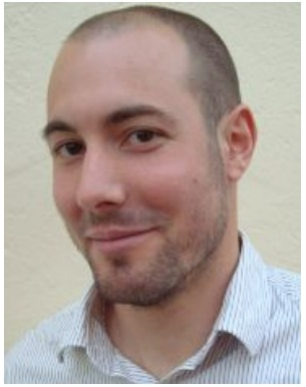
# Scalable N-Way Model Matching Using Multi-Dimensional Search Trees


Alexander Schultheiß

Paul Maximilian Bittner

Thomas Thüm



Timo Kehrer

HUMBOLDT-UNIVERSITÄT ZU BERLIN

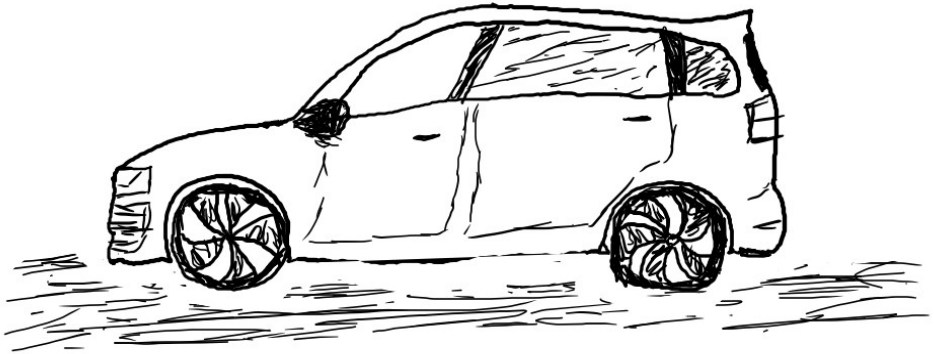
ulm university universität uulm

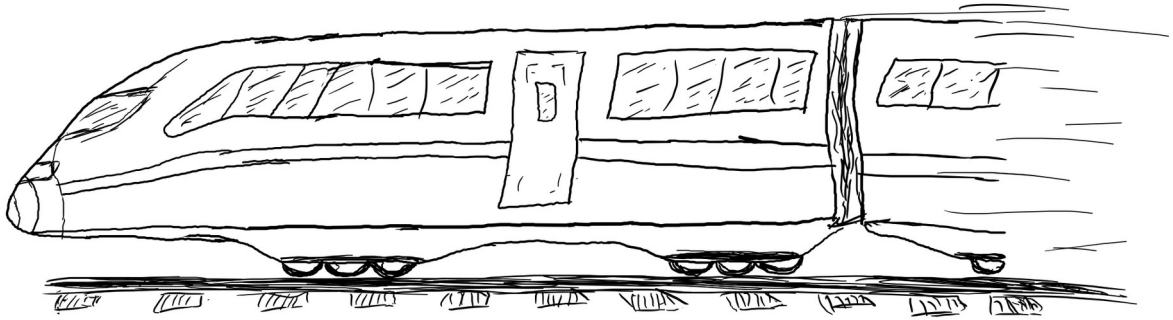
ulm university universität uulm

UNIVERSITÄT BERN

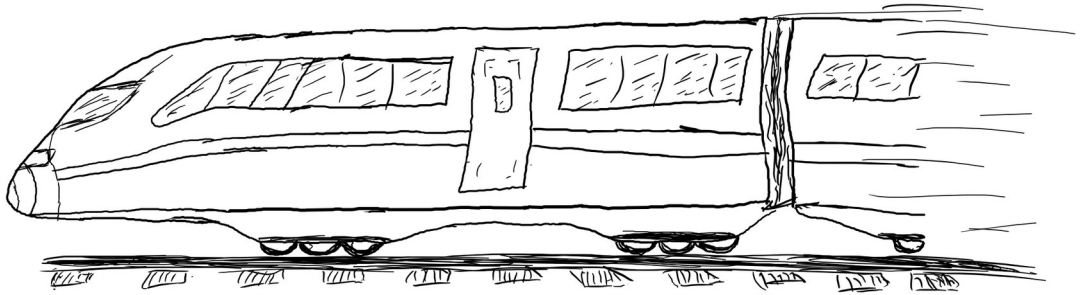
# Why do we need model matching?

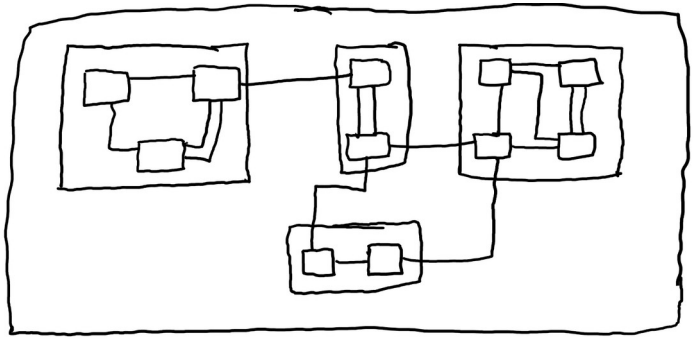
# Model-driven development is crucial in certain domains

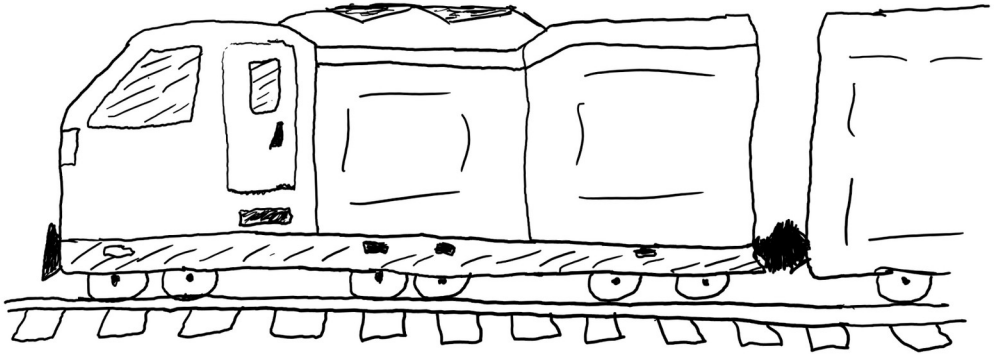


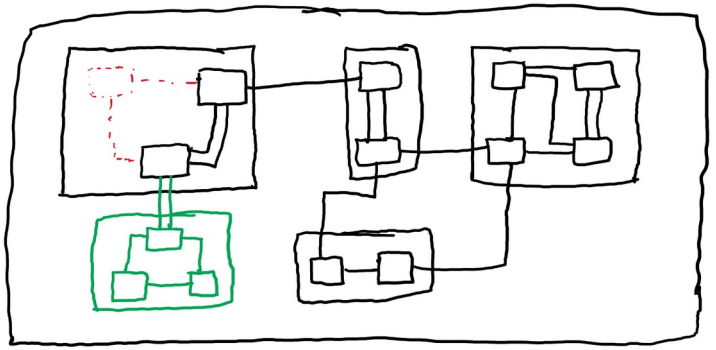


# Different variants of a model are required

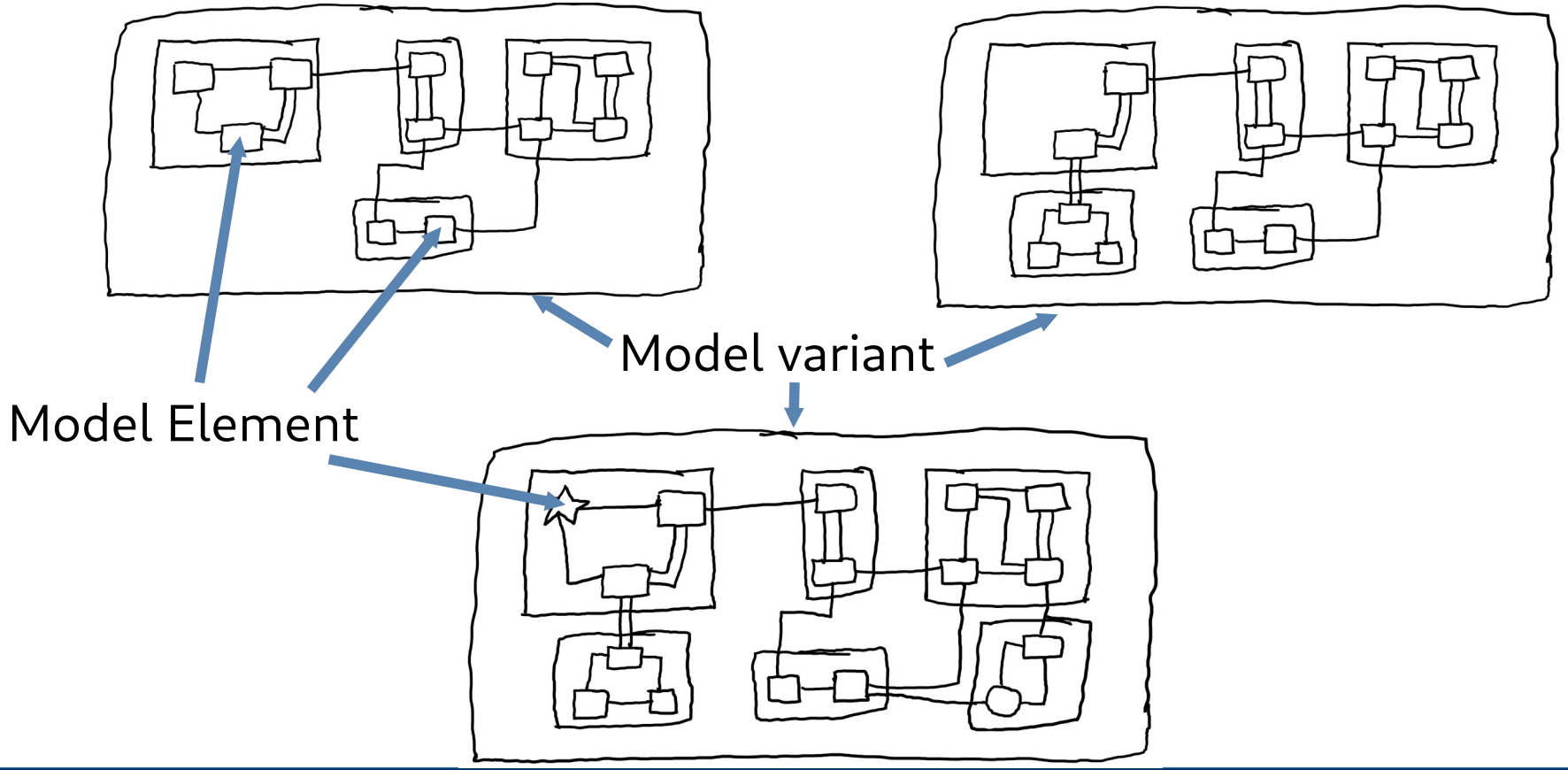




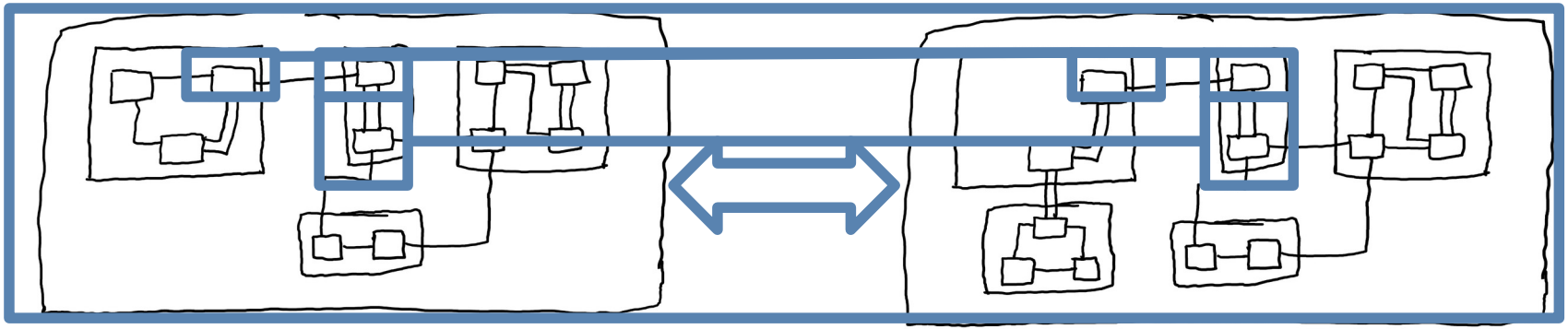


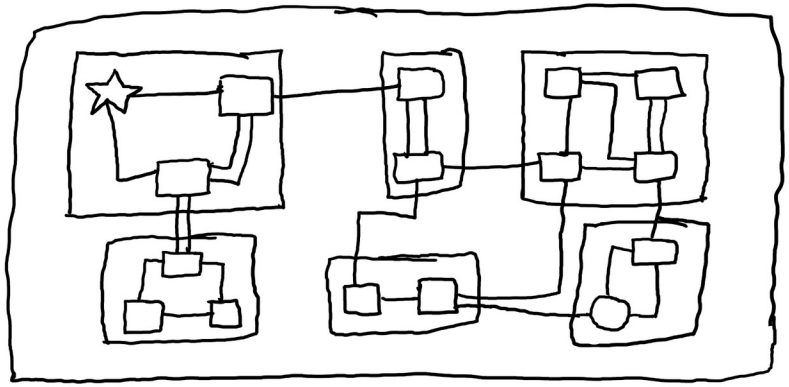


# We want to match common elements

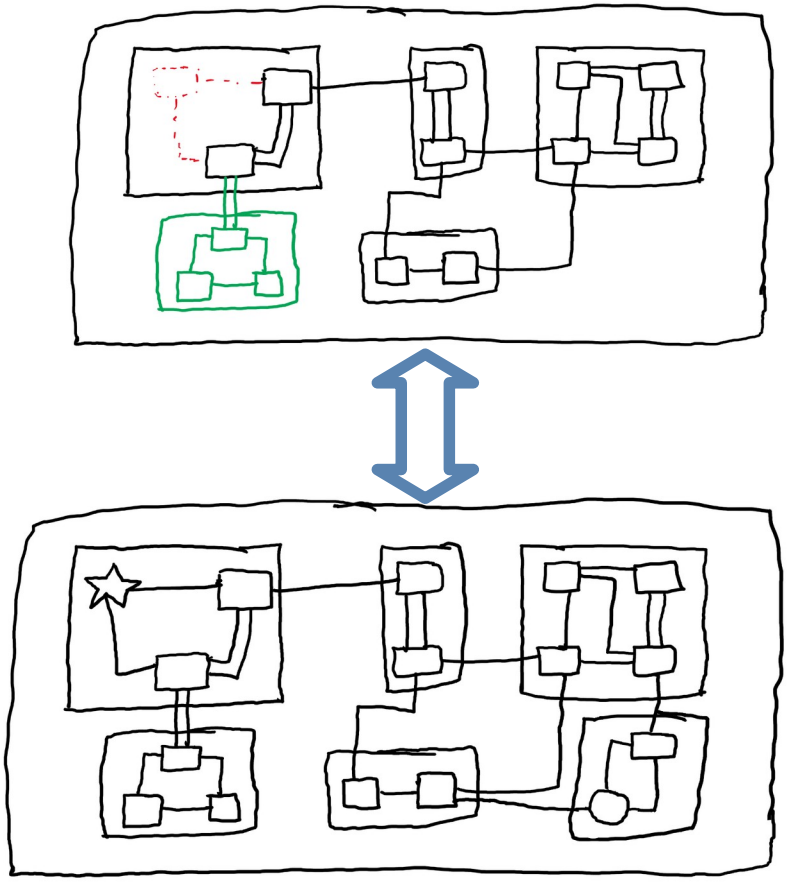


# Pairwise matching considers models one-by-one

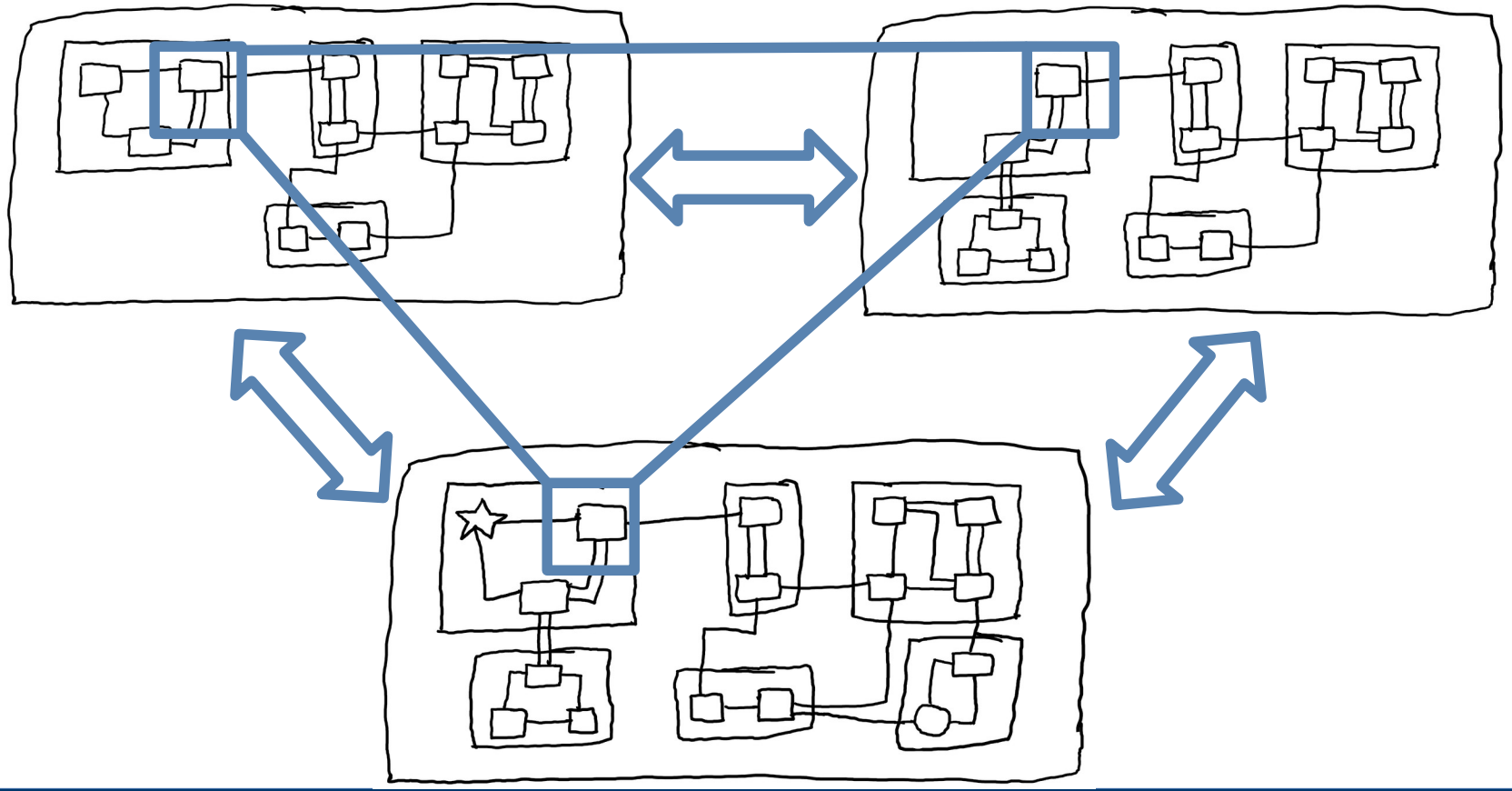




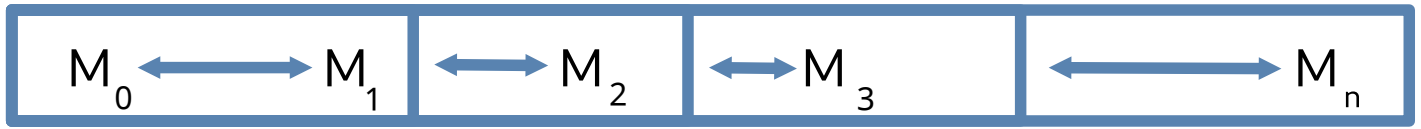
# Pairwise matching considers models one-by-one



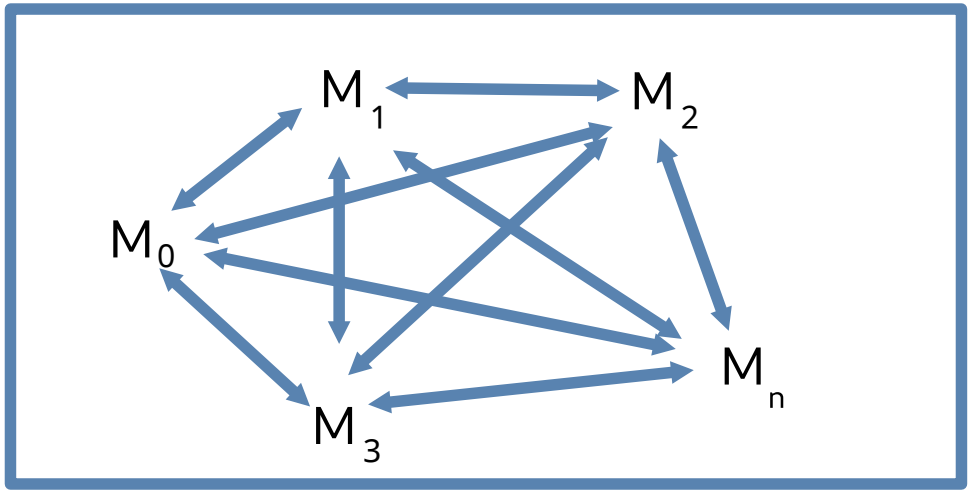
# N-way matching considers all models at once



