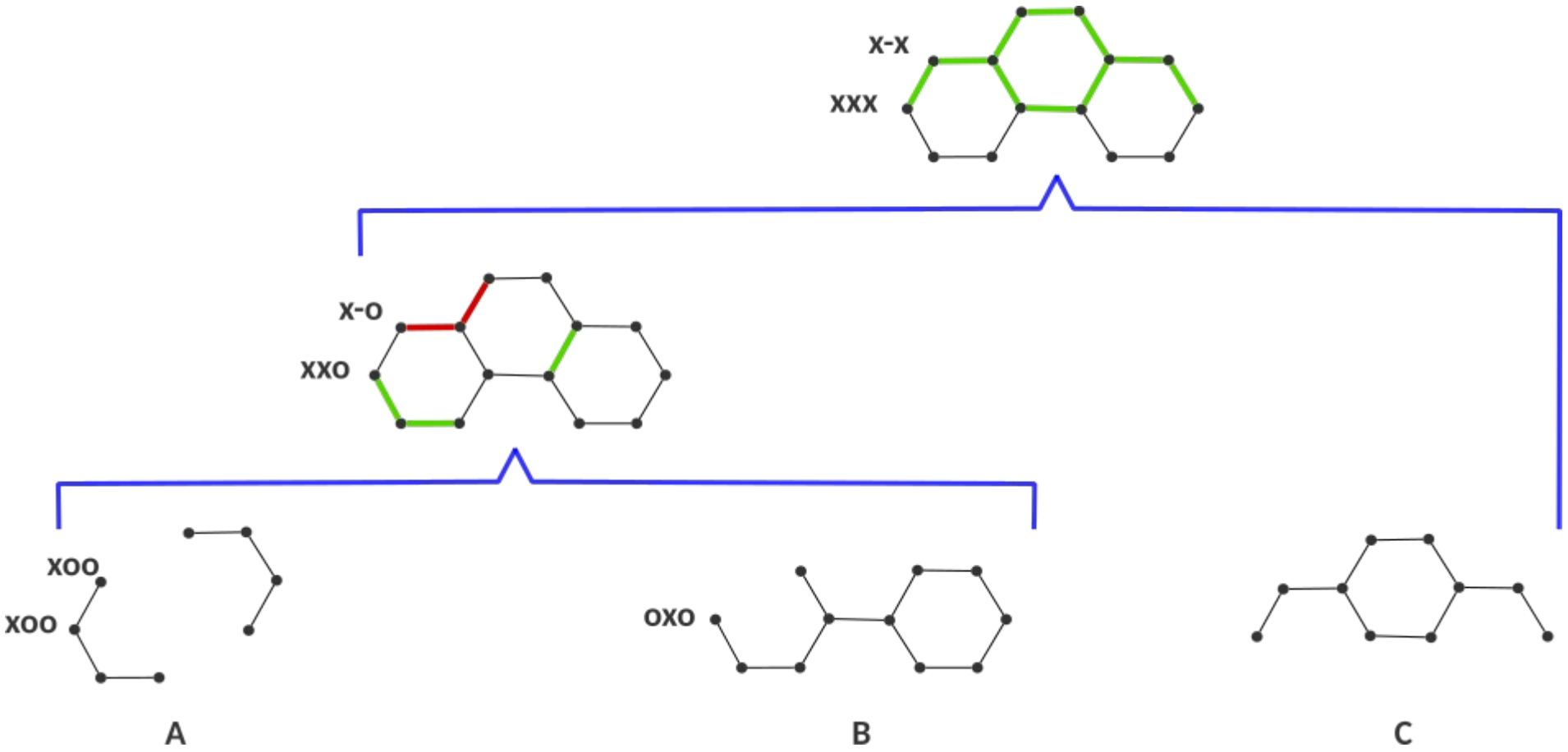
Ambiguous Edges – but what if … ?



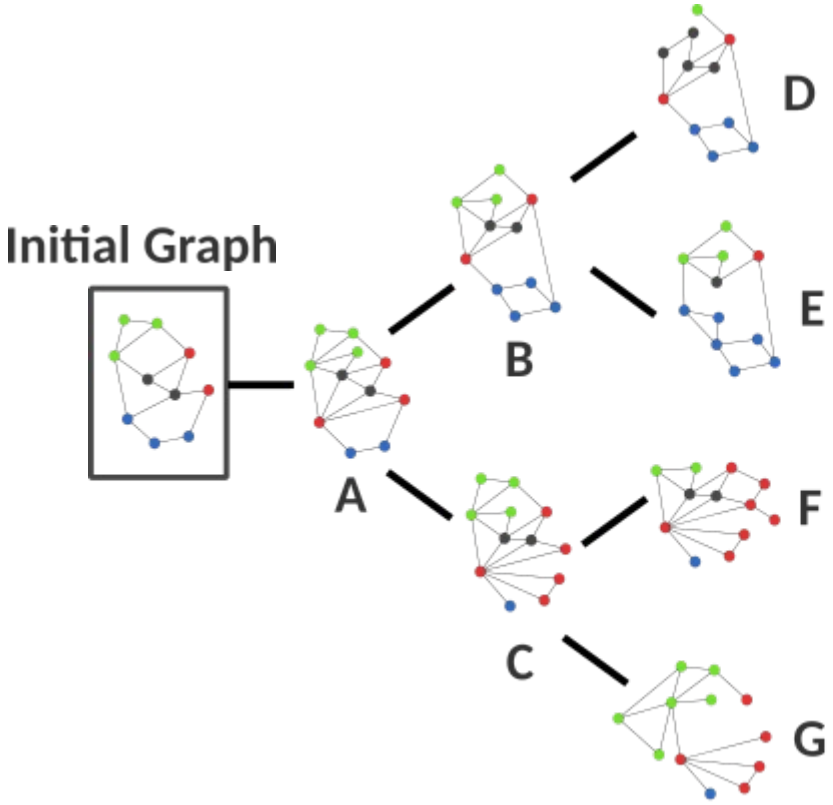
Ambiguous Edges – still a valid alignment ...

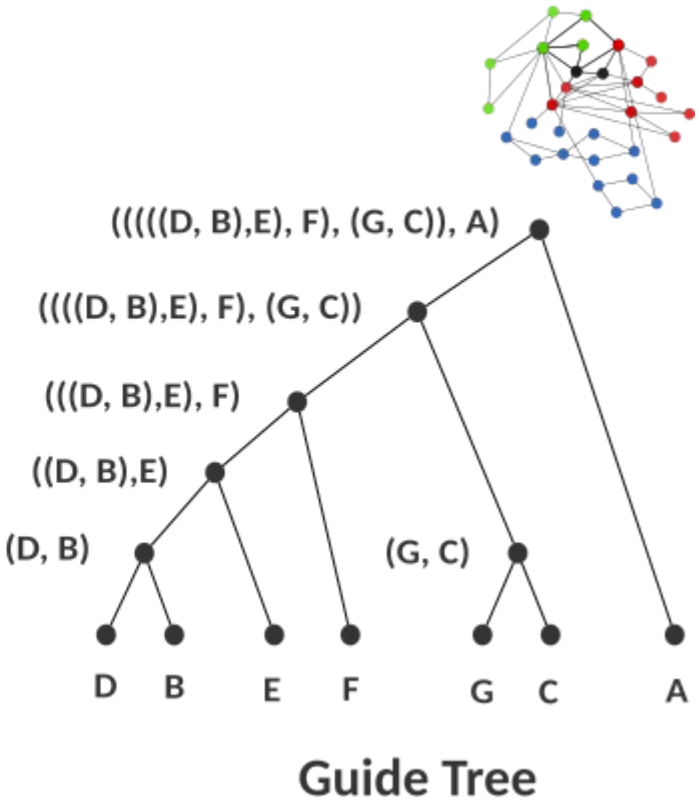


Ambiguous Edges – … and gives better results

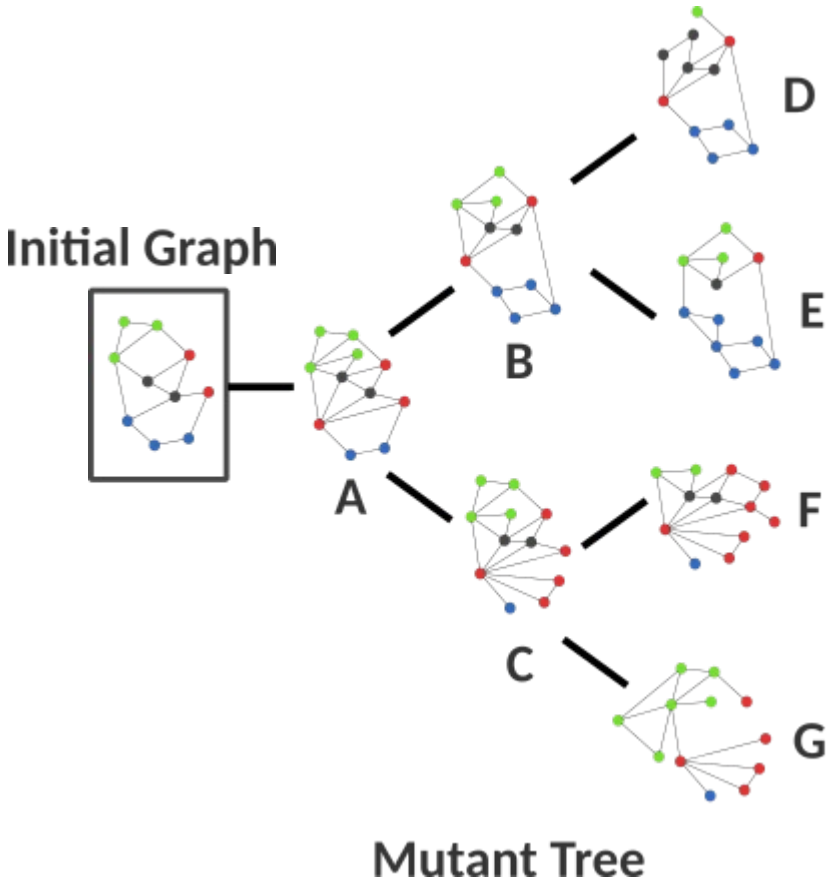


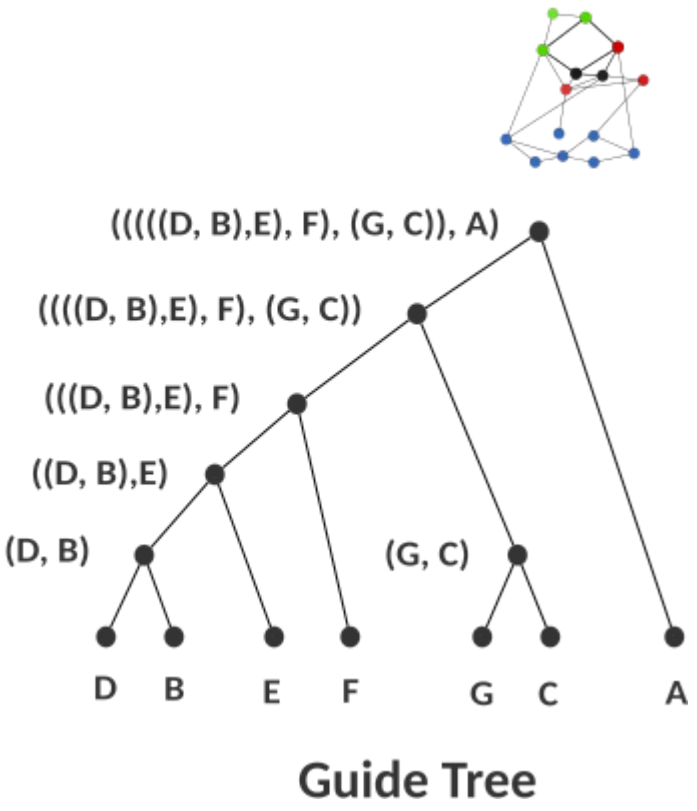
Mutants – generation and alignment of mutants



