Refining Imprecise Labels

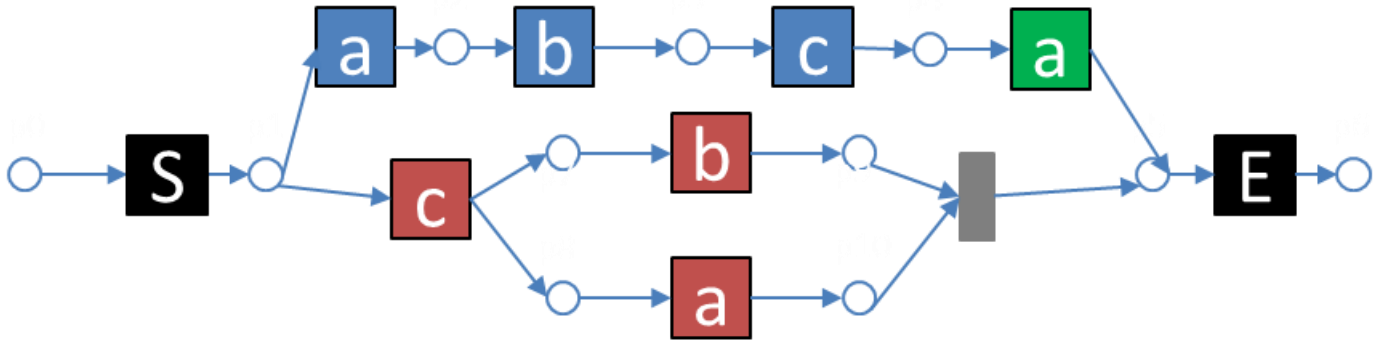
Xixi Lu

Dirk Fahland

Wil van der Aalst

Eindhoven University of Technology

Practical Problem and Motivation

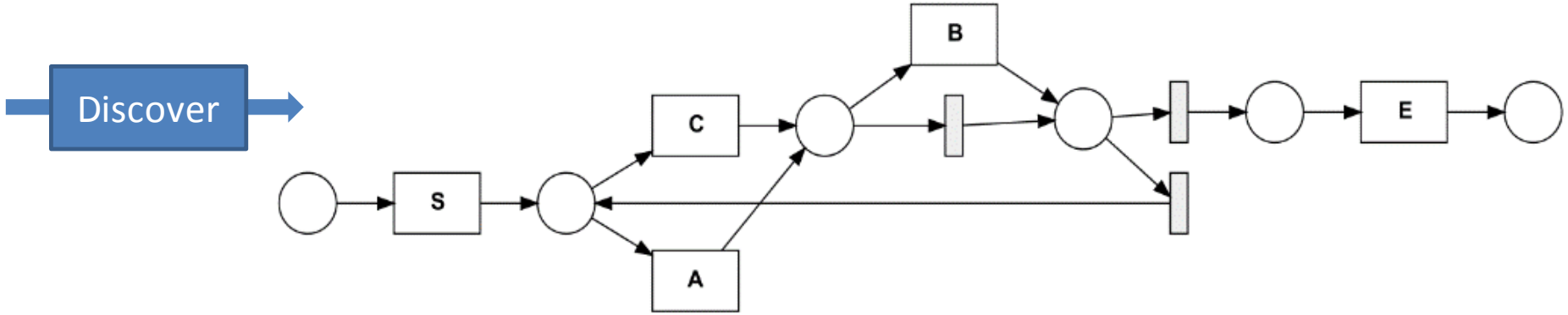


Log with imprecise labels

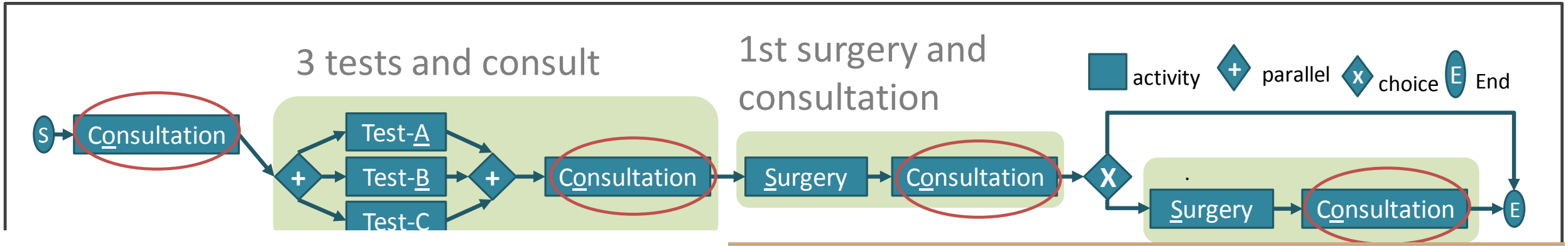
t1 = s abc a e

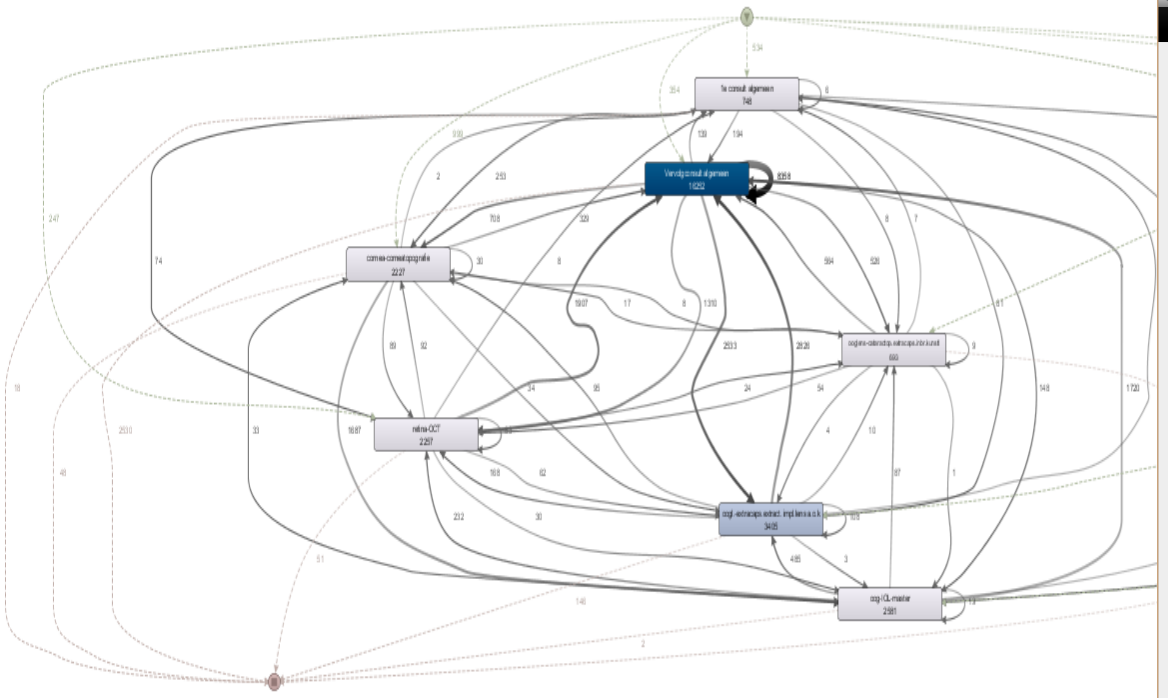
t2 = s cba e

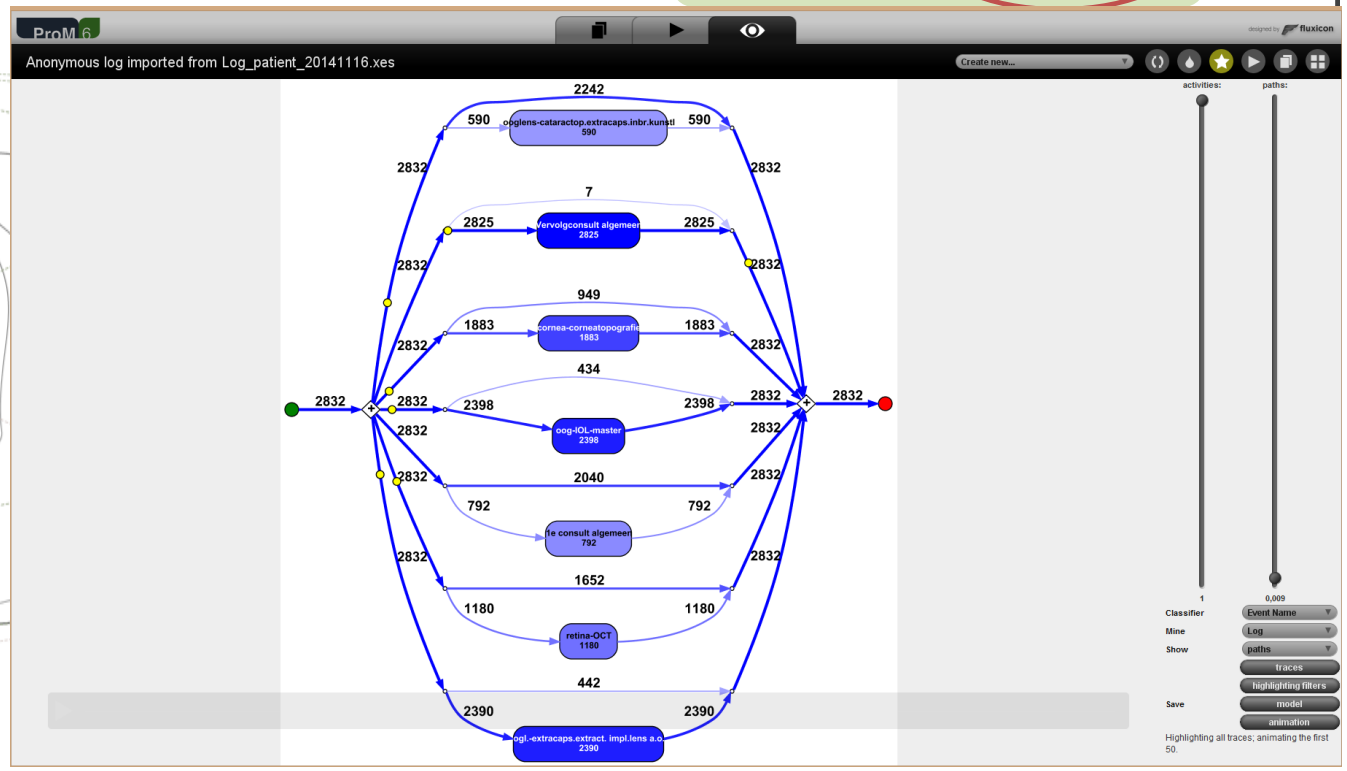
t3 = s cab e



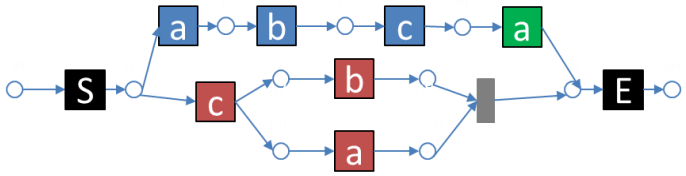
Real-life Example : Hospital Data







2. Problem Setting



Our focus : Problem of label refinement

imprecise

Precise

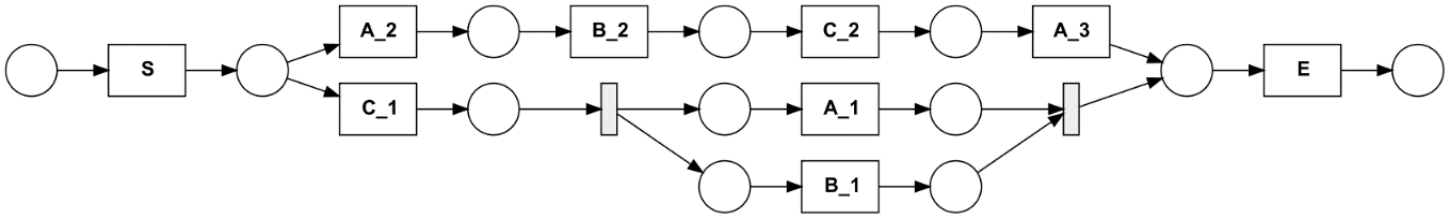
t1 = < s abc a e >
t2 = < s cba e >
t3 = < s cab e >

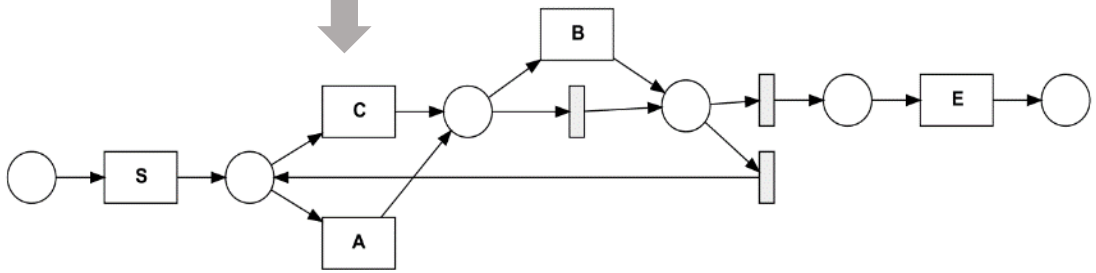
Refining Label

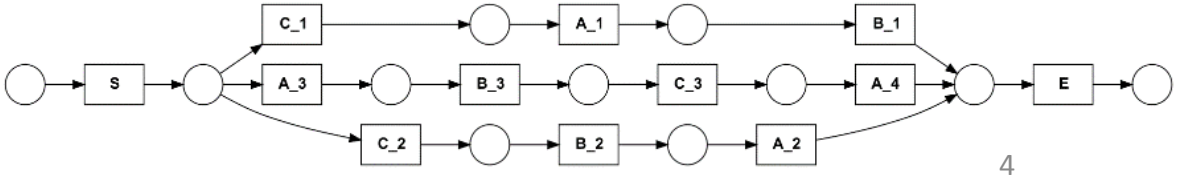
<s $a_2b_2c_2$ a_3 e>
<s $c_1b_1a_1$ e>

The optimal labels are unknown!

<s $a_3b_3c_3a_4$ e>
<s $c_2b_2a_2$ e>
<s $c_1a_1b_1$ e>







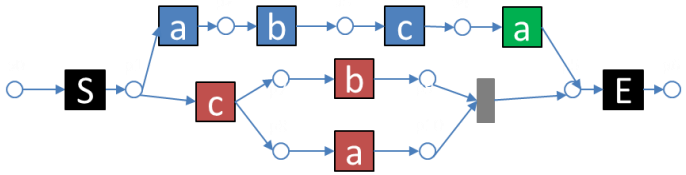
Aims and Complexity

- Refine labels to
  - (A1) help discover a model closer to the system
  - (A2) help discover more precise and structured models
    yet preserve fitness
  - (A3) provide users with more options and different views w.r.t. a log

- Complications

Complications	Decision
(A1) The system is unknown and depends on the domain experts	Use heuristics
(A2) More precise log ≠> more precise model ≠> higher precision, depends on the algorithm and the precision metric used.	Have a backup plan => Return the original log, so we are not worse

Complication w.r.t. A2



Our focus : Problem of label refinement

imprecise

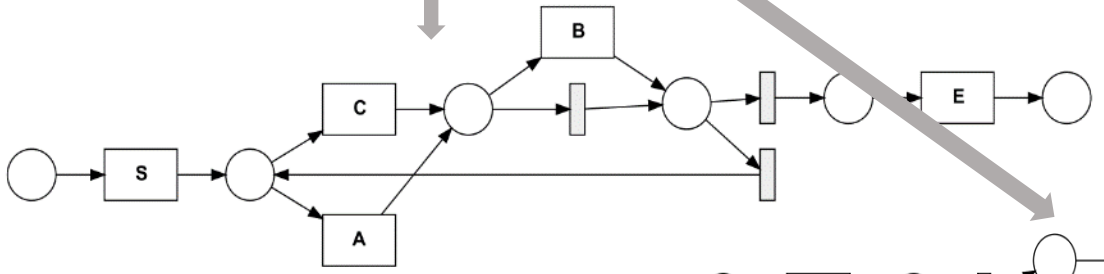
Precise

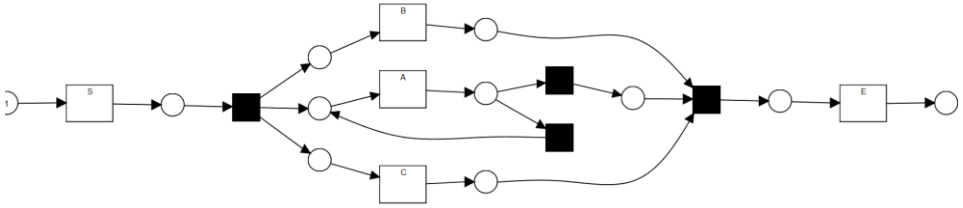
t1 = < s abc a e >
t2 = < s cba e >
t3 = < s cab e >

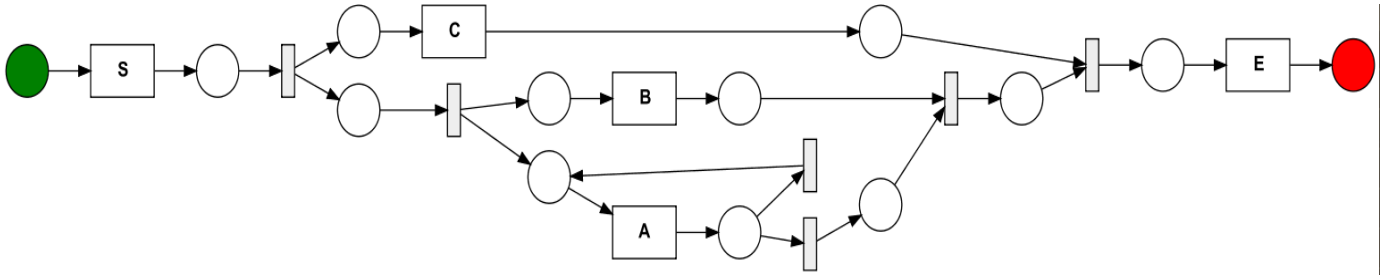
Refining Label

<s $a_2b_2c_2$ a_3 e>
<s $c_1b_1a_1$ e>
<s $c_1a_1b_1$ e>

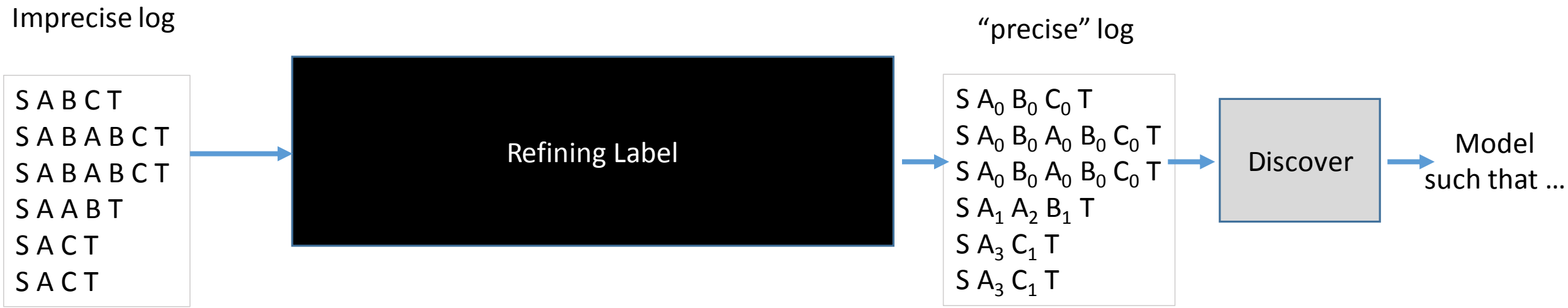
<s $a_3b_3c_3a_4$ e>
<s $c_2b_2a_2$ e>
<s $c_1a_1b_1$ e>



