

Rule-based stochastic Batch-Reactor Simulator

Supervisor

Univ.-Prof. Mag. Dr. Christoph Flamm

Gillespie

- 1. When does my reaction happen?

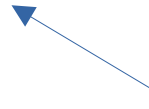
I Calculate the event rates

```
rates = [k1*property_rate1, k2*property_rate2, k3*property_rate3,..]
```

I calculate the sum of my rates.

```
dt = 1 / sum_of_my_rates * p1
```

properties p1..[0..1]



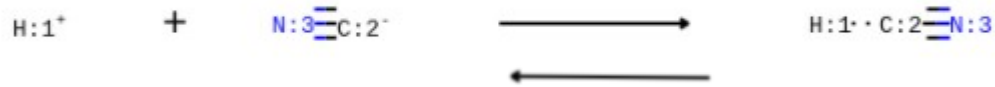
- 2. Which reaction occurs at that time.

My properties have a certain likelihood to appear

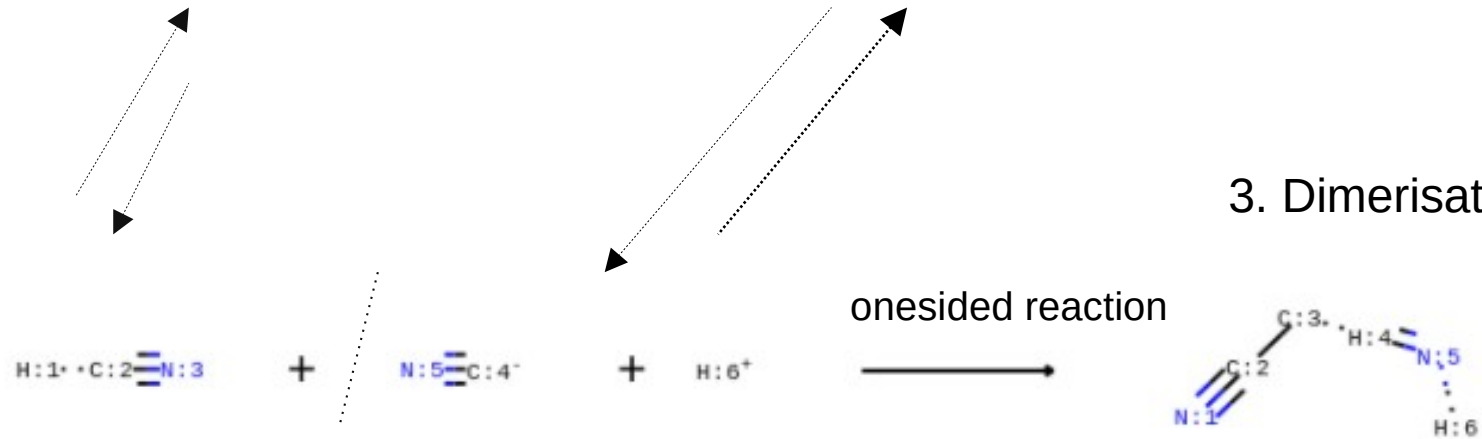
```
if p2 * rate_sum <= rates[0]
```

```
p2..[0..1]
```

1. Assoziation k_a



2. Dissoziation k_d



[H:1][C:1]#[N:1]
 [H:2][C:2]#[N:2]
 [H+:3]
 [C-:3]#[N:3]
 [H+:4]
 [C-:4]#[N:4]
 [H+:5]
 [C-:5]#[N:5]

Flask[0]
 Flask[1]
 Flask[2]
 Flask[3]
 Flask[4]
 Flask[5]
 Flask[6]
 Flask[7]

HCN : [0, 1]
 CN- : [3, 5, 7]
 H+ : [2, 4, 6]

Cartesian pairs:

r1: H-C#N -> H+ + C#N- (k_dis) 2 Possibilities
 r2: H+ + C#N- -> H-C#N (k_ass) 9 Possibilities
 r3: C#N- + H-C#N + H+ -> N#C-C(H)=NH (k_dim) 18 Possibilities

kdiss.R1: [0, 1]

