University of Glasgow

System risk assessment with discrete-state components

Ji-Eun Byun

Lecturer in Smart, Sustainable, and Resilient Infrastructure

James Watt School of Engineering, University of Glasgow

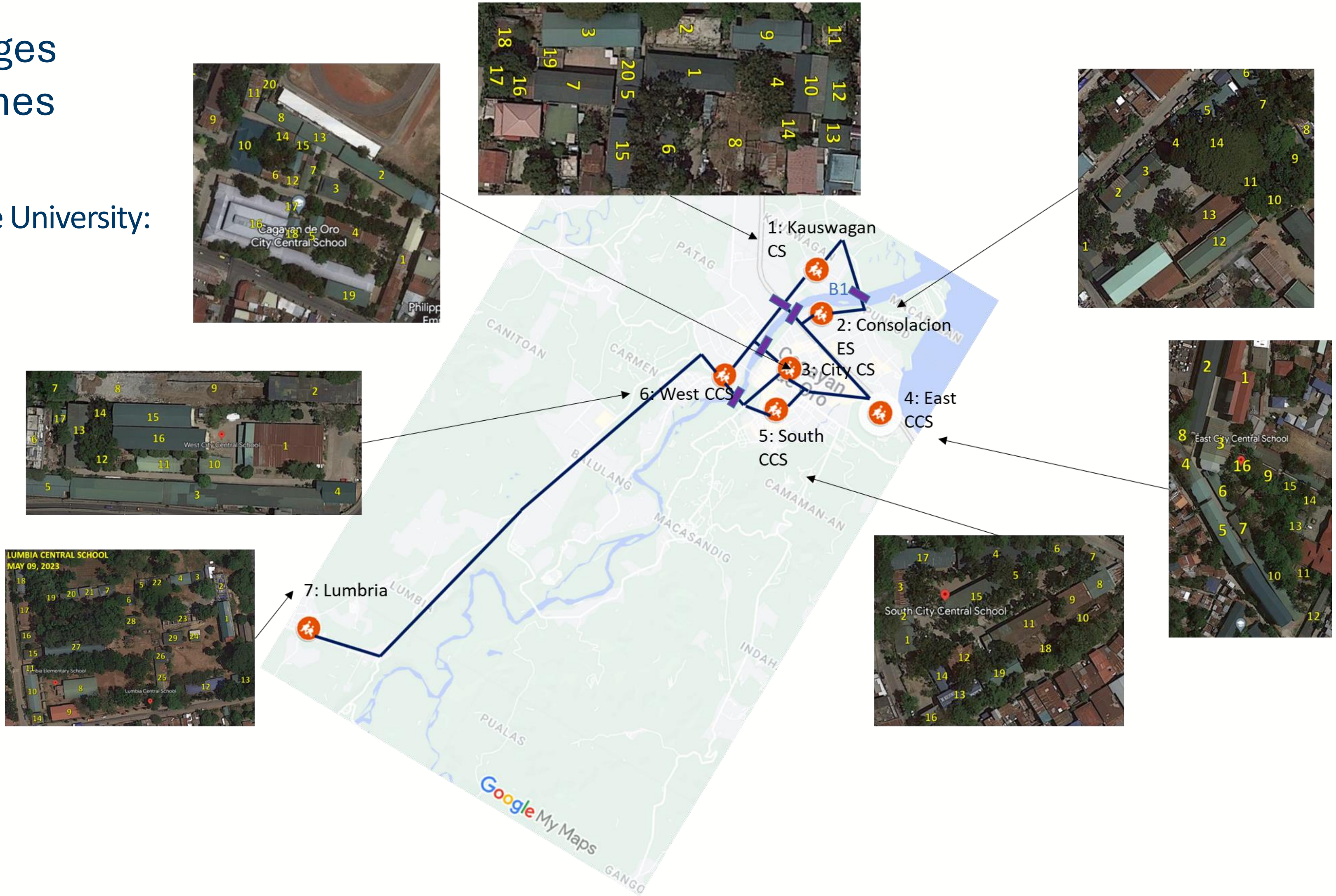
WORLD CHANGING GLASGOW

A WORLD TOP 100 UNIVERSITY

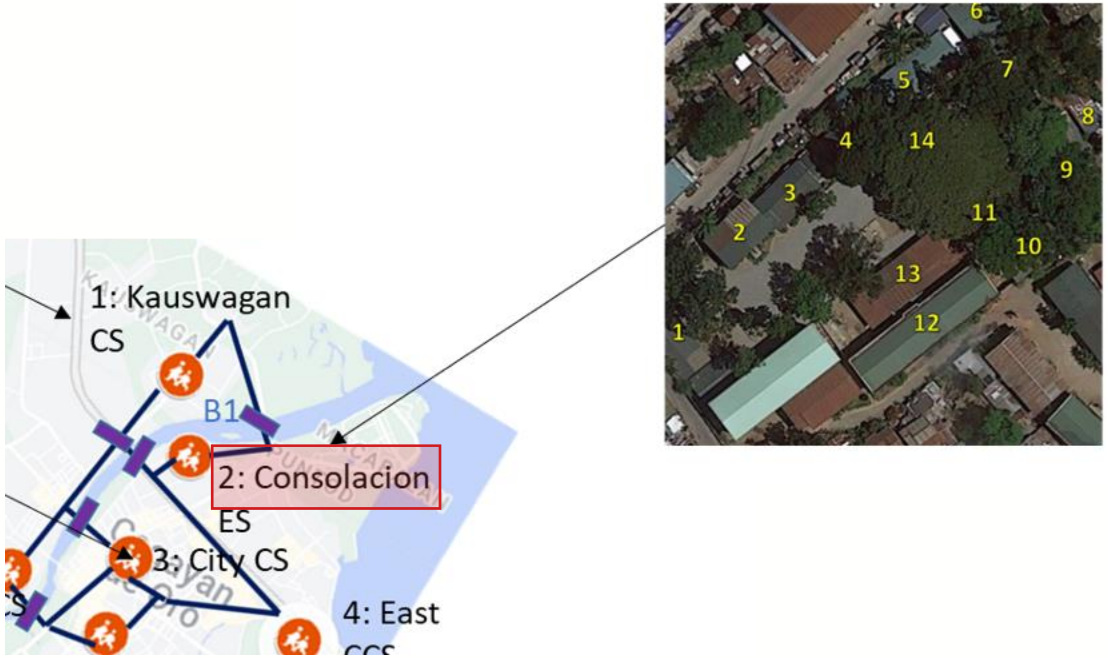
Inherent uncertainties in structural functionalities

