

**Vertrouwelijk: Machine Learning voor het verlagen van  
buffervoorraden**

---



Naam	Studentnummer	Eerste examiner	Bedrijf	Datum
S. A. Rang	1655299	Jos Schmeltz	EKB Houten	22-6-2018

# VOORWOORD

## MANAGEMENTSAMENVATTING

# Inhoudsopgave

<b>Voorwoord</b>	<b>2</b>
<b>Managementsamenvatting</b>	<b>3</b>
<b>Figuren- en Tabellenlijst</b>	<b>6</b>
<b>Afkortingenlijst</b>	<b>6</b>
<b>Begrippenlijst</b>	<b>7</b>
<b>1 Inleiding</b>	<b>8</b>
<b>2 Organisatorische Context</b>	<b>9</b>
2.1 Het bedrijf	9
2.2 Bedrijfsgegevens	10
2.3 Persoonsgegevens	10
<b>3 De Opdracht</b>	<b>11</b>
3.1 De kwestie	11
3.2 De afstudeeropdracht in het kort	11
3.3 Doelstelling	11
3.4 Hoofdvraag en deelvragen	12
3.5 Onderzoeksmethoden	12
<b>4 Theoretisch Kader</b>	<b>13</b>
4.1 Tsubaki Nakashima	13
4.1.1 Rollenfabriek lay-out	13
4.1.2 Buffervoorraden	14
4.1.3 Omstellingen	14
4.2 EKB Manufacturing Intelligence	14
4.2.1 Overall Equipment Effectiveness	14
4.3 Theory of Constraints	16
4.3.1 Overeenkomsten met LEAN	17
4.4 Machine Learning	18
4.4.1 De perceptron	18
4.4.2 Neural Networks	18
4.4.3 Activation Functions	19
4.4.4 Genetic algorithms	20
4.4.5 Neuro Evolution of Augmenting Topologies	21
<b>5 Onderzoek</b>	<b>23</b>
5.1 Data	23
5.1.1 Basis gegevens	23
5.1.2 Data uit EKB Manufacturing Intelligence	23
5.1.3 Uitzonderingen	24
5.1.4 Externe data	24
5.1.5 Conclusie	25
5.2 Simulatie	26

5.2.1	Simulatiesoftware keuze .....	26
5.2.2	JaamSim .....	27
5.2.3	Conclusie.....	29
5.3	Machine learning .....	30
5.3.1	Algoritme types.....	30
5.3.2	Genetic algorithm frameworks .....	30
5.3.3	Conclusie.....	31
5.4	Experimenten .....	32
5.4.1	Benchmarking .....	32
<b>Literatuur .....</b>		<b>35</b>
<b>Bijlagen.....</b>		<b>39</b>
Bijlage A: Plan van Aanpak.....		39
Bijlage B: Simulatiesoftware requirements onderzoek .....		66
Bijlage C: Genetic algorithm framework requirements onderzoek.....		73

## FIGUREN- EN TABELLENLIJST

FIGUUR 1: EKB ORGANOGRAM .....	9
FIGUUR 2: LAY-OUT ROLLENFABRIEK TSUBAKI NAKASHIMA.....	13
FIGUUR 3: EMI MODULES VAN DE ROLLENFABRIEK VAN TSUBAKI NAKASHIMA.....	14
FIGUUR 4: EMI OEE ANALYSE VAN DE ROLLENFABRIEK VAN TSUBAKI NAKASHIMA.....	15
FIGUUR 5: EMI OEE STATUS CATEGORIEËN VAN DE ROLLENFABRIEK VAN TSUBAKI NAKASHIMA.....	15
FIGUUR 6: DE VIJF FOCUSSTAPPEN VAN TOC .....	16
FIGUUR 7: AGILE SOFTWARE DEVELOPMENT CYCLE.....	17
FIGUUR 8: DE VIJF FASEN VAN LEAN .....	17
FIGUUR 9: PERCEPTRON.....	18
FIGUUR 10: PERCEPTRON OUTPUT FORMULE .....	18
FIGUUR 11: NEURAL NETWORK.....	19
FIGUUR 12: STANDAARD SIGMOÏD FORMULE .....	19
FIGUUR 13: PLOT VAN DE SIGMOÏD FORMULE.....	19
FIGUUR 14: PLOT VAN DE STANDAARD RELU .....	20
FIGUUR 15: PLOT VAN DE LEAKY RELU.....	20
FIGUUR 16: BASIS STRUCTUUR EN TERMINOLOGIE VAN GA'S.....	21
FIGUUR 17: AANVULLENDE MUTATIEMOGELIJKHEDEN VAN HET NEAT ALGORITME .....	22
FIGUUR 18: DATA UIT DE EMI DATABASE EN EXTERNE BRONNEN .....	24
FIGUUR 19: JAAMSIM GUI.....	27
FIGUUR 20: JAAMSIM CONTAINER OBJECTEN .....	29
TABEL 1: BEDRIJFSGEGEVENS VAN EKB.....	10
TABEL 2: PERSOONSGEGEVENS VAN BETROKKENEN.....	10
TABEL 3: METHODEN MATRIX .....	12
TABEL 4: SIMULATIESOFTWARE AFWEGING.....	26
TABEL 5: GA-FRAMEWORK AFWEGING .....	31
TABEL 6: JAAMSIM BENCHMARKS.....	32

