Heuristic exploration of the Smartgrid Case

Angelo Groot, Joeri Fresen, Tamar Vis

May 30^{th}, 2018

1 Introduction

In order to make family homes more sustainable, they can be outfitted with rooftop solar cells. These cells generate energy during daytime to be used after sundown. The generated energy has to be stored; this can be done by implementing one or more batteries in a neighbourhood and connecting all the houses in this neighborhood to these batteries. The capacity, location and houses connected to every battery form a set of variables which can be manipulated in order to reduce costs. In this report two problems regarding this cost-reduction are introduced and algorithms for solving them numerically are evaluated.[4]

Battery type	Capacity (J)	Cost(€)
PowerStar	450	900
Imerse-II	900	1350
Imerse-III	1800	1800

Table 1: Battery types, their capacity and costs.

The first of these subproblems ("case A") provides the problem of connecting the houses to the batteries in a scenario ("grid") in which the positions of houses and batteries, batteries' capacities and houses' outputs are given. This is a constrained optimization problem. The constraint lies in the idea that for every battery in the grid the cumulative outputs of all houses connected to it may not exceed the battery's capacity. The optimization measure in this problem is the cumulative Manhattan distance [1] of the connections, multiplied by the cost-per-unit-length of a connection: 9 [4]. The problem is characterized by a problem complexity of $\frac{b^h}{p}$[2] with b the number of batteries, h the number of houses and p the number of permutations possible for any given list of connections. Because p depends on both b and h, this factor has been neglected in calculations, resulting in a overestimated but definite upper bound complexity. [3]

The second of these cases ("case B") provides the optimization (that is, minimization) problem of finding the number of batteries and their type and location that yields the smallest cost. Three types of batteries are considered, as listed in table 1. The score function for the optimization problem is the sum of the costs of house-battery connections as calculated in case A and the cumulative cost of all batteries in the grid, according to their type (see table 1). The state space complexity of this problem is $\sum_{i=0}^{u} c \cdot \frac{r!}{(r-n)!}$ in which u means the maximum

[1] The Manhattan distance between to objects is the distance between the object measured along two defined and orthogonal axes. In this case, these are 2-dimensional (i.e., a x- and a y-value) and all values are integers. The Manhattan distance is thus calculated $|x_a - x_b| + |y_a - y_b|$[3]

[2] The batteries can be connected to each house, for this we take b^h. To eliminate isomorphic states (e.g 2 houses that are connected to the same battery, but in different chronological order) we divide by p.

[3] In terms of houses, this problem becomes too big to completely search: considering a base of 5 batteries, the 10^{16}-limit of searchability is reached for h being 23 houses. The factor p is a function of permutations and thus scales slower than the power function aforementioned. This means that for larger h's, the power function prevails. Considering this, we expect the state space to be too large to comletely search in a feasible amount of time for a realistic number of houses.

Neighbourhood	mean house output	house output(std dev)	battery capacity	state space
1	50	14.3	1507.00	$7 \cdot 10^{104}$
2	50	9.2	1508.25	$7 \cdot 10^{104}$
3	50	2.9	1506.75	$1.4 \cdot 10^{104}$

Table 2: Some statistics about the grids used for testing given algorithms.

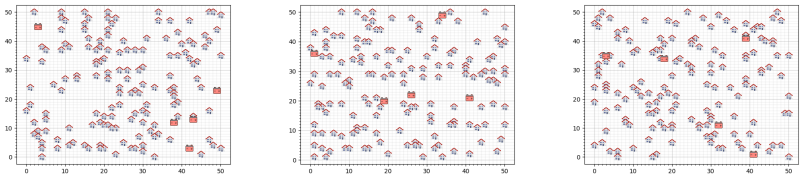


Figure 1: Dummy neighbourhoods for case A: battery capacity and locations are given.

number of batteries necessary to present enough energy-storage capacity, n the means the number of batteries, r the value represented by the initial number of possible positions in which a battery can be placed and c the number of combinations with length i.[4][4]

2 Methods

For both cases, multiple algorithms were implemented and evaluated using three standard configurations of houses (see figure 1). In these dummy-grids (neighbourhoods), house outputs varies, but battery capacity remains constant per grid. Some descriptive statistics of these dummy grids are listed in table 2. For case B, the locations and capacities of the per-grid given batteries are ignored.

