Developing H2PC for BN Structure Learning: A Comparative Study

Lorenzo Venturi

11/04/2025

Fundamentals of AI and KR – Module 3

Master's Degree in Artificial Intelligence, University of Bologna

PROJECT OBJECTIVES

- *Implement* the **H2PC** algorithm

pyAgrum

Pandas

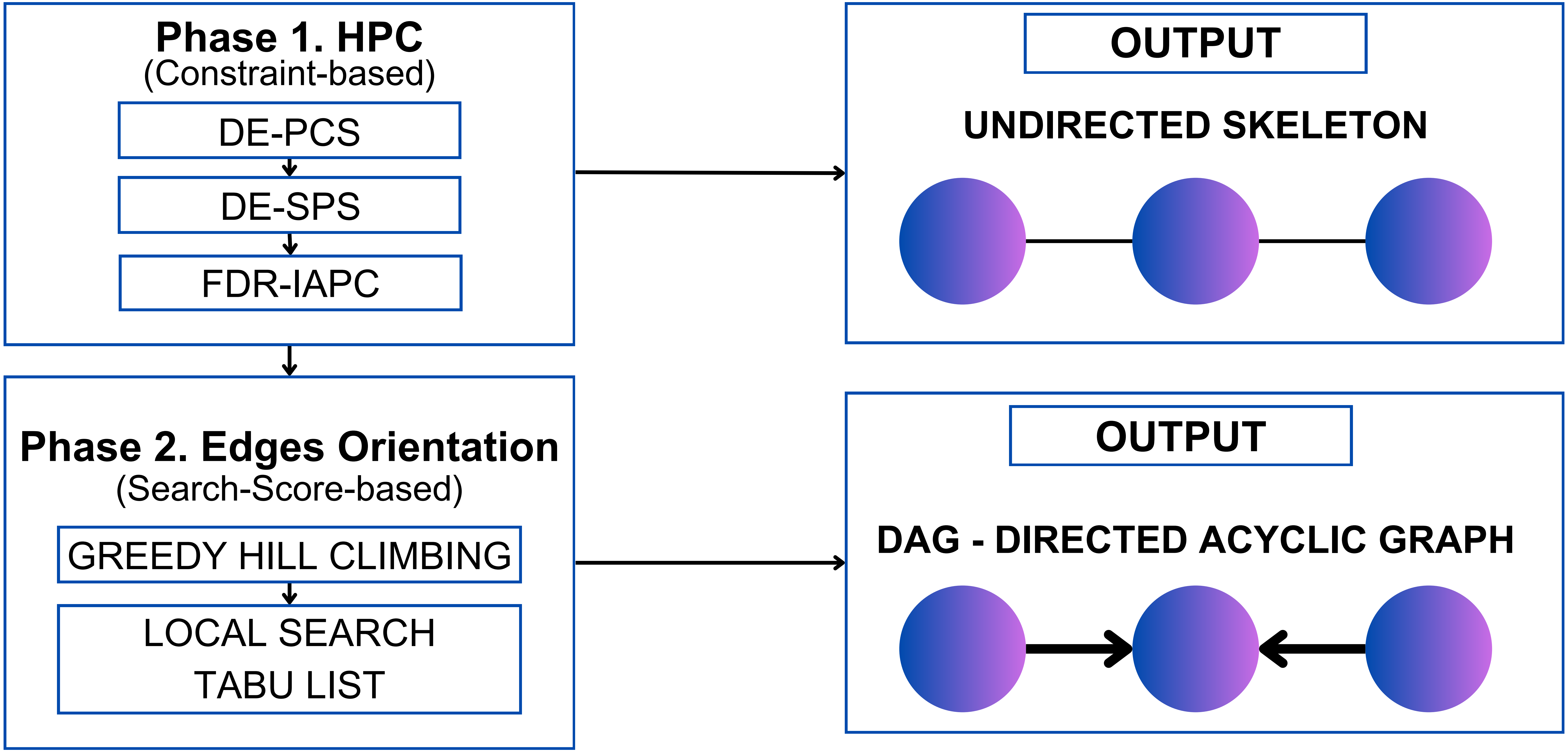
- *Compare* performances of H2PC with **MMHC** (Max-min hill climbing)

pgmpy

- *Visualize and compare* the **networks structures**

graphViz

H2PC ALGORITHM



CORE CONCEPTS C.B. PHASE

Independence tests
pyAgrum.BnLearner.chi2

Markov Boundary: *minimum* set of a Markov Blanket

HPC: receives a target node **T**, returns an estimation of $\mathbf{PC_T}$.

1. **DE-PCS**(Data-efficient Parents & Children Superset)
   - ✅ returns: $\mathbf{PC_T}$ with restriction on conditioning size **<=1** (ex. $T \perp\!\!\!\perp X \mid Y$)
2. **DE-SPS**(Data-efficient Spouses Superset)
   - ✅ returns: $\mathbf{SP_T}$ with restriction on conditioning size **<=2** (ex. $T \perp\!\!\!\perp X \mid \{Y1, Y2\}$)
3. **FDR-IAPC**(Incremental Association Parents and Children with False Discovery Rate control)
   - 3.1. **FDR-IAMB**(Incremental Association Markov Blanket with False Discovery Rate control)
     - ✅ returns: the true **Markov Boundary** with **FDR** check
   - ✅ returns: output of FDR-IAMB without spouses, the final $\mathbf{PC_T}$

CORE CONCEPTS S.S. PHASE

∀ variable in dataset apply **HPC** → combine **Markov Boundaries** → **Skeleton Network**

↓

GreedyHillClimb search

pyAgrum.BnLearner.useGreedyHillClimbing()

↓

pyAgrum.BnLearner.learnBN()

↓

DAG

↓

pyAgrum.BnLearner.setInitialDag(DAG)

↓

Local Search with tabu list

pyAgrum.BnLearner.useLocalSearchWithTabuList()

↓

FINAL DAG

DATASETS

	NODES	EDGES	AVG. MB SIZE	PARAMETERS	SIZE	TRAIN SIZES
ASIA	8	8	2.5	18	10.000	[100,500,1000,2000, 3000,5000,8000]
SACHS	11	17	3.09	178	1.000	[100,200,400,600,800]
ALARM	37	46	3.51	509	10.000	[100,500,1000,2000, 3000,5000,8000]

METRICS EVALUATION

<u>Structure metrics</u> (based on the skeleton of the network)

Undirected

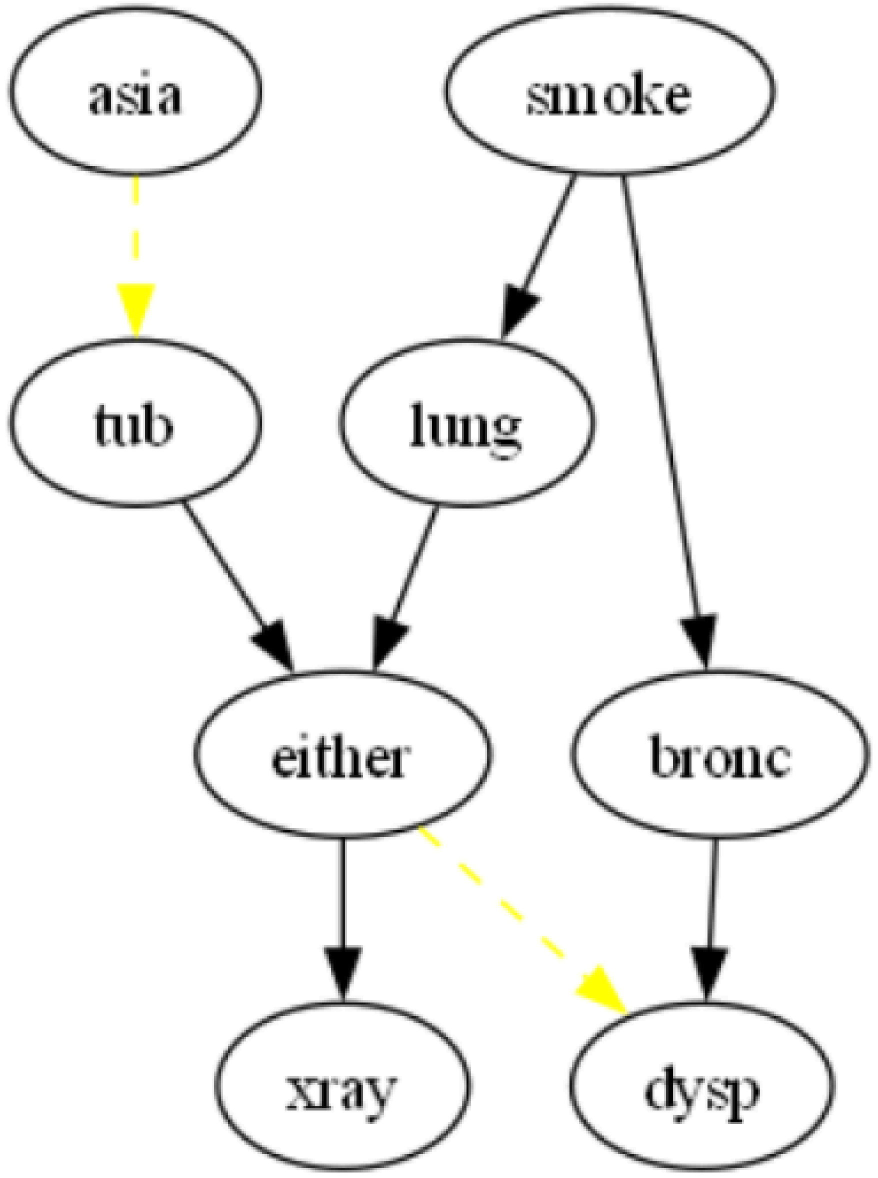
- **False Positive Edge Ratio** → $\frac{\text{false positive edges}}{\text{true positive edges}}$
- **Structure Precision** → $\frac{\text{true positive edges}}{\text{number of edges}}$
- **Structure Recall** → $\frac{\text{true positive edges}}{\text{true number of edges}}$

