bit_flipping_sp

May 26, 2025

[1]:
```
import numpy as np
import random
import math
def process_check_node(mu):
    v_pass=dict()
    for i in range(6):
        filtered_mu = {key: val for key, val in mu.items() if key[0] == i}
        for j,v in filtered_mu.items():
            index_mu=filtered_mu.copy()
            index_mu.pop(j)

            v_pass[j] = 0
            for _,mu_values in index_mu.items():
                v_pass[j] =  mu_values + v_pass[j]
            v_pass[j] = v_pass[j]% 2
    return v_pass

def process_judge(v_factor,v_pass,H):
    x_pred=np.zeros(6)
    for j in range(6):
        filtered_vpass = {key: val for key, val in v_pass.items() if key[1] ==␣
 ↪j}.copy()
        num_zero=0
        num_one=0
        for k,v in filtered_vpass.items():
            if v == 0:
                num_zero += 1
            elif v == 1:
                num_one += 1
        if num_zero >= 2:
            x_pred[j] = 0
        elif num_one >= 2:
            x_pred[j] = 1
        else:
            x_pred[j]=v_factor[j]
    print(f"x_pred={x_pred}")
    if np.all(np.dot(H,x_pred) == np.zeros(6)):
        print("x_pred is correct")
    else:
        print("x_pred is wrong")
        # x_pred=(0<kai).astype(int)

def process_variable_node(v_factor,v_pass):
    mu=dict()
    for j in range(6):
        filtered_vpass = {key: val for key, val in v_pass.items() if key[1] ==␣
 ↪j}.copy()
        # print(f"filtered_vpass{filtered_vpass}")
        for i,v in filtered_vpass.items():
            dot_target_dic=filtered_vpass.copy()
            dot_target_dic.pop(i)

            # print(f"dict={dot_target_dic}")

            num_zero=0
            num_one=0
            for k,v in dot_target_dic.items():
                if v == 0:
                    num_zero += 1
                elif v == 1:
                    num_one += 1

            if num_zero >= 1:
                mu[i] =0
            elif num_one >= 1:
                mu[i] =1
            else:
                mu[i] = v_factor[j]
            # print(f"mu[{i}]={mu[i]}")
    return mu
```

[2]:
```
def bitflipping_decoder(v_factor):
    def print_msg(msg):
        for i in range(6):
            for j in range(6):
                if (i,j) in msg:
                    print(f"msg[{i},{j}]={msg[(i,j)]}",end=" ")
            print("")

    H=np.array(
        [
            [1,1,0,0,0,0],
            [0,1,1,0,0,0],
            [0,0,1,1,0,0],
            [0,0,0,1,1,0],
            [0,0,0,0,1,1],
            [1,0,0,0,0,1]
        ]
    )

    mu=dict()
    for i in range(H.shape[0]):
        for j in range(H.shape[1]):
            if H[i,j] == 1:
                mu[(i,j)] = int(v_factor[j])
    print_msg(mu)

    mu=mu.copy()
    for t in range(2):
        print(f"t={t}")

        v_pass=process_check_node(mu)
        print("v_pass")
        print_msg(v_pass)

        process_judge(v_factor,v_pass,H)

        mu=process_variable_node(v_factor,v_pass)
        print("mu")
        print_msg(mu)

init_v=np.array([1,1,0,0,0,0])
print(init_v)
bitflipping_decoder(init_v)
```

```
[1 1 0 0 0 0]
msg[0,0]=1 msg[0,1]=1
msg[1,1]=1 msg[1,2]=0
msg[2,2]=0 msg[2,3]=0
msg[3,3]=0 msg[3,4]=0
msg[4,4]=0 msg[4,5]=0
msg[5,0]=1 msg[5,5]=0
t=0
v_pass
msg[0,0]=1 msg[0,1]=1
msg[1,1]=0 msg[1,2]=1
msg[2,2]=0 msg[2,3]=0
msg[3,3]=0 msg[3,4]=0
msg[4,4]=0 msg[4,5]=0
msg[5,0]=0 msg[5,5]=1
x_pred=[1. 1. 0. 0. 0. 0.]
```

```
x_pred is wrong
mu
msg[0,0]=0 msg[0,1]=0
msg[1,1]=1 msg[1,2]=0
msg[2,2]=1 msg[2,3]=0
msg[3,3]=0 msg[3,4]=0
msg[4,4]=0 msg[4,5]=1
msg[5,0]=1 msg[5,5]=0
t=1
v_pass
msg[0,0]=0 msg[0,1]=0
msg[1,1]=0 msg[1,2]=1
msg[2,2]=0 msg[2,3]=1
msg[3,3]=0 msg[3,4]=0
msg[4,4]=1 msg[4,5]=0
msg[5,0]=0 msg[5,5]=1
x_pred=[0. 0. 0. 0. 0. 0.]
x_pred is correct
mu
msg[0,0]=0 msg[0,1]=0
msg[1,1]=0 msg[1,2]=0
msg[2,2]=1 msg[2,3]=0
msg[3,3]=1 msg[3,4]=1
msg[4,4]=0 msg[4,5]=1
msg[5,0]=0 msg[5,5]=0
```