# Virtual Neanderthals : a study in agent-based modelling Late Pleistocene hominins in western Europe

Scherjon, F.

**Author**: Scherjon, F.
**Title**: Virtual Neanderthals : a study in agent-based modelling Late Pleistocene hominins in western Europe
**Issue Date**: 2019-05-28

# PART THREE: SETTING THE STAGE

*"On two occasions I have been asked, - 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' In one case a member of the Upper, and in the other a member of the Lower House put this question. I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."*

(Babbage 1864)

# 7. OVERVIEW, DESIGN CONCEPTS AND DETAILS (ODD) OF HOMININSPACE

## 7.1 Introduction

This chapter provides elements of the design of the HomininSpace modelling system and follows the Overview, Design concepts and Details protocol (ODD) (Grimm *et al.* 2006). The ODD was introduced in an effort to standardize the publication and documentation of Agent Based Models (ABMs) and to allow duplication of simulation runs and replication of simulation results (Antón *et al.* 2014; Grimm *et al.* 2006; Müller *et al.* 2014). The open access repository for agent-based models (OpenABM, Janssen *et al.* 2008) includes the ODD protocol as a requirement in their Model Review process[28]. HomininSpace was documented in such a manner that it passed this review process allowing incorporation of HomininSpace into the OpenABM repository allow open access to code and facilitating replication and re-use by other researchers.

This chapter includes several sections selectively adapted from the HomininSpace ODD document, which is in original form included in the Supplementary Materials. The ODD protocol prescribes several elements including purpose of the model, the process overview and scheduling (Section 7.2), the simulation period and stochasticity in the model (Section 7.3), and the model parameters (Section 7.4). The simulation results will be compared against archaeological data to assess the validity of the implemented model (Section 7.5). To enable autonomous modification of model parameter values a Genetic Algorithm has been implemented (Section 7.6 and Section 7.7). These last two sections are largely based on a previously published paper (Scherjon 2016). The model with documentation and simulation results is available online from the OpenABM website[29].

## 7.2 Process overview and scheduling

A simulation in HomininSpace always starts with an initialization phase (detailed in the flowchart in Figure 25). The system is initialized by reading configuration and data files, either from file or as entered by the user in the User Interface (UI). The configuration consists of a simulation number, a random seed, a set of parameter values and the model settings that define what model elements participate in this simulation. Data files contain

[28] See https://www.openabm.org/page/model-review-overview, accessed 27 October 2014.
[29] See https://www.openabm.org/model/5294/version/1/view, accessed December 2017.

grid data, climate data for modern day and past environments, and archaeological data. The simulation grid is created and filled with the environment information. Factories, single grid cells that represent population core areas where new hominin groups can be produced, are created if included in the model. Data for the reconstructed global sea levels and yearly mean temperatures for the simulation period are read from file and stored in memory.