2.1 Algorithms

For case A, multiple algorithms were tested. The two algorithms yielding the best score, the Random Battery Cycler and the Random Battery Cycler with additional Hillclimber. These algorithms are described in the figures 8 and 4. A description of the algorithms used in case B is provided in figures 8 and 4. For case B, three algorithms were implemented: one resembling the K-means algorithms (K-bat) and one using a statistical method (Bat-migration). The third algorithm used is the population based algorithm, Batpropagation. The first, K-bat, is depicted in figure 6. The second, Bat-migration, is explained in figure 7 and the third is explained in the figure 4.

[4]We assume all batteries are unique, which is not always the case. This is therefore an overestimation

Table 3: Z-scores and p-values of solutions provided by the Random Battery Cycling algorithm, with and without sequent hillclimber.

Algorithm	Z-value nbh 1	Z-value nbh 2	Z-value nbh 3
Random Battery Cycler	-12.4	-16.5	-16.1
Random Battery Cycler + Hillclimber	-12.5	-16.5	-16.1

Algorithm	p-value nbh 1	p-value nbh 2	p-value nbh 3
Random Battery Cycler	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$
Random Battery Cycler + Hillclimber	$< 10^{-5}$	$< 10^{-5}$	$< 10^{-5}$

2.2 Analysis

The given algorithms for both cases are evaluated by comparing the solution to a large sample of random solutions. The solution is, for case A, valued according to its value for the optimization measure, as given in the Case Introduction. For case B, the value of a solution was valued according to the absolute minimum bound in the connections between houses and batteries (neglecting capacaty restraints), with the cumulative prices of battery added. Because the optimizing, in both cases, means minimization, a smaller score represents a better solution. Although the random solutions are not normally distributed[5], it represents a distribution of sample means [6] of a non-normal distribution, and is thus presumed [7] to resemble a normal distribution.[5]. The mean (μ) and standard deviation (sd) of the random solution distribution are calculated. Then, the fraction $\frac{score-\mu}{sd}$ represents a score for the quality of the generated solution per algorithm (z-score). This score also provides a chance value representing the chances of a random solution finding algorithm finding a solution with this cost, or better.[6] For iterative solutions, this z-score is plotted as function of running time.

3 Results

As means of quantitatively evaluating the algorithms, the Z-scores and subsequent p-scores have been calculated.

3.1 Z-scores

[5]The distribution is a discrete one, as all solution values are integer

[6]The mean value is defined by $\frac{1}{n} \cdot \sum_{i=0}^{n} x(i)$ with x being the score for house i. Because for every grid, the number of houses n is constant, the score $\sum_{i=0}^{n} x(i)$ is equal to the mean multiplied by a constant scalar.

[7]The number of samples generated by the random solver is a million. Although the required number of samples for the Central Limit Theorem to be valid depends on the population distribution, all examples handle a number of the order of 10's[5][1]. Furthermore, the histogram of the solutions provided by the random solution finder highly resembles a normal distribution.

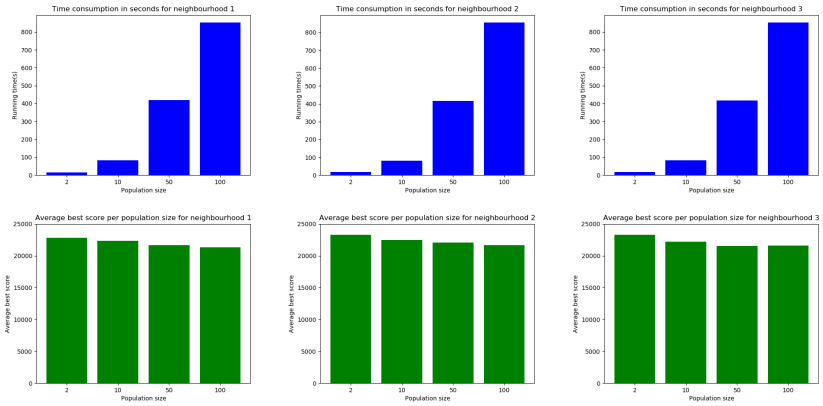


Figure 3: Time consumption and best score values for grid 1 for 2, 10, 50 and 100 individuals per population.