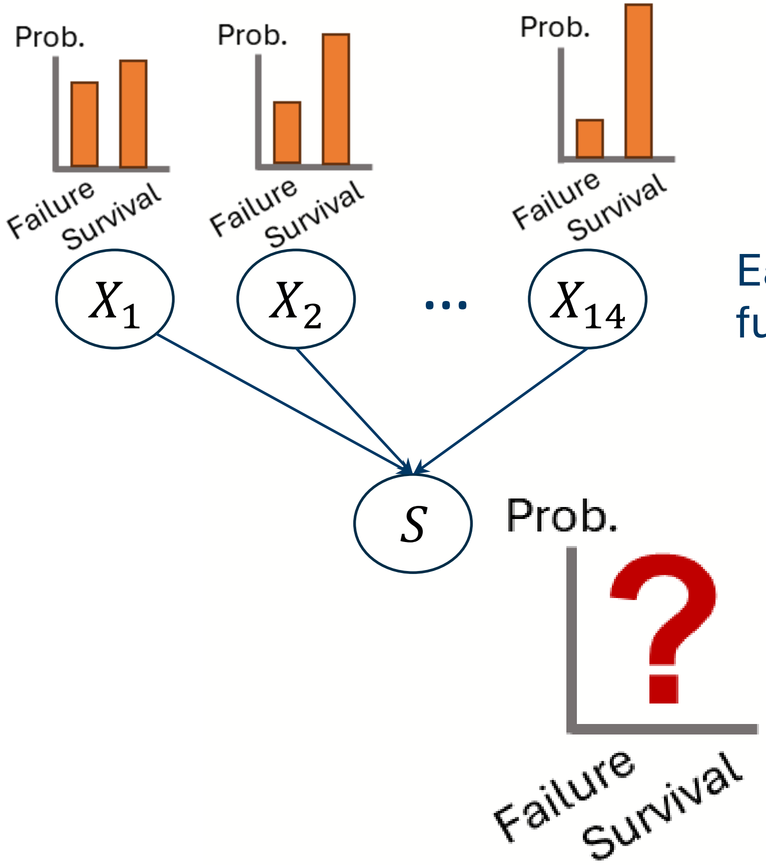
School complexes and bridges
in Cagayan de Oro, Philippines

Data obtained from past projects by
UCL, Xavier University, and De La Salle University:



Inherent uncertainties in structural functionalities





Each building's functionality

Ji-eun.byun@glasgow.ac.uk | Lecturer in Smart Sustainable and Resilient Infrastructure

System risk assessment with discrete-state components

Day 1

- Background and definition
- Sampling vs. decomposition
- Introduction to MBNPy

Day 2

- A tutorial on risk assessment of school complexes in Cagayan De Oro, Philippines

At the end of this tutorial, you will

- Understand the concept of system risk assessment and
- Analyse your system risk problems using MBNPy module.

Ji-eun.byun@glasgow.ac.uk | Lecturer in Smart Sustainable and Resilient Infrastructure

University of Glasgow

Challenge in calculating system probability

Exponential number of possible combinations

Component probabilities

$P(X_1)$

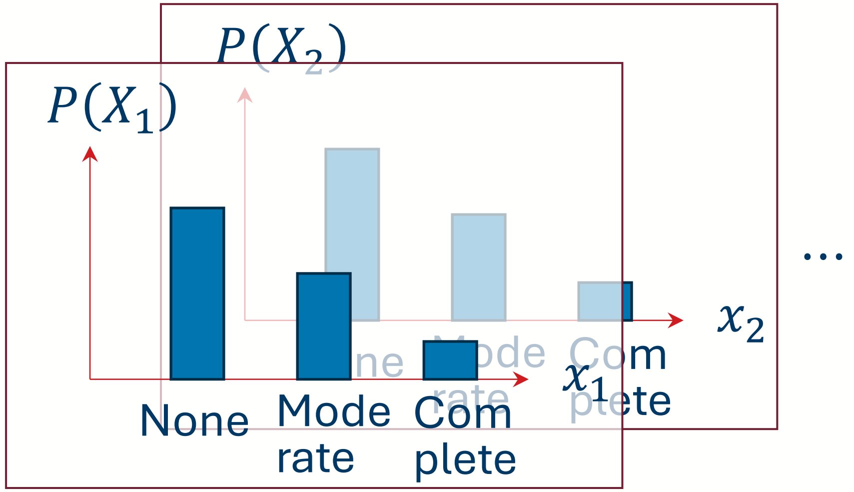
$\vdots$

$P(X_N)$

System function

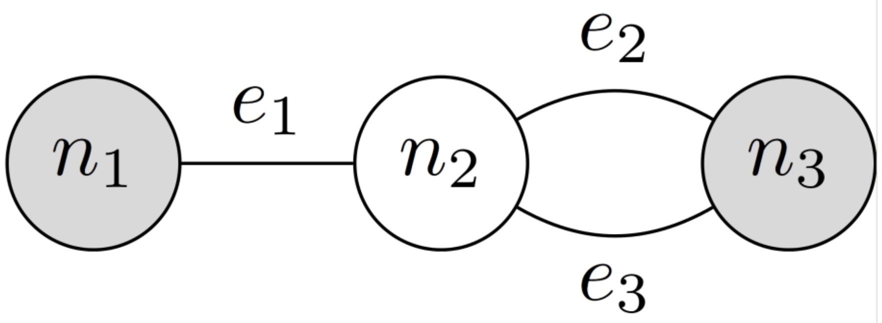
$s = f(x_1, \ldots, x_N)$

$$P(S = \text{fail}) = \sum_{\boxed{x_1,\ldots,x_N}} I[f(x_1,\ldots,x_N) = \text{fail}] \cdot P(x_1,\ldots,x_N)$$



Challenge in calculating system probability
Toy example

e_n's state is represented by random variable X_n.



failure survival

$P(X_1 = 0) = 0.1, P(X_1 = 1) = 0.9$
$P(X_2 = 0) = 0.2, P(X_2 = 1) = 0.8$
$P(X_3 = 0) = 0.3, P(X_3 = 1) = 0.7$

System survival: connection between n_1 and n_3

$(X_n = k) := x_n^k$

$(x_1^1, x_2^1, x_3^1) \to s^1, P(x_1^1, x_2^1, x_3^1) = 0.9 \cdot 0.8 \cdot 0.7 = 0.504$

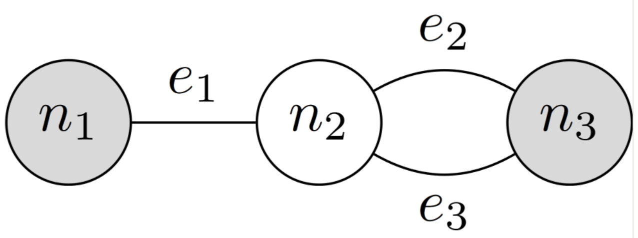
$(x_1^1, x_2^1, x_3^0) \to s^1, P(x_1^1, x_2^1, x_3^0) = 0.9 \cdot 0.8 \cdot 0.3 = 0.216$

$(x_1^1, x_2^0, x_3^1) \to s^1, P(x_1^1, x_2^0, x_3^1) = 0.9 \cdot 0.2 \cdot 0.7 = 0.126$

$(x_1^0, x_2^1, x_3^1) \to s^0, P(x_1^0, x_2^1, x_3^1) = 0.1 \cdot 0.8 \cdot 0.7 = 0.056$

Challenge in calculating system probability

Toy example (cont'd)



System survival: connection between n_1 and n_3

$(x_1^1, x_2^1, x_3^1) \rightarrow s^1, P(x_1^1, x_2^1, x_3^1) = 0.9 \cdot 0.8 \cdot 0.7 = 0.504$

$(x_1^1, x_2^1, x_3^0) \rightarrow s^1, P(x_1^1, x_2^1, x_3^0) = 0.9 \cdot 0.8 \cdot 0.3 = 0.216$

$(x_1^1, x_2^0, x_3^1) \rightarrow s^1, P(x_1^1, x_2^0, x_3^1) = 0.9 \cdot 0.2 \cdot 0.7 = 0.126$

$(x_1^0, x_2^1, x_3^1) \rightarrow s^0, P(x_1^0, x_2^1, x_3^1) = 0.1 \cdot 0.8 \cdot 0.7 = 0.056$

$(x_1^1, x_2^0, x_3^0) \rightarrow s^0, P(x_1^1, x_2^0, x_3^0) = 0.9 \cdot 0.2 \cdot 0.3 = 0.054$

$(x_1^0, x_2^1, x_3^0) \rightarrow s^0, P(x_1^0, x_2^1, x_3^0) = 0.1 \cdot 0.8 \cdot 0.3 = 0.024$

$(x_1^0, x_2^0, x_3^1) \rightarrow s^0, P(x_1^0, x_2^0, x_3^1) = 0.1 \cdot 0.2 \cdot 0.7 = 0.014$

$(x_1^0, x_2^0, x_3^0) \rightarrow s^0, P(x_1^0, x_2^0, x_3^0) = 0.1 \cdot 0.2 \cdot 0.3 = 0.006$

$+$

$0.056 + 0.054 + 0.024 + 0.014 + 0.006 = \mathbf{0.154}$

$$P(S = \text{fail}) = \sum_{x_1,\dots,x_N} I[f(x_1,\dots,x_N) = \text{fail}] \cdot P(x_1,\dots,x_N)$$

University of Glasgow

Challenge in calculating system probability
Toy example (cont'd)

$$\left.\begin{array}{l}
(x_1^1, x_2^1, x_3^1) \rightarrow s^1, P(x_1^1, x_2^1, x_3^1) = 0.9 \cdot 0.8 \cdot 0.7 = 0.504 \\
(x_1^1, x_2^1, x_3^0) \rightarrow s^1, P(x_1^1, x_2^1, x_3^0) = 0.9 \cdot 0.8 \cdot 0.3 = 0.216 \\
(x_1^1, x_2^0, x_3^1) \rightarrow s^1, P(x_1^1, x_2^0, x_3^1) = 0.9 \cdot 0.2 \cdot 0.7 = 0.126 \\
(x_1^0, x_2^1, x_3^1) \rightarrow s^0, P(x_1^0, x_2^1, x_3^1) = 0.1 \cdot 0.8 \cdot 0.7 = 0.056 \\
(x_1^1, x_2^0, x_3^0) \rightarrow s^0, P(x_1^1, x_2^0, x_3^0) = 0.9 \cdot 0.2 \cdot 0.3 = 0.054 \\
(x_1^0, x_2^1, x_3^0) \rightarrow s^0, P(x_1^0, x_2^1, x_3^0) = 0.1 \cdot 0.8 \cdot 0.3 = 0.024 \\
(x_1^0, x_2^0, x_3^1) \rightarrow s^0, P(x_1^0, x_2^0, x_3^1) = 0.1 \cdot 0.2 \cdot 0.7 = 0.014 \\
(x_1^0, x_2^0, x_3^0) \rightarrow s^0, P(x_1^0, x_2^0, x_3^0) = 0.1 \cdot 0.2 \cdot 0.3 = 0.006
\end{array}\right\} (\text{No. of states})^N$$

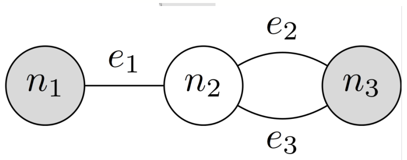
e.g. 100 binary-state components: $2^{100} > 10^{30}$
Simply impossible number!

Any solution?

Common solution: Monte Carlo Simulation

- Brute-force generation of samples by given probabilities
- E.g.

```
prob = [0.1, 0.2, 0.3];
samples = zeros(10, 3);
for i=1:3
    samples(:,i) = randsample([0,1], 10, true, [prob(i), 1-prob(i)]);
end
```



Try 1: $\hat{P}(s^0) = 0.4$

samples =

1	1	0
0	1	1
1	1	1
0	1	0
0	0	0
1	1	1
1	1	1
1	1	1
1	1	1
0	1	0

Try 2 : $\hat{P}(s^0) = 0$

samples =

1	1	1
1	1	1
1	1	1
1	1	1
1	1	1
1	1	0
1	0	1
1	1	1
1	1	1
1	1	1

Try 3 : $\hat{P}(s^0) = 0.4$

samples =

0	0	1
0	1	0
1	1	0
1	1	1
1	0	0
1	1	1
1	1	1
1	1	1
0	1	0
1	1	1

Recall $P(s^0) = 0.154.$

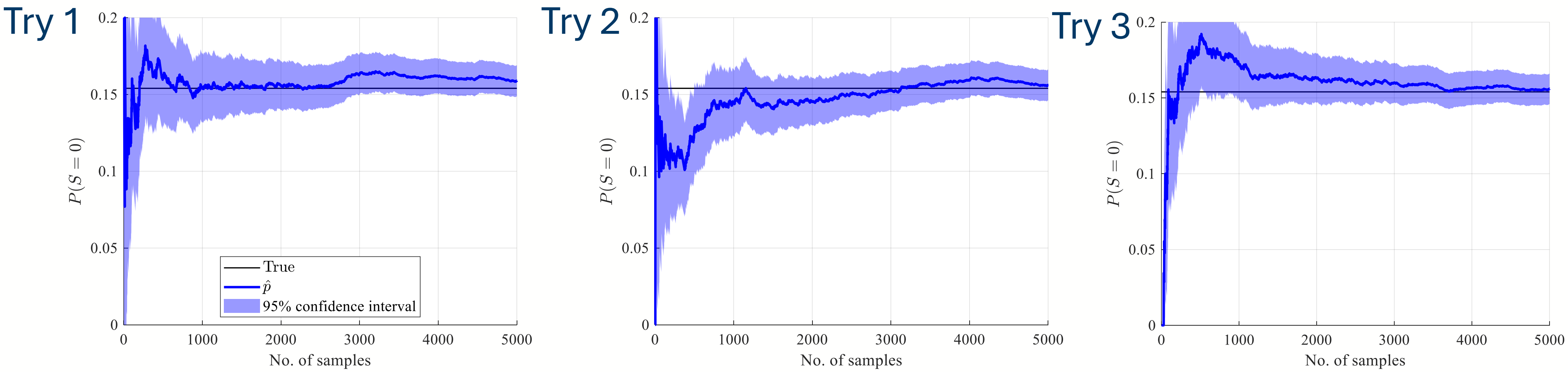
What's wrong?

MCS needs enough samples.

- Measure of confidence: coefficient of variance

$$\hat{\delta} = \frac{\sqrt{\widehat{\mathrm{Var}}[\hat{p}]}}{\hat{p}} = \sqrt{\frac{(1-\hat{p})}{N\hat{p}}} \approx \sqrt{\frac{1}{N\hat{p}}}$$

- Required number of samples for $\hat{\delta}$, $N = 1/\hat{p}\hat{\delta}^2$
  - E.g. we want $\hat{\delta} = 5\%$, we need $N = 1/(0.154 \cdot 0.05^2) = 2{,}597$ samples.



Complexity of MCS

Required number of samples for $\hat{\delta}$, $N = 1/\hat{p}\hat{\delta}^2$

- N increases with a lower $\hat{p}$.
- $N\hat{p} = 1/\hat{\delta}^2 = (\text{No. of failure samples})$
    - For $\hat{\delta} = 0.05$, MCS should see $1/(0.05)^2 = 400$ failure samples.
- E.g. $\hat{p} = 0.001$ and $\hat{\delta} = 0.05$, $N = 400{,}000$.
    - If it takes 1 second for a system simulation, it takes 5 days.

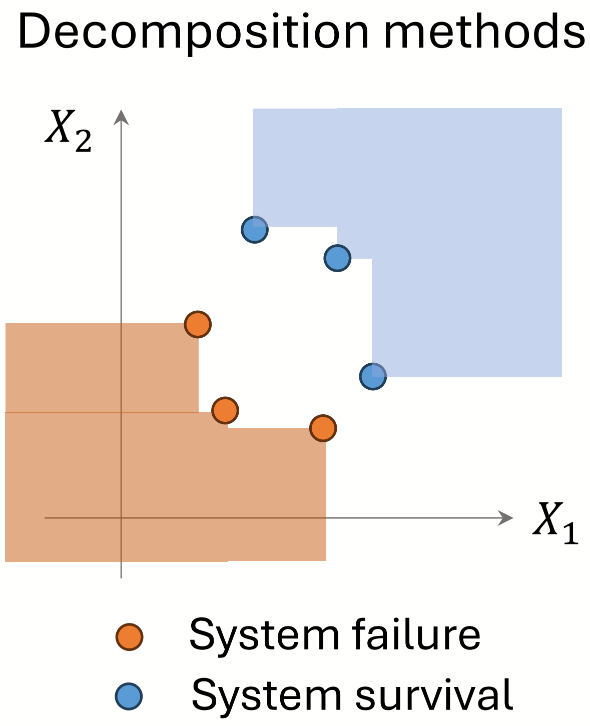
Whenever component probabilities $P(X_n), n = 1, \dots, N$, changes, MCS needs to be done all over again.

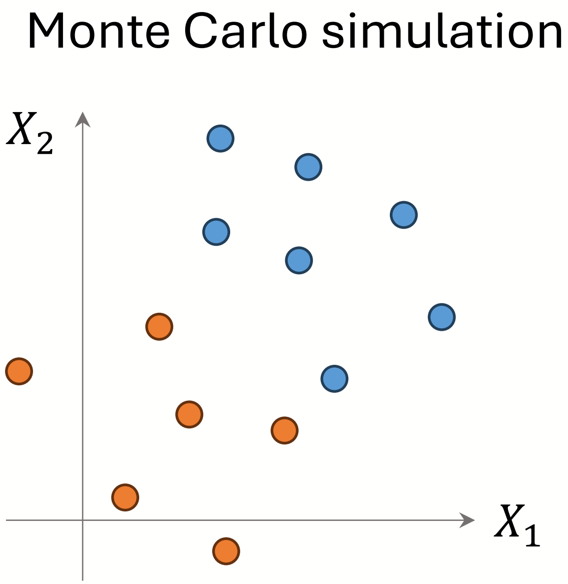
- I.e. previous results cannot be re-used.

Decomposition methods as an alternative solution

Use understandings of a system event for computational efficiency

- Coherency: Better component states do not worsen system state.



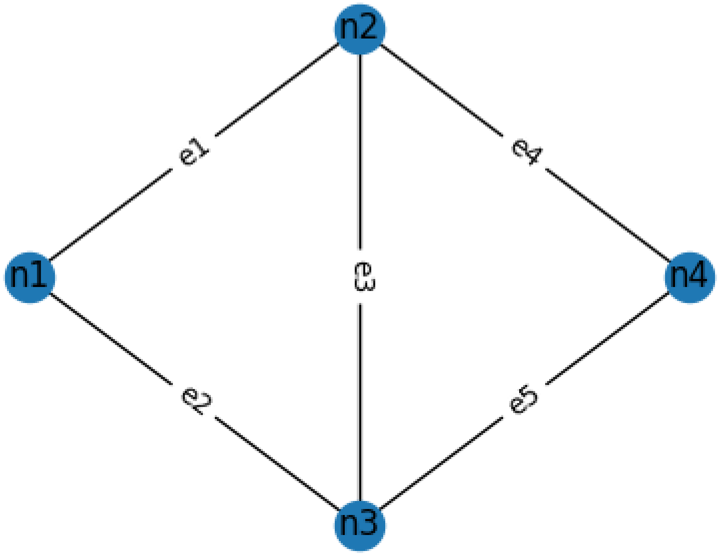


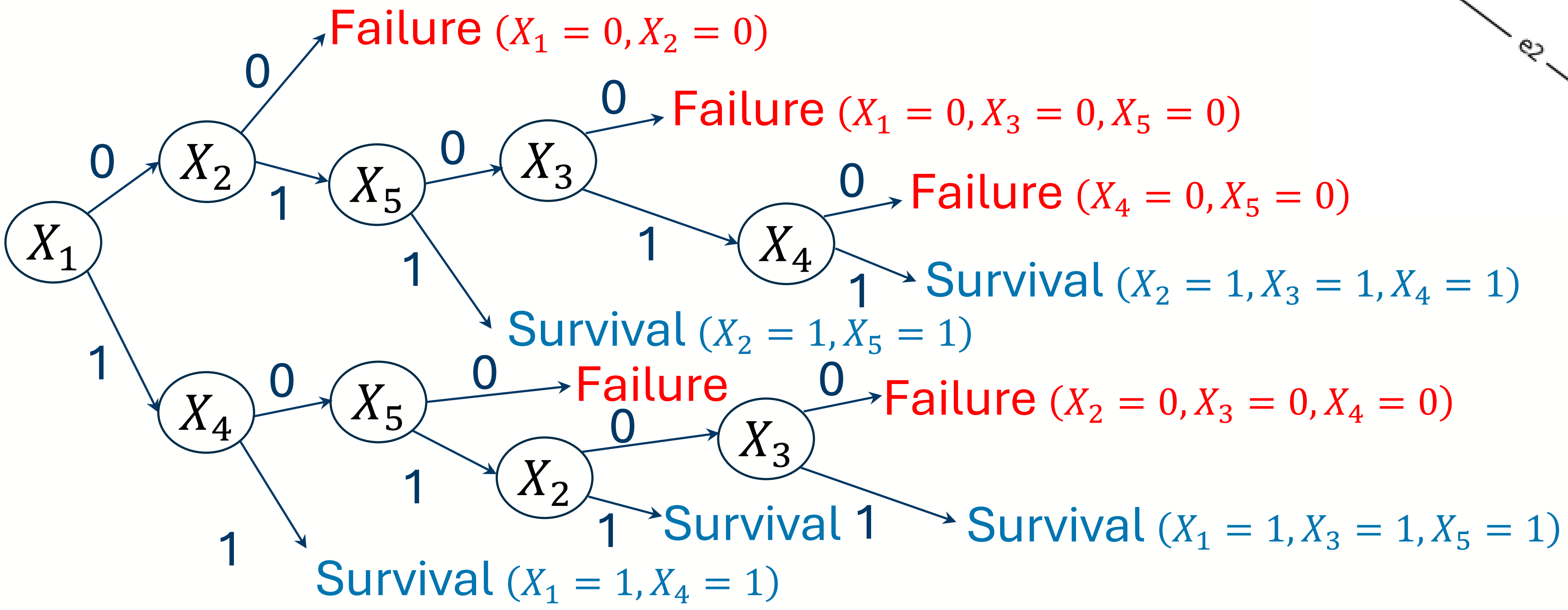
Decomposition methods as an alternative solution

Example: decomposition

Problem

- X_n's state (representing e_n): 0 (failure) or 1 (survival)
- System failure: dysconnectivity between n1 to n4

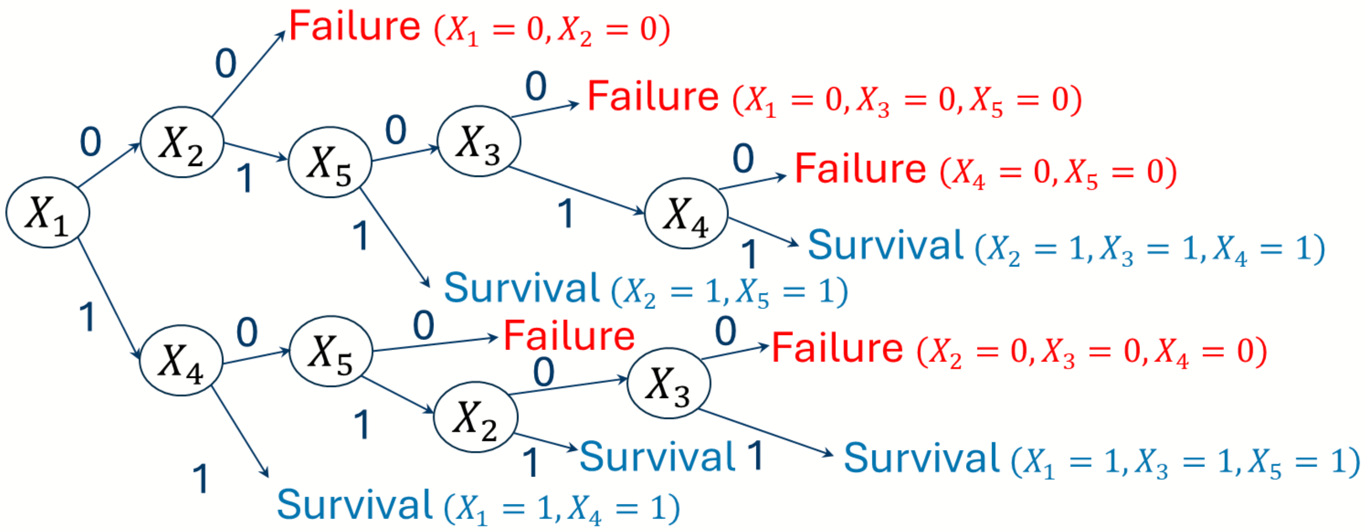




- $2^5 = 32$
  $\rightarrow$10 branches
- 4 survival rules and
  4 failure rules.

Decomposition methods as an alternative solution
Example: computation

- Failure probability = $\sum_{\text{fail branch}} P(\text{fail branch})$



$P(x_n^0) = 0.01, P(x_n^1) = 0.99, n = 1,2$
$P(x_n^0) = 0.05, P(x_n^1) = 0.95, n = 3,4$
$P(x_n^0) = 0.10, P(x_n^1) = 0.90, n = 5$

$\rightarrow$ Failure prob.
$= 0.01 \cdot 0.01 + 0.01 \cdot 0.99 \cdot 0.10 \cdot 0.05 + \cdots$
$= 0.0001 + 0.0000495 + \cdots$

- Probability update: No need for another B&B.

Updated probabilities to
$P(x_n^0) = 0.2, P(x_n^1) = 0.8$ for $n = 1$

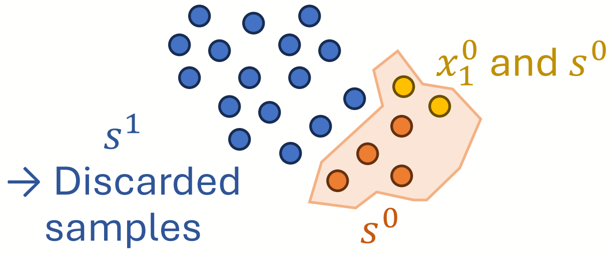
$\rightarrow$ Failure prob. $= 0.2 \cdot 0.01 + 0.2 \cdot 0.99 \cdot 0.10 \cdot 0.05 + \cdots$

University of Glasgow

Comparison

MCS

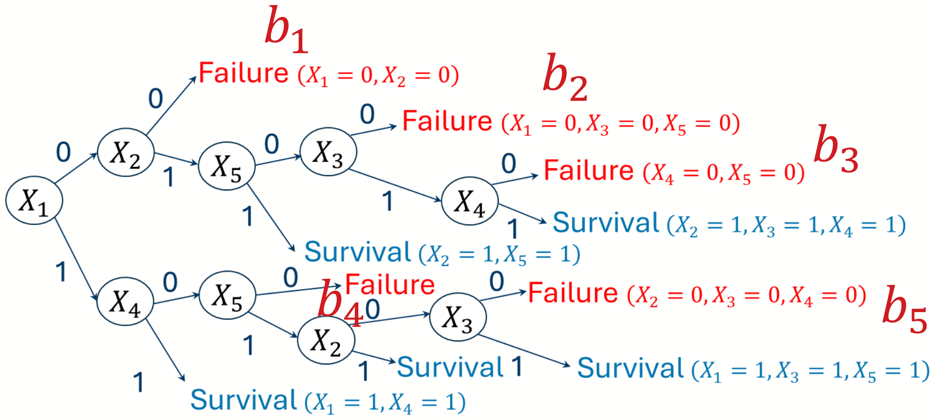
- **Complexity** $O(P(s^0)^{-1})$
- **Probability update** requires a new round.
- **Conditional probability** complexity
$$O(P(s^0)^{-2})$$
  - E.g. component importance measure $P(x_n^0|s^0)$



Effective number of samples
$N \to N \cdot P(s^0)$

Decomposition

- O(# branches), where # branches ↑↑ with # rules ↑ with # components.
- Update adds little computation cost.
- Conditional doesn't increase complexity.

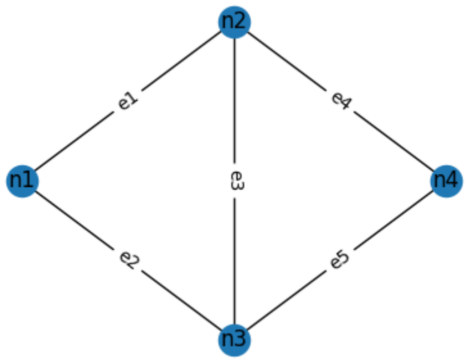


$P(x_n^0|s^0) = \dfrac{P(x_n^0, s^0)}{P(s^0)}$,
where
$P(x_n^0, s^0) = P(b_1) + P(b_2) + P(b_3)$

MBNPy package
Decomposition using BRC

BRC algorithm

- Input 1: System event




```
nodes = {'n1': (0, 0),
         'n2': (1, 1),
         'n3': (1, -1),
         'n4': (2, 0)}

edges = {'e1': ['n1', 'n2'],
         'e2': ['n1', 'n3'],
         'e3': ['n2', 'n3'],
         'e4': ['n2', 'n4'],
         'e5': ['n3', 'n4']}
```

```
def net_conn(comps_st, od_pair, edges, varis): # maximum flow analysis

    G = nx.Graph()
    for k,x in comps_st.items():
        G.add_edge(edges[k][0], edges[k][1]) # we add each edge
        G[edges[k][0]][edges[k][1]]['capacity'] = varis[k].values[x] # define capacity as 0 if state = 0 or 1 if state = 1

    # perform maximum flow analysis between the OD pair
    G.add_edge(od_pair[1], 'new_d', capacity=1) # add a new edge with capacity 1 to ensure we find only ONE path.
    f_val, f_dict = nx.maximum_flow(G, od_pair[0], 'new_d', capacity='capacity', flow_func=shortest_augmenting_path)

    if f_val > 0: # if the flow between the OD pair is greater than 0, the two nodes are connected
        sys_st = 's'

        # We can infer an associated minimum survival rule in case of network connectivity.
        min_comps_st = {}
        for k, x in comps_st.items():
            k_flow = max([f_dict[edges[k][0]][edges[k][1]], f_dict[edges[k][1]][edges[k][0]]])
            if k_flow > 0: # the edges with flows > 0 consitute a minimum survival rule.
                min_comps_st[k] = 1

    else:
        sys_st = 'f'

        # In case of system failur        nimum                          rd.
        min_comps_st = None

    return f_val, sys_st, min_comps_st
```

Either
‘s’ or ‘f’

E.g. {‘x1’: 1, ‘x4’: 1}
If unknown, return
None

MBNPy package
Decomposition using BRC

BRC algorithm

- Input 2: Probability

```
probs = {'e1': {0: 0.01, 1: 0.99}, 'e2': {0: 0.01, 1: 0.99}, 'e3': {0: 0.05, 1: 0.95},
         'e4': {0: 0.05, 1: 0.95}, 'e5': {0: 0.10, 1: 0.90}}
```

MBNPy package
Decomposition using BRC

BRC algorithm

- Run

```
sys_fun = lambda comps_st : net_conn(comps_st, od_pair, edges, varis)
```

```
brs, rules, sys_res, monitor = brc.run(probs, sys_fun)
```

✓ 0.0s

```
*Final decomposition is completed with 11 branches (originally 13 branches).

***Analysis completed with f_sys runs 8: out_flag = complete***
The # of found non-dominated rules (f, s): 8 (4, 4)
Probability of branchs (f, s, u): (5.1688e-03, 9.95e-01, 0.0000e+00)
The # of branches (f, s, u), (min, avg) len of rf: 11 (5, 6, 0), (2, 2.50)
Elapsed seconds (average per round): 1.15e-02 (1.28e-03)
```

MBNPy package
Decomposition using BRC

BRC algorithm

- Outcome 1: Identified rules

```
print(rules['s'])
print(rules['f'])
```

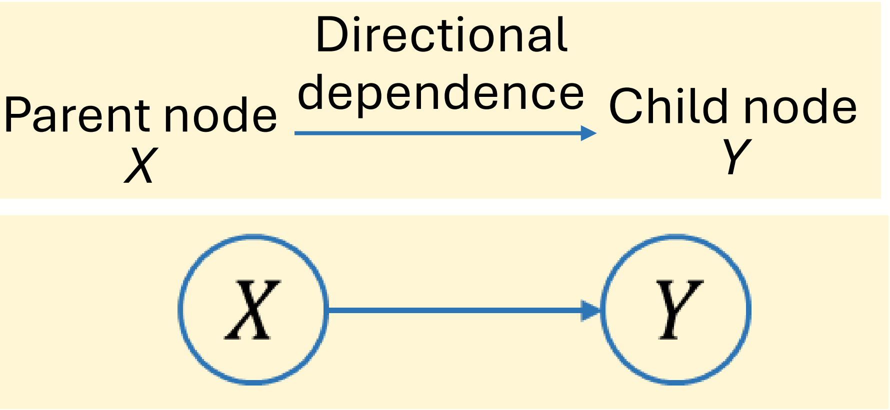
✓ 0.0s

```
[{'e1': 1, 'e4': 1}, {'e2': 1, 'e5': 1}, {'e2': 1, 'e3': 1, 'e4': 1}, {'e1': 1, 'e3': 1, 'e5': 1}]
[{'e4': 0, 'e5': 0}, {'e1': 0, 'e2': 0}, {'e1': 0, 'e3': 0, 'e5': 0}, {'e2': 0, 'e3': 0, 'e4': 0}]
```

MBNPy package
Inference by Matrix-based Bayesian network

Bayesian network as probabilistic graphical models

- Random variables – circular nodes
- Directional dependence – directed edges
    - ∋ Causal dependence
- Directed acyclic graph
- Each node is quantified by a distribution conditional on parent nodes
- Joint distribution = product of all distributions

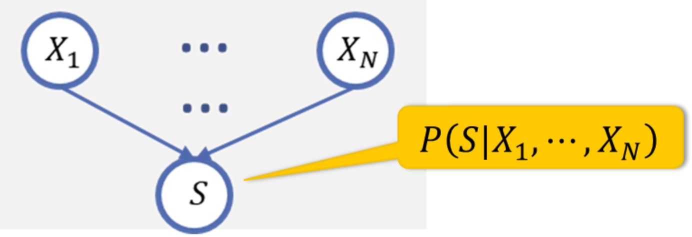


$$P(Y|X)$$

$$P(X,Y) = P(X) \cdot P(Y|X)$$

MBNPy package
Inference by Matrix-based Bayesian network

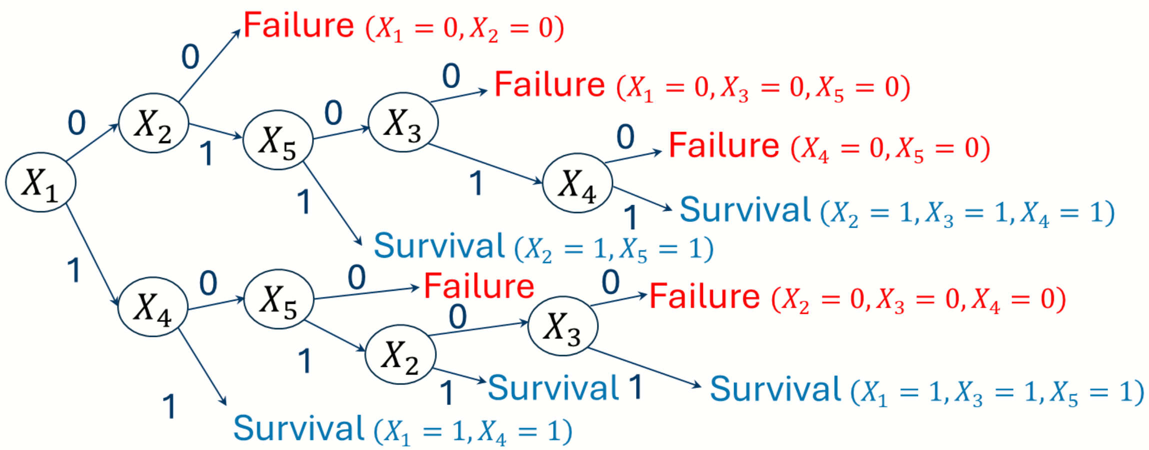
MBN: Alternative BN data structure for system events



$$\left.\begin{aligned}
&P(S=0\,|\,X_1=0, X_2=0, \cdots, X_N=0)\\
&P(S=1\,|\,X_1=0, X_2=0, \cdots, X_N=0)\\
&P(S=0\,|\,X_1=1, X_2=0, \cdots, X_N=0)\\
&\qquad\vdots\\
&P(S=1\,|\,X_1=1, X_2=1, \cdots, X_N=1)
\end{aligned}\right\} 2^{N+1}$$

$x_n^2 := x_n^0$ or x_n^1

- Idea: composite state representing multiple states



$$\begin{array}{cccccc}
S\,| & X_1 & X_2 & X_3 & X_4 & X_5
\end{array}$$

$$\mathbf{C} = \begin{bmatrix}
0 & 0 & 0 & 2 & 2 & 2\\
0 & 0 & 1 & 0 & 2 & 0\\
0 & 0 & 1 & 1 & 0 & 0\\
1 & 0 & 1 & 1 & 1 & 0\\
1 & 0 & 1 & 2 & 2 & 1\\
0 & 1 & 2 & 2 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 1\\
1 & 1 & 0 & 1 & 0 & 1\\
1 & 1 & 1 & 2 & 0 & 1\\
1 & 1 & 2 & 2 & 1 & 2
\end{bmatrix} \quad \mathbf{p} = \begin{bmatrix}
1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0
\end{bmatrix}$$

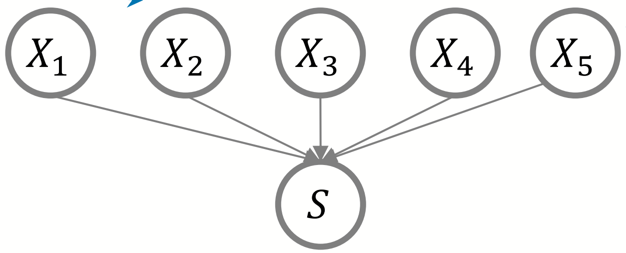
MBNPy package
Inference by Matrix-based Bayesian network

Variable elimination

$\mathbf{C} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} 0.05 \\ 0.95 \end{bmatrix}$

$\mathbf{C} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

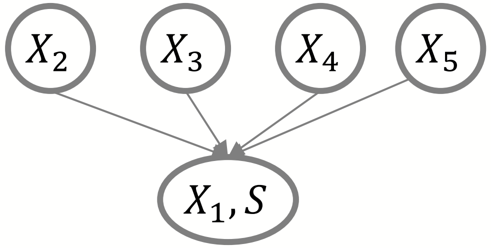
$\mathbf{p} = \begin{bmatrix} 0.01 \\ 0.99 \end{bmatrix}$



$\mathbf{C} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} 0.10 \\ 0.90 \end{bmatrix}$

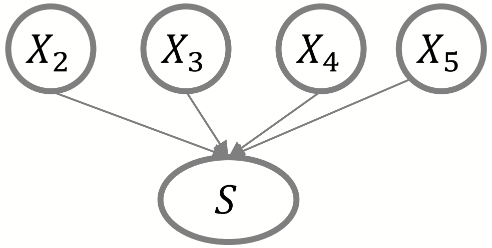
Given elimination order $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5$,

Product



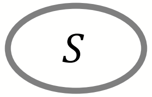
$$P(S|X_1, X_2, \dots, X_5) \cdot P(X_1) = P(S, X_1|X_2, \dots, X_5)$$

Sum



$$\sum_{X_1} P(S, X_1|X_2, \dots, X_5) = P(S|X_2, \cdots, X_5)$$

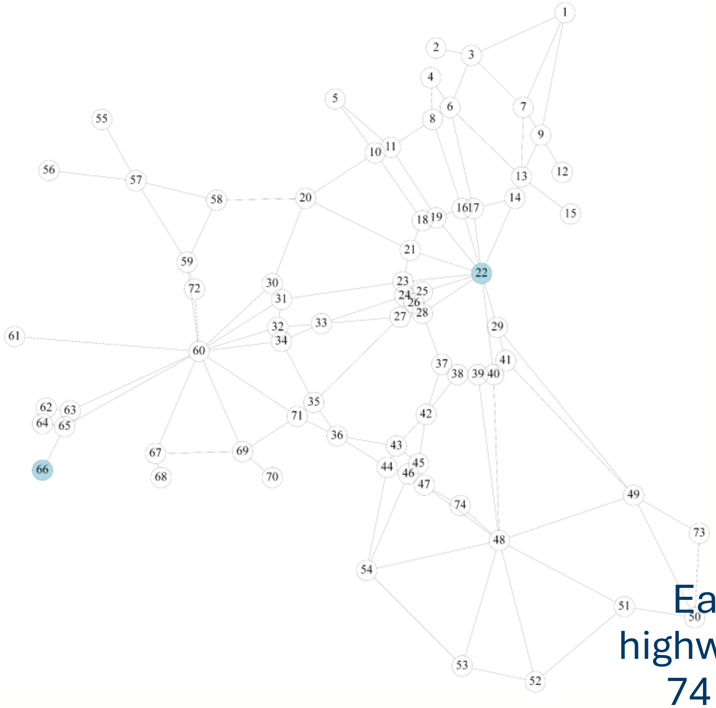
...



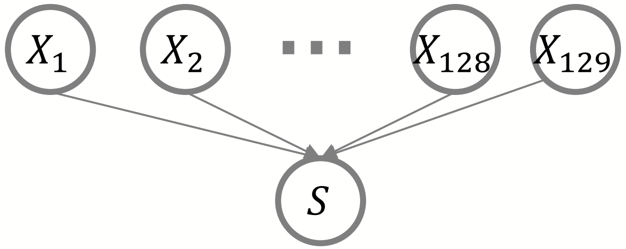
$P(S)$ $\mathbf{C} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} 5.17 \cdot 10^{-3} \\ 9.95 \cdot 10^{-1} \end{bmatrix}$

EMA benchmark highway network

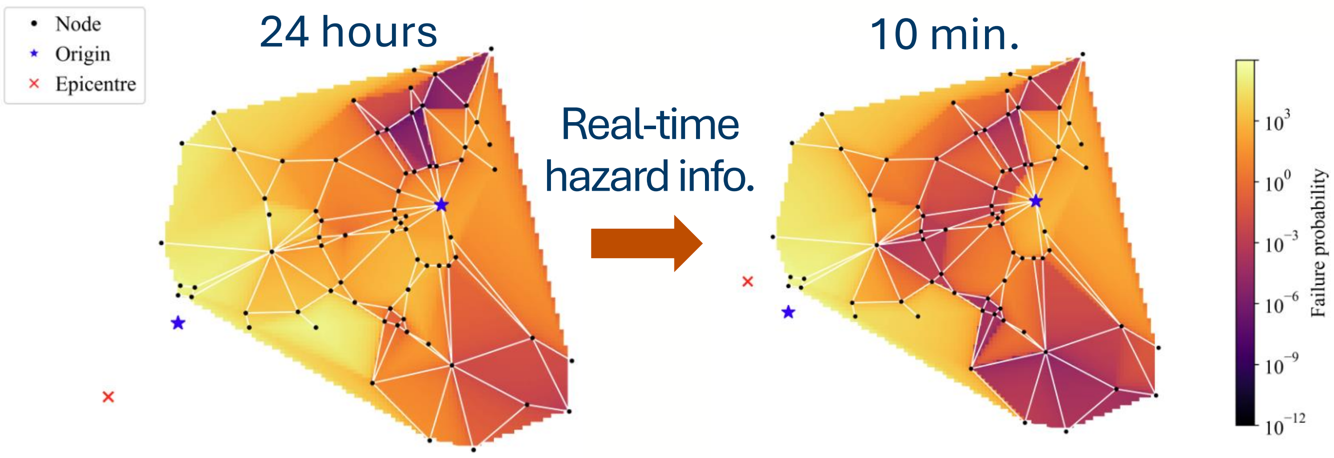
- System failure: more than twice the usual time to airports (blue nodes) → 72 system events
- Termination: 5% bound width w.r.t. lower bound or more than 50,000 branches → MCS on unspecified branches.

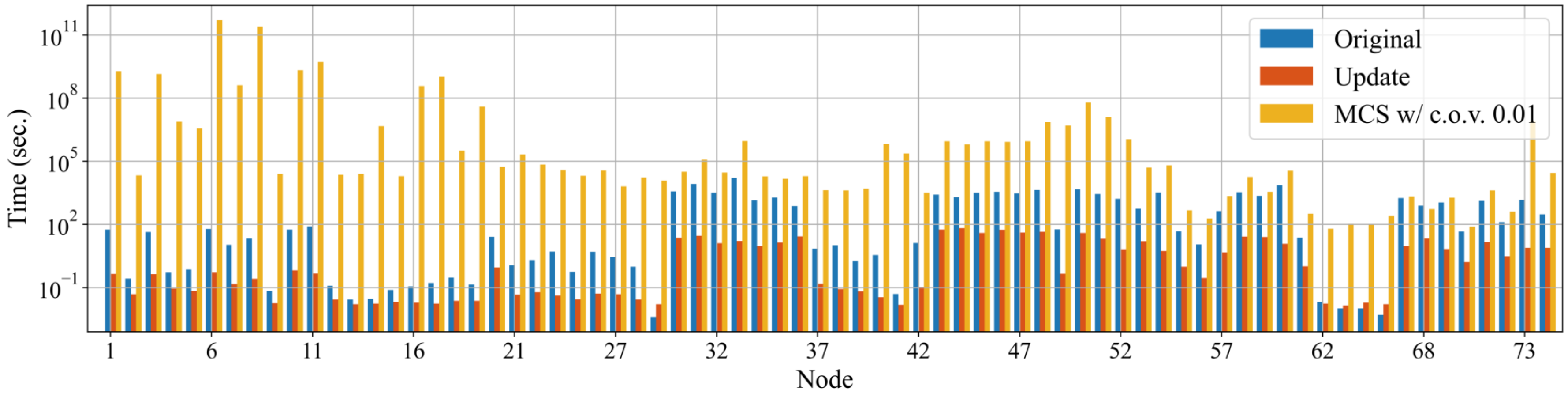


Eastern Massachusetts highway benchmark network: 74 nodes and 129 edges



EMA benchmark highway network

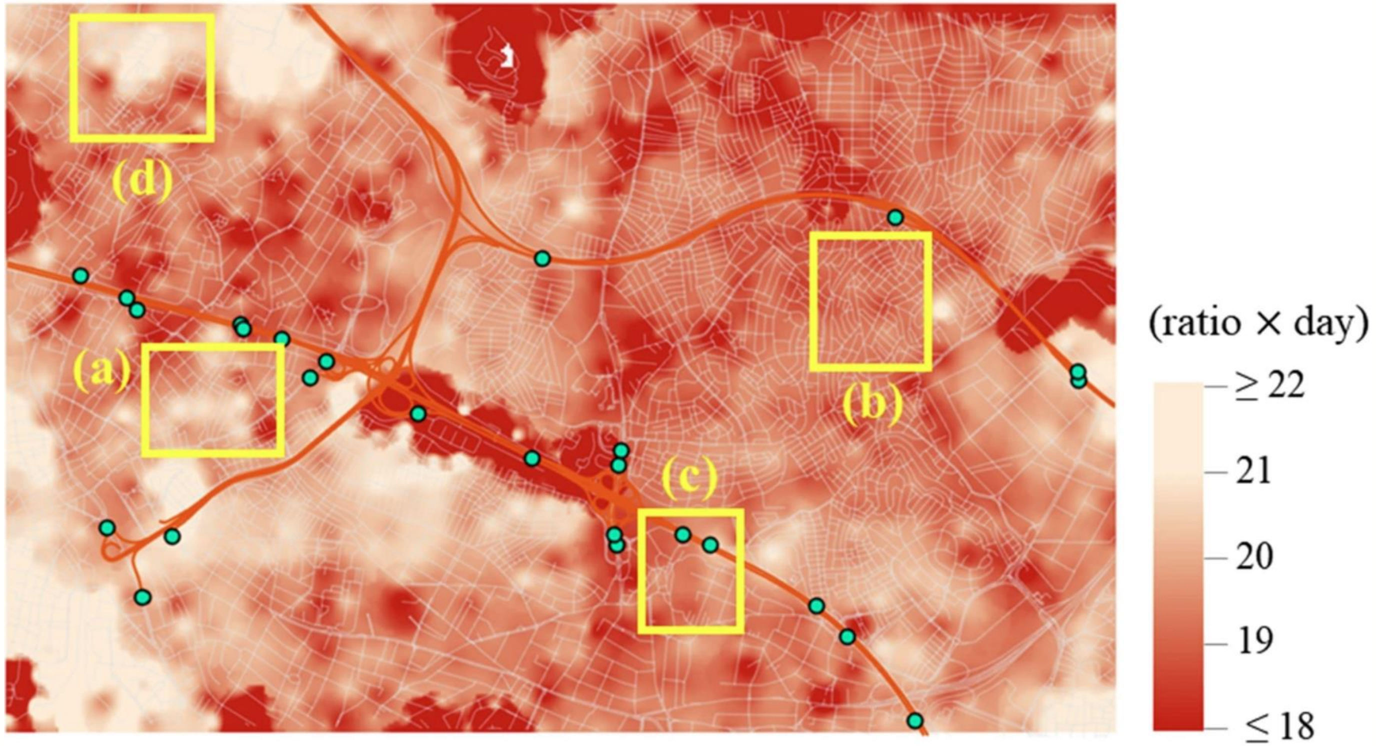




Computation time for original and update scenarios and MCS

Full-scale community-level risk analysis

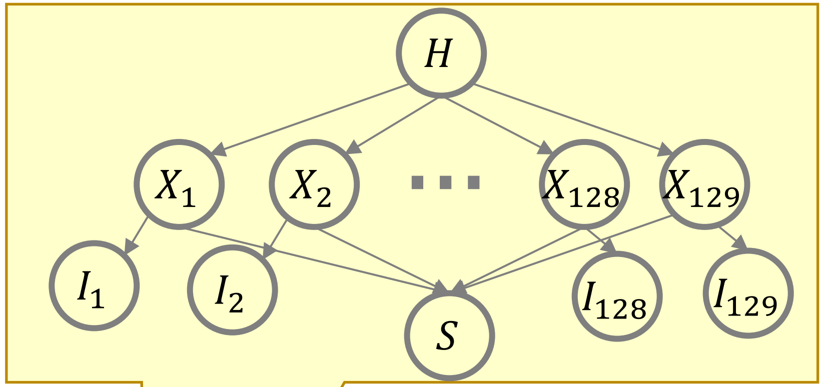
Map Image from Google, © 2021 Istanbul, Turkey



Reference: Byun, J. E. and D'Ayala, D. (2022). Urban seismic resilience mapping: a transportation network in Istanbul, Turkey. *Scientific reports*, *12*(1), 8188.

University of Glasgow

Conclusion



- Summary of Day 1
  - Challenge in system risk assessment – exponential cost w.r.t. # components
  - Introduced solution: decomposition method
  - Implementation using MBNPy package
  - Large-scale benchmark application
  - NB More detailed analyses are possible by adding more nodes.
- Between Day 1 and Day 2:
  https://github.com/jieunbyun/mbnpy_tutorial
  - 01_brc tutorial.ipynb – application of the tutorial example (est. 30 minutes)
  - 02_school reliability.ipynb – application to school complex reliability (est. 1 hour)
- Day 2

If you have any questions while trying,
drop me an email: ji-eun.byun@glasgow.ac.uk

Ji-Eun Byun | Lecturer in Smart Sustainable and Resilient Infrastructure