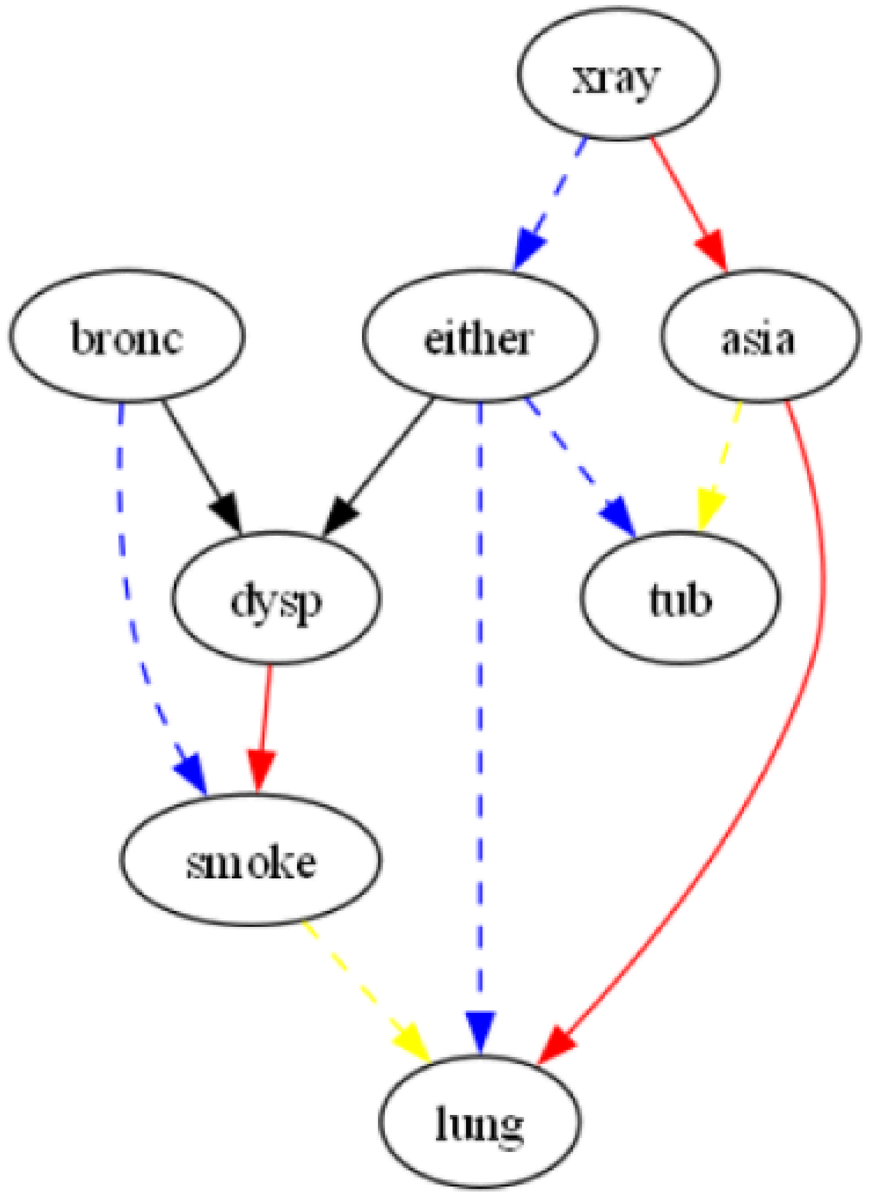
<u>Performance indicators</u> (on the DAG)

- **SHD** (structural hamming distance) → **add**,**delete**,**reverse** edges to reach the true network
- **BIC Score** → measures goodness of the fit (**higher better**)
- **CPU time** → measures optimality of the algorithm (**lower better**)

