Learning Deep Generative Models of Graphs

School of Industrial and Management Engineering, Korea University

Jong Kook, Heo

KOREA UNIVERSITY

DMQA

Contents

Research Purpose

Overview

- ❖ Learning Deep Generative Models of Graphs(arXiv, 2018)
  - 2022년 3월 11일 기준 418회 인용
  - Pointer Networks 와 Show and Tell 의 제1저자인 Oriol Vinyals 가 참여
  - Synthetic Graph와 Molecular graph 등 다양한 그래프를 생성 할 수 있으며, Conditional Generation 도 일부분 가능 **(본 리뷰에서는 Unconditional Molecular Generation 에 초점을 맞추어 진행)**

Learning Deep Generative Models of Graphs

Yujia Li [1] **Oriol Vinyals** [1] **Chris Dyer** [1] **Razvan Pascanu** [1] **Peter Battaglia** [1]

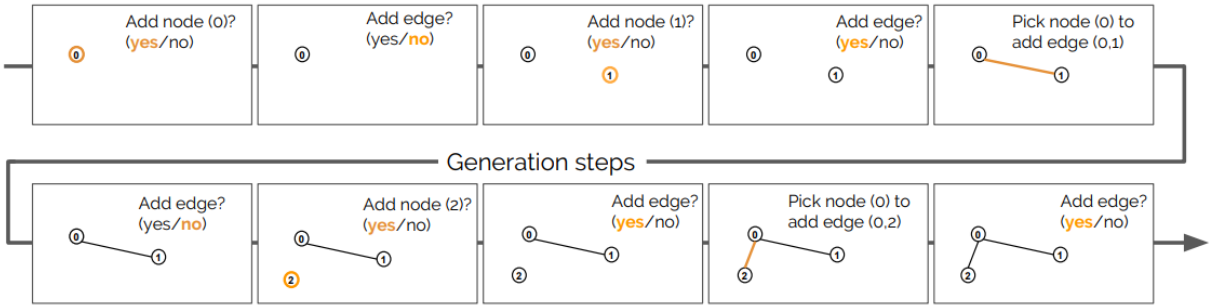


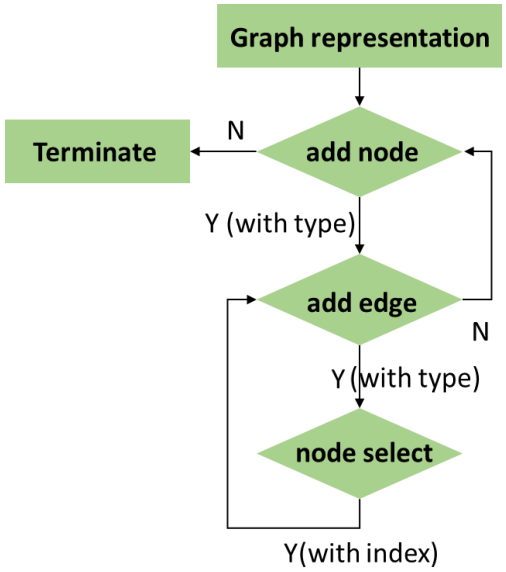
Figure 1. Depiction of the steps taken during the generation process.

DMQA

DGMG

❖ Sequential Graph Generation Process

- DGMG 의 그래프 생성 과정은 크게 3가지 단계로 구성되어 있음
  1. Add Node : 현재까지 생성된 분자에서 **어떤 타입의 원자를 추가**할지 결정(Terminate 포함)
  2. Add Edge : (1) 에서 추가된 원자를 **어떤 타입의 결합으로 연결**할지 결정(Terminate 포함)
  3. Node Select: (1) 과 (2)로 정해진 원자 결합을 기존 분자의 **어디에 연결 할지** 결정