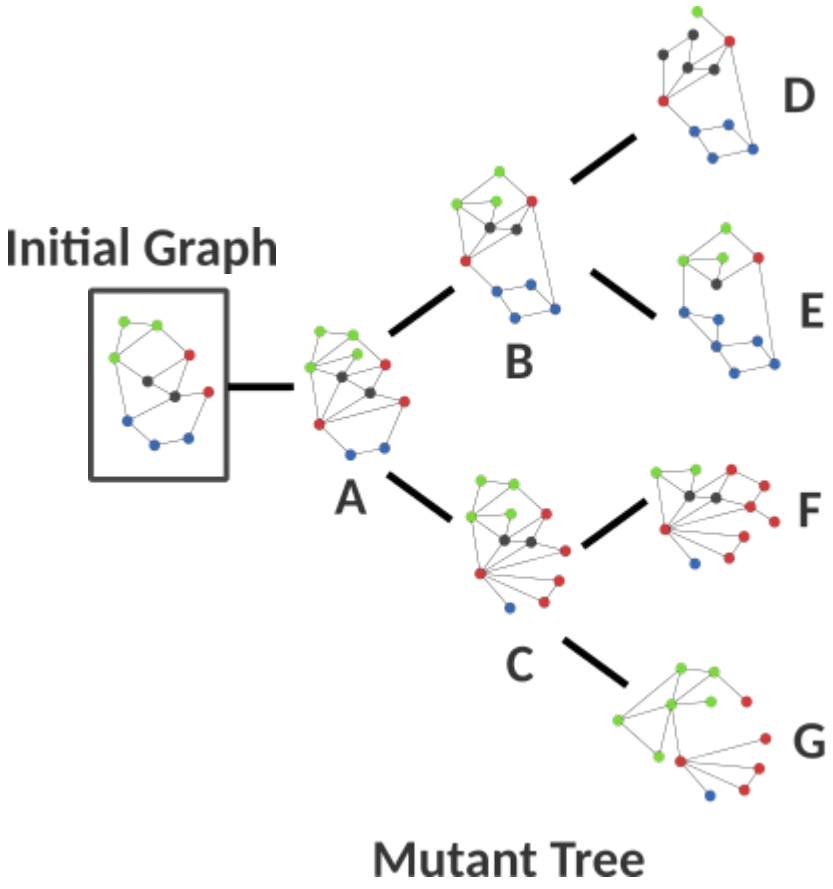


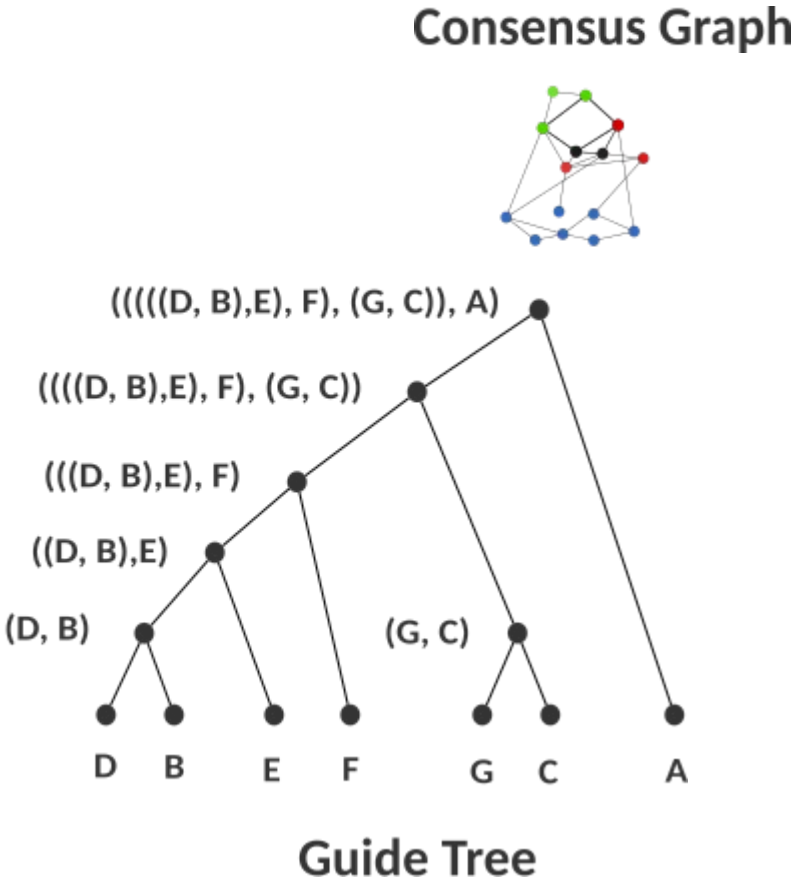
Mutants – consensus graphs





Mutants – consensus graphs

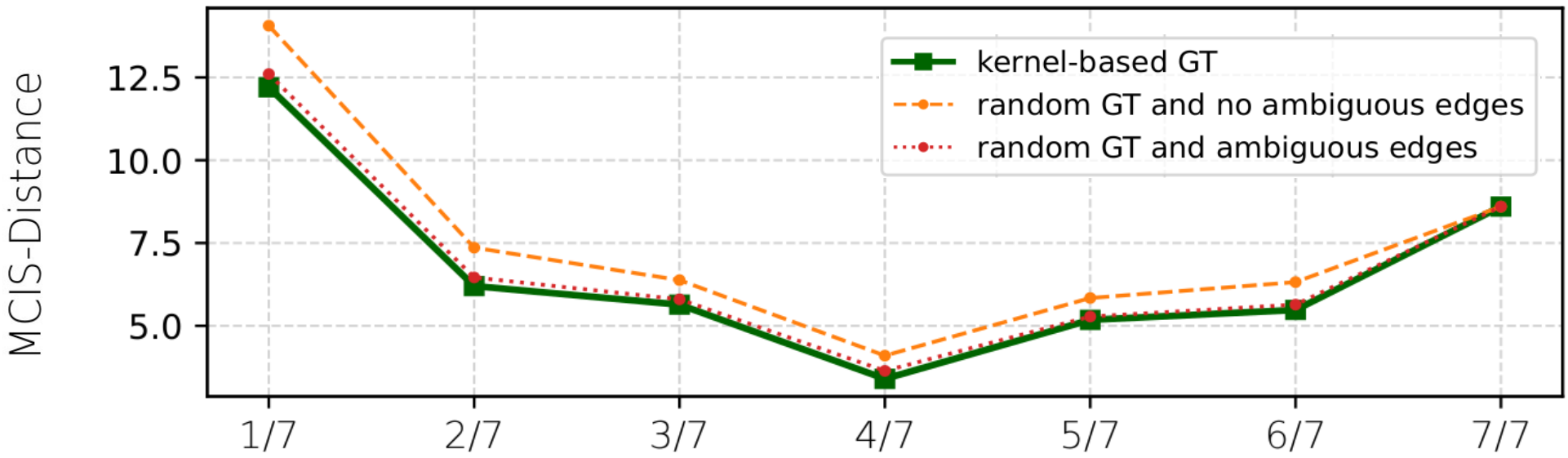




Mutants – analysis and results

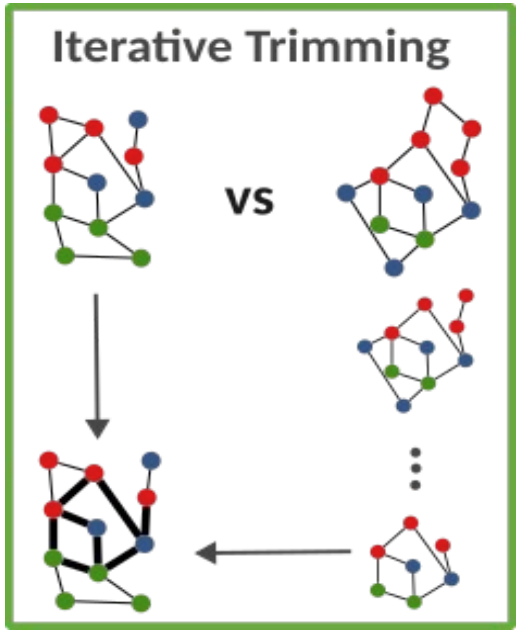
- 50 scenarios, that is, 50 initial graphs
- with 16 vertices each