## AFKORTINGENLIJST

AFKORTING	UITLEG
EKB	Elektro Kasten Bouwen Industriële Automatisering is het afstudeerbedrijf. EKB heeft vestigingen in Houten, Beverwijk, Someren, Drachten en Haaksbergen. De afstudeerder heeft gewerkt bij de vestiging in Houten (voor meer informatie zie hoofdstuk 2.1)
EMI	EKB Manufacturing Intelligence is het software pakket van EKB waarmee gebruikers inzicht krijgen in de productiviteit en kwaliteit van industriële productieprocessen (voor meer informatie zie hoofdstuk 4.2)
GA	Genetic algorithm (voor de definitie zie paragraaf 4.4.4)
ML	Machine learning (voor de definitie zie hoofdstuk 4.4)
NEAT	Neuro Evolution of Augmenting Topologies (voor de definitie zie paragraaf 4.4.5)
OEE	Overall Equipment Effectiveness (voor de definitie zie paragraaf 4.2.1)
TN	Tsubaki Nakashima is een klant van EKB en heeft een rollenfabriek in Veenendaal die als business case voor deze afstudeerstage is gebruikt (voor meer informatie zie hoofdstuk 4.1)
TOC	Theory of Constraints (zie voor de definitie hoofdstuk 4.3)

## BEGRIPPENLIJST

BEGRIJP	DEFINITIE
Buffervoorraden	De voorraad van producten of halffabricaten die staat te wachten tussen twee productielijnen tot ze verder verwerkt kunnen worden (voor meer informatie zie paragraaf 4.1.2)
Constraint	De zwakste schakel van een proces. Dit kan een productielijn, maar ook een beleid of werknemer zijn die de rest van de processen ophoudt.
Fitness	De score van een individu van een population
Population	De bevolking van een GA
Productielijn	Een serie geschakelde verzameling machines waarmee in een fabriek producten worden geproduceerd
Rol	Een stalen cilinder van variabele lengte en diameter die in de rollenfabriek van TN wordt geproduceerd
Stilstand	Een tijdsperiode waarin een productielijn niet kan produceren
Uitval	Geproduceerde producten die zijn afgekeurd en niet verkocht kunnen worden

# 1 INLEIDING



## 2 ORGANISATORISCHE CONTEXT

In dit hoofdstuk wordt het afstudeerbedrijf geïntroduceerd en wordt de rol van de afstudeerder binnen de organisatie beschreven. Daarnaast zijn de bedrijfs- en persoonsgegevens opgenomen.

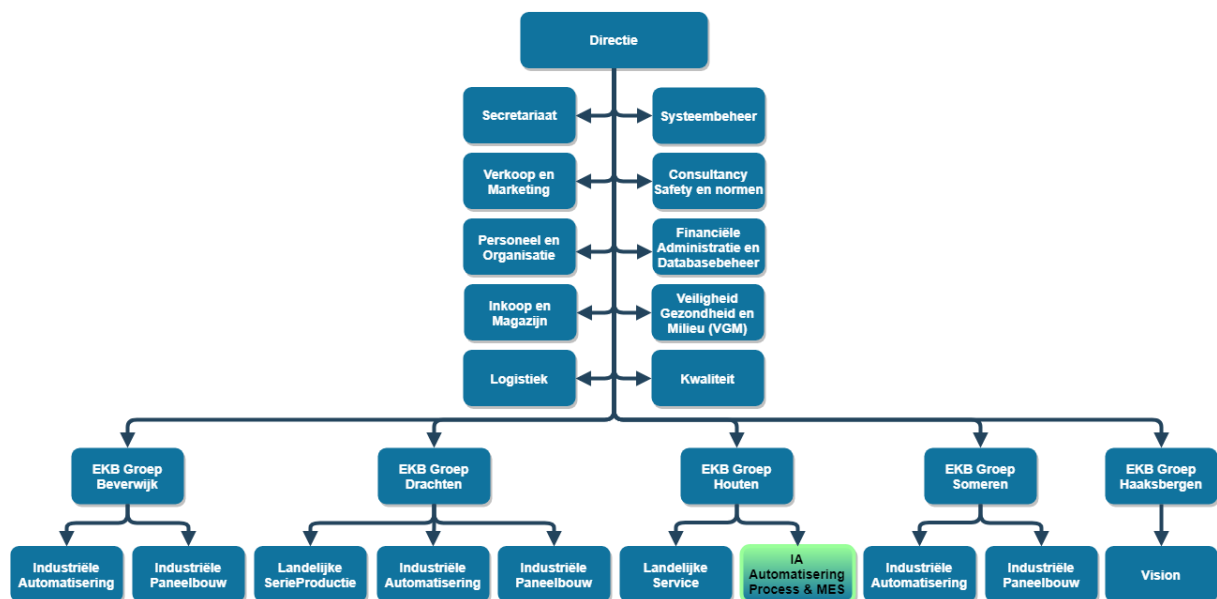
### 2.1 Het bedrijf

Elektro Kasten Bouwen Industriële Automatisering (EKB) is actief op het gebied van industriële automatisering en richt zich vooral op het aaneensluiten en implementeren van processen hierbinnen.

Met 200 medewerkers verdeeld over vijf vestigingen bieden zij automatiseringsoplossingen voor de Nederlandse industrie.

EKB realiseert industriële automatiseringsprojecten voor de Nederlandse eindgebruikers en machinebouwers. EKB is vooral actief in de sectoren metaal, voedingsmiddelen, offshore en fijn chemie.

Het organogram van EKB is weergegeven in Figuur 1. Binnen de organisatie werkt de afstudeerder op de Manufacturing Execution Systems (MES) afdeling van software engineering in Houten. Andere onderdelen van EKB zijn industriële automatisering, industriële paneelbouw, service en vision. Deze onderdelen worden voornamelijk bij de andere vestigingen (Drachten, Someren, Haaksbergen en Beverwijk) tot uitvoer gebracht.



Figuur 1: EKB organogram

Noot. Aangepast van "EKB Groep (Totaal)", door EKB, 2017, 21 februari. Geraadpleegd op 24 mei 2018, van <http://intranet.ekb.nl/Documenten%20Personeelszaken/Organogrammen%20EKB%20Groep.pdf>

## 2.2 Bedrijfsgegevens

Tabel 1: Bedrijfsgegevens van EKB

NAAM	ADRES	POSTCODE	TELEFOON	WEBSITE
EKB	Meidoornkade 19	3992 AG Houten	+31 30 711 14 80	<a href="http://www.ekb.nl/nl/home/">http://www.ekb.nl/nl/home/</a>

## 2.3 Persoonsgegevens

Tabel 2: Persoonsgegevens van betrokkenen

NAAM	FUNCTIE	E-MAILADRES	TELEFOON	LINKEDIN
S. A. Rang	Afstudeerder	<a href="mailto:stefan.rang@student.hu.nl">stefan.rang@student.hu.nl</a>	+31 6 34 10 04 29	<a href="https://www.linkedin.com/in/stefan-rang-8b0635101/">https://www.linkedin.com/in/stefan-rang-8b0635101/</a>
J. Schmeltz	Eerste examiner	<a href="mailto:jos.schmeltz@hu.nl">jos.schmeltz@hu.nl</a>	onbekend	onbekend
F. Verbruggen	Docentbegeleider/ Tweede examiner	<a href="mailto:frank.verbruggen@hu.nl">frank.verbruggen@hu.nl</a>	+31 6 12 20 22 74	<a href="https://www.linkedin.com/in/frank-verbruggen-5080a720/">https://www.linkedin.com/in/frank-verbruggen-5080a720/</a>
A. Roelofsen	Bedrijfsbegeleider	<a href="mailto:a.roelofsen.ekb.nl">a.roelofsen.ekb.nl</a>	+31 6 20 96 48 14	<a href="https://www.linkedin.com/in/auke-roelofsen-273b7918/">https://www.linkedin.com/in/auke-roelofsen-273b7918/</a>
M. de Lange	Product owner	<a href="mailto:m.de.lange@ekb.nl">m.de.lange@ekb.nl</a>	+31 6 51 83 97 90	<a href="https://www.linkedin.com/in/michiel-de-lange-a1b04b3/">https://www.linkedin.com/in/michiel-de-lange-a1b04b3/</a>
G. Bargeman	Contactpersoon Tsubaki Nakashima	<a href="mailto:ger.bargeman@europe.tsubaki-nakashima.com">ger.bargeman@europe.tsubaki-nakashima.com</a>	+31 6 24 36 58 85	<a href="https://www.linkedin.com/in/ger-bargeman-b3332a14/">https://www.linkedin.com/in/ger-bargeman-b3332a14/</a>
M. Kok	Software engineer	<a href="mailto:m.kok@ekb.nl">m.kok@ekb.nl</a>	+31 6 12 60 62 73	<a href="https://www.linkedin.com/in/maarten-kok-316374109/">https://www.linkedin.com/in/maarten-kok-316374109/</a>

## 3 DE OPDRACHT

Een afstudeeropdracht ontstaat vaak uit een probleem of te benutten kans van een bedrijf. In het geval van deze afstudeeropdracht gaat het om een probleem van een klant van EKB.

### 3.1 De kwestie

Sinds 2009 ontwikkelt EKB een eigen softwarepakket genaamd EKB Manufacturing Intelligence (EMI), gericht op industriële toepassing. EMI is vooral bedoeld om inzicht te krijgen in de productiviteit en kwaliteit van industriële productieprocessen. Deze data worden op dit moment voornamelijk gebruikt om een overzichtelijk beeld te krijgen van de huidige situatie, maar nog niet om bepaalde productieprocessen te optimaliseren.

Vanaf de start van de ontwikkeling van EMI is er vanuit de industrie aangegeven dat er in toenemende mate beheer en sturing van interne buffervoorraden gewenst wordt. Hierbij wordt geëist dat Theory of Constraints (TOC) wordt toegepast in combinatie met machine learning (ML).

Als business case voor deze afstudeeropdracht wordt de data van de rollenfabriek van Tsubaki Nakashima (TN), een klant van EKB, te Veenendaal gebruikt. De buffervoorraden worden daar momenteel beheerd op basis van ervaring. Het is niet duidelijk hoe de buffervoorraden tussen de productielijnen zo afgestemd kunnen worden dat de voorraden verminderen terwijl de productie vergroot wordt.

### 3.2 De afstudeeropdracht in het kort

Het verlagen van de buffervoorraden zorgt indirect voor kostenvermindering. Volgens Goldratt en Cox (2007) resulteert het verlagen van de voorraden echter alleen in een verhoging van de winst als ook de productie verhoogd wordt. Om dit te bereiken moeten dus zowel de verlaging van de buffervoorraden als de verhoging van de productie even zwaar meetellen.

Naast een verbredend en kritisch onderzoek naar ML en de huidige situatie bij TN welke EKB van de student eist, dient de student een werkend product op te leveren waarin daadwerkelijk ML toegepast en aantoonbaar gemaakt is voor de gebruiker. De afstudeerstage betreft dan ook een productopdracht.

Omdat het onduidelijk is hoe zowel de buffervoorraden te verlagen als de productie te verhogen zal ML worden gebruikt om een algoritme tot stand te laten komen te op basis van een fabriekssimulatie de hoogtes van de buffervoorraden af kan wegen. Om dit te kunnen implementeren in EMI moet echter wel eerst worden onderzocht hoe de simulatie opgezet kan worden zodat deze generiek gebruikt kan worden voor andere klanten van EKB.

### 3.3 Doelstelling

De te bouwen uitbreiding van EMI moet ervoor gaan zorgen dat klanten van EKB hun buffervoorraden zo laag mogelijk kunnen houden terwijl de productie zo hoog mogelijk is.

Middels de beschikbare data in EMI, de simulatie en het ML algoritme moet het mogelijk worden de doorlooptijden van halffabricaten te verminderen. Hierdoor worden de voorraden tussen de verschillende productielijnen kleiner en verminderen de kosten per product.

### 3.4 Hoofdvraag en deelvragen

Uitgaande van de opdrachtschrijving van EKB is de volgende hoofdvraag geformuleerd:

***Hoe kunnen machine learning algoritmes, gericht op Theory of Constraints, in EMI worden geïmplementeerd om de buffervoorraden van EKB klanten te verminderen?***

#### Hoofdvraag decompositie

Om een ML algoritme te kunnen trainen zijn er allereerst een of meerdere relevante datasets nodig. Het is dus van belang dat er eerst onderzocht wordt welke data er nodig zijn. Om het eindproduct onafhankelijk van andere software te houden moet de data zoveel mogelijk uit de EMI database komen.

Om de ML algoritmes in de praktijk toe te kunnen passen moet de training worden gedaan in een simulatie. Hiervoor zullen meerdere simulatie-softwarepakketten worden vergeleken op requirements.

Na de implementatie kunnen de simulaties gestart worden. Deze simulaties resulteren in een verzameling van buffervoorraadgroottes die vergeleken kunnen worden met de huidige situatie bij TN om erachter te komen of de algoritmes een verbetering hebben opgeleverd.

De hieruit voortvloeiende deelvragen zijn als volgt:

- 1 Welke data uit EMI en externe data zijn er nodig om simulaties uit te kunnen voeren?
- 2 Welk simulatie-softwarepakket is het meest geschikt om in EMI te implementeren?
- 3 Welke machine learning algoritmes zijn het meest geschikt?
- 4 Voldoen de machine learning algoritmes aan de verwachtingen van EKB?

### 3.5 Onderzoeksmethoden

Per deelvraag is in Tabel 3 vastgesteld welke onderzoeksmethoden gebruikt zullen worden.

