Experiment 2

Introduction

Kmeans is constantly centering batteries, reassigning houses to them and then repeating the same step again. Because the restrictions of capacity and output are so tight sometimes when reassigning the houses no better connections can be made than before, or no legal connections can be made. Because for the next step of recentering the batteries a legal solution is needed. This experiment is carried out to gather evidence for whether it is better to use the hillclimber to switch houses until the connections are legal or if its better to "bruteforce" a legal connection by letting the district dis and reconnect until all houses are connected. This can be done by adding the True keyword to the district.connectGreedy function(figure 1). This shuffles the house-list before the connectGreedy function for each house is called, yielding different starting points.

Methods

For testing this 10x1000 runs with the winning settings of the prior experiment were done. The mean values were plotted (figure 2) and the best found values in 1000 iterations were compared using a ttest.

Results

From visual inspection of figure 2it seems as if a good solution is reached much faster when not using the hillclimber in kmeans. However when testing the differences in mean best solution found it doesnt yield significantly better solutions *($p > 0.1$)*.

Discussion

The advantage in the bruteforce reconnecting variant lays in its efficienty, a legal solutions can be found faster then when using the hillclimber-switcher. This could be because the hillclimber is working agains kmeans because he doesnt want to center or move the batteries but just reconnect to optimize price and produce a legal solution.

```
# If the reconnection results in disconnected houses new attempts are
# made until all houses are connected
while (district.allConnected == False):
    # OPTION A: using hillclimber to reconnect
    for house in district.disconnectedHouses:
        hillclimbSwitcher(house, district, True)
        if len(district.disconnectedHouses) == 0:
            return
    # OPTION B: using disconnect and connectGreedy, if that fails connectRandom
    district.disconnect()
    # the first hundred times a greedy connection is tried
    if c < 100:
        district.connectGreedy(True)
        c += 1
    # If that doesnt work the district is reconnected random
    # to prevent an endless loop
    else:
        print("Connecting random")
        district.connectRandom()
        c += 1
        if c > 200:
            print("random infinite")
            district = contestants[0]
            c = 0
```

Figure 1: Hillclimber or bruteforce

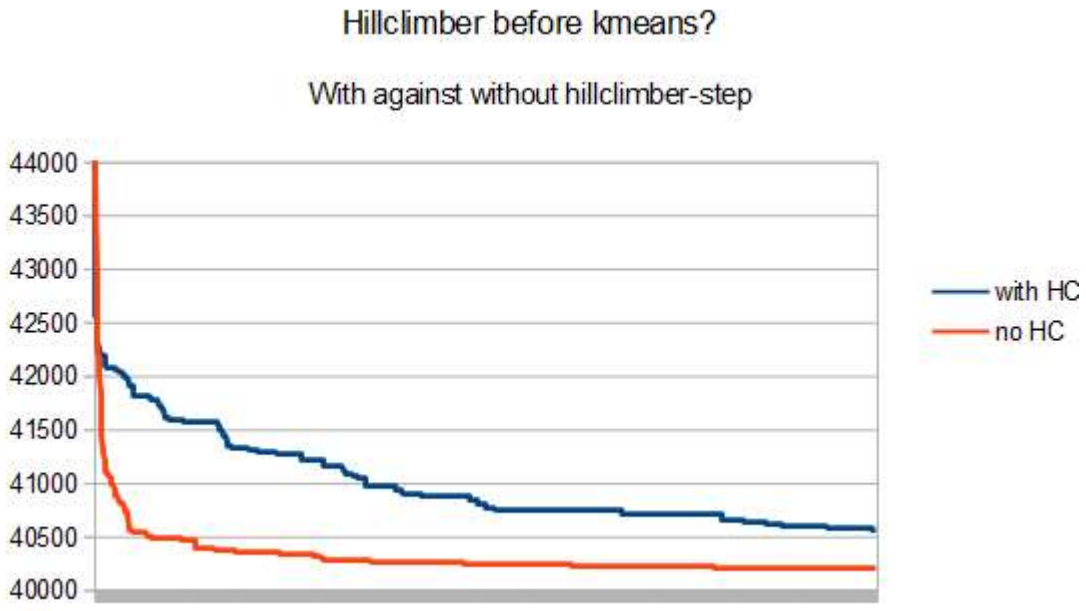

Figure 2: Hillclimber-step