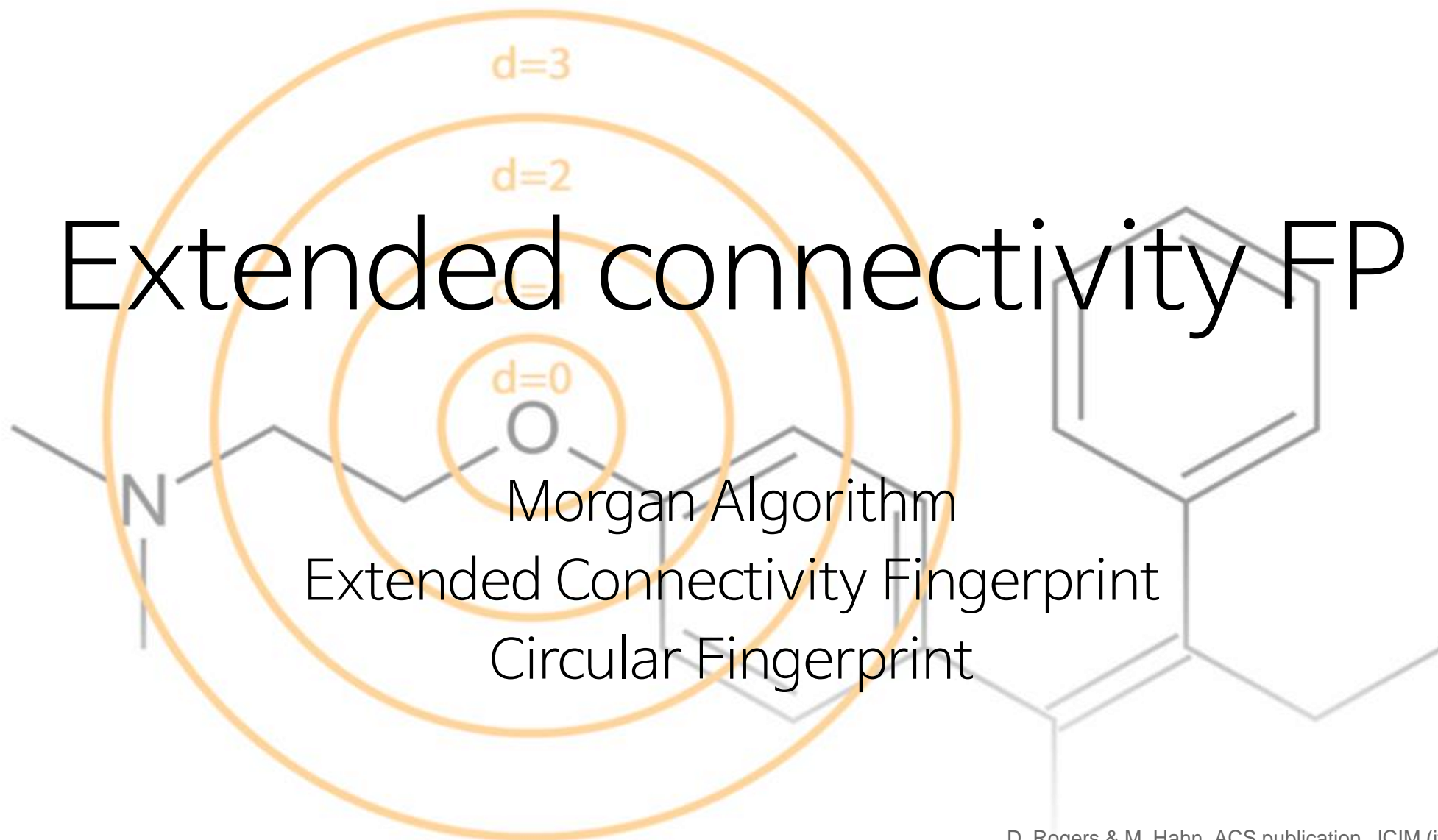


Extended connectivity FP



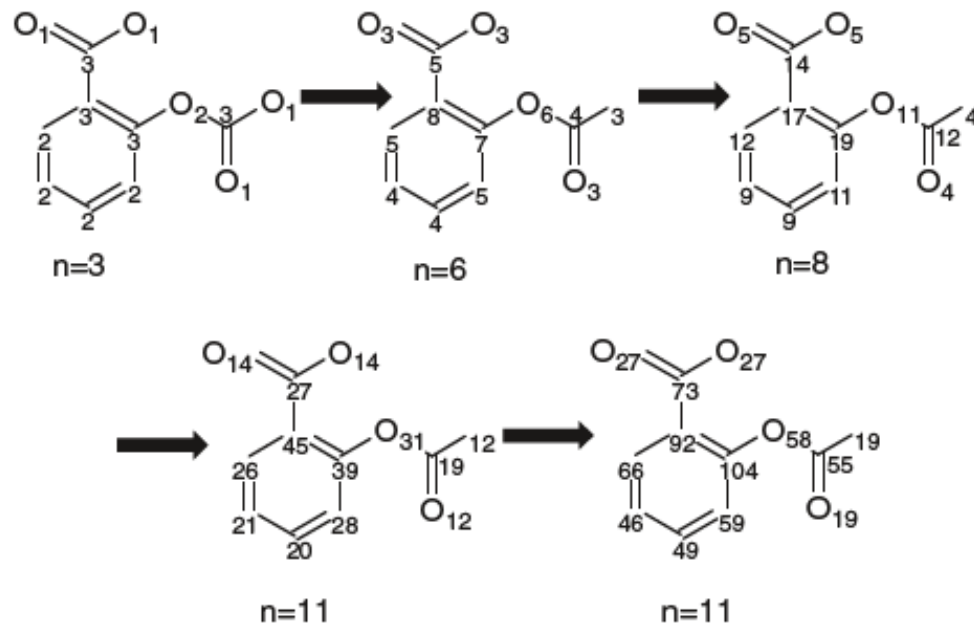
D. Rogers & M. Hahn, ACS publication, JCI (if=3.97), 2010.

- Extended connectivity Fingerprints (ECFP) 의 개요
- ECFP의 구현방법
 - Relation to Morgan Algorithm
 - ECFP Generation Process
 - Initial assignment of atom identifier
 - Iterative updating of identifier
 - Duplicate structure removal
 - Duplicate identifier removal
 - Choice of hash function
 - Circular Fingerprints
 - Application
 - Conclusion

- 역사적으로 topological FP는 분자의 하부구조의 파악과 유사성 조사를 위해서 개발되었다.
- ECFP는 계산이 빠르고 무한에 가까운 다양한 분자의 특징을 표현할 수 있는 방법이다.
- ECFP는 분자의 활동과 관련한 분자의 특징을 알 수 있게 도와준다.
- ECFP는 유사성 조사, 군집화 (clustering), virtual screening 주제에서 좋은 성능을 보인다.
- ECFP는 neural graph fingerprint 이전의 state-of-the-art 성능을 가지고 있는 fingerprint 이다.