DMQA

DGMG

❖ Sequential Graph Generation Process

- GNN 의 Message Passing 을 이용한 분자 생성을 위해 DGMG 는 5가지 모듈로 구성

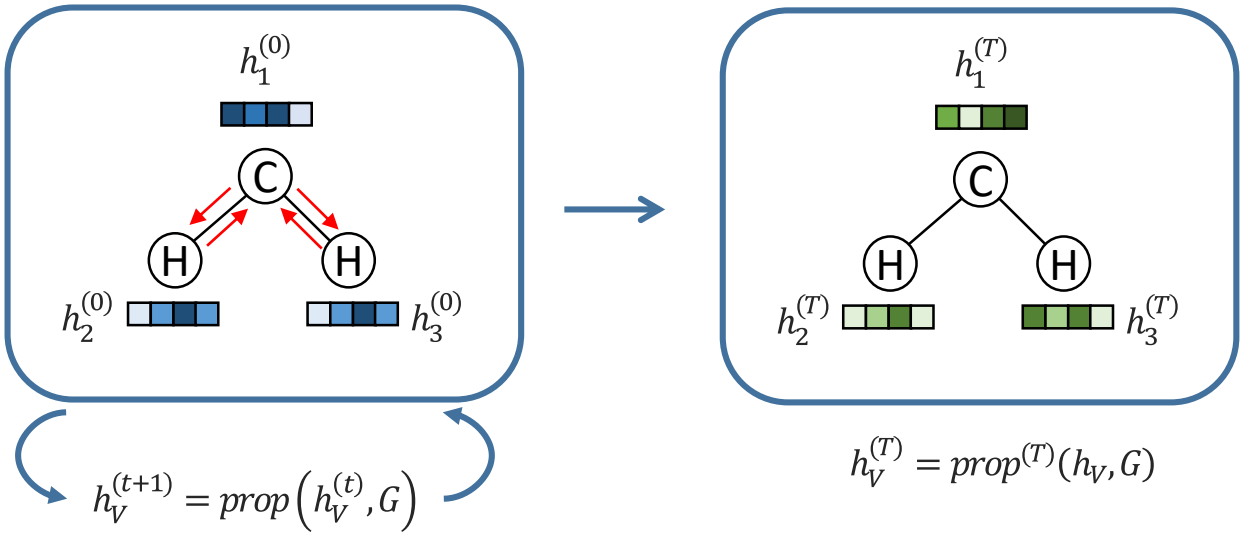
1. Graph Propagation Module : node 와 edge 특성을 T번 반복하여 message passing

- ✓ Incoming message aggregation : $a_v = \sum_{u:(u,v)\in E} f_e(h_u, h_v, x_{u,v})\ \forall v\ in\ V$
- ✓ Node update : $h'_v = f_n(a_v, h_v)\ \forall v\ in\ V$

1 round of propagation
$prop(h_V, G)$

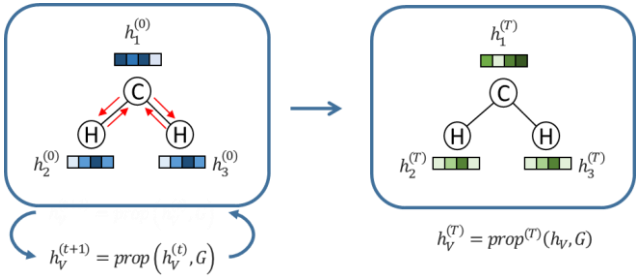
- ✓ $h_V^{(t+1)} = prop\left(h_V^{(t)}, G\right)\ where\ h_V = \{h_1, \ldots, h_{|V|}\}$

$Graph\ G = (V, E)$
$h_v : node\ feature\ for\ v \in V$
$x_{u,v} : edge\ feature\ for\ (u,v) \in E$
$f_e : Concat\ and\ MLP$
$f_n : GRUCell$