For case A, with for every grid a million random samples as reference, a Z-score was calculated. The Z-scores for the best solution found are listed in table 3. For case B, another reference random sampling was done. This random reference consisted of at least 5000 random solutions. Any solution consisted of a random combination of batteries provided their cumulative capacities were larger than or equal to the cumulative outputs of the neighbourhood's houses (thus enabling the satisfaction of the case A-constraint). For case B, the population based algorithm popbat has yielded the results as shown in figure 4. The corresponding z-scores can be found in figure 5

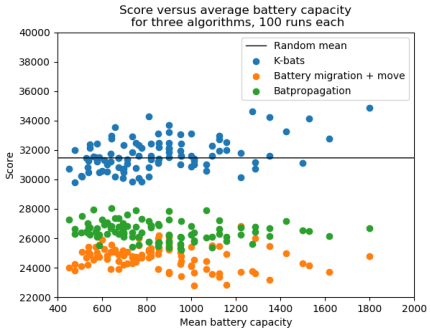


Figure 2: A comparison between the solutions found by the K-bats, Battery migration and Batpropagation algorithm. All were valued according to the battery costs of the solution and the length of the connections found by using the Random Battery Cycle with subsequent Hillclimber. The given case being one of minimization, the higher scores represent a lesser quality. Note the solutions are only shown for grid 1; neighbourhoods 2 and 3 provided similar results.

3.2 Comparison

The three algorithms are compared in figure 2. It can easily been seen that the K-bats algorithm scored overall worse than the other algorithms. Moreover, the solutions provided by the K-bats algorithm are roughly centered around the mean of the random sam-

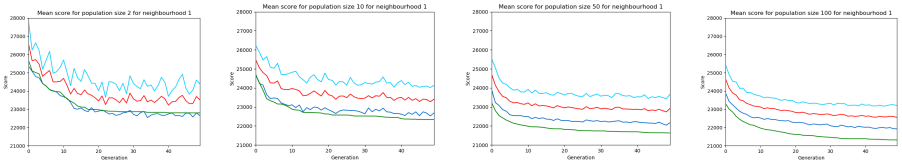
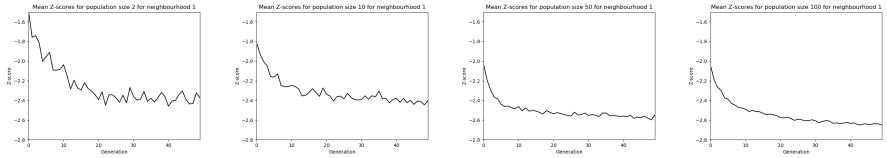


Figure 4: Absolute scores of the population based algorithm. Scores, thus fitness, was the lower bound of the by the algorithm generated battery positions.

Figure 5: Z-scores of the solutions of the population based algorithm.



pling. This means, that it did not score significantly better than an algorithm finding a solution completely at random. However, the other two algorithms did produce better results. Of these, the Battery migration algorithm (the move-heuristic included) score the best for this case.

3.3 Time consumption

Beside the z-scores of algorithm solutions, time consumption was observed per algorithm. For the population based algorithm, the running times per neighbourhood were shown in figure 3. As a reference, the best score found was shown. While time consumption increases, the best values found does not increase significantly.

3.4 Observations

Although on average, the quality of solutions generated by the population based algorithm improved over time, the individual tests show that after 2 to 10 generations (varying per test run), no change in solution's lower bounds was found. Similarly, the change in upper bound disappeared as well after 2 to 5 generations.

4 Conclusion and Discussion

Of a number of tried algorithms, the Random Battery Cycler and the same algorithm with subsequent hillclimber have been found to find the best results for case a. Therefore, the results of the Random Battery Cycler and subsequent hillclimber have been used in determining the scores of case b. In case b, three

algorithms were tested. Of these, the K-bats algorithm produced results close to the mean of the scores of solutions generated at random. However, the Batmigration and Batpropagation algorithms produced better results. Of these, the Batmigration used a very limited timespan (in the range of half a second per run, excluding Random Battery Cycling for valuing the solution) whereas the Batpropagation algorithm represented a larger running time. Herein was a tradeoff between the risk of running into a local optimum (for small population sizes, yielding small running times) and the running time (for large population size, increasing the running time). Neither of the used algorithms were able to give a guaranteed best solution. Considering the state space, it is expected that a complete search is impossible for any feasible running time. Furthermore, it is expected that the K-bats algorithm might prove to be more valuable after additional research. In this implementation, only the locations for placing batteries were determined by the algorithm, ignoring battery capacity and the total output of houses for which the given battery location is the closest battery location. Three algorithms were evaluated for the designing of a system with rooftop panels and neighbourhood-scale energy storage. Given algorithms were able to optimize the system in terms of cost. This way, a sustainable system is made more economically feasible. More research might yield algorithms and heuristics further optimizing the economical cost of the considered system.

References

[1] Central limit theorem: Definition and examples in easy steps.

[2] Vreda Pieterse and Paul E. Black. greedy algorithm. `https://xlinux.nist.gov/dads//HTML/greedyalgo.html`, 2005.

[3] Vreda Pieterse and Paul E. Black. Manhattan distance. `https://xlinux.nist.gov/dads/HTML/manhattanDistance.html`, 2005.

[4] Daan van den Berg. Smartgrid. `http://heuristieken.nl/wiki/index.php?title=SmartGrid`, april 2018.

[5] Michael C. Whitlock and Dolph Schluter. *The Analysis of Biological Data*. Robers and company publishers, 4950 South Yosemity Street, Greenwood Village, CO80111 USA, 2 edition, 2015.

[6] Michael C. Whitlock and Dolph Schluter. *The Analysis of Biological Data*. Robers and company publishers, 4950 South Yosemity Street, Greenwood Village, CO80111 USA, 2 edition, 2015.

