```python
import numpy as np

# start the values off at an extreme
x1, x2, x3, x4 = 0, 0, 0, 0
z1, z2, z3, z4 = 0, 0, 0, 0

# number of epochs
N=100

curr_sol = None

def objective(x1, x2, x3, x4):
    return 12*x1 + 16*x2 + 22*x3 + 8*x4


def constraint(x1, x2, x3, x4):
    return 4*x1 + 5*x2 + 7*x3 + 3*x4


def evaluate(curr_sol, ys, zs):
    y1, y2, y3, y4 = ys
    z1, z2, z3, z4 = zs
    candidate_sol = objective(y1, y2, y3, y4)

    if constraint(y1, y2, y3, y4) <= 14:
        if curr_sol is None: # if no valid solution has yet been found
            return candidate_sol, y1, y2, y3, y4

        diff = candidate_sol - curr_sol
        print("difference between candidate and current best", diff)
        if diff > 0:
            print("accept new")
            return candidate_sol, y1, y2, y3, y4
        else:
            print("not more optimal. skip")
            return curr_sol, z1, z2, z3, z4

    print('constraint violated. skip')
    return curr_sol, z1, z2, z3, z4


for i in range(N):
    # randomly reassign 1 value
    exec("%s = %d" % (np.random.choice(['x1', 'x2', 'x3', 'x4']),
    np.random.choice([0,1])))

    curr_sol, z1, z2, z3, z4 = evaluate(curr_sol, (x1, x2, x3, x4), (z1, z2, z3, z4))

print("final:\n\tx1: {}\n\t x2: {}\n\t x3: {}\n\t x4: {}\n\tobjective: {}".format(z1,
z2, z3, z4, curr_sol))
```