Discrete Event Simulations in Forest Technology

Linus Jundén

Master’s Thesis
Master of Science in Engineering Physics

Supervisor: Martin Servin
Examiner: Johanna Björklund

June, 2011

UMEÅ UNIVERSITY
FACULTY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF PHYSICS/UMIT RESEARCH LAB
SE-901 87 UMEÅ
SWEDEN
Abstract

Development of a tool for discrete event simulations in forest technology, dependent on spatial components, has successfully been initialized in this thesis project. These simulations may be used to optimize the way the forest is used and to evaluate new machine concepts in forestry.

The Python library for discrete event simulation, SimPy, was chosen as the foundation for the tool. The developed tool can handle spatial objects such as moving machines, trees and boulders. Support for continuous linear movements was also added, which has resulted in a model that partially overlaps continuous and discrete event simulations without any additional computational costs.

The result is simulations of a forest with machines operating in it. Two pilot simulations, one of a thinning machine and one of a planting machine, were performed with useful results.

The new simulation tool shows promising properties. Limitations and improvements are discussed, with the conclusion that continued development is recommended.

Keywords: Discrete Event Simulation, Forest, Spatial entities, SimPy, Python, Planting Machine, Thinning
Sammanfattning

I det här examensarbetet har ett verktyg för diskret event simuleringar inom skogsteknik utvecklats. Sådana simuleringar kan användas för att optimera arbete inom skogsbruksamt för att utvärdera nya maskintekniska koncept.

Programmeringsspråket Pythons bibliotek för diskret event simulering, SimPy, valdes som grund för verktyget. Det utvecklade verktyget kan hantera spatiala objekt såsom träd, stenar och stubbar. Stöd för kontinuerliga rörelser har även utvecklats, vilket resulterat i en modell som delvis överlappar kontinuerliga simuleringar och diskret event simuleringar.

Resultatet är kvalitativa simuleringar av skog och skogsmaskiner. Två lyckade simuleringsstudier har gjorts för Sveriges Lantbruksuniversitet, en simulering av en gallringsmaskin och en simulering av en planteringsmaskin.

Det nya simuleringsverktyget uppvisar lovande egenskaper, även om det har sina begränsningar. Fortsatt utveckling av verktyget rekommenderas starkt.

Nyckelord: Diskret Event Simuleringsverktyg, Skogsteknik, Spatiala Objekt, SimPy, Python, Planteringsmaskin, Gallring
## Contents

1 Introduction ........................................... 1
   1.1 Background ......................................... 1
   1.2 Previous work ....................................... 2
       1.2.1 International .................................. 2
       1.2.2 Sweden ......................................... 2
   1.3 Purpose ............................................... 2
   1.4 Objectives ........................................... 2
   1.5 Collaborators ....................................... 3
   1.6 Methods .............................................. 3
       1.6.1 Evaluation study of discrete event modeling and simulation .. 3
       1.6.2 Modeling framework .............................. 3
       1.6.3 Prototype simulation tool ......................... 4
   1.7 Test cases ............................................. 4
       1.7.1 Multi-armed tree planting machine .................. 4
       1.7.2 Continuous thinning with a new crane head ........... 4

2 Theory and Methods ..................................... 6
   2.1 Discrete Event Simulation ............................. 6
       2.1.1 Activity oriented paradigm ......................... 7
       2.1.2 Event oriented paradigm ........................... 8
       2.1.3 The process-oriented paradigm ..................... 8
   2.2 SimPy ................................................. 9
   2.3 Terrain ............................................... 10
       2.3.1 Stumps and roots .................................. 10
       2.3.2 Stones/Boulders ................................. 12
   2.4 Common algorithms ................................... 13
       2.4.1 Driver/Operator .................................. 13
       2.4.2 Roads ............................................ 14
       2.4.3 Cranes .......................................... 14
       2.4.4 Continuous movements in a discrete time frame .... 14
       2.4.5 Coordinate systems ................................ 15
   2.5 Planting Machine ..................................... 16
       2.5.1 Goal of the simulation ............................. 16
       2.5.2 Working pattern for planting device ............... 16
       2.5.3 Ideal planting coordinates ......................... 17
       2.5.4 Micro-site selection sequence ...................... 17
       2.5.5 Planting Sequence ................................ 18
List of Figures

1.1 CAD pictures of the test cases. .............................................. 5
2.1 Discrete event example. ......................................................... 7
2.2 The event scheduling scheme. .................................................. 8
2.3 An example of a generated stump with roots. .............................. 11
2.4 Cylindrical coordinate system for the planting machine. .................. 15
2.5 Moving pattern for the 2a planting machine, and the archimedian spiral used in the micro-site selection. ................................. 18
2.6 The variables used in the angle limits calculation. ......................... 23
2.7 Vehicle and thinning corridor information. ................................. 25
2.8 Structure of the program ......................................................... 29
3.1 A corridor thinning simulation. ............................................... 34
3.2 Planting machine results for varying automation and terrain. ............... 36
3.3 Varying terrain and crane limits ............................................. 37
3.4 Variation of the angle limits for the two-armed planting machine. ........... 38
3.5 Variation of the mounding blade width. .................................... 39
3.6 Varying automation configuration for the thinning machine. ............... 40
A.1 A bounding circle for a polygon. ............................................ 49
B.1 Varying inner crane radius and terrain .................................... 54
B.2 Planting stands ................................................................. 55
B.3 Thinning stands ............................................................... 56
List of Tables

2.1 Distribution variables for the root model. 11
2.2 Velocities for the crane movements 22
2.3 Automation configuration table for the planting machine 24
2.4 Terrain type parameters 24
2.5 Properties for the crane head 26
2.6 Automation table for the thinning machine 29

3.1 Automation configuration table for the planting machine 35
3.2 Automation configuration table for the thinning machine 40
List of Algorithms

1. Root splitting .................................................. 12
2. Working pattern for planting device. .............................. 17
3. Micro site selection sequence. .................................. 18
4. Mounding .......................................................... 20
5. Planting ............................................................. 21
6. Tree cutting approach, an example .............................. 26
7. Algorithm for corridor selection ................................. 29
8. Distance from point to line. ...................................... 49
9. Determining if two line segments intersect in 2D .......... 50
10. Intersecting line segment and sphere ......................... 51
11. 2D collision detection .......................................... 52
Chapter 1

Introduction

1.1 Background

The forest industry and researchers have for decades performed computer system analysis of different processes in the forest. This work has optimized interactions and working patterns in the field. In later years, simulation has also been used as a tool to evaluate new models and new ideas before real implementation. This procedure saves a lot of money and effort by discarding non-working ideas and favoring good ideas.

Although there is a background of simulation studies, no general approach has been taken for simulations in the forest industry, at least in Sweden. A picture emerges where one “reinvents the wheel” for each simulation, which affects the quality of the simulations in a negative way and results in a lot of extra work. Simplified and more accurate simulations could push the development of new machines and new concepts in the forest industry and optimize cost effectiveness of forest operations. Many parameters could be investigated. A few examples are size and composition of machine systems, the number of workers in a certain sector and environmental impact etc. The list is long and so are the potential savings.

The interactions of the machines and people in a forest is a really complex system. One cannot model this with traditional “box-edge-box” models (see theory section), which could be used for the later part of the production chain such as road transportation and sawmill processing. For an accurate model one has to consider the locations of the trees and machines.

With the current development in measurement and GIS fields, we may soon be in a situation where every single tree in a forest is recorded in a database with position, size, species etc. Overall one can see a trend of gathering more and more information about specific forests. For example the Swedish mapping, cadastral and land registration authority (Lantmäteriet) will in the coming years map all of Sweden with an elevation precision better than 50cm and a grid smaller than 2m. Having this information available hugely expands the potentials of optimization work in forestry based on spatial entities, something that is partly done today but not with a common platform.

This master’s thesis project explores the world of simulations in forest environments, making pilot simulations demanded from forestry R&D, investigating potentials and limits and inventing new concepts.
1.2 Previous work

1.2.1 International

The literature survey paints a picture of two strongholds that produces scientific articles about discrete event simulation in forestry. One of them is the University of Joensuu, Finland, where Asikainen in his PhD thesis identified the importance and potentials of discrete event simulation in the forest industry [2]. Asikainen et al. have since then produced a number of articles on the subject. This group uses the commercial simulation software WITNESS in their studies [3][16]. According to the performed software study (see section 3.1.1) the handling of spatial components are limited in these simulations.

The other group with a long list of publications is in the Technical University of Münich, where Ziesak et al. have produced several articles [24]. A lot of work has been put into the development of a virtual forest, with exact locations of thousand of trees. This environment has been used to put up a model in the software Automod for the simulations. This group uses spatial information about trees to make simulations of the production chain during harvest.

Apart from this, work has been done in Canada, USA and Italy to mention a few places. An overall impression is that the handling of spatial variables is not really mentioned, except by the Münich group. Typical questions that are answered with these simulations are if a certain new way of working is preferred, or if it gives rise to big queues. For example, the Joensuu group published an article in 2010 regarding the efficiency of a cable yarding system, called skyline system, in steep Norwegian terrain [3]. The study resulted in an optimal choice of skyline system, with an optimal number of trucks for the road transportation.

1.2.2 Sweden

Sweden has a tradition of forestry simulations, for example there is a unit at the University of Agricultural Science (SLU) in Umeå that does simulations, e.g. [8] and [4]. Umeå University also has some prior experiences of discrete event simulations in forestry [19]. The UMIT research lab with the offspring companies Algoryx and Oryx are engaged in similar studies but with a physics based simulation approach.

The Forest Research Institute of Sweden (skogforsk) has a logistics group that does a lot of optimization work connected to forestry, often in collaboration with Linköping University [11].

1.3 Purpose

The purpose of the project has been to:

− Determine which modeling and computational approaches of the discrete event method are suitable in forest technology and make an inventory of the simulation tools available.

− Facilitate ongoing research activities in system analysis for forest technology using discrete event simulation

1.4 Objectives

The project had the following objectives:
1. Evaluation study regarding suitable choice of modeling and computational approach and choice of simulation tools,

2. A modeling framework

3. A prototype simulation tool

4. Demonstration of the framework and simulation tool by application in on-going forestry technology research projects

1.5 Collaborators

The project was carried out in close collaboration with professor Urban Bergsten and PhD student Back Thomas Ersson at the Swedish University of Agricultural Sciences (SLU). They provided insight into the related processes and environments.

1.6 Methods

The project can be broken down into several processes. The result of the project was strongly dependent on how smoothly these different components of the problem were solved. Compare e.g. producing a completely new software from scratch to adapting a fully working software made for the required conditions.

There were a lot of open questions that determined the path of the project, which made it an interesting and challenging problem-solving situation.

1.6.1 Evaluation study of discrete event modeling and simulation

The evaluation study was setup to answer the following questions:

- What are the alternative approaches for discrete event simulation of processes in forestry technology (planting, thinning, harvesting etc)? List pro and cons with the different approaches for use in forestry technology R&D.

- What software tools exist that can be used for the given purpose? List of pro and cons (price, computational efficiency, flexibility, user friendliness, output data presentation).

- What framework exists for modeling discrete event systems (language and file formats for representing a system, a state, system elements and event rules)?

- What framework exists for building new discrete event simulation tools (general or dedicated API, post-processing and visualization)

- Choice of approach, modeling framework and simulation tool in the project.

1.6.2 Modeling framework

- Adapt an existing or create a new framework

- Representation of a general processes in forestry technology as a discrete event system

- Possibility to specify a particular system (elements, properties, event rules)
1.6.3 Prototype simulation tool

- Adapt existing tool or develop a new that can run discrete event systems generated by the modeling framework.
- User-friendly support for designing/modifying forestry machine systems, import/load of environment data, machine definitions
- Post-processing and visualization for system analysis of specific forestry systems. Possibility for automatic generation of reports on productivity, fuel consumption, time studies of different functions and operation modes.

1.7 Test cases

Two test cases was set up. They are parts of current research at SLU, which is interesting since studying these cases not only tested the limits of discrete event simulation, but also brought new knowledge about the studied processes. For deeper information about the simulation models, see the theory/model section.

1.7.1 Multi-armed tree planting machine

Tree planting is today done manually in Sweden, no machine exists that production-wise can compete with manual planting. The proposed machine could be a big step in productivity, which is the reason for the simulation study. Basically, the idea is to add another planting arm to an existing machine with a single planting arm on it. Adding another arm to the machine gives rise to a queuing problem since the driver can only survey one task at a time. Furthermore, the two arms of the machine have some constraints in their relative movements, they cannot move freely in relation to each other.

The simulation studied, among many things, the required automation improvements needed to increase productivity. The proposed look of this new machine can be seen in figure (1.1a).

1.7.2 Continuous thinning with a new crane head

Rikard Wennberg and Julia Forsberg have in their master’s thesis project constructed a crane-mounted harvester head for continuous thinning. The new concept looks promising and the first step in its implementation is supposed to be one-armed, mounted on a 15-20 ton harvester without any automation [12]. The crane head can be seen in Figure (1.1b).

To have a name for this invention, it is called the RJ-Head from now on. This simulation study took a step into the future with this new invention and measured the productivity of a machine with two arms and, as with the planting machine case, studied the developments needed in form of automation. Time and development-wise closer concepts, such as a one-armed machine without automation, were also studied. Furthermore, a simulation of an existing thinning machine was performed for comparison.
1.7. Test cases

(a) A two-armed planting machine

(b) The new harvester head for continuous thinning.

Figure 1.1: CAD pictures of the test cases.
Chapter 2

Theory and Methods

2.1 Discrete Event Simulation

“Drama is life with the dull bits cut out”, a famous quote by Alfred Hitchcock. Discrete event simulation simulates a process, with the dull parts cut out.

The foundations of discrete event simulation are the idea of representing the world by a state and representing time as events instead of being continuous. A machine-forest system can be represented as a state by the position of the entities, species and size of the trees etc, although the number of configurations is infinite if space is continuous. Representing time as events shows no problem with respect to time itself, but rather when considering the time derivatives, i.e velocities. Movements have to be represented as a time delay and a switch of position. During this time delay, the state of the model will not correctly represent the system. A system that is sensitive to this is not suitable for discrete event simulation.

Not all problems can be addressed by this method, it requires certain characteristics to be valid. Most of all, the system must not be sensitively dependent on a differential equation with respect to time, such as the movement case mentioned above, since these equations need a continuous time integration.

In order to decide if a system can correctly be modeled in discrete event time, one has to realize that almost no systems are in truth discrete in time. The world as we know it is continuous in its movements, and a discrete event model of this world is an approximation. The validness of this approximations cannot generally be determined using mathematics. To control if the model is valid, one has to compare the statistics from the model with real world statistics.

As an example of a discrete event representation of reality, picture yourself a bird that flies, lands and flies to a new location. The one dimensional movement can be seen in Figure (2.1). A discrete event representation of this process may consist of three time stamps, the initial one, the one where the bird lands and the final location. This movement can also be seen in Figure (2.1) and is what mathematicians call a step function. Simply put, a model of the bird’s movement that needs a correct location of the bird in between these points is not suitable for discrete event simulations. A model that only needs the spots where the bird lands can simplify the bird’s movements to the discrete event model, and gain a lot of simplicity to the model without loosing any accuracy at all.

The movements does not have to be linear or polynomial for this to work, they just have to be predictable between the events, with possibility to calculate the time to move to a specified spot, or the distance moved for a specific time. In other words, the simulation needs
2.1. Discrete Event Simulation

A model for the movements. It can be stochastic or predictable, as long as the movement-time integration can be performed.

![Figure 2.1: A moving bird in one dimension, an example of the difference between reality and the discrete event representation of reality.](image)

Naturally, discrete event simulations are performed in systems with no spatial dependency at all. For example, a factory with assembly lines could be represented as a graph with nodes, where the goods are processed, and edges where transportation of goods are done. At both these places it takes a certain, possibly randomly distributed, time to process the good. Goods can be in queues for being processed etc. No place in this system shows any specific dependency of spatial entities, other than that the time of transportation between two points has to be known.

Harvesting a forest could, with some tweaks, be represented in the same way. The difference here is that the process order is determined by artificial intelligence of the machine, the order of processing is not known beforehand. Furthermore, the times for the processes are not known in advance, they have to be calculated from the state of the system, i.e position of machine, machine cranes and trees. This system shows much higher complexity than the factory system, which goes away from the spatial dependency by integrating distance and velocity, ending up with only the time variable.

### 2.1.1 Activity oriented paradigm

Professor Norm Matloff divides discrete event simulation into three different paradigms: Activity oriented, event oriented and process oriented [17].

The activity oriented paradigm is the most naive one, and obviously shows worst performance. The idea is to divide time into discrete time steps, with constant time increments. For every step, the program checks the state of the system for an event, and possibly updates the state. Most of the time steps, nothing happens and the computations were just a waste of computer power. It should be mentioned that nothing is wrong with the idea of constant time increments, most continuous simulations use this approach. The problem is that this method does not take advantage of the discrete nature of the system.
2.1.2 Event oriented paradigm

The event oriented method is much more efficient than the activity oriented one. Instead of searching the system at every time step, all events are handled by a priority queue, where the time is simply updated to the next event’s time, skipping all the time steps in-between where nothing happens. When an event occur, it updates the state. This new state of the system could mean that some of the pending events are infeasible, which means that they have to be deleted from the queue. This deletion also has to be handled by possibly adding a new, feasible, event. An illustration of this procedure, the so called event scheduling scheme, can be seen in Figure 2.2.

![Event Scheduling Scheme](image)

Figure 2.2: The event scheduling scheme, taken from [6].

In comparison to the activity oriented paradigm, this method takes advantage of the discrete nature of the system, resulting in a much faster program that can handle bigger systems.

2.1.3 The process-oriented paradigm

The process oriented paradigm is constructed to suit a certain type of discrete event systems. It is based on entities, processes and resources. An entity is undergoing a process as it flows through the discrete event simulation. In our chosen library SimPy, entities are called processes, which is quite confusing. From now on, processes and entities are equivalent.

A process/entity could for example be a forest machine or a customer in a bank. In the bank example, the resource would be the teller at the bank desk. In the forest machine case, the resource could be the driver, with limited simultaneous capacity and a crane could be an entity. In a typical case, processes are queuing for resources and perform tasks when they have acquired the resource. Continuing with the example of representing the driver as a resource, an entity would use the following procedure during a simulation:

1. It comes to a task that requires the driver’s attention
2. It enters the queue for the driver, i.e requests service from the driver
3. When it gets the driver’s attention, it remains in service for a period of time. During this time, the driver is busy and new entities have to wait in the queue for the driver.

4. It releases the driver and the task is done.

The process-oriented paradigm is designed for exactly this kind of systems, making it easy to handle the processes described above. Time is handled in the same way as in the event scheduling scheme, through a priority list, which makes this a fast method with easier implementation than the event oriented one.

2.2 SimPy

The pre-study resulted in that the discrete event library for Python, SimPy, was used. For more information about this choice, see the result section 3.1.

A complete description of SimPy is beyond the scope of this thesis, for detailed information see the available information at SimPy’s home page [23]. This section will give a short description of how SimPy works and about specific uses of the program. SimPy uses the process oriented simulation paradigm, see section 2.1.3. Resources and processes are supported, and in fact an entity can be both a resource and a process.

A process must have a process execution method (PEM) implemented. All commands to the SimPy program, given by the Python `yield` command, must be executed in this method. This requirement comes from the nature of generators, the Python structure that SimPy is built on, and it adds some constraints for the code. A naively way of programming would mean that these constraints forces almost all of the code into the PEM. However, a programmer goes around this by putting most of the code in private subroutines that return a list of commands for the PEM to execute. Here is an example:

```python
class Machine(Process):
    def PEM(self):
        # An example of how to divide the code
        while not simEnd:
            for cmd in moveToNext(): yield cmd
            ...
            for cmd in harvestTree(): yield cmd
            ...

    def moveToNext(self):
        commands=[]
        ... commands.extend([(hold, self, ReactionTime)])
        ...
        commands.extend([(request, self, driver), (hold, self, timeToMove), (release, self, driver)])
        return commands
```

Naive programming would put the movement and harvest commands in the process execution method. A better way is to make a list of the commands in another method (line 10, 12 and 14) and execute them in the PEM (line 4 and 6). In this way, processes with thousands of lines of code could be written, but still, with good documentation and comments, be quite easy to get a grip of through the PEM. However, one must be careful using this procedure, since the world around the process can change while a command is in the list and about to be executed.
2.3 Terrain

The implemented terrain model consists of trees, stumps, roots, boulders and holes after
digging. The idea is that this model should be extended in the future to include 3D topogra-
phic information and more types of obstacles. Depending on priorities, the model can
also be extended to include ground properties, such as elasticity, and different layers, e.g.
humus layer.

The model used by the thesis simulation is mostly in two dimensions, representing stand-
ing trees as spheres and felled trees as rectangles, although volumetric information is avail-
able for statistical measures. However, the roots and stones uses z-coordinates as well since
the mounding requires three dimensions. The reason for not doing the simulations in three
dimensions directly is that simplicity was striven for, and the 2D model was valid for these
cases.

2.3.1 Stumps and roots

The stumps are one of the most interesting parts of the terrain model. Sophisticated research
has been performed in Finland, where tree roots have been carefully measured and a model
has been setup [15]. The model is in three dimensions and very detailed. Parts of the model
have, together with Björkhem’s model from 1975 [22], been used to generate stumps from a
given breast height diameter, $d_{bh}$, with spatial positions taken from Herlitz’ type stands [14].

The root plate for a stump is defined as the zone of rapid taper, which occurs at a
distance of 0.5-1 m from the stem depending on the tree species. This area is defined as
non plant-able. The root plate is modeled as a circular disc, centered in the middle of the
stump and with a radius defined by equation (2.1):

$$r_p = \frac{d_{bh}}{2} + 0.5$$

(2.1)

where the unit meter has been used. In other words, the root plate has a minimum radius
of 0.5 meters and the radius increases linearly with $d_{bh}$. An example of a root plate can be
seen in Figure (2.3), where a finished stump is shown.

The root model has a resemblance to the tree abstract data type. Each root has a parent
and may have one or several children. The number of children, length and direction of edges
and similar properties will be described in this section. Most variables are generated by
a normal distribution, with a mean and standard deviation. Some properties differ from
species to species. In the models, only roots with a radius $r > 1$cm have been considered.
The number of first order lateral roots, spreading out from the root plate, is given by
equation (2.2):

$$n = 1.822 + 0.19 d_{bh}$$

(2.2)

with $d_{bh}$ given in meters. Each of these roots may fork immediately at the end of the root
plate. The initial radius of the first order lateral roots follows:

$$r_R = \frac{-0.18943 + 2.96 d_{bh}}{2n}$$

(2.3)

So far, the root model follows Björkhem et al’s work [22]. No stochastic properties have
been introduced yet.

The taper factor, $t_f$, is defined as the percentage that the radius of the root decreases by,
as seen in equation (2.4):

$$r_{\text{final}} = r_{\text{ini}}(1 - t_f)$$

(2.4)
2.3. Terrain

Figure 2.3: An example of a generated stump with roots.

Table 2.1: Distribution variables for the root model.

<table>
<thead>
<tr>
<th>var.</th>
<th>spruce</th>
<th>pine</th>
<th>birch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{f_1}$</td>
<td>$\mathcal{N}(0.04, 0.0206^2)$</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$t_{f_2}$</td>
<td>$\mathcal{N}(0.096, 0.0156^2)$</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$t_{f_3}$</td>
<td>$\mathcal{N}(0.1, 0.011^2)$</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$p_1$</td>
<td>9%</td>
<td>14%</td>
<td>11%</td>
</tr>
<tr>
<td>$p_2$</td>
<td>78%</td>
<td>77%</td>
<td>70%</td>
</tr>
<tr>
<td>$p_3$</td>
<td>13%</td>
<td>9%</td>
<td>19%</td>
</tr>
<tr>
<td>$p_{\text{sink}}$</td>
<td>13%</td>
<td>13%</td>
<td>13%</td>
</tr>
<tr>
<td>$q_1$</td>
<td>1</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$q_2$</td>
<td>(0.70, 0.30)</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$q_3$</td>
<td>(0.5, 0.25, 0.25)</td>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>$\alpha$ [degr.]</td>
<td>max($\mathcal{N}(12, 14.8^2), 0$)</td>
<td>max($\mathcal{N}(13, 17.3^2), 0$)</td>
<td>max($\mathcal{N}(12, 15.8^2), 0$)</td>
</tr>
</tbody>
</table>

The distributions for $t_f$ can be seen in Table (2.1). The $i$ is how many roots the previous node forked into.

When a node forks, there is a certain probability for one, two and three sub-roots to be produced. There is also a probability for the root to become a so-called sinker root, that goes straight vertically. These probabilities, $p_i$, can be seen in Table (2.1).

The $q$-factors for a node is defined as the quota of the incoming cross-sectional area of a root that is carried over to the next root. Typically, roots in a branch have a main root, that is bigger than the other ones. The $q$-factors, $q_i$ for different number of sub-roots can also be seen in Table (2.1). Roots penetrate the ground in different angles. These angles have been investigated by Kalliokoski et al. [15], and the angles in relation to the horizontal plane, $\alpha$, for the different tree species can be seen in Table (2.1).

In order to handle the roots in a collision detection situation, the depth of the roots has to be considered. A 2D collision detection system is preferred, since it is much faster than 3D and follows the philosophy of discrete event simulations better, one does not have to face the problem at a microscopic level. So how should the roots be handled in 2D, and still take
the depth into account. The method used here splits the roots into segments, and gives each segment a constant depth. In this way, the depth of the roots is correctly described, and the roots can still be implemented as polygons, compared to three-dimensional polyhedrons or cylinders, with bigger computational costs.

**Algorithm 1 Root splitting**

Require: taper factor $t_f$, root length $L$, initial radius of root $r_{ini}$, vertical direction of root $\alpha$, horizontal direction of root $\beta$, initial depth of root $z_{ini}$. Maximum length of root segment $L_{lim}$.

1: $z_{middle} \leftarrow z_{ini} - \frac{L \cos(\alpha)}{2}$ //depth at the middle
2: $r_{middle} \leftarrow r_{ini} \left(1 - \frac{1-t_f}{2}\right)$
3: if $L < L_{lim}$ then
    4: Create two new roots with this algorithm. Both with the length $L/2$, but one with $z_{in} = z_{middle}, r_{in} = r_{middle}$ and the other one with the same initial conditions as this root.
    5: else
    6: Create root with four nodes given from $r_{in}, r_{fin}, L, \beta$ and constant depth $z_{middle}$
    7: end if

A price of precision is payed by having the depth of the root segments as "stair-steps", but if bigger precision is required one can easily change the maximum length of the root segments, $L_{lim}$.

### 2.3.2 Stones/Boulders

In this thesis, the stones are only used for the planting machine simulation. However, the stone model may be used later by other simulations.

Stones are modeled as spheres, with different radii. Stones are not generated for the whole map at the beginning of the simulation, positioning them without collisions has the complexity $O(n^2)$ and the number of stones are really big. Spatial partitioning could remove some of this complexity, but it is still way too many stones. Instead, a Terrain method is called with a position and radius, and the stones in this area are returned.

Data for the density and mean value of the stones are taken from Andersson et al. who have measured stone frequency and mean surface areas for different terrain types. These terrain types are the ones used in the planting machine simulation, see section 2.5.10. Unfortunately, no data is available regarding the distribution of the boulder sizes. This model assumes that the boulders sizes are exponentially distributed, with the given mean value and a cut-off at 5cm side. The stones were only recorded when the side was 5cm or bigger. This suits our model quite good, since smaller stones should not be of interest. Since only the mean value of the stones is given, one has to decide whether the boulder diameter, the cross sectional area or the volume should be distributed around the given value. The three different choices give different distributions. Since the data from Andersson et al. is about the cross-sectional area, this approach is used.

Andersson et al. modeled their stones as cubes. With 30 years of computer development, we can safely model them as spheres instead, which will be faster in the implemented collision detection and in my opinion represents reality better. Since the threshold of 5cm side is for a cube, and the cross-sectional area of the stones should be equal, the corresponding
threshold for spheres is given by:

\[ r_{\text{min}}^2 \pi = 0.05^2 \Rightarrow r_{\text{min}} = \sqrt{\frac{0.05^2}{\pi}} \approx 0.0282 \text{ m} \quad (2.5) \]

Since the smallest stones are removed, and minimum of the exponential distribution is zero, the distribution follows:

\[ A \sim r_{\text{min}}^2 \pi + \text{Exp}(\lambda) \quad (2.6) \]

Where \( \text{Exp}(\lambda) \) is a value taken from an exponential distribution where \( \lambda \) is the so called rate parameter. In an exponential distribution, the mean value is given by \( \mu = \frac{1}{\lambda} \). Since the smallest stones are removed, the relationship changes to:

\[ \lambda = \frac{1}{A_{\text{mean}} - r_{\text{min}}^2 \pi} \quad (2.7) \]

where \( A_{\text{mean}} \) is the mean surface area for the boulders.

## 2.4 Common algorithms

Some of the algorithms are for a specific machine, while others are common for several simulations. This section covers the common methods.

### 2.4.1 Driver/Operator

A machine is in most cases operated by an operator. One of the main reasons for taking this new approach on forest simulations is that semi-autonomous systems can be investigated, so the operator class must be able to queue tasks for the operator, and statistics for this queue are necessary. Furthermore, the option for the operator to go on break at a certain, possibly stochastic, interval is also needed for the simulations to be realistic.

The SimPy Resource class gives a lot of help in implementing the driver. It handles the queue and queuing statistics. By inheriting this class, together with the Process class\(^1\), a mixture of a Process and a Resource can be achieved, where the driver can actually queue tasks himself. The syntax for the yield commands looks like:

```python
1 yield (request, self, self, prio)
2 yield (release, self, self)
```

where the first argument is the operation requested, e.g. seizing a resource or releasing it. The second argument is the entity requesting this, almost always \( \text{self} \). The third argument is the Resource, which most of the times is e.g. \( \text{driver} \) but in this particular case is also \( \text{self} \). The fourth argument is the priority, and is only used for requests. A high priority puts this command in the front of the queue.

This mix of Process and Resource looks confusing, but it works flawlessly. In order for the operator to take a break, put a hold command in between line 1 and 2.

The Resource class has a monitor instance that keeps statistics regarding the queue times and lengths. A problem rises with the above method, since the operator itself is recorded as an instance in the queue, and while resting, the monitor records the driver as working. A few subroutines for modifying these monitors, which are called after the rest, solves the problem.

\(^1\)Yes, Python allows multiple inheritance
UsesDriver

The class UsesDriver is inherited by all instances that uses the driver. It simplifies the command structure, since commands and if statements for resource control are quite long and makes the code ugly and hard to understand. Each time a command is executed, this class is called with the time and the automation dictionary instance for the process, e.g. 'plant', as arguments. The methods return a list of commands that either requests for the driver or releases the driver, and does the requested command. This class also supports priority requests, by overriding one of the sub-methods the priority could be set higher if e.g. a planting device's planting head want attention before the other device gets it.

2.4.2 Roads

A Road class has been programmed that is quite useful. Roads appear a lot in the forest, and for most of the time two machines cannot meet except at intersection points. The road is spatially located with a polygon. By using collision detection, one can determine if a certain machine is on the road or not. The Road is implemented as a Resource which means that machines can stand in line, waiting for access to a certain road. The idea of the road model is taken from the model implemented by the Münich group [1].

Roads were used in the thinning simulation for the stick-roads and corridors. Each road instance was given a list of the trees inside it. Through this list, the machine kept track of which trees to harvest before moving and which corridors that were not suitable and contained too big trees.

2.4.3 Cranes

The crane model will mainly follow Eliasson’s work [7], i.e the time it takes to move a crane is the maximum of the radial movement and the angular movement:

$$ t_{crane} = \max \left( \frac{\theta}{\omega}, \frac{r}{v_r} \right) $$

(2.8)

where $\omega$ is the angular velocity and $v_r$ is the velocity in the radial direction of the crane.

2.4.4 Continuous movements in a discrete time frame

As mentioned earlier, not all processes can be described by a discrete event model. Sometimes the exact position during movements is needed. For example, the two-armed planting machine should know where the respective arms are during movements since they put restrictions on each other. This section describes a method that keeps track of the position of a machine part at any time instance, not just at the discrete time stamps, for movements with constant velocity. This is done without any significant increase in computations.

Consider the movement of the bird from the introduction, Figure (2.1), again. If the velocity, $v$, would be constant, one could save the starting position, $p_{ini}$, ending position, $p_{fin}$ and the corresponding times $t_{ini}$, $t_{fin}$. The position, $p$, of the entity at an arbitrary time in the interval $t_{ini} < t < t_{fin}$ would be given by:

$$ p_t = p_{ini} + \frac{t - t_{ini}}{t_{fin} - t_{ini}} \cdot d $$

(2.9)

where $d$ is the unnormalized vector between $p_{ini}$ and $p_{fin}$. 
Thinking object oriented, it would be the bird’s responsibility to keep track of its own position. If another entity, e.g. a hunter, would like to know the bird’s position, it would call the bird’s `getPos` method. The bird would then use the current time, its stored information about the current movement and equation (2.9) to calculate its own position.

This is not a computing intensive procedure. If the `getPos` method is not called, the simulation continues in its discrete manner with no interruptions. The only operations added is storage of the variables in equation (2.9), which is only a few operations. The vector $\mathbf{d}$ only has to be calculated if `getPos` is called.

In the simulation model, a class `movesContinuously`, has been implemented with the methods `getPos` and `setPos` that handles the above mentioned procedure and variables.

It should be mentioned that this only works for velocities that can be expressed with an equation, arbitrary velocities would demand that an array of intermediate positions would be calculated every time the entity moved. This would be too time consuming and against the idea of these kinds of simulations.

### 2.4.5 Coordinate systems

The system is described by several coordinate systems. These are either cylindrical/polar or Cartesian, see Figure (2.4) for an example. Some processes are best described by cylindrical coordinates and some by Cartesian. Conversion functions makes switching rather easy, but since these functions are called extremely many times they take up a lot of the computational time required.

#### The world

Simply put, the world is described by $(x,y,z)$ coordinates with the origin at the lower left of the screen. The standard size is $40 \times 50\text{m}^2$ since the data files for roots are this big, but the program is flexible and can have other sizes.

#### The Machine

Most of the processes around the machine are described with cylindrical coordinates. The origin is at the beginning of the crane and $\theta = 0$ is to the drivers right, see Figure (2.4).

![Cylindrical coordinate system for the planting machine.](image)
Converting between the coordinate systems

If several coordinate systems are to be used, smooth conversions between these systems is needed. The procedure is standard in mathematics, where first a common origin is fixed by using the translation matrix, then the coordinates are translated into local coordinates for the other type of coordinate system. Finally, rotations are handled by multiplying with the rotational matrix $R$ or its inverse $R^{-1}$.

2.5 Planting Machine

The simulation of a planting machine is a part of Back Thomas Ersson’s PhD work, and will result in an article about it [8]. I provided help with the modeling and made the simulation part. This section is a mix of my work and Ersson’s work. Only mentioning my work would not give a full picture of the model.

The two-armed planting machine (2a) can be seen in Figure (1.1a). The four blue parts are called the planting heads. This is where the mounding and planting is done. Each pair of planting heads are together called a planting device, the heads in a planting device have a close connection since they have to mound and move at the same time, creating a queue situation in the case of re-moulding and prolonged planting phases. The working pattern for this machine is in overlapping half circles, where the machine is idle while planting and moves “backwards”, to not drive over the newly planted seedlings.

This machine exists today, but with only one crane arm and one planting device, called the M-planter. This is the most productive planting device for use in Nordic clear cuts with moraine soils [9], but it still does not compete with manual tree planting. The suggested two-armed machine could lower the cost of mechanized planting by increasing the productivity of the machine. However, this would require some degree of semi-automation since that driver cannot simultaneously handle two cranes. Semi-autonomous cranes are supervised by a human operator but have some tasks in their work cycles that are automated. The planting part is for example automated today in the M-planter.

2.5.1 Goal of the simulation

The main reason for doing this simulation is to investigate the productivity increase when adding another planting arm to the existing machine, and what automation tasks that are needed for it to work properly. The theoretical maximum should, without e.g. velocity improvements, be double the productivity of the current machine.

2.5.2 Working pattern for planting device

This section describes the overall working pattern for the planting devices. This pattern is iterated until the conditions for the simulation to stop are fulfilled, or until the planting device makes itself passive.
2.5. Planting Machine

Algorithm 2 Working pattern for planting device.

Require: startposition, information about surroundings, other planting device, machine specifications, list if ideal positions (see figure (2.5a).

1: while not stopped do
2:   if Driver is busy and automove is enabled then
3:     Move automatically to the next ideal position.
4:   end if
5:   Search in a spiral pattern for next planting spot. (Algorithm 3)
6:   if Planting spot was found then
7:     mound (Algorithm 4)
8:       if mounding was successful then
9:         plant (Algorithm 5)
10:   end if
11: end if
12: Break the loop if both planting devices have reached their end of the optimal path and did not find any planting spots in last search.
13: end while

2.5.3 Ideal planting coordinates

In the 2a case, the planting devices starts on the left side, as seen from the operator, of the half-circle working area. The way the planting devices are moved from there are of course a matter of how the obstacles are distributed in the working area. However, there is an ideal working pattern used by the automatic moving functions of the machine.

In a perfect world, this optimal pattern would be derived theoretically. However, an approach for this optimization has not been found. It is a really complex problem since space is continuous and the rules for the allowed positions are pretty deep. Therefore, the naively best working pattern is used. The pattern can be seen in Figure (2.5a). The idea is to minimize the time that the cranes waits for the other crane to move. The used pattern has identical traveling length, which should minimize queuing situations related to the movement pattern. It spans the half circle in a good way and there is enough room for the planting devices at each location.

2.5.4 Micro-site selection sequence

This stage is where the driver selects a location for the planting device to work, given a starting point and the movement restrictions from the cranes. In a typical case for the 2a-planting machine, movement has already occurred to the spatially best position, in some cases automatically. It is now up to the driver to choose the micro-site to use for the mounding and planting. A spiral searching pattern will be used, with an iterative approach that simply goes from the inside of the spiral and out, testing if the different positions are suitable for mounding and planting. Here, one faces a dilemma. The closer it is between the points in the spiral, the better the simulation becomes, but at the same time one has to pay a price of computational speed. Accuracy has been prioritized before computational speed. The chosen spiral is the famous archimedian spiral. It has the convenient property of conserved distance between spiral segments. The archimedian spiral follows the equation:

\[ r = a + b\theta \] (2.10)
where \( r \) and \( \theta \) are the polar coordinates and \( a, b \) are constants that determine the look of the spiral. \( a = 0.1 \) and \( b = 0.05 \) gives a good spiral for this application. In order to have a conserved density of points along this spiral, one can let \( d\theta \) follow the function:

\[
d\theta = \frac{d\theta_{\text{ini}}}{r}
\]

where \( d\theta_{\text{ini}} \) is the initial angular increment. Using equations (2.10) and (2.11), one ends up with the spiral in Figure (2.5b).

![Figure 2.5: Moving pattern for the 2a planting machine, and the archimedian spiral used in the micro-site selection.](image)

**Algorithm 3** Micro site selection sequence.

**Require:** information about surroundings, other planting device, machine specifications, list of spiral positions given by equations (2.10) and (2.11): \( \text{spiralPos} \). Terrain function \( \text{getObstacles}(\text{pos}, R) \) that returns the obstacles within a radius \( R \) from position \( \text{pos} \). Collision detection \( \text{collide}() \).

1. \( R \leftarrow \sqrt{\left(\frac{1}{2}\right)^2 + \left(\frac{2.01}{2}\right)^2} \)
2. \( \text{while not end of } \text{spiralPos} \) do
3. \( \text{pos } \leftarrow \text{spiralPos.next} \)
4. \( \text{for } \text{obst in } \text{getObstacles}(\text{pos}, R) \) do
5. \( \text{if } \text{collide}(\text{self, obst}) \) then
6. \( \text{return } \text{pos} \)
7. \( \text{end if} \)
8. \( \text{end for} \)
9. \( \text{end while} \) //Micro-site was not found, continue to next predetermined ideal position.

### 2.5.5 Planting Sequence

The planting phase is generally executed in parallel for each planting device’s planting heads, although some parts happen simultaneously for both the heads. When this phase is
initialized, the micro-site selection sequence (Algorithm 3) has already identified the position as obstacle-free, but obstacles below ground are still a threat. Stones and invisible roots around the given location are given by the *Terrain* class.

### The mounding phase

In the mounding sequence, the planting head digs and heaps soil. The seedling is thereafter planted on that heap. The first mounding is done simultaneously for both the planting heads, if re-mounding is necessary, it will result in a queuing situation for the planting head who is not re-mounding. Re-mounding is necessary if the planting head strikes an immobile boulder during the mounding phase, or if there is too many stones in the heap during the planting phase. It is also necessary if it strikes a root with a radius > 1cm.

The mounded volume is a half cylinder with a radius of 0.2m and a side of 0.45m, as described by equation (2.12):

\[
\begin{align*}
  z^2 + y^2 &< 0.2^2 \\
  -0.225 &\leq x &\leq & 0.225 \\
  z &\leq & 0
\end{align*}
\]  

(2.12)

Underground roots follow the rules:

- Roots are only relevant if bigger than 20mm in diameter. Roots smaller than that are expected to be broken in pieces by the planting head.

- Roots bigger than 20mm in diameter affect the planting if they appear inside the volume described by equation (2.13).

There are two different scenarios that can happen when the planting device hits a root. The mound can be classified as unsuccessful, which aborts the attempt to plant at the specific site, or the mound can be extended. The angle in relation to the planting head determines the outcome. If the root is parallel with the crane, i.e orthogonal to the planting head, the mound can be extended. If the root is orthogonal to the crane, or parallel with the planting head, mounding cannot be extended and planting is aborted. A 45° line distinguishes between these two options.

\[
\begin{align*}
  z^2 + y^2 &< 0.15^2 \\
  -0.2 &\leq x &\leq & 0.2 \\
  z &\leq & 0
\end{align*}
\]  

(2.13)

The overall mounding phase is described by Algorithm (4).
Algorithm 4 Mounding

Require: position pos, nearby boulders boul, nearby underground roots roots. The cross sectional area limit for an immobile boulder immobile.
1: for r in roots: do
2:     if any part of r is inside V1 (eq. 2.13 for V1) then
3:         $\theta \leftarrow$ angle between root segment and crane
4:         if $|\theta| < 45^\circ$ then
5:             struckedIm $\leftarrow$ True
6:         else
7:             Abort //Cannot plant at this loc.
8:         end if
9:     end if
10: end for
11: sumA $\leftarrow$ 0
12: struckedIm $\leftarrow$ False
13: $B \leftarrow$ empty list
14: for b in boulders do
15:     if center of b is inside V2 (eq. 2.12 for V2) then
16:         append b to $B$
17:         sumA = sumA + b.area //cross-sectional area.
18:     if b.area $>$ immobile then
19:         struckedIm $\leftarrow$ True
20:     if b vertically occupies more than 50% of the hole then
21:         Abort //Cannot plant at this loc.
22: end if
23: end if
24: end if
25: end for

If a planting head is forced to make a re-mound during the planting phase, it is considered to always succeed.

2.5.6 The planting phase

The planting phase is executed individually by each planting head and is associated with lots of queuing. If one head runs into problems the other head must often wait for the other to finish before continuing.

The output is associated with the boulders in the $B$ list in Algorithm (4), and planting is not initialized if the mounding phase was aborted. These boulder are stochastically distributed in a pile of the volume dimension in equation (2.12), but with an inversed z-coordinate.
2.5. Planting Machine

Algorithm 5 Planting

Require: position pos, B, sumA and struckedIm from Algorithm (4). The position of the
boulders B has been randomized inside the heap. Function reMound() that re-mounds
and interrupts the other planting head. plant() exits the algorithm and puts a seedling
into the ground.

1: if \( \text{sumA} < 4 \text{ dm}^2 \) then
2: \hspace{1em} if biggest cross sectional area in B is smaller than that of a 10dm sided square then
3: \hspace{2em} plant()
4: \hspace{1em} else if struckedIm then
5: \hspace{2em} reMound(), plant()
6: \hspace{1em} else
7: \hspace{2em} for \( \Delta \text{pos} \) in \([(0,0); (0,5\text{cm}); (0,-5\text{cm})]\) do
8: \hspace{3em} if \( \text{pos} + \Delta \text{pos} \) and 10cm down does not
\hspace{3em} collide with boulders in B then
9: \hspace{4em} plant()
10: \hspace{3em} end if
11: \hspace{2em} end for
12: \hspace{1em} reMound(), plant()
13: \hspace{1em} end if
14: else
15: \hspace{1em} Abort
16: end if

2.5.7 Machine Movement

The placement of the machine will be randomized within the 40x50\(\text{m}^2\) area. An important
factor for getting a correct value for the productivity is to have a timewise correct modeling
of the machine movements. This time will be the same for both the one-armed and two-
armed machine, but taking it into consideration is still important since the relative time
used for moving the machine will be bigger for a more effective machine. Since the working
pattern consist of overlapping half-circles, a simple but nevertheless accurate model would
be to use the distance between two points in the overlapping half-circles and a constant
machine speed. The used distance is the crane’s maximum length, in reality the distance
moved is slightly smaller, but at the same time this model does not consider acceleration
of the machine, so rounding up should be correct. The machine’s speed is estimated to 0.5
m/s.

2.5.8 Cranes

The two-armed planting machine has a sophisticated crane model, the cranes can be seen
in Figure (1.1a). Obviously, there are lots of degrees of freedom to handle. The machine
can rotate 360\(^\circ\), each of the sub-cranes can rotate 30\(^\circ\), five cranes can move radially. In the
simulation, this creates a big problem regarding checking if a certain position is allowed or
not, given the constraints for all the possible movements. Some simplifications have to be
made:

- The big crane closest to the machine has a constant length, i.e it does not move
  radially.
Table 2.2: Velocities for the crane movements

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radial movement of all cranes, $v_r$</td>
<td>1.6 m/s</td>
</tr>
<tr>
<td>Angular movement of main-crane, $\omega_{mainCrane}$</td>
<td>70.8°/s</td>
</tr>
<tr>
<td>Angular movement of sub-cranes, $\omega_{subCrane}$</td>
<td>15°/s</td>
</tr>
</tbody>
</table>

- The sub-cranes are handled as solid pieces, with a maximum and minimum length, resulting in a total of three cranes in the two-armed case.

- If the main crane needs to move, this movement is done first with the angular velocity from Table (2.2). After that, the sub-cranes move. The time for this is calculated using equation (2.8).

- A sub-crane can be moved while the other sub-crane is working. This requires sophisticated development regarding the interactions of the crane’s different hydraulic systems, but is essential in order to get a working and productive system.

- The sub-cranes are equipped with rotators, enabling them to keep a certain angle in relation to the machine, although the main crane is moving.

The sub-cranes are limited by the angle they have in relation to the main crane:

$$\theta_{lim1} \leq \theta \leq \theta_{lim2}$$  \hspace{1cm} (2.14)

These limits will be varied in the sensitivity analysis, a simple estimation of the torque and force that the cranes can handle gives the limits $\theta_{lim1} = 0$, $\theta_{lim2} = 45^\circ$. Machine factors will decide how big angles the cranes can handle in reality and this study cannot go further in the estimation of these limits.

To numerically decide if moving a planting device to a certain point exceeds the limits has proven to be very computing intensive. A system of equations can be setup from the variables in Figure (2.6):

$$\frac{L_2}{\tan \theta_4} - r = \frac{L_2}{\tan \theta_2}$$  \hspace{1cm} (2.15)

$$\frac{L_1}{\tan \theta_1} = \frac{L_1}{\tan \theta_3} - r$$  \hspace{1cm} (2.16)

$$\sin \theta_4 = \frac{L_2}{r_2}$$  \hspace{1cm} (2.17)

$$\sin \theta_3 = \frac{L_1}{r_1}$$  \hspace{1cm} (2.18)

$$\theta_3 + \theta_4 = \alpha$$  \hspace{1cm} (2.19)

where $\alpha, r, r_1, r_2$ are constants which can easily be computed. The problem could easily be algebraically reduced to the set of equation and constraints:

$$\frac{1}{\tan(\alpha - \theta_3)} - \frac{r}{r_2 \sin(\alpha - \theta_3)} = \frac{1}{\tan \theta_2}$$  \hspace{1cm} (2.20)

$$\frac{1}{\tan \theta_3} - \frac{r}{r_1 \sin \theta_3} = \frac{1}{\tan \theta_1}$$  \hspace{1cm} (2.21)

$$\theta_3 \leq \alpha$$  \hspace{1cm} (2.22)

$$\theta_{lim1} < \theta_1 < \theta_{lim2}$$  \hspace{1cm} (2.23)

$$\theta_{lim1} < \theta_2 < \theta_{lim2}$$  \hspace{1cm} (2.24)
One ends up with six variables and five equations/constraints, an underdetermined system. What is interesting here is if there exists a set \((\theta_1, \theta_2)\), that holds the constraints and equations (2.20-2.24), given the constants \(\alpha, r, r_1, r_2\).

The used approach in this simulations is the iterative one, testing a range of values and discarding it if none of the tested value fulfills equations (2.20-2.24). This leaves room for optimizing the code for faster and more precise execution. Linear programming may be the way to go.

For the one-armed machine there is only one crane and the case becomes simpler. The velocities for the crane movements can be seen in Table (2.2).

2.5.9 Degree of Automation

Table (2.3) summarizes the simulated steps and their degree of automation. Most of the automation will be varied in the sensitivity analysis.

2.5.10 Terrain types

Eight different terrain types, shown in Table (2.4) have been setup, with different parameters for the stones and roots. Herlit’z [14] type stand 554 will be used for the roots, except for terrain type 2b that uses type stand 553, with more roots, and type 1 that has no roots at all. The stands can be seen in appendix, Figure (B.2).

The data for the stone densities and sizes are derived from the studies by Andersson et al’s measurements from the late 70’s [18].
Table 2.3: Automation configuration table for the planting machine

<table>
<thead>
<tr>
<th>Moment</th>
<th>Time consumption</th>
<th>Automated 1a/2a</th>
<th>Automation varied in S-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mounding - dig</td>
<td>3s</td>
<td>N/Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mounding - heap</td>
<td>2s</td>
<td>N/Y</td>
<td>Y</td>
</tr>
<tr>
<td>Planting</td>
<td>3s</td>
<td>Y/Y</td>
<td>N</td>
</tr>
<tr>
<td>Move to next ideal pos</td>
<td>see tab. 2.2</td>
<td>N/Y</td>
<td>Y</td>
</tr>
<tr>
<td>Choosing micro-site</td>
<td>1s</td>
<td>N/N</td>
<td>N</td>
</tr>
<tr>
<td>Move to micro-site</td>
<td>1s</td>
<td>N/N</td>
<td>N</td>
</tr>
<tr>
<td>Switching focus between arms</td>
<td>2s</td>
<td>-/N</td>
<td>N</td>
</tr>
<tr>
<td>halt mounding/-planting</td>
<td>1s</td>
<td>N/Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 2.4: Terrain type parameters.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Stumps per ha</th>
<th>Boulder freq. [1/m²]</th>
<th>Mean boulder surf. area [dm²]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Afforestation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Reforestation, “granåker”</td>
<td>230</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2b</td>
<td>Reforestation, “granåker”</td>
<td>635</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Reforestation, clear-cut, moraine soils</td>
<td>230</td>
<td>28</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>Reforestation, clear-cut, moraine soils</td>
<td>230</td>
<td>43</td>
<td>1.3</td>
</tr>
<tr>
<td>5</td>
<td>Reforestation, clear-cut, moraine soils</td>
<td>230</td>
<td>23</td>
<td>3.2</td>
</tr>
<tr>
<td>6</td>
<td>Reforestation, clear-cut, moraine soils</td>
<td>230</td>
<td>51</td>
<td>1.5</td>
</tr>
<tr>
<td>7</td>
<td>Reforestation, clear-cut, moraine soils</td>
<td>230</td>
<td>102</td>
<td>0.7</td>
</tr>
</tbody>
</table>

2.6 Thinning Machine

The simulation of the thinning machine evaluates a new crane head. The model is simpler than the one of the planting machine in the sense that it does not require any soil information, such as roots and stones. Instead, effort is put on correctly modeling the geometric harvesting pattern.

2.6.1 Goal of the simulation

The obvious overall goal for the thinning machine simulation was to produce an accurate simulation of the proposed machine in the environment created by this thesis project. Smaller and more measurable goals can also be set up, and the following list summarizes them, ordered by importance:
1. Simulate a harvester, with the two set ups of a single arm and two arms, that moves in a dense forest. Also implement a thinning machine that is used today, harvesting tree by tree, for comparison.

2. Measure the productivity, defined as trees per hour, and compare it between the two cases.

3. Vary the degree of automation, and measure the resulting workload for the operator.

4. Use Cranab Access cranes, with an allowed tilt of 30 degrees, and analyze how it affects the machine productivity.

5. Calculate the torque from the harvester head during the simulation, and relate it to the required machine mass and distance between the wheels.

6. Implement the machine as both a harvester and a harwarder, and analyze the result.

### 2.6.2 Model

**Machine**

Since this simulation mostly simulates a future machine, there was some freedom in designing the used vehicle. For simplicity, specifications from an existing, one-armed, forest machine was be used.

**The vehicle** Komatsu's 901.4 harvester was used, the machine can be seen in Figure (2.7a). With a weight of 15 tons, it is quite small and makes a smaller impact on the environment.

![Vehicle and thinning corridor information](image)

Figure 2.7: Vehicle and thinning corridor information.
Table 2.5: Properties for the crane head

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>2m</td>
</tr>
<tr>
<td>Width</td>
<td>1m</td>
</tr>
<tr>
<td>Depth</td>
<td>1.5m</td>
</tr>
<tr>
<td>Mass</td>
<td>1000kg</td>
</tr>
<tr>
<td>Maximum mass of trees</td>
<td>350kg</td>
</tr>
<tr>
<td>Tree diameter at breast height ($d_{bh}$)</td>
<td>$5cm &lt; d_{bh} &lt; 10cm$</td>
</tr>
<tr>
<td>Maximum tree Height</td>
<td>10m</td>
</tr>
</tbody>
</table>

The second crane was placed on the other side of the cabin and the cabin was assumed to be able to rotate independently of the cranes. This would not be possible on the current machine, but the simulation takes place in the future and the relevant properties for this simulation can be taken from the 901.4 harvester.

The machine moves along a strip road and tries to minimize the stop points, harvesting only occurs when the machine is at rest. The strip road is harvested continuously as the machine proceeds along it and is located "vertically" on the map. An example of a strip road can be seen in figure (2.7b), it is placed horizontally in the picture and the fan shaped pattern are the used corridors.

**The crane head** As mentioned, the crane head in Figure (1.1b), called RJ-Head, was used. Discrete event simulations are not suitable for simulation of the head in detail, continuous simulation approaches are preferred for that. The simulation instead focuses on properties like the movement pattern and maximum load for the harvester head. A typical chop-scenario would be that the head approaches the tree at a certain speed, chops it down and adds it to the head’s storage volume, as seen in Algorithm (6). A stochastic failure rate could possibly be added, but so far the simulation expect the driver to always make correct decisions regarding if a tree is possible to cut.

**Algorithm 6** Tree cutting approach, an example

```
Require: positions $p_{tree}$ and $p_{head}$, radius $r_{tree}$, velocity $v_{head}$, time $t$, function for 2D distance calculation $dist()$, function to determine if there is room for tree $fits(tree)$, list of trees in head $trees$, process time $t_{chop}$.

1: $t \leftarrow t + \frac{dist(p_{tree}, p_{head})}{v_{head}}$
2: $p_{head} \leftarrow p_{tree}$
3: if $fits(tree)$ then
4:    insert tree into trees
5: end if
6: $t \leftarrow t + t_{chop}$
```

The discrete nature of the above algorithm simplifies the process and actually makes it easier to produce accurate results. Not all problems needs to be addressed at a microscopic level. Of course, Algorithm (6) is very simplified, it does not at all describe the process of handling the weight of the head and its trees and the crane and vehicle interactions do not participate in this part, but it gives a good picture of the nature and philosophy of spatial discrete event simulations.

The properties and limits for the crane head can be seen in Table (2.5).
2.6. Thinning Machine

Cranes The crane model follows the common model, described in section (2.5.8). The velocities are identical to the planting machine velocities, see Table (2.2). There was no time to implement the Access crane.

A maximum crane length of 11 m was used for both cranes.

Working pattern

The working pattern is a description of how the machine works. The tasks are divided into different sections, which could be automatic or non-automatic.

Movement The movement of the machine is in stages, performing no work at all during movements. Cranes are in their tightest position, closest to the machine, during movements. Movements occur along a strip road, as described in Figure (2.7b). The strip road referred to here are the horizontal corridors in the Figure. The simulation starts at one end of the strip road and ends when the machine has reached the other end. The machine harvests both the strip road and the corridors.

Crane movement The fan shaped corridors in Figure (2.7b) shows how the crane moves. Some overlaps of corridors may also take place. This geometrical thinning pattern has proven to increase efficiency during thinning [21]. Research is currently being performed at the Swedish University of Agricultural Sciences (SLU) regarding how this thinning pattern affects the trees and ground properties, but this simulation assumes that this working pattern is preferred. If the storage maximum of the crane head is reached, the crane moves back to the machine, leaves the trees next to it and then continues into the corridor.

The corridors are slightly narrower than the ones in Figure (2.7b), and the number of corridors per side of the strip road are increased to five for the cranes with the RJ-Head. Simulations with the conventional crane head have three corridors per side. The width of the corridors is 2 meters for the conventional ones and the new head uses narrower corridors, with a width equal to the width of the crane head.

Between 40-70% of the trees are harvested during a simulation.

Environment

Trees The branches of the trees might be a problem for the new crane head. Because of this, the tree species are reduced to pine and birch, spruce are avoided since its branches have the worst characteristics.

The spatial information about the trees are taken from a study [5], where the trees for several stands suitable for thinning have been carefully measured for areas of $25 \times 40$m. Information is available about the position, breast height diameter ($d_{bh}$), mass, volume and more, for this study irrelevant things. The number of stands was initially around 60. However, a first analysis of the data files discarded most of these files as unsuitable, many of them contained too much spruce trees, a tree that has been identified as unsuitable for the new crane head. Furthermore, a lot of the stands had too few small trees. Finally, only six stands qualified as suitable. This is a problem since the terrain is the only stochastic thing in the model, apart from that the model is deterministic. In order to get a measurement of the error one needs some stochastic properties. The remedy to this problem was to randomize the starting x-position for the vehicle. The tree position in the stands can be seen in appendix, Figure (B.3).
Terrain variables, such as elevation and obstacles other than trees, are not accounted for in this simulation.

2.6.3 Degree of Automation

Table (2.6) summarizes the simulated working steps and their degree of automation.

2.6.4 Corridor selection

The corridor selection algorithm is a process where the algorithm is crucial for the outcome of the simulation. The number of stop points for the machine should be minimized for optimized speed. Furthermore, between 40% to 70% of the trees should be harvested. The maximum number of corridors per side and stop point is set to three for the conventional crane heads and five for the new RJ-Head. This is because the RJ-Head has a narrower corridor width than the conventional head. The conventional head uses a width of 2 meters while the RJ-Head’s corridor width is the same as its own width, 1 meter.

The length of the corridors is always the maximum length of the crane. Algorithm (7) describes the process of choosing corridors. It has an iterative method and leaves room for time improvements, but gives a good result.

The iterative method in line 8 of the algorithm takes a lot of time, since a small angular step is needed to make sure that no optimal setup is missed. This part can certainly be improved if the model should be used at a larger scale.

2.7 Program Structure

A map of the used program can be seen in Figure (2.8). The terrain and operator classes are described in sections (2.3) and (2.4.1) and algorithms for the collision detection can be seen in appendix A.

As one can see, the machines class is the "spider in the net", it uses most of the created modules. The machine class is also where most of the code is situated, which is quite logical since the artificial intelligence is the most complex part of the program.

The simulationClasses module sets up the simulation parameters. If, for instance, one wants to vary the crane radius limits and continue with simulations as long as the standard deviation is above a certain value, then these requests are handled here. In the current non-GUI version, the interactions with the user is in the simulationClasses and in some cases with the postProcessing, for tweaking graph outputs to desired looks. A GUI would communicate with these parts and with the PlotWorld module.

The PlotWorld output can be seen in Figure 3.1, which is an example of a thinning simulation. This class is called at a certain delay which is very short when creating movies and longer for debugging. The standard values are to just show the output at the end of the simulation. For long runs PlotWorld is not used at all.
Table 2.6: Automation table for the thinning machine.

<table>
<thead>
<tr>
<th>Moment</th>
<th>Time consumption</th>
<th>Automated (Y/N/Vary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move machine</td>
<td>1 m/s</td>
<td>N</td>
</tr>
<tr>
<td>Move arm out</td>
<td>1.5 m/s</td>
<td>Vary</td>
</tr>
<tr>
<td>Move arm in after harvest</td>
<td>1.5 m/s</td>
<td>Vary</td>
</tr>
<tr>
<td>Chop a tree</td>
<td>3s</td>
<td>Vary</td>
</tr>
<tr>
<td>Place harvested trees at the side of the strip road</td>
<td>7-10s</td>
<td>Vary</td>
</tr>
<tr>
<td>Switch operator focus between arms</td>
<td>3 s</td>
<td>N</td>
</tr>
</tbody>
</table>

Figure 2.8: Structure of the program. Rectangles are existing libraries while ellipses are new modules. Arrows indicate that a module/library is using the instance pointed from.

Algorithm 7 Algorithm for corridor selection

Require: crane length $L$, corridor width $W$, number of corridors per side $n$, start position $pos$. Corridor(pos, side, $\theta$, $L$, $W$) class. Map limits xlim, ylim

1: $\theta_{\text{min}} \leftarrow \frac{\pi}{4}$ //Minimum angle, in rel. to the strip road.
2: $\sigma \leftarrow \frac{\pi}{3n}$
3: while True do
4:     // (left, right):
5:     for side in [-1, 1] do
6:         for $i$ in [0, 1, 2, ..., $n$] do
7:             $\alpha = \text{side} \cdot \left(\theta_{\text{min}} + \frac{\pi - 2\theta_{\text{min}}}{n-i}\right)$ //Angle in rel. to strip road
8:             $c \leftarrow \text{corridor}(pos, side, \theta, L, W)$ for $\theta \in [\alpha - \sigma, \alpha + \sigma]$
9:         Save the corridor with the most choppable trees.
10:     end for
11: end for
12: $pos.y \leftarrow pos.y + \cos(\theta_{\text{min}})L$
13: if $pos.y > \text{ylim}$ then
14:     break loop
15: end if
16: end while
Chapter 3

Results

3.1 Pre-study

The first aim of the pre-study was to investigate if this project was reasonable for a master thesis project. An example where this would not have been the case would be if a finished software for this situation was found.

The second aim was to get further insight into the discrete event simulation subject and in particular its uses in the forest industry.

3.1.1 Software study

There are numerous different tools for modeling discrete event simulations. Evaluating them all would be a project in itself, there was no time to do a throughout evaluation of all the programs. Instead of this, a limited study was performed where the main functions of the different programs were analyzed. A certain group of criterion’s were identified as crucial:

1. Handling of spatial variables such as trees, boulders and roots.
2. Ability to include terrain factors in the spatial environment.
3. High modeling flexibility.
4. Visualizations for debugging and verification purposes.
5. Ability to store statistics during simulation.

The first criterion was also identified as the hardest one. During the investigation, material from previous software studies by Swain was proven useful, although these investigations did not cover the spatial problems [20]. The following list summarizes the investigated software:

- AnyLogic
- Arena
- GoldSim
- NetSim
- Plant Simulation
3.1. Pre-study

- Renque
- SimEvents (Matlab)
- SIMUL8
- Promodel
- GeneSim
- erlSim
- OMNeT
- AutoMod
- WITNESS
- Enterprise Dynamics

Most simulation software are made for assembly-line like environments. Putting up an environment with thousands of trees, roots and other obstacles and let machines interact with them is far away from that kind of environment. Thus, a lot of the programs could immediately be discarded. The pricing was also an important factor, where some of the programs had a cost of tens of thousands of euros. It could be worth the money if the program supplies a good solution to our problems, but after testing the programs and reading program descriptions, this has not been observed.

In the program list, Automod stands out since it has been used in a German effort to simulate the production chain in a forest [24]. The program seems to be able to keep track of spatial entities such as trees. However, the simulations are run in 2D [13], which make detailed terrain modeling impossible. Possibility for extension of the terrain model has been identified as important. Moving is done with a so-called routing table, similar to a graph [1]. Thus, movements are quite restricted, which is a downside. The visualization tools are impressive but the other limits are too big for this to be an interesting alternative.

Conclusion

Simulations in the forest industry is a delicate and complex case which has to be done from the bottom up. No finished software was found that could do this, although time was a factor that limited the investigation. Thus, a program of our own has to be implemented.

One more advantage with this option is that a successful implementation could be used commercially, expanding the potentials of the project.

3.1.2 Library study

Is a library needed?

Could a discrete event library be created from scratch? This would have several benefits since programming language could be chosen from the beginning and external open source licenses would not be a problem in a potential future of expansion.

The licensing problem of the libraries is not really a problem, since there are several different open source licenses. The license used in e.g. SimPy is GNU Lesser General Public License, which does not have any serious restrictions regarding commercial use of programs using the library. Furthermore, if this would be a problem one could face it later and make a new library.
How time-consuming is it to create a new library?

Creating, testing and documenting a new library would be very time consuming, and time is scarce in this project. However, the methodology is briefly described here.

One of the foundations of a standard discrete event simulation program is the event scheduling scheme, it can be seen in Figure (2.2). The system is described by its state. The state could be changed by events, which are ordered in a priority queue with time as the priority. Depending on the nature of the event, following events could be classed as infeasible, a situation that has to be handled by the entities involved, resulting in new events coupled to the new state of the system. The process oriented routine follows a slightly different methodology, but the idea is the same and the flowchart only looks slightly different.

In practice, this is all implemented by using concurrency, where functions of every entity are active simultaneously, waiting for permission from the event list to perform its actions. In the case of infeasible events, the methods of the entities are written to deal with these changes of state.

Debugging would probably be the most time consuming part in implementing a new library. The time of the project could certainly be allocated for better things than this, tested and good working libraries already exist.

Is implementation possible in Matlab?

Matlab’s limited support for threads make this hard to implement in a nice and intuitive way in Matlab. Mex-files probably have to be used which is quite complicated if the external programs have to keep track of the state of the system, preferably stored in Matlab.

The plug-in SimEvents should also be mentioned. The program in itself seems to be based on assembly-line like scenarios, just as most of the other software. However, it is not impossible that some internal routines can be used in plain m-files. This scenario demands the purchase of both Matlab and SimEvents licenses. There are no big advantages compared to the free alternatives.

The Libraries

There are several free libraries available in different languages. Some of them are outdated and the licensing of them differ a bit. A lot of their properties originate from the programming language that uses them. In the search process, a lot of the languages were sorted out for different reasons. For example some of them are outdated, being constructed before the real object oriented languages were born.\(^1\)

The following factors were identified as important when comparing:

- Flexibility - are there any restrictions?
- Speed
- Applicability for the relevant simulations
- Activity of the library, continuous updates and an active user community is preferred.

---

\(^1\) Object oriented programming was actually developed out of Simula, a discrete event simulation language born in Norway in the 1960’s. This language evolved into C++ which inspired the creation of Java, and the rest is history.
**3.2. Simulation model**

SimPy

SimPy is a library for Python. In contrast with most of its competitors, SimPy uses Python’s generator class instead of threads. This results in more elegance in the implementation, where the command `yield` is used for the entities to interact with the rest of the simulation.

The user community is quite active and so are the updates, with the most recent update in June 2010. The documentation is very good, with a lot of examples and manuals.

On the downside, Python is generally slower than both Java and C++.

Tortuga

Tortuga is based on Java, which is a widespread computer language that is acknowledged to be rather fast. Every entity is represented by a thread, which limits the number of entities by the number of threads supported by the Java Virtual Machine. This number is currently 6000, which should be enough for our needs.

The good things about Tortuga ends here. The ship seems to be abandoned with no updates and bad documentation.

C++Sim

This library was written in the early nineties and was later accompanied by JavaSim for Java. C++Sim have been used a lot but it seems like the founder lost control over the libraries and its homepage and user community was lost. The libraries are not maintained any more and the use of them seem to have declined.

SSim

A library for C++ and Java. It has seen some updates through the years but an overall comment is that it is badly documented and its usage does not seem to be that widespread. However, it has the requested functionality and deserves to be mentioned.

**Conclusions**

At some point a decision has to be made. The SimPy library was identified as the most active library. Usage is hard to estimate but SimPy is probably the most used library as well. The syntax of SimPy and the Python language overall is a big plus. Python is a widely used, platform independent, programming language which is supported by numerous pre- and post-processing programs. In this way, the extraction of information from for example GIS is simple and the visualizations could possibly be shown in a GIS environment. Python will provide, if SimPy shows some limiting behavior the source code is open and it can be modified.

On the downside this library might be slightly slower than the others. The reasons are several, but comparing to C++ the main reason is that Python runs through a virtual machine, while C++ is compiled into machine code before execution. However, computing intensive code segments can be written in C, with a simple Python-C interaction using the ctypes library.

A decision was made. The project is using SimPy.

**3.2 Simulation model**

The simulation model was created successfully, although one never gets done taking such tools to perfection.
3.2.1 Performance

The performance, in terms of speed, of the program is acceptable. The implementation of the two test simulations took more computations than one could think at first sight, which shows the importance of good algorithms.

A typical simulation takes around a minute to finish. The tool in itself is rather fast, the slowness comes from the algorithms used by the various simulations, for example the way the planting machine scans the surroundings. These algorithms can most certainly be optimized, but this has not been prioritized since they are not a part of the model in itself. The reasoning was that as long as a simulation series can be completed over night at a single processor no big effort should be put at making the code more efficient.

When making a profile run, one sees that collision detection and coordinate system switching takes up big ratios of the time. Optimizing these functions, possibly by writing them in C-code instead of Python, would certainly speed up the program. The main problem however is the algorithms that are using these functions unnecessary many times.

3.2.2 Visualization

A snap-shot from a simulation can be seen in Figure (3.1). One can see a debug picture/video, that shows the current state of the simulation. Three monitor windows are also available, that shows the statistics of choice, in this case the trees harvested, driver waiting queue and driver working pattern. This visual interface could of course be turned off in case of long runs. The visualization should be seen as a debug tool.

![Simulation Image](image)

Figure 3.1: A corridor thinning simulation.
Table 3.1: Automation configuration table for the planting machine.

<table>
<thead>
<tr>
<th>Sim. No</th>
<th>AutoMound</th>
<th>AutoMove</th>
<th>AutoHalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>D</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>E</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 3.2.3 File Formats

The output file format for the simulation is simply a `.txt` file, with columns separated by tabs. This format was chosen since it can be read by all post processing programs worth mentioning. Furthermore, a tag was added to indicate the beginning of the data sampling in the file. This means that before this tag, various information about the simulation can be added, such as description of the variables. This standardization was useful in the post processing work, which was performed in Python with the `matplotlib` library for the plots.

Work was not put on specifying a system, such as location and size of trees, from a data file, since data was only taken from one source, namely the old type stands from Bredberg and Herlitz [5][14]. These data files are definitely not recommended as standard files for the future, but since they were the only source used no time was put on setting up a standard and converting the existing files to this standard. A future investigation has to decide what to use, one option is the XML format for example and another could be a database standard, such as SQL.

### 3.2.4 GUI potential

A graphical user interface could simplify the creation of models and simulations. One could determine machine parameters, such as crane length, machine speed etc and simulation parameters, such as number of iterations or desired standard deviation of mean value. Plotting and post processing could also to some extent be achieved, but for more detailed post processing, i.e. producing detailed plots for articles, data could be extracted in a common output data sheet.

All this is clearly possible, it is just a matter of design and time. The time in this project have not been prioritized to create such a GUI, instead simply finding out that it is possible.

### 3.3 Planting Machine

#### 3.3.1 Automation

The different automation stages used in the simulation can be seen in Table (3.1). One can observe that not all combinations have been tested, for instance having only the halt task automated makes no sense. The result for different terrains can be seen in Figure (3.2). When the stone frequency and size increases, one can see significant decrease in productivity. Also observe that the auto-halt function (difference between B and D) shows an increasingly importance with increasing boulder occurrence. The difference is negligible in terrain type 1, since their simply does not exist any stones or stumps in this terrain type, while the difference in productivity is really big in terrain type 7. Properties of the different terrain types can be seen in Table (2.4).
Observe that the result for terrain type 1 shows no variance. This is because these simulations are deterministic, the stochastic parts of the model are in the stone and roots, and terrain type 1 does not have any of those.

Adding another arm to the machine does increase productivity, but not with as much as hoped. The following sections will investigate this further.

As one would predict, configuration $E$ gives the best result, and the following measurements will have this configuration.

![Figure 3.2: Results, while varying automation and terrain. The error bars indicate two standard deviations, i.e 95% confidence intervals. For configuration descriptions of system A-E, see Table (3.1).](image)

**3.3.2 Crane Limits**

The inner and outer crane limits determine the working area for the cranes. It is not clear how this area affects the productivity, which is why these limits have been varied in the sensitivity analysis. The result can be seen in Figures (3.3) and (B.1). One can conclude that varying the inner radius has no big impact on productivity, while the outer radius is positively correlated with the productivity, with a gain of up to 25% in productivity (for terrain type 2).

**3.3.3 Angle Limits**

The angular limits for the planting machine is a crucial part for the productivity of the two-armed machine. This information is important when designing the machine parts,
3.3. Planting Machine

Figure 3.3: Results, while varying outer crane radius and terrain. The error bars indicate two standard deviations, i.e. 95% confidence intervals.
since bigger angles probably means more expense. Figure (2.5) shows the result, simulated in terrain type 3. One can see that increasing the outer angle by just $20^\circ$ will increase productivity by around 10%.

![Figure 3.4: Variation of the angle limits for the two-armed planting machine in terrain type 3. Default values are $[0^\circ, 45^\circ]$](image)

**3.3.4 Mounding blade width**

Figure (3.5) shows the results when varying the width of the mounding blade. One can clearly see that this variable is strongly connected with the density of stones. The principle is pretty clear, the bigger the head, the bigger the probability of striking a big root or stone. Thus, a small head is preferable from this aspect. On the other hand, one has to consider that a smaller head means less mineral soil for the plant.
Figure 3.5: Variation of the mounding blade width.
Table 3.2: Automation configuration table for the thinning machine.

<table>
<thead>
<tr>
<th>Config.</th>
<th>Move arm out</th>
<th>Move arm in</th>
<th>Dump trees</th>
<th>Chop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>B</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>E</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

3.4 Thinning Machine

3.4.1 Automation

The automatic configuration table can be seen in Table (3.2). As with the planting machine, all possible configurations have not been tested. For example, moving the arm in after harvest was identified as the easiest task to automate. Thus, this task is automated in all cases but A.

The result for the different automation configurations is shown in figure (3.6). First of all, one can notice that, Ceteris paribus, the machine with the RJ-Head is almost 100% more productive than the conventional head. Adding automation and another arm may increase productivity with as much as 200%. One can also see that adding automatic movements of the crane increases productivity the most, while automatic chopping of trees does not benefit as much. To automate the process of releasing the trees from the head and place them on the ground would also significantly increase the efficiency of the machine.

Figure 3.6: Variation the level of automation for the thinning machine with 95% confidence intervals. For information about configurations A-E, see Table 3.2.
3.4.2 Corridor statistics

Statistics from the corridors was something that was specifically asked for. The average sum of tree weight in the corridors can be seen in Figure (3.7). Remember that the conventional head uses wider and fewer corridors than the RJ-Head, resulting in more trees per corridors. Statistics regarding the number of trees in the corridors can be seen in appendix, Figure (B.4). Information can also be found regarding the number of trees in the stick road in Figure (B.4c).

One has to remember that simulations were performed for only six tree stands, which is the reason for the clustering and wide range of values regarding the statistics from the corridor.

![Figure 3.7: The average sum of tree weights in corridors.](image)
Chapter 4

Conclusions and discussion

4.1 The performed simulations

**Planting Machine**  Figure (3.2) shows that adding another crane increases productivity, but the increase is not always that big compared to the one-armed case. The cranes are limiting each other and the queues for the driver reduces the productivity increase. However, the relative difference increases with the density, but not necessarily the size, of the boulders. In terrain type 7, the two-armed planting machine with the highest degree of automation had a double efficiency. This could be compared with terrain type 2b, where the productivity was equal.

The automatic movement procedure does not add as much productivity as desired and hoped for. This could be related to that the pattern of planting points is not optimized, the naively best pattern has been chosen but it has not been shown that it is the most effective one. Furthermore, the automatic movements do not move the planting device to a plant-able position, the driver has to choose where to plant. This means that in some cases, the crane goes back the same way that it went forward, which takes more time than if the driver would have identified that spot from the beginning and manually moved the crane there. The automatic movements have great complexity and this algorithm could definitely be optimized further, designing it is an art of engineering itself.

As expected, increasing the crane range improves productivity. The plots for this show big fluctuations, and it is certainly not an artifact of the limited precision, which the deterministic results in terrain type 1 shows. This behavior probably comes from how the planting devices are fit into the working area. For example, with a planting device of length 50cm and crane limits [4m, 9m], one can fit 5 planting spots on the length (one needs room for the heap and for the hole). If the limits are instead [4.1m, 9m], one can suddenly only fit 4 planting spots, a decrease of 20%. Of course, the simulations are much more complex than that, with obstacles in the way and moving patterns to keep track of, but the reason for these errors probably originate from simple things as the above example. Optimized moving patterns could probably improve productivity for fixed crane limits, but the productivity as a function of crane limits will still probably show some “Heaviside behavior”.

With the above reasoning, one would expect a periodic behavior. The density of points is too small to effectively analyze the result with e.g. Fourier analysis or auto correlation, one needs more points for that. A long run, with much smaller increment steps for the cranes, may be performed at a later stage if this is interesting.

Angle limits show the expected behavior, where wider limits leads to increased efficiency
of the machine. These simulations are not suitable for determining the limits, for doing that one needs calculation on the torque at all machine parts and abrasion resistance. In fact I think that trying it in reality is the best approach for determining these angles. What these simulations can do is determining how much one would gain by extending these limits, or how much one would loose by narrowing them. The results show that these limits are indeed important for the productivity, and if a narrow solution is the only possible one the machine will be a failure.

Variation of the width of the mounding blade has a huge impact on the productivity of the machine in terrain types with boulders. Figure (3.5) shows the dependencies and one can clearly see that the difference is only significant for the terrain types with boulders. The width of the blade should be minimized, but one has to keep in mind that when the width of the blade decreases the volume of soil decreases as well. It is probably a good idea to investigate how much soil the plant needs, since this volume shows such critical behavior depending on the width of the blade.

Again, it should be mentioned that these simulations do not investigate what is possible, e.g. possible angles. They investigate the usefulness of certain key inventions or changes. For instance, it has been shown that automatic movements in its current state does not increase effectiveness of the machine. Instead, resources should be put on developing other processes, such as mounding.

All the investigated factors have to be considered when making an economic analysis of the usefulness of developing this new machine. Without doing this economic analysis, the results suggests that increasing productivity by adding another planting arm demands big investments in the different semiautonomous functions. Furthermore, the result differ a lot depending on which terrain that is used. Thus, the properties of the terrain where the machine is supposed to work have to be considered.

**Thinning Machine**  The simulations of the Thinning machine shows that there is no doubt about the usefulness of the new crane head invention. A doubling of the productivity should certainly make this product commercially possible. The big question marks regarding this head are outside of the scope of this simulation study. The simulations assumes that the branches of the trees does not create a problem, this is a potential threat that has to be investigated. Furthermore, how the small trees would be taken from the forest is not studied. The branches could be a problem for a forwarder, limiting the amount of trees it could take. Making chips of the trees at site is a possibility.

As automation of forest machines proceeds, the two craned model will definitely be useful for harvesting. The simulation shows a productivity increase of almost 200% for the RJ-Head and around 75% increase for the conventional crane head. For this to be reality, development in sensors and computer system has to proceed. Furthermore, the mounting of the cranes have to be fixed.

The performed simulations do compare the different machines, but the resulting output shows a really high correlation with the stand files. In order to get data that can be compared with real data, more tree stands needs to be considered, possibly by a stochastic model or by more real measurements. Since 90% of the stand files were discarded because of bad properties, such as too big trees or too much spruce, an analysis of the forest is recommended. As mentioned above, this new invention does increase productivity really much, but it only works in certain conditions and the question is how many such forests that exists.
4.2 Simulation tool

Every tool has its limitations, and so does this one. SimPy sometimes gives you a hard time tweaking the system into the desired properties, for example a process cannot wait for several things to happen. Although these constraints, I have not yet found a situation that could not be handled with clever programming and use of the existing functions. The choice of discrete event library was good, I cannot see how any of the commercial programs could compete with the flexibility found with this solution. Depending on how far we go with this project, if any limitations in SimPy are found, resources could possibly be allocated to contribute with the development of SimPy.

The limitations found are rather in the discrete event method than in the used libraries. One should not forget that splitting up the time, that in the real world is continuous, into discrete time steps is a step from reality, and not all simulations are suitable for this approach. Determining if the discrete event method is suitable for a simulation takes some skill, and is actually one of the things that limits the user-friendliness of this tool. A person that views this program as a “magic box” will never be able to make this decision, which may result in a simulation that does not match reality and possibly in bad decisions following the simulation result.

A simulation that is dependent on differential equations is not suitable for discrete event modeling, continuous simulations are preferred for that kind of simulations. Differential equations are more common in simulation than one may think at first. If e.g. the crane movements of a machine is really important one ends up in a situation that may be unsuitable for discrete event simulations. The simulation of the planting machine is in the borderland here, since the two cranes are dependent on the position of the other crane. When a crane is moving, only the extreme positions, i.e. the start and end points, are used. This means that the other crane either senses the movement too early or too late. What makes the planting machine simulations accurate is that the two crane heads have their own domains, one covers the right side and one the left side. Irregularities occur in the intersection of these domains, but the overall result should not be seriously affected by this since this intersection is quite small. In the thinning machine case this is not a problem at all, since the two cranes work at their own sides of the road and their domains do not intersect at all.

Support for continuous movements was developed during the last week of the thesis, too late to incorporate it into the simulation model. These algorithms take care of some of the problems, but not all. Since movements still occur in a discrete manner, choosing a movement path has to be done at a certain time instance. To do this, the entity needs to know the position of the other entity, e.g. the other crane, not only at the current time instance, but for the time that the future movements will take. This is not impossible with the current model, but one still ends up in a master-slave situation. Support for this kind of situations requires collision detection for moving objects, something that surely is possible but still does not solve the problem completely, and takes computer power from the rest of the simulation. I am not saying that these kinds of simulations are impossible with the developed tool, but a continuous simulation method with smaller time steps is preferred.

A future software with pre-programmed AI machines that can be released in an arbitrary forest is absolutely possible. The demand of such a software has to be investigated before any bigger steps are taken in this direction. This thesis has assumed that there is a big interest out there, but in reality we only know that SLU are interested. However, the potentials should be quite obvious to forest owner companies and machine constructors.

It has to be pointed out that programming new machines for such a software is something that professionals have to handle. There is no way to build a tool where the user interactively
can specify algorithms for how a machine should work, such an attempt would gravely affect the quality of the simulations. There is a reason that people are still programming. However, a user would certainly be able to interactively change machine and terrain parameters such as speed, fuel consumption, position of trees etc.

There is a lot of work left, and the future will tell how far we will go. I do not see any limitations regarding the chosen path, rather in the discrete event methodology. It covers a certain domain of problems and models, but far from all. But nothing is wrong with that, one cannot put up a universal tool for all simulations. If one knows the limitations and borders of this tool it will prove useful and, if development continues, take discrete event simulations in forestry to a new level.
Bibliography


[14] A. Herlitz. Typestands for clear cutting. Technical report, Skogshögskolan, institutio-
nen för skogsteknik, 1975.

branching structures: axis differentiation and segment tapering in coarse roots of three 

ctl-harvesting on sensitive sites in finland – a simulation study. In Precision Forestry 

[17] Norm Matloff. Introduction to Discrete-Event Simulation and the SimPy Language, 
2006.

[18] Andersson P.-O, H. Berglund, and Bäckström. Simulering av maskinella planteringsor-

[19] Ola Ringdahl, Thomas Hellström, and Ola Lindroos. Potentials of possible machine 
systems for directly loading logs in cut-to-length harvesting. submitted, 2011.


[21] Lars Sängstuvall, Dan Bergström, Tomas Lämås, and Tomas Nordfjell. Simulation 
of harvester productivity in selective and boom-corridor thinning of young forests. 
manuscript.

strength in forest soils. root mapping and plate loading tests in thinning-stage stands 
of norway spruce. Technical report, Skogshögskolan, inst. för växtekologi och marklära, 
1975.

[23] Tony Vignaux, Klaus Muller, and Bob Helmbold. Simulation with SimPy - In Depth 
Manual.

Appendix A

Collision detection.

The model must include some kind of collision detection. It does not have to involve movements and time steps since the events happen in a discrete manner.

The simplest approach would be to represent all objects as either cubes or spheres. It would then be easy to detect if two objects are intersecting each other in the sphere case by simply calculating the distance between their centers and comparing it to the sum of their radii. In the box case the x and y coordinates are simply compared instead. However, a system where objects are represented as both spheres and cubes causes problems, since the two methods above does work for one kind of geometry at a time. Arbitrary polygons causes even larger problems.

Bounding volumes

Besides the above mentioned first approach, every object can be represented with a so called bounding volume that covers its volume. An example in the 2D case can be seen in figure (A.1), where a concave polygon is represented as a circle.

As can be seen, this method should just be used as a first test, since the circle also cover areas that are not within the polygon. However, the method with bounding volumes/areas saves lots of computations since it minimizes the number of more advanced and time consuming tests.

Distance from point to line

To compute the distance from a line segment $AB$ to a point $C$, one can project $C$ down to the line with a dot product and divide the tests into three cases. This procedure is shown in Algorithm (8).

Check if two line segments intersect

For collision detection of polygons it is crucial to have a method for testing if two line segments intersect. The algorithm described here uses the fact that in two dimensions, unless two straight lines are parallel, they have an intersection point. If this point is calculated it can easily be checked if it is outside the respective line segments.

Let us define the two line segments as vectors $AB$ and $CD$. The first line is written in
Algorithm 8 Distance from point to line.

**Require:** A line segment vector $AB$, a vector from $A$ to $C$, $AC$ and the vector $BC$

1: Compute $AB \cdot AC$
2: if $AB \cdot AC \leq 0$ then
3: \hspace{1em} $d \leftarrow AC \cdot AC$ \hspace{1em} // $C$ is closest to point $A$, use Pythagoras theorem:
4: else if $AB \cdot AC \geq AB \cdot AB$ then
5: \hspace{1em} $d \leftarrow BC \cdot BC$ \hspace{1em} // $C$ is closest to point $B$, use Pythagoras theorem.
6: else
7: \hspace{1em} The projected point is on the line segment $AB$, distance is given by $CD \cdot CD$, where $D$ is the closest point. $D$ is given by $D = A + \frac{AC}{AB}$, and can be substituted in the equation above which gives:
8: \hspace{1em} $d \leftarrow AC \cdot AC - \frac{(AC \cdot AB)^2}{AB \cdot AB}$
9: end if
10: return $d$

explicit form and the second line in implicit form:

$$L_1(t) = A + t(B - A) \tag{A.1}$$
$$L_2 : \hspace{1em} n \cdot (X - C) = 0 \tag{A.2}$$
To find the intersection point, equation (A.1) substitutes $X$ in equation (A.2):

\[
\mathbf{n} \cdot (A + t(B - A) - C) = 0 \quad \iff \\
\mathbf{n} \cdot (A - C) + t(\mathbf{n} \cdot (B - A)) = 0 \quad \iff \\
t(\mathbf{n} \cdot (B - A)) = \mathbf{n} \cdot (C - A) \quad \iff \\
t = \frac{\mathbf{n} \cdot (C - A)}{\mathbf{n} \cdot (B - A)} 
\tag{A.3}
\]

where $\mathbf{n}$ is perpendicular to $CD$. From equation (A.1), one can see that the point described by equations (A.1) and (A.3) lies on $L_1$ only if $0 \leq t \leq 1$. Thus, Algorithm (9) is used:

**Algorithm 9** Determining if two line segments intersect in 2D

**Require:** Two line segment vectors $AB, CD$.

1. Check if the lines are parallel:
2. if $AB \cdot CD$ equals $|AB||CD|$ then
3. check if $A$ is on $CD$ with algorithm (8)
4. if $A$ is on $CD$: then
5. return True
6. end if
7. end if
8. $\mathbf{n} \leftarrow CD \times (0, 0, 1)$, where the z coordinate 0 has been added to $CD$
9. $t \leftarrow \frac{\mathbf{n} \cdot (C - A)}{\mathbf{n} \cdot (B - A)}$
10. if $0 \leq t \leq 1$ then
11. return True
12. else
13. return False
14. end if

**Intersecting line segment and sphere**

Let the line be described as:

\[
R(t) = P + td 
\tag{A.4}
\]

where $P$ is the line origin, $d$ is a normalized vector that gives the direction of the line and $0 \leq t \leq t_{\text{max}}$. Let the sphere be defined by:

\[
(X - C) \cdot (X - C) = r^2 
\tag{A.5}
\]

where $C$ is the center of the sphere and $r$ is its radius. By substituting $R(t)$ from equation (A.4) for $X$ in equation (A.5), one gets:

\[
(P + td - C) \cdot (P + td - C) = r^2 
\tag{A.6}
\]

Using $\mathbf{m} = P - C$ and some algebra, one gets:

\[
t^2 + 2(\mathbf{m} \cdot d)t + (\mathbf{m} \cdot \mathbf{m}) - r^2 = 0 
\tag{A.7}
\]

A second degree polynomial is now derived for $t$. It has three different outcomes: one double root, two Real roots and two complex roots. If there is a double root, the line hits
the tangent of the sphere. In the case of two Real roots, the line penetrates the sphere and has two points of connection. If the roots are complex, no intersection occurs.

Observe that this is for an infinite line. In order for this test to work for a line segment, one can check if $0 \leq t \leq 0$, which is demanded by equation (A.4).

Algorithm 10 can be used for the test.

---

**Algorithm 10 Intersecting line segment and sphere**

**Require:** A line segment vector $AB$ and a sphere central position $C$ and its radius $r$.

1. Determine $t_{max}$ and $d$ from equation (A.4).
2. calculate the values of $t_i$ from equation (A.7).
3. if $t_1$ and $t_2$ are complex then
4. Return False
5. else if $0 \leq t_1 \leq t_{max}$ or $0 \leq t_2 \leq t_{max}$ then
6. return True
7. else
8. return False
9. end if
Overall algorithm for collision detection

Algorithm 11 2D collision detection

Require: Two objects \( o_1 \) and \( o_2 \), with central positions and bounding radii, \( r_i \). If the objects are polygons, they have a list of the nodes in the polygon.

1: calculate the distance between the object centers, \( d \).
2: if \( d < r_1 + r_2 \) then
3: return True
4: end if
5: if \( o_1 \) and \( o_2 \) are circular then
6: return False
7: end if
8: if \( o_1 \) or \( o_2 \) is circular then
9: for edges between object nodes: do
10: use algorithm (10) to decide if the edge intersects with the circle.
11: if edge intersects with circle then
12: return True
13: end if
14: end for
15: check if an arbitrary node of the polygon is inside the circle or vice versa.
16: if \( o_i \) is inside \( o_j \) then
17: return True
18: end if
19: else
20: for edges between nodes in \( o_1 \) do
21: for edges between nodes in \( o_2 \) do
22: Use algorithm (9) to decide if the edges intersect.
23: if The edges intersect then
24: return True
25: end if
26: end for
27: end for
28: check if an arbitrary node of \( o_1 \) is inside \( o_2 \) or vice versa.
29: if \( o_i \) is inside \( o_j \) then
30: return True
31: end if
32: end if
33: return False
Appendix B

Additional pictures and plots
Figure B.1: Results, while varying inner crane radius and terrain. The error bars indicate two standard deviations, i.e. 95% confidence intervals.
Figure B.2: The stump distributions of the different terrain files used for the planting machine simulations. The roots are randomly distributed for each simulation but the stump positions are fixed.
Figure B.3: The tree distributions for the different thinning stands used.

Figure B.4: The average number of trees in corridors and stick roads.