Faculty of Science (FNWI)

Heurestieken

Huizenprobleem van Amstelhaege

Puja Chandrikasingh , Luuk v.d. Kleij, Radmir Leushuis

11059842, 11004517, 10988270

22 Mei 2018

1 Experiments

1.1 Steepest Ascend Hill Climber vs Stochastic Hill Climber

We define the steepest ascend hill climber used in this case as an algorithm which works the following way:

1) Select a random house to move

2) Select a random distance to move the selected house (both Δx and Δy are randomly selected)

3) Evaluate the score function when moving the house in the top (0, $y + \Delta y$), right($x + \Delta x$, 0), left (-$x + \Delta x$, 0) an bottom (0, -$y + \Delta y$) position using the random distance (as can be seen in Figure 1)

4) Select the highest improvement to the score function (if improvement is negative or not feasible repeat from step 1)

5) Move the house into the direction which results in the highest improvement

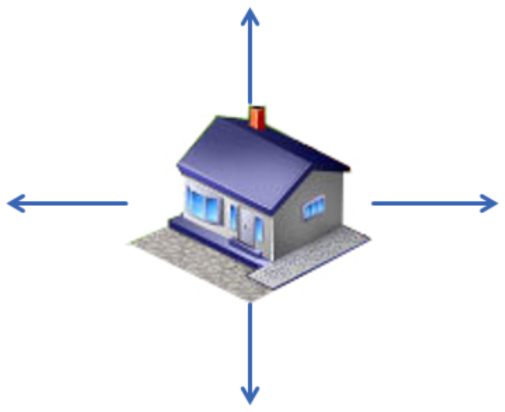

Figure 1: Visualization of what direction a house using the steepest ascend hill climber can be moved

We define the stochastic hill climber used in this case as an algorithm which works the following way:

1) Select a random house to move

2) Select a random distance to move the selected house (both Δx and Δy are randomly selected)

3) Evaluate the score function when moving the house in the following direction: $(x^{new}, y^{new}) = (x, y) + (\Delta x, \Delta y)$

4) If improvement is negative or not feasible repeat from step 1

5) Move the house into the specified direction

Figure 2: Visualization of what direction a house using the stochastic hill climber can be moved

The first thing which can be noticed is that the steepest ascend hill climber has to evaluate a minimum of four solutions, while the stochastic hill climber has to evaluate a minimum of only one. Thus, given both algorithms select the same house to move and distance, the stochastic hill climber is likely to find a feasible improvement of the score function quicker than the steepest ascend hill climber. If temporary restricting the maximum amount of times the algorithms can try to find an improvement before continuing to the next iteration to compare the "speed" of both algorithms, the following two bar charts surface when running 1000 simulations (with distances randomly selected out of the range of $[-1, 1]$):

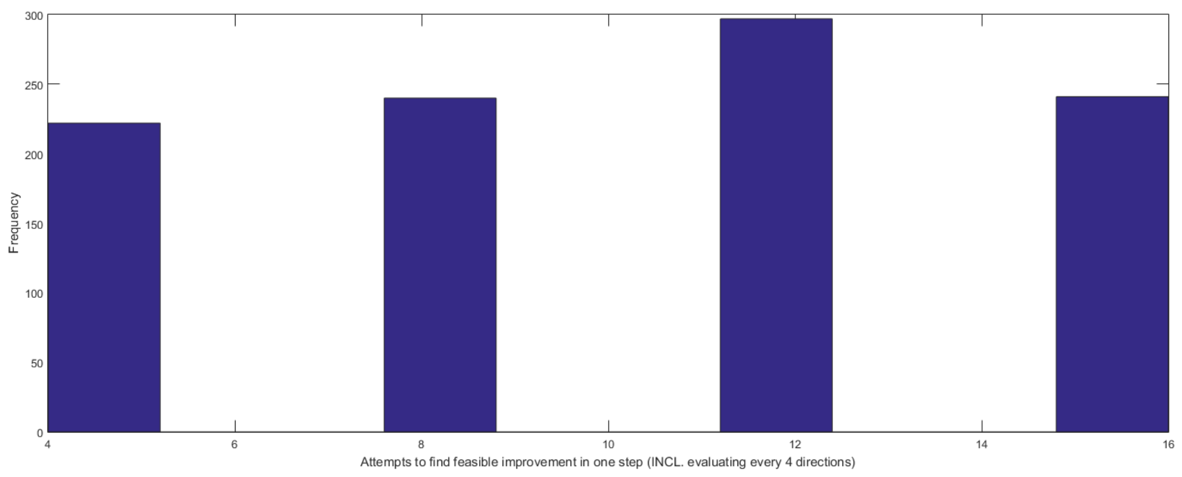

Figure 3: Bar chart of attempts made to find improvement for steepest ascend hill climber

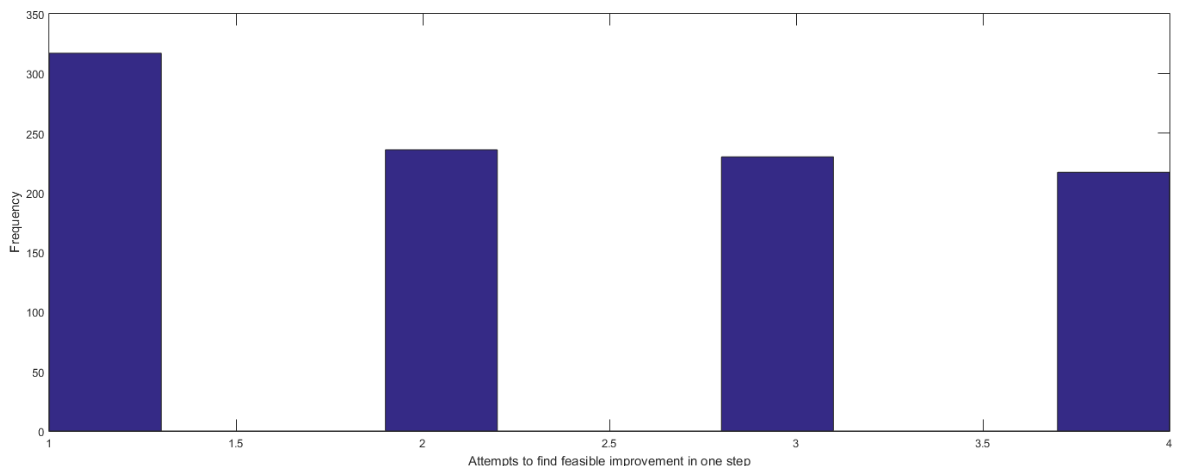

Figure 4: Bar chart of attempts made to find improvement for stochastic hill climber

As can be seen the bar charts above, the stochastic hill climber almost always requires a smaller or equal amount of tries before finding a feasible improvement. This is logical as it has to evaluate only one improvement instead of four. However, speed is not the only criteria, which should be examined to conclude which algorithm is preferable over the other. We also evaluate which algorithm yields a higher score when ran a 1000 times. This gives us the following 2 bar charts:

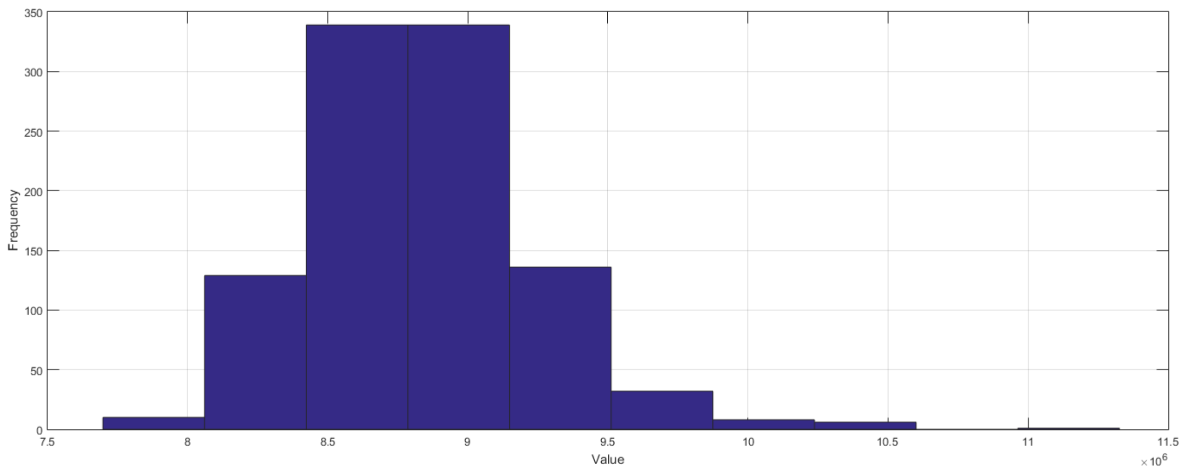

Figure 5: Bar chart of scores found for steepest ascend hill climber

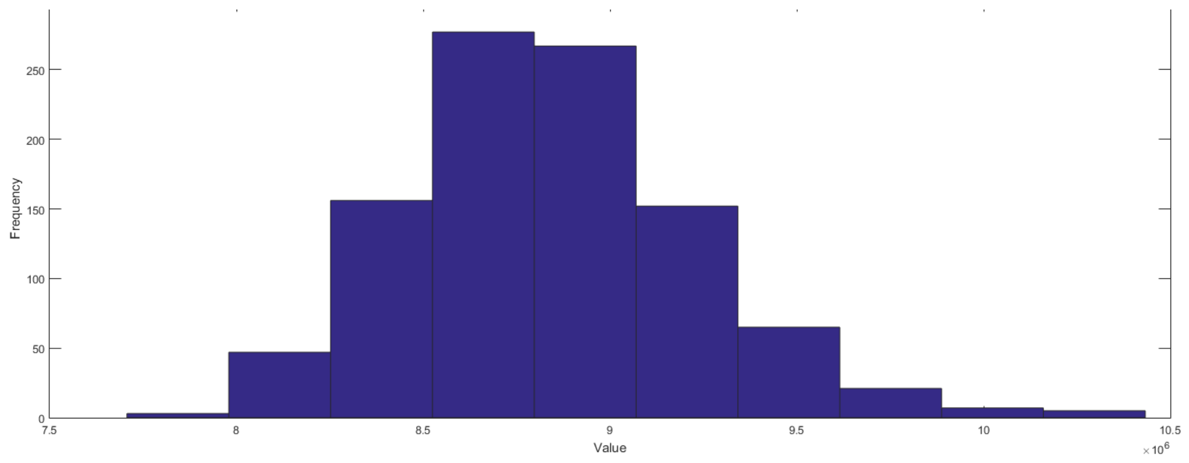

Figure 6: Bar chart of scores found for stochastic hill climber

By looking at the sample averages (8.7362e+06 for steepest ascend hill climber and 8.7593e+06 for stochastic hill climber) we suspect that the stochastic hill climber yiels on average a higher score. To test whether the larger mean is statically significant, we test the following hypothesis:

$H_0 : \mu_{stoch} > \mu_{non-stoch}$ vs $H_1 : \mu_{stoch} \leq \mu_{non-stoch}$

Using the following test statistic:

$$t = \frac{(\bar{\mu}_{stoch} - \bar{\mu}_{non-stoch})}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{n}}} = \frac{8.7593e+06 - 8.7362e+06}{\frac{1}{\sqrt{1000}}(1.4841e+11 + 1.5799e+11)} = 2.384093145e-06$$

As the test statistic is very small and definitely smaller than the critical value for a 95% confidence interval, there is not enough statistical evidence to conclude that the mean of the stochastic hill climber is statistically higher than that of the steepest ascend hill climber. However, given the increased speed of the stochastic hill climber, it is preferable to the steepest ascend hill climber.

1.2 Constant range vs Variable range

In the past the magnitude (distance) of a step is randomly (uniform) selected from a constant range. However, this means that the range is the same at the end as at the beginning of the algorithm. This causes inefficiencies, since the algorithm has the same chance of taking large steps in every iteration. To make the algorithm faster we define the following 3 functions:

$$\alpha_i = 50 \cdot \left(\frac{max\_iterations}{i}\right)^{\frac{1}{3}} \cdot \left(\frac{20}{tot\_number\_houses}\right)^3 \tag{1}$$

where: i = iteration number i; $i \in \mathbb{Z}^+$; i $\in [0, max\_iterations]$

$$\beta_i = \frac{improvement_i}{value_{i-1}} * 100 \tag{2}$$

where: i = iteration number i; $i \in \mathbb{Z}^+$; i $\in [0, max\_iterations]$

$$\epsilon_i = random(0, \frac{\alpha + \beta}{2}) \tag{3}$$

where: i = iteration number i; $i \in \mathbb{Z}^+$; i $\in [0, max\_iterations]$

$random$ = function which chooses a value from uniform distribution

Next we define the function combining these 3 elements: $range_i = \alpha_i + \beta_i + \epsilon_i$. Each component plays a crucial role in determining the variable range. The α (decaying learning rate) is a measure for how far the algorithm is in its iterations compared to the maximum number of iterations. As $\frac{\delta \alpha_i}{\delta i} = -\frac{50}{3} \cdot \frac{(max\_iterations)^{\frac{1}{3}}}{i^{\frac{4}{3}}} \cdot \left(\frac{20}{tot\_number\_houses}\right)^3 < 0$, we can see that α indeed is a strictly decreasing function for the number of iterations.

The β (momentum) is a measure for how large the relative improvement was in the last iteration. As big improvements tend to follow big improvements and small improvements follow small improvements (especially in the beginning and end of the iterations), this term makes sure that the range accounts for this phenomena.

The last term ϵ (correction term) is crucial for breaking potential faulty trends. As the expectation is that both α and β will decrease as the number of iterations grow. It is possible that (given that the magnitude of the step is chosen randomly out of $range_i$) the chosen step is very small. This will probably lead to a small improvement which in turn will lead to the next β_{i+1} also being small, thus the range will fall into a decreasing trend. To test whether this trend is a correct one and not a faulty one ϵ takes on a random value. If the trend is a correct one the $range_i$ will almost immediately return to the trend, if not than the $range_i$ escapes the trend and moves into the correct trend.

To test whether this variable range leads to a faster algorithm we run 1000 iterations using the stochastic hill climber (capping the maximum number of tried allowed to make before moving to the next iterations by 20) and examine the following 2 bar charts:

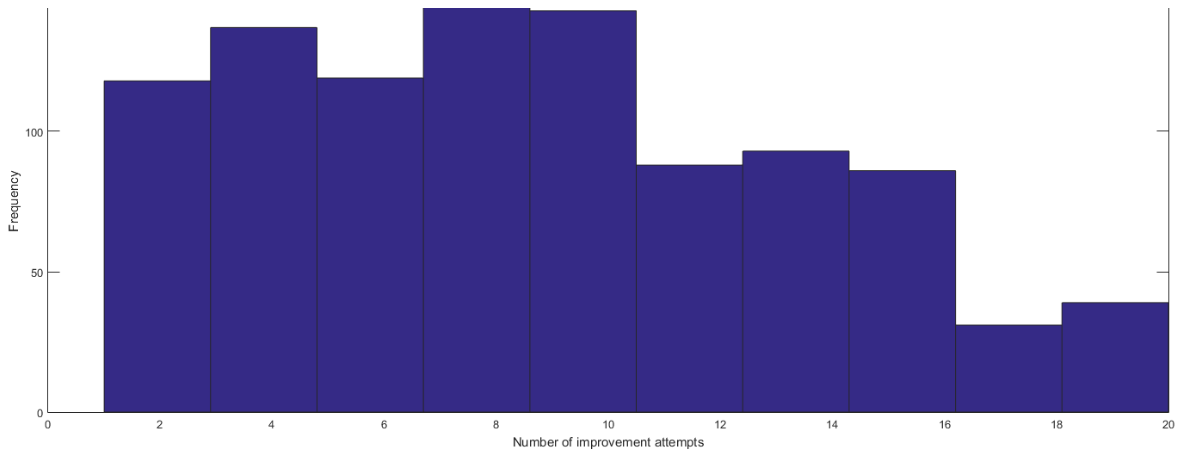

Figure 7: Bar chart of attempts made to find improvement with constant range [-10,10]

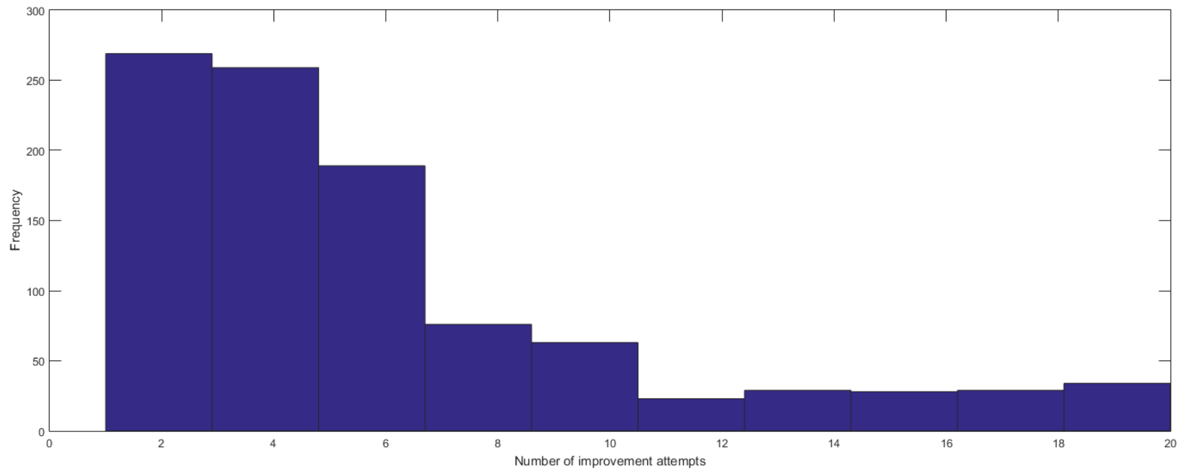

Figure 8: Bar chart of attempts made to find improvement with variable range

By simply looking at the averages we can see that using the variable range (5.7167) is preferable over using a constant range (8.6797). To further test this we approximate the frequencies with a continuous gamma distribution. This yields the following results:

Cons. Range $\sim \Gamma(2.3024, 3.7698)$

Var. Range $\sim \Gamma(1.6660, 3.4313)$

First we calculate at what point the CDF of the Gamma distribution of the constant range reaches the value of 0.5 (midpoint) by solving the following equation.

$\int_0^x f(q; 2.3024; 3.7698)dq = 0.5 \;\Leftrightarrow\; x = 7.4605$

Next we enter the found midpoint in the CDF of the Gamma distribtuion of the variable range:

$F(7.4605; 1.6660; 3.4313) = \int_0^{7.4605} f(q; 2.3024; 3.7698)dq = 0.7307$

Since 0.7307>>0.5, we can easily conclude that using the variable range is vastly preferable to using a constant range.

1.3 Testing Different Cooling Schemes for Simulated Annealing

We define the following different cooling schemes:

$$T_n = T_0 - \frac{n(T_0 - T_N)}{N} \tag{4}$$

where: T_n = linear cooling scheme; $T_n \in \mathbb{Z}^+$;

T_0 = starting temperature; $T_n \in \mathbb{Z}^+$;

$$T_n = T_0 \left(\frac{T_n}{T_N}\right)^{\frac{n}{N}} \tag{5}$$

where: T_n = exponential cooling scheme; $T_n \in \mathbb{Z}^+$;

T_0 = starting temperature; $T_n \in \mathbb{Z}^+$;

$$T_n = T_N + \frac{T_0 - T_N}{1 + e^{0.3(n - \frac{N}{2})}} \tag{6}$$

where: T_n = sigmoidal cooling scheme; $T_n \in \mathbb{Z}^+$;

T_0 = starting temperature; $T_n \in \mathbb{Z}^+$;

$$T_n = \frac{T_0}{log(n+1)} \tag{7}$$

where: T_n = geman cooling scheme; $T_n \in \mathbb{Z}^+$;

T_0 = starting temperature; $T_n \in \mathbb{Z}^+$;

We define 2 different end temperatures, one being 200 and the other being 0.01. To get an idea of how the temperature develops over the increasing iterations the following line graphs are made:

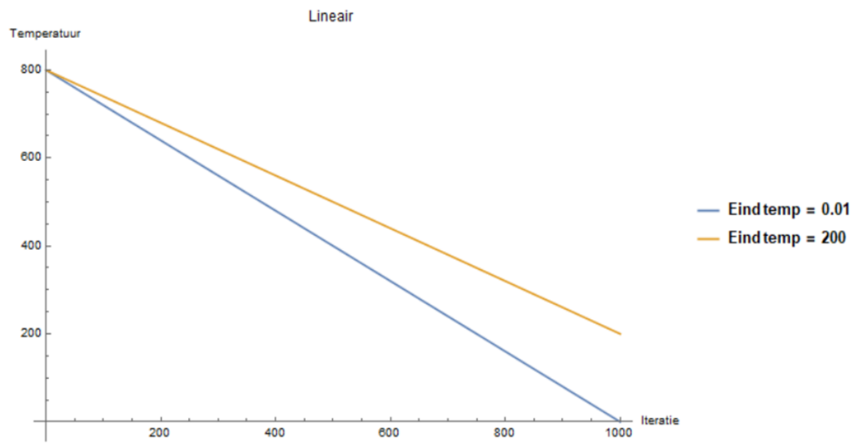

Figure 9: Line graph of the temperature using the linear cooling scheme

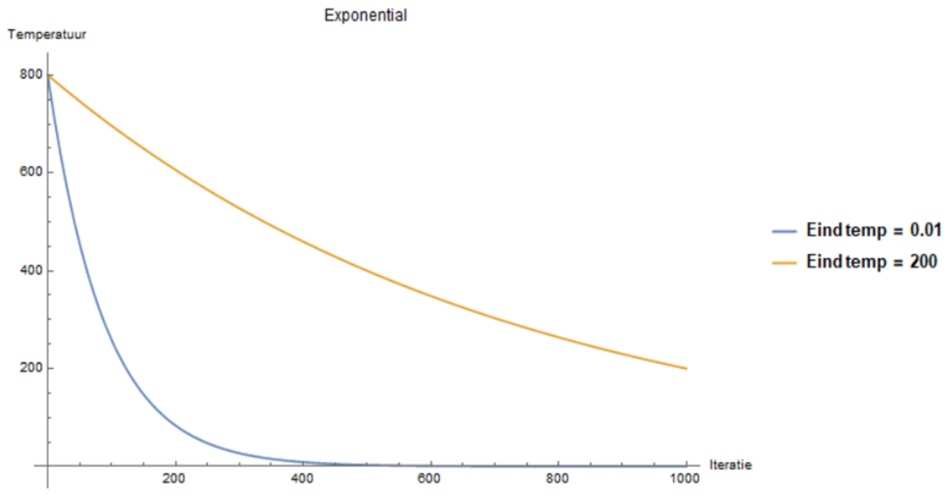

Figure 10: Line graph of the temperature using the exponential cooling scheme

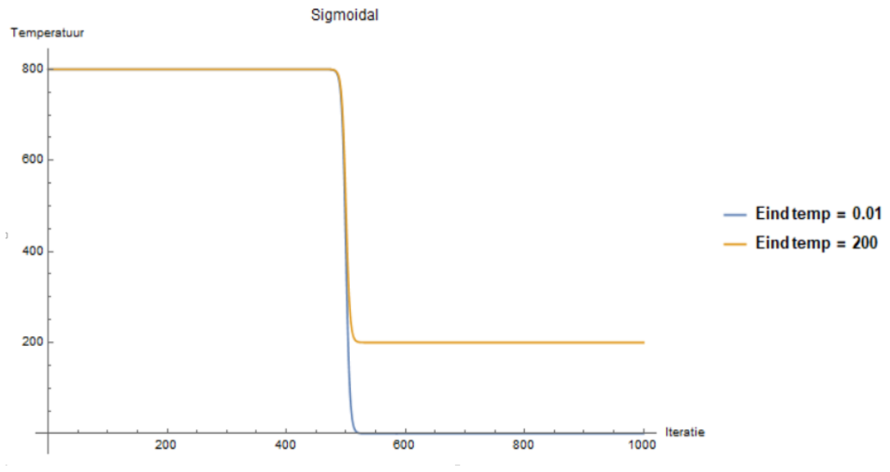

Figure 11: Line graph of the temperature using the sigmoidal cooling scheme

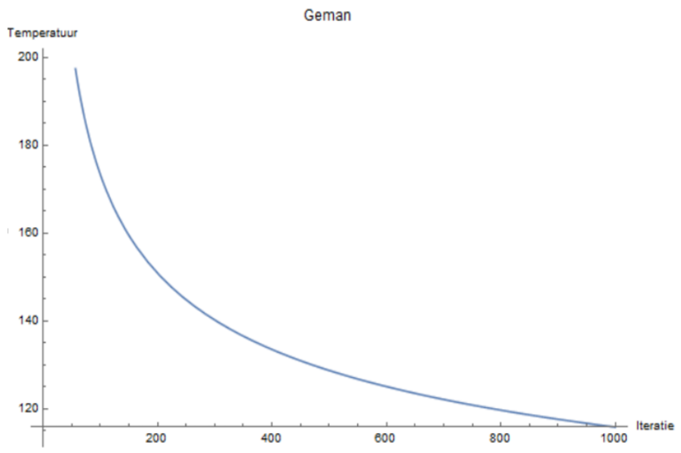

Figure 12: Line graph of the temperature using the geman cooling scheme

Now that we have an idea of how the temperature develops, we will study the chance of accepting an negative improvement of -20 for the different cooling schemes:

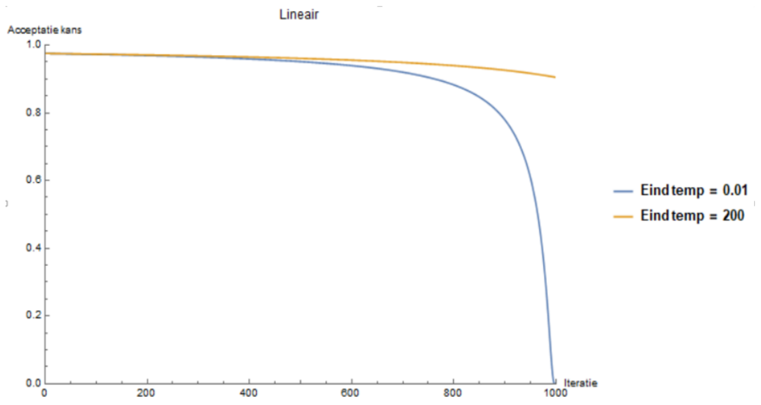

Figure 13: Line graph of the chance of accepting -20 as improvement using the linear cooling scheme

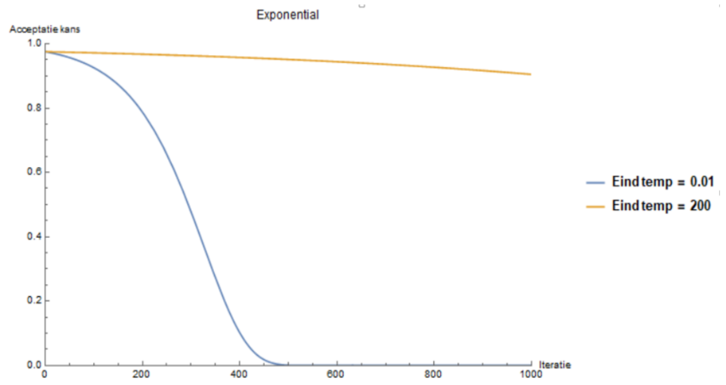

Figure 14: Line graph of the chance of accepting -20 as improvement using the exponential cooling scheme

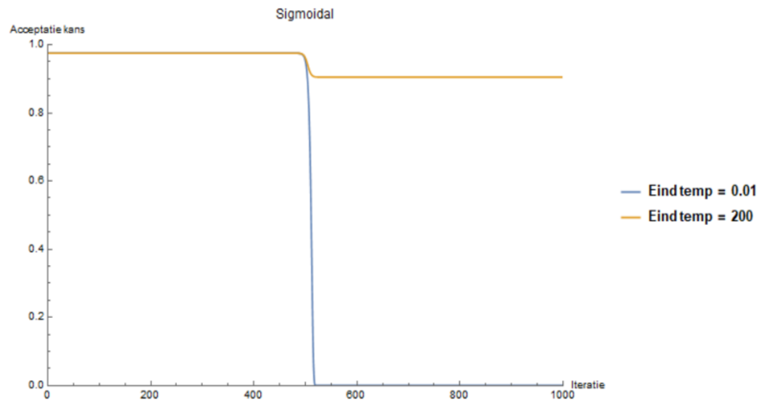

Figure 15: Line graph of the chance of accepting -20 as improvement using the sigmoidal cooling scheme

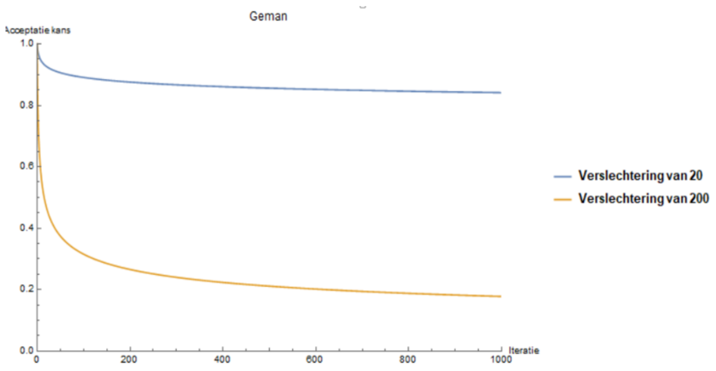

Figure 16: Line graph of the chance of accepting -20 as improvement using the geman cooling scheme

Now that we have a general idea of how all of the temperatures and chances look, we will look at which temperature in combination with which end temperature yields the highest solution results. By running 200 iterations 300 times and storing the average for all different cooling schemes we get the following results:

Cooling scheme	*End temperature*	*Average Score*
Linear	200	9924088.355
Exponential	200	9924487.727
Sigmoidal	200	9889662.271
Linear	0.01	9896427.264
Exponential	0.01	**9940630.441**
Sigmoidal	0.01	9940595.771
Geman	N/A	9919185.291

Table 1: Table with all average scores for different cooling schemes

As can be seen above using the exponential cooling scheme with an end temperature of 0.01 yields on average the highest scores (9940630). The runner up, the sigmoidal cooling scheme with 0.01 as end temperature (9940596), also yields on average high scores. To test which one is preferable over the other we look at a bar chart of all scores using the corresponding cooling schemes:

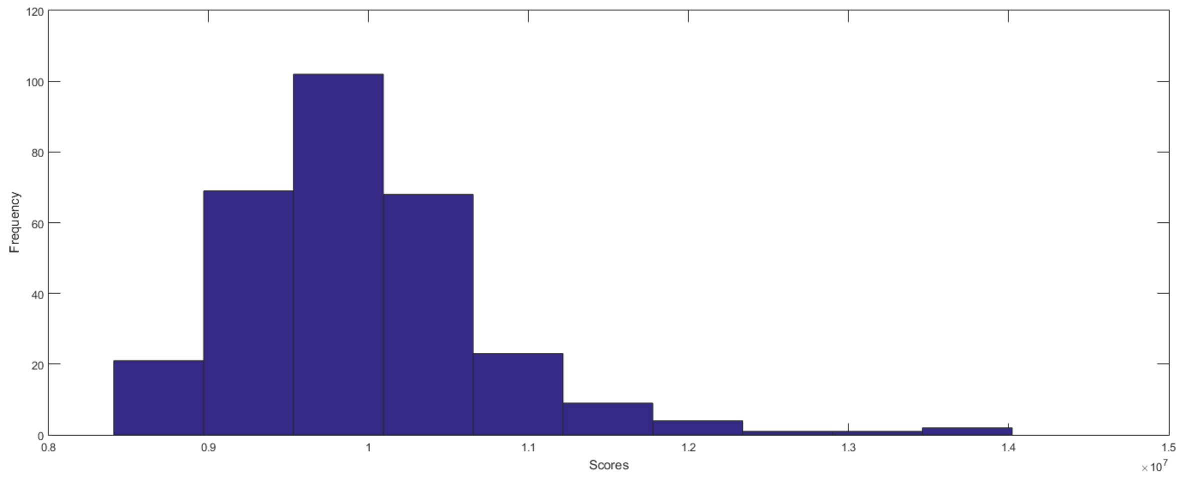

Figure 17: Bar graph of the scores using the sigmoidal cooling scheme (end temp = 0.01)

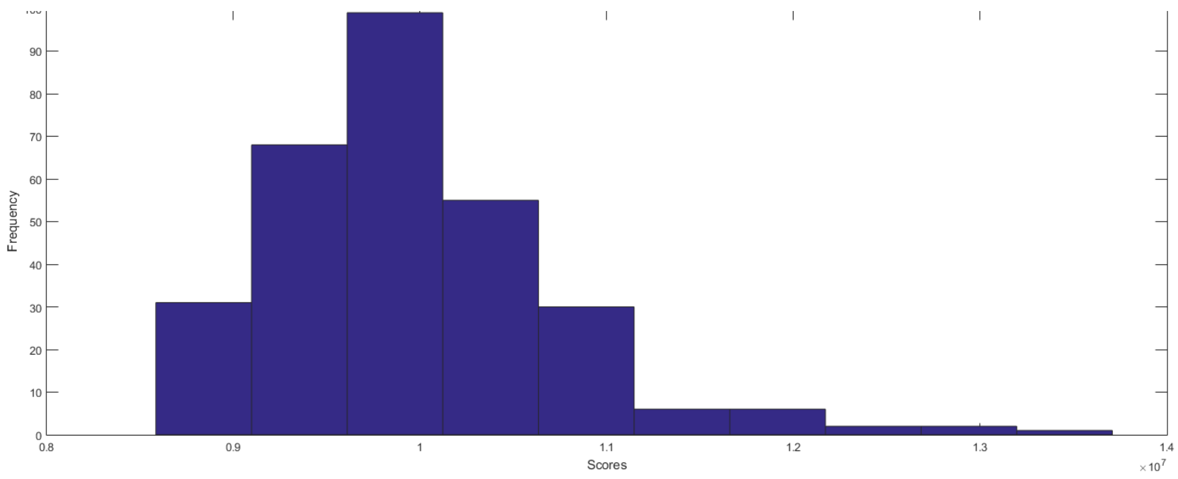

Figure 18: Bar graph of the scores using the exponential cooling scheme (end temp = 0.01)

To test whether the exponential cooling scheme is preferable we calculate the mean of the sigmoidal cooling scheme (9.8898e+06). Next we approximate both of the cooling schemes using a normal distribution this yields:
$sig \sim N(9.9406e+06, 780930)$
$exp \sim N(9.94063e+06, 772948)$
The first thing we notice is that the approximation of the exponential cooling scheme also has a higher mean. Another crucial thing we notice is that the variance is lower than the variance of the sigmoidal cooling scheme. Thus we can conclude that using the exponential cooling scheme with an end temperature of 0.01 is preferable over all other cooling schemes.

1.4 Stochastic Hill Climber and Simulated Annealing vs Combination

A major problem with a stochastic hill climber is that it has a large chance of getting stuck in a local maximum. An algorithm which solves this problem is the simulated annealing algorithm, by allowing negative improvements the chances of getting stuck in a local maximum are significantly reduced. However, allowing negative improvements to be accepted at the start of the algorithm usually is not that beneficial, as can also be seen in Table 2 in the average score of the simulated annealing algorithm. This is due to the beneficial effect of the simulated annealing, being able to escape a local maximum, surfacing only relatively late (when the algorithm gets stuck in a maximum). Ideally, an algorithm uses a stochastic hill climber up until it gets stuck in a local maximum (measured by the number of iterations the change is below a chose threshold) and then switches to a simulated annealing. We made such an algorithm which we call the combination algorithm. Running 100 iterations 300 times and taking the average leads to the following values in Table 2.

Algorithm	*Average Score*
Simulated Ann.	9.1708e+06
Stochastic Hill	9.4792e+06
Combination	9.8845e+06

Table 2: Table with all average scores for different algorithms

As can be seen the combination algorithm outperforms the individual components by a significant amount. By studying the bar charts of all optimizations:

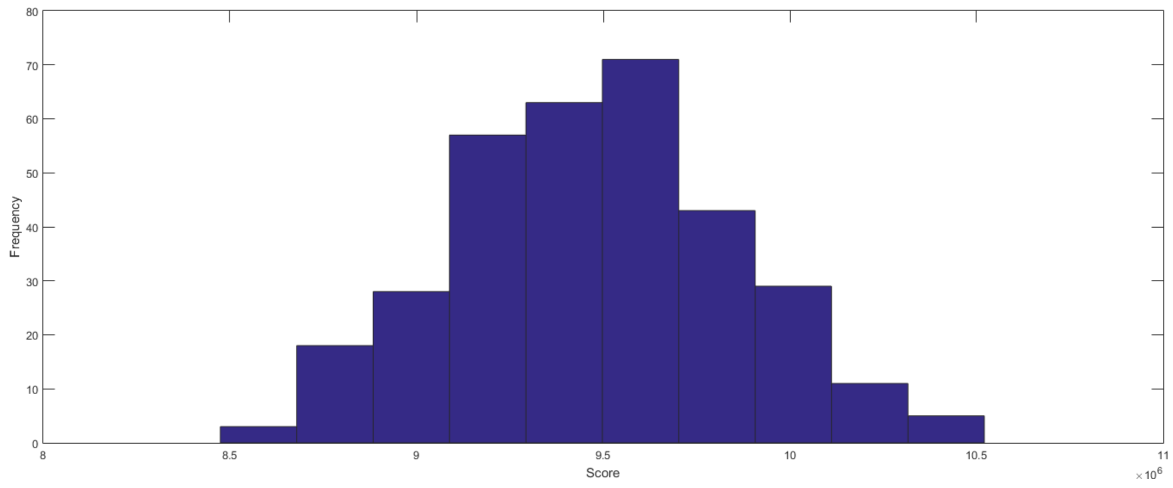

Figure 19: Bar graph of the scores using the simulated annealing algorithm

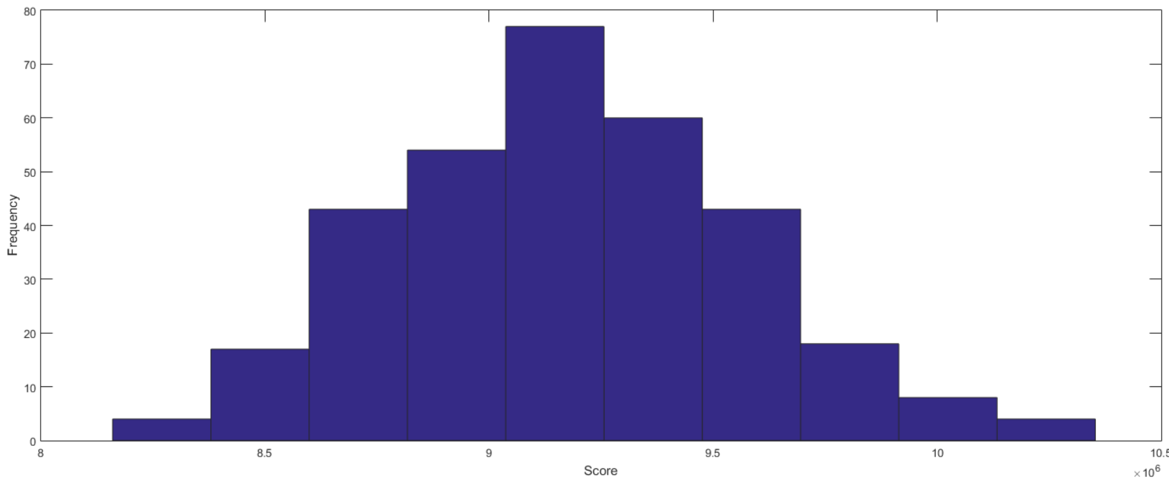

Figure 20: Bar graph of the scores using the stochastic hill algorithm

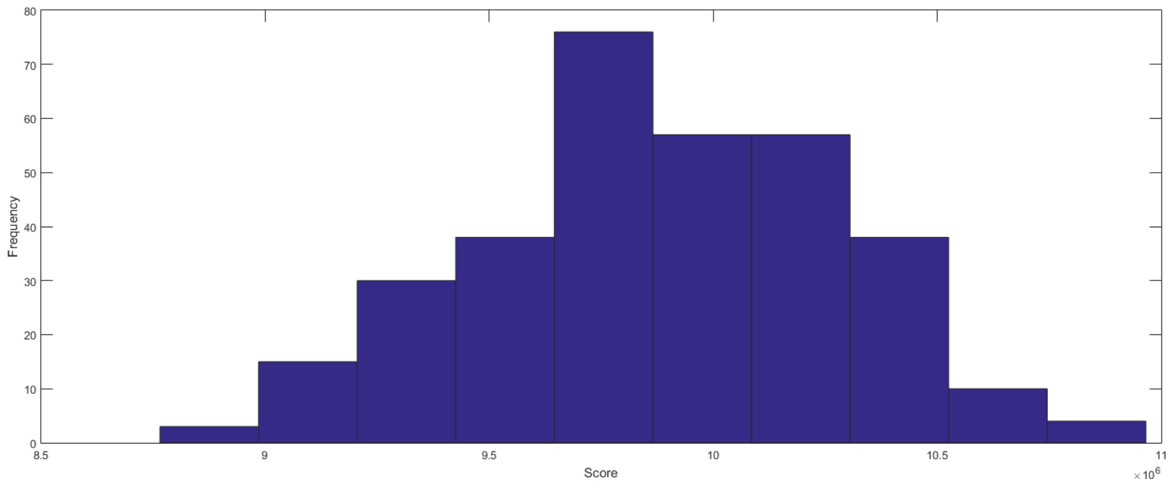

Figure 21: Bar graph of the scores using the combination algorithm

We can also see that the combination algorithm in Figure 21 is severely skewed to the right, which means that a large portion of the scores are positioned right of the average. This would imply that more observations are larger than its average than smaller. Given that the average of the combination algorithm is already 405300 larger than the nearest competitor (stochastic hill) this provides ample evidence to conclude that the combination algorithm is preferable to its 2 individual components.

1.5 Minimization of difference from optimal vs Random

When determining the upper bound of the problem an optimal free space for each type of house is determined. This is of course a gross overestimation. However, it does provide some rope in the matter of how much each type of house should have in free space. This provides a method of generating a starting solution. By minimizing the total sum squared of the difference between the free space of each house and the ideal free space of each house a starting solution is created. After this, the usual algorithms can move the houses even further to find the combination which maximizes the total value of the grid. In this section we test if minimizing the difference yields final solutions (using combination algorithm after minimization) with a higher total value than generating a random solution and optimizing it with the combination algorithm. By running 30 iterations 300 times, we get the following results:

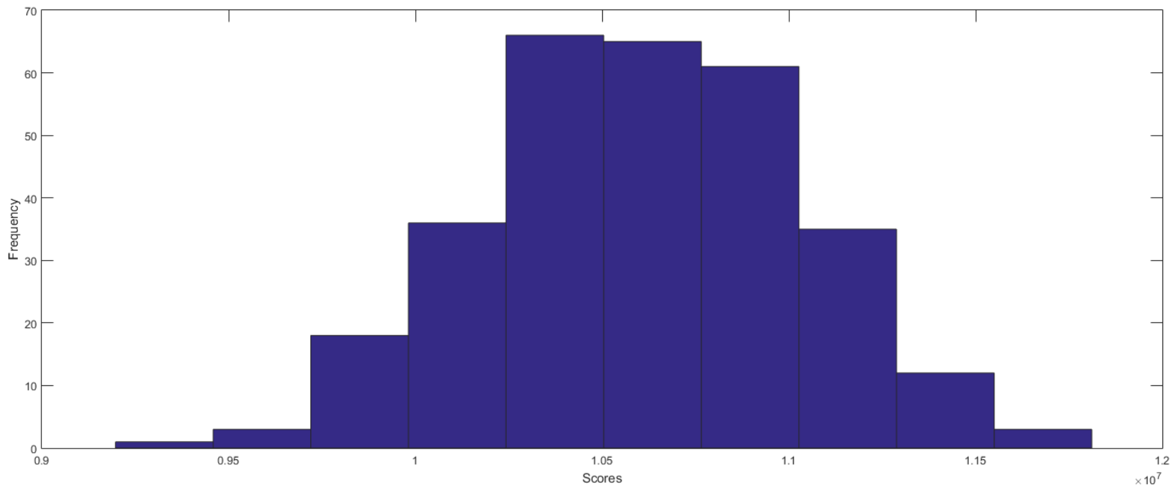

Figure 22: Bar graph of the scores using the random starting solution

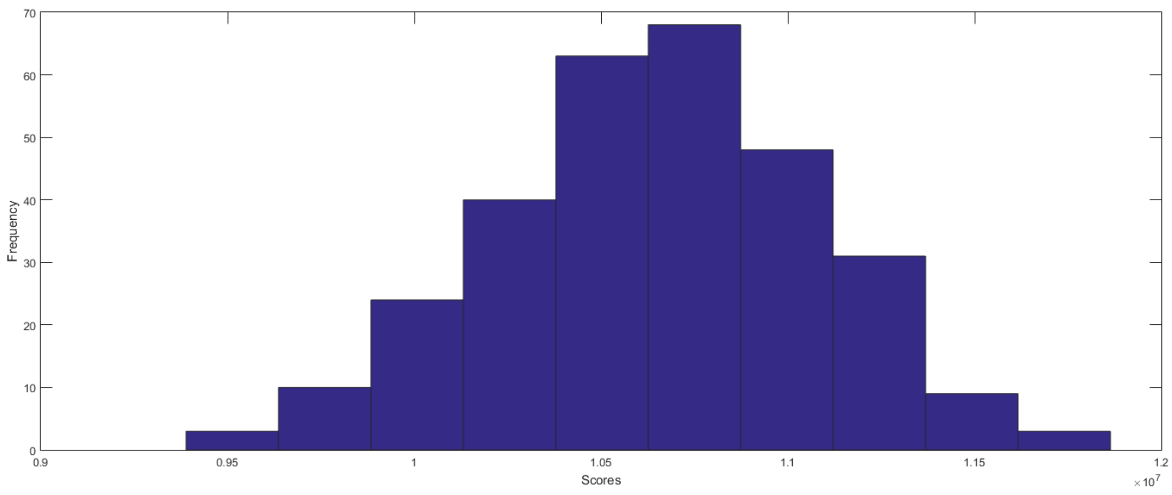

Figure 23: Bar graph of the scores using the minimization starting solution

The average scores for these algorithms are given in the following table:

Algorithm	*Average Score*
Random + Combination	9.1708e+06
Minimization + Combination	9.4792e+06

Table 3: Table with all average scores for different algorithms

As can be seen above the minimization + combination algorithm yields on average considerable higher scores than using a random solution as starting solution. Approximating the distributions gives us the following distributions:
$Random + Combination \sim N(1.06162e + 07, 425980)$
$Minimization + Combination \sim N(1.06479e + 07, 437325)$
Although the mean of the Minimization + Combination algorithm is higher, so is the standard deviation. However, the difference in standard deviations (11345) is a lot smaller than the difference in means (31700). Thus, the minor increase in uncertainty does not weigh up against the larger increase in means. The conclusion being that that the Minimization + Combination algorithm is preferable over the Random + Combination algorithm.

1.6 Exhaustive Search vs MinMax algorithm

By bringing the problem back to the discrete form (from a continuous one), the state space is no longer infinite. Thus, we can use exhaustive search to find the solution which is most likely to be the highest obtainable value (given the starting solution). This way we can use the scores of the exhaustive search as a benchmark for our best algorithm (minmax algorithm). Below are solutions (incl. calculated total value) using exhaustive search shown for each version of the problem:

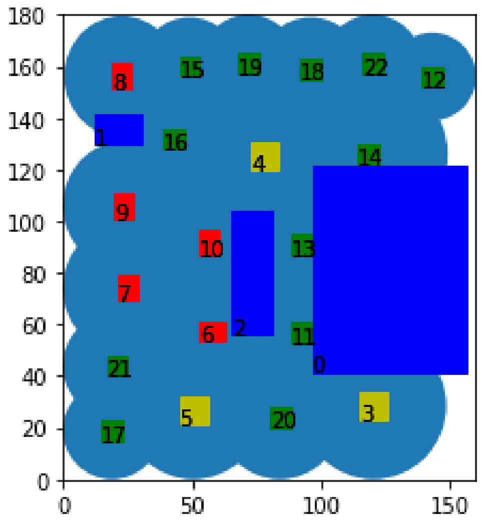

Figure 24: Visualization of final solution for 20 houses with score = 12,529,731

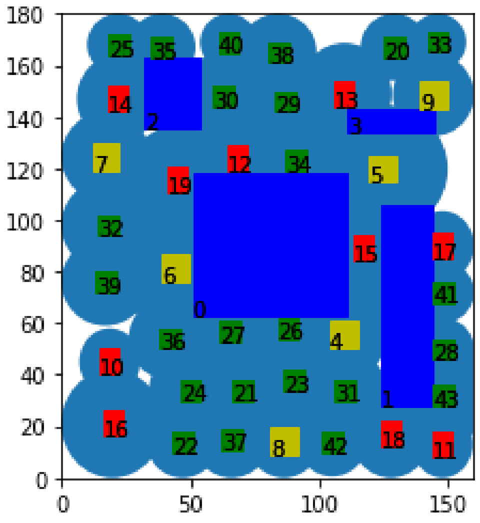

Figure 25: Visualization of final solution for 40 houses with score = 20,035,169

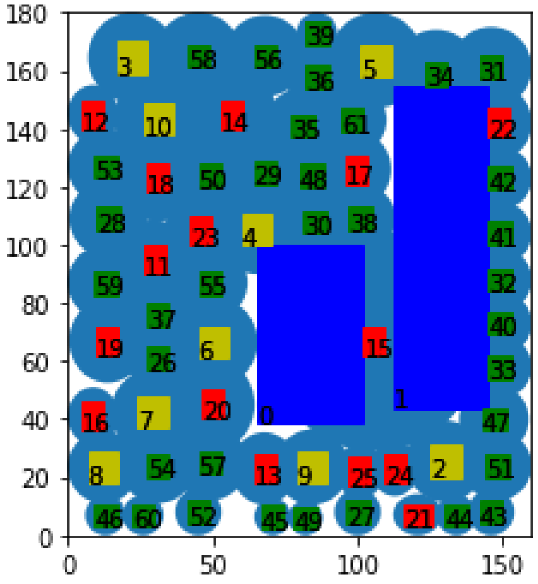

Figure 26: Visualization of final solution for 60 houses with score = 26,501,981

When comparing these solutions with the algorithm which yields on average final solutions with the highest value (minmax algorithm, described last subsection), we get the following results:

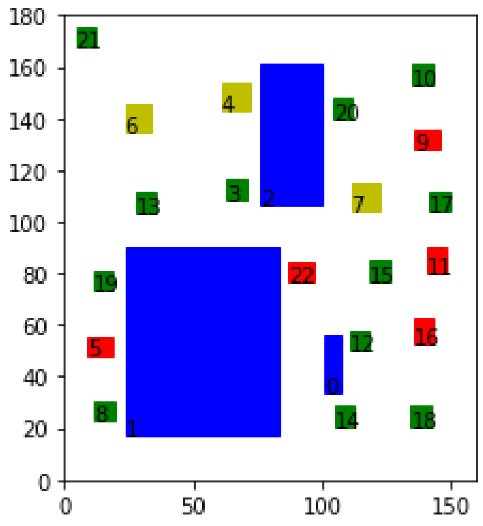

Figure 27: Visualization of final solution for 20 houses with score = 12,014,551

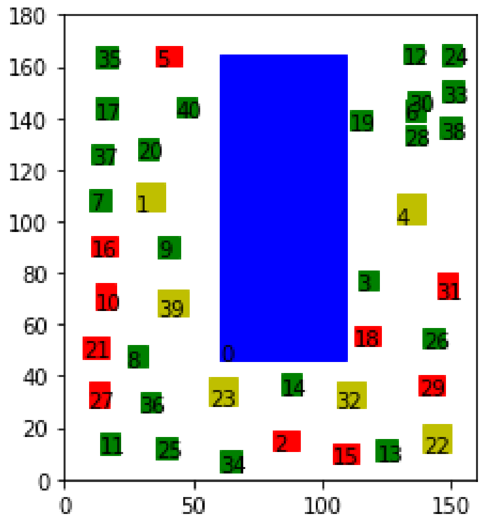

Figure 28: Visualization of final solution for 40 houses with score = 19,739,432

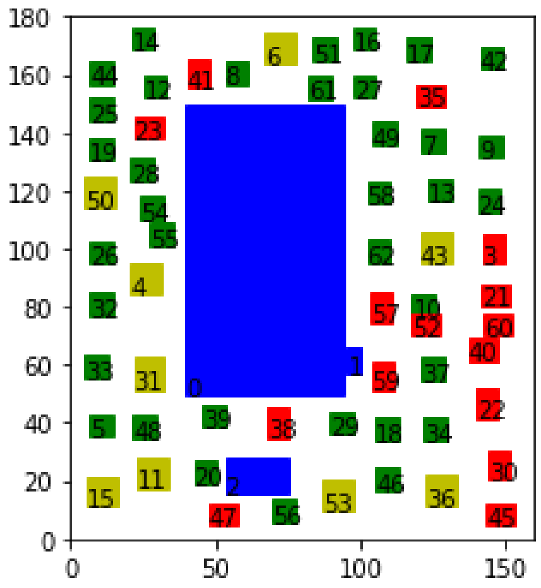

Figure 29: Visualization of final solution for 60 houses with score = 25,945,992

As can be seen above the minmax algorithm comes pretty close to the exhaustive search, while only taking a fraction of the time (2-5 min.) it takes the exhaustive search (1-6 hours) to find its final solution.

NOTE: In the 40 visualisation, it seems like houses 30 and 6 are in each other. This is not the case. The coordinates located in the matrix show clearly there is the minimum free space required between them. This is a error in the visualisation.