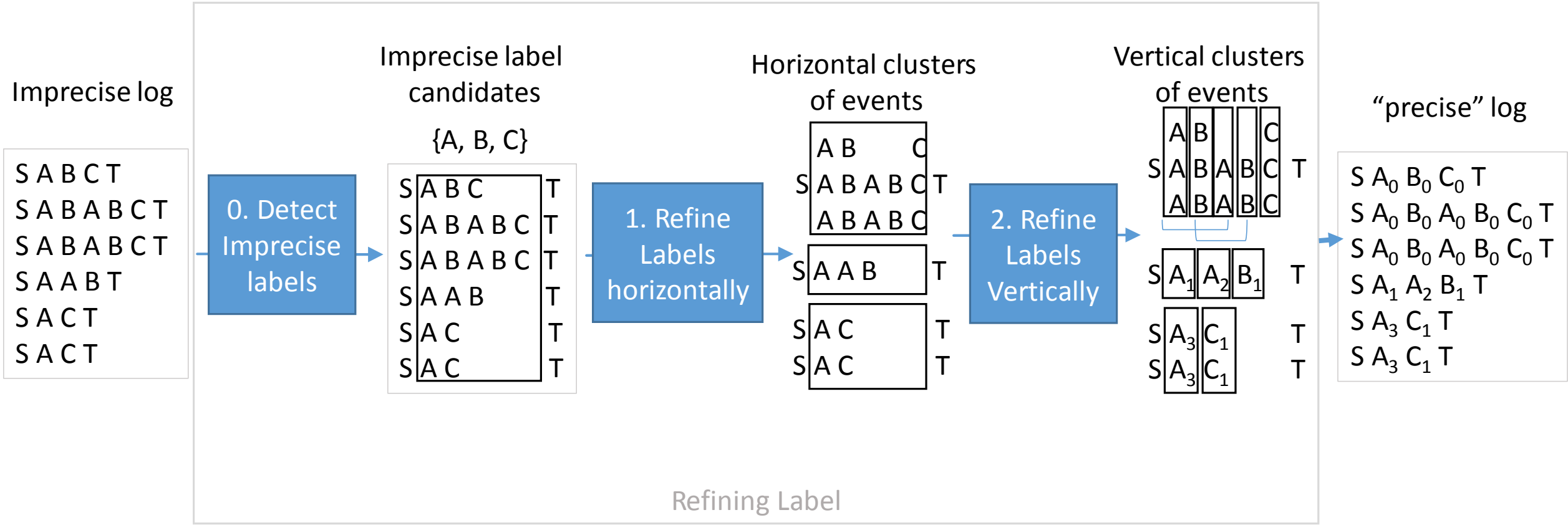




Research Problem

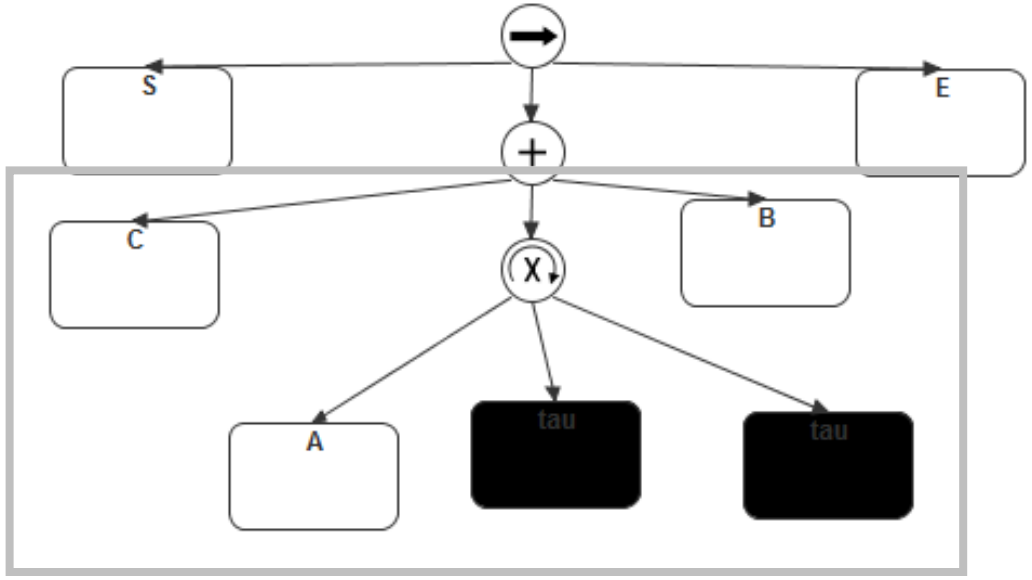


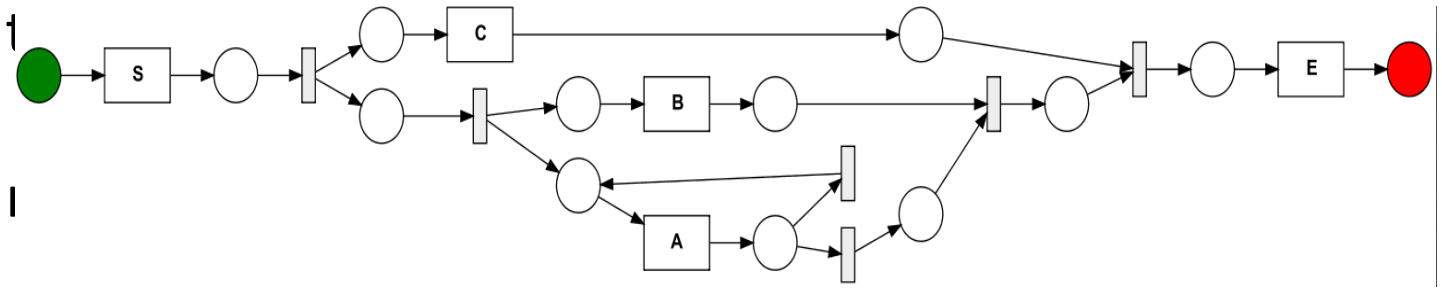
General Framework



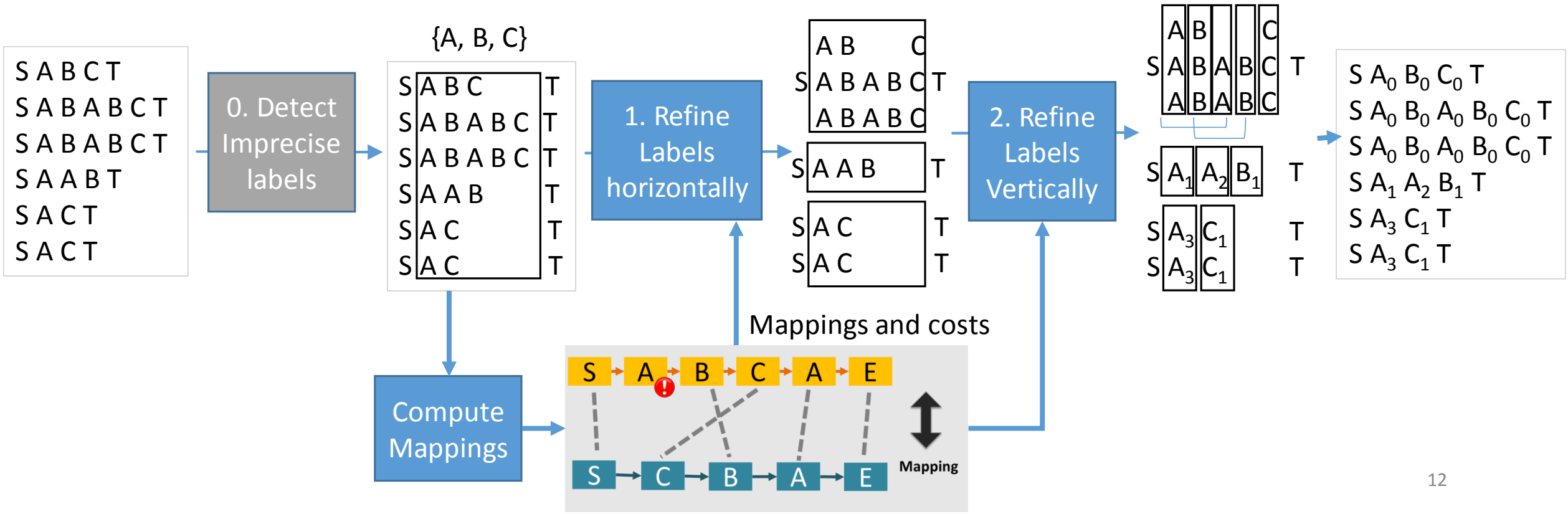
Detect Imprecise Label Candidates

- Many ways…
  - 1) Oracle, known as domain expert
  - 2) Using inductive miner to parse log and find imprecise nodes
    e.g. (flower) loops, AND-splits/join

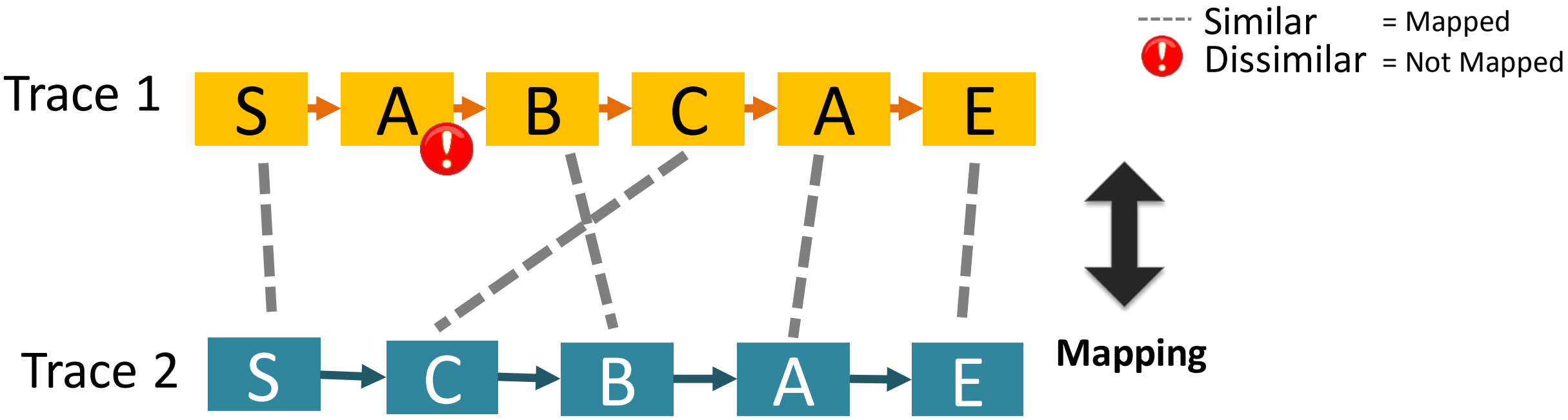




General Framework



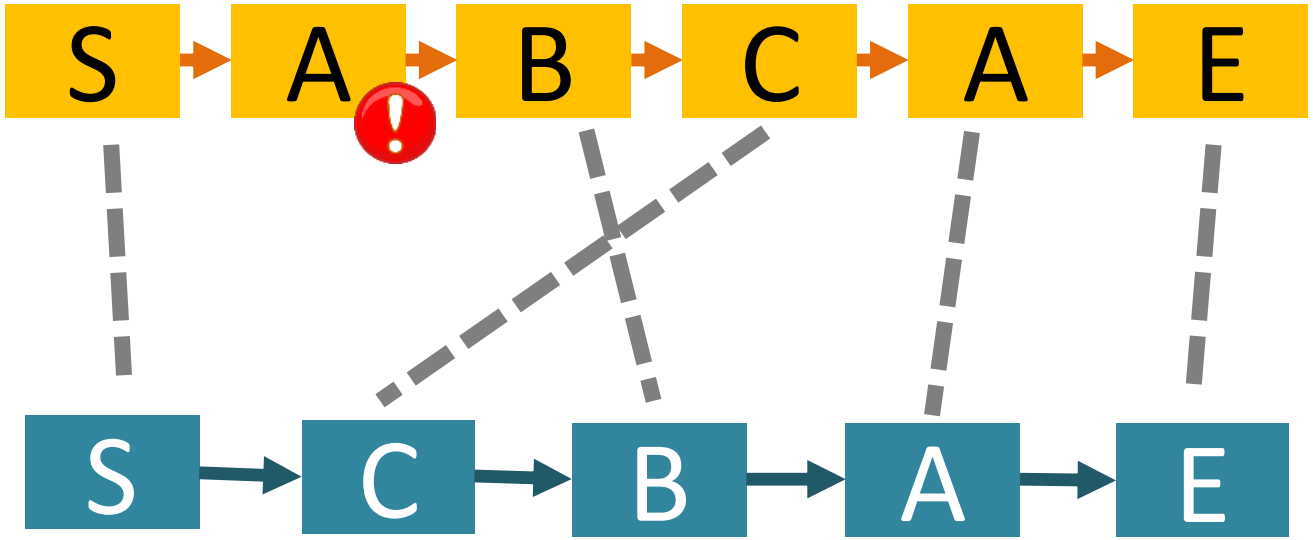
Mapping : Similar and Dissimilar Events



Quantify Mappings

Using cost function to quantify dissimilarity

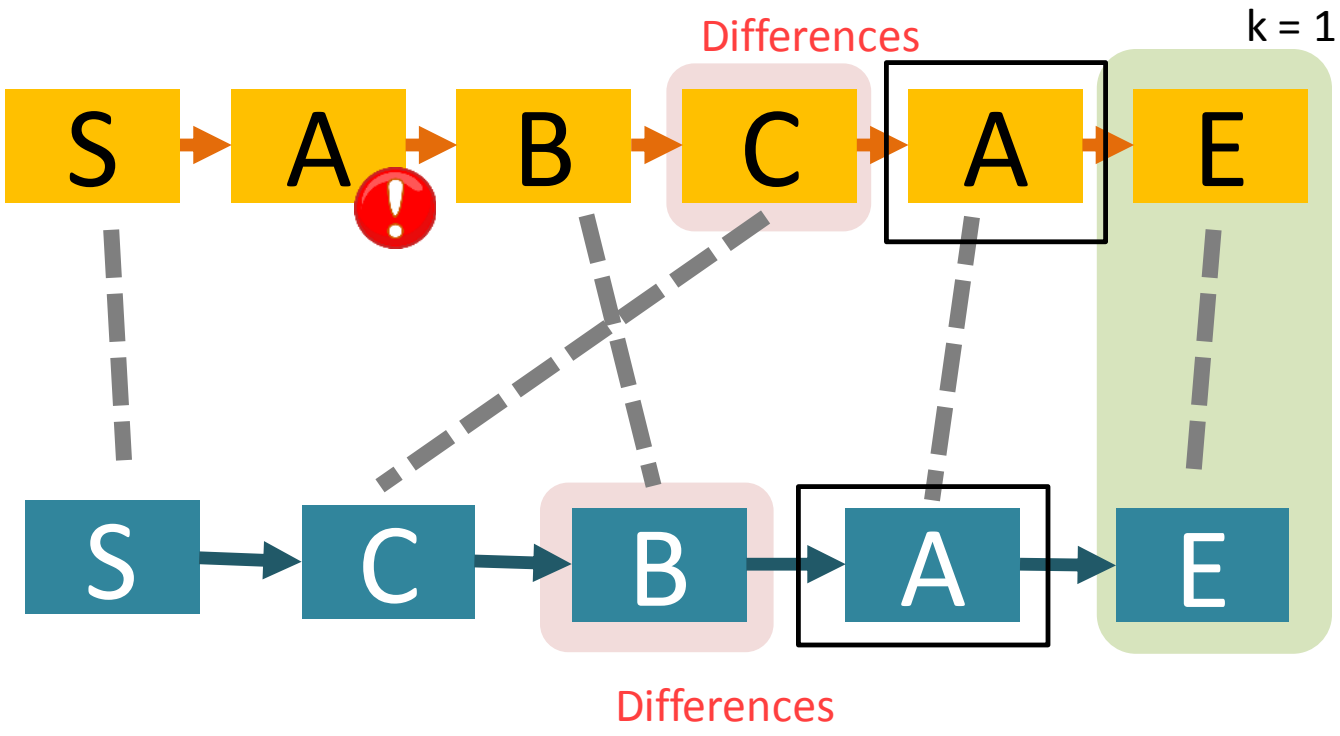
(1) Differences in k-neighbors



Quantify Mappings

Using cost function to quantify dissimilarity

(1) Differences in k-neighbors

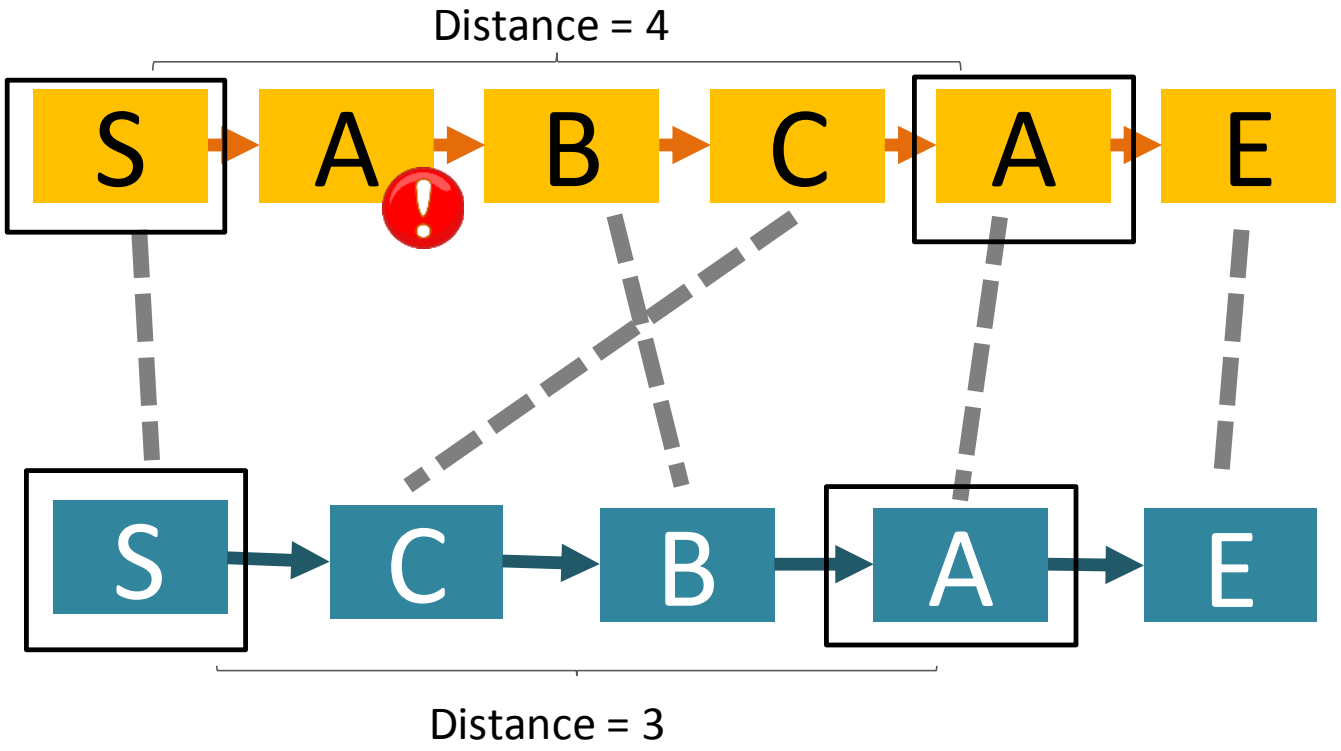


Quantify Mappings

Using cost function to quantify dissimilarity

(1) Differences in k-neighbors

(2) Differences in structure



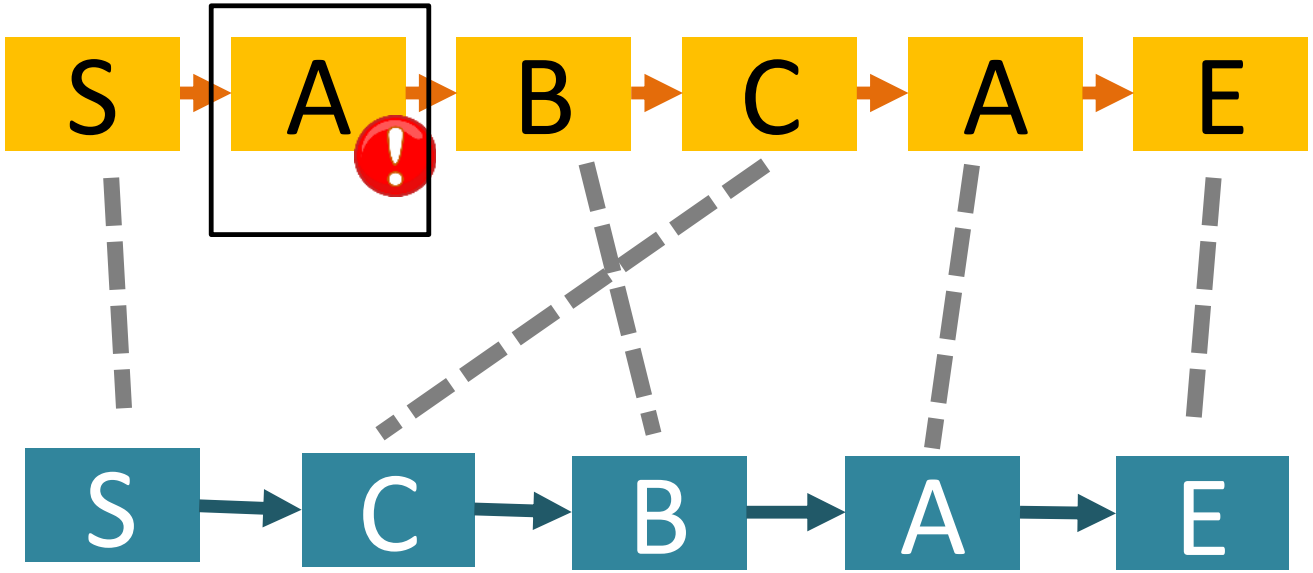
Quantify Mappings

Using cost function to quantify dissimilarity

(1) Differences in k-neighbors

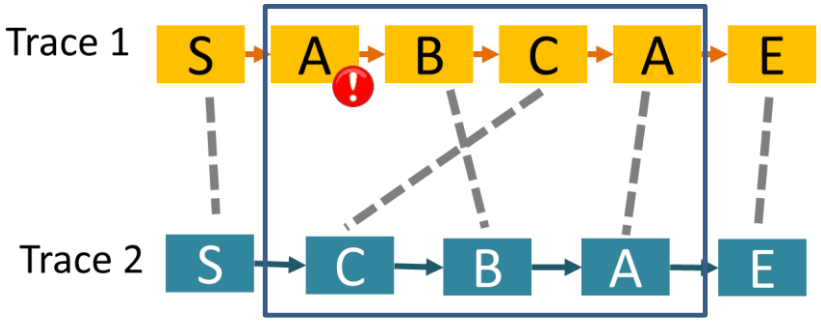
(2) Differences in structure

(3) #Dissimilar events

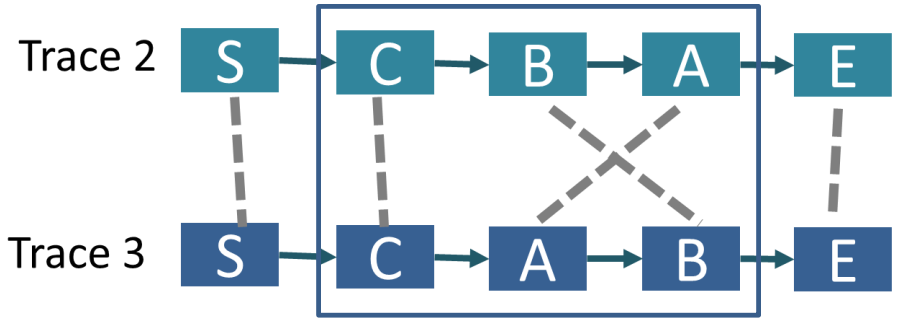


Quantify Mappings

- Using cost function to quantify dissimilarity
  - Optimal mapping captured most similar pairs
- Compute cost for the imprecise label candidates only
- Using existing algorithms

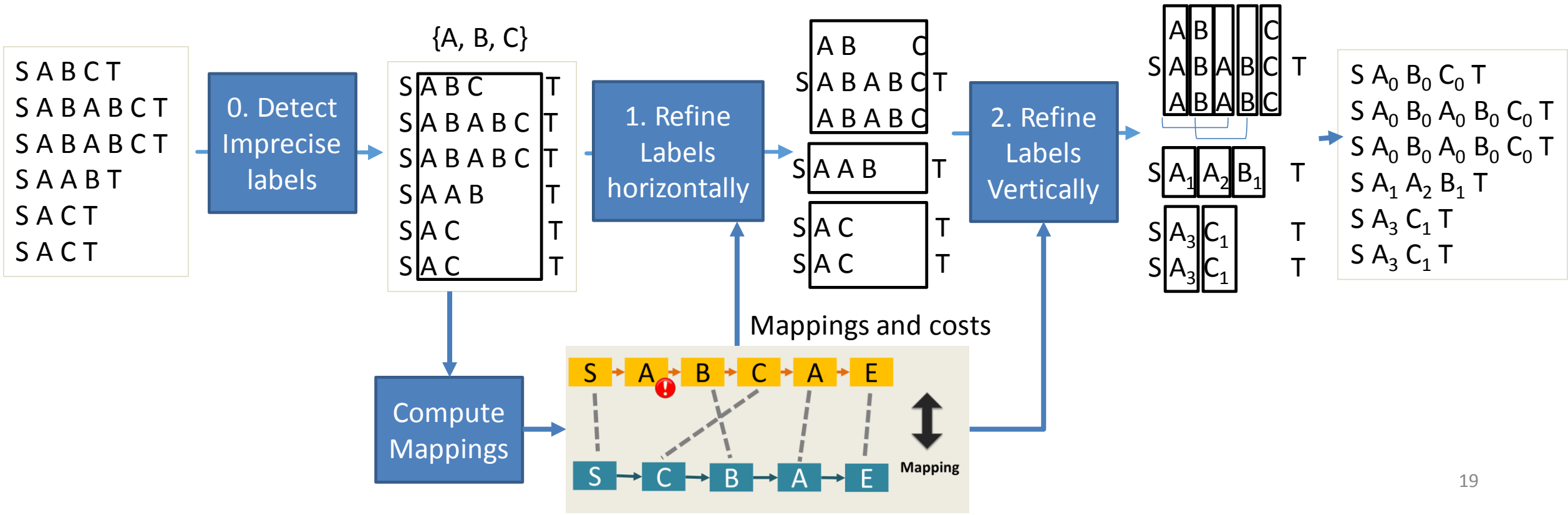


Cost = 15



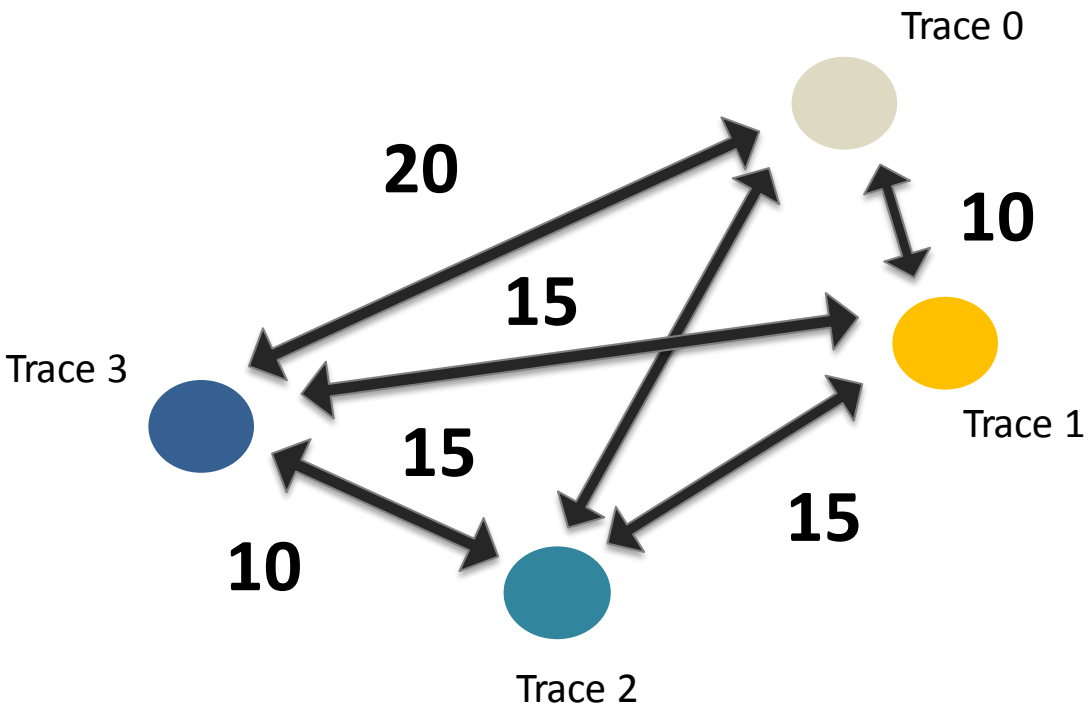
Cost = 10

General Approach



1. Refine horizontally using cost (3D)

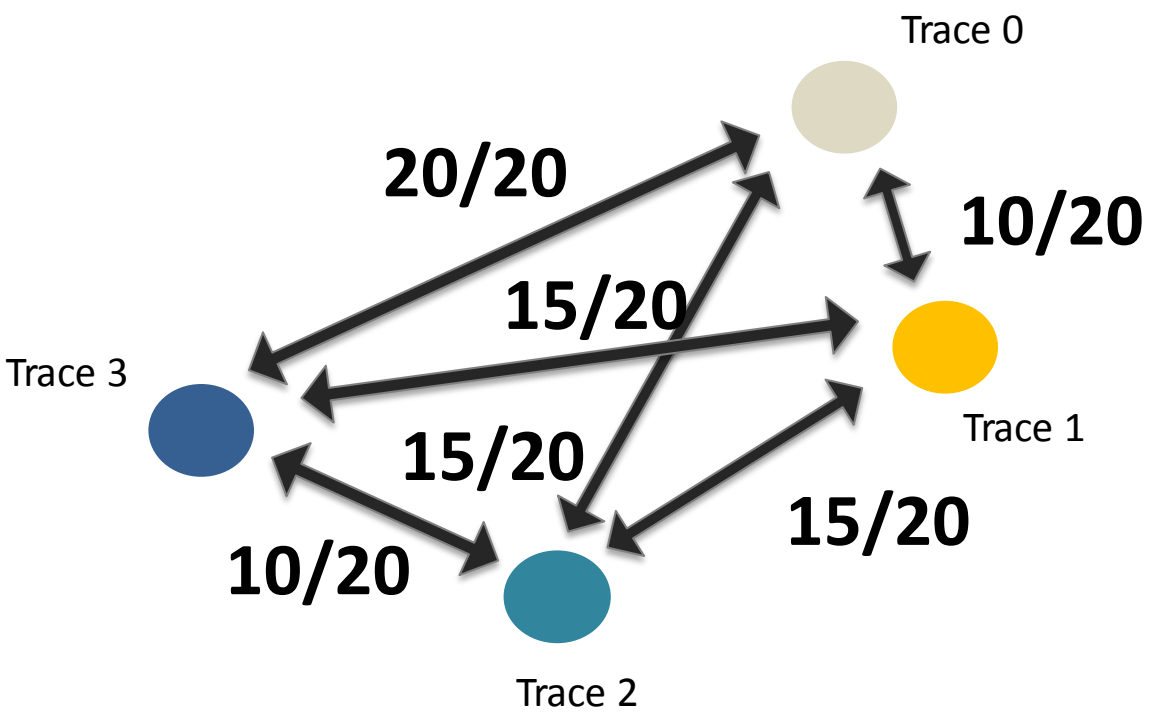
a) Normalize costs between 0 and 1



A trace

0.2 A mapping with normalized cost

1. Refine horizontally using cost (3D)



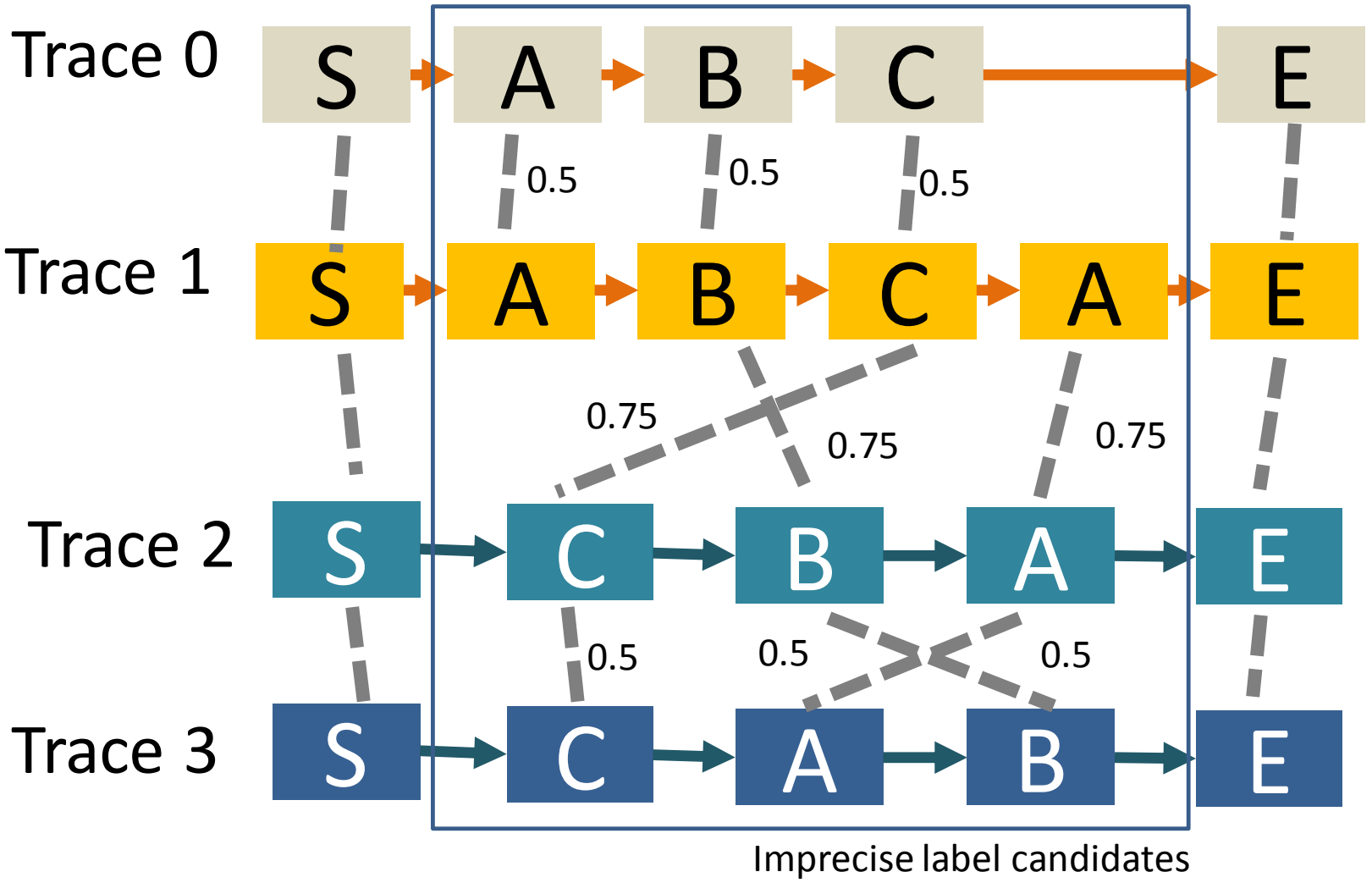
a) Normalize costs between 0 and 1

b) Set cost threshold, e.g. 0.6

A trace

0.2 A mapping with normalized cost

1. Refine horizontally using cost (2D)

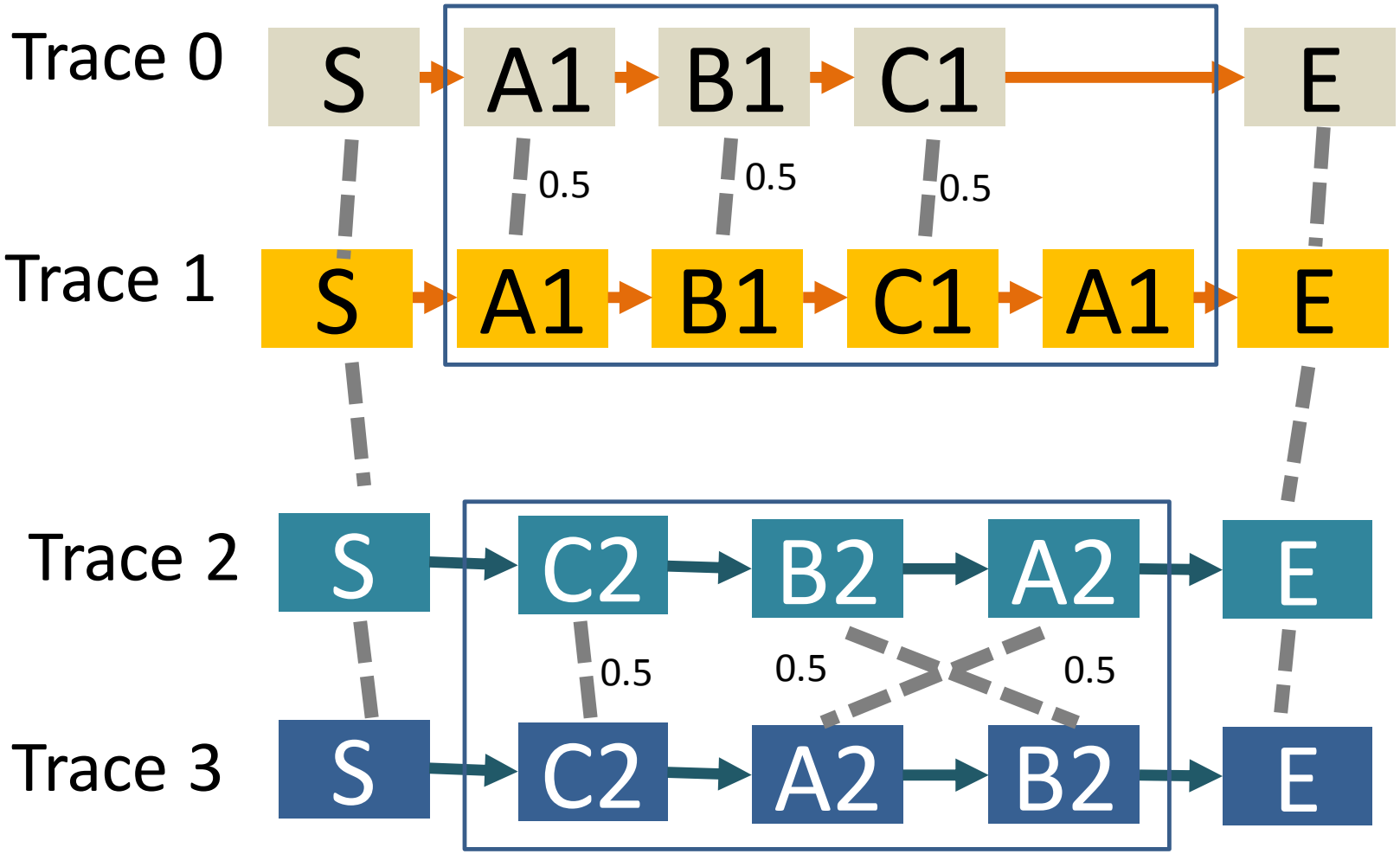


a) Normalize costs between 0 and 1

b) Set cost threshold, e.g. 0.6

c) Keep edges if cost ≤ cost threshold

1. Refine horizontally using cost (2D)

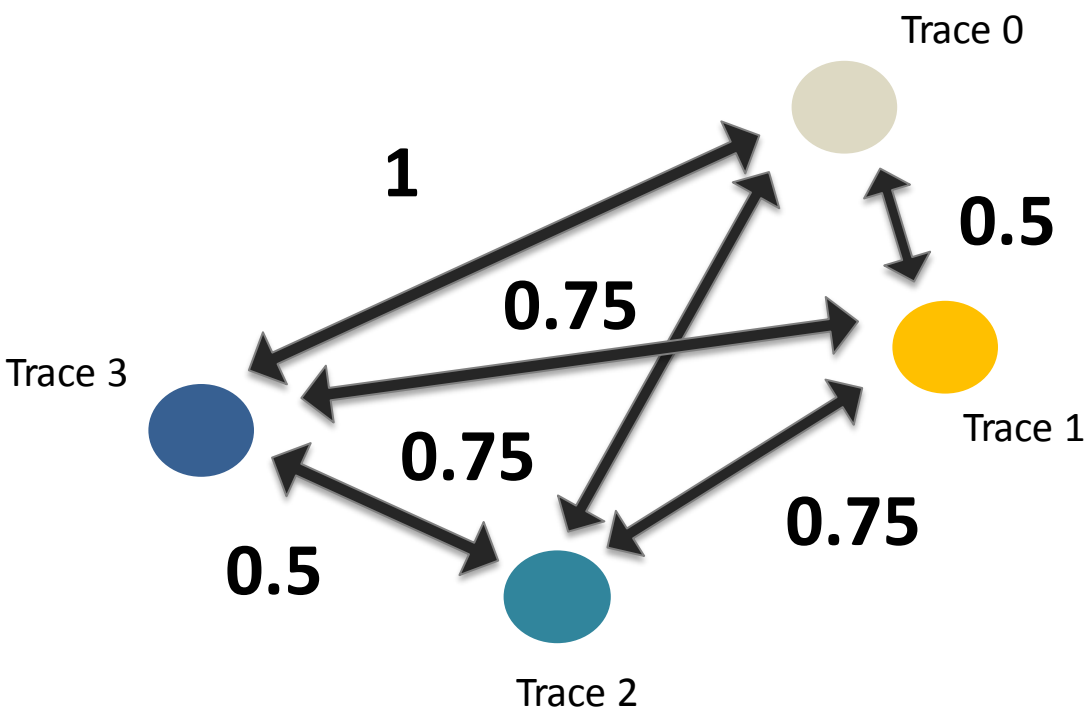


a) Normalize costs between 0 and 1

b) Set cost threshold, e.g. 0.6

c) Keep edges if cost ≤ cost threshold

1. Refine horizontally using cost (3D)



a) Normalize costs
between 0 and 1

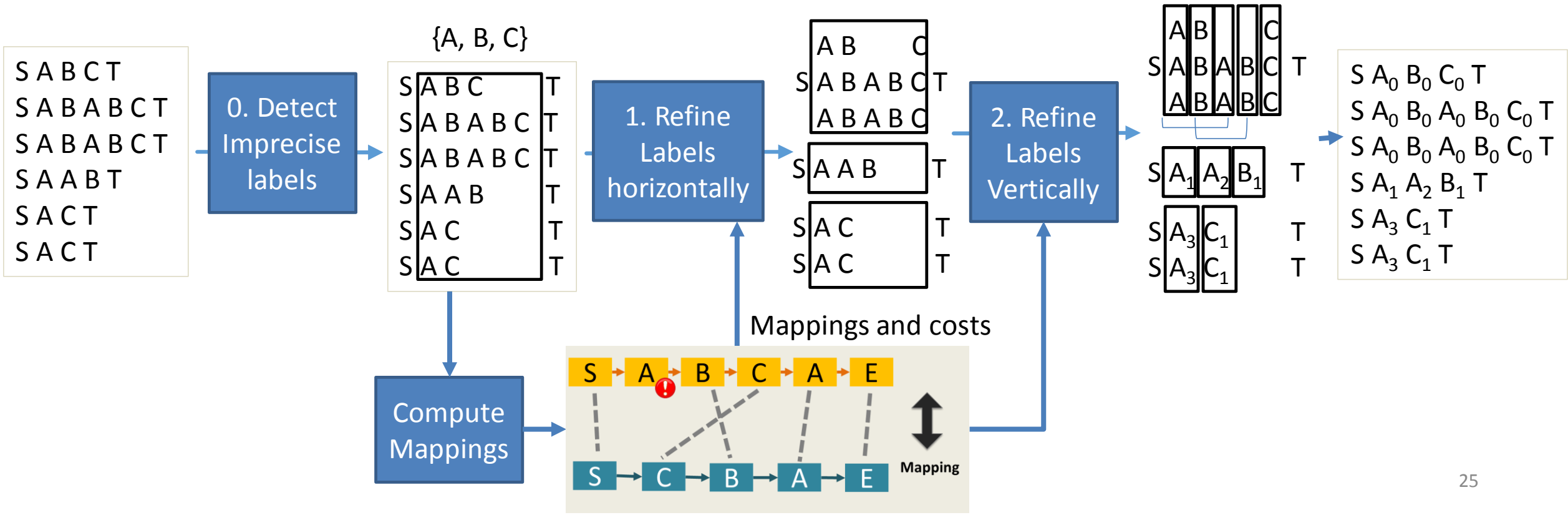
b) Set cost threshold, e.g. 0.6

c) Keep edges
if cost ≤ cost threshold

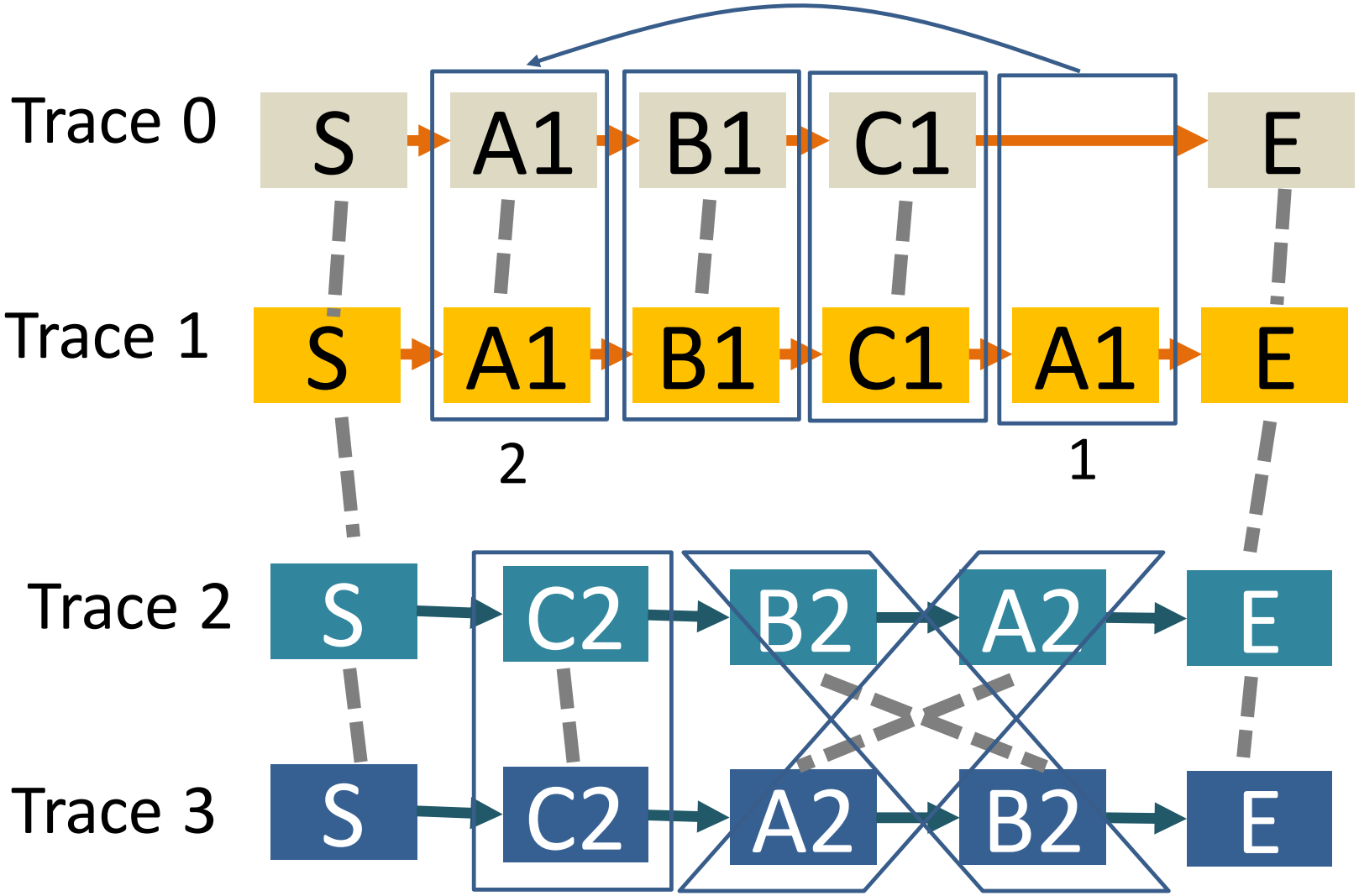
A trace

0.2 A mapping
with normalized cost

General Approach

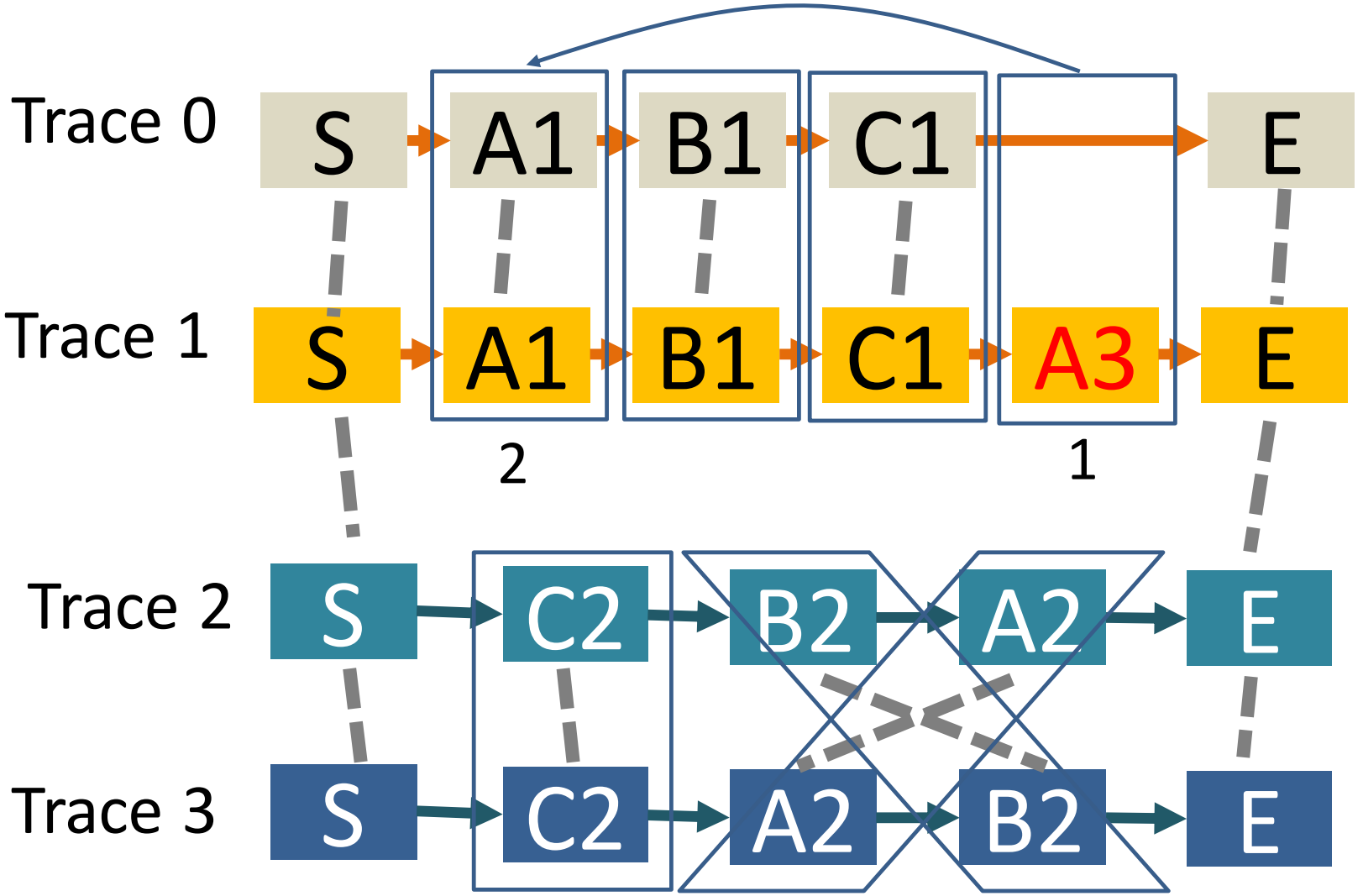


2. Refine labels vertically using Frequency



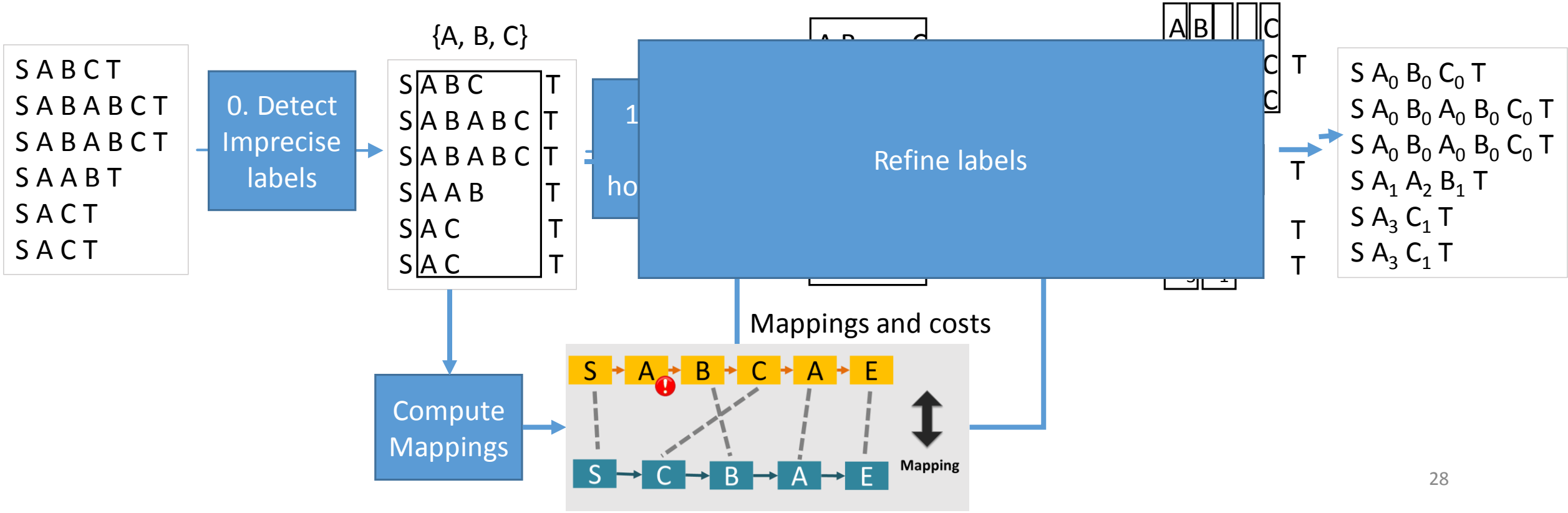
Unfolding threshold = 60%

2. Refine labels vertically using Frequency

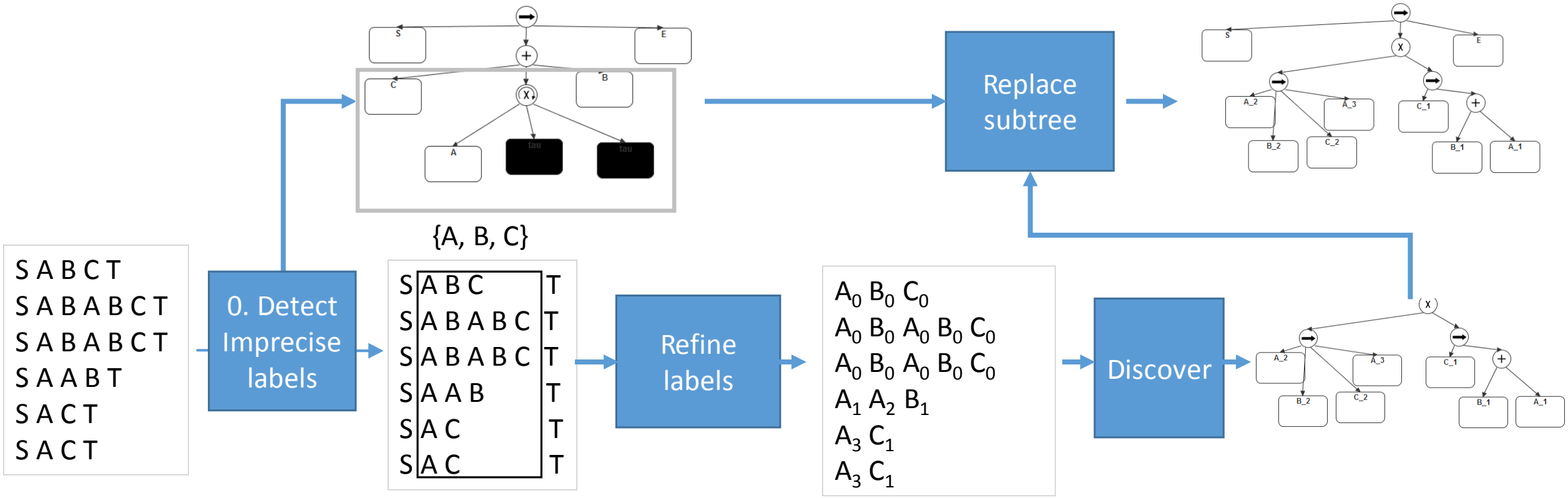


Unfolding threshold = 40%

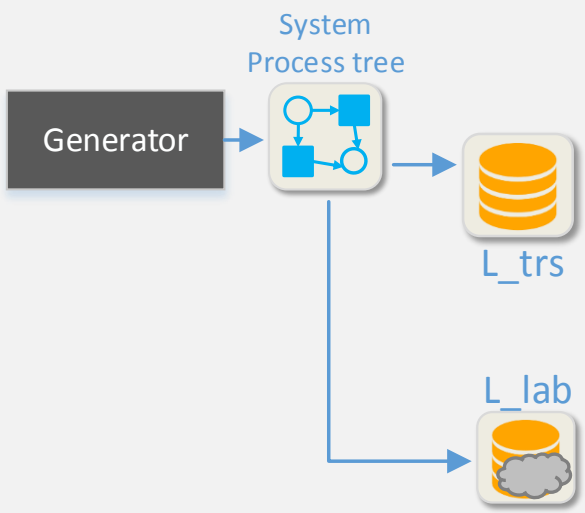
General Approach



Local Refining - Using Inductive Miner

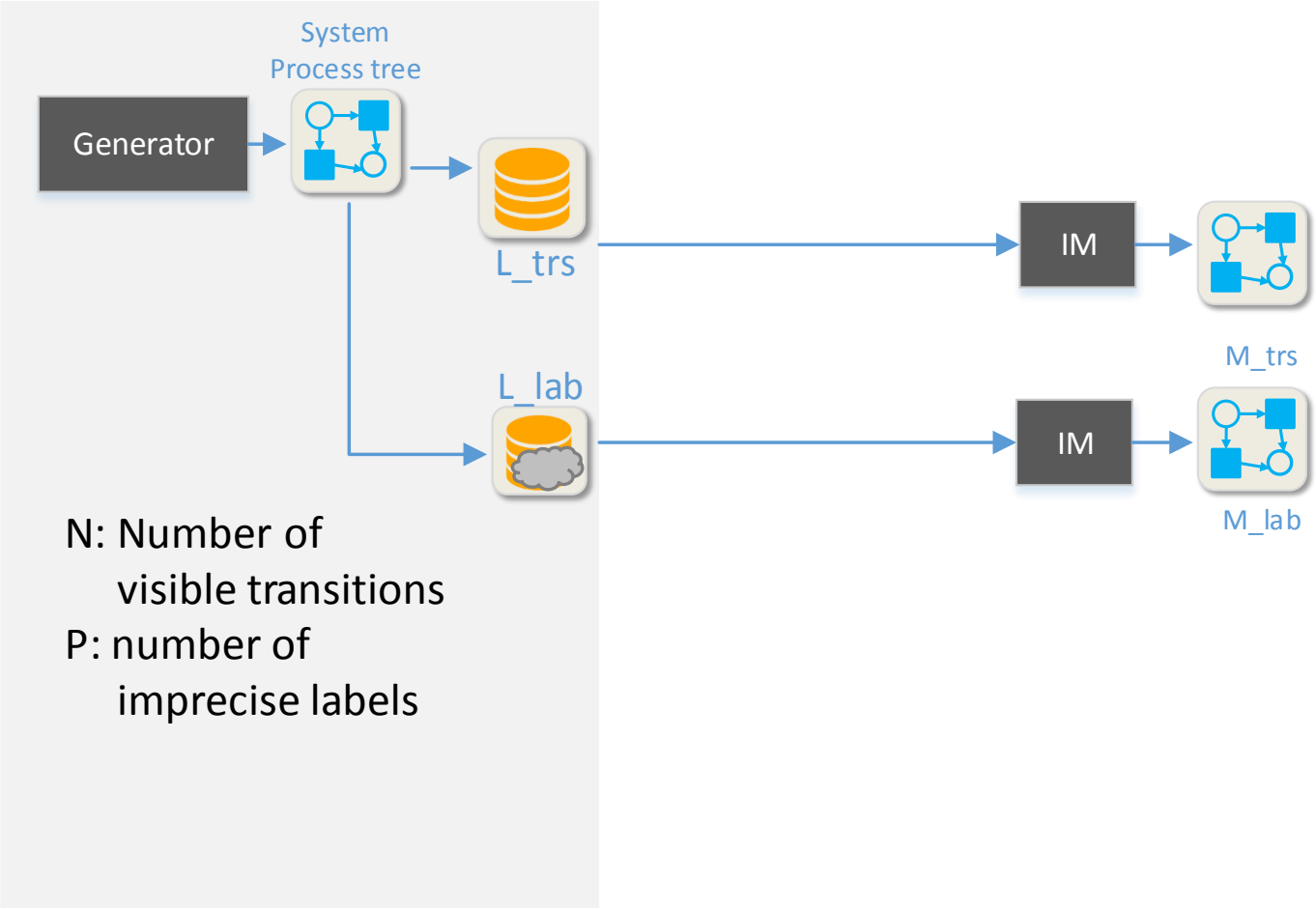


Evaluation (A1)

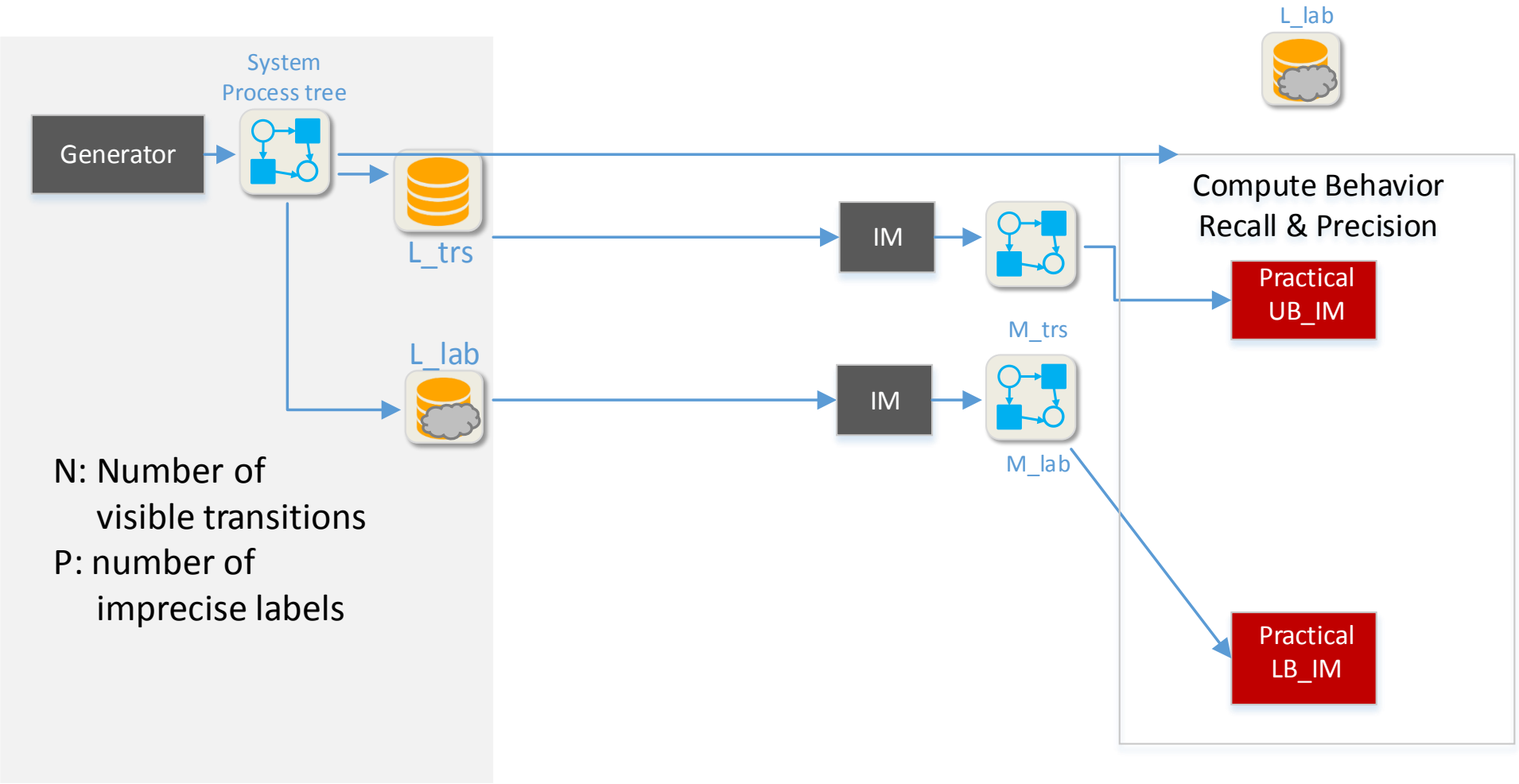


N: Number of
visible transitions
P: number of
imprecise labels

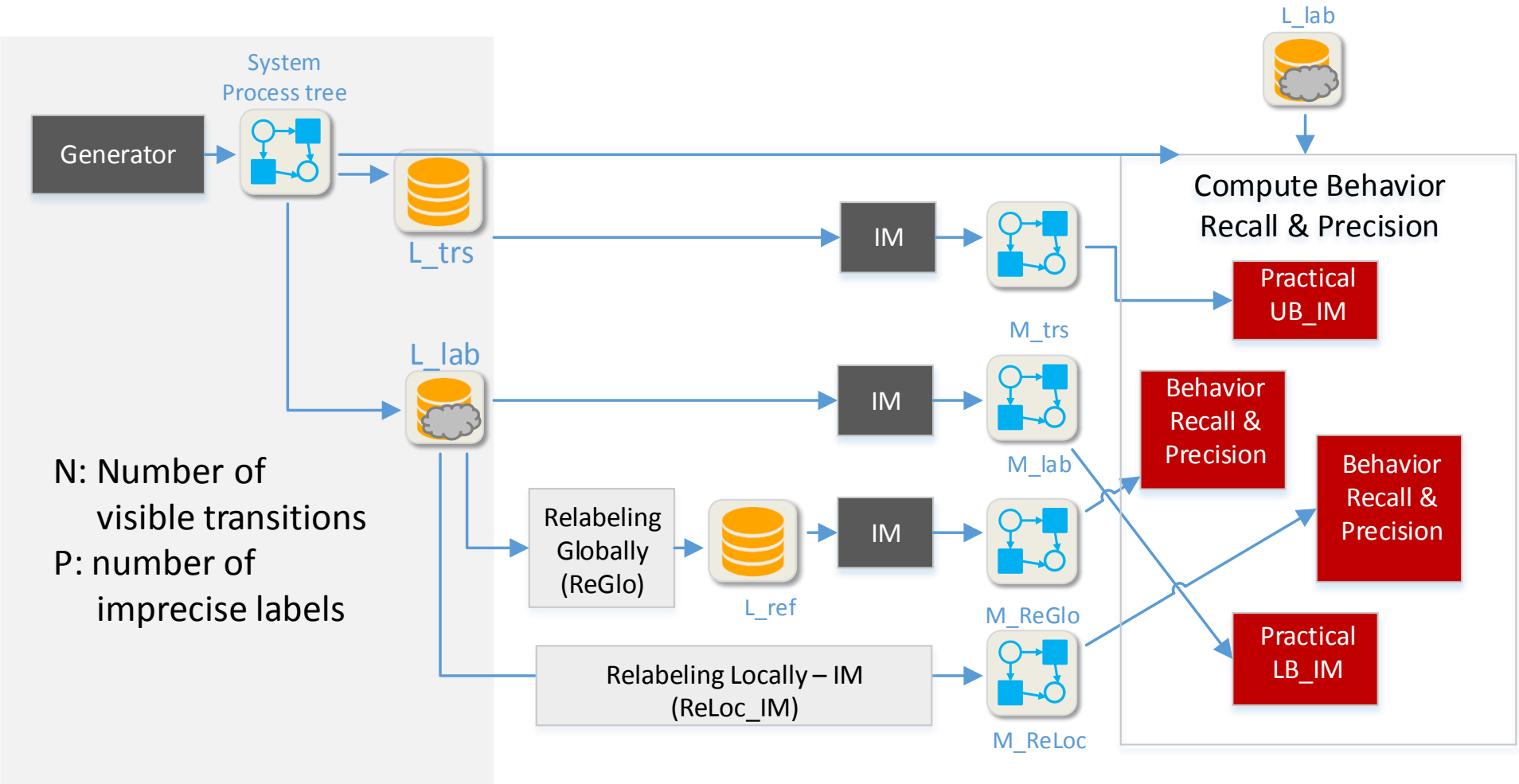
Evaluation (A1)



Evaluation (A1)

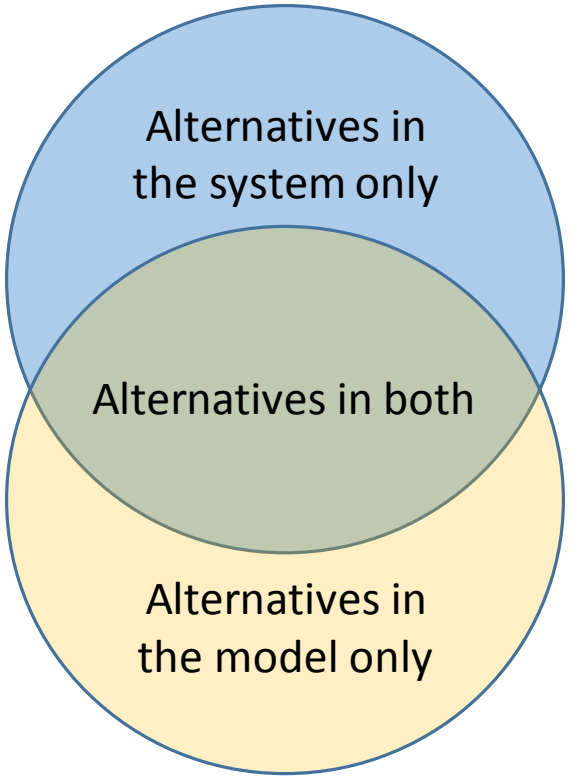


Evaluation (A1)

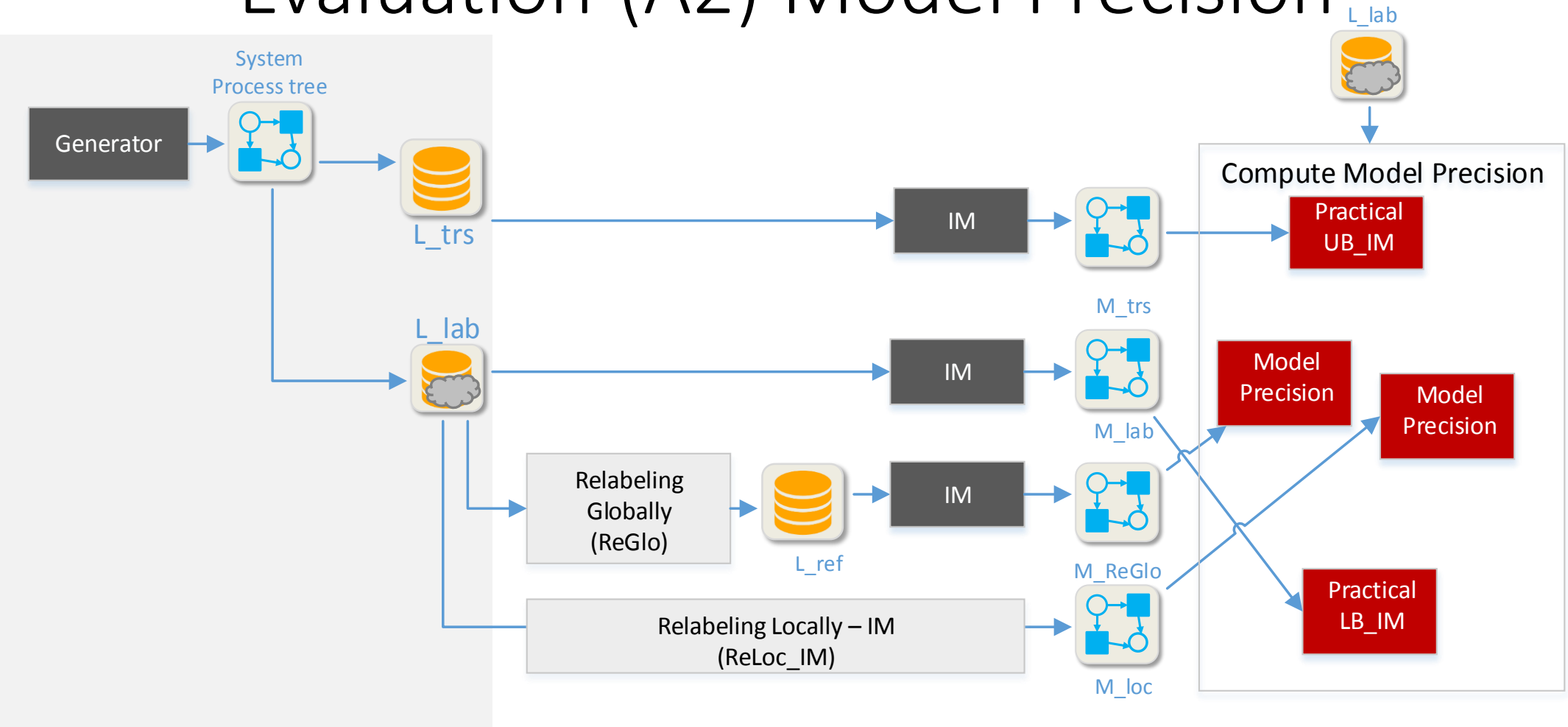


Evaluation (A1) Behavior Precision and Recall

- For each event,
  compare behavior in the system
  and in the discovered model

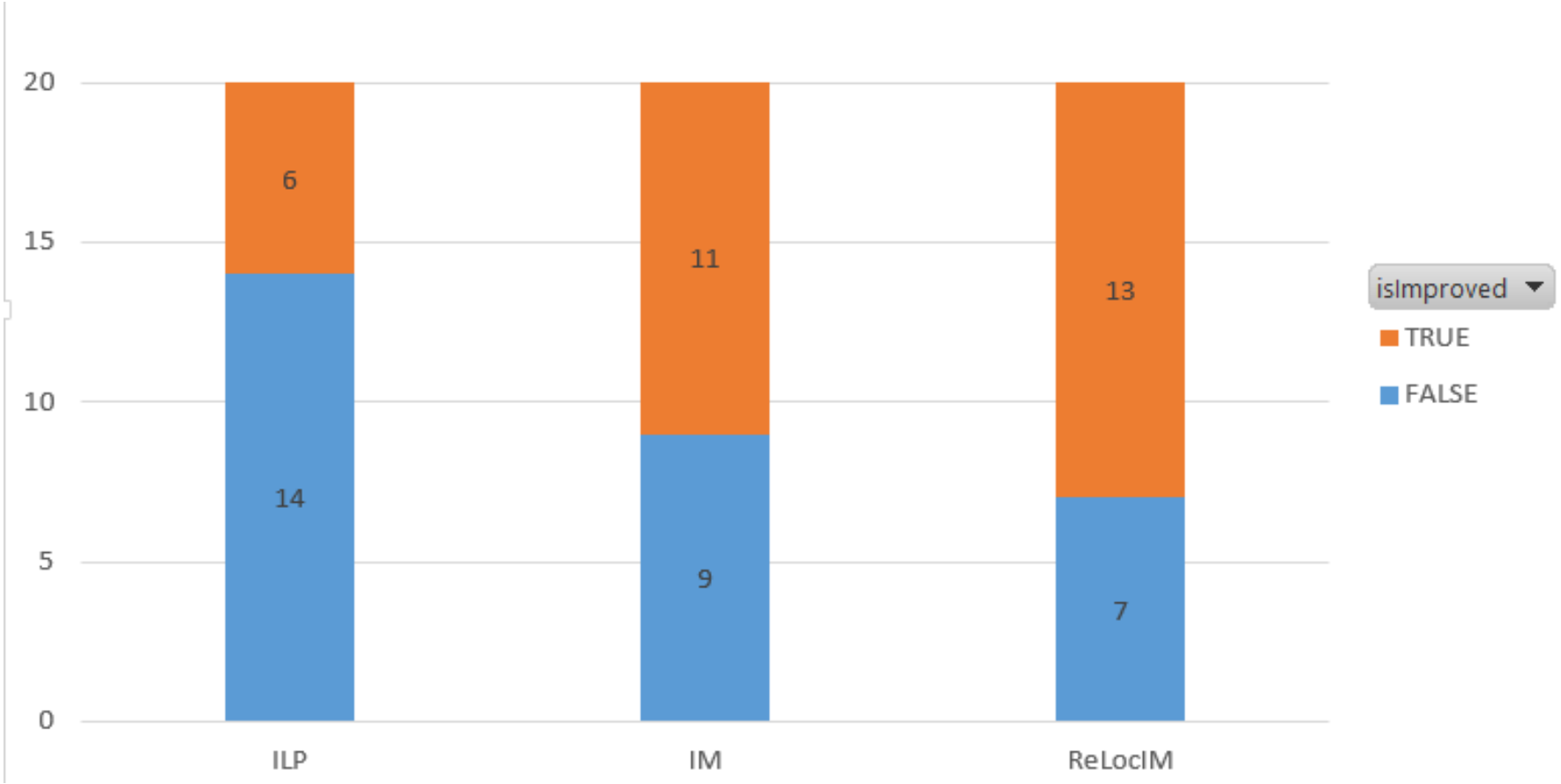


Evaluation (A2) Model Precision



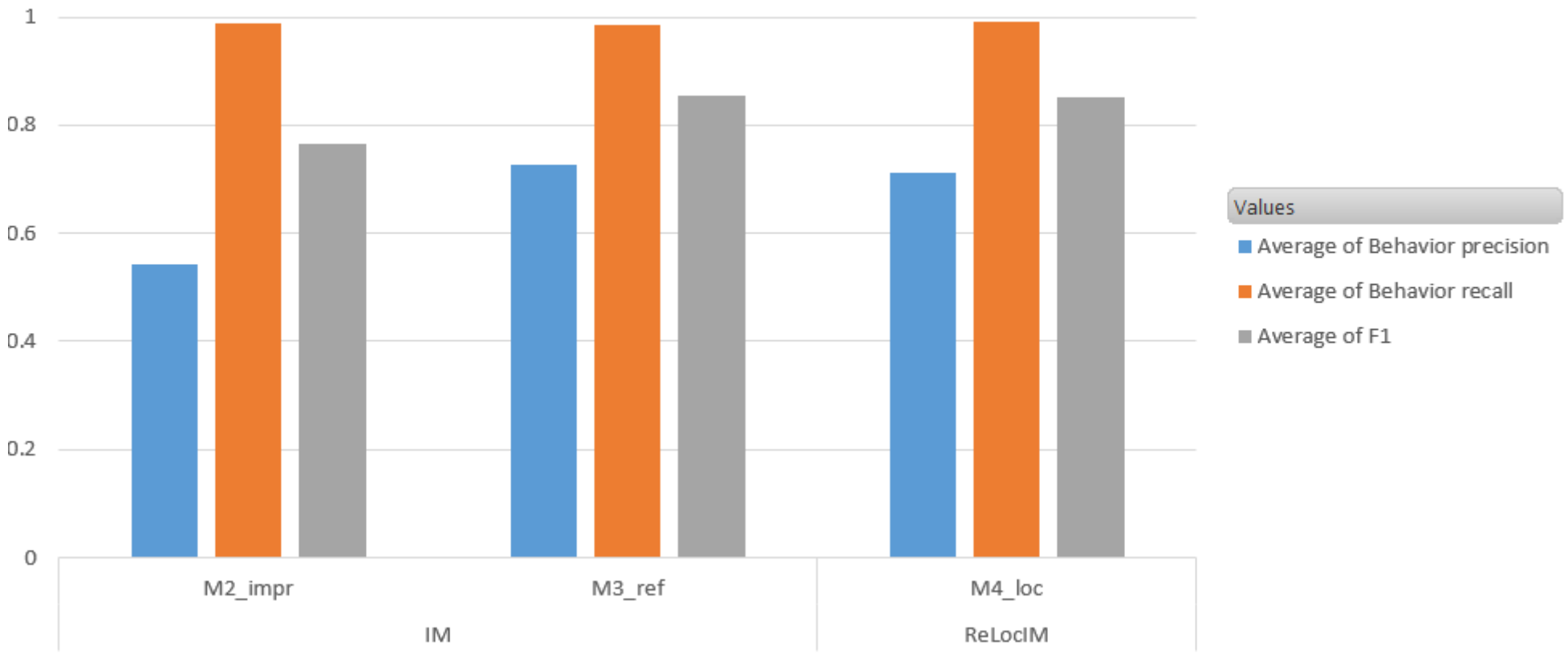
EXP1) N20(x20),P4, (Oracle Detection)

- Number of models improved



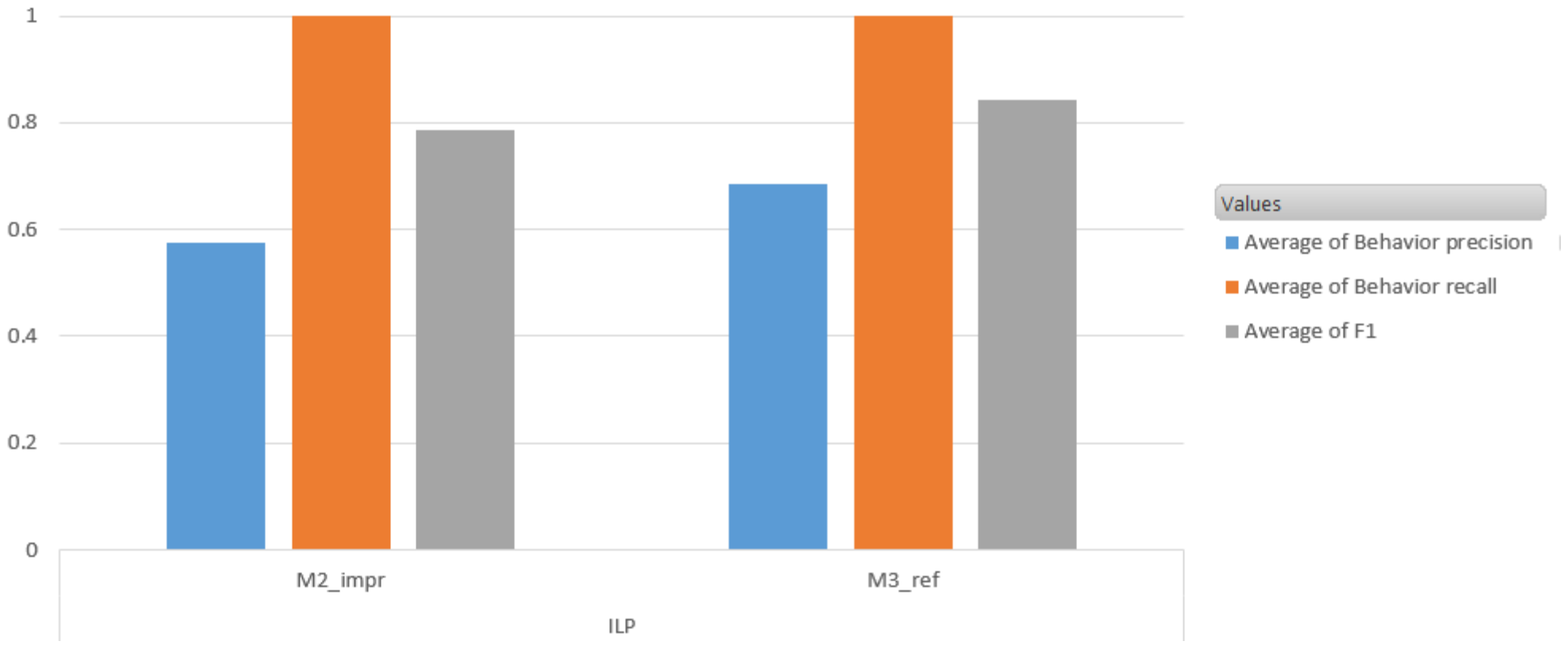
EXP1) N20(x20),P4, (Oracle Detection)

(A1) behavior recall and precision absolute improvements (IM)



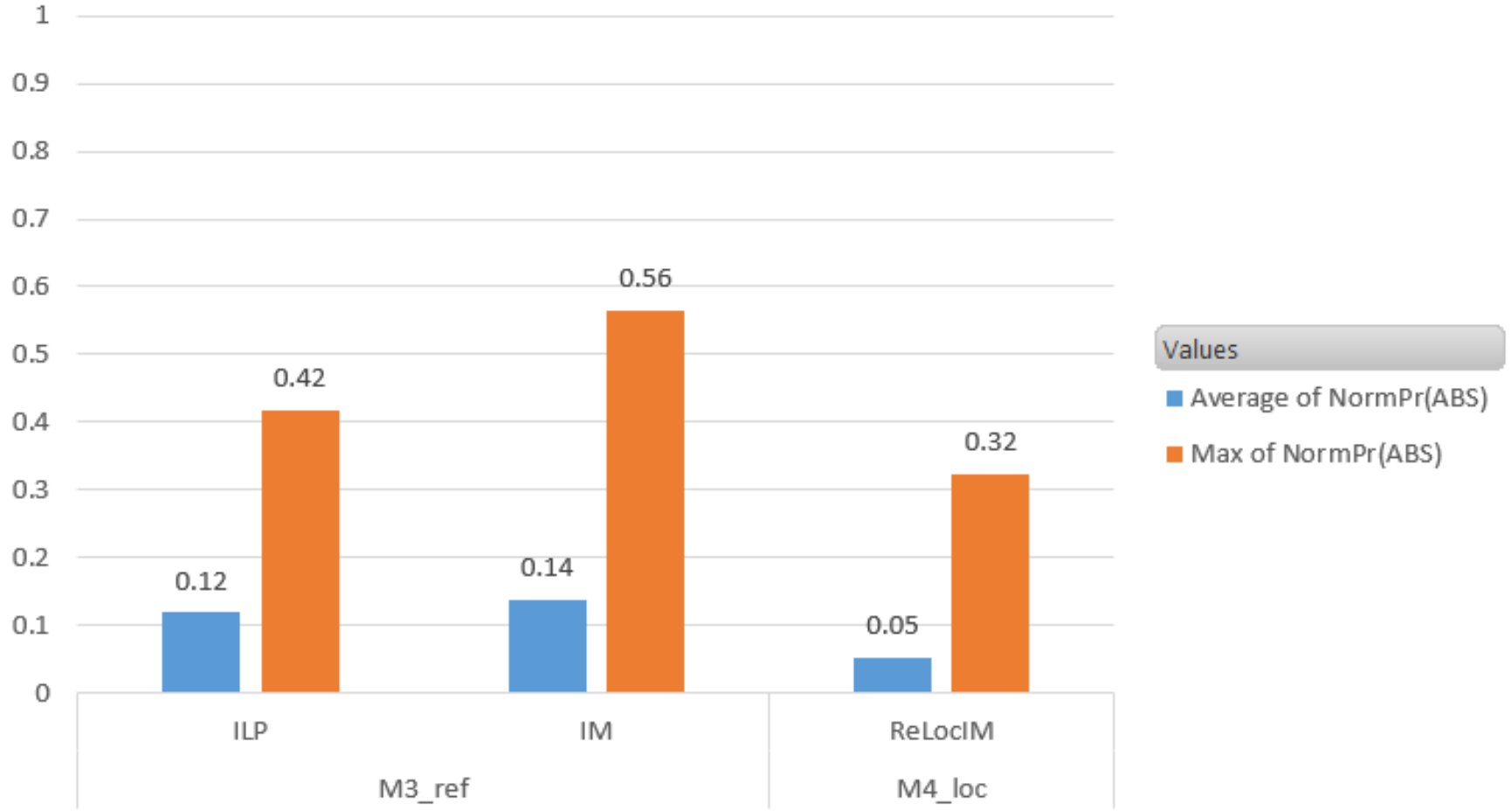
EXP1) N20(x20),P4, (Oracle Detection)

(A1) behavior recall and precision absolute improvements (ILP)