- and 7 mutants produced in the (full) binary tree fashion
- by randomly removing 1-2 existing vertices and adding 1-2 vertices in each step as mutations

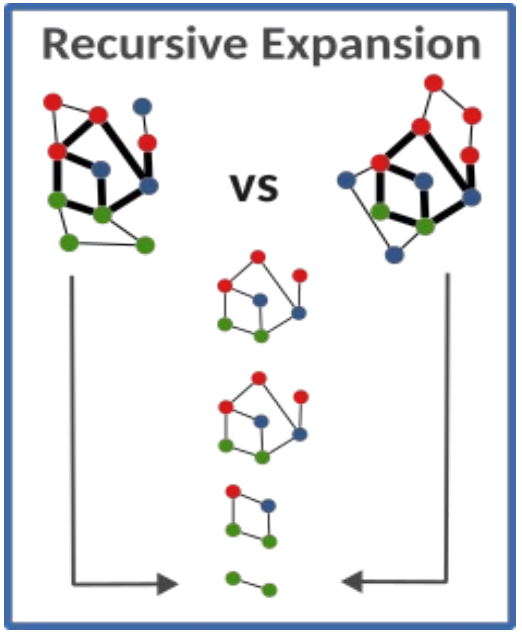
Recall the graph-edit-distance (MCS-distance) between G and H is: **$d(G, H) = |V(G)| + |V(H)| - 2|MCS(G, H)|$**



Running Times – methods for MCS search

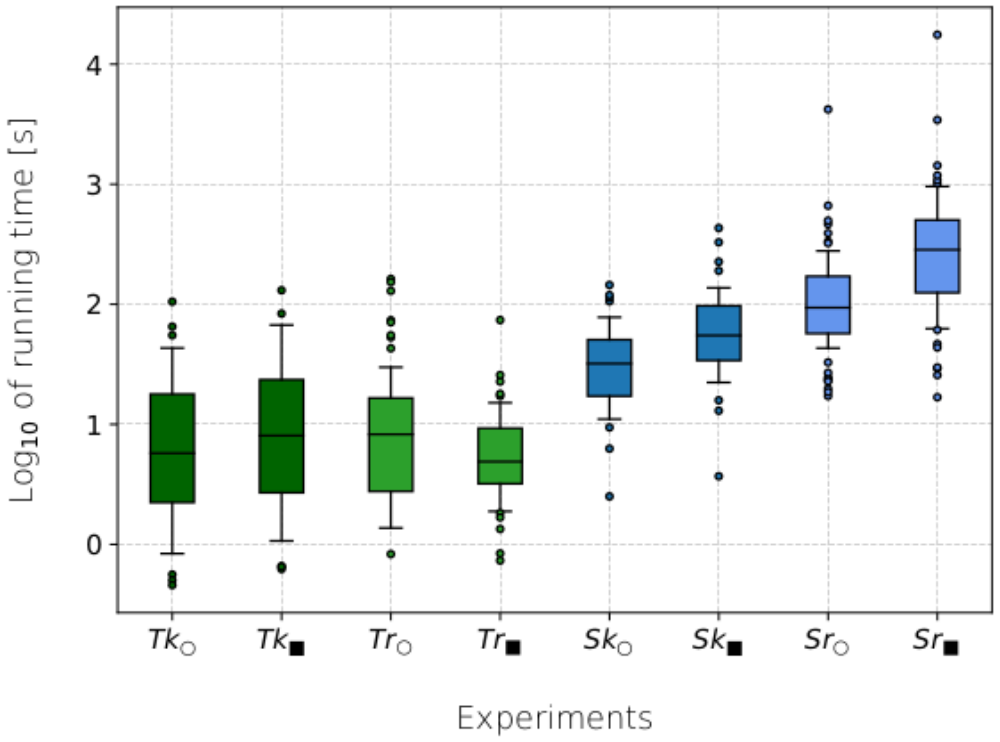
We made two MCS search methodologies based on our own implementations of variants of the VF2 algorithm **[3].**
- the iterative trimming implements VF2 for **subgraph** isomorphism
- the recursive expansion is an adaptation of the VF2 for MCS search





[3] VF2 algorithm: L. P. Cordella et al. "A (sub)graph isomorphism algorithm for matching large graphs," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pp. 1367-1372, Oct. 2004, **doi: 10.1109/TPAMI.2004.75**.

Running Times



Comparison of running times [s] of the eight experiments carried each over the 50 scenarios: T and S refer to the use of `Iterative_Trimming` and `VF2_step`, respectively. Kernel-based (k) or random (r) guide trees show a moderate but systematic advantage of a kernel-based similarity. The exclusion (○) or inclusion (■) of ambiguous edges is also compared.

Conclusion and future work

> ambiguous edges provide in practice better alignments that are able to *correct* biases introduced by random trees.

> The (slower) performance of the **Recursive Expansion** vs the good performance of the **Iterative Trimming** can be better explained by the following factors:

(1) the recursive expansion works faster for smaller **MCS** (less of half the vertices of the smaller from the 2 graphs)
***** the trimming seems to still work "well-enough" for those same cases**

(2) it has been reported that back-tracking MCS methods (as the Recursive Expansion) have problems when reaching independent sets, and alternatives have been proposed based on vertex covers **[4]**

(3) the MCS recursive expansion based on the VF2 actually looses various properties that make the VF2 efficient, like proposing smaller number of candidates for expansion and being able to discard those not having the same degree outside the MCS

> Different bounds should be implemented for the recursive expansion so that it works as a proper branch-and-bound algorithm, but these should take into account different scoring schemes and ambiguous edges. Moreover they should be really easy to compute, because they should be evaluated in every state of the search space (or seek for alternatives to reduce their repeated evaluation)

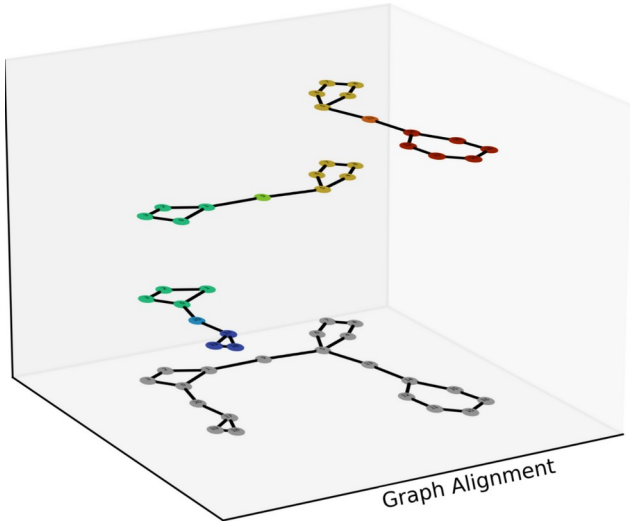
> Heuristics for MCS search need to be handled carefully, because even though they might work properly in a pairwise manner, they produce bigger alignments (were less nodes are being matched), affecting the next alignments in the progressive MGA.