Tabel 3: Methoden matrix

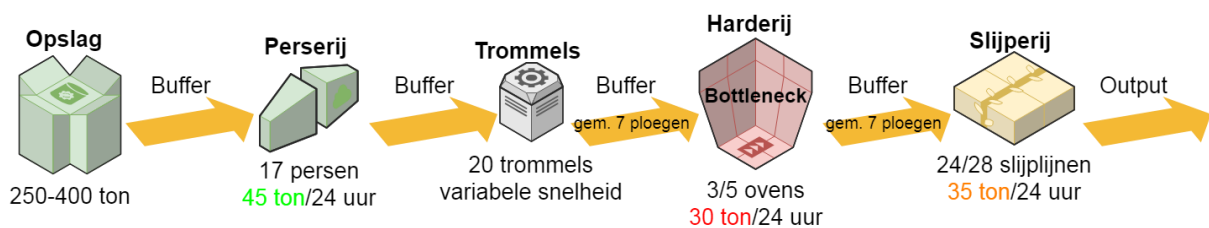
DEELVRAAG	KWALITATIEF OF KWANTITATIEF	ONDERZOEKSMETHODE	RESULTAAT
Welke data uit EMI en externe data zijn er nodig om simulaties uit te kunnen voeren?	Kwantitatief	Deskresearch	Verzameling van data waar de algoritmes op kunnen trainen
Welk simulatie-softwarepakket is het meest geschikt om in EMI te implementeren?	Kwalitatief	Exploratief onderzoek / vergelijkend onderzoek	Een simulatie-softwarepakket
Welke machine learning algoritmes zijn het meest geschikt?	Kwalitatief	Exploratief onderzoek / vergelijkend onderzoek	Een of meerdere machine learning algoritmes
Voldoen de machine learning algoritmes aan de verwachtingen van EKB?	Kwalitatief	Experimenteel onderzoek	Proof of concept

## 4 THEORETISCH KADER

Dit hoofdstuk is bedoeld om meer inzicht te geven in onderwerpen die in dit onderzoek veelvoorkomend zijn. Zowel voorkennis als vergaarde kennis uit het vooronderzoek is hierin verwerkt.

### 4.1 Tsubaki Nakashima

Deze afstudeeropdracht is uitgevoerd met de data van de rollenfabriek van TN te Veenendaal. Deze rollenfabriek maakt stalen cilindrische lagers voor onder andere auto onderdelen en hydraulische apparaten en is onderverdeeld in vijf hallen zoals te zien is in Figuur 2. Dit figuur is het resultaat van meerdere gesprekken met de contactpersoon bij TN. Hieronder volgt een samenvatting van deze gesprekken (G. Bargeman, persoonlijke communicatie, 14 februari 2018, 22 februari 2018 en 2 maart 2018).



Figuur 2: Lay-out rollenfabriek Tsubaki Nakashima

#### 4.1.1 Rollenfabriek lay-out

In de eerste hal wordt het staal in haspels opgeslagen met een totale capaciteit tussen de 250 en 400 ton staalhaspels.

In de tweede hal worden de staalhaspels in de zeventien parallelle persmachines geladen waar ze worden geperst in kleine cilinders met een gemiddelde snelheid van 45 ton per 24 uur. Elke persmachine heeft haar eigen marges met betrekking tot de lengte en diameter van deze cilinders en staalhaspels. Vanaf dit moment spreekt TN van 'rollen' in plaats van staalhaspels en cilinders. De rollen worden na de perserij vervoerd per 500 kg in containers waarvan de rollenfabriek er in totaal 475 bezit.

In de derde hal worden de rollen geschuurd in twintig zogeheten 'trommels' die net als de persmachines parallel draaien. Een trommel verwerkt per keer een batch van 500 kg rollen (een container per keer). De snelheid waarmee getrommeld wordt is variabel en afhankelijk van de lengte en diameter van de rol en de kwaliteit van de voorgaande persmachine. Deze twintig trommels zijn tot nu toe altijd in staat geweest de buffervoorraden tussen de persen en de trommels laag te houden en vormen dan ook geen probleem voor TN.

De vierde hal bevat vijf ovens waar de geschuurde rollen gehard worden op hoge temperatuur. Ook dit gebeurt per container van 500 kg. Ondanks dat de ovens niet serie geschakeld zijn en bijna elk type rol kunnen verwerken, zijn dit toch de constraints van de rollenfabriek vanwege de lage productiesnelheid. Daarnaast wordt er maar van drie van de vijf ovens data geregistreerd door EKB.

In de vijfde en laatste hal worden de containers geleegd in de slijperij en worden de rollen in de definitieve vorm geslepen door 28 slijplijnen. Deze productielijnen bestaan uit een aantal verschillende machines die serie geschakeld zijn, maar de slijplijnen als geheel zijn parallel geschakeld. Ook hebben de slijplijnen net als in de perserij ieder een marge voor bepaalde rollen.