RESULTS ASIA

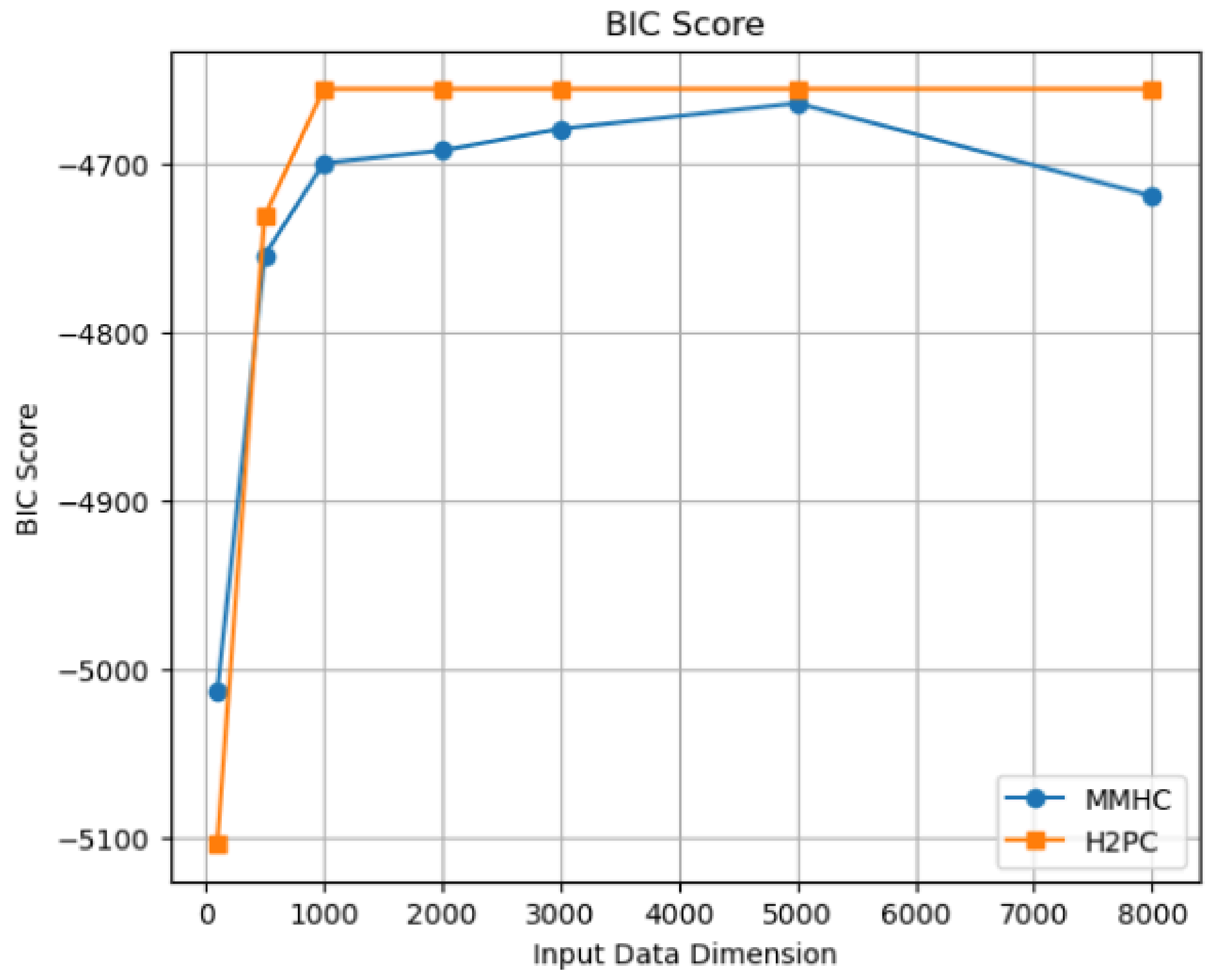
Red Edges: Incorrect edges — Blue Edges: Reversed edges — Yellow Edges: Missing edges





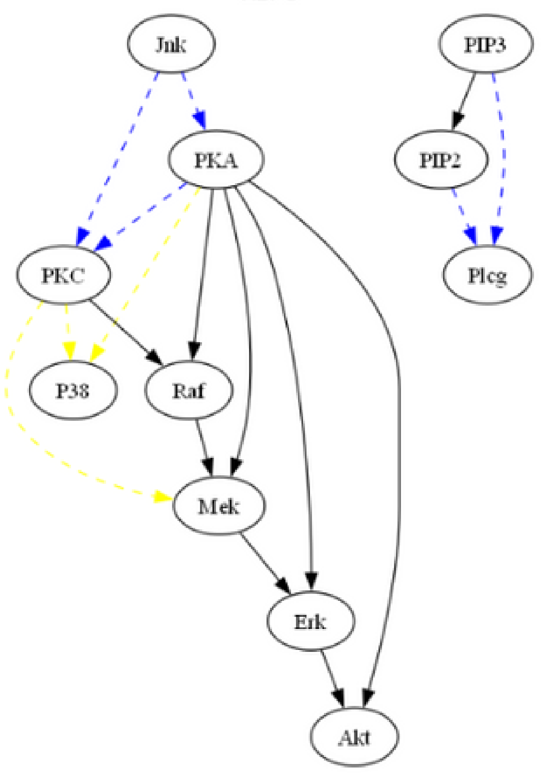
Metrics ratio

Metrics	100	500	1000	2000	3000	5000	8000
FPE	0.00	0.33	0.00	0.00	0.00	1.00	0.00
Precision	2.00	1.50	1.29	1.33	1.29	1.00	1.50
Recall	1.50	1.50	0.86	1.00	0.86	0.86	1.00
SHD	0.89	0.40	0.50	0.57	0.57	0.80	0.22
BIC Score	1.018	0.995	0.991	0.992	0.995	0.998	0.987
Ex.Time	0.018	0.011	0.012	0.057	0.082	0.139	0.148