> Experiments should be carried over bigger graphs, but for that we need a more efficient MCS search methodology and it should be implemented in C++

[4] Abu-Khzam et al. "The Maximum Common Subgraph Problem: Faster Solutions via Vertex Covers", 2007 IEEE/ACS International Conference on Computer Systems and Applications, Amman, Jordan, 2007, pp. 367-373, https://ieeexplore.ieee.org/document/4230982

Thank you for your attention

PROG
GRA
ALIGN
PROGRALIGN



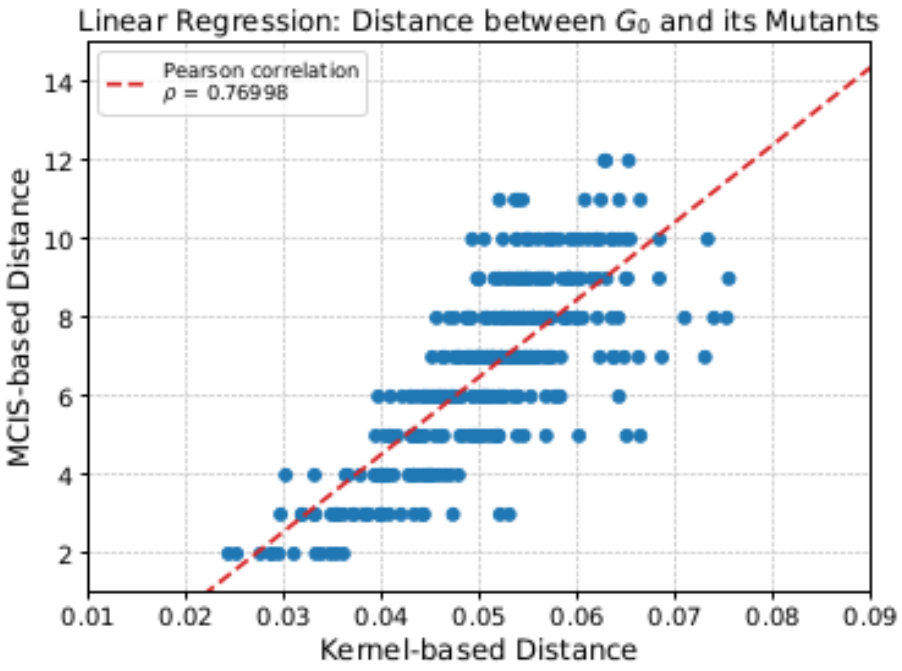
Funding Institution

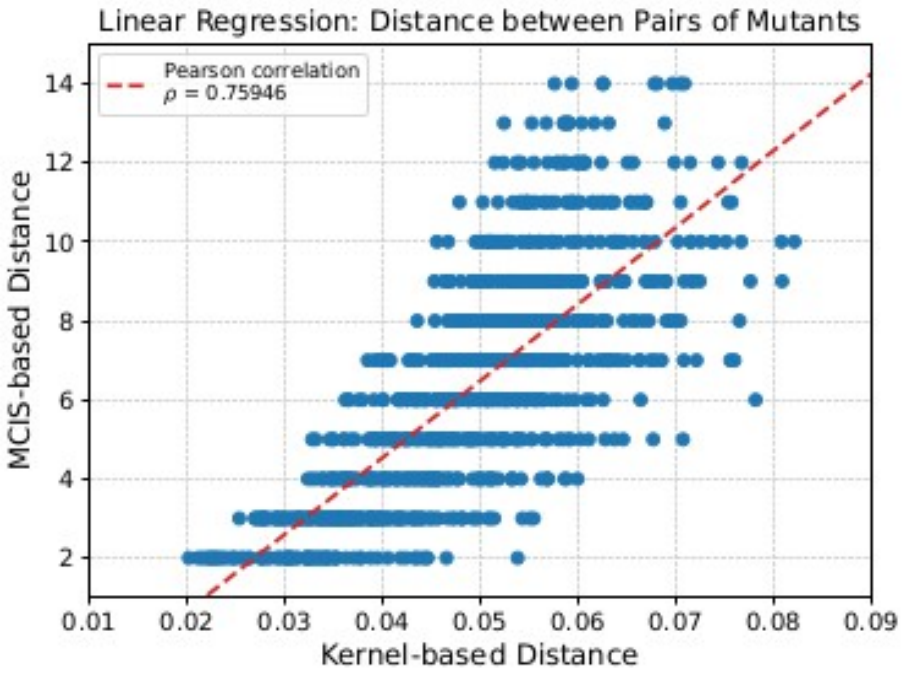
Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", **project identification number: ScaDS.AI.**

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

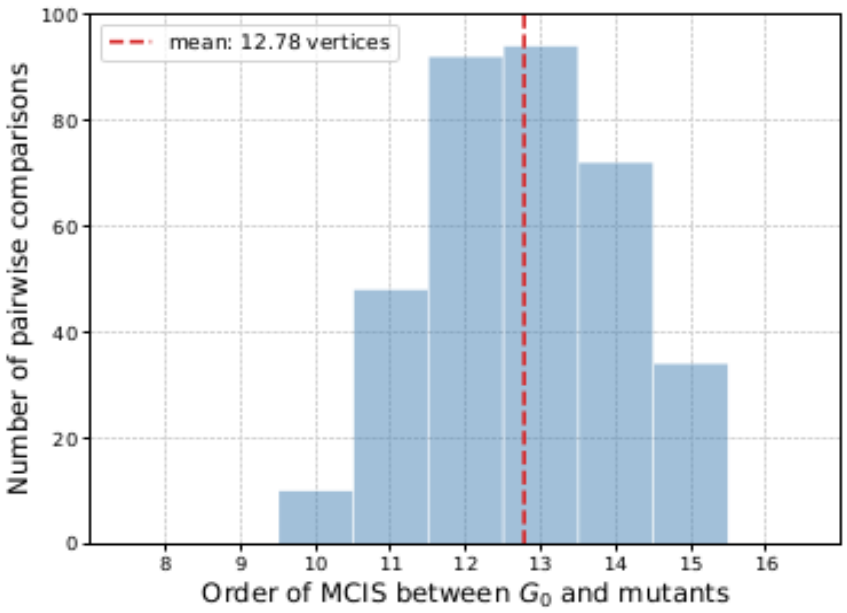
SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

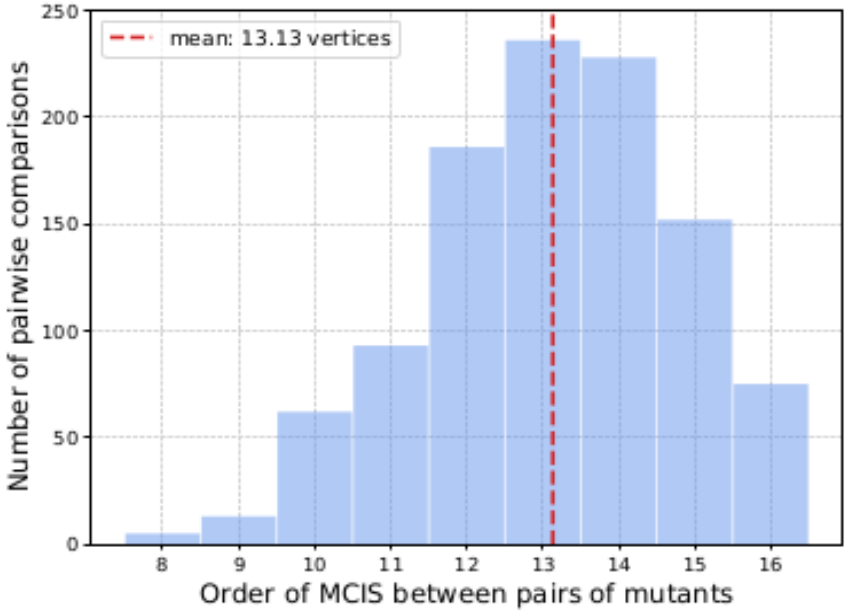
Extra slide – quality of mutants





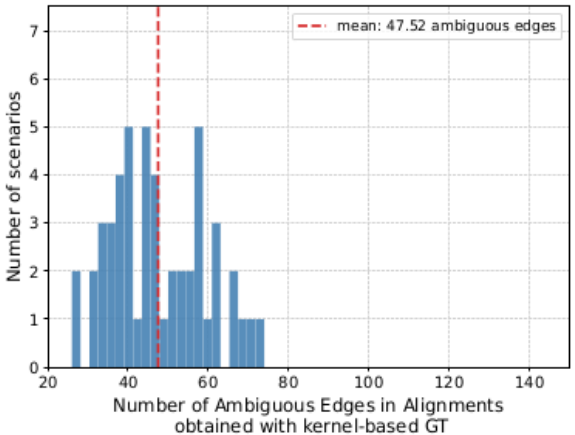
Extra slide – order of MCS's in sets of mutants



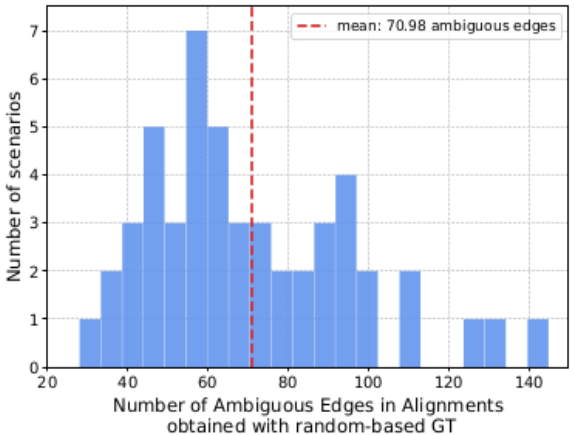


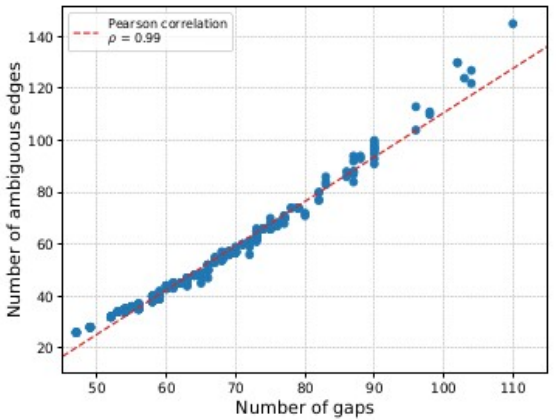
Extra slide – ambiguous edges

(a)



(b)



