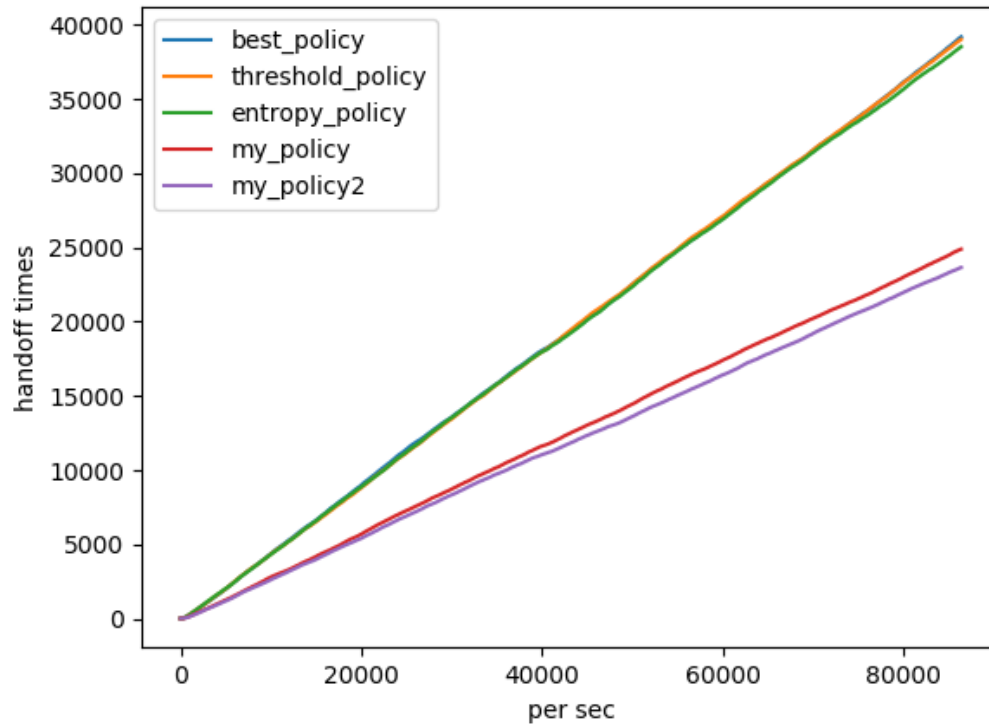


1.圖表



2.source code

```
def best_policy(car):
    # hand off ...
    old = car.hold
    pold = car.power
    pnew = pold
    new = old
    for a in range(0, 4): # find p max
        bs = bs1 if a == 0 else bs2 if a == 1 else bs3 if a == 2 else bs4
        if pnew < bs[car.y][car.x]:
            pnew = bs[car.y][car.x]
            new = a + 1
    if pnew > pold or pold < pmin: # old not the max
        car.change(new)
        return True
    else:
        return False
pass
```

```

def threshold_policy(car):
    # hand off ...
    old = car.hold
    pold = car.power
    pnew = pold
    new = old
    for a in range(0, 4): # find p max
        bs = bs1 if a == 0 else bs2 if a == 1 else bs3 if a == 2 else bs4
        if pnew < bs[car.y][car.x]:
            pnew = bs[car.y][car.x]
            new = a + 1
    if (pnew > pold and pold < threshold) or pold < pmin: # old not the max and <
threshold
        car.change(new)
        return True
    else:
        return False
    pass

def entropy_policy(car):
    # hand off ...
    old = car.hold
    pold = car.power
    pnew = pold
    new = old
    for a in range(0, 4): # find p max
        bs = bs1 if a == 0 else bs2 if a == 1 else bs3 if a == 2 else bs4
        if pnew < bs[car.y][car.x]:
            pnew = bs[car.y][car.x]
            new = a + 1
    if pold + entro < pnew or pold < pmin: # old not the max and diff entropy
        car.change(new)
        return True
    else:
        return False
    pass
    pass

def my_policy(car):
    # hand off ...

```

```

old = car.hold
pold = car.power
pnew = pold
new = old
for a in range(0, 4): # find p max
    bs = bs1 if a == 0 else bs2 if a == 1 else bs3 if a == 2 else bs4
    if pnew < bs[car.y][car.x]:
        pnew = bs[car.y][car.x]
        new = a + 1
    if (pnew > pold and car.duration > 75*3) or pold < pmin: # old not the max and
last 75*3 sec
        car.change(new)
        return True
elif pold < pnew: # time for pold not the max
    car.elapse()
    return False
else:
    return False
pass
pass
def my_policy2(car):
    # hand off ...
    old = car.hold
    pold = car.power
    pnew = pold
    new = old
    for a in range(0, 4): # find p max
        bs = bs1 if a == 0 else bs2 if a == 1 else bs3 if a == 2 else bs4
        if pnew < bs[car.y][car.x]:
            pnew = bs[car.y][car.x]
            new = a + 1
    if pold < pmin: # until pmin
        car.change(new)
        return True
    else:
        return False
pass
pass

```

3.introduction to your policy

best_policy average power:-114.72121321773818

threshold_policy average power:-114.6940126347598

entropy_policy average power:-114.91339595752127

my_policy average power:-117.05076337331167

my_policy2 average power:-117.1380324903485

mypolicy:當車子累積超過 75*3 秒原本負責 BS 功率不是最大再 handoff，可減少部分剛好繞出一圈(走三段)又回到原本 1/4 區域的 handoff，比 best/threshold policy handoff 較少次但 average power 比較低，彈性沒 entropy policy 高(由於參數問題在此例 entropy policy 有較多次 handoff)。

mypolicy2:直到功率要小於 pmin 時才 handoff，handoff 必最少次，但 average power 最低。