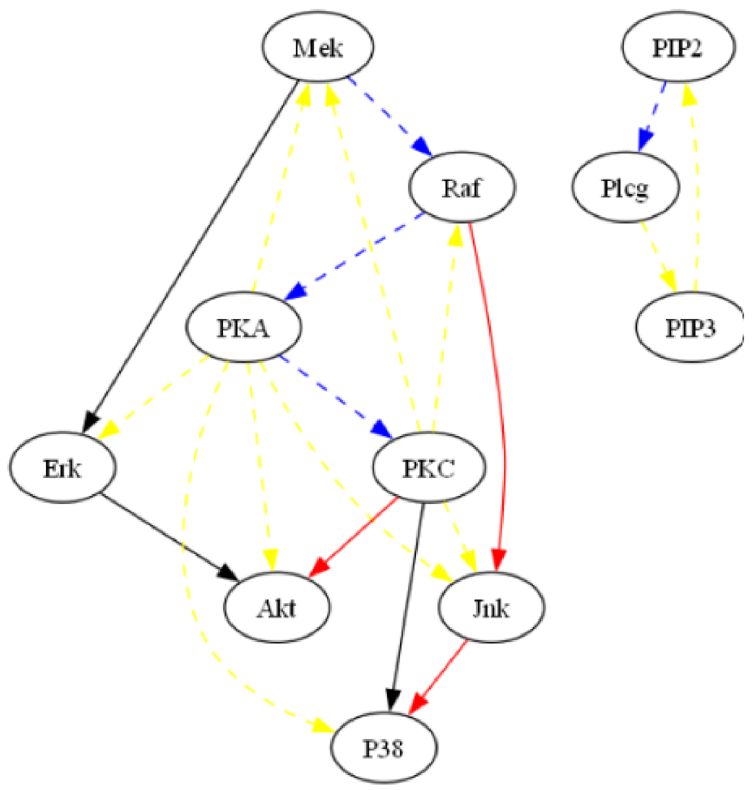


RESULTS SACHS

H2PC

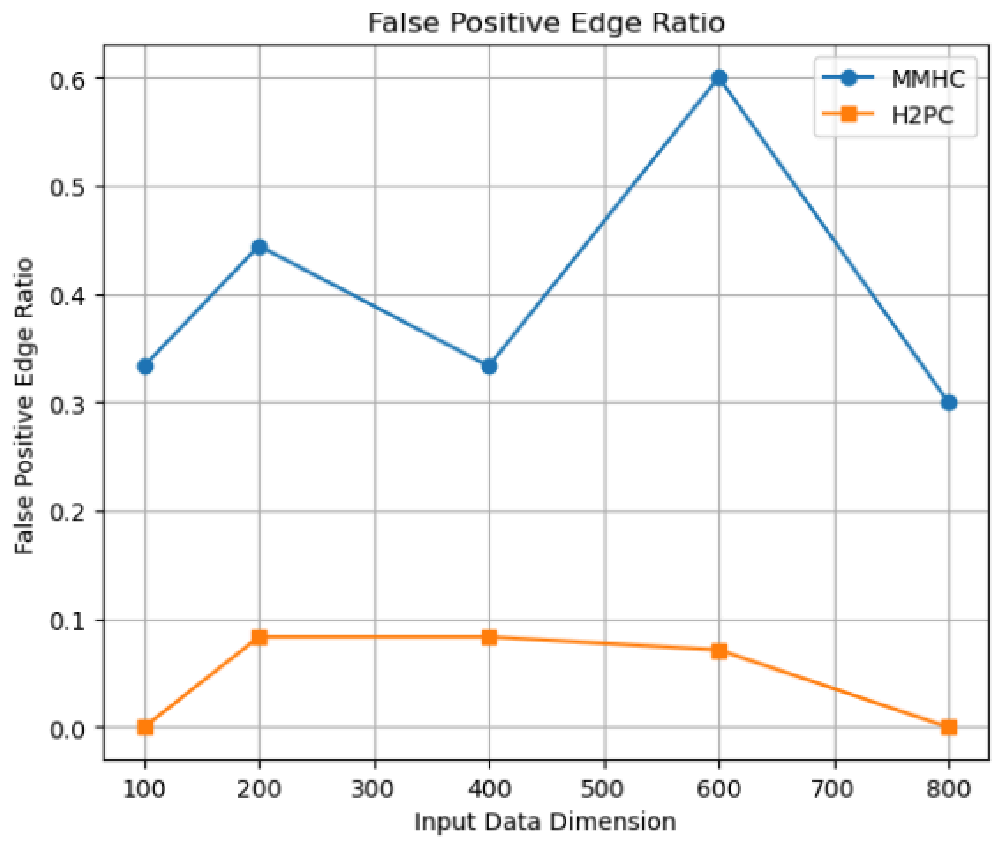


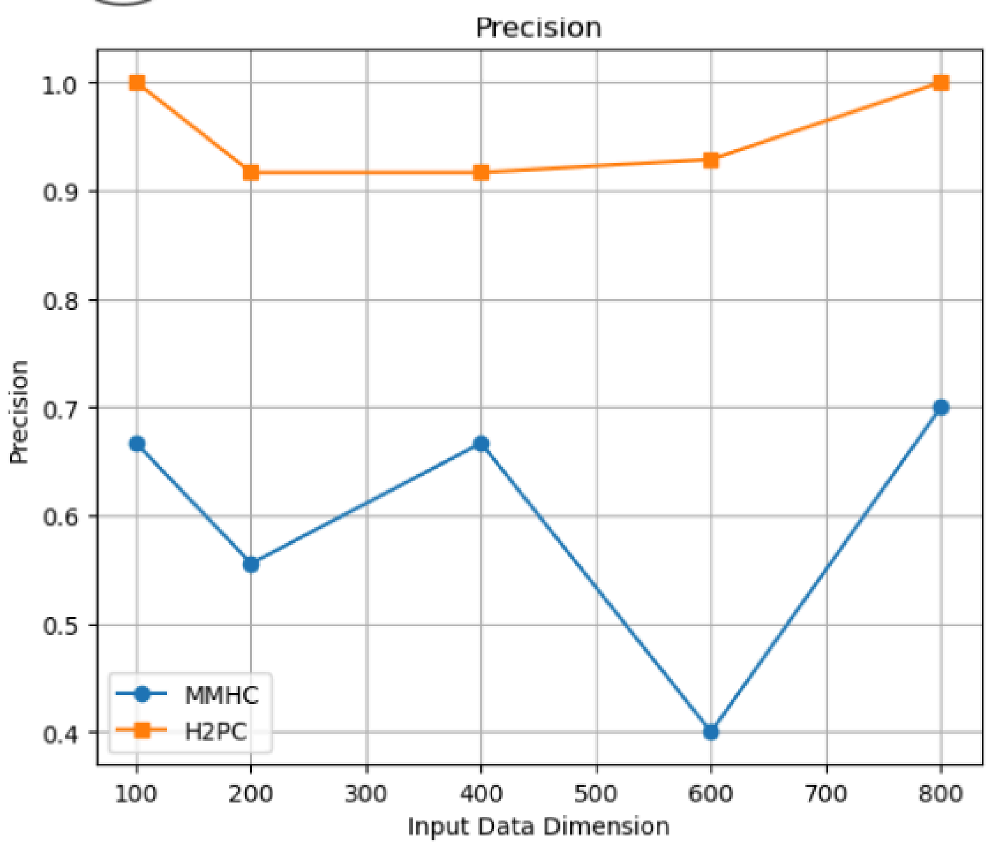
MMHC

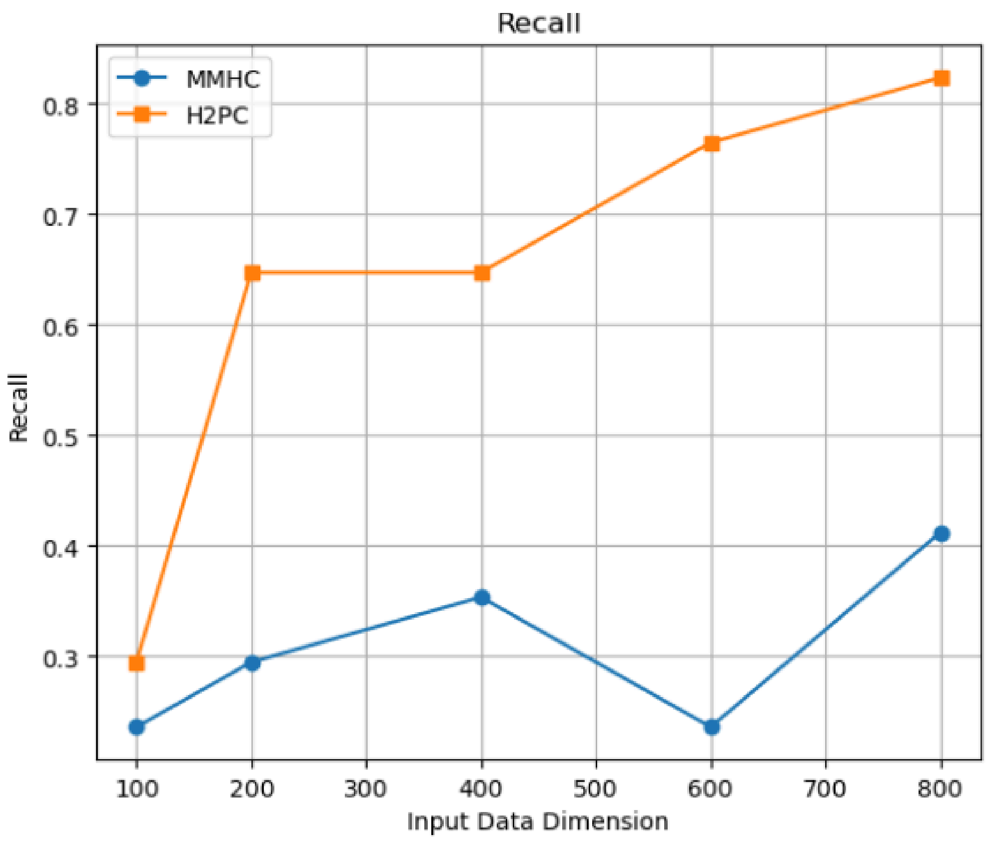


Metrics ratio

Metrics	100	200	400	600	800
FPE	0.00	0.187	0.250	0.119	0.00
Precision	1.500	1.650	1.375	2.321	1.429
Recall	1.250	2.200	1.833	3.250	2.000
SHD	0.824	0.579	0.765	0.450	0.471
BIC Score	0.942	0.956	0.950	0.950	1.012
Ex. Time	7.9e-5	9.9e-5	1.99e-4	1.30e-4	9.7e-5

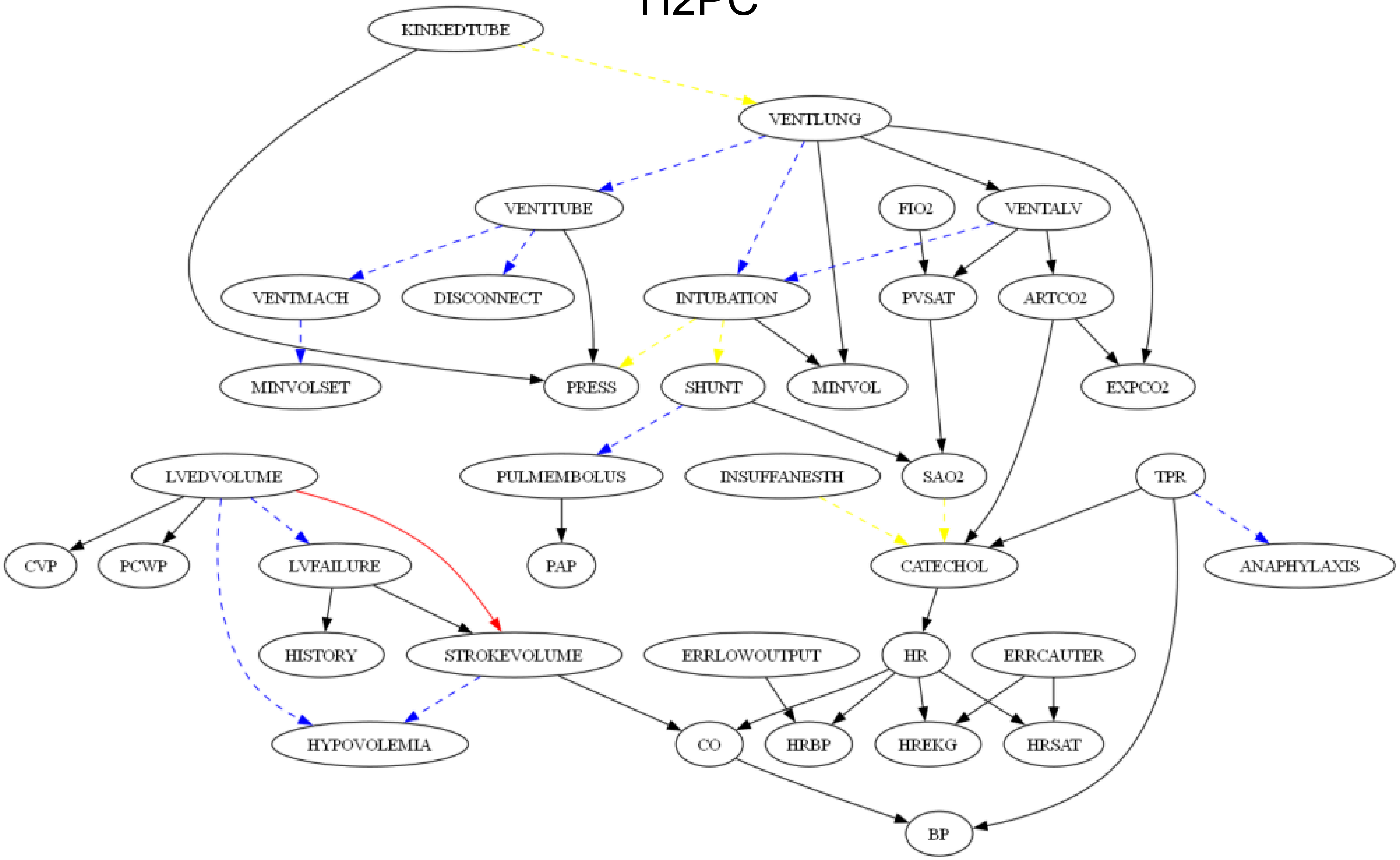






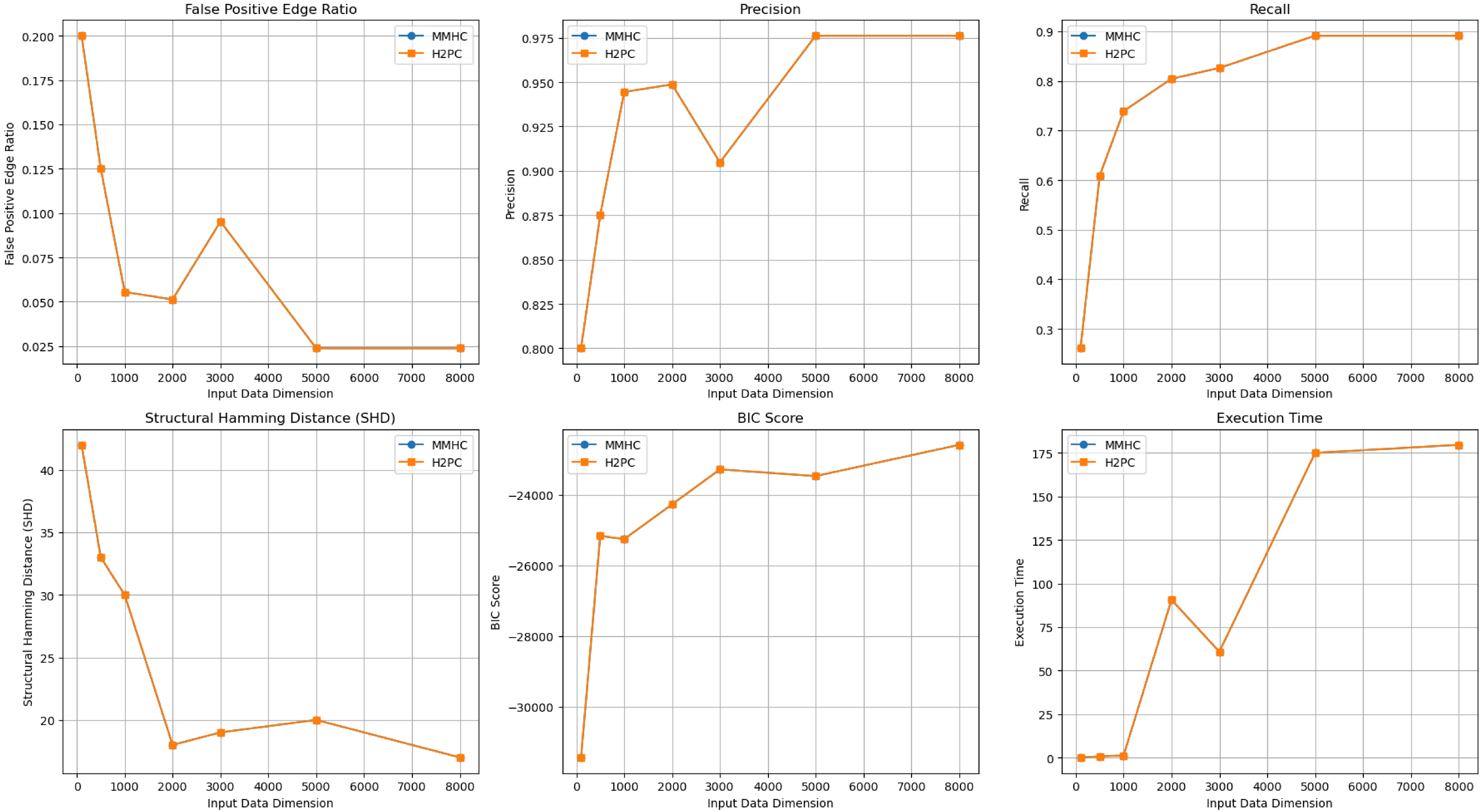
RESULTS ALARM

H2PC



Red Edges: Incorrect edges   Blue Edges: Reversed edges   Yellow Edges: Missing edges

RESULTS ALARM_2



CONCLUSION AND FUTURE IMPLEMENTATIONS

ORIGINAL PAPER RESULTS (R IMPLEMENTATION)	RESULTS (MY PYTHON IMPLEMENTATION)
H2PC OUTPERFORMS MMHC	H2PC OUTPERFORMS MMHC
H2PC **SLOWER** THAN MMHC	H2PC **FASTER** THAN MMHC
HIGHER FALSE POSITIVE EDGES	GENERALLY **LOWER** FALSE POSITIVE EDGES
MORE TESTS DONE	LESS TESTS DONE

Future ideas:

1. Implement the MMHC algorithm in pyAgrum and repeat the tests
2. Complete the tests with various independence thresholds
3. Try to implement a parallelization of the processes to fasten up