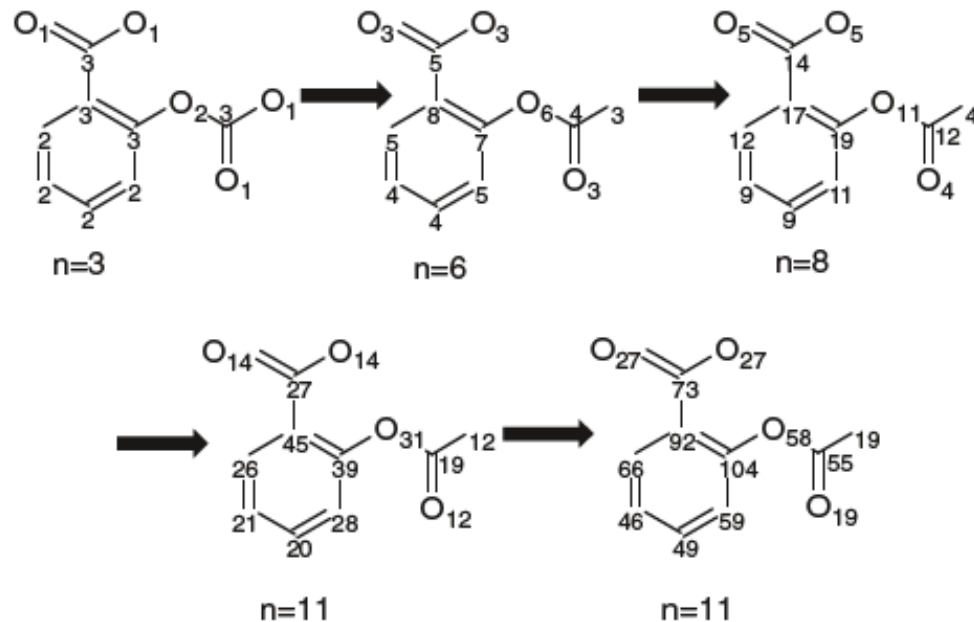
Relation to Morgan Algorithm

- ECFP는 Morgan algorithm [Morgan, 1965]의 변형을 이용하여 개발됨
- Morgan algorithm은 동형이성 (isomorphism) 문제를 해결하기 위해 제안됨
- Morgan algorithm의 구현을 위해서는 iterative process가 요구됨
- Iteration은 different connectivity value (n)가 최대에 도달할 때까지 반복



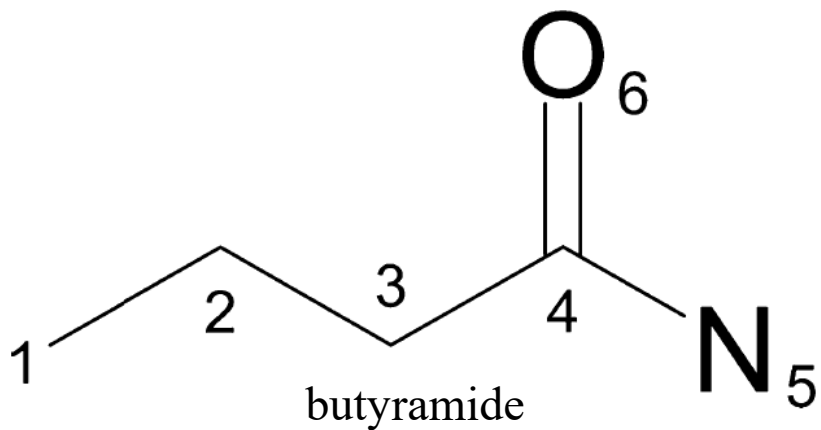
Relation to Morgan Algorithm



- 중간 결과는 모두 삭제되고 사용되지 않음
- Morgan algorithm에 대비되는 ECFP의 개선점
 - ECFP는 중간결과를 삭제하지 않고 fingerprint의 일부로써 사용한다.
 - 완벽하게 정확한 disambiguation이 요구되지 않기 때문에 알고리즘적인 조정이 가능하다.



- 하나의 atom 당 하나의 integer type의 identifier를 할당한다.
- 각각의 atom identifier는 이웃한 atom의 identifier를 반영하여 업데이트를 한다.
 - 업데이트를 할 때 구조적인 중복 (structural duplicate) 이 존재하는지를 고려한다.
- Iteration 중에 같은 identifier가 나타날 경우 삭제한다.
- ECFP Generation Process
 - Initial assignment of atom identifier
 - Iterative updating of identifier
 - Duplicate structure removal
 - Duplicate identifier removal

ECFP Generation Process



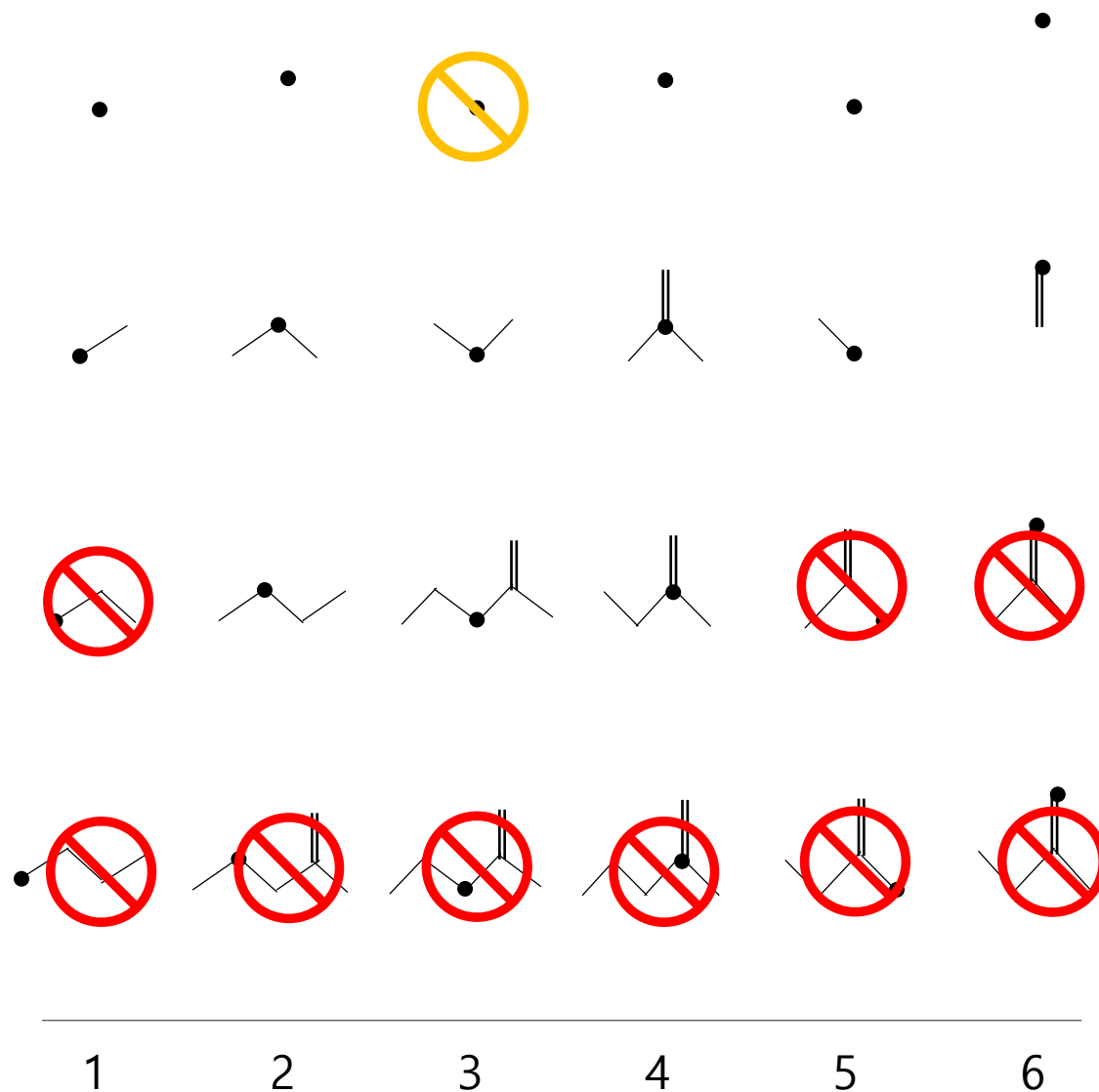
- 이전 iteration의 중복 하부구조 삭제 
- 같은 iteration의 중복 하부구조 삭제 

initial state

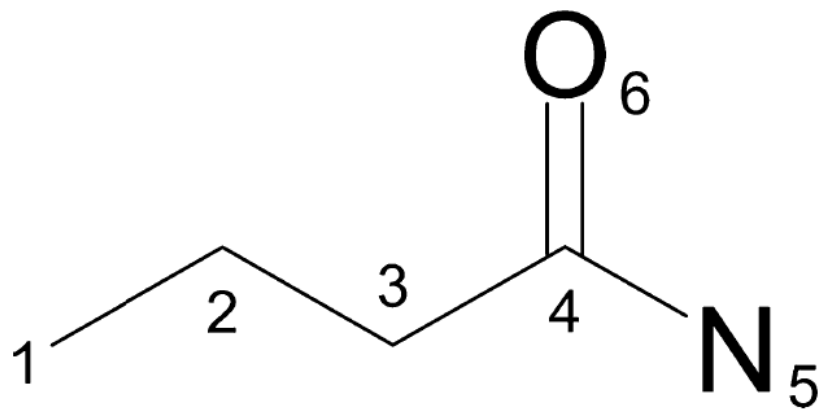
iteration 1

iteration 2

iteration 3



ECFP Generation Process



- identifier는 원래 2^{32} bit 의 크기로 할당
- hash function은 합리적이라면 어떤 것이든 사용가능
 - python, R에 관련 library 多
- 같은 group이 있을 경우, 동일한 identifier 생성가능
 - 삭제 or 표시 후 보관
- ECFP로 직접 바꿔주는 package 및 library 多

initial state

ECFP_0

making array

hashing

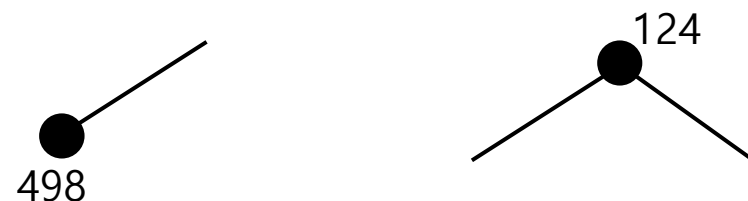
ECFP_1



(237, 679, ...)

[1, 237, 1, 679]

[1, 679, 1, 237, 1, 326]



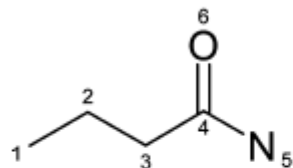
(237, 679, ..., 498, 124, ...)

- iteration이 한번 끝날 때마다 appending
- 중복되는 identifier가 있으면 삭제

1

2

ECFP Generation Process



> <ECFP_0>	> <ECFP_2>	> <ECFP_4>	> <ECFP_6>
734603939	734603939	734603939	734603939
1559650422	1559650422	1559650422	1559650422
-1100000244	-1100000244	-1100000244	-1100000244
1572579716	1572579716	1572579716	1572579716
-1074141656	-1074141656	-1074141656	-1074141656
	863188371	863188371	863188371
	-1793471910	-1793471910	-1793471910
	-1789102870	-1789102870	-1789102870
	-1708545601	-1708545601	-1708545601
	-932108170	-932108170	-932108170
	2099970318	2099970318	2099970318
		-87618679	-87618679
		1112638790	1112638790
		-627599602	-627599602

{실제로 구현된 ECFP 알고리즘}

Circular Fingerprints

Algorithm 1 Circular fingerprints

```

1: Input: molecule, radius  $R$ , fingerprint length  $S$ 
2: Initialize: fingerprint vector  $f \leftarrow 0_S$ 
3: for each atom  $a$  in molecule
4:    $r_a \leftarrow g(a)$   $\triangleright$  lookup atom features
5: for  $L = 1$  to  $R$   $\triangleright$  for each layer
6:   for each atom  $a$  in molecule
7:      $r_1 \dots r_N = \text{neighbors}(a)$ 
8:      $v \leftarrow [r_a, r_1, \dots, r_N]$   $\triangleright$  concatenate
9:      $r_a \leftarrow \text{hash}(v)$   $\triangleright$  hash function
10:     $i \leftarrow \text{mod}(r_a, S)$   $\triangleright$  convert to index
11:     $f_i \leftarrow 1$   $\triangleright$  Write 1 at index
12: Return: binary vector  $f$ 

```

> <ECFP_6>	<idx>
734603939	675
1559650422	118
-1100000244	12
1572579716	418
-1074141656	403
863188371	602
-1793471910	234
-1789102870	447
-1708545601	118
-932108170	270
2099970318	905
-87618679	326
1112638790	782
-627599602	

$\text{mod}(\text{identifier}, 2^{10})$

$[0, \dots, 0] \longrightarrow [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \dots, 0]$

Diagram illustrating the generation of molecular fingerprints from a chemical structure (a five-membered ring with an oxygen atom and a nitrogen atom, and a carbonyl group) using three different diameters (0, 2, and 4) for the neighborhood expansion process.

Diameter 0: Shows the initial structure with atoms labeled X (black), O (red), and N (blue). The fingerprint is generated from the immediate neighbors of the selected atom.

Diameter 2: Shows the structure expanded to include neighbors at a distance of 2. The fingerprint is generated from the neighbors of the neighbors.

Diameter 4: Shows the structure expanded to include neighbors at a distance of 4. The fingerprint is generated from the neighbors of the neighbors of the neighbors.

The resulting fingerprints are listed on the right, corresponding to the three diameters:

- Diameter 0:** -1266712900, -1216914295, 78421366, -887929888, -276894788
- Diameter 2:** -744082560, -798098402, -690148606, 1191819827, 1687725933, 1844215264
- Diameter 4:** -252457408, 132019747, -2036474688, -1979958858, -1104704513

The diagram shows a mapping from identifier list representations to fixed-length binary representations. The identifier list representations are integers, and the fixed-length binary representations are 32-bit strings. Arrows indicate the mapping, with some arrows pointing to the same binary string, illustrating bit collisions.

Identifier list representation:

- 1266712900
- 1216914295
- 78421366
- 887929888
- 276894788
- 744082560
- 798098402
- 690148606
- 1191819827

Fixed-length binary representation:

- 1687725933
- 1844215264
- 252457408
- 132019747
- 2036474688
- 1979958858
- 1104704513

Hash function

Bit collisions



Algorithm 1 Circular fingerprints

```
1: Input: molecule, radius  $R$ , fingerprint length  $S$ 
2: Initialize: fingerprint vector  $\mathbf{f} \leftarrow \mathbf{0}_S$ 
3: for each atom  $a$  in molecule
4:    $\mathbf{r}_a \leftarrow g(a)$   $\triangleright$  lookup atom features
5: for  $L = 1$  to  $R$   $\triangleright$  for each layer
6:   for each atom  $a$  in molecule
7:      $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$ 
8:      $\mathbf{v} \leftarrow [\mathbf{r}_a, \mathbf{r}_1, \dots, \mathbf{r}_N]$   $\triangleright$  concatenate
9:      $\mathbf{r}_a \leftarrow \text{hash}(\mathbf{v})$   $\triangleright$  hash function
10:     $i \leftarrow \text{mod}(\mathbf{r}_a, S)$   $\triangleright$  convert to index
11:     $\mathbf{f}_i \leftarrow 1$   $\triangleright$  Write 1 at index
12: Return: binary vector  $\mathbf{f}$ 
```

Algorithm 2 Neural graph fingerprints

```
1: Input: molecule, radius  $R$ , hidden weights  $H_1^1 \dots H_R^5$ , output weights  $W_1 \dots W_R$ 
2: Initialize: fingerprint vector  $\mathbf{f} \leftarrow \mathbf{0}_S$ 
3: for each atom  $a$  in molecule
4:    $\mathbf{r}_a \leftarrow g(a)$   $\triangleright$  lookup atom features
5: for  $L = 1$  to  $R$   $\triangleright$  for each layer
6:   for each atom  $a$  in molecule
7:      $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$ 
8:      $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^N \mathbf{r}_i$   $\triangleright$  sum
9:      $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$   $\triangleright$  smooth function
10:     $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$   $\triangleright$  sparsify
11:     $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$   $\triangleright$  add to fingerprint
12: Return: real-valued vector  $\mathbf{f}$ 
```