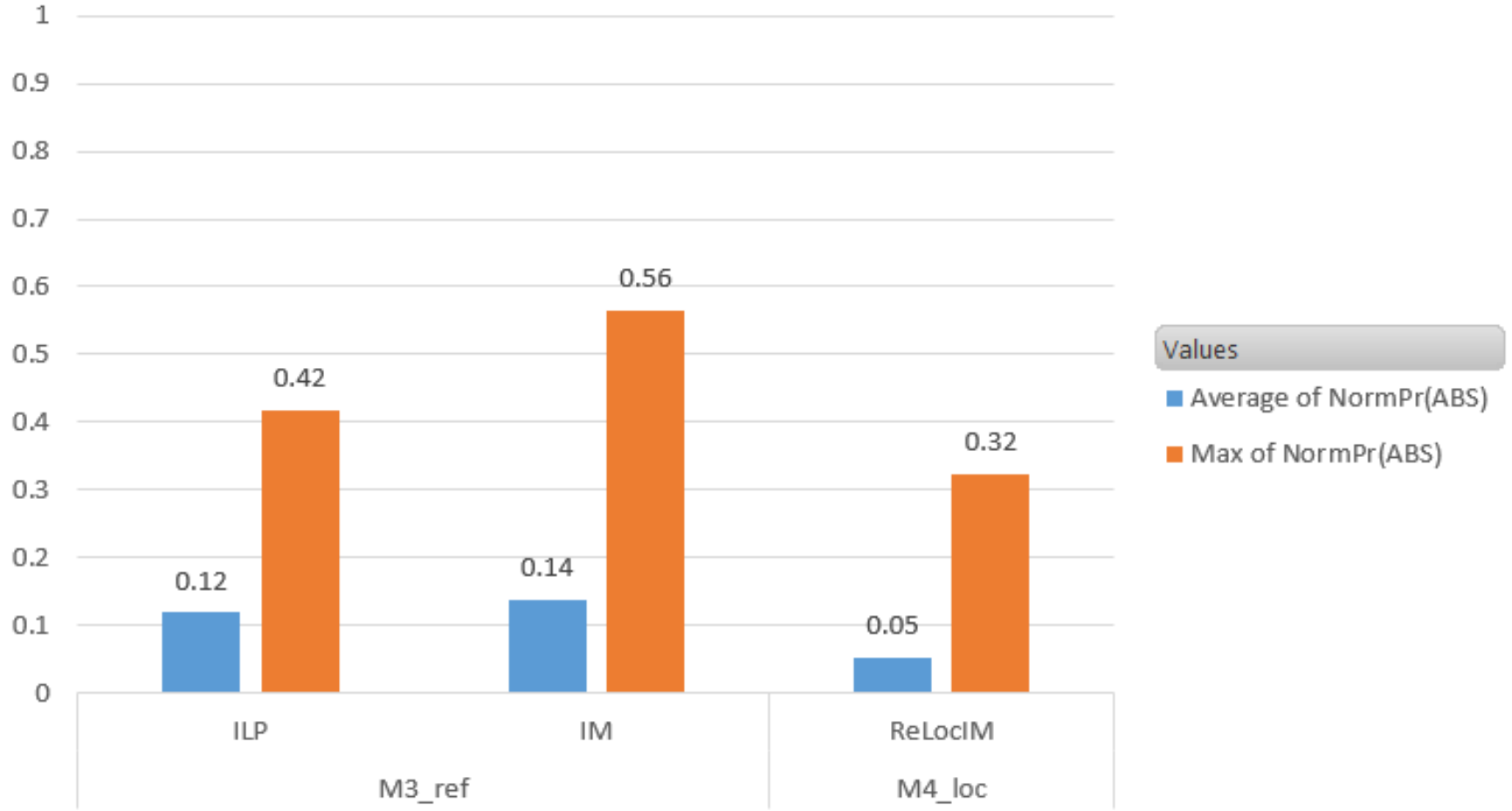
EXP1) N20(x20),P4, (Oracle Detection)

- Absolute Model Precision Improvements



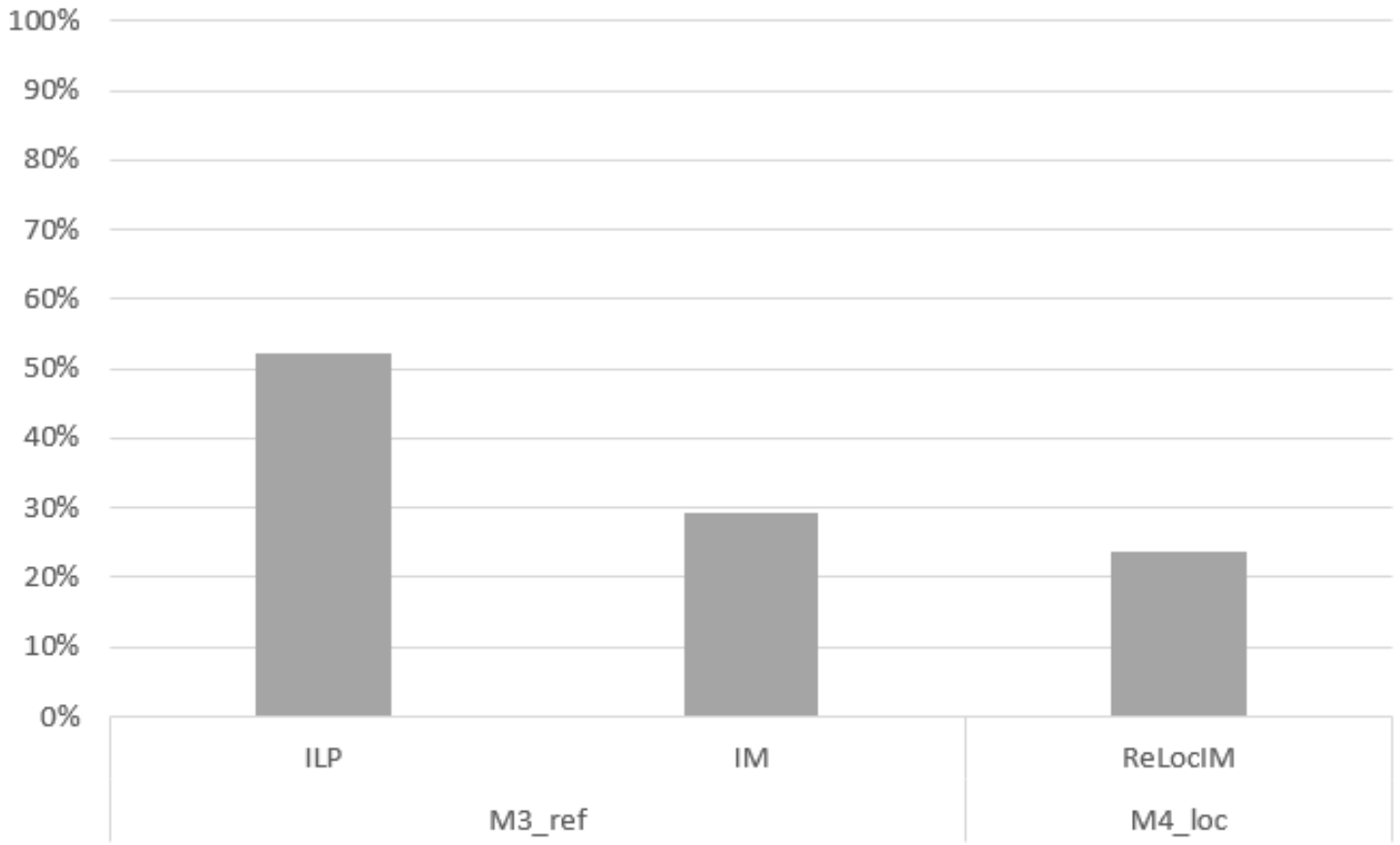
EXP1) N20(x20),P4, (Oracle Detection)

- Absolute Model Precision Improvements

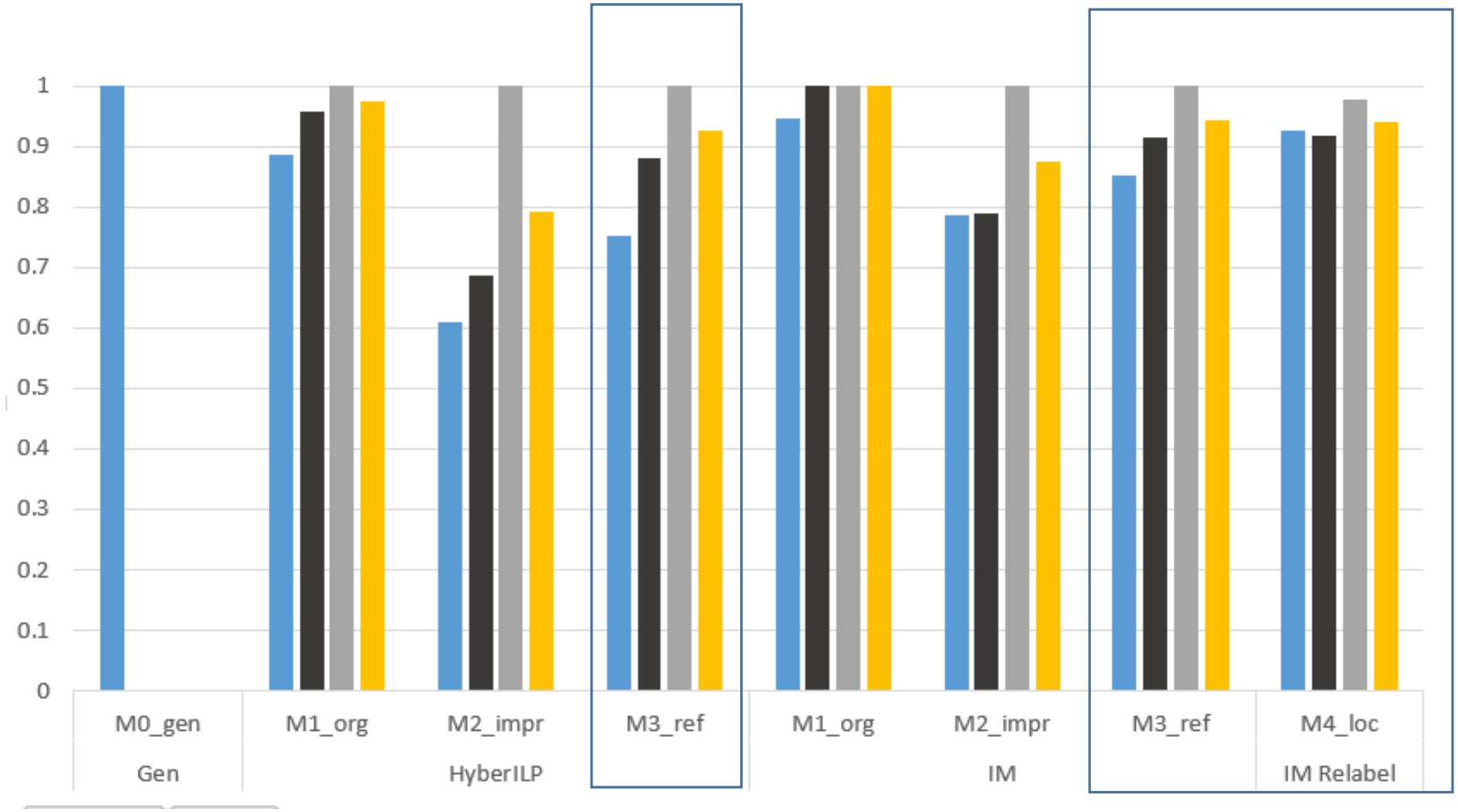


EXP1) N20(x20),P4, (Oracle Detection)

- % of maximal possible improvements achieved

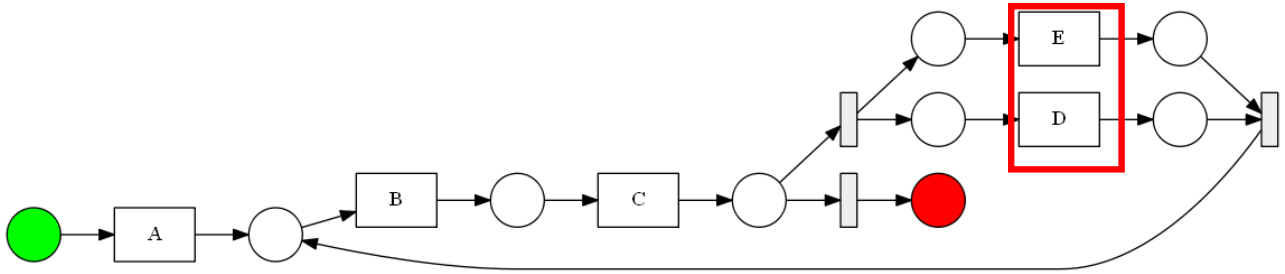


EXP1) N5(x5),P2(x4), (IM Detection)

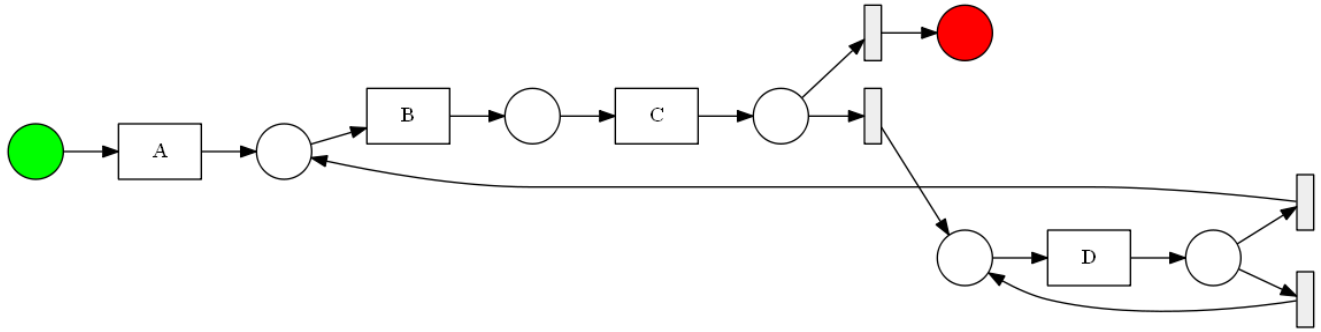


N5(x5),P2(x4), Examples (working)

M_gen

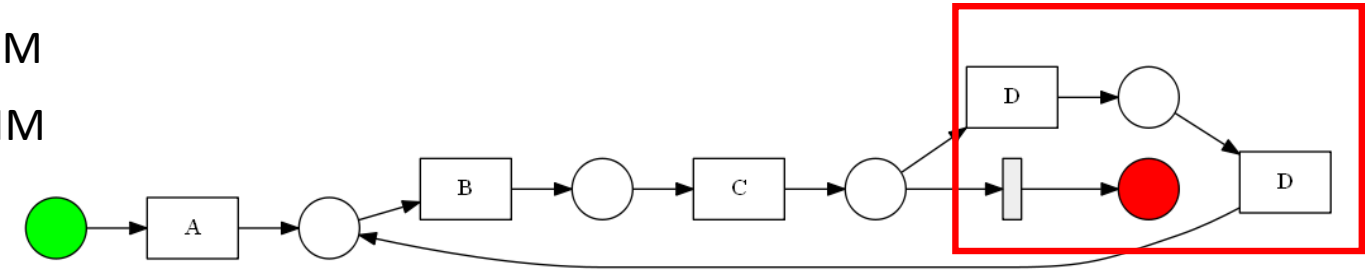


M_impr



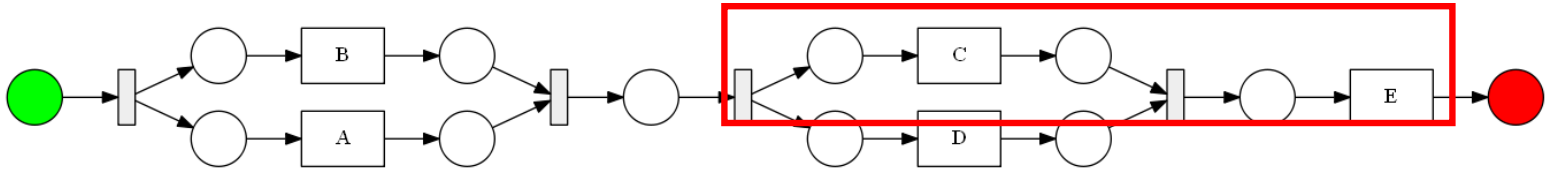
M_ref_IM

M_loc_IM

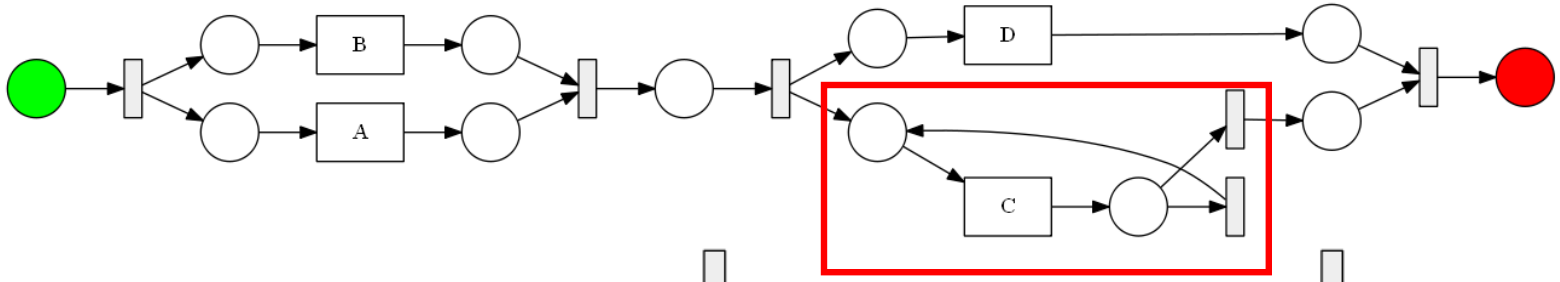


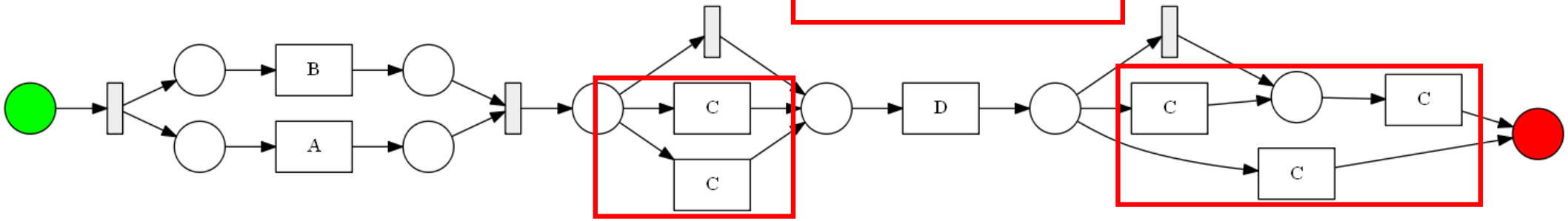
N5(x5),P2(x4), Examples (not working)

M_gen

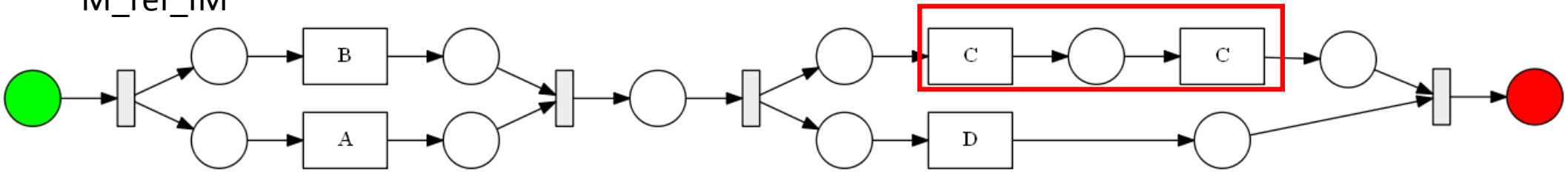


M_impr



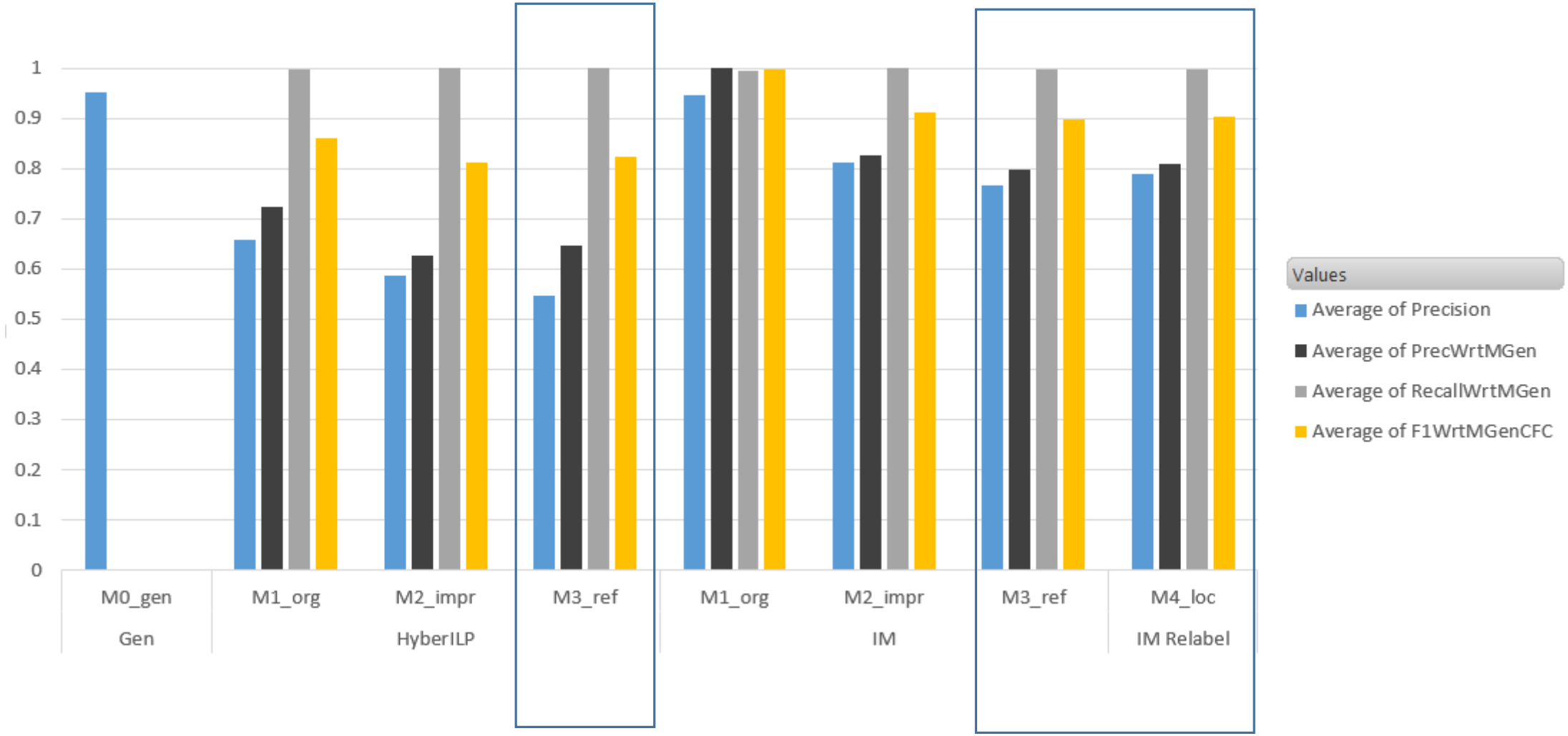


M_ref_IM



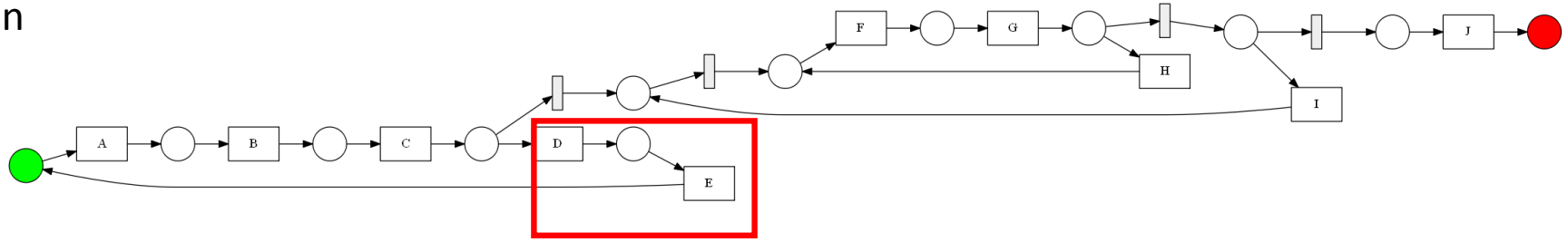
M_loc_IM

N10(x5),P2(x4), (IM Detection)

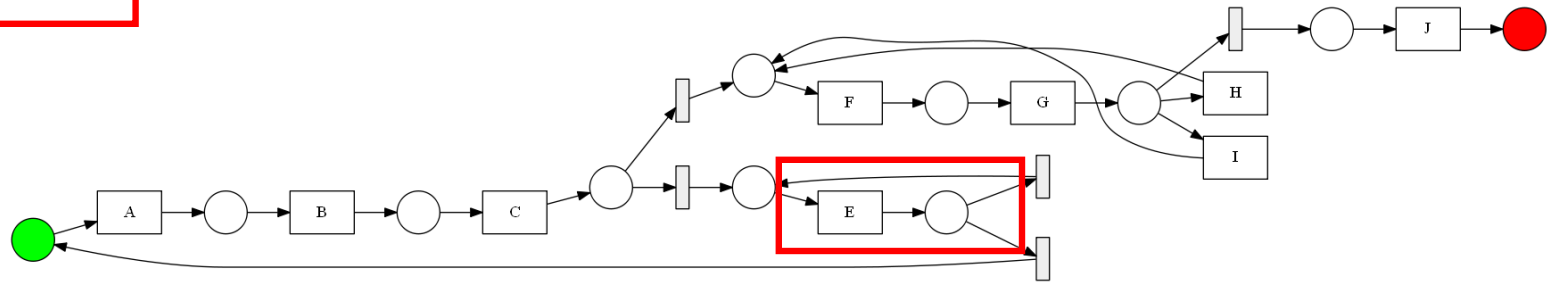


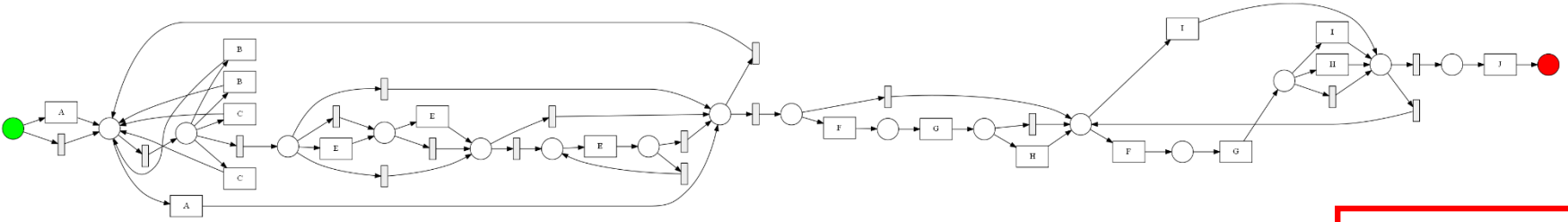
N10(x5),P2(x4), working and not

M_gen

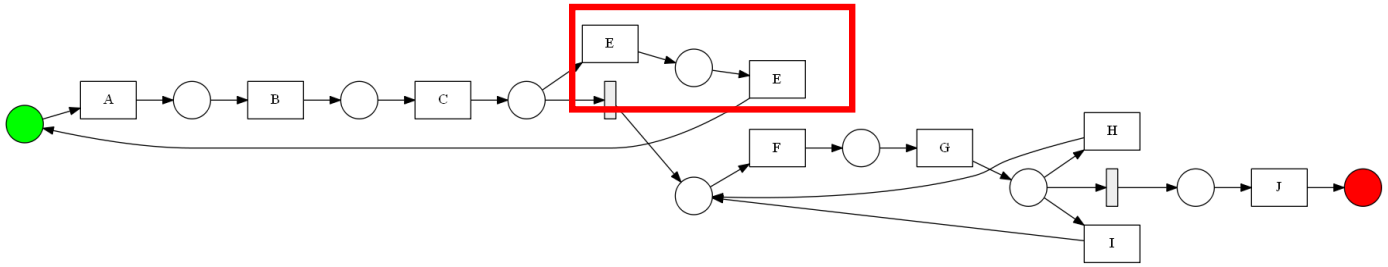


M_impr



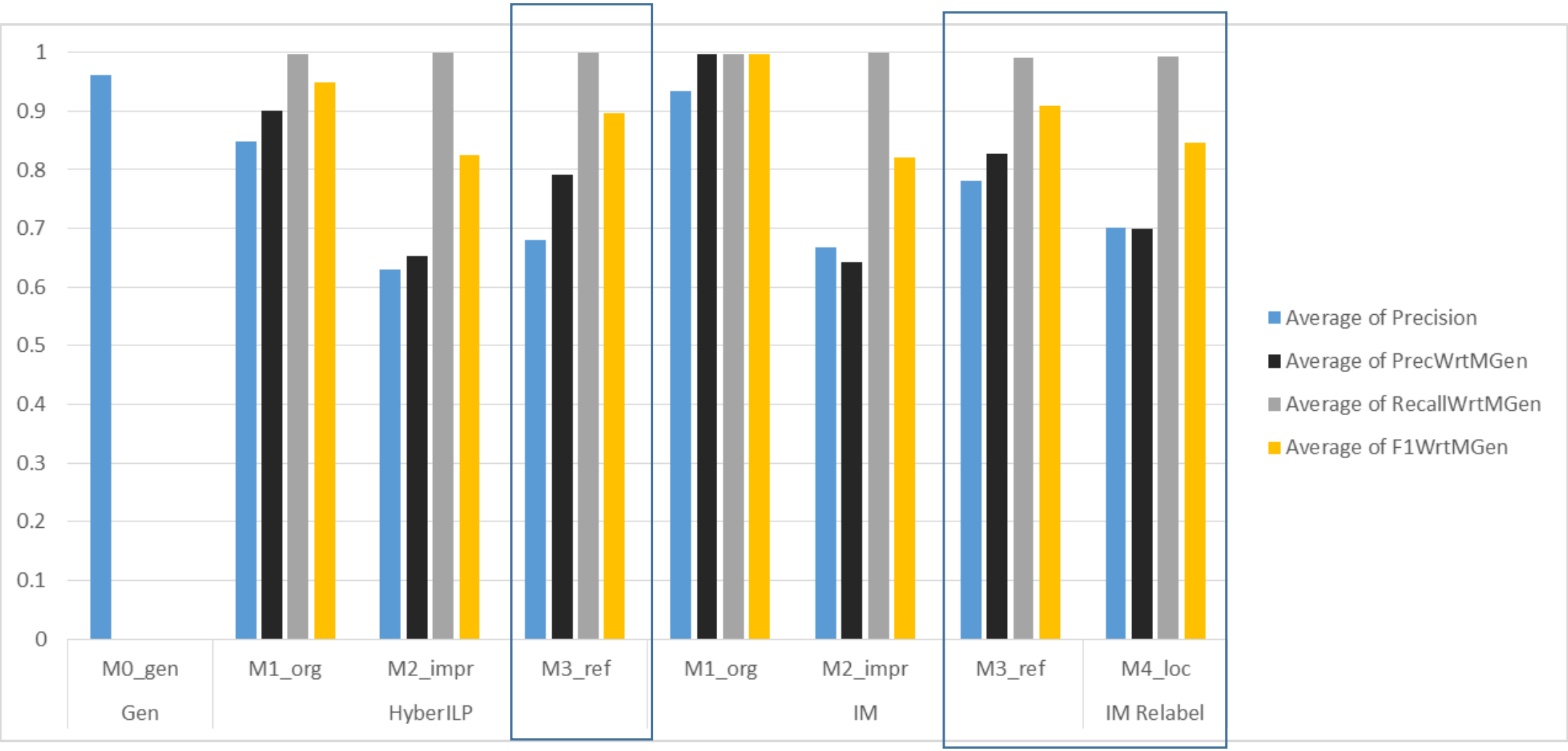


M_ref_IM



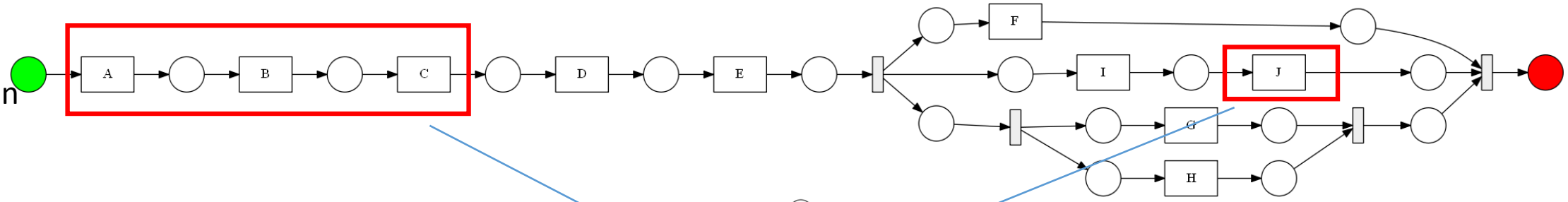
M_loc_IM

EXP3) N10(x5),P4(x4), (Oracle Detection)

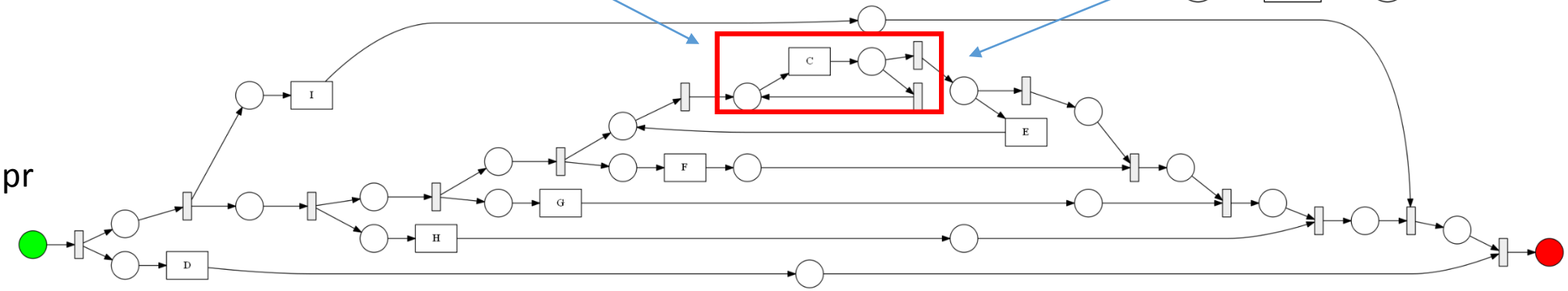


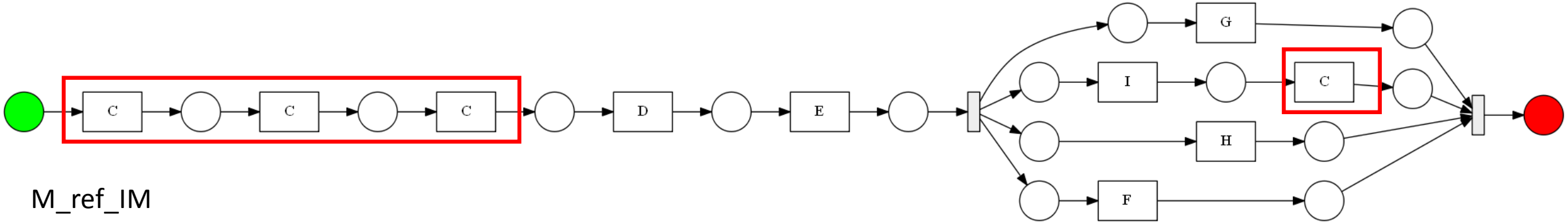
N10(x5),P4(x4), working

M_gen



M_impr

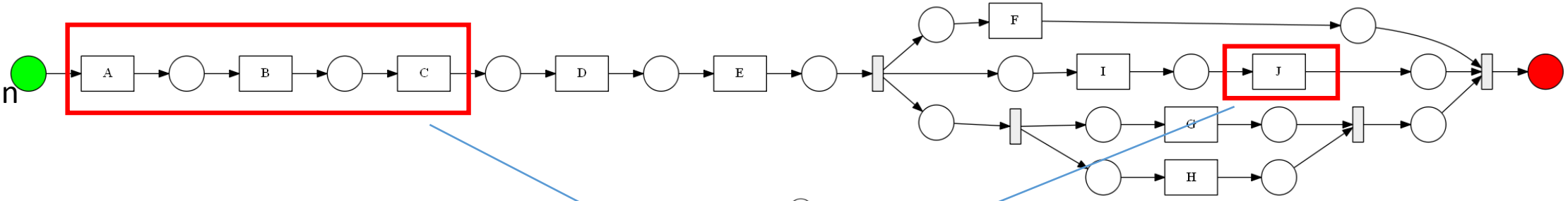




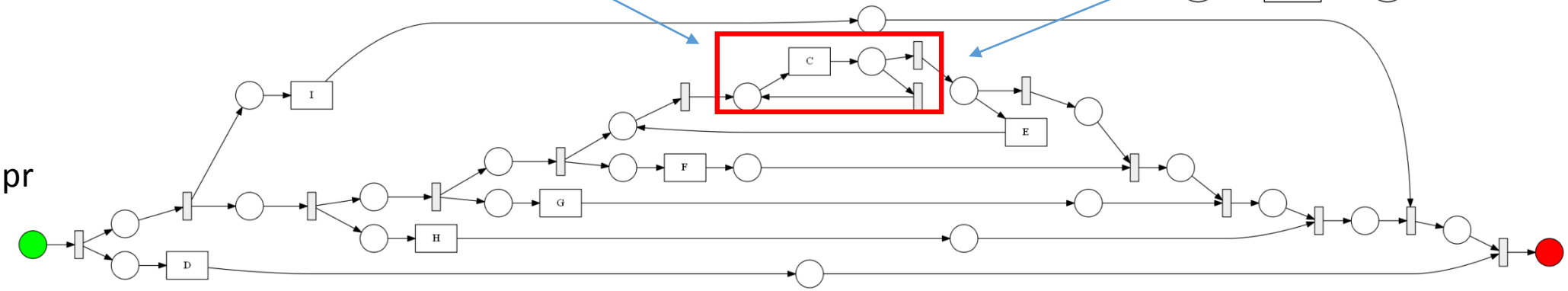
M_ref_IM

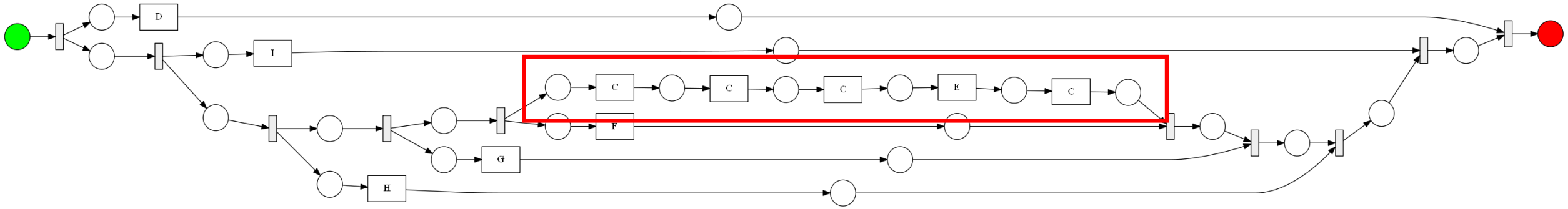
N10(x5),P4(x4), not working

M_gen



M_impr

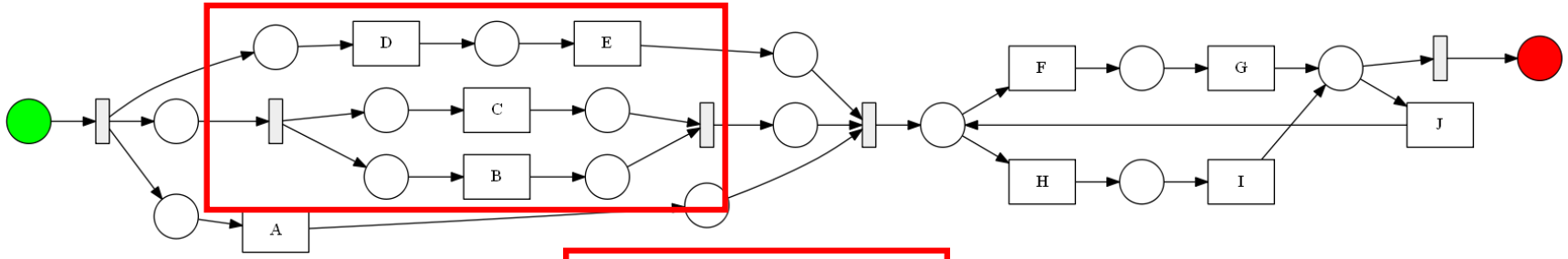




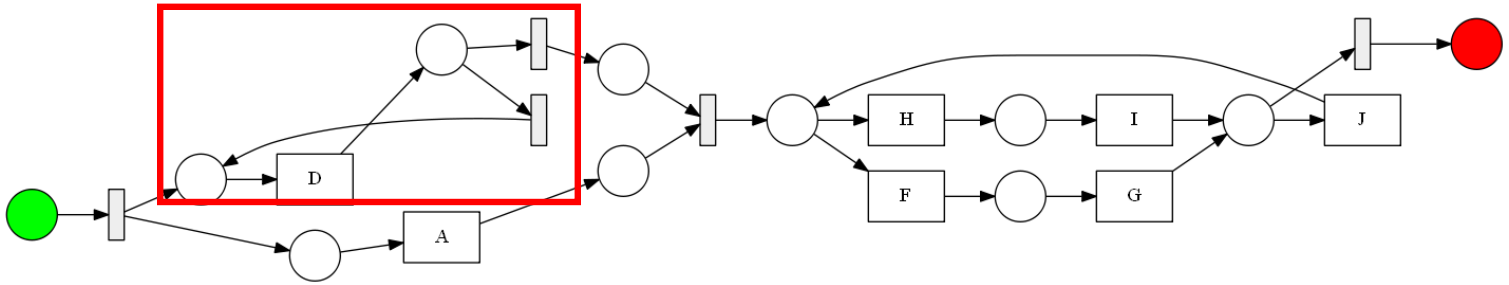
M_loc_IM

N10(x5),P4(x4), not working

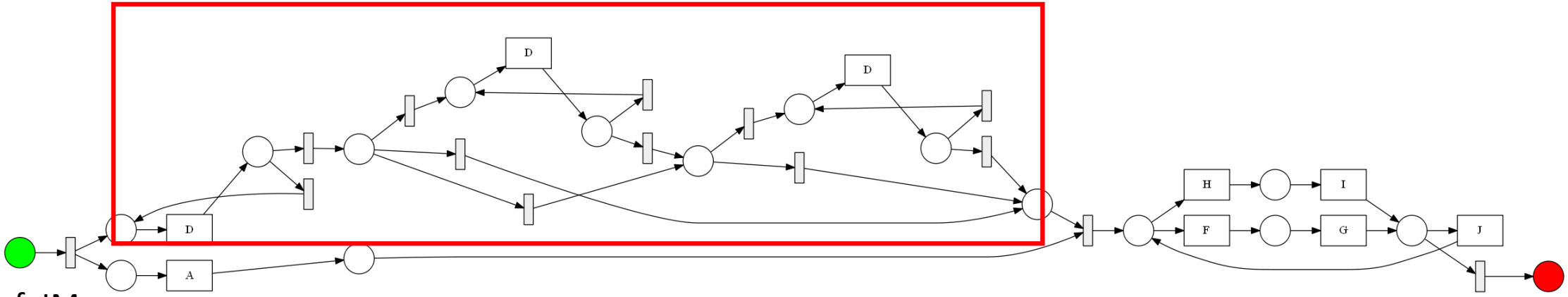
M_gen



M_impr

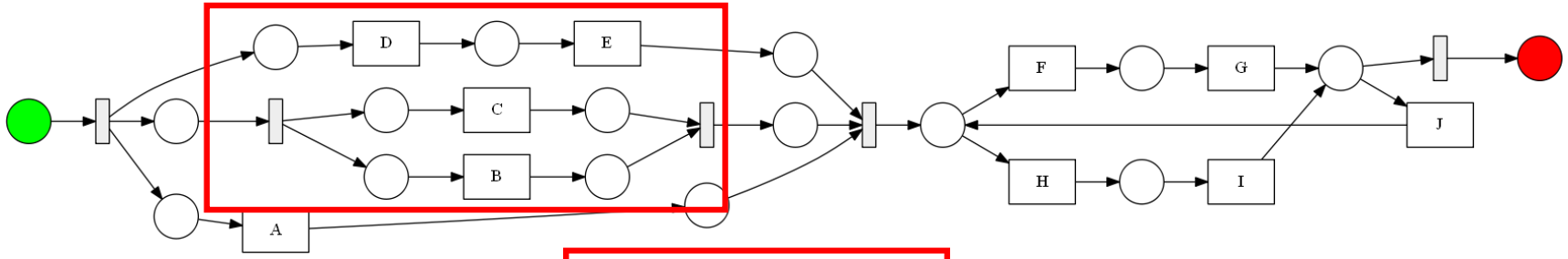


M_ref_IM

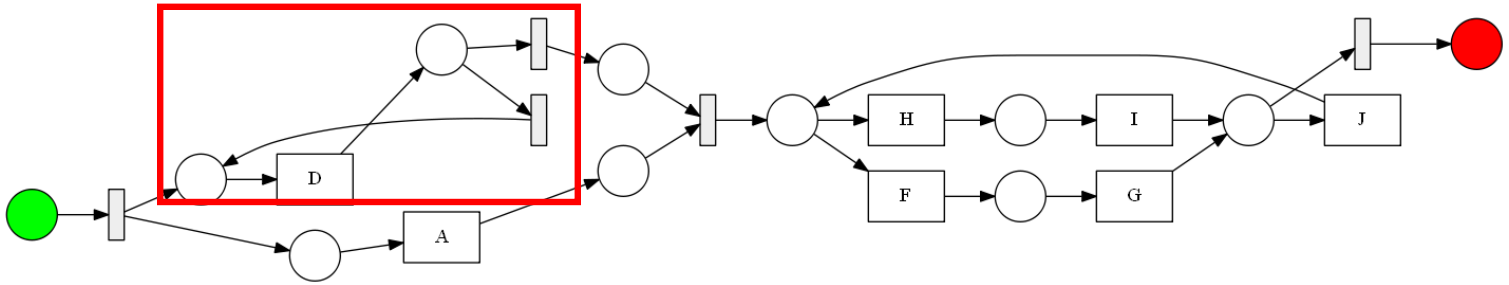


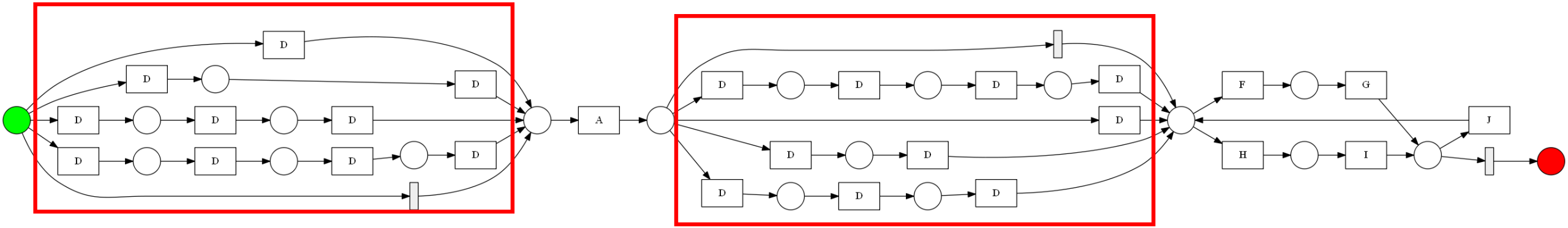
N10(x5),P4(x4), not working

M_gen



M_impr

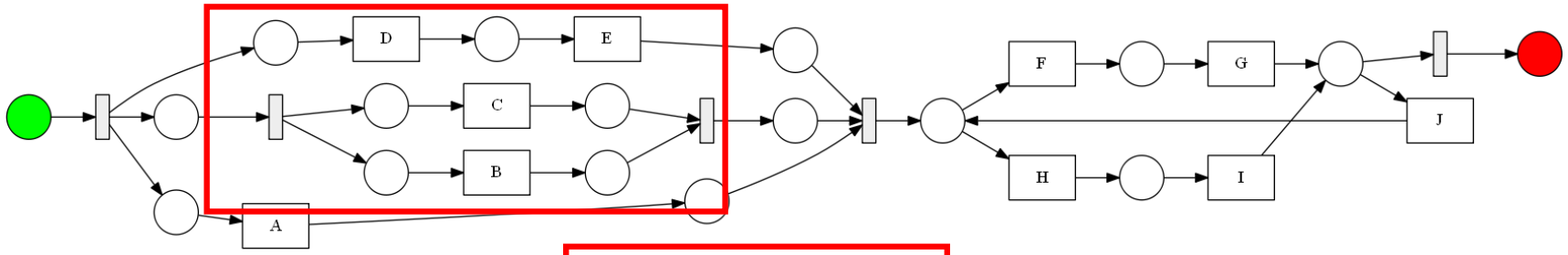




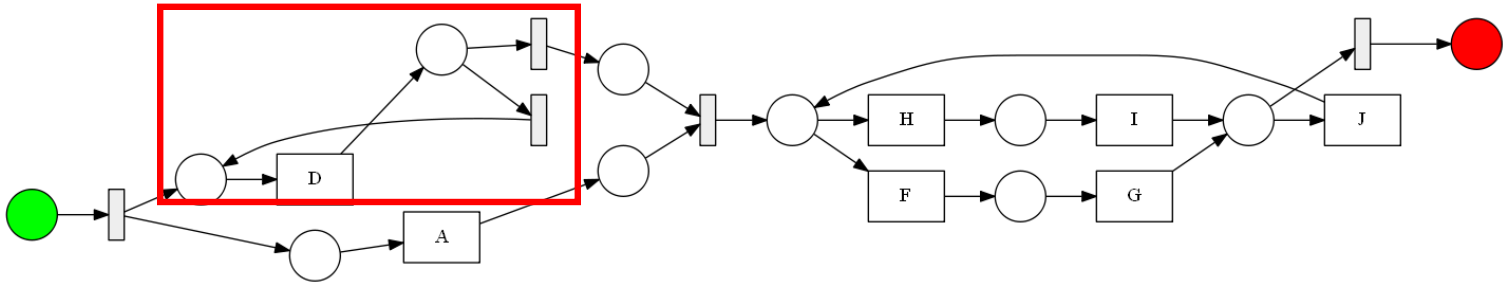
M_loc_IM

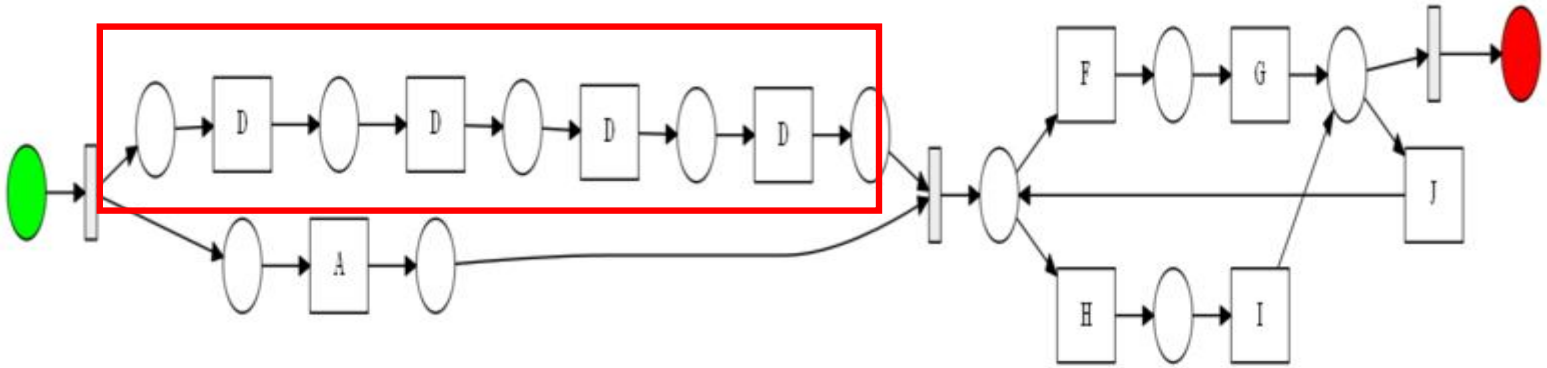
N10(x5),P4(x4), working, if change threshold

M_gen



M_impr

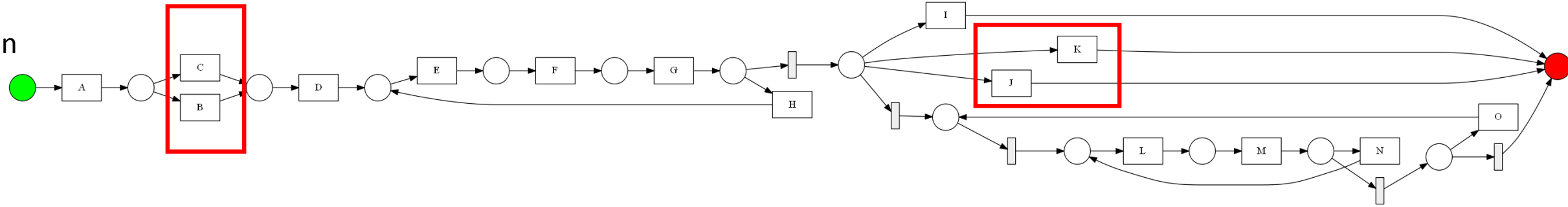




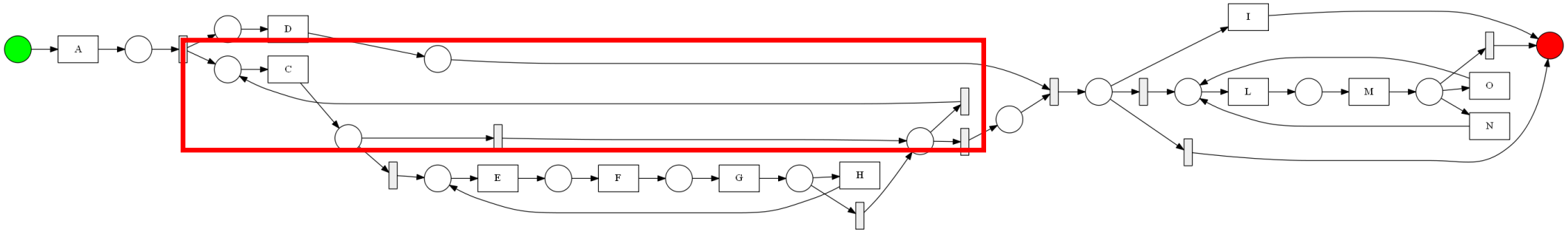
M_loc_IM

N15(x5),P4(x4), working

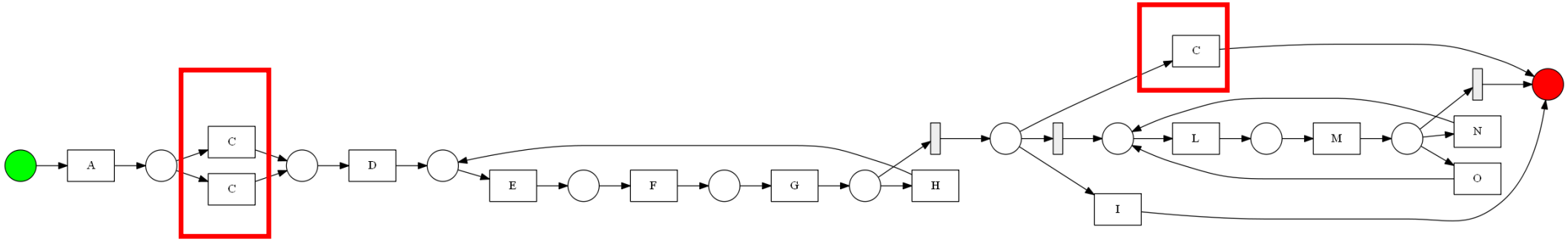
M_gen



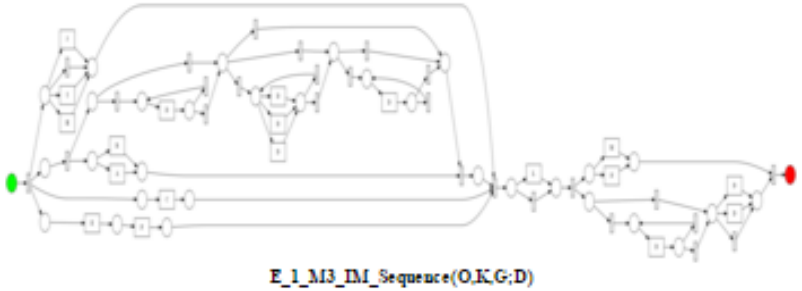
M_impr

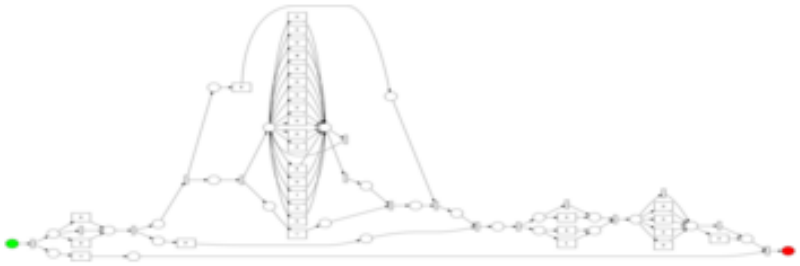


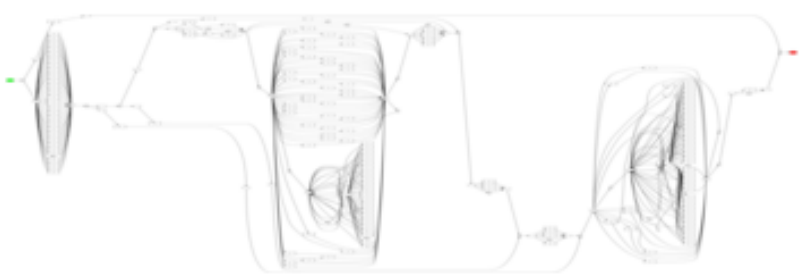
M_ref_IM



N15(x5),P4(x4), not working



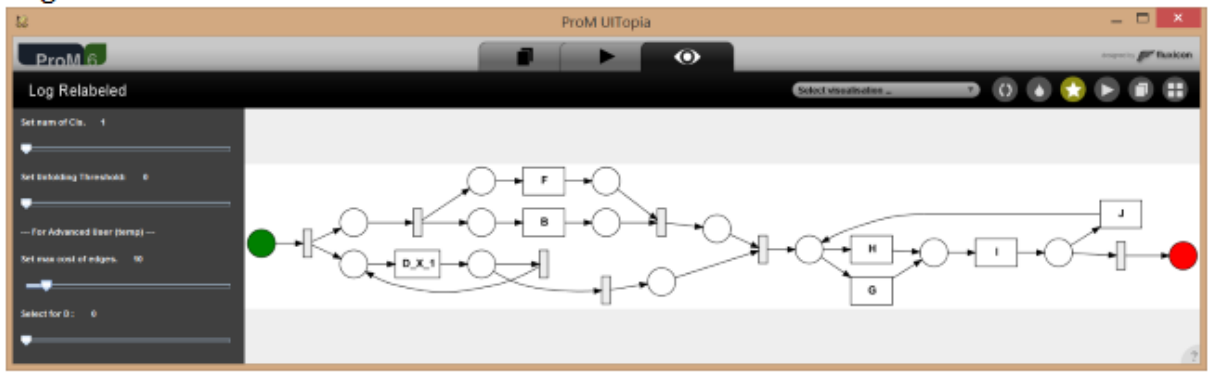




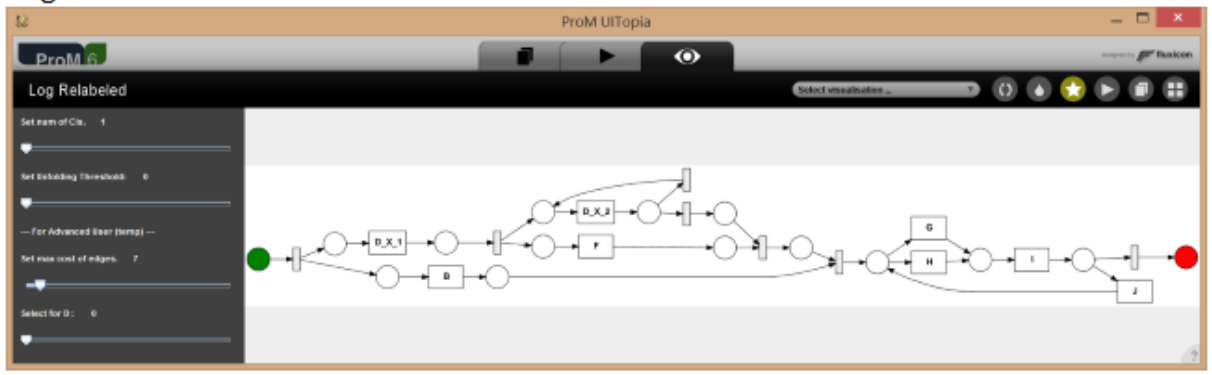
Three Main Causes For Fail to Refine

- Edge cost thresholds and unfolding threshold too low or too high

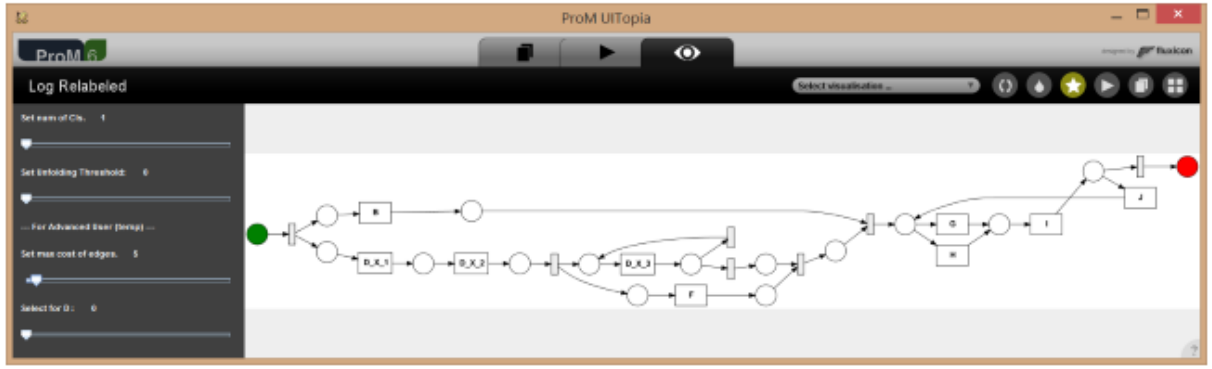
Edge cost = 10



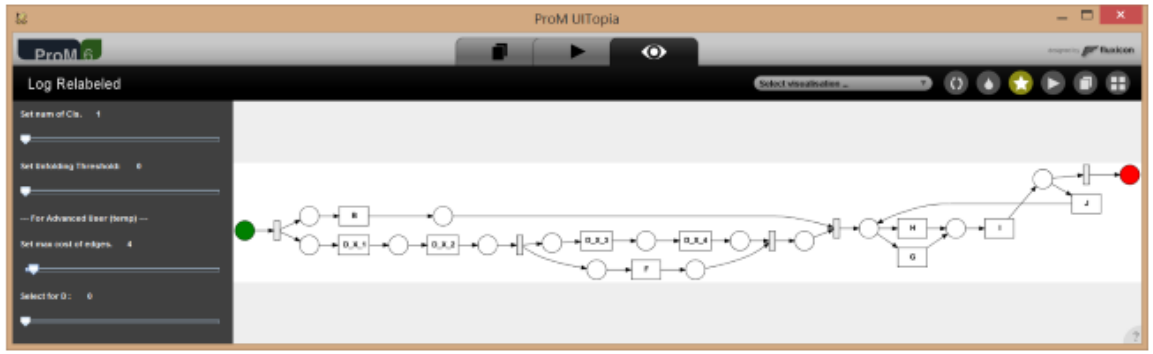
Edge cost = 7



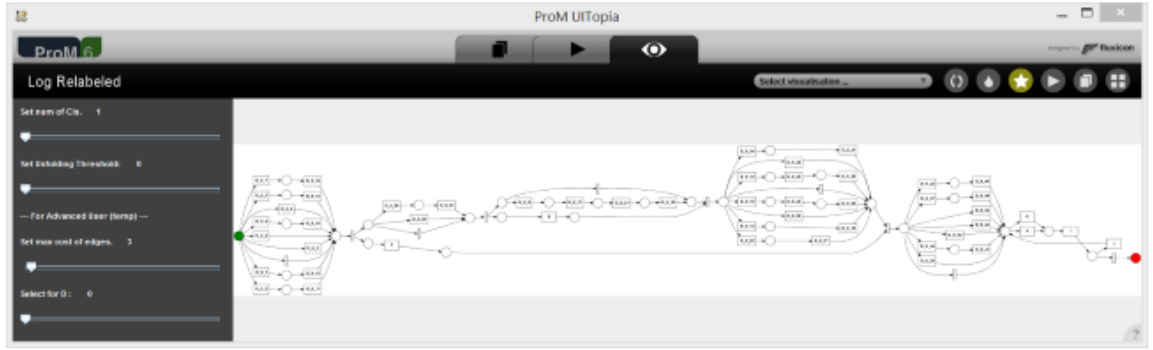
Edge cost = 6



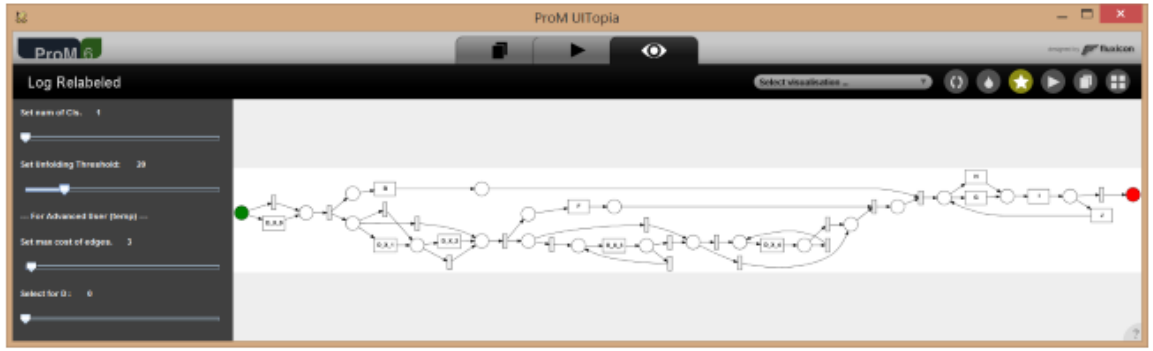
Edge cost = 4



Edge cost = 3;



With edge cost = 3, refold by set unfold threshold back to 20.. Flower loop.



Three Main Causes For Fail to Refine

- Edge cost thresholds and unfolding threshold too low or too high
- Loops too long
- Too many concurrent events

Future Work, Conclusion and Questions?

- More experiments…