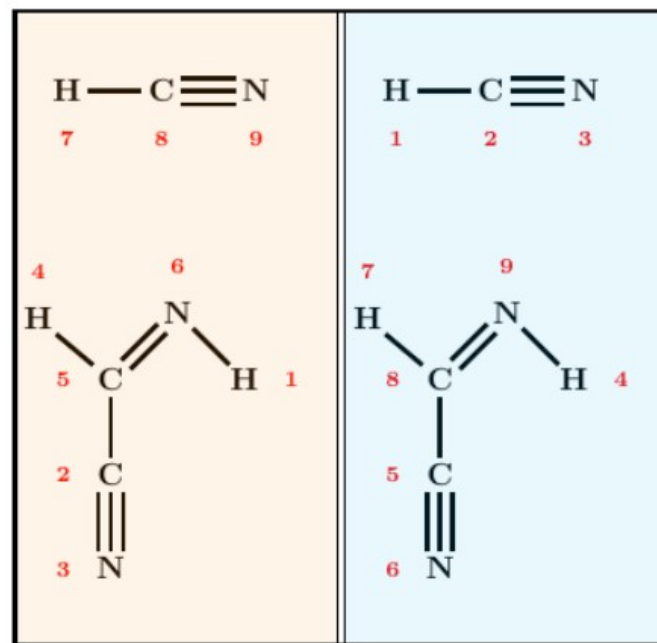
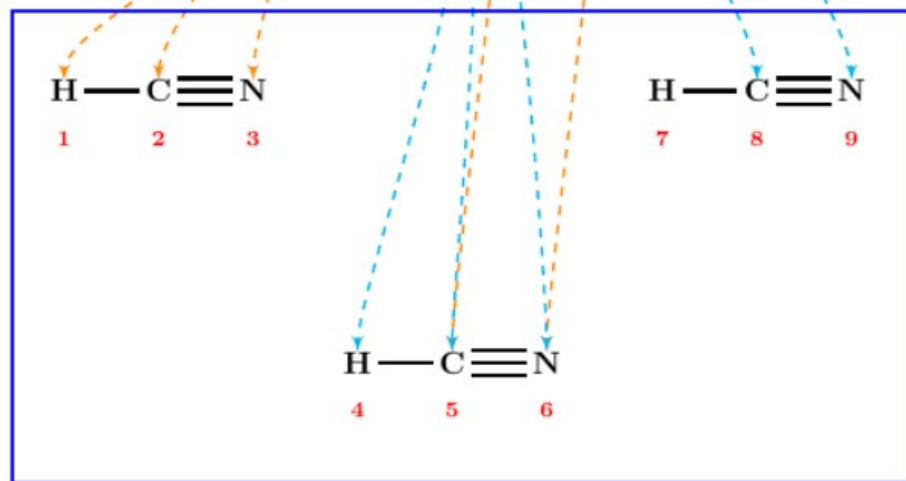
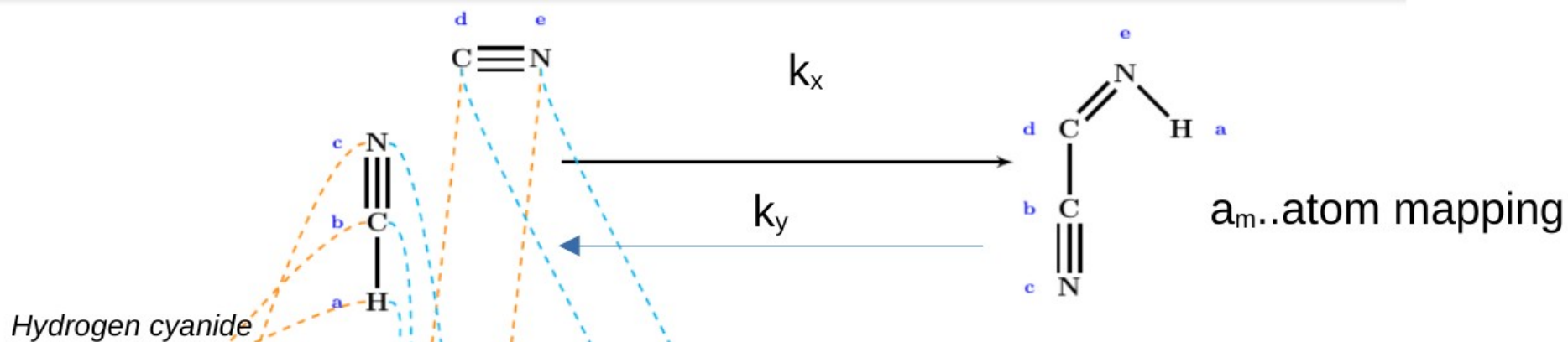
Kass.:R2: [(3, 2), (3, 4), (3, 6), (5, 2), (5, 4), (5, 6), (7, 2), (7, 4), (7, 6)]

Kdim.:R3: [(3, 0, 2), (3, 0, 4), (3, 0, 6), (3, 1, 2), (3, 1, 4), (3, 1, 6), (5, 0, 2), (5, 0, 4), (5, 0, 6), (5, 1, 2), (5, 1, 4), (5, 1, 6), (7, 0, 2), (7, 0, 4), (7, 0, 6), (7, 1, 2), (7, 1, 4), (7, 1, 6)]

$$\text{Rates} = [2 * k_{\text{diss}}, 9 * k_{\text{ass}}, 18 * k_{\text{dim}}]$$

1. Time: `tau = np.random.exponential(scale=1/sumrates)`
2. I choose my reaction with selecting one of my cartesian pairs with my weightet probabilities.

So I assure that I select with my reaction my molecules simultaneously!



SMILES and SMARTS as graph

```
from pysmiles import read_smiles
```

```
def convert(smiles):
```

```
    mol = read_smiles(smiles)
```

```
    mol_with_H = read_smiles(smiles, explicit_hydrogen=True)
```

```
    tuple = mol.nodes(data='element')
```

```
    tuple_h = mol_with_H.nodes(data='element')
```

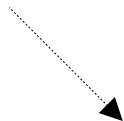
```
    tuple_h_to_dict = dict(tuple_h)
```

```
    return tuple, tuple_h, tuple_h_to_dict
```

```
mid2smi = {"0": "C#N", "1": "C#N", "3": "C#N", "4": "[C-]#N", "5": "[C-]#N",
```

```
each_atom_has_his_index_tupels: [(0, 'C'), (1, 'N'), (2, 'H')], [(3, 'C'), (4, 'N'), (5, 'H')], [(6, 'C'), (7, 'N'), (8, 'H')], [(9, 'C'), (10, 'N'), (11, 'H')], [(12, 'C'), (13, 'N')],
```

```
select_dict: {'0': ('C#N', [(0, 'C'), (1, 'N'), (2, 'H')]), '1': ('C#N', [(3, 'C'), (4, 'N'), (5, 'H')]), '2': ('C#N', [(6, 'C'), (7, 'N'), (8, 'H')]), '3': ('C#N', [(9, 'C'), (10, 'N'), (11, 'H')]),
```



```
templatea_dict = {'3': ('C#N', [(9, 'C'), (10, 'N'), (11, 'H')]), '7': ('[C-]#N', [(18, 'C'), (19, 'N')]), '13': ('C(C=N)
```

Class for Rearranging my molecules

```
new_Tupel = Tuple([(30, 'H'), (1, 'N'), (3, 'C'), (4, 'H'), (17, 'C')],  
                  [(12, 'H'), (2, 'C'), (3, 'H'), (9, 'C'), (40, 'N'), (11, 'C')])
```

Transformer Classes for generating my reactions:

```
form = C(select_dict, 'C#N', "[H+]", "[C-]#N", template_dict)
```

```
transformer = form.transformer()
```



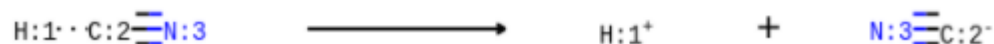
Gillespie

In RdKit

- 1. Templat

```
# define reaction(s) HC#N <=> H+ + -C#N
dis = AllChem.ReactionFromSmarts('[H:1][C:2]#[N:3]>>[H+:1].[C-:2]#[N:3]')
ps = dis.RunReactants([flask[1]])
ps
print("applied reaction dis to molecule 2 flask")
dis
```

applied reaction dis to molecule 2 flask

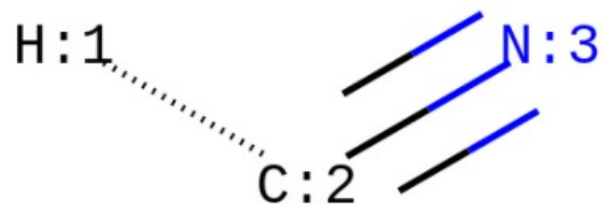


- 2. read my patterns

```
patt0 = '[H:1][C:2]#[N:3]'
```

```
e0_patt = Chem.MolFromSmarts(patt0)
```

```
e0_patt
```



Atom mapping:

```
patt0 = '[H:1][C:2]#[N:3]'
educt = flask[1]
e0_patt = Chem.MolFromSmarts(patt0)

# construct embedding of educt SMART into educt
e0_matches = educt.GetSubstructMatches(e0_patt)
xxxx = [x.GetAtomMapNum() for x in e0_patt.GetAtoms()]
yyyy = [x.GetAtomMapNum() for x in educt.GetAtoms()]
eaidss = dict(zip(xxxx, yyyy))
print(xxxx)
print(yyyy)
print(eaidss)
```

```
[1, 2, 3]
[3, 3, 3]
{1: 3, 2: 3, 3: 3}
```

```
p1, p2 = fixAtomNumRealto2(flask[2], '[H:1][C:2]#[N:3]',
                             ps[0][0], '[H+:1]',
                             ps[0][1], '[C-:2]#[N:3]')
```

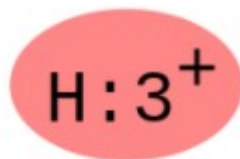
Assign reaction to my new products

```
def setAtomNums(mol, pattern, embedding):
    matches = mol.GetSubstructMatches(pattern)
    aids = [ x.GetAtomMapNum() for x in pattern.GetAtoms()]
    aidss = list(zip(matches[0], aids))
    # print(aidss)

    # iterate over the atoms in the product molecule
    for (x,y) in aidss:
        # print(x,',',y)
        a = mol.GetAtomWithIdx(x)
        # set AtomMapNum from educt as AtomMatchNum for product
        a.SetAtomMapNum(embedding[y])

    return mol
```

```
flask before update
[H:1][C:1]#[N:1]
[H:2][C:2]#[N:2]
[H:3][C:3]#[N:3]
flask after update
[H:1][C:1]#[N:1]
[H:3][C:3]#[N:3]
[H+:3]
[C-:3]#[N:3]
```

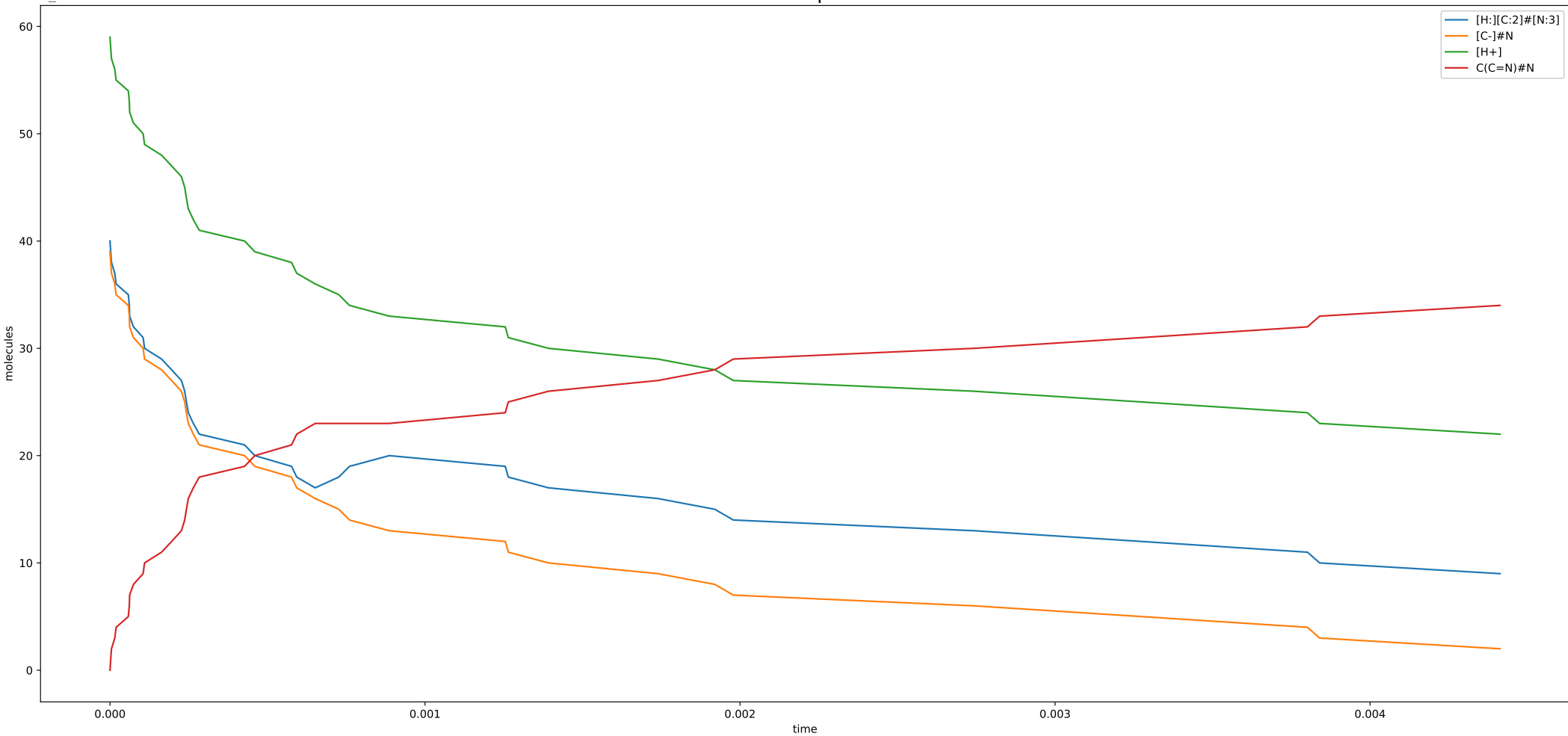


```

0
cartesian: [(0, 23), (0, 24), (0, 25), (1, 23), (1, 24), (1, 25), (2, 23), (2, 24), (2, 25), (3, 23), (3, 24), (3, 25), (4, 23), (4, 24), (4, 25), (5, 23), (5, 24), (5, 25), (6, 23), (6, 24), (6, 25), (7, 23), (7, 24), (7, 25), (8, 23), (8, 24), (8, 25), (9, 23), (9, 24), (9, 25), (10, 23), (10, 24), (10, 25), (11, 23), (11, 24), (11, 25), (12, 23), (12, 24), (12, 25), (13, 23), (13, 24), (13, 25), (14, 23), (14, 24), (14, 25), (15, 23), (15, 24), (15, 25), (16, 23), (16, 24), (16, 25), (17, 23), (17, 24), (17, 25), (18, 23), (18, 24), (18, 25), (19, 23), (19, 24), (19, 25), (20, 23), (20, 24), (20, 25), (21, 23), (21, 24), (21, 25), (22, 23), (22, 24), (22, 25), (23, 23), (23, 24), (23, 25), (24, 23), (24, 24), (24, 25), (25, 23), (25, 24), (25, 25), (26, 23), (26, 24), (26, 25), (27, 23), (27, 24), (27, 25), (28, 23), (28, 24), (28, 25), (29, 23), (29, 24), (29, 25), (30, 23), (30, 24), (30, 25), (31, 23), (31, 24), (31, 25), (32, 23), (32, 24), (32, 25), (33, 23), (33, 24), (33, 25), (34, 23), (34, 24), (34, 25), (35, 23), (35, 24), (35, 25), (36, 23), (36, 24), (36, 25), (37, 23), (37, 24), (37, 25), (38, 23), (38, 24), (38, 25), (39, 23), (39, 24), (39, 25), (40, 23), (40, 24), (40, 25), (41, 23), (41, 24), (41, 25), (42, 23), (42, 24), (42, 25), (43, 23), (43, 24), (43, 25), (44, 23), (44, 24), (44, 25), (45, 23), (45, 24), (45, 25), (46, 23), (46, 24), (46, 25), (47, 23), (47, 24), (47, 25), (48, 23), (48, 24), (48, 25), (49, 23), (49, 24), (49, 25), (50, 23), (50, 24), (50, 25), (51, 23), (51, 24), (51, 25), (52, 23), (52, 24), (52, 25), (53, 23), (53, 24), (53, 25), (54, 23), (54, 24), (54, 25), (55, 23), (55, 24), (55, 25), (56, 23), (56, 24), (56, 25), (57, 23), (57, 24), (57, 25), (58, 23), (58, 24), (58, 25), (59, 23), (59, 24), (59, 25), (60, 23), (60, 24), (60, 25), (61, 23), (61, 24), (61, 25), (62, 23), (62, 24), (62, 25), (63, 23), (63, 24), (63, 25), (64, 23), (64, 24), (64, 25), (65, 23), (65, 24), (65, 25), (66, 23), (66, 24), (66, 25), (67, 23), (67, 24), (67, 25), (68, 23), (68, 24), (68, 25), (69, 23), (69, 24), (69, 25), (70, 23), (70, 24), (70, 25), (71, 23), (71, 24), (71, 25), (72, 23), (72, 24), (72, 25), (73, 23), (73, 24), (73, 25), (74, 23), (74, 24), (74, 25), (75, 23), (75, 24), (75, 25), (76, 23), (76, 24), (76, 25), (77, 23), (77, 24), (77, 25), (78, 23), (78, 24), (78, 25), (79, 23), (79, 24), (79, 25), (80, 23), (80, 24), (80, 25), (81, 23), (81, 24), (81, 25), (82, 23), (82, 24), (82, 25), (83, 23), (83, 24), (83, 25), (84, 23), (84, 24), (84, 25), (85, 23), (85, 24), (85, 25), (86, 23), (86, 24), (86, 25), (87, 23), (87, 24), (87, 25), (88, 23), (88, 24), (88, 25), (89, 23), (89, 24), (89, 25), (90, 23), (90, 24), (90, 25), (91, 23), (91, 24), (91, 25), (92, 23), (92, 24), (92, 25), (93, 23), (93, 24), (93, 25), (94, 23), (94, 24), (94, 25), (95, 23), (95, 24), (95, 25), (96, 23), (96, 24), (96, 25), (97, 23), (97, 24), (97, 25), (98, 23), (98, 24), (98, 25), (99, 23), (99, 24), (99, 25), (100, 23), (100, 24), (100, 25), (101, 23), (101, 24), (101, 25), (102, 23), (102, 24), (102, 25), (103, 23), (103, 24), (103, 25), (104, 23), (104, 24), (104, 25), (105, 23), (105, 24), (105, 25), (106, 23), (106, 24), (106, 25), (107, 23), (107, 24), (107, 25), (108, 23), (108, 24), (108, 25), (109, 23), (109, 24), (109, 25), (110, 23), (110, 24), (110, 25), (111, 23), (111, 24), (111, 25), (112, 23), (112, 24), (112, 25), (113, 23), (113, 24), (113, 25), (114, 23), (114, 24), (114, 25), (115, 23), (115, 24), (115, 25), (116, 23), (116, 24), (116, 25), (117, 23), (117, 24), (117, 25), (118, 23), (118, 24), (118, 25), (119, 23), (119, 24), (119, 25), (120, 23), (120, 24), (120, 25), (121, 23), (121, 24), (121, 25), (122, 23), (122, 24), (122, 25), (123, 23), (123, 24), (123, 25), (124, 23), (124, 24), (124, 25), (125, 23), (125, 24), (125, 25), (126, 23), (126, 24), (126, 25), (127, 23), (127, 24), (127, 25), (128, 23), (128, 24), (128, 25), (129, 23), (129, 24), (129, 25), (130, 23), (130, 24), (130, 25), (131, 23), (131, 24), (131, 25), (132, 23), (132, 24), (132, 25), (133, 23), (133, 24), (133, 25), (134, 23), (134, 24), (134, 25), (135, 23), (135, 24), (135, 25), (136, 23), (136, 24), (136, 25), (137, 23), (137, 24), (137, 25), (138, 23), (138, 24), (138, 25), (139, 23), (139, 24), (139, 25), (140, 23), (140, 24), (140, 25), (141, 23), (141, 24), (141, 25), (142, 23), (142, 24), (142, 25), (143, 23), (143, 24), (143, 25), (144, 23), (144, 24), (144, 25), (145, 23), (145, 24), (145, 25), (146, 23), (146, 24), (146, 25), (147, 23), (147, 24), (147, 25), (148, 23), (148, 24), (148, 25), (149, 23), (149, 24), (149, 25), (150, 23), (150, 24), (150, 25), (151, 23), (151, 24), (151, 25), (152, 23), (152, 24), (152, 25), (153, 23), (153, 24), (153, 25), (154, 23), (154, 24), (154, 25), (155, 23), (155, 24), (155, 25), (156, 23), (156, 24), (156, 25), (157, 23), (157, 24), (157, 25), (158, 23), (158, 24), (158, 25), (159, 23), (159, 24), (159, 25), (160, 23), (160, 24), (160, 25), (161, 23), (161, 24), (161, 25), (162, 23), (162, 24), (162, 25), (163, 23), (163, 24
```

k_diss: 1
k_ass: 1
k_dim: 1

Gillespie



k_diss: 1.8
k_ass: 1
k_dim: 0.8

Gillespie