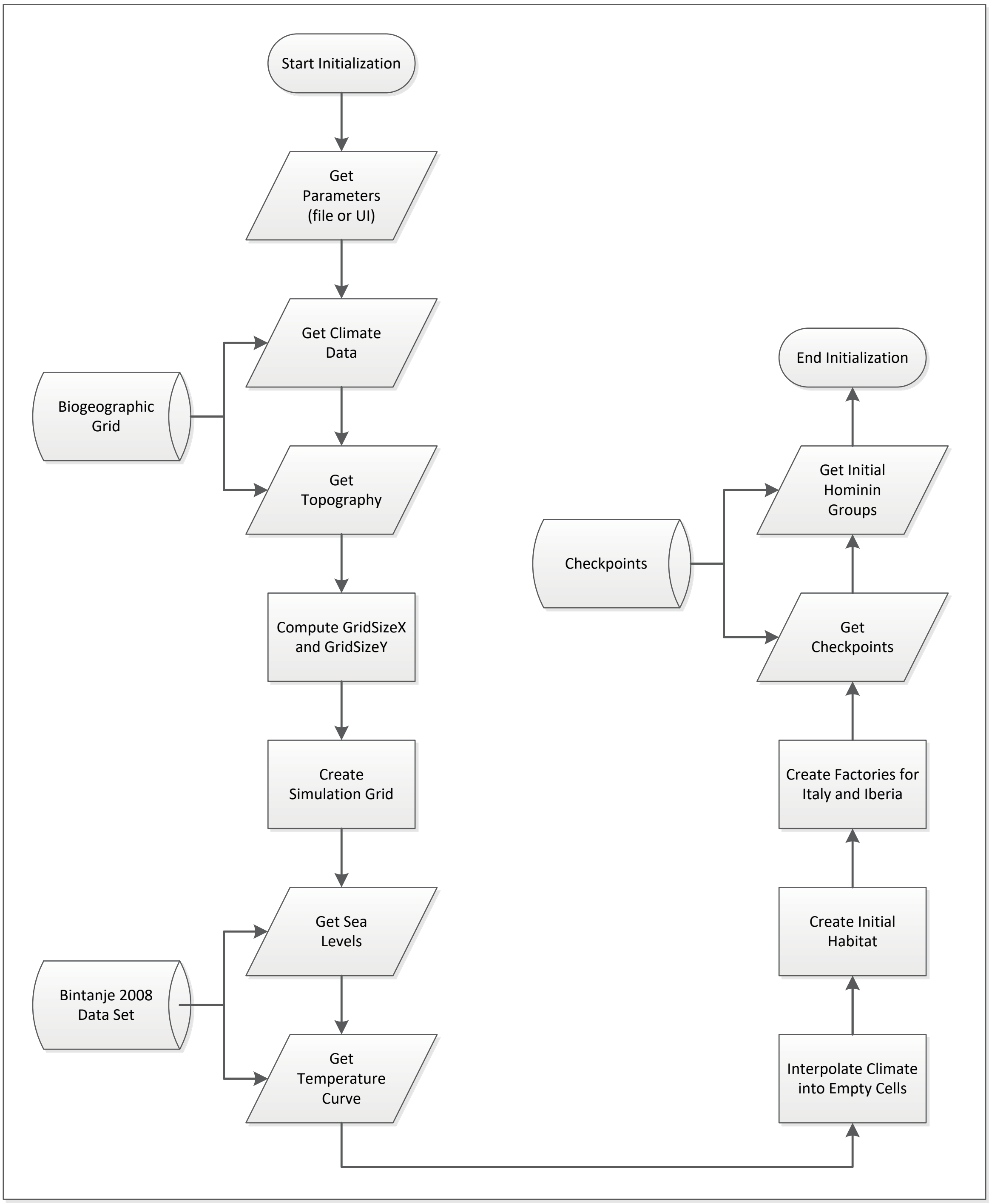


**Figure 25: Flowchart presenting the Initialization process in HomininSpace.**

After initialization of the system the main simulation run routine is executed. Within this loop the simulation executes the model, time step after time step. The sequence of events for a time step is given in Figure 26. The first action in each time step is updating the environment with new climatic data and topographical information (incorporating sea level changes). Ultimately, the edible biomass is calculated for each grid cell.

After updating all grid cells each hominin group that can move will scan the environment delimited by two times the foraging range (the radius leap-frog pattern as described by Binford (1982)). The location which is most attractive to the group because it has the highest energy levels becomes the destination for this group. Upon moving to this new location the foraging range for this group is recalculated, and presence is registered with the grid cells and resources are claimed. The groups will then consume the resources they require (if available) and the hominins in the group will either produce offspring or they will die, depending on the obtained resources.

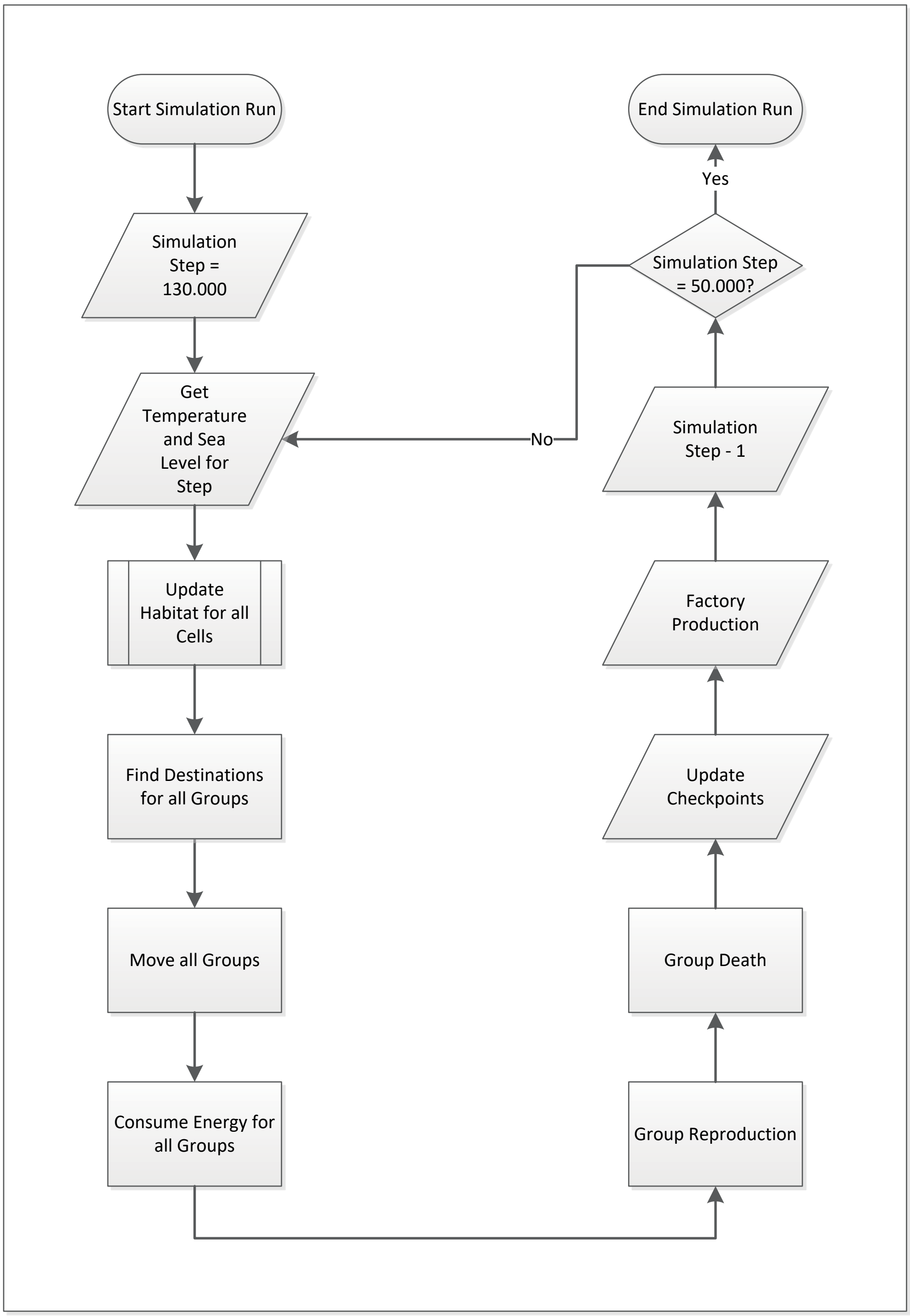

Start Simulation Run
Simulation Step = 130.000
Get Temperature and Sea Level for Step
Update Habitat for all Cells
Find Destinations for all Groups
Move all Groups
Consume Energy for all Groups
Group Reproduction
Group Death
Update Checkpoints
Factory Production
Simulation Step - 1
Simulation Step = 50.000?
No
Yes
End Simulation Run