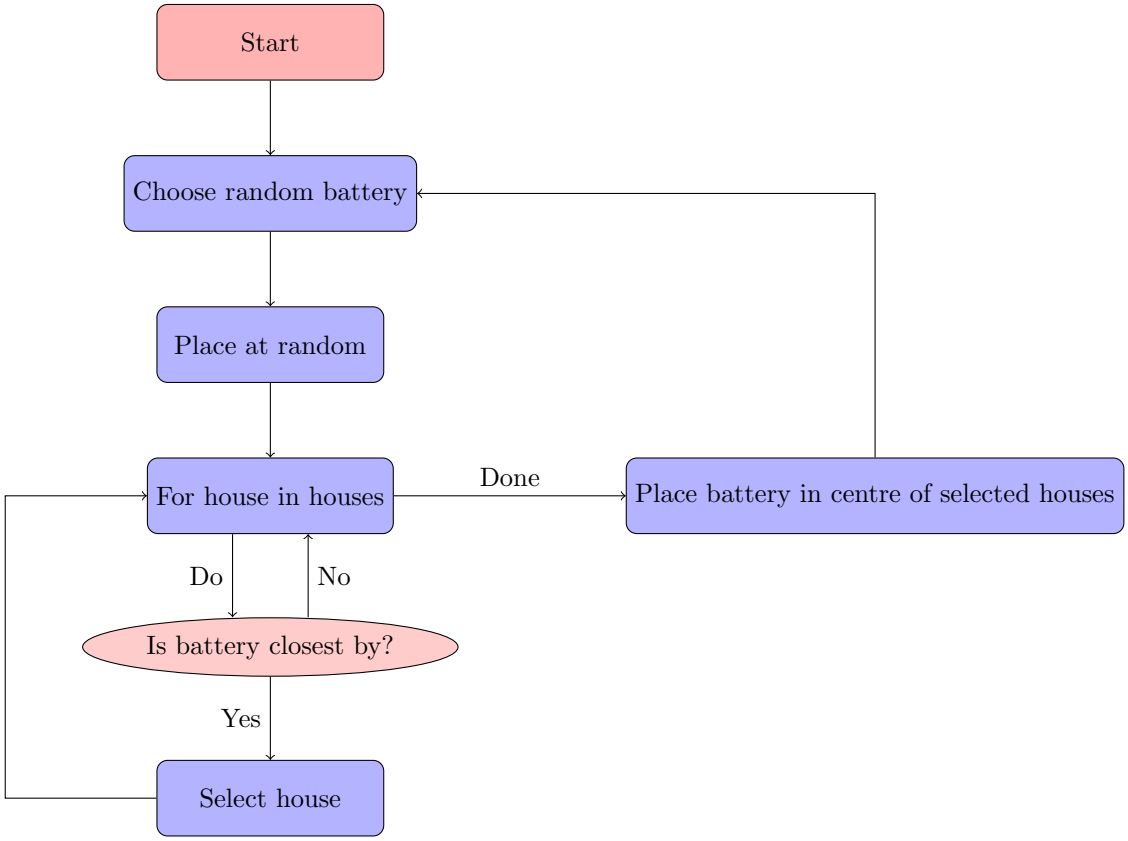
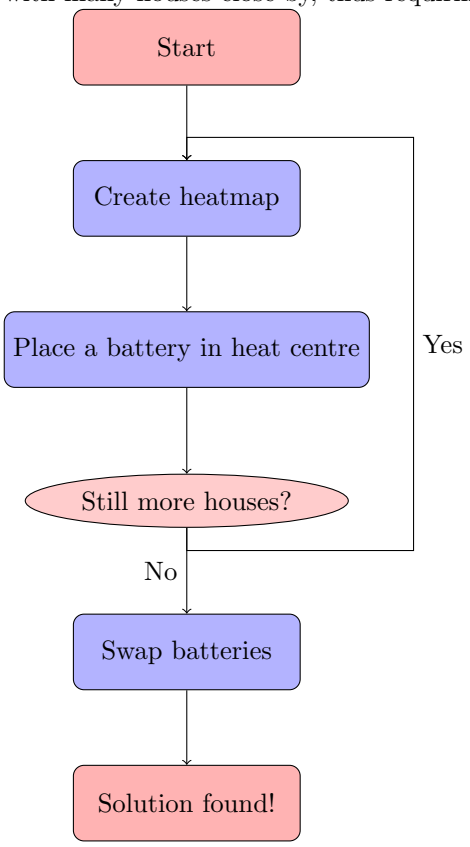


Figure 6: Flow chart for the k-bat algorithm. This algorithms consists of the following steps. First, for each battery, a random battery is chosen. Then, each iteration consists of choosing, for each battery, all houses for which the battery is the closest by lying battery, are selected. Then, the battery is positioned in the middle of the selected houses. These steps are repeated until no significant change is made by the algorithm (i.e., the change per iteration is smaller than a given parameter).

Figure 7: Flow chart for the bat-migration algorithm. The algorithm uses the concept of a heatmap. Two types of heatmap are used: the global heatmap, being the local density. The local density is a measure calculated as $\frac{1}{\sum_{n=0}^{n=N} d(house,battery)}$ with $d()$ being the Manhattan distance function between the house n of N and the point in the grid. In contrast, the global density is calculated as $\sum_{n}^{N} \frac{1}{d}$ In this measure, the distance of a house to a point is weighted more. The algorithm first creates a local heatmap, using above given function, for every point. It then selects the point at which the "heat"-value is largest and selects a battery to place. Then, houses are connected, in increasing order of distance to the battery, to the battery, until the battery can no longer support any more houses. A second heatmap is generated, ignoring the already connected houses. These steps are repeated until all houses are connected. Then, the algorithms creates a global heatmap. Between the by the local heat map found locations, the batteries are swapped in such way that the batteries with the largest capacity are placed in the places with the highest value for the global heatmap. This is, because the global heatmap returns a high value for locations with many houses close by, thus requiring a battery with a large capacity.



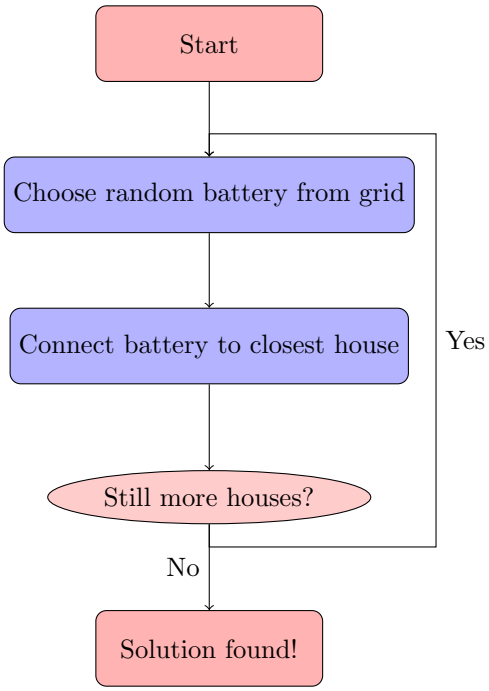


Figure 8: Flowchart for Random Battery Cycler algorithm (case a). The Random Battery Cycler randomly picks one of the batteries in each iteration. Then, the algorithm finds, out of all unconnected houses in the grid, the unconnected house closest to the chosen battery and connects it to the chosen battery. These steps are repeated until all houses are connected. This way, the algorithm works like a greedy algorithm [2]. As a variation, the solution provided by this algorithms has been used as a starting point for a steepest-ascent hillclimber (see image 4.

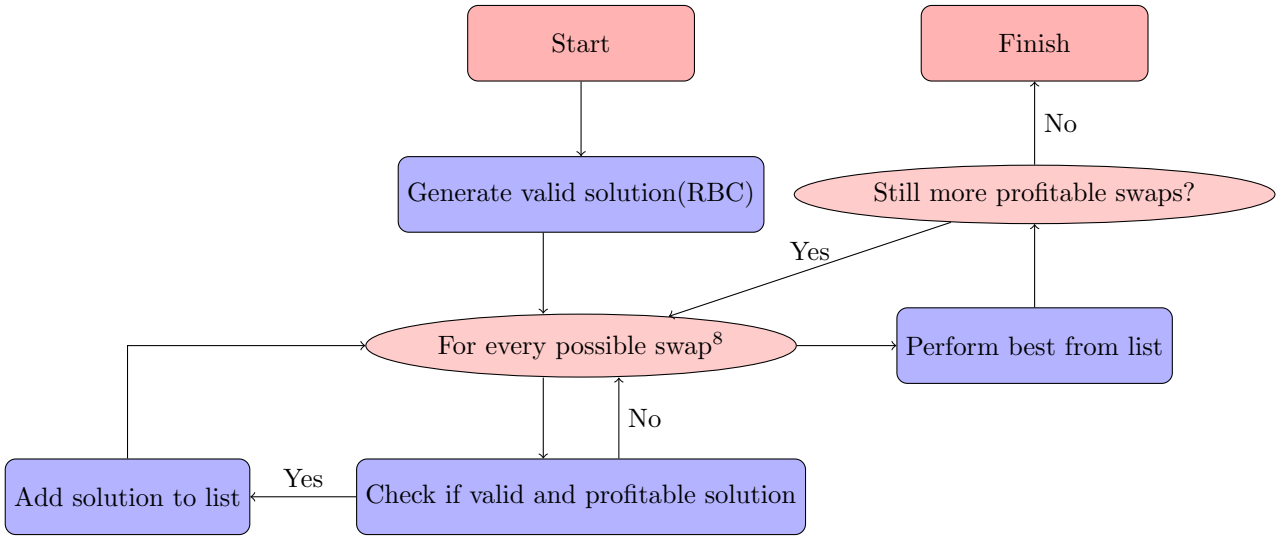


Figure 9: Flowchart for the combination of Random Battery Cycler followed by n iterations of hillclimber. For the details of the Random Battery Cycler algorithm, see figure 8. The hillclimber makes a list of every valid (i.e., constraint satisfying), profitable (i.e., lowering the total connection distance) swaps for a solution given by the Random Battery Cycler. Then, it performs, from that list, the most profitable swap. It then tries again to make a list of valid and profitable swaps. If none exists, it returns the given solution. Else, it repeats the given steps.