r1 = (r1 - 500)/500.0

r1[r1 == -1.0] = -20

# print(r1)

r1 = np.sum(r1)

# print('r1',r1)

r2 = (1-(energy\_cost / energy\_cost\_max))\*20.0

# print('r2',r2)

reward = r1+r2

收敛之后基站趋于全部关闭。可能是因为r2太小了。 2017 6-21 9:50

r1 = (r1 - 500)/500.0

r1[r1 == -1.0] = -20

# print(r1)

r1 = np.sum(r1)

# print('r1',r1)

r2 = (1-(energy\_cost / energy\_cost\_max))\*40.0

# print('r2',r2)

reward = r1+r2

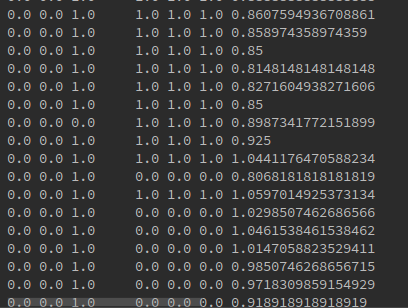
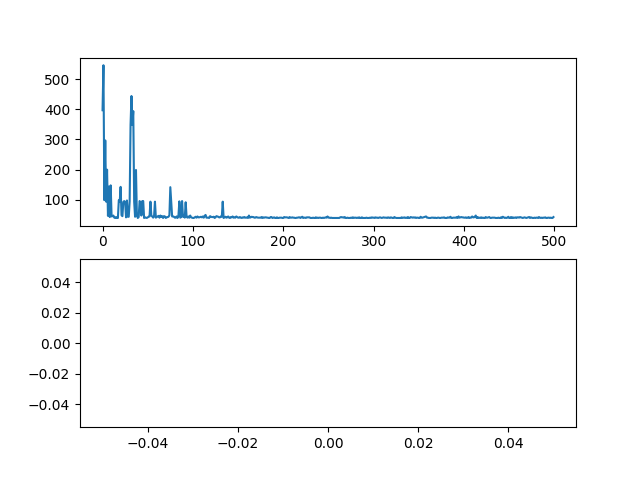
效果比之前好很多 2017 6-21 11:02

问题：无法衡量EE的好坏，没有对比方案

对比方案：

有电的时候开，没电的时候关

加入了对比方案，效果如图所示。



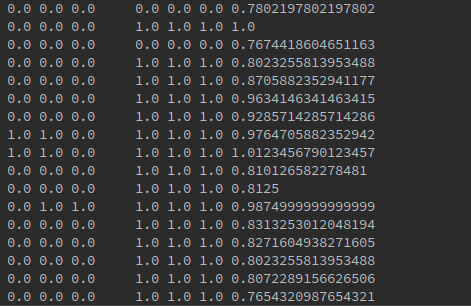
基本可以保证能效比对比方案有提升

问题：

可能是由于能效的reward占比较大，导致收敛的结果是以EE为主，每次迭代只有40步，显然这种方式是不可行的。

可能的解决方案：

删除R1，增加系统break的惩罚。



可以不断迭代 但可以明显看出基站趋于关闭状态更多。

可能的解决方案 降低break的惩罚 调整环境参数（可能是基站接入的人太少了）

解决方案：增加了系统用户到达率，降低了break带来的惩罚，并同时使用三个参数比较不同效果

参数1：

r1 = (r1 - 500)/500.0

r1[r1.iloc[:,0]!=-1.0]=0

r1[r1.iloc[:,0]==-1.0]=-50

# print(r1)

r1 = np.sum(r1)[0]

# print('r1',r1)

r2 = (1-(energy\_cost / energy\_cost\_naive))\*30.0

# print('r2',r2)

reward = r1+r2

参数2

r1 = (r1 - 500)/500.0

r1[r1.iloc[:,0]!=-1.0]=0

r1[r1.iloc[:,0]==-1.0]=-50

# print(r1)

r1 = np.sum(r1)[0]

# print('r1',r1)

r2 = (1-(energy\_cost / energy\_cost\_naive))\*40.0

# print('r2',r2)

reward = r1+r2

参数3

r1 = (r1 - 500)/500.0

r1[r1.iloc[:,0]!=-1.0]=0

r1[r1.iloc[:,0]==-1.0]=-50

# print(r1)

r1 = np.sum(r1)[0]