**Figure 26: Flowchart of the Simulation process.**

## 7.3 Simulation time and stochasticity

Each simulation time step is defined to be one year. One year is the most natural complete set of environmental events, including a full seasonal cycle that includes summer and winter and dry and wet periods. It is also a unit that is readily recognized in the geological and biological records allowing reconstructions of environmental variables to some extent. Each simulation has a starting time expressed in years ago and a number of time steps (years) to execute. In the experiments performed for this research all simulations start at

131 ka and end at 50 ka. That means that every simulation run has 81,000 time steps. In each time step the following actions are executed in this order:

1. Increment timer, get new global sea level and global mean temperature;
2. Update all habitat cells by recalculating available energy;
3. Move all hominin groups and update foraging ranges;
4. For all groups: consume required energy, grow if there is sufficient energy, check for merge and split possibilities, perform reproduction and then check for death conditions;
5. Check production conditions for each factory (if present). If favourable, produce one new group per factory.

One of the aims for HomininSpace is to minimize the influence of random events and to create a deterministic simulation system. One place in the program where the influence of randomness is still present is in the movement and consumption order for the hominin groups. Each time step the complete list of groups is randomly ordered and then traversed sequentially with each group moving to its preferred area. Other locations where random values play a role is where a random number between -1 and 1 is added to the energy level of each HabitatCell to prevent a biased choice between equal destinations due to preferential system list ordering. Note that energy levels are computed in integers. This added random value prevents groups always selecting for instance the left path when presented equal route choices. The random seed, an integer value driving the random number generator, is included as an identifying element for the simulations. Each time a simulation is run with the same parameter values and the same random seed value, identical results are produced.

Parameter values for the Standard parameter set were generated randomly, specifically without bias (that means no normal distribution for any of the values). Selection of the parameter values for improvement in the subsequent parameter value space exploration is random, but organized in tournaments (see Section 7.6) to reduce the random character in the final distribution of parameter values.

## 7.4 Model parameters and the `ViabilityIndex`

In the preceding chapters the building blocks for the underlying model in HomininSpace have been identified. Some of the variables of the model have been marked as parameter, indicating that these can be changed. Table 13 lists and describes all 16 available parameters. There are numerous other model elements but these cannot be simply altered by the user, since they are defined as constants or otherwise implemented in the source code. Parameters were selected as candidates for variation because they are disputed in the literature (birth rates, energy requirements, temperature tolerances, foraging range), because they are directly related to other modifiable parameters (mortality rates) or because they were found important in agent dynamics and interaction (cohort sizes, number of years before group maturity). A simulation is uniquely defined by the combination of the parameter values and the seed value for the random number generator.

**Table 13: The 16 parameters that can be changed in the HomininSpace model.**

| *Name* | *Description* |
|---|---|
| ***1. BirthRate*** | The number of females that conceive this year (a value of 33% means one child per three females, or one child every three years for a female). |
| ***2. DeathRate_PreFertileCohort*** | Death rate for the pre-fertile segment per group, the percentage that does not survive. |
| ***3. DeathRate_FertileCohort*** | Death rate for the fertile segment per group, the percentage that does not survive. |
| ***4. DeathRate_PostFertileCohort*** | Death rate for the post-fertile segment, the percentage that does not survive. |
| ***5. Subsistence_PreFertileCohort*** | Energy needs for an individual in the pre-fertile segment of a group in kcal. |
| ***6. Subsistence_FertileCohort*** | Energy needs for an individual in the fertile segment of a group in kcal. |
| ***7. Subsistence_PostFertileCohort*** | Energy needs for an individual in the post-fertile segment of a group in kcal. |
| ***8. Years_Before_Group_Maturity*** | Period in years before a newly created group can interact with other groups (join) or settle. |
| ***9. GroupSize_BeforeMerge*** | Groups can merge with other groups if their total size is smaller than this value. |
| ***10. GroupSizeFertile_BeforeMerge*** | Groups can merge with other groups if their size of the fertile segment is smaller than this value. |
| ***11. GroupSize_BeforeSplit*** | A group can procreate by splitting in two, thereby generating a new group. This value is the minimum size before a group can split. |
| ***12. Temperature_Tolerance*** | Minimum temperature that can be sustained by the hominins. If the coldest temp in a year falls below this value the group of hominins dies. |
| ***13. CohortSize_PreFertile*** | The size of the pre-fertile segment in a group, with age boundaries in years. |
| ***14. CohortSize_Fertile*** | The size of the fertile segment in a group in years[30]. |
| ***15. Calories_Per_Kg_Meat*** | The number of kilocalories that can be extracted from one kilogram of meat. |
| ***16. Max_ForagingRange*** | Maximum annual foraging range from the current location. Groups cannot forage outside this range (in grid cells). |

The parameters are used in a demographic model that computes new population numbers based on the given values. The `ViabilityIndex` is the theoretical number of individuals remaining when a default hominin group (size 25) is followed for 100 years, given birth- and death rate values. For instance, for parameter value set 5 (see Table 21)

[30] Note that the length of the post-fertile segment is not limited, but chances of survival decrease progressively with age.

the `ViabilityIndex` = 548, meaning that the population of 25 individuals has grown to 548 after 100 years. The algorithm to calculate this number is given in Figure 27. The `ViabilityIndex` is used in the analysis of the simulation results and gives a rough indication of the viability of a given population without taking resource availability into account. It is passed as a parameter for presentation purposes only and not included in the model.

```
# calculate viability index by simulating 100 years and counting survivors
 sizePreFertile = 8
 sizeFertile = 12
 sizePostFertile = 5

 # check if viable population by running for 100 years, assuming sufficient
   resources
 for (year in 1:100 ) {

   # size of age category
   becomingFertile = sizePreFertile / sSizePreFertileSegment
   # size of fertile age category
   becomingPostFertile =  sizeFertile / sSizeFertileSegment;
   newBorns =  (sizeFertile / 2.0) * (sBirthRate / 100.0);

   dPre = sizePreFertile * (sDeathRate_PreFertile/100.0);
   dFertile = sizeFertile * (sDeathRate_Fertile/100.0);
   dPost = sizePostFertile * (sDeathRate_PostFertile/100.0);

   sizePreFertile =  sizePreFertile + newBorns;
   sizePreFertile = sizePreFertile  - becomingFertile;
   sizeFertile = sizeFertile  + becomingFertile;
   sizeFertile = sizeFertile  - becomingPostFertile;
   sizePostFertile = sizePostFertile  + becomingPostFertile;
   sizePreFertile = sizePreFertile  - dPre;
   sizeFertile = sizeFertile  - dFertile;
   sizePostFertile = sizePostFertile  - dPost;

 }

 totalPopulationSize = sizePreFertile + sizeFertile + sizePostFertile
 cat (",")
 # there is always a fraction left, even if unviable population
 cat (floor(totalPopulationSize))
```

**Figure 27: Calculating the ViabilityIndex for a group with 25 individuals. Code fragment is taken from the “CreateHSParameters.R” source file.**

## 7.5 How to match the archaeology

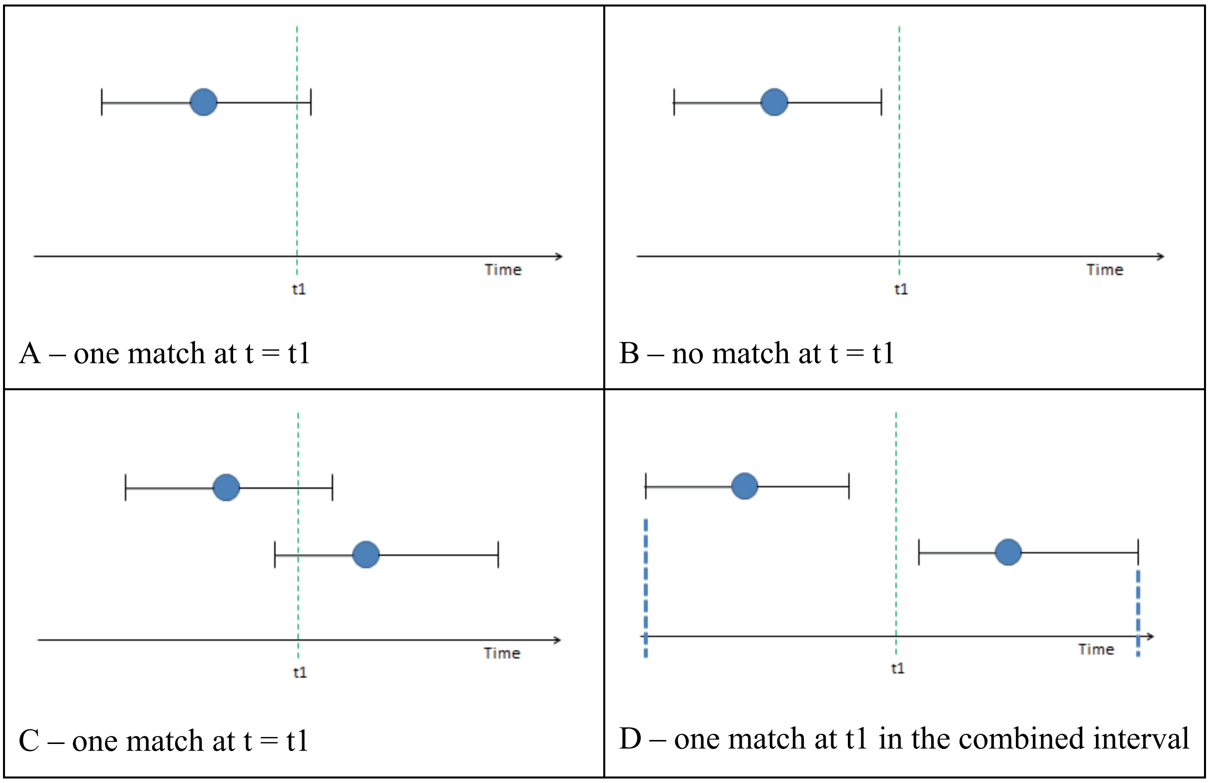
HomininSpace uses an extensive collection of archaeological presence data: the CSTs with associated intervals. Simulations are scored against this archaeological data set. When comparing simulation results against the archaeological attested presence it is essential to assume that this record is representative for the actual presence in the past (Wheatley 2004). When using presence only data those areas where nothing has been found (or where nobody has searched) cannot influence the results. The effect of using presence-only data is clearly illustrated with the theoretical example of the omni-present hominin that will automatically be the best matching Neanderthal. To avoid this one can introduce *absence areas*. These are periods in time and space where the model states that hominins were *not* present, a fact that can be used in model validation (Jiménez-Valverde and Lobo 2007) (see Subsection 9.4.1). Unfortunately, only limited absence data on Neanderthals is available. On the other hand, when the simulation results match the archaeology, it fits the data regardless of how this data was collected. It then becomes important to define what constitutes a 'fit' with the archaeological data.

### 7.5.1 Translating simulation results into numbers: `MatchedIntervalCoverage`

This study assumes that an archaeological site that presents evidence for the presence of Neanderthals (for instance stone tools) is prove of the past presence of at least one Neanderthal group (of whatever size). Since a single stone knapping individual can produce hundreds of stone tools, it is very difficult to assess the number of presence events that created any archaeological assembly (see for instance the supplementary information in Kolodny and Feldman (2017)). In HomininSpace a database is constructed of presence intervals constructed using radiometric dates with one standard deviation (SD). Each interval is from a date minus one SD to that date plus one SD. If intervals overlap, or if two intervals are from the same layer they are combined into one interval using the minimum and maximum values. Each checkpoint has one or more intervals.

During the simulations, modelled presence of agents is compared against these intervals to compute the score or *precision* of each simulation (Crooks *et al.* 2015). If an agent visits a checkpoint (either by foraging in the same grid cell or by making that grid cell its home range) the simulation time step is compared against the interval(s) for that location. If that specific time step falls inside an interval, one point is added to the score for that interval. This is done for each time step that any group visits the checkpoint. A time step can be scored only once (if multiple groups forage in the same grid cell, this still count as one archaeology producing visit). If the time step does not fall inside an interval this is also registered, but it does not add to the simulation score.

The score for a single interval is the number of visits divided by the length of the interval. Thus the maximum score for each interval is 1, when visited by as many visits as the length of the interval in years. HomininSpace thus assumes that more visits to the same location in a single interval make for a better match. HomininSpace does not interpret the interval structure: if two different archaeological layers have overlapping dating intervals, there are two intervals to be matched (if these two intervals are however in one layer they are combined into one interval). The scoring procedure is illustrated in Figure 28 and Figure 29. With a more precise date (smaller sigma) the maximum score is easier to achieve.



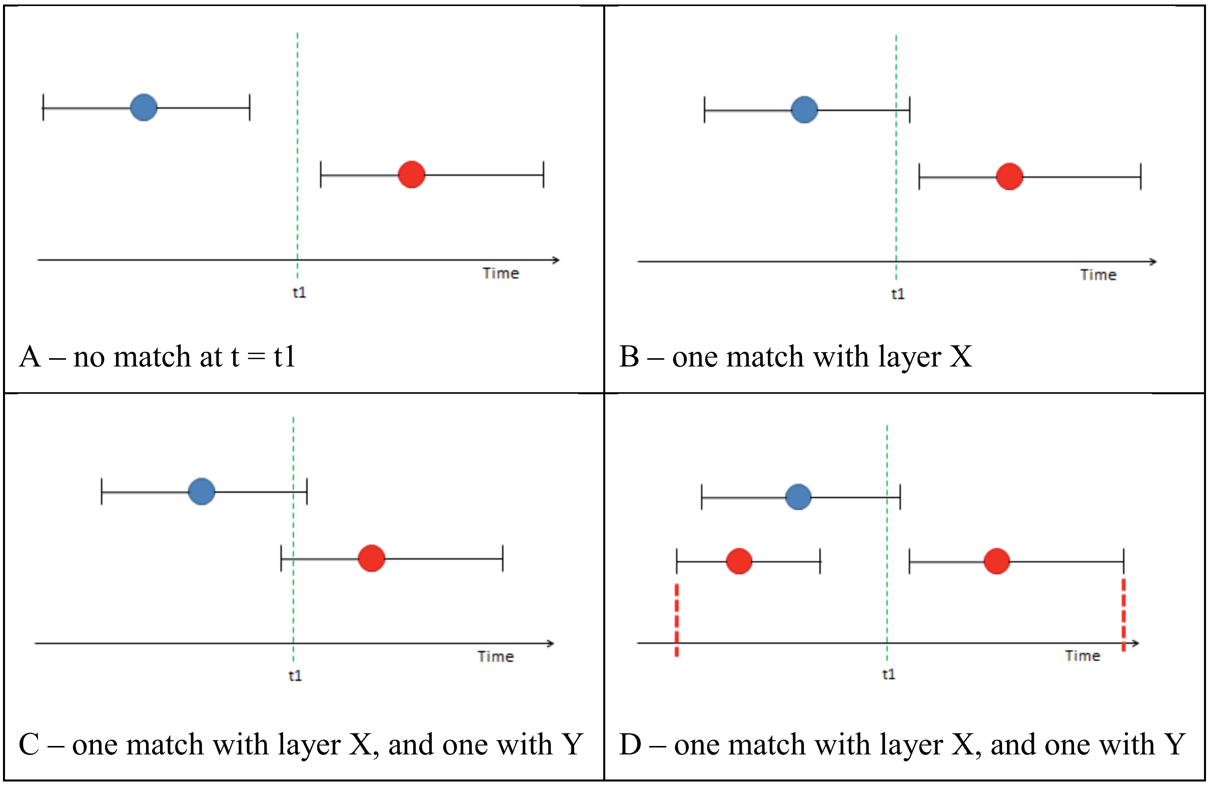
A – one match at t = t1

B – no match at t = t1

C – one match at t = t1

D – one match at t1 in the combined interval

**Figure 28: Counting matching visits. A visit at simulation time *t1* matches a dated interval for a single archaeological layer (A, C and D). Only in B there is no match.**



A – no match at t = t1

B – one match with layer X

C – one match with layer X, and one with Y

D – one match with layer X, and one with Y

**Figure 29: Counting matching visits. A visit at time *t1* can match two dated intervals that represent two archaeological layers (blue dots for layer X, red for layer Y).**

In HomininSpace a simulated visit at a checkpoint in a time step does not mean hominins were there only one time. It means that within one year a group of hominins visited the site at least once and could have left archaeology to be dated as part of the archaeological record of the site. Each time step in the simulation represents one year, and thus a single group can visit a checkpoint only once per time step. Other groups that live in the area can visit the checkpoint in the same time step, but only one visit per time step is actually counted. A checkpoint can be re-visited in next time steps. It is assumed that one or more visits can create the archaeological layer as it is identified by the archaeologists, but that more visits give a better chance of producing the archaeology. That means that a higher score implies a better model, or more plausible parameter values.

The fitness function in the Genetic Algorithm (the next subsection) that finds new parameter value combinations requires a single number that indicates how successful a simulation is: the *simulation score*. This number is used to compare one simulation against another. A larger number suggest a better match with the archaeology and thus a better model. To compute the total simulation score an interval score is calculated for each interval. This is the actual number of visits divided by the length of the interval. The division by the length ensures that intervals that are more accurate and thus shorter have more weight in the total simulation score. The simulation score is calculated by adding all interval scores together for all checkpoints (Equation 9). This number is referred to as `MatchedIntervalCoverage`. The maximum possible value or theoretical maximum of this variable in the configuration as used for the simulations in this research is a score of 39,200. In this thesis any general reference to simulation score or simulation results refers to this number.

$$\boldsymbol{MatchedIntervalCoverage} = \sum_{k=1}^{CSTs} \sum_{n=0}^{intervals} \frac{\boldsymbol{visitCount}}{\boldsymbol{length}}$$

**Equation 9: Combining interval scores for all checkpoints to compute the simulation score.**

## 7.6 Genetic Algorithms - searching through N-dimensions

Optimization of agent-based models is a complex problem. Computation of a solution for the model involves the execution of a simulation in all but the most trivial cases. The search for optimal solutions is therefore computationally expensive, especially with an extensive environment and when the number of parameter combinations is large. An exhaustive algorithm like a Monte-Carlo search implementation would require too many computational resources to efficiently find optimal solutions. Instead this research applies Genetic Algorithm (GA) based search and optimization techniques in an attempt to characterize Neanderthal models that match the archaeology well while exploring the parameter space (Grimm and Railsback 2005; Revay and Cioffi-Revilla 2018).

In a typical GA, a population of potential solutions is evolved in the solution space towards higher values of a fitness function (Forrest 1993). Genetic Algorithms form, together with Evolutionary Strategies and Genetic Programming, the Evolutionary Algorithm programming paradigm (Coello Coello 2002). Differences between these techniques are fading (Michalewicz 1992, 132), and while the chosen terminology no longer reflects our current understanding of genetics, they all aim to simulate an evolutionary process in the computer (Coello Coello 2002). These are nature-inspired numerical optimization techniques that operate in a virtual laboratory. Basic elements in evolutionary algorithms are a population of individuals to work with, a string with values which can be manipulated that define individuals (referred to as the genes or the chromosome of that individual), a fitness function that calculates how well adapted an individual is within the modelled environment (Michalewicz 1992), and an optimization technique targeting optimal solutions by modifying individuals. The values for model parameters are adjusted until experimental optima are reached with no substantial improvement in the simulation results for any newly generated models (Calvez and Hutzler 2006). Compared to random search methods GA implement a stochastic search with directed randomness.

The term Genetic Algorithm was coined by John H. Holland (1975) but the general principle was already recognized by Alan Mathison Turing in the 1948 essay 'Intelligent Machinery': '*There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value'* (Turing 1948, 16). The idea is that an individual represents a single solution to the problem at hand; that individuals vary from one to the other; that an evaluation function represents the environment in calculating the fitness of each individual; and that the computer searches for the individual with the highest fitness value (the optimal solution) in the problem space, without actually programming that solution, but instead by manipulating the genes of the individuals mimicking the processes of variation and natural selection (Coello Coello 2002). The technique is often referred to as meta-heuristic due to its domain independence.

GAs are very well suited for exploring nonlinear search spaces, with potentially multiple locations present that can yield good solutions, for instance in social agent-based modelling studies (Revay and Cioffi-Revilla 2018). A non-traditional methodology is required when traversing the larger resulting parameter spaces searching for optimal solutions, where exhaustive search algorithms are replaced by heuristic procedures. A GA differs from more traditional optimization and search algorithms in at least four ways (Goldberg 1989):

- A GA searches in a population of points, not just in a single point;
- GAs operate with probabilistic, unbiased transition rules, not deterministic ones;
- A GA operates on a set of parameter values, not individual parameters themselves, and does not require any explicit knowledge of the actual structure of the solution space (what these parameters are about);
- A direct, explicit fitness function is used.

In the structure of a GA, a probabilistic selection is made from the population based on some measurement of the individual's adaptation to the environment or its fitness within it. The design of the fitness function is at the discretion of the modeler. After the selection of one or more individuals, operators are applied on the selection. These operators are inspired by the early perception of how the genomes of offspring were created in nature. They include Mutation, where random changes are made to the genes of an individual, and Crossover (a special kind of Recombination), where the genes of two individuals are mixed in sequence. Then the fitness function is calculated for the generated offspring, and the new individuals are inserted into the population, generally replacing other individuals. The process is then restarted, until some stop criteria are met. Generally the stop criterion is a certain value for the fitness function being attained, or no measurable improvement of the calculated fitness values after a certain number of generations.

An Agent-based Model (ABM) is a complex system aiming to reproduce the dynamics of the real world, and generally one that cannot be solved mathematically. Therefore GAs form a perfect exploration and tuning method for such models (Calvez and Hutzler 2006). Emerging properties, high parameter sensitivity, and non-linear solutions often characterize the solution space for such models. There is no analytical description, and simulation results cannot easily be reduced to the input data. There is development and emergence in a non-trivial way. As such they are well suited to modelling complex systems and to explore this complexity. Archaeological research questions connected to such models often incorporate the resultant output of the (inter)actions of many individuals through time, and as such can be targeted by ABM techniques (Lake 2014). However, there have been few attempts to apply GAs to archaeological simulations, although they are implicitly used in the more widely applied Genetic Algorithm for Rule-Set Prediction tool (GARP) for biogeographical niche modelling (Banks, d'Errico, Peterson, Vanhaeren, *et al.* 2008).

Every simulation in HomininSpace is characterized by a combination of parameter values and the random seed. This combination of parameter values is referred to as the *chromosome* for that individual simulation. In HomininSpace, a GA is then used to search for the parameter value combination that when simulated optimally matches the archaeology. First, an initial solution population is constructed by varying parameter values randomly, and then running all the simulations with these parameter sets. Subsequently, simulation results are used to select promising parameter combinations, generate new individuals (new parameter combinations), and running simulations for that evolved offspring in a search for a better solution.

## 7.7 Traversing the parameter space

Values chosen for the parameters will decide how well the simulation matches the archaeological presence data, and thus score the simulation results into the `matchedIntervalCoverage` variable. Maximizing this variable is one of the aims of this study. All the value combinations for all parameters form a 16-dimensional parameter space, with each parameter adding one dimension in which variation is possible. The search traverses this parameter space. Previous research has shown that human insight and analysis falls short in identifying the best matches, and some form of machine learning is needed where no explicit instructions nor biases steer the system (Scherjon 2015a).

### 7.7.1 Generating the Standard parameter value set

The *Standard* parameter value set is a collection of 1500 generated parameter value combinations which are used as the starting point for every exploration. Simulations for these 1500 parameter combinations are executed first, after which a genetic algorithm will attempt to optimize individuals from this set. The parameter values for the Standard set have been generated randomly, with values taken indiscriminately from an interval defined by minimum and maximum values (see Table 14). I specifically use a random probability distribution for each parameter (Iman and Conover 1980) and do not construct a (optimized) Latin Hypercube sample with equal distances or attempt any other structuring of the search space (Bates *et al.* 2003). Instead I use the genetic algorithm for the optimization of the sampling (Liefvendahl and Stocki 2006).

Each value from this interval has an equal chance of being selected (specifically *not* implementing a normal distribution of values for parameters like birth or mortality rates). The generation of this Standard set is done with an R script named "CreateHSParameters.R" and is referred to as spanning the parameter space. The minimum and maximum possible values are constructed as a wide interval around the default value, which has been taken from the literature (Scherjon 2015a). This avoids the (researcher) bias in selecting values for these parameters. For example, 1500 times a random value is selected for the BirthRate parameter from the interval [1..50], one for each parameter value combination in the Standard set.

**Table 14: Minimum, maximum, step size, and default values for the Standard parameter set.**

| *Name* | *Minimum* | *Maximum* | *Step* | *Default* |
|---|---|---|---|---|
| ***1. BirthRate*** | 1 | 50 | 1 | 33 |
| ***2. DeathRate_PreFertileCohort*** | 1 | 15 | 1 | 4 |
| ***3. DeathRate_FertileCohort*** | 1 | 15 | 1 | 2 |
| ***4. DeathRate_PostFertileCohort*** | 1 | 15 | 1 | 8 |
| ***5. Subsistence_PreFertileCohort*** | 2000 | 5000 | 250 | 3000 |
| ***6. Subsistence_FertileCohort*** | 2000 | 5000 | 250 | 4000 |
| ***7. Subsistence_PostFertileCohort*** | 2000 | 5000 | 250 | 3500 |
| ***8. Years_Before_Group_Maturity*** | 1 | 10 | 1 | 5 |
| ***9. GroupSize_BeforeMerge*** | 1 | 5 | 1 | - |
| ***10. GroupSizeFertile_BeforeMerge*** | 1 | 5 | 1 | - |
| ***11. GroupSize_BeforeSplit*** | 6 | 150 | 10 | - |
| ***12. Temperature_Tolerance*** | -30 | -5 | 1 | -18 |
| ***13. CohortSize_PreFertile*** | 8 | 20 | 1 | 15 |
| ***14. CohortSize_Fertile*** | 15 | 40 | 1 | 25 |
| ***15. Calories_Per_Kg_Meat*** | 2500 | 3500 | 50 | 3000 |
| ***16. Max_ForagingRange*** | 1 | 15 | 1 | 15 |

### 7.7.2 Searching the highest scoring parameter value set with Genetic Algorithms

The parameters that are used in the model underlying the HomininSpace simulation system are described in Table 13. Parameter values are based on ethnographic data with wide plausible extended value ranges from which values are randomly selected. Systematical exploration of the total parameter space employing these 16 parameters would require an exponential number of simulations. For example, if one takes only three values per parameter (a minimum, a maximum and a value in between), this would require the simulation of a 3^16 parameter value sets (or almost 300 million unique simulations). Therefore the combinatorial explosion with so many parameters requires a non-exhaustive exploration. Nevertheless, due to the non-linear character of the solution space, a systematic exploration of the parameter space is needed (Calvez and Hutzler 2006). A Genetic Algorithm (GA) implements such a search.

The implemented GA in HomininSpace selects individuals from the population of solutions using a tournament selection procedure (Miller and Goldberg 1995). For the selection of each individual, a selection round or tournament is organized in which *n* individuals are randomly chosen from the total population. From this subset the highest scoring (most optimal) solution is declared winner. This procedure ensures that even mediocre solutions can (occasionally) produce offspring and prevents swamping of the population by the offspring of a few extraordinarily good performing individuals.

The selected individuals participate in the creation of the next generation new individuals through the application of several operators. HomininSpace implements a integer-coded GA where each individual is represented by a string of 16 integer values (the chromosome), one integer for each model parameter (Herrera *et al.* 1998). The operators that are applied are *mutation* and *crossover*. Then additional simulations for the newly generated parameter combinations are executed, the fitness value computed, and the offspring is added to the general population. This process is repeated until the end of each experiment (Figure 30).

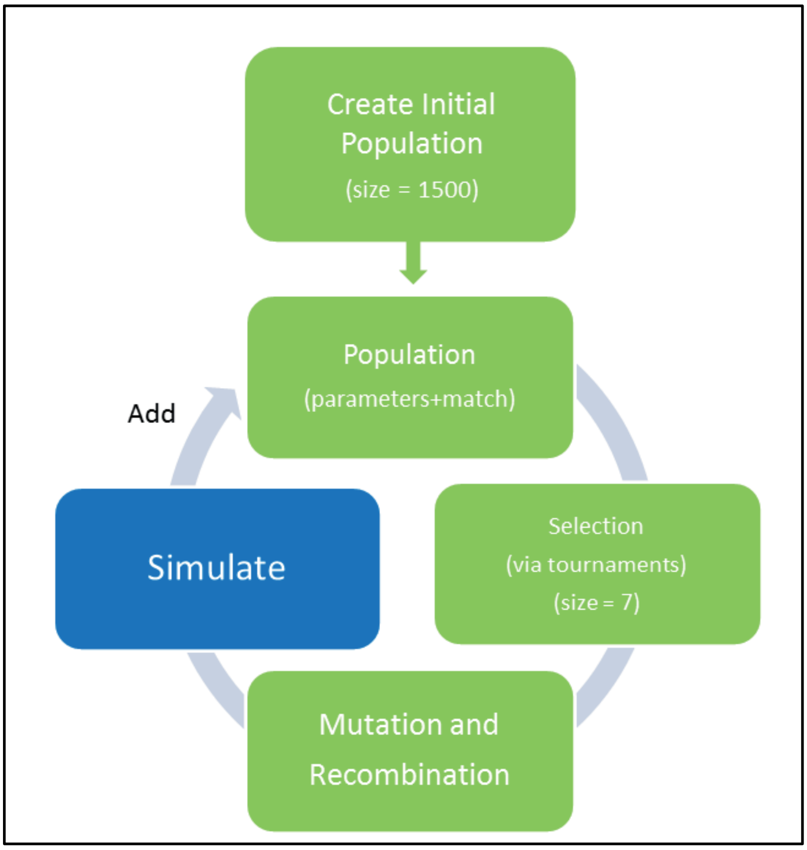


**Figure 30: Functional representation of the implemented GA.**

The mutation operator is implemented as the random modification of one single parameter value by plus or minus 10%. This implements a low rate mutation mechanism to increase coverage of the search space and to prevent convergence to a local optimum (Yao 1993). An additional advantage of the chosen mutation operator is that when needed, the search space can be expanded even beyond the original chosen random value domain borders. In other words, values can be created that are not present in the (initial) population (Aleksandra *et al.* 1997). Crossover aims to recombine two good parent solutions into potentially even better offspring solutions. Implemented is a uniform multi-point crossover, where the offspring is a single individual that is a stochastic mix of the parameter values from both parents (Syswerda 1989). This combines very well with tournament selection (Aleksandra *et al.* 1997, 7860). The algorithm ensures that both parents contribute the same amount of genetic data. Seven individuals (four crossover results, three point mutations) are created in each generation, because the hardware running the experiments has seven parallel processors available for executing simulations.

For HomininSpace, the generation size has been defined as seven new individuals for the very practical reason that in each computer that participated in the experiments seven computing cores were available for execution of the simulations. Thus seven simulations were run simultaneously. Since offspring of each generation is added to the population without replacement of parents or other less performing individuals, no lineage is terminated prematurely, and each individual competes until the very end of each experiment. This ensures that the best performing individuals survive intact through each generation. However, successful lineages still might tend to dominate the tournaments, swamping more successful but unfortunate individuals. This must be taken into account when reporting successful individuals and when defining the stop criterion of the algorithm.

Since the GA methodology cannot guarantee that the optimum solution has been found, the algorithm must have a stop criterion that allows a sufficient number of iterations before halting the search. Since swamping by individual lineages does not always occur (especially when many similar scoring individuals are present in the solution base) the stop criterion for HomininSpace has been defined as follows:

- A minimum of 500 new individuals must have been generated, that is more than one third of the original population size;
- If more than 100 new individuals have been generated without any improvement in the simulation score the search ends.