DMQA

DGMG



❖ Sequential Graph Generation Process

- GNN 의 Message Passing 을 이용한 분자 생성을 위해 DGMG 는 5가지 모듈로 구성

2. Graph Representation Module : T 번의 Propagation 후의 Graph level representation 을 추출

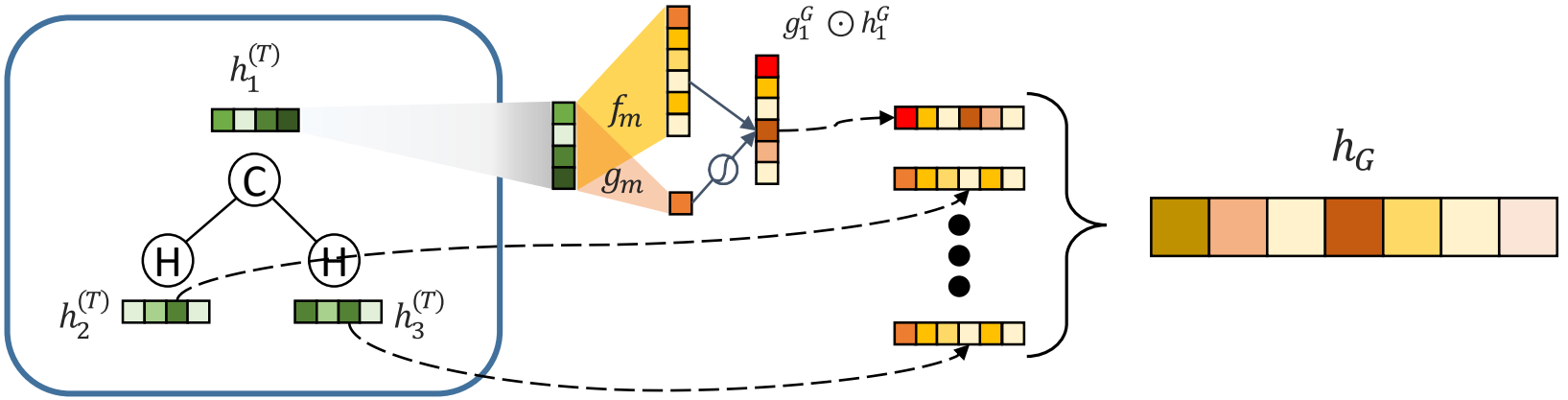
✓ Gate function : $g_v^G = \sigma\left(g_m(h_v)\right)$

✓ Node mapping : $h_v^G = f_m(h_v)$

✓ Gated Sum for Graph Representation : $h_G = \sum_{v \in V} g_v^G \odot h_v^G = R\left(h_V^{(T)}, G\right)$

h_v: *node feature after T rounds of propagation*

g_m, f_m: *Linear mapping function*
σ: *Sigmoid Function*

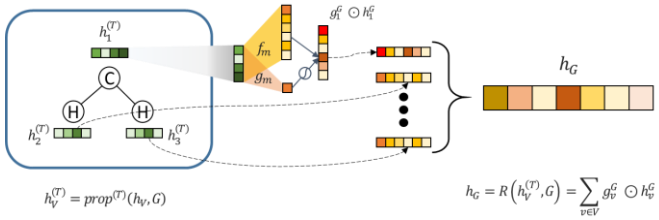


$$h_V^{(T)} = prop^{(T)}(h_V, G)$$

$$h_G = R\left(h_V^{(T)}, G\right) = \sum_{v \in V} g_v^G \odot h_v^G$$

DMQA

DGMG



❖ Sequential Graph Generation Process

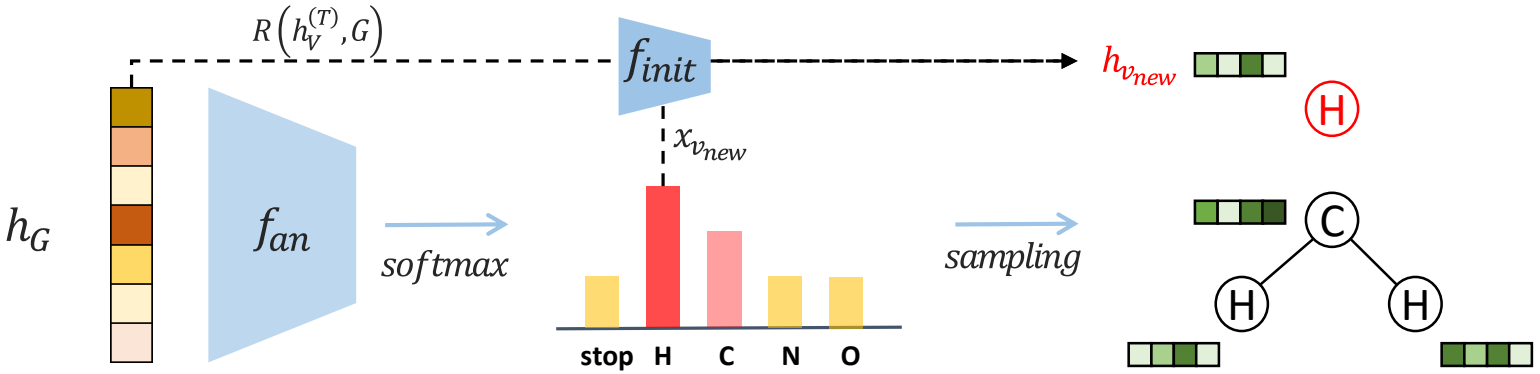
- GNN 의 Message Passing 을 이용한 분자 생성을 위해 DGMG 는 5가지 모듈로 구성