#### 4.1.2 Buffervoorraden

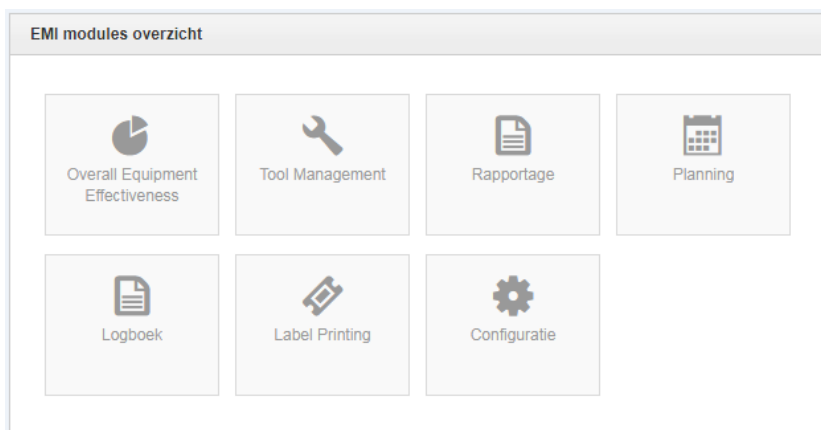
Tussen de hallen bevinden zich de buffervoorraden. Dit zijn voorraden van containers met rollen die nog verwerkt moeten worden door de volgende hal. In het geval van de buffervoorraden tussen de opslag en de perserij zijn dit echter geen containers maar staalhaspels. De buffervoorraden voor en na de harderij zijn in de huidige situatie gemiddeld zeven ploegen groot. Een ploeg is een maatstaf van TN die gebruikt wordt om aan te geven dat een bepaalde hoeveelheid rollen er acht uur over doet om verwerkt te worden in de slijperij. De gemiddelde buffervoorraad rollen die voor en na de harderij aanwezig zijn duren dus 56 uur om verwerkt te worden door de slijperij.

#### 4.1.3 Omstellingen

Bepaalde productielijnen kunnen meerdere verschillende producten produceren. Door een productielijn om te stellen kan er een ander product worden geproduceerd. Een omstelling kan echter uren duren. Het is daarom voor TN van groot belang dat het aantal omstellingen minimaal is.

### 4.2 EKB Manufacturing Intelligence

EKB Manufacturing Intelligence is de applicatie waarin de software van deze afstudeeropdracht geïmplementeerd is. Het bevat de modules Overall Equipment Effectiveness (OEE), Tool Management, Rapportage, Planning, Logboek, Label Printing en Configuratie zoals te zien in Figuur 3.



Figuur 3: EMI modules van de rollenfabriek van Tsubaki Nakashima  
Noot. Rang, S. A. (2018, 3 april). EMI modules van de rollenfabriek van Tsubaki Nakashima [Foto], Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/Default.aspx](http://emi-demo.ekb.nl/EMI_NNN/Default.aspx)<sup>1</sup>

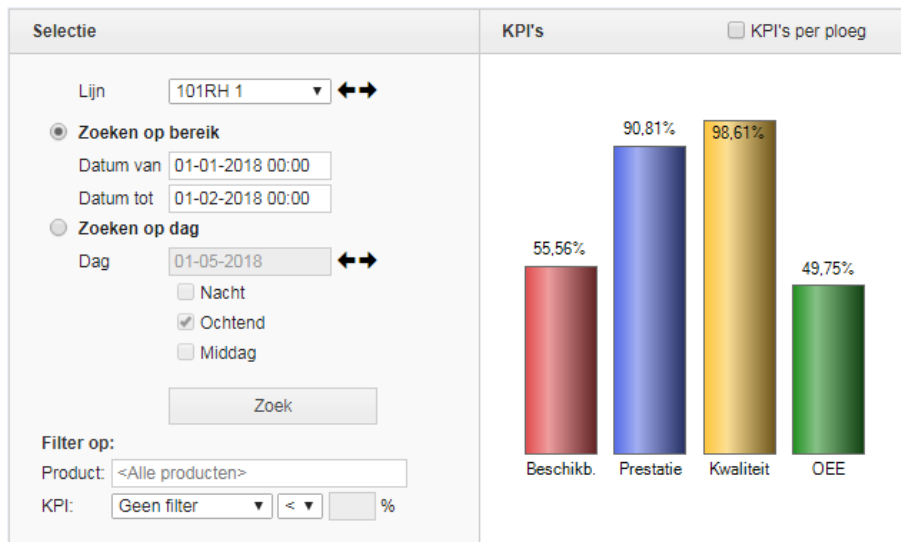
EMI communiceert met een SQL Server database waarin voor elke klant van EKB data voor deze zeven modules gestandaardiseerd worden bijgehouden. Deze standaardisatie heeft bijgedragen aan het generiek houden van de software.