# Pairwise and n-way in direct comparison



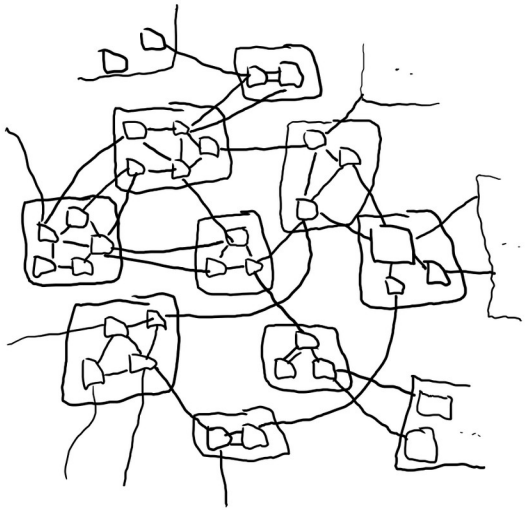

Pairwise

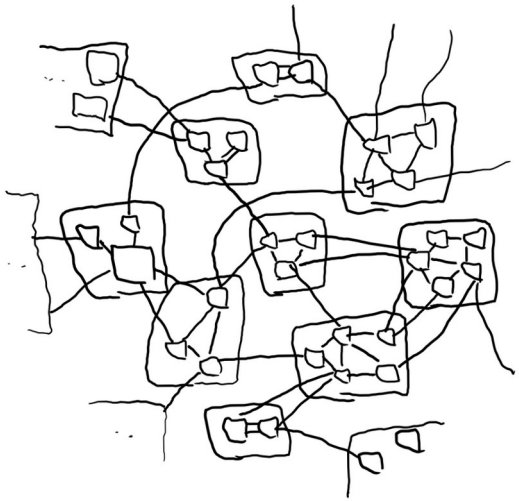


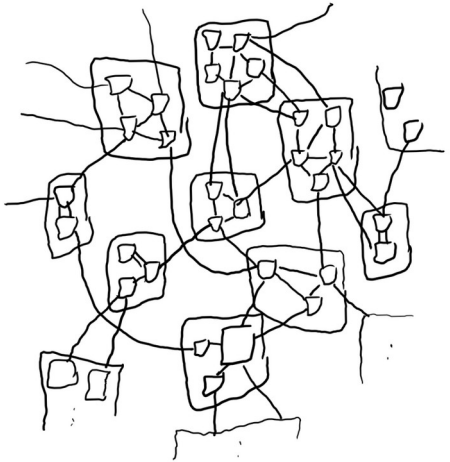

n-way

# Why do we need scalable n-way model matching?

# Models can be large



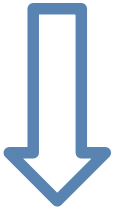




We present RaQuN

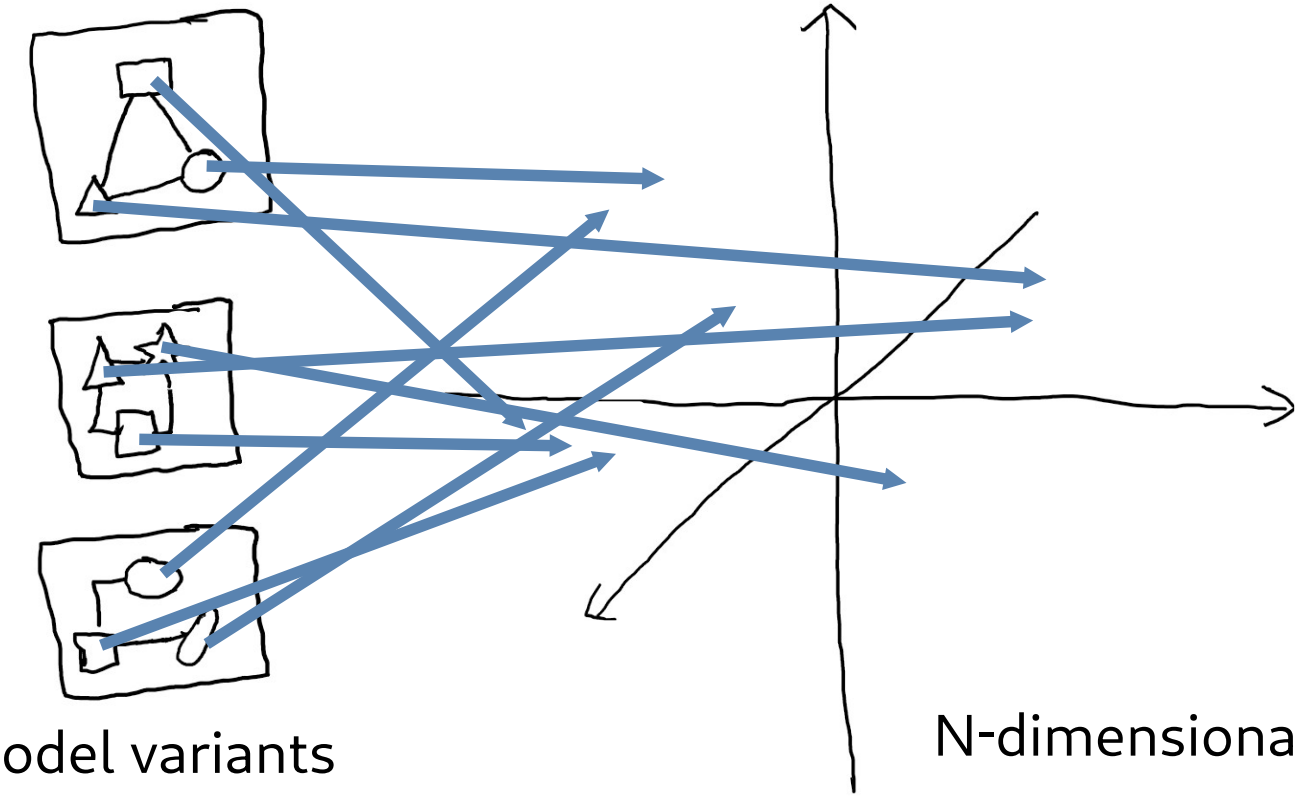
# What is the idea?

Fewer element comparisons



Faster matching

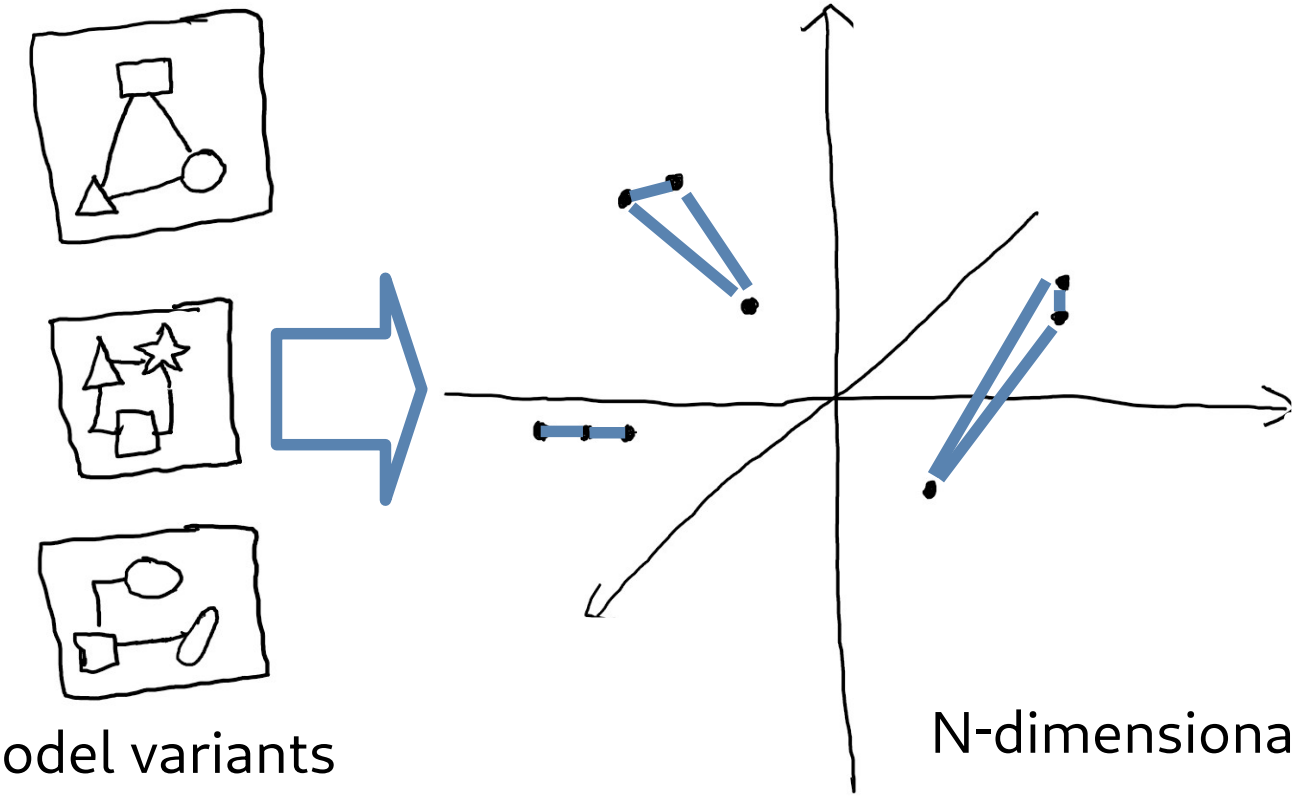
# First: Map elements to points in a vector space



Model variants

N-dimensional vector space

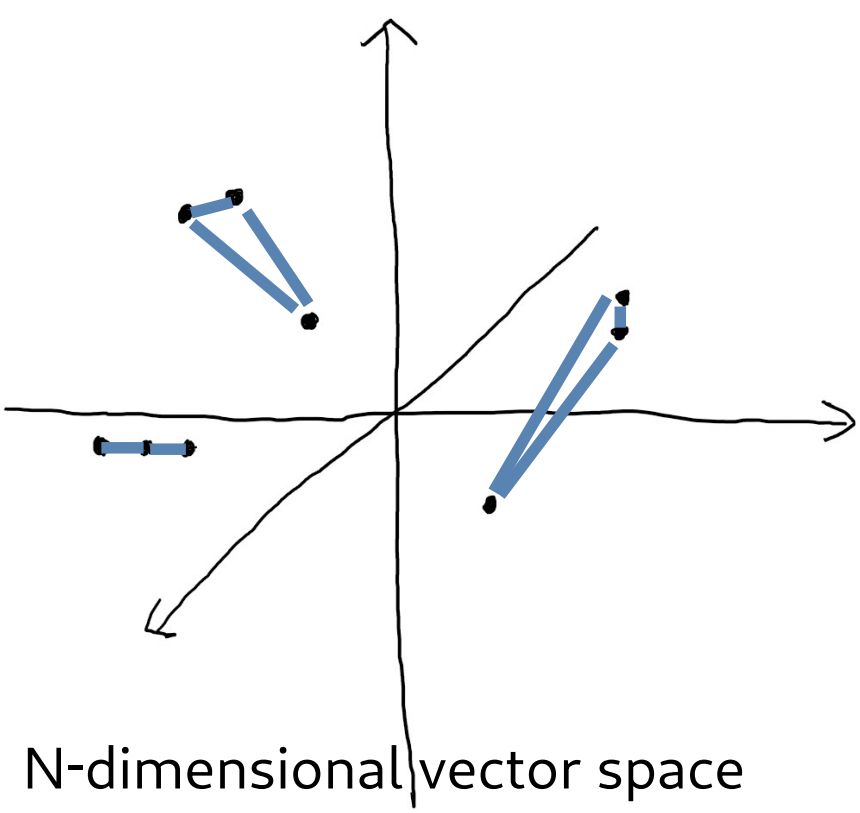
# Second: Collect the nearest neighbors of each element



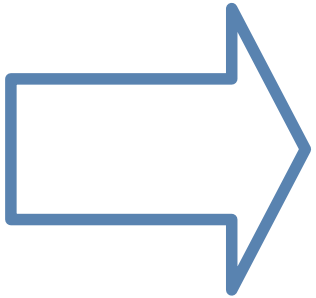
Model variants

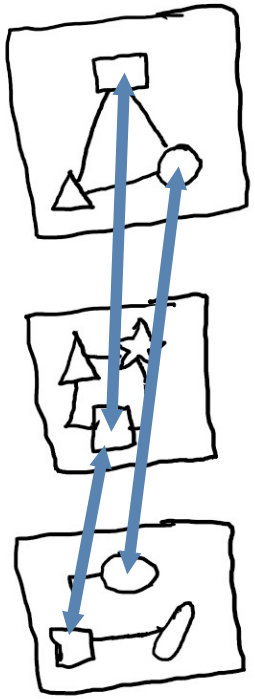
N-dimensional vector space

# Third: Compare and match neighboring elements



N-dimensional vector space

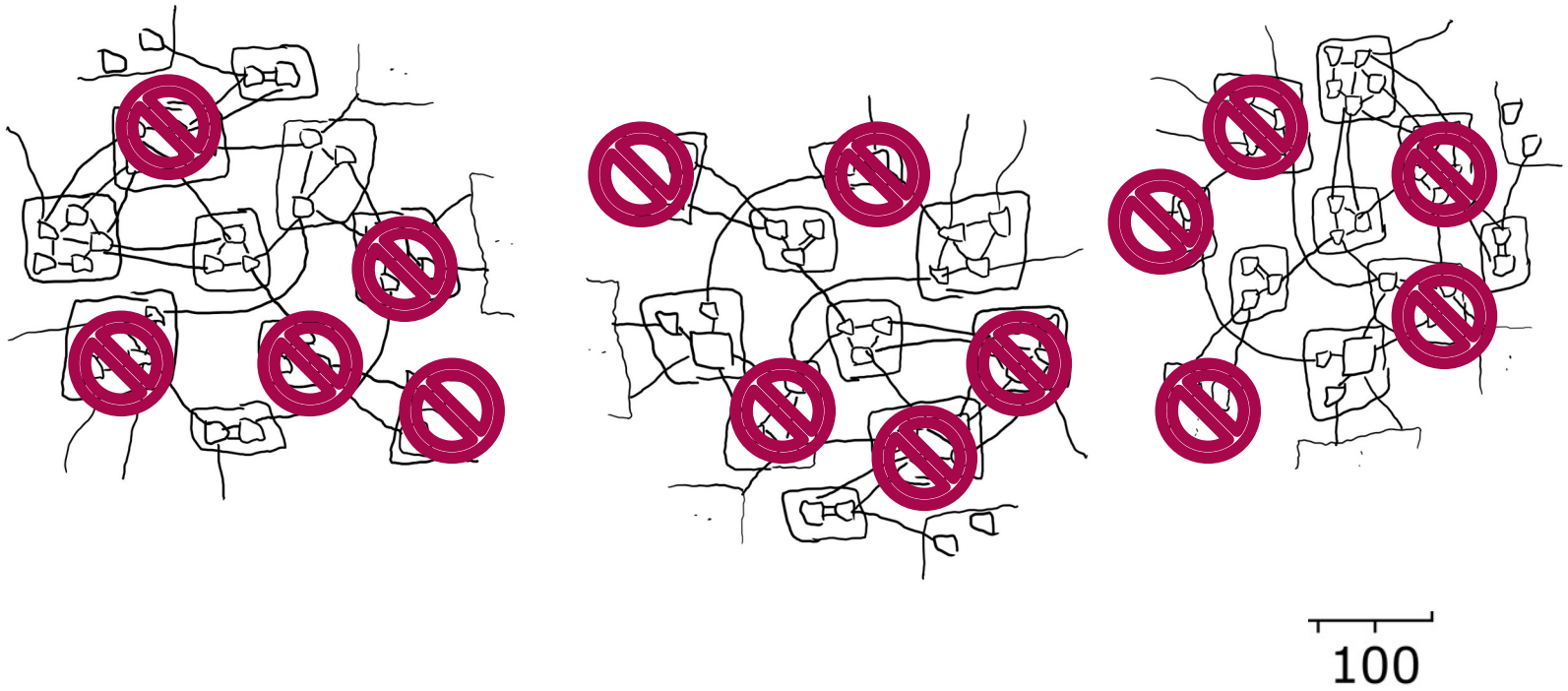




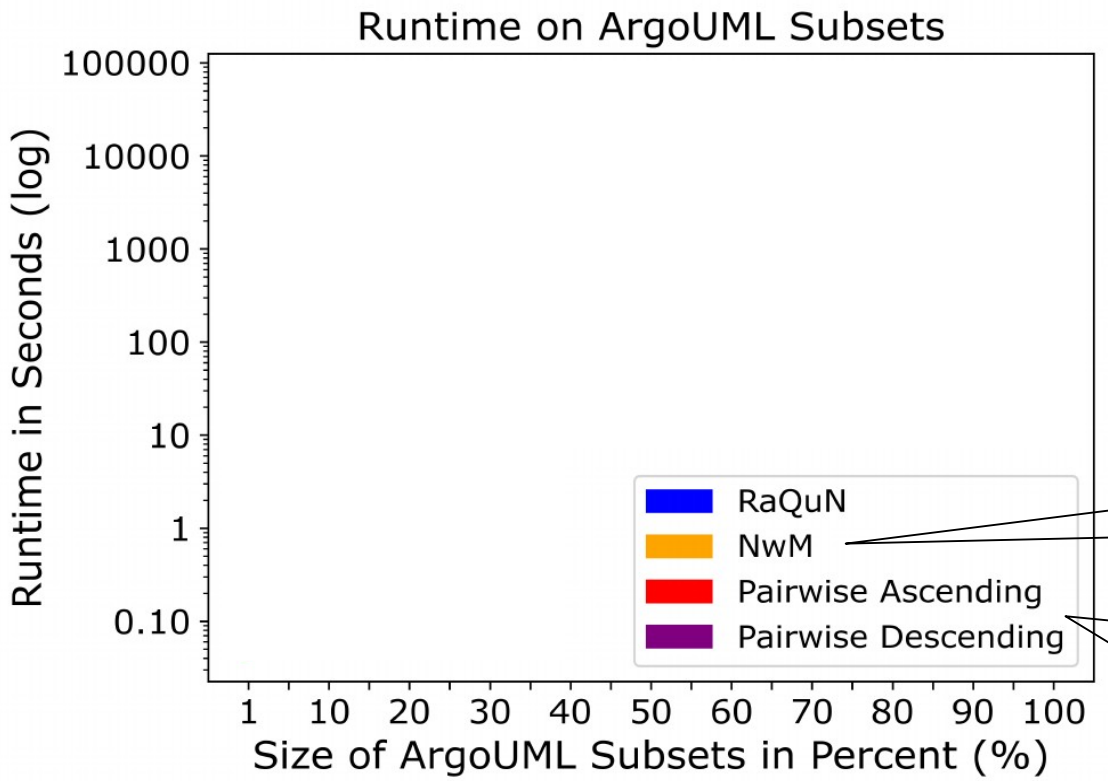
Model variants

# How well does RaQuN scale?

# Scalability on ArgoUML



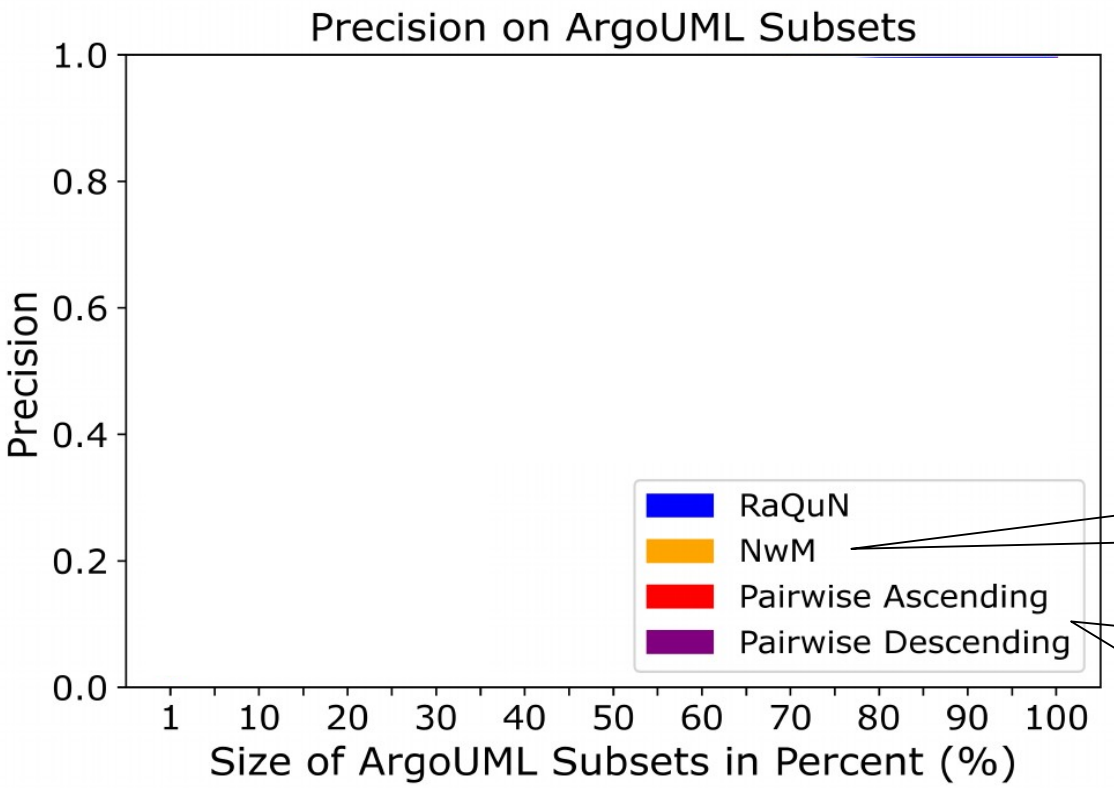
# How well does RaQuN scale?




J. Rubin and M. Chechik, "N-Way Model Merging," in Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, 2013, pp. 301–311

Apply Hungarian algorithm:
H. W. Kuhn, "The Hungarian Method for the Assignment Problem,"
Nordic J. Computing, vol. 2, no. 1-2, pp. 83–97, 1955.

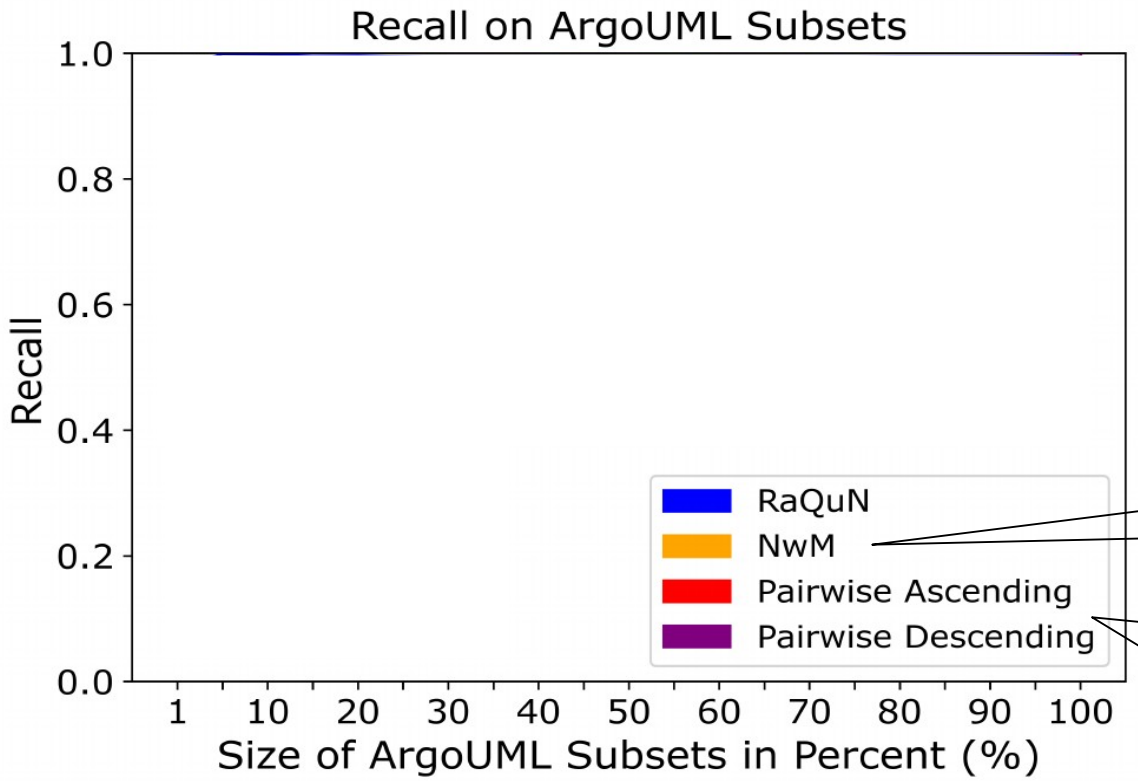
# Are the formed matches correct?




J. Rubin and M. Chechik, "N-Way Model Merging," in Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, 2013, pp. 301–311

Apply Hungarian algorithm:
H. W. Kuhn, "The Hungarian Method for the Assignment Problem," Nordic J. Computing, vol. 2, no. 1-2, pp. 83–97, 1955.

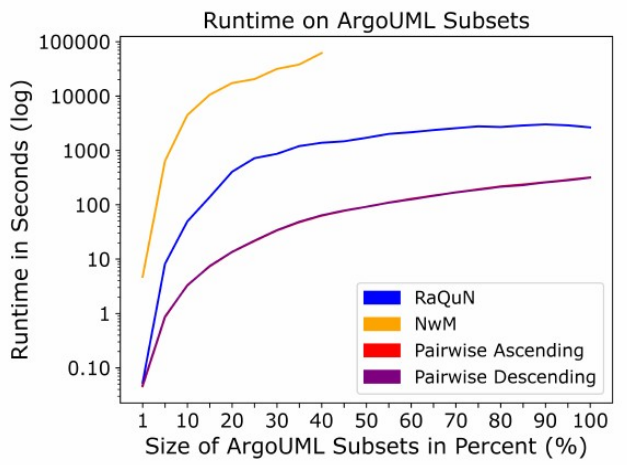
# Have all relevant matches been found?

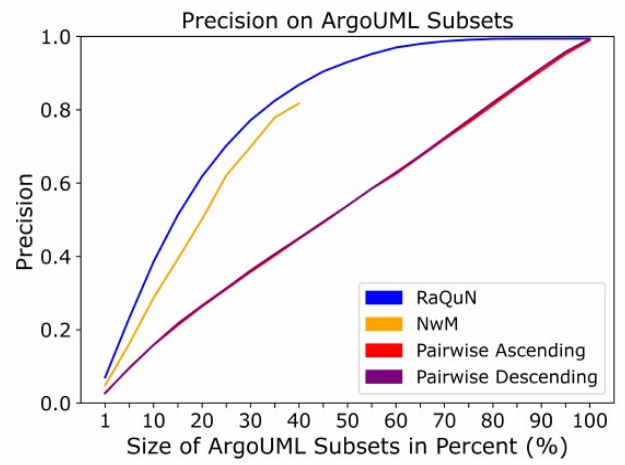


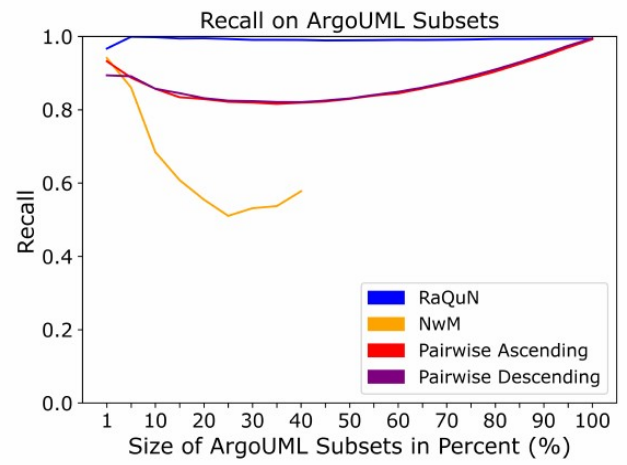

J. Rubin and M. Chechik, "N-Way Model Merging," in Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, 2013, pp. 301–311

Apply Hungarian algorithm:
H. W. Kuhn, "The Hungarian Method for the Assignment Problem,"
Nordic J. Computing, vol. 2, no. 1-2, pp. 83–97, 1955.

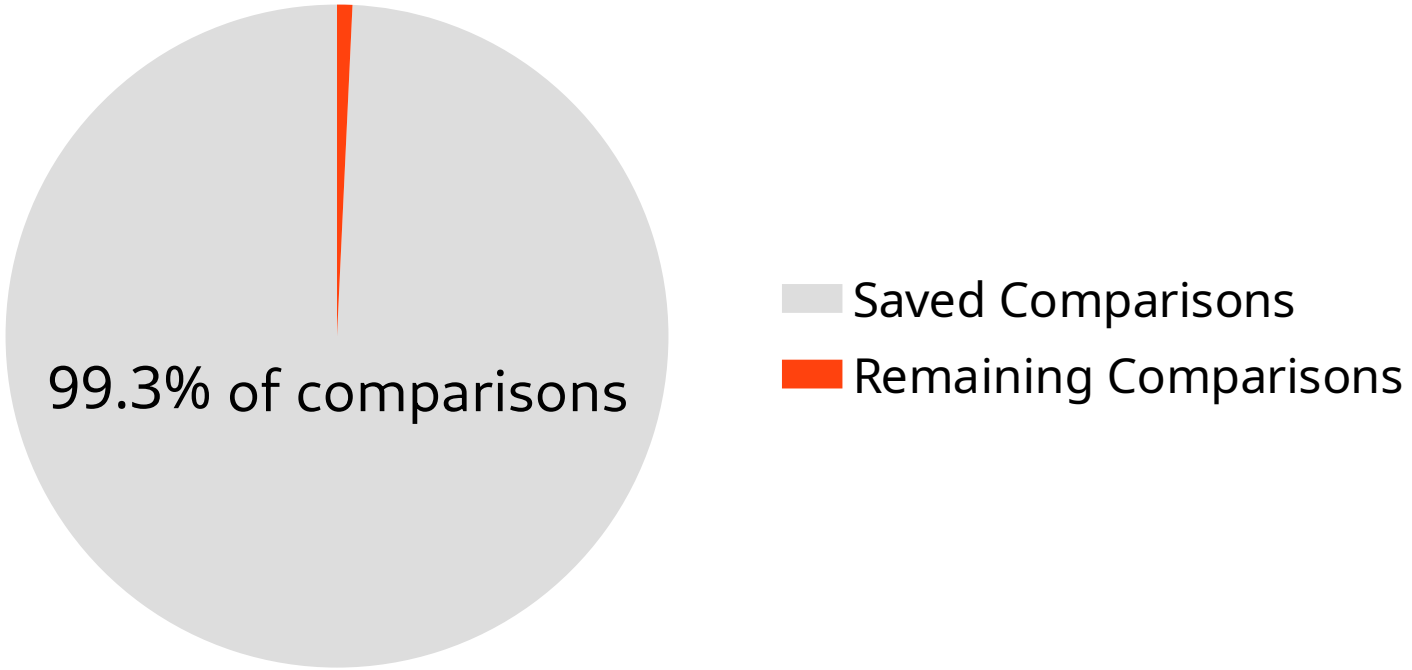
# Does RaQuN achieve better runtime/quality?



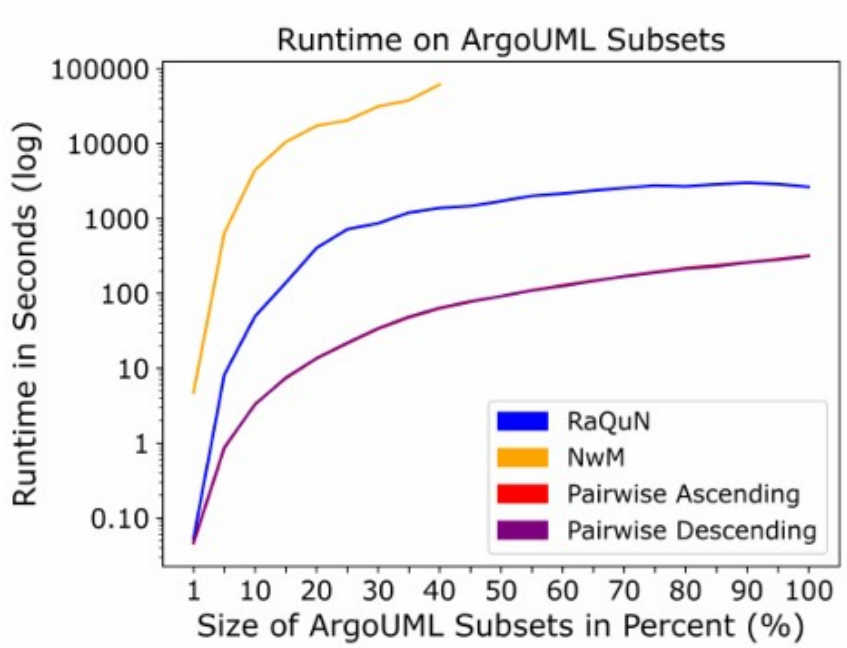
# How many comparisons are saved?

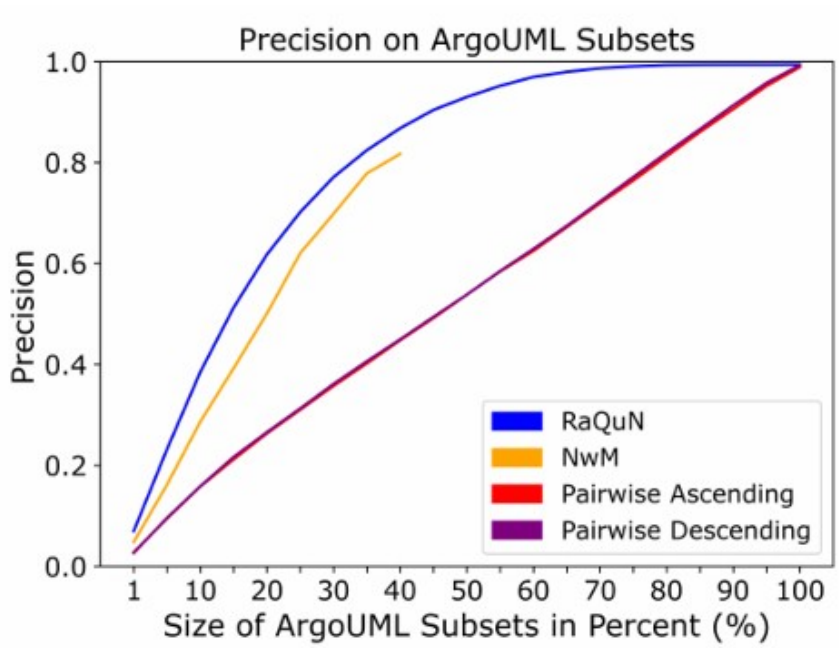


# Future Work

- Specialize on different domains
- Extend to n-to-m element matching
- Explore scenarios with interactive matching

# Summary


Runtime on ArgoUML Subsets
Runtime in Seconds (log)
100000
10000
1000
100
10
1
0.10
1 10 20 30 40 50 60 70 80 90 100
Size of ArgoUML Subsets in Percent (%)
RaQuN
NwM
Pairwise Ascending
Pairwise Descending

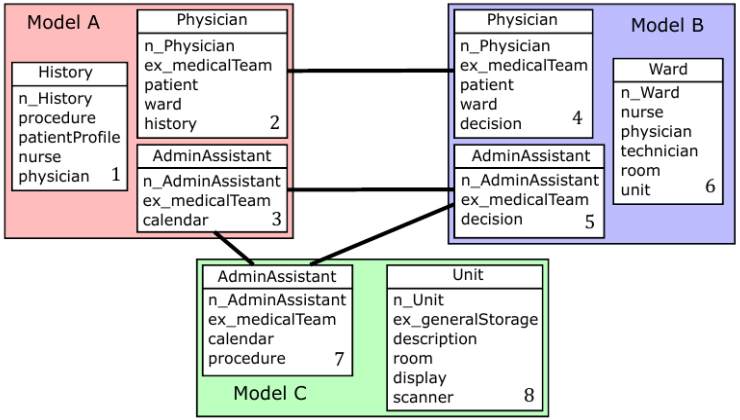

Precision on ArgoUML Subsets
Precision
1.0
0.8
0.6
0.4
0.2
0.0
1 10 20 30 40 50 60 70 80 90 100
Size of ArgoUML Subsets in Percent (%)
RaQuN
NwM
Pairwise Ascending
Pairwise Descending

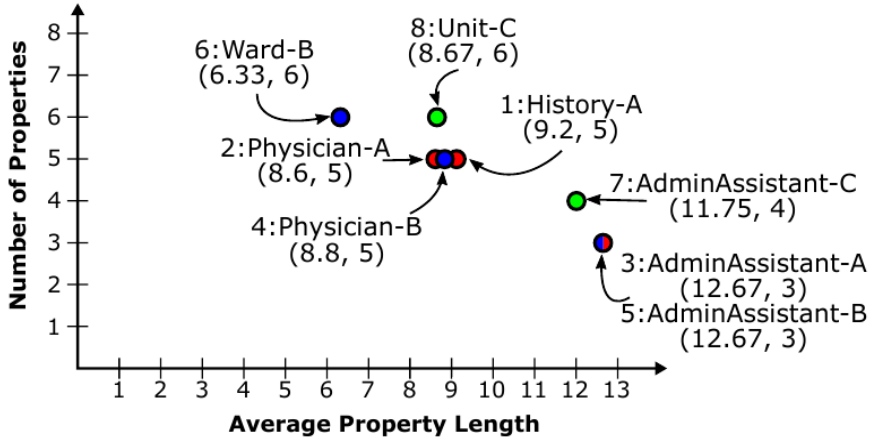
# Appendix

# We consider a variety of subjects

| | Model Type | #Models | Elements | |
|---|---|---|---|---|
| | | | Avg. | Median |
| Hospital | Simple class diag. | 8 | 27.62 | 26 |
| Warehouse | Simple class diag. | 16 | 24.25 | 22 |
| Random | Synthetic | 100 | 26.99 | 26 |
| Loose | Synthetic | 100 | 28.88 | 29 |
| Tight | Synthetic | 100 | 25.01 | 25 |
| PPU Structure | SysML block diag. | 13 | 32.15 | 32 |
| PPU Behavior | UML statemachines | 13 | 221.85 | 228 |
| bCMS | UML class diag. | 14 | 67.71 | 63 |
| BCS | Component/connector | 18 | 78.78 | 72 |
| ArgoUML | UML class diag. | 7 | 1,752.86 | 1,749 |
| Apo-Games | Simple class diag. | 20 | 63.05 | 60 |

# Example vectorization



# # Comparisons saved

| Dataset | Full N-Way Matching #Comparisons | RaQuN #Comparisons | Saved |
|---|---|---|---|
| Hospital | 21,211 | 1,229 | 94.2% |
| Warehouse | 70,037 | 6,408 | 90.9% |
| Random | 33,600 | 2,499 | 92.6% |
| Loose | 26,244 | 2,670 | 89.8% |
| Tight | 25,237 | 2,215 | 91.2% |
| PPU Structure | 80,620 | 41,533 | 48.5% |
| PPU Behavior | 3,814,644 | 259,827 | 93.2% |
| bCMS | 416,571 | 106,585 | 74.4% |
| BCS | 939,346 | 253,862 | 73.0% |
| ArgoUML | 64,521,622 | 481,179 | 99.3% |
| Apo-Games | 750,319 | 31,880 | 95.8% |

# Weight metric

Given a match $t$, the weight is calculated as

$$w(t) = \frac{\sum_{2 \leq j \leq |t|} j^2 \cdot n_j^p}{n^2 \cdot |\pi(t)|} \tag{2}$$

where $|t|$ denotes the size of the match, $n_j^p$ the number of properties that occur in exactly $j$ elements of the match, and $\pi(t)$ is the set of all distinct properties of all elements in $t$.

# Detailed Results (1)

TABLE II

COMPARISON OF ACHIEVED WEIGHTS AND RUNTIMES ACROSS ALL ALGORITHMS, AVERAGED OVER 30 RUNS FOR EACH SUBJECT.

| Algorithm | Hospital | | Warehouse | | Random | | Loose | | Tight | | Apo-Games | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) |
| RaQuN | **4.92** | **0.08** [0.07, 0.12] | **1.63** | **0.32** [0.27, 0.93] | **1.04** | **0.07** [0.05, 0.13] | **1.03** | **0.13** [0.07, 0.33] | **0.94** | **0.07** [0.05, 0.11] | **18.27** | 71.12 [41.30, 104.58] |
| NwM | 4.49 | 20.64 [16.18, 24.22] | 1.46 | 74.51 [29.53, 84.87] | 0.80 | 24.00 [1.23, 76.02] | 0.79 | 22.40 [1.12, 70.68] | 0.88 | 39.75 [1.63, 66.60] | 17.91 | 5,462.91 [3,742.22, 6,597.83] |
| Pairwise Ascending | 4.49 | 0.31 [0.28, 0.44] | 1.11 | 0.36 [0.33, 0.45] | 0.79 | 0.21 [0.10, 0.31] | 0.74 | 0.14 [0.08, 0.22] | **0.94** | 0.22 [0.16, 0.34] | 12.96 | **10.42** [9.32, 12.04] |
| Pairwise Descending | 4.72 | 0.16 [0.14, 0.22] | 1.27 | 0.36 [0.32, 0.53] | 0.78 | 0.15 [0.10, 0.23] | 0.74 | 0.14 [0.08, 0.25] | 0.93 | 0.22 [0.17, 0.33] | 16.40 | 10.68 [9.27, 13.50] |

| Algorithm | PPU Structure | | PPU Behavior | | bCMS | | BCS | | ArgoUML | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) | Weight | Time (in s) |
| RaQuN | **28.95** | 0.85 [0.81, 0.90] | **164.53** | 18.32 [16.39, 20.34] | **41.53** | 2.84 [1.40, 3.58] | **51.17** | 12.58 [8.10, 18.22] | **1727.65** | 2,647.76 [1,483.48, 3,324.70] |
| NwM | 28.64 | 9.27 [7.84, 10.07] | 146.35 | 4,616.30 [3,410.86, 5,919.65] | 41.25 | 247.31 [227.52, 275.95] | 43.20 | 330.25 [235.74, 388.45] | - - - timeout - - - | |
| Pairwise Ascending | 28.65 | 0.27 [0.22, 0.38] | 145.46 | 11.03 [10.32, 12.61] | 39.76 | 1.16 [1.13, 1.21] | 43.83 | 5.69 [3.85, 7.43] | 1702.91 | 318.26 [297.43, 379.76] |
| Pairwise Descending | 28.89 | **0.26** [0.21, 0.42] | 142.56 | **10.84** [10.20, 13.01] | 38.76 | **1.04** [1.01, 1.07] | 47.16 | **4.84** [3.37, 6.73] | 1710.25 | **314.88** [302.28, 329.35] |