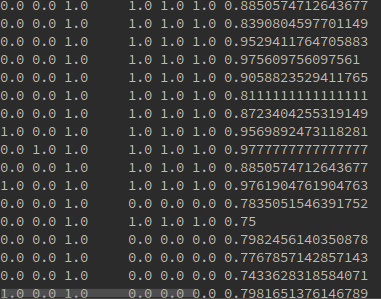
# print('r1',r1)

r2 = (1-(energy\_cost / energy\_cost\_naive))\*50.0

# print('r2',r2)

reward = r1+r2

该三个参数energy\_cost的比重不同，但最后的效果都很差，三个参数收敛之后每一次的迭代次数均为40次左右。



可以看出，三组不同的参数对于能效的提升还是很明显的。所以进一步的实验方向应该是想办法让系统多运行一段时间。

可能的解决方案：

修改r1

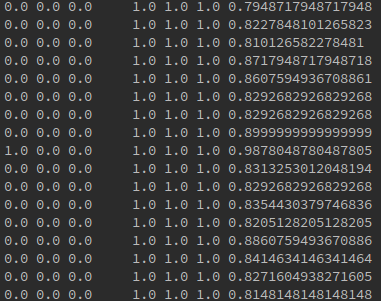
r1+=1

增加单步奖励，rl可能会想办法多走一些步。

结果：

可能r2还是太大了，收敛结果是41步

将用户到达率重新修改为0.2，r2占的比重为30



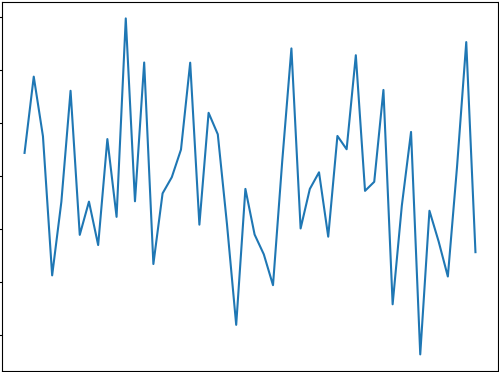
可以看出，动作的选择又趋于所有的基站全关了，这没什么意义。

三个程序同时跑

run 到达率增加到0.3 r2的倍数增加为40

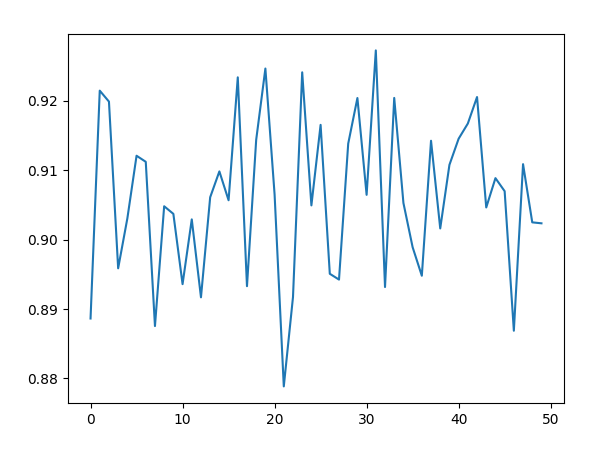
run1 到达率不变 r2 的倍数增加为40

run2 到达率增加到0.3 r2的倍数不变

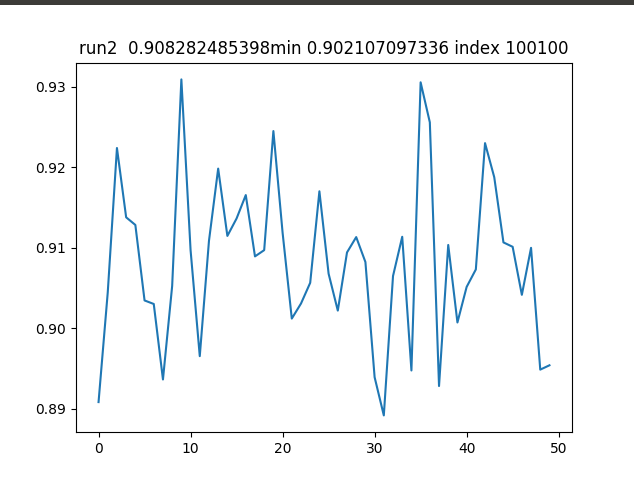


run1的结果要好一些。最大值0.94最小值0.84左右

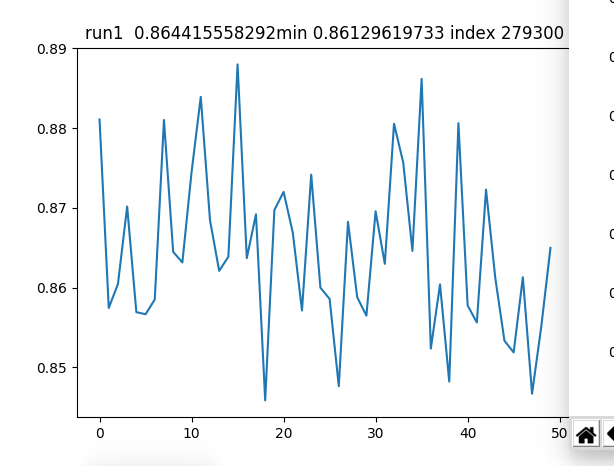
但三个程序都有同样的问题：趋于全关。怀疑是系统本身的问题 。。anyway 增加r2倍数。



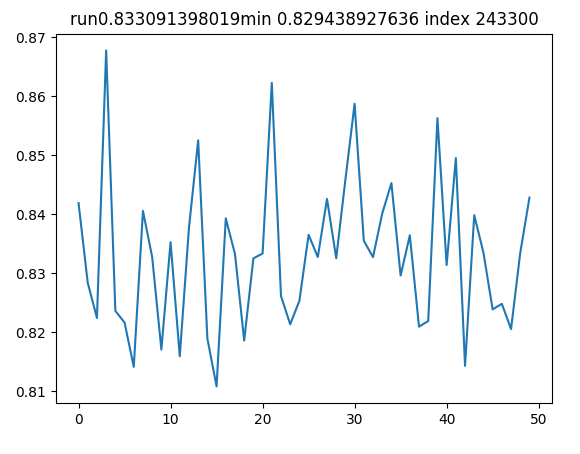
好吧 趋于全关就趋于全关吧。。。。。。

N=3

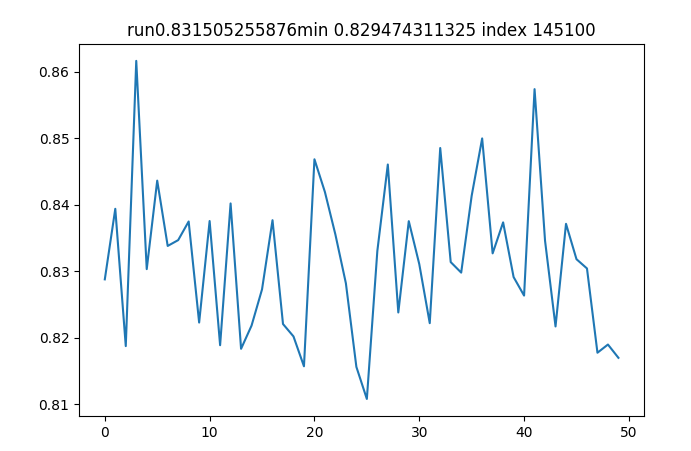
N=4



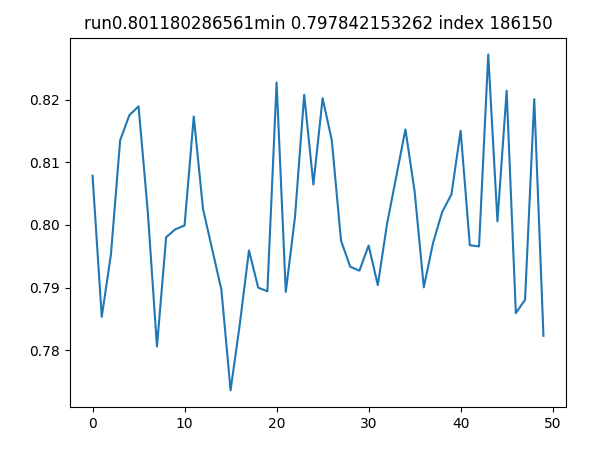
N=5



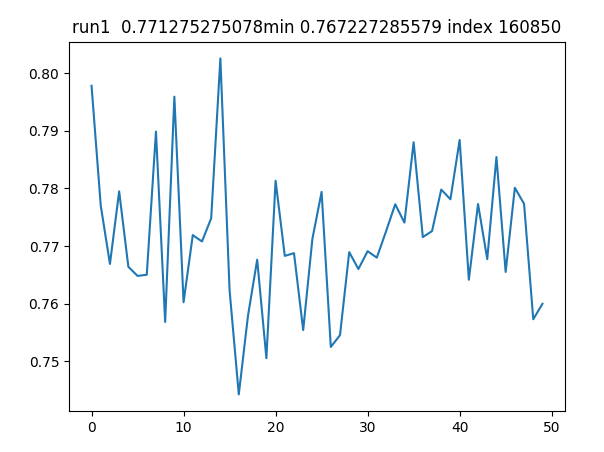
N=5这里有一些不合理，我认为不应该这样的 index应该更大一些才对



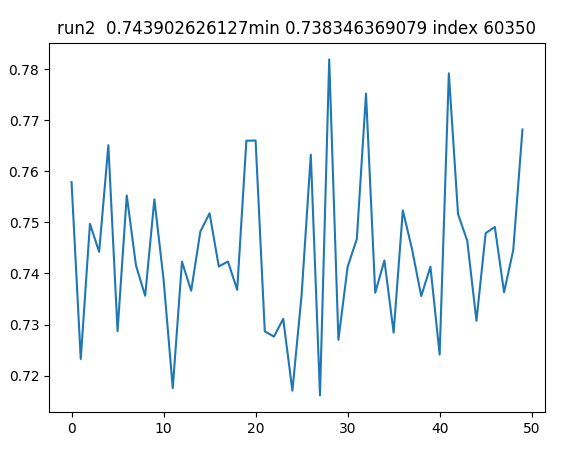
更加不合理了 。。。

下图为N=6时。

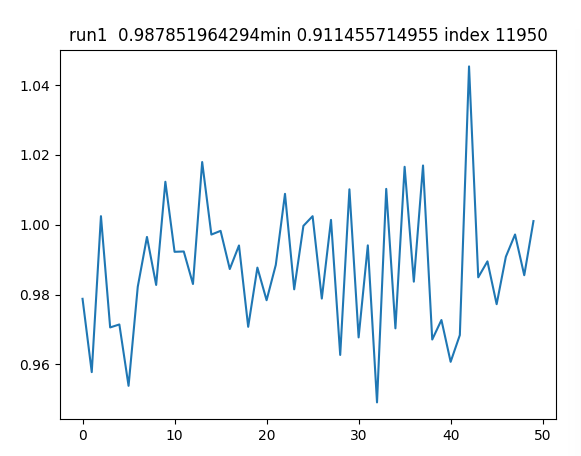
N=7时。



N=8时

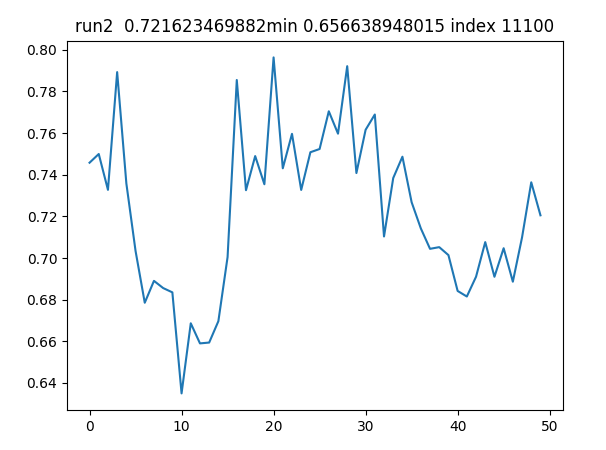
接下来则使用了分布式的Q-learning方式。。

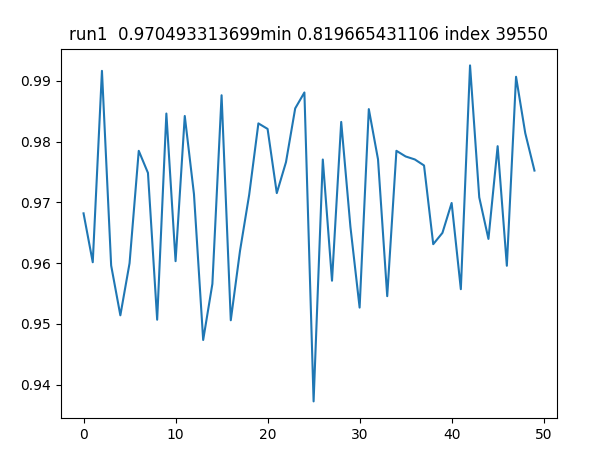
N=7时



明显感觉到收敛更快，但是能效不如集中式。

N=20



但多次实验之后发现分布式QLEARNING方式的收敛速度并不稳定，有时甚至会出现无法收敛的情况。

之前的实验的收敛结果都趋向于基站全关，之后我修改了能量收集和用户到达的函数，终于不是全关了 。