#### 4.2.1 Overall Equipment Effectiveness

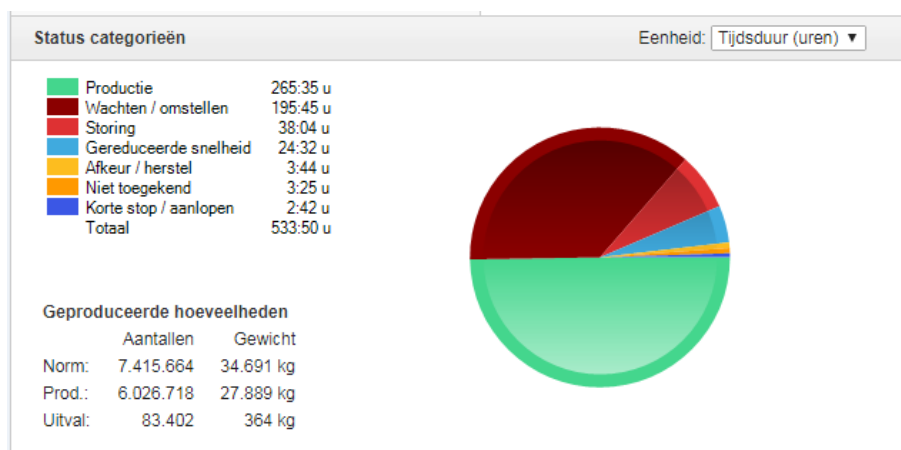
In Figuur 4 is een voorbeeld van de OEE analyse in EMI te zien van een persmachine genaamd '101RH1'. De rode staaf van de KPI's is de beschikbaarheid van de machine, oftewel het percentage van de geselecteerde tijdsperiode dat er daadwerkelijk is geproduceerd. In Figuur 5 zijn de status categorieën te zien. De beschikbaarheid KPI wordt berekend door de status categorieën 'wachten/omstellen', 'storing' en 'niet toegekend' te delen door het totaal aantal uren en dit percentage van 100% af te trekken. Ook is er nog een onzichtbare status categorie 'Uit bedrijf'. Bij TN draait alleen de harderij in het weekend en zijn de perserij en de slijperij in het weekend uit bedrijf. Deze 'Uit bedrijf'-tijd wordt niet bij de status categorie 'totaal' meegerekend.

<sup>1</sup> Bron afkomstig van de EMI software (niet publiekelijk toegankelijk) van EKB.

De blauwe KPI-staaf is de prestatie van de machine. Een prestatie van 50% betekent dat de machine gemiddeld op de helft van de norm-snelheid heeft gedraaid gedurende de tijd dat de machine niet stil stond. De status categorieën 'gereduceerde snelheid' en 'korte stop / aanlopen' vallen onder de prestatie KPI.



Figuur 4: EMI OEE analyse van de rollenfabriek van Tsubaki Nakashima  
Noot. Rang, S. (2018, 3 april). EMI OEE analyse van de rollenfabriek van Tsubaki Nakashima [Foto], Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/OEE/OEEAnalyse.aspx](http://emi-demo.ekb.nl/EMI_NNN/OEE/OEEAnalyse.aspx)<sup>2</sup>



Figuur 5: EMI OEE status categorieën van de rollenfabriek van Tsubaki Nakashima  
Noot. Rang, S. (2018, 3 april). EMI OEE analyse van de rollenfabriek van Tsubaki Nakashima [Foto], Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/OEE/OEEAnalyse.aspx](http://emi-demo.ekb.nl/EMI_NNN/OEE/OEEAnalyse.aspx)<sup>2</sup>

De gele KPI-staaf is de kwaliteit van productie op de machine. Deze KPI geeft het percentage aan van de hoeveelheid geproduceerde producten die niet behoren tot de 'afkeur/herstel'-status categorie.

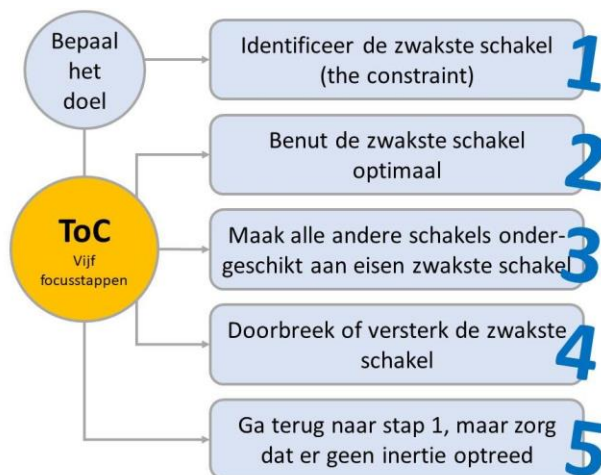
Tot slot worden de beschikbaarheid-, prestatie- en kwaliteit- KPI's respectievelijk vermenigvuldigd. Dit resulteert in de OEE-KPI (de groene staaf). De OEE kan ook worden berekend door de status categorie 'productie' te delen door de status categorie 'totaal' (ook weer exclusief de 'Uit bedrijf' status categorie).

<sup>2</sup> Bron afkomstig van de EMI software (niet publiekelijk toegankelijk) van EKB.