3. Add Node Module : 그래프 정보로부터 새롭게 추가될 원자를 샘플링, 해당 노드 벡터를 초기화

✓ Add Node function : $f_{add\,node}(G) = softmax(f_{an}(h_G))$

✓ Initialization for newly added node : $h_{v_{new}} = f_{init}(R\left(h_V^{(T)}, G\right), x_{v_{new}})$

f_{an} : MLP function
f_{init} : Linear function
x_v : input feature (i.e. atom type)

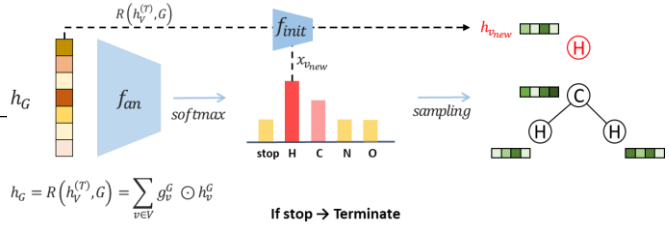


$$h_G = R\left(h_V^{(T)}, G\right) = \sum_{v \in V} g_v^G \odot h_v^G$$

If stop → Terminate

DMQA

DGMG



❖ Sequential Graph Generation Process

- GNN 의 Message Passing 을 이용한 분자 생성을 위해 DGMG 는 5가지 모듈로 구성

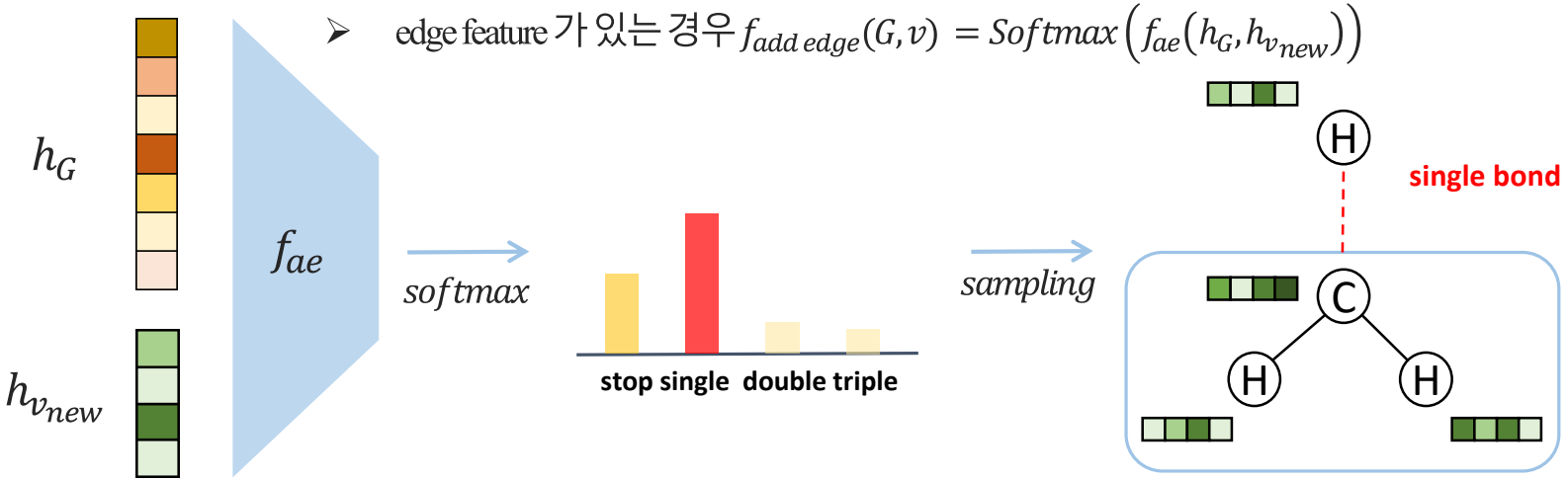
4. Add Edge Module: 그래프 정보와 새롭게 추가될 원자를 입력하여 어떤 타입의 결합으로 연결할지 결정

✓ Add Edge function

f_{ae}: *MLP function*

➢ edge feature 가 없는 경우 $f_{add\,edge}(G,v) = \sigma\left(f_{ae}\left(h_G, h_{v_{new}}\right)\right)$

➢ edge feature 가 있는 경우 $f_{add\,edge}(G,v) = Softmax\left(f_{ae}\left(h_G, h_{v_{new}}\right)\right)$

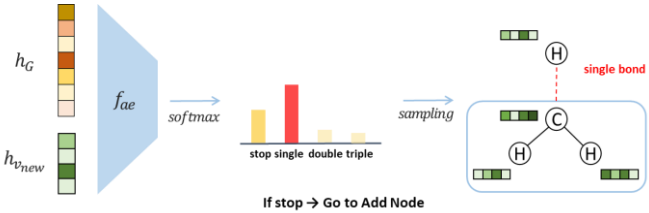


$$h_G = R\left(h_V^{(T)}, G\right) = \sum_{v \in V} g_v^G \odot h_v^G$$

$$h_{v_{new}} = f_{init}(R\left(h_V^{(T)}, G\right), x_v)$$

If stop → Go to Add Node

DGMG



❖ Sequential Graph Generation Process

- GNN 의 Message Passing 을 이용한 분자 생성을 위해 DGMG 는 5가지 모듈로 구성

5. Node Select Module : 새로운 원자와 해당 결합을 기존 그래프의 어떠한 노드에 연결할지 결정

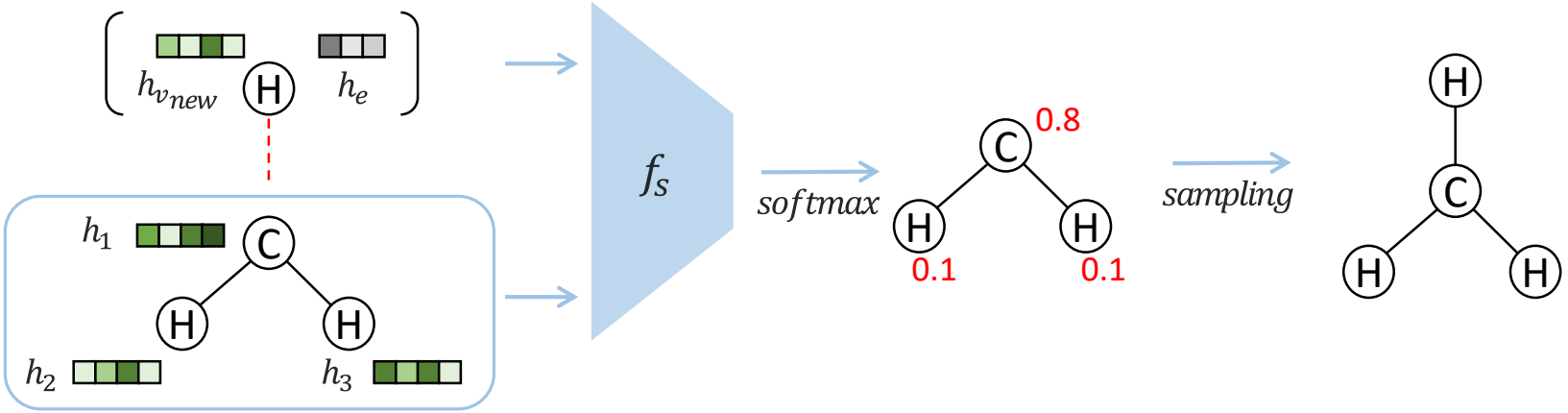
✓ Node Select with Scoring

➢ edge feature 가 있는 경우 $s_u = Softmax\left(f_s\left(h_u, h_{v_{new}}\right)\right) \; \forall u \in V$

➢ edge feature 가 있는 경우 $s_u = Softmax\left(f_s\left(h_u, h_{v_{new}}, h_e\right)\right) \forall u \in V$

f_s : *MLP function*
h_e : *edge feature (i.e. bond type)*

✓ **새로운 edge 가 추가된 후 1. Graph Propagation, 2. Graph Representation, 4. Add Edge 순서로 진행**



Experiments

❖ Training Results

- Maximize Log-likelihood
  - ✓ 하나의 Graph 는 node indexing 에 따라 여러가지 순서로 표현이 가능 → Possible Orderings = (# of nodes)!
  - ✓ 따라서 node ordering strategy(π) 의 분포까지 고려하여 계산하여야함
    - ➢ 만약 ordering 의 분포를 알 수 없다면 Random Ordering 이라 가정
    - ➢ 분자 그래프의 경우 Canonical SMILES 라는 순서로 원자 순서를 표기 가능(Fixed Ordering)
    - ➢ Fixed Ordering 이 존재할 경우 Negative Log-likelihood 가 더 작음**(학습이 더 잘됨)**

$$\mathbb{E}_{p_{data}(G,\pi)}[\log p(G,\pi)] = \mathbb{E}_{p_{data}(G)}\mathbb{E}_{p_{data}(\pi|G)}[\log p(G,\pi)]$$

Table 2. Molecule generation results. N is the number of permutations for each molecule the model is trained on. Typically the number of different SMILES strings for each molecule < 100.

Arch	Grammar	Ordering	N	NLL	%valid	%novel
LSTM	SMILES	Fixed	1	21.48	93.59	81.27
LSTM	SMILES	Random	< 100	**19.99**	93.48	83.95
LSTM	Graph	Fixed	1	22.06	85.16	80.14
LSTM	Graph	Random	$O(n!)$	63.25	91.44	91.26
Graph	Graph	Fixed	1	20.55	**97.52**	90.01
Graph	Graph	Random	$O(n!)$	58.36	95.98	**95.54**

Experiments

❖ Evaluation of Generated Molecules(논문 외)

- DGMG Generation with ZINC dataset
  - ✓ ZINC Dataset : Molecular Distribution Learning & Goal-Directed Generation 을 위한 분자 데이터셋
  - ✓ 그림 1. : Train Dataset
  - ✓ 그림 2. : DGMG 를 통해 생성한 분자(**빈 칸은 화학적으로 유효하지 않은 분자를 생성한것으로 추정됨**)
  - ✓ 그림 3. 합성 가능성(SA), Drug-likeness(QED) 등 분자의 화학 지표에 대한 학습 데이터셋과 생성된 분자의 분포 비교

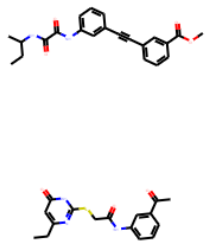


그림 1.

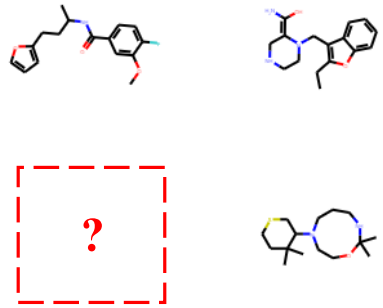


그림 2.

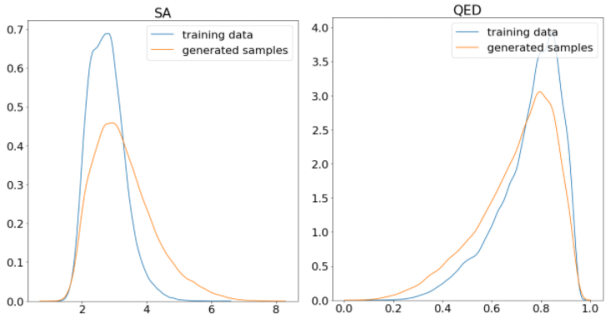


그림 3.

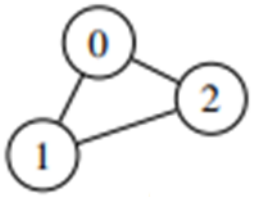
DMQA

Conclusion

❖ Limitation

- Validity Problem : DGMG 는 화학적 유효성에 대한 가정이 생성 과정에 들어가있지 않기 때문에 invalid 한 분자를 생성하는 경우가 발생

- Long Sequence Problem : DGMG 는 생성 과정이 매우 길기 때문에 큰 분자를 잘 생성하지 못함




```
<add node (node 0)>
  <don't add edge>
<add node (node 1)>
  <add edge>
  <pick node 0 (edge (0, 1))>
  <don't add edge>
<add node (node 2)>
  <add edge>
  <pick node 0 (edge (0, 2))>
  <add edge>
  <pick node 1 (edge (1, 2))>
  <don't add edge>
<don't add node>
```

13 steps

- Ordering Problem : 하나의 그래프를 표현하거나 생성하는 과정 무수히 많은 Permutation 이 존재하기 때문에 학습이 어려움

$Max\ P(e_{2,3}|G)$ ≠ $Max\ P(e_{1,3}|G)$