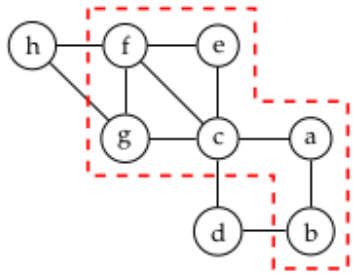


Extra slide – why the original VF2 doesn't work for MCS search

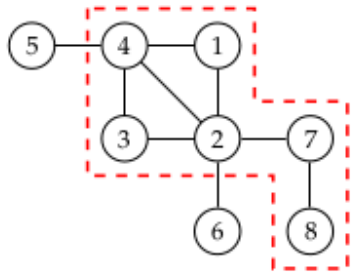
Algorithm 4: `candidate_matches_original`$(G_1, G_2, M, \preceq)$

Data: Graphs G_1, G_2 and a matching $M \subset V(G_1) \times V(G_2)$
Result: Set P of match candidates for extending M
`// initialize` P_0
$P_0 \leftarrow \emptyset$;
`// candidates are neighbors of match but not in the match`
for $(n1, n2) \in M$ **do**
  $N_1 \leftarrow$ neighbors of n_1 in G_1 not matched by M;
  $N_2 \leftarrow$ neighbors of n_2 in G_2 not matched by M;
  $P_0 \leftarrow P_0 \cup N_1 \times N_2$;
end
`// alternatively candidates are all unpaired vertices`
if $P_0 = \emptyset$ **then**
  $N_1 \leftarrow$ set of unmatched vertices in $V(G_1)$;
  $N_2 \leftarrow$ set of unmatched vertices in $V(G_2)$;
  $P_0 \leftarrow N_1 \times N_2$;
end
`// get only the` $\preceq$`-minimum of such vertices`
$m \leftarrow \min_{\preceq}\{y : (x, y) \in P_0\}$;
$P \leftarrow \{(x, y) \in P_0 : y = m\}$;
return P

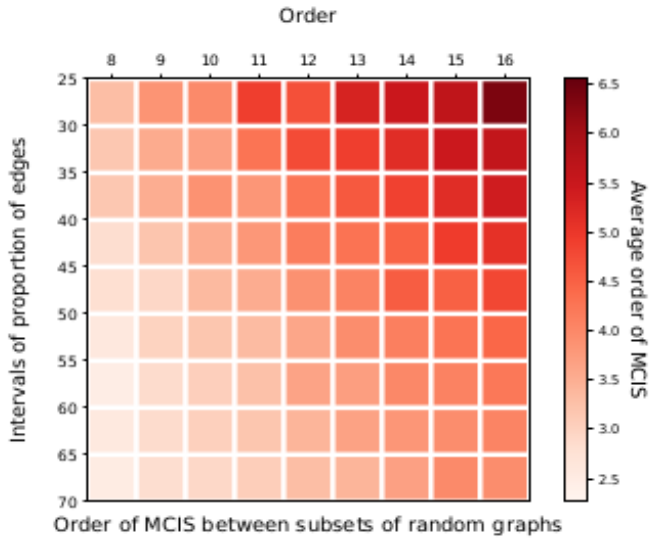
(G_1)

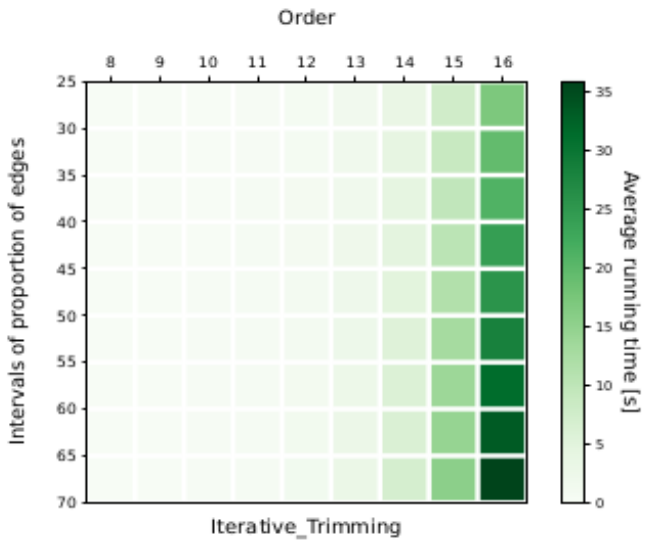


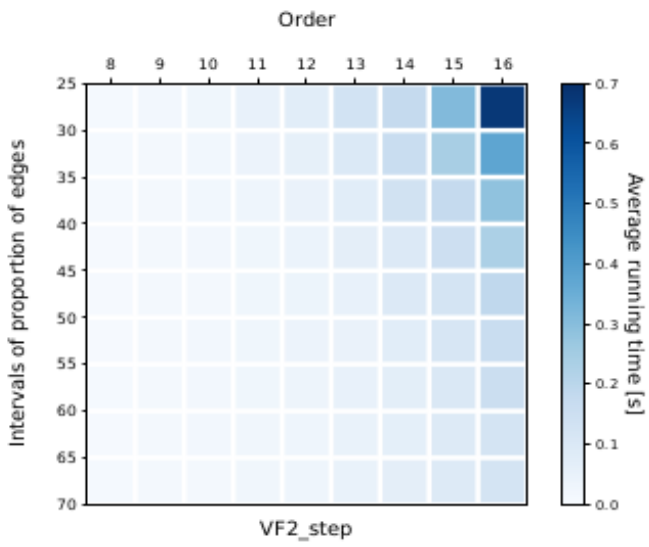
(G_2)



Extra slide – running time over sets of random graphs







Extra slide – problem with independent sets