### 4.3 Theory of Constraints

In 2000 schreef Goldratt het boek *'Het Doel'* waarin de basisprincipes van TOC op spannende wijze worden verteld. Volgens een samenvatting van Goldratt en Cox (2007) gaat dit boek over een manager die zijn fabriek wil verbeteren. De manager komt erachter dat lokale verbeteringen irrelevant zijn en dat alleen verbeteringen aan de constraint effect hebben op de prestatie van de fabriek. Alle andere processen zijn afhankelijk van de constraint.

Volgens Procesverbeteren.nl (2017) is TOC, net als LEAN, een logistieke verbetermethode om kortere doorlooptijden te bereiken. TOC wordt vaak samengevat in de formule:  $T = I - OE$ . Hierbij staat 'T' voor 'Throughput', oftewel de doorlooptijd. De 'I' staat voor voorraad (inventory). Inventory bestaat in de context van TOC uit alle investeringen in dingen die een bedrijf later wil verkopen. 'OE' zijn de 'Operational Expenses', de kosten die een bedrijf maakt om inventory om te zetten in throughput. TOC kan worden toegepast in vijf stappen (Figuur 6).



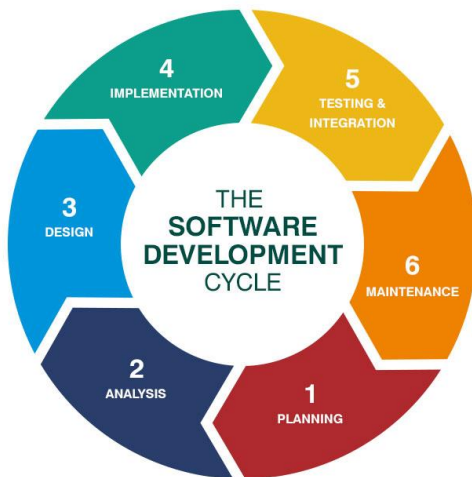
Figuur 6: De vijf focusstappen van TOC

Noot. Herdrukt van "Theory of Constraints: Goldratt", door Managementmodellensite.nl, (2018). Geraadpleegd van [https://managementmodellensite.nl/theory-constraints-goldratt/#.WxY\\_VO6FPRZ](https://managementmodellensite.nl/theory-constraints-goldratt/#.WxY_VO6FPRZ)

De eerste stap van TOC is het identificeren van de zwakste schakel, de constraint. In het geval van software development is dit de implementatie (Figuur 7). Dit proces duurt het langst en houdt de processen testen & integratie en onderhoud op.

Stap twee is het optimaal benutten van de constraint. Volgens Goldratt en Cox (2007) kan dit door meer personeel aan te nemen, een nieuw beleid voor bijvoorbeeld pauzes en door de constraint altijd te bemannen. Dit laatste is meestal al van kracht bij software development. Het vinden van goede software developers is in Nederland echter nog steeds een lastige taak. Steeds meer bedrijven zijn bereid flink te investeren in detachering of recruitment om aan nieuw personeel te komen.





Figuur 7: Agile software development cycle

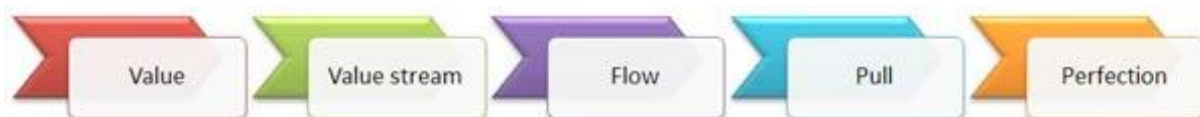
Noot. Herdrukt van "What Is the Software Development Life Cycle?", door Hussung, T., (2016, 10 maart). Geraadpleegd van <https://online.husson.edu/software-development-cycle/>

Stap drie, het ondergeschikt maken van alle andere schakels aan de eisen van de zwakste schakel. Anders gezegd, alle andere schakels helpen de zwakste schakel. In de software development cyclus zou dit kunnen worden bereikt door de SCRUM-projecttechniek toe te passen waarbij scrummeetings centraal staan. Ook wordt hiermee stap vier van TOC toegepast, het doorbreken of versterken van de zwakste schakel. Volgens Goldratt en Cox (2007) kan de doorlooptijd worden verminderd door het werk op te delen in stukken; sprints in deze context.

Tot slot worden alle stappen herhaald voor de nieuwe constraint die is ontstaan. Het zou kunnen zijn dat de analyse-fase in het slop is geraakt omdat de focus volledig op de implementatie-fase is gericht. Wel moet er worden voorkomen dat er inertie optreedt, oftewel dat er altijd volgens een vaste werkwijze wordt gewerkt en invloeden van buitenaf geen effect hebben hierop. Een organisatie moet altijd rekening blijven houden met haar klanten en de werkwijze hierop aanpassen.

#### 4.3.1 Overeenkomsten met LEAN

Goldratt heeft zijn TOC gebaseerd op LEAN. Er zijn dