TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO MÔN HỌC

BÀI TẬP LỚN KTLT

TÌM HIỂU PROBABILISTIC GRAPHICAL MODELS VÀ THƯ VIỆN PGMPY

Giáo viên hướng dẫn: PSG.TS. Huỳnh Quyết Thắng

Nhóm sinh viên thực hiện : Đỗ Tiến Đạt
Hoàng Thành Đạt
Hoàng Trung Dũng
Nguyễn Minh Quang

Lớp : KSTN CNTT K60

HÀ NỘI
Ngày 16 tháng 5 năm 2017

Mục lục

Đề bài

Dựa vào Mô hình mạng Bayes cho dự đoán lỗi và độ tin cậy của phần mềm (mô hình 1) (coi như 1 time frame), đề xuất mô hình phụ thuộc thời gian với 3 time frame của mô hình 1. Lập trình: xây dựng mạng Dynamic Bayesian Network cho mô hình đề xuất trên. So sánh các phân phối có điều kiện tính được của 2 mô hình (1 time frame và 3 time frame).
Gợi ý: thử xây dựng mạng dynamic theo 2 phương pháp:

a) Định nghĩa mạng Dynamic như như một mạng Bayes bình thường với số nút x3.

b) Định nghĩa mạng Dynamic theo thư viện.

Giới thiệu chung

Mô hình xác suất dạng đồ thị là một mô hình xác suất sử dụng đồ thị để biểu diễn phụ thuộc có điều kiện giữa các biến ngẫu nhiên một cách trực quan.

Có hai nhóm mô hình xác suất đồ thị chính bao gồm: Bayesian Networks biểu diễn quan hệ tương quan có chiều (nhân quả) thông qua một đồ thị có hướng (vì thế hay còn được gọi là mô hình đồ thị có hướng) và ss chỉ biểu diễn quan hệ tương quan mà không nêu rõ quan hệ nhân quả (tương ứng còn được gọi là mô hình đồ thị vô hướng).

1 Lý thuyết về probabilistic graphical models

1.1 Mạng Bayes

Mạng Bayes (Bayesian network hoặc Bayesian belief network hoặc belief network) là một đồ thị có hướng phi chu trình mà trong đó:

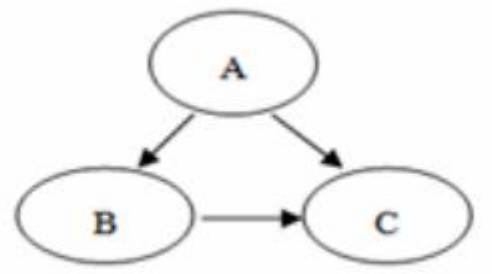
- Các nút biểu diễn các biến.
- Các cạnh biểu diễn các quan hệ xác xuất phụ thuộc (conditional probability distribution - CPD) giữa các biến và phân phối xác suất địa phương cho mỗi giá trị nếu cho trước giá trị của các cha của nó.

Nếu có một cạnh từ nút A tới nút B, thì biến B phụ thuộc trực tiếp vào biến A, và A được gọi là cha của B. Nếu với mỗi biến Xi, tập hợp các biến cha được ký hiệu bởi parents(Xi), thì phân phối có điều kiện phụ thuộc của các biến là tích của các phân phối địa phương.

$$\Pr(X1,...,Xn)=\Pi_{i=1}^{n} \Pr(Xi|\text{parents}(Xi))$$

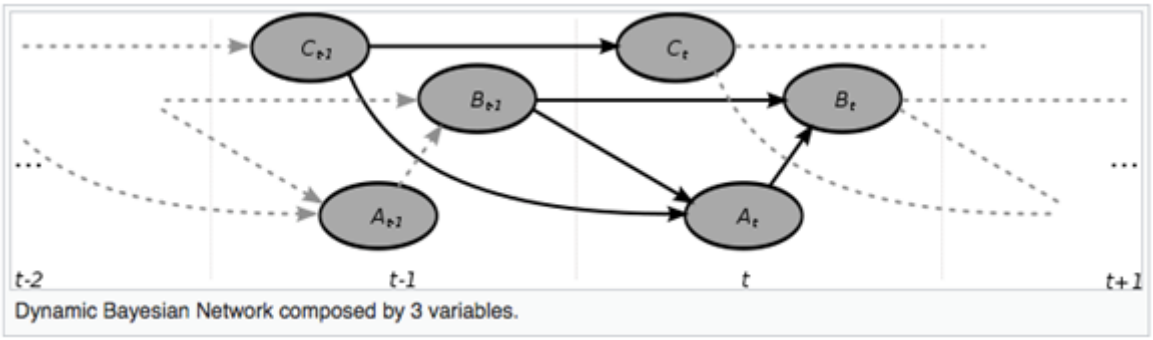
Nếu Xi không có cha, ta nói rằng phân phối xác suất địa phương của nó là không có điều kiện, ngược lại thì gọi là có điều kiện. Nếu biến được biểu diễn bởi một nút được quan sát, thì ta nói rằng nút đó là một chứng cứ (evidence node).

Ví dụ về mạng Bayes có 3 nút A, B, C: nút A nhận 2 giá trị Đúng|Sai, nút B nhận các giá trị Thấp|Trung bình|Cao có chứng cứ là A và nút C nhận các giá trị Bật|Tắt có chứng cứ là A và B.

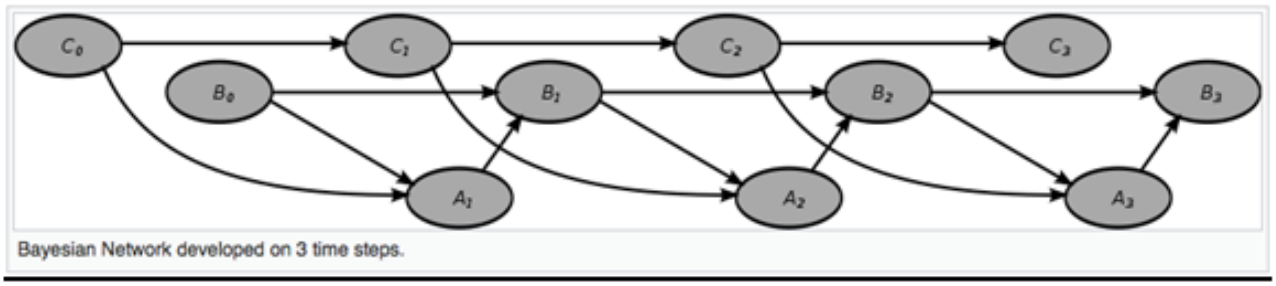


1.2 Mạng Bayes động

Mạng Bayes động (Dynamic Bayesian Network - DBN) là mạng Bayes mà ở đó các nút ở các thời điểm liên tiếp nhau sẽ có quan hệ ảnh hưởng đến nhau. Cụ thể: các phân phối xác suất của mọi nút ở mọi thời điểm T là xác định nếu đã biết phân phối xác suất của các nút liên quan đến nó ở thời điểm T và T-1.



Mạng Bayes động với 3 biến

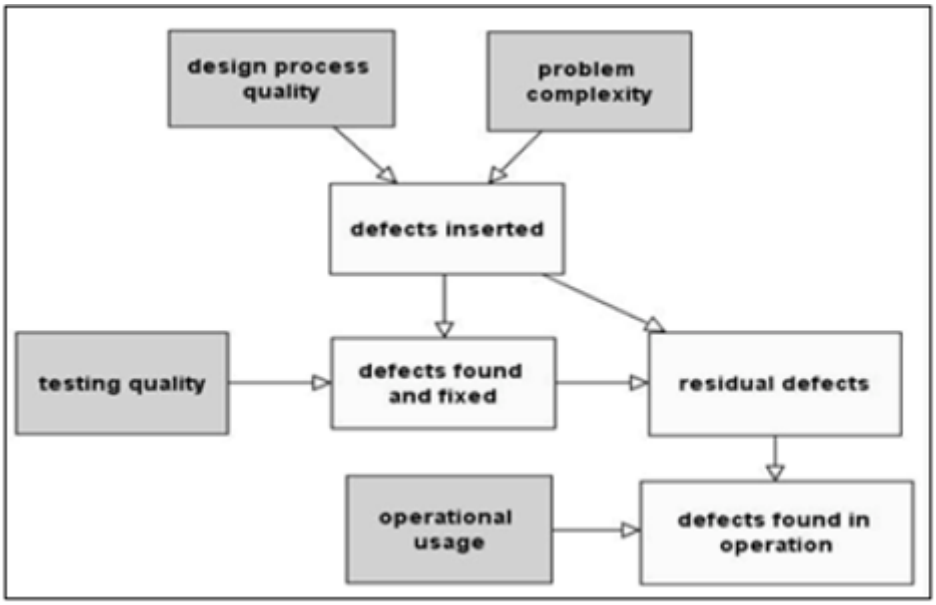


Mạng Bayes động với 3 time step

2 Xây dựng mạng Bayes cho bài toán dự đoán lỗi và độ tin cậy của phần mềm với thư viện pgmpy

2.1 Mô hình mạng Bayes cho dự đoán lỗi và độ tin cậy của phần mềm

Mô hình mạng Bayes cho dự đoán lỗi và độ tin cậy của phần mềm được mô tả bằng hình vẽ bên dưới:



Model 1

Mô hình bao gồm:

- Các nút :

  - TQ: Test quality
  - DPQ: design process quality
  - C: complexity
  - DI: defects inserted

  Là các nút đã biết giá trị 0-4 (nếu không biết giá trị thì coi như phân phối đều với xác suất mỗi sự kiện là 0.2)

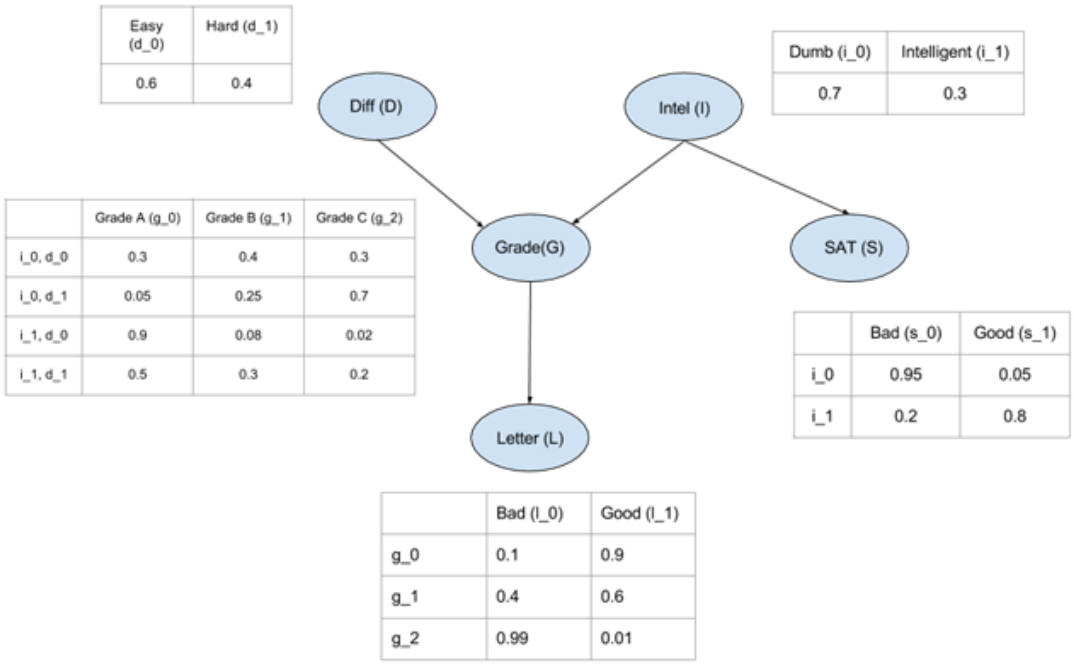
  - DFT: defects found in testing
  - RD: residual defects
  - OU: operational usage
  - DFO: defects found in operation

  Là các nút cần ước lượng phân phối xác suất.

- Các cạnh :
  DPQ -> DI, C -> DI, DI -> DFT, TQ -> DFT, DI -> RD, DFT -> RD, RD -> DFO, OU -> DFO

2.2 Cấu trúc mạng trong pmgpy

Trong pgmpy chúng ta xác định cấu trúc mạng và các phân phối xác suất cố định Conditional Probability Distributions (CPD) riêng biệt và sau đó liên kết chúng với cấu trúc. Đây là một ví dụ để xác định mô hình sinh viên:



Easy (d_0)	Hard (d_1)
0.6	0.4

Dumb (i_0)	Intelligent (i_1)
0.7	0.3

	Grade A (g_0)	Grade B (g_1)	Grade C (g_2)
i_0, d_0	0.3	0.4	0.3
i_0, d_1	0.05	0.25	0.7
i_1, d_0	0.9	0.08	0.02
i_1, d_1	0.5	0.3	0.2

	Bad (s_0)	Good (s_1)
i_0	0.95	0.05
i_1	0.2	0.8

	Bad (l_0)	Good (l_1)
g_0	0.1	0.9
g_1	0.4	0.6
g_2	0.99	0.01

In [6]:
```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
model = BayesianModel([('D', 'G'), ('I', 'G'), ('G', 'L'), ('I', 'S')])
cpd_d = TabularCPD(variable='D', variable_card=2, values=[[0.6, 0.4]])
cpd_i = TabularCPD(variable='I', variable_card=2, values=[[0.7, 0.3]])
cpd_g = TabularCPD(variable='G', variable_card=3,
                   values=[[0.3, 0.05, 0.9,  0.5],
                           [0.4, 0.25, 0.08, 0.3],
                           [0.3, 0.7,  0.02, 0.2]],
                   evidence=['I', 'D'],
                   evidence_card=[2, 2])

cpd_l = TabularCPD(variable='L', variable_card=2,
                   values=[[0.1, 0.4, 0.99],
                           [0.9, 0.6, 0.01]],
                   evidence=['G'],
                   evidence_card=[3])

cpd_s = TabularCPD(variable='S', variable_card=2,
                   values=[[0.95, 0.2],
                           [0.05, 0.8]],
                   evidence=['I'],
                   evidence_card=[2])

model.add_cpds(cpd_d, cpd_i, cpd_g, cpd_l, cpd_s)

model.check_model()
```

Out[6]:
```
True
```

In [7]:
```
model.get_cpds()
```

Out[7]:
```
[<TabularCPD representing P(D:2) at 0x7f82b084c208>,
<TabularCPD representing P(I:2) at 0x7f82b0840eb8>,
 <TabularCPD representing P(G:3 | I:2, D:2) at 0x7f82b084c128>,
<TabularCPD representing P(L:2 | G:3) at 0x7f82b0840e80>,
<TabularCPD representing P(S:2 | I:2) at 0x7f82b0840d30>]
```

In [8]:
```
print(model.get_cpds('G'))
```

I	I_0	I_0	I_1	I_1
D	D_0	D_1	D_0	D_1
G_0	0.3	0.05	0.9	0.5
G_1	0.4	0.25	0.08	0.3
G_2	0.3	0.7	0.02	0.2

In [10]:
```
model.get_cardinality('G')
```

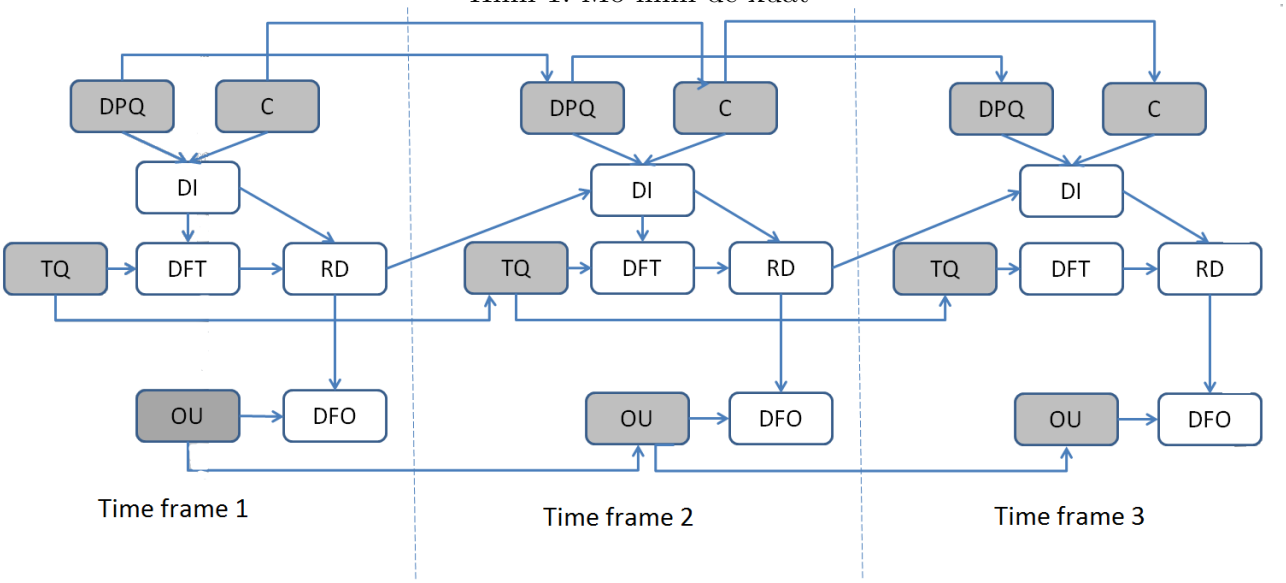
Out[10]:
```
3
```

End of document

3 Mô tả chương trình

3.1 Mô hình đề xuất

Hình 1: Mô hình đề xuất



Trong đó, các nút DPQ, C, TQ, OU có giá trị giữ nguyên ở các thời điểm. Theo nhóm hiểu, mô hình này biểu diễn cho việc thực hiện kiểm thử nhiều lần, do đó, chất lượng thiết kế và độ phức tạp, chất lượng bộ test là không đổi, có thể thay đổi đôi chút song cũng không đáng kể phải tăng, giảm giá trị TQ (do cùng nhóm người viết chương trình kiểm thử). Sau khi kiểm thử lần 1, số lỗi dư thừa sẽ bằng số lỗi phát sinh ở time frame 2, thực hiện kiểm thử lần 2, số lỗi dư thừa của lần 2 bằng số lỗi phát sinh ở lần kiểm thử thứ 3. OU là tình huống giả định cho mức sử dụng của người dùng, vì vậy nó được giữ không đổi. Nếu người dùng sử dụng phần mềm sau khi đã được kiểm thử nhiều lần, chắc chắn số lỗi tìm thấy trong lúc vận hành sẽ nhỏ hơn kiểm thử một lần (DFO3 phải nhỏ hơn DFO1).

Mạng đề xuất cho bài toán đã được xây dựng theo hai cách (mạng Bayesian với số nút x3 và mạng Dynamic Bayesian) dựa vào thư viện pgmpy, cả hai chương trình đều lấy cùng dữ liệu được sinh ra bởi file *data_generating.py*. Đồng thời, nhóm cũng đã chỉnh sửa thư viện pgmpy để chương trình chạy hiệu quả hơn.

3.2 File sinh dữ liệu - data_generating.py

Dữ liệu được sinh ra cho tất cả các nút có trong mạng. Trong đó:

Ở thời điểm 1:

- Các nút DPQ, C, TQ, OU: dữ liệu lấy ngẫu nhiên trong đoạn 0 đến 4, phân phối đều.
- DI: Sinh ngẫu nhiên theo phân phối chuẩn trong đoạn 0 đến gấp hai lần giá trị kỳ vọng. Kỳ vọng μ của DI được lấy phụ thuộc vào DPQ và C. Cụ thể $\mu := mu*(C+1)*(5-DPQ])/10$.
- DFT: Sinh ngẫu nhiên theo phân phối chuẩn phụ thuộc vào DI và TQ. Cụ thể, với các giá trị TQ bằng 4, 3, 2, 1, 0, giá trị μ được lấy tương ứng bằng $0.9, 0.7, 0.5, 0.3, 0.1$. Sau đó lấy ngẫu nhiên giá trị trong đoạn 0 đến 1 theo phân phối chuẩn với kỳ vọng μ và phương sai 0.001. Giá trị ngẫu nhiên này sau đó được nhân với giá trị DI được giá trị DFT.

- RD: Lấy bằng hiệu DI và DFT. DFO = DI - DFT.
- DFO: Lấy phụ thuộc RD và OU. Cách làm tương tự như đối với DFT.

Ở thời điểm 2:

- Các giá trị DPQ2, TQ2, C2, OU2 lấy bằng các giá trị tương ứng DPQ, TQ, C, OU ở thời điểm 1.
- DI2 lấy bằng RD ở thời điểm 1.
- Các nút còn lại sinh ngẫu nhiên tương tự như ở thời điểm 1

Ở thời điểm 3 dữ liệu được sinh tương tự như thời điểm 2.

3.3 Định nghĩa mạng Dynamic như như một mạng Bayes bình thường với số nút x3 - gui.py

Mô tả:

- Định nghĩa mạng bayes
- Lưu lại các giá trị *state_names*, tức là các giá trị có thể có của mỗi nút, ví dụ nút DI có thể nhận các giá trị 0, 1, 3, 4, 5 thì $state\_names['DI'] = [0, 1, 3, 4, 5]$. Điều này cần thiết để vẽ đồ thị phân phối xác suất được chính xác, khi bộ dữ liệu có kích thước nhỏ DI không có đủ các giá trị nguyên trong khoảng 0 đến n có thể làm đồ thị vẽ sai (ví dụ plt.plot(Distribution['DI']) bị sai).
- Xử lý phân đoạn dữ liệu, chia dữ liệu thành các khối nhỏ hơn. Tìm phân phối xác suất của các khối này bằng module VariableElimination sau đó thực hiện tính toán, ta được phân phối xác suất cần biểu diễn.
- Dãn trục hoành các nút để thể hiện giá trị lớn nhất ở trục hoành là số DI lớn nhất, việc này giúp quan sát đồ thị dễ hơn.
- Đồ thị phân phối của 3 thời điểm được vẽ trên 3 figure (do 1 figure chứa quá nhiều khó quan sát). Trong đó, figure 1 thể hiện phân phối các nút ở thời điểm 1 và 2, figure 2 thể hiện phân phối ở thời điểm 2 và 3, figure 3 thể hiện phân phối ở thời điểm 1 và 3.

3.4 Định nghĩa mạng Dynamic theo module có sẵn trong pgmpy - guiDynamic.py

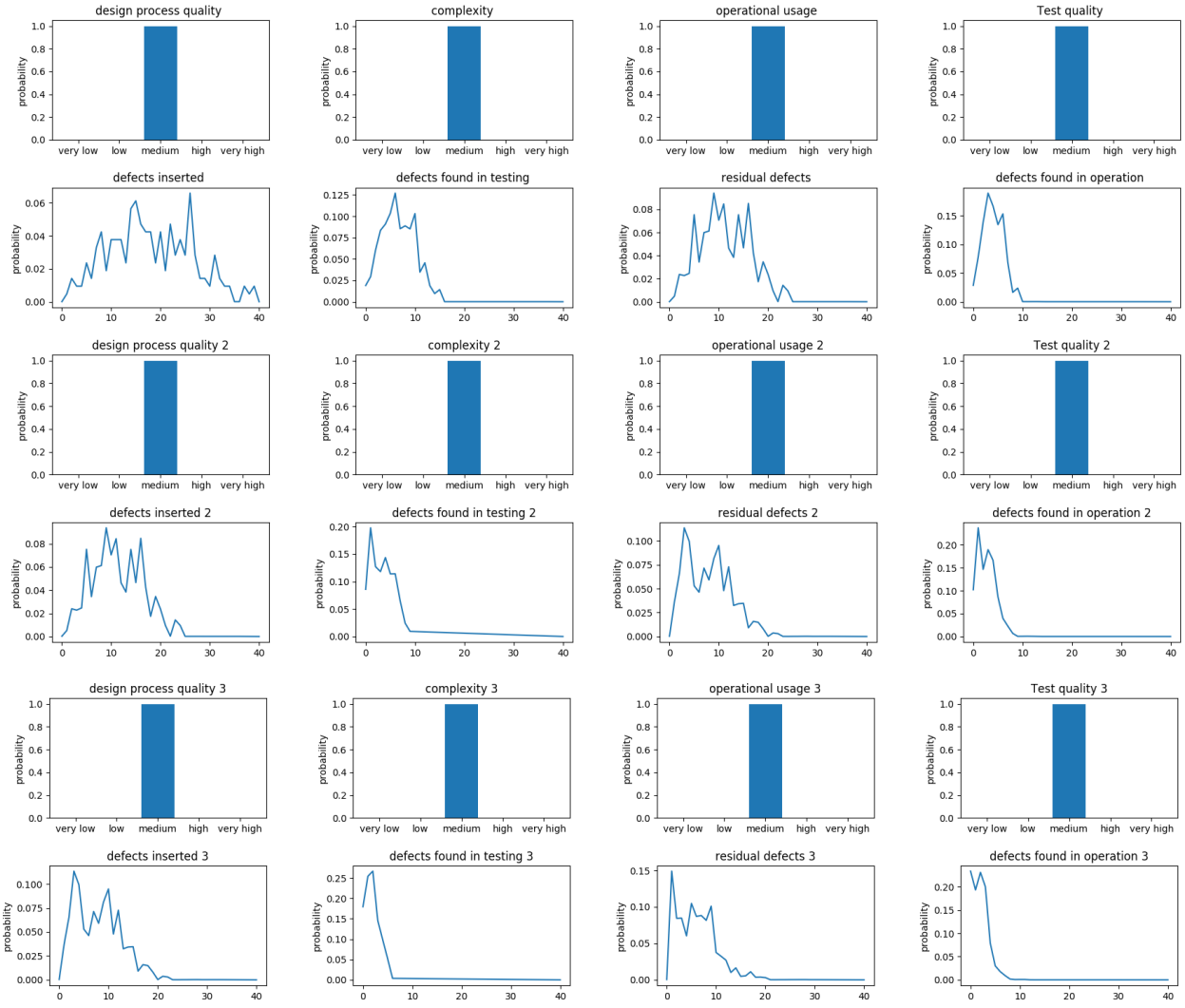
Mô tả: Module DynamicBayesianNetwork và DBNInference trong pmgpy hiện đang trong quá trình phát triển, nên nhóm có nhiều vướng mắc không giải quyết triệt để được. Hiện tại nhóm đã định nghĩa mạng Dynamic Bayesian, đọc dữ liệu sinh ra từ file *data_generating.py* và thực hiện tính toán phân phối xác suất có điều kiện tại mỗi nút bằng phương pháp Maximum Likelihood và *add_cpds* vào các nút trong mạng. Do module MaximumLikelyhood lỗi với mạng Dynamic Bayesian và query trong module DBNInference có nhiều lỗi của thư viện, nhóm đã viết thư mục DynamicBayesian (bằng cách hiệu chỉnh mã các module có sẵn) để thao tác với mạng Dynamic Bayesian.

Kết quả thu được đối chiếu với cách xây dựng mạng Bayesian với số nút x3. Trong đó, đồ thị phân phối xác suất của các nút ở thời điểm 1 là giống nhau, đồ thị phân phối xác suất của các nút ở các thời điểm 2 sai khác một chút, và ở thời điểm 3 sai khác khá lớn.

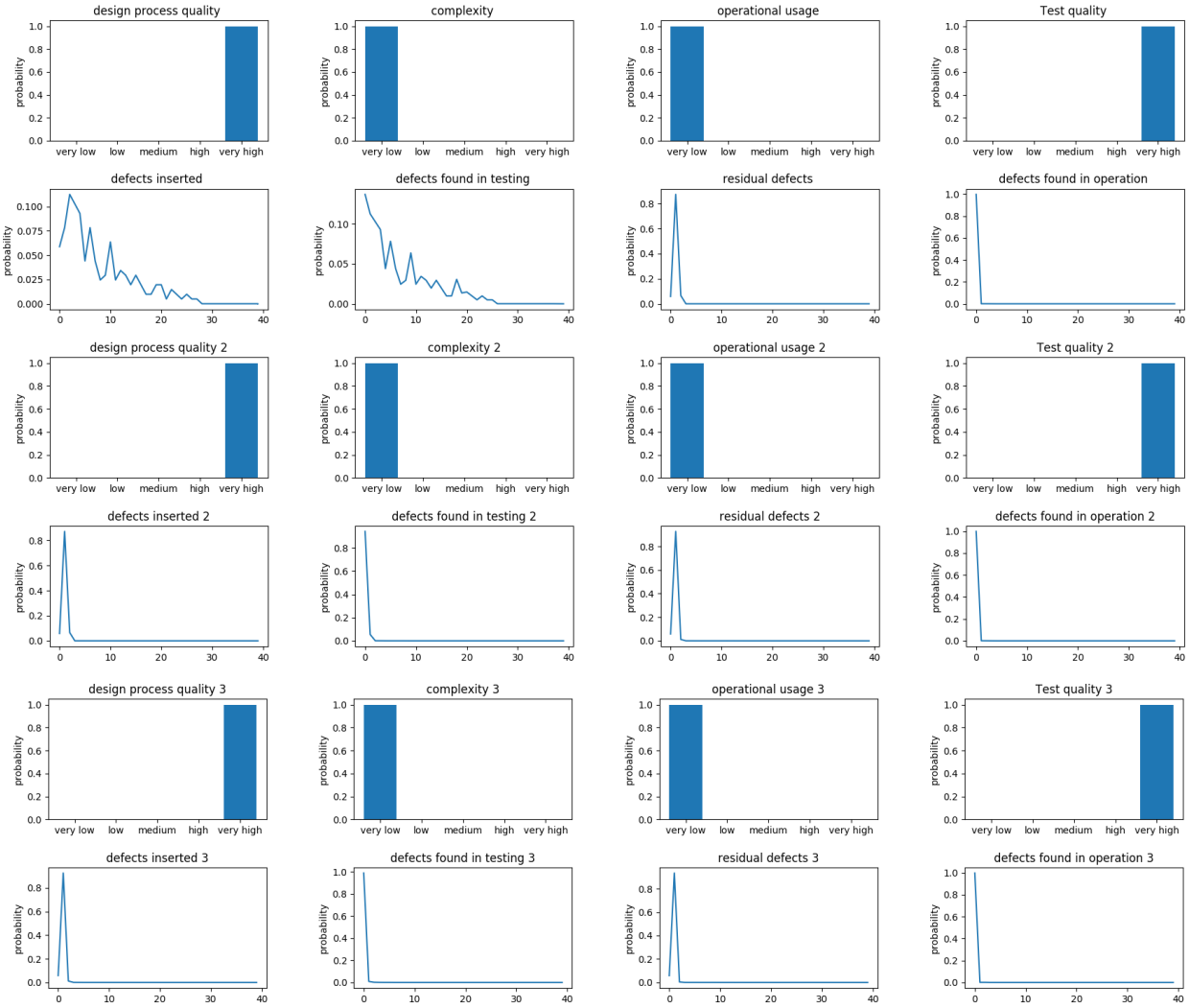
4 Mô tả kết quả thu được

4.1 Phương pháp định nghĩa mạng Bayesian bằng số nút x3

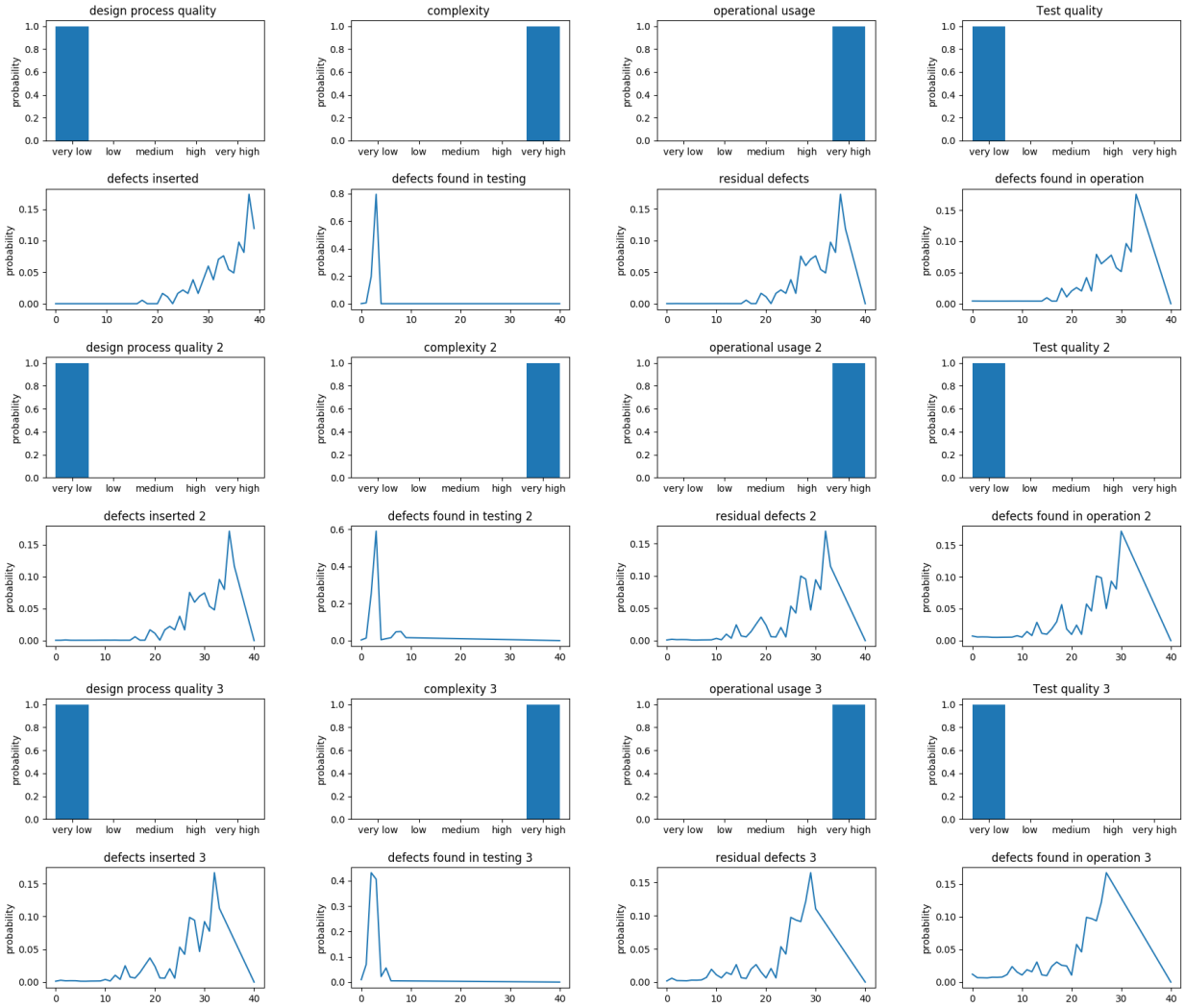
Hình 2: DPQ, C, TQ, OU nhận giá trị tương ứng 2, 2, 2, 2



Hình 3: DPQ, C, TQ, OU nhận giá trị tương ứng 4, 0, 4, 0



Hình 4: DPQ, C, TQ, OU nhận giá trị tương ứng 0, 4, 0, 4



Để kiểm tra tính đúng đắn của mô hình, ta tiến hành thực nghiệm qua 3 lần chạy thử với các giá trị đầu vào (DPQ, C, TQ, OU) lần lượt là (2, 2, 2, 2), (4, 0, 4, 0), (0, 4, 0, 4) và thu được kết quả như các hình 2, 3, 4 ở trên.

Ta thấy ở trường hợp đầu, chất lượng thiết kế và độ phức tạp của chương trình nằm ở mức trung bình, kéo theo số lượng lỗi xuất hiện ở thời điểm đầu tiên cũng nằm ở mức trung bình, xác suất nằm trong khoảng từ 10 đến 30 lỗi là khá cao. Trong khi đó, trường hợp 2 ứng với chất lượng thiết kế rất tốt và độ phức tạp rất thấp thì số lượng lỗi xuất hiện đã ít hơn hẳn, đồ thị phân phối lệch về phía gốc tọa độ, xác suất cao nhất chỉ vào khoảng 2 đến 3 lỗi. Ngược lại, trường hợp 3 ứng với chất lượng thiết kế rất kém, độ phức tạp cao thì trường hợp xuất hiện trên 35 lỗi lại có xác suất cao nhất.

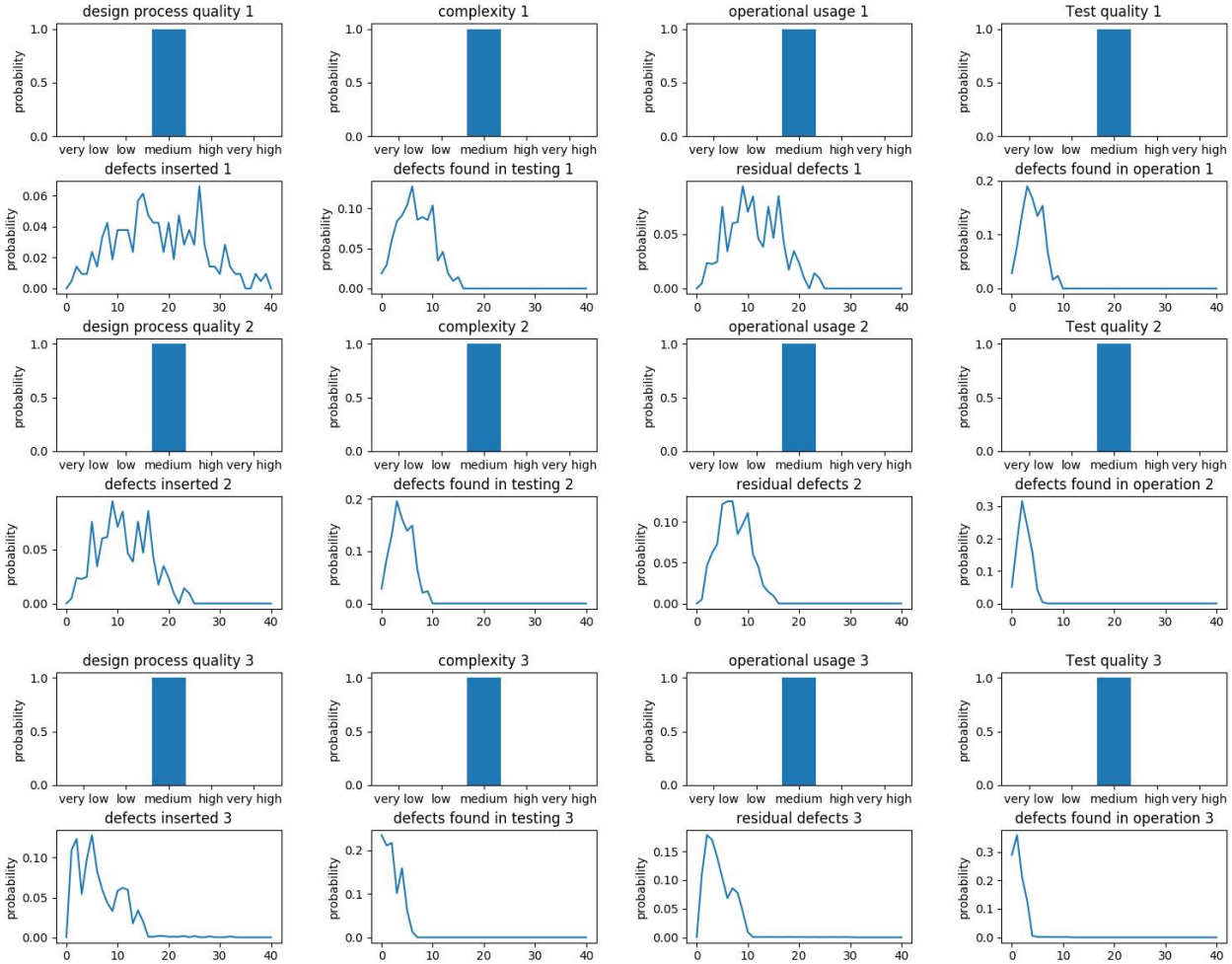
Đối với chất lượng kiểm thử, ở trường hợp đầu, chất lượng kiểm thử ở mức trung bình nên ở thời điểm đầu, số lượng lỗi còn lại vẫn còn khá cao. Qua hình 2 ta có thể thấy, số lượng lỗi đã giảm một nửa. Ở trường hợp 2, chất lượng kiểm thử rất tốt, số lượng lỗi tìm được lớn, gần bằng số lỗi xuất hiện, dẫn đến số lượng lỗi còn lại rất nhỏ. Ở trường hợp cuối thì ngược lại, chất lượng kiểm thử kém, số lượng lỗi tìm được rất ít (chưa quá 5 lỗi), do đó số lượng lỗi còn dư vẫn còn cao.

Cuối cùng, với hiệu suất vận hành trung bình như ở trường hợp đầu, số lỗi tìm thấy khi sử dụng cũng bằng khoảng một nửa số lượng lỗi còn dư. Ở trường hợp 2, hiệu suất vận hành kém, đồng thời số lỗi còn dư cũng thấp nên hầu như không tìm thêm được lỗi trong quá trình dùng. Trường hợp cuối cùng, hiệu suất vận hành cao nên số lượng lỗi tìm thấy tương đương với số lượng lỗi còn dư.

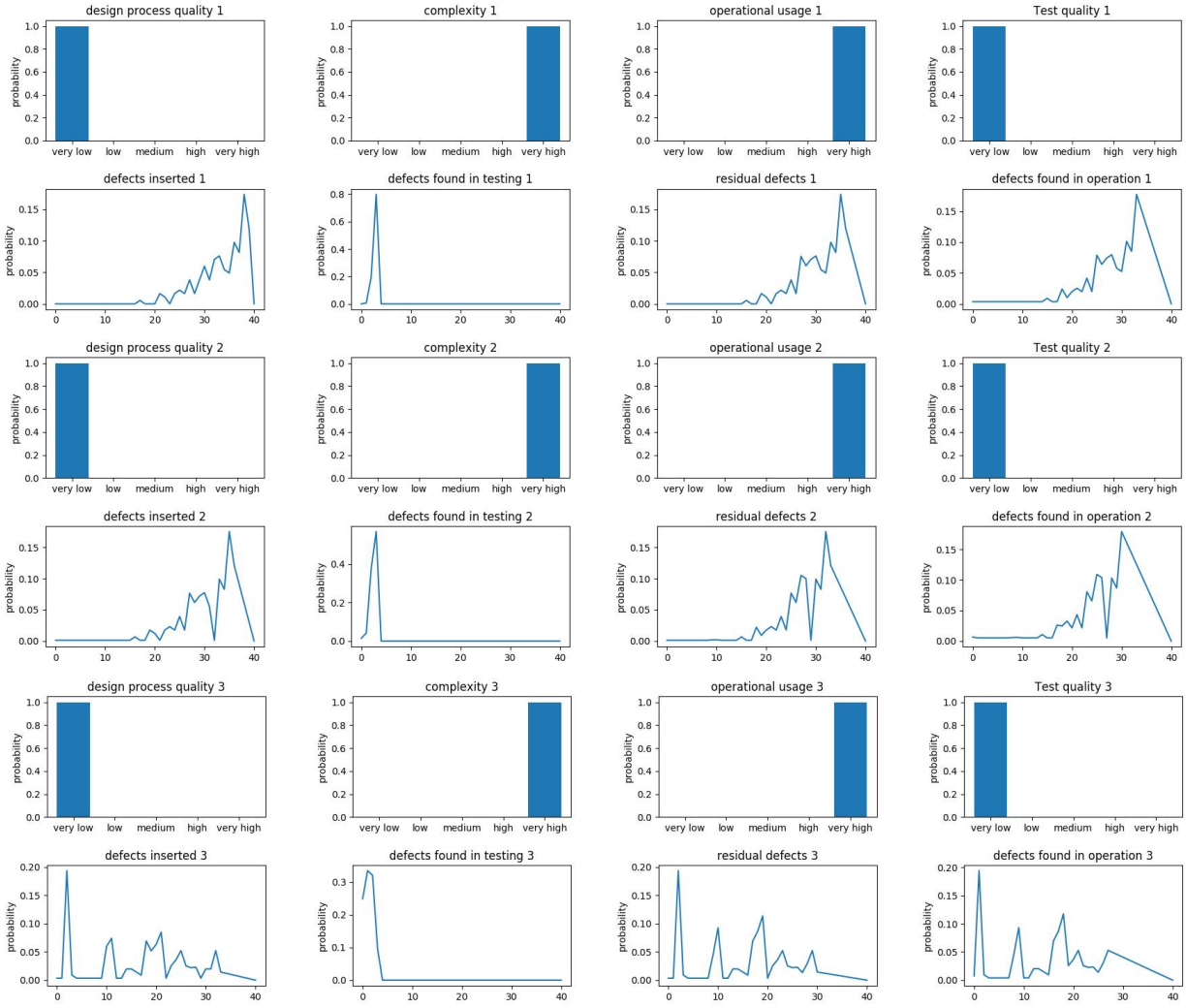
4.2 Phương pháp xây dựng mạng Dynamic Bayesian theo thư viện DynamicBayesian

Kết quả khi sử dụng thư viện pgmpy để định nghĩa mạng Bayes động được thể hiện ở hình 5 và hình 6. Khi so sánh với kết quả ở hình 2 và hình 6 ta có thể thấy đồ thị phân phối xác suất của các nút ở thời điểm 1 là giống nhau, ở thời điểm 2 có sai khác nhỏ và ở thời điểm 3 sai khác khá lớn.

Hình 5: DPQ, C, TQ, OU nhận giá trị tương ứng 2, 2, 2, 2



Hình 6: DPQ, C, TQ, OU nhận giá trị tương ứng 0, 4, 0, 4



Tài liệu tham khảo

[1] Ankur Ankan, Abinash Panda, *Mastering Probabilistic Graphical Models using Python.*

[2] Kevin Patrick Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, http://www.cs.ubc.ca/~murphyk/Thesis/thesis.pdf

[3] Matthias Daniel, Ette Harrison Etuk, *Predicting Software Reliability and Defects Using Bayesian Networks* https://www.eecs.qmul.ac.uk/~norman/papers/defects_plagiarised_ANNOTATED.pdf

[4] Video Probabilistic Graphical Models in Python by Aileen Nielsen: https://www.youtube.com/watch?v=DEHqIxX1Kq4

[5] Hướng dẫn xây dựng mạng Bayes: https://github.com/pgmpy/pgmpy_notebook/tree/master/notebooks