Untitled

August 23, 2022

[1]:
```
from IPython import display
from collections import Counter
from tabulate import tabulate
from tqdm.auto import tqdm
import gzip
import pickle
import time

import sympy as sp
import random

import perceval as pcvl
import perceval.lib.symb as symb
n = 14        #number of photons at the input
m = 50       #number of modes
N = 50000  #number of samplings
Unitary_60 = pcvl.Matrix.random_unitary(m) #creates a random unitary of␣
 ↪dimension 60
mzi = (symb.BS() // (0, symb.PS(phi=pcvl.Parameter("_a")))
       // symb.BS() // (1, symb.PS(phi=pcvl.Parameter("_b"))))
pcvl.pdisplay(mzi)
Linear_Circuit_60 = pcvl.Circuit.decomposition(Unitary_60, mzi,
                                               phase_shifter_fn=symb.PS,
                                               shape="triangle")
```

```
<IPython.core.display.HTML object>
```

[2]:
```
Sampling_Backend = pcvl.BackendFactory().get_backend("CliffordClifford2017")
```

[3]:
```
#one can choose which mode he/she wants at input, or we can choose it randomly
def Generating_Input(n, m, modes = None):
    "This function randomly chooses an input with n photons in m modes."
    if modes == None :
        modes = sorted(random.sample(range(m),n))
    state = "|"
    for i in range(m):
        state = state + "0"*(1 - (i in modes)) +"1"*(i in modes)+ ","*(i < m-1)
```

```
    return pcvl.BasicState(state + ">")

input_state = Generating_Input(n, m)
print("The input state: ", input_state)
```

```
The input state:  |1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,1,0,0,1,0,1,1,1,0,0,1,0
,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1>
```

[4]:
```
s1 = input_state
print(s1)
```

```
|1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,1,0,0,1,0,1,1,1,0,0,1,0,0,0,1,0,0,0,0,1,1
,0,0,0,0,0,0,0,0,0,0,1>
```

[5]:
```
input_state = Generating_Input(n, m)
#print("The input state: ", input_state)
s2 = input_state
print(s2)
```

```
|0,0,0,1,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,0,0,0,1,0,1
,0,1,0,1,1,0,1,0,0,0>
```

[6]:
```
coding = []
for i in range(1, 20):
    if 1:
        coding.append('0')
    elif 2:
        coding.append('1')
    else: coding.append('*')
    #else coding[i] = '*'
```

[7]:
```
coding = []
for i in range(0, 0):
    if s1[i] == 1 and s2[i] == 0:
        coding.append('0')
    elif s1[i] == 0 and s2[i] == 1:
        coding.append('1')
    else: coding.append('*')
    #else coding[i] = '*'
print("The first sampling result S1" , s1)
print("The second sampling results S2" , s2)
print("Coding" , coding)

r = ''
for i in coding:
    if i != '*':
       # r.append(i)
```

```
        r = r + i
print("The final random number sequence:", r)
```

```
The first sampling result S1 |1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,1,0,0,1,0,1,
1,1,0,0,1,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1>
The second sampling results S2 |0,0,0,1,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,1,1,1,0,0,1,0,0,0,1,0,1,0,1,0,1,1,0,1,0,0,0>
Coding ['0', '*', '*', '1', '*', '*', '*', '*', '*', '*', '*', '1', '0', '1',
'0', '*', '*', '*', '*', '0']
The final random number sequence: 0110100
```

[8]:
```
#print("The sampled outputs are:")
#S1 =  pcvl.BasicState("|>")
#for _ in range(10):
   # print(Sampling_Backend(Unitary_60).sample(input_state))

 #print(S1)
```

[ ]:
```
```

[9]:
```
# if we want to launch parallel process
worker_id=1

#store the input and the unitary
with open("%dphotons_%dmodes_%dsamples-worker%s-unitary.pkl"␣
 ↪%(n,m,N,worker_id), 'wb') as f:
    pickle.dump(Unitary_60, f)

with open("%dphotons_%dmodes_%dsamples-worker%s-inputstate.pkl"␣
 ↪%(n,m,N,worker_id), 'w') as f:
    f.write(str(input_state)+"\n")


with gzip.open("%dphotons_%dmodes_%dsamples-worker%s-samples.txt.gz"␣
 ↪%(n,m,N,worker_id), 'wb') as f:
    start = time.time()
    for i in range(N):
        f.write((str(Sampling_Backend(Unitary_60).sample(pcvl.
 ↪BasicState(input_state)))+"\n").encode());
    end = time.time()
    f.write(str("==> %d\n" % (end-start)).encode())
f.close()
```

[10]:
```
import gzip
```

[11]:
```
worker_id = 1
count = 0
bunching_distribution = Counter()

with gzip.open("%dphotons_%dmodes_%dsamples-worker%s-samples.txt.
 ↪gz"%(n,m,N,worker_id), "rt") as f:
    for l in f:
        l = l.strip()
        if l.startswith("|") and l.endswith(">"):
            try:
                st = pcvl.BasicState(l)
                count+=1
                bunching_distribution[st.photon2mode(st.n-1)]+=1
            except Exception:
                pass
print(count, "samples")
print("Bunching Distribution:", "\t".join([str(bunching_distribution[k]) for k␣
 ↪in range(m)]))
```

```
50000 samples
Bunching Distribution: 0        0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       1       0       4       7
4       9       31      24      49      88      93      142     269     310
294     530     740     1211    1149    1802    2450    3664    4837    4973
6852    9234    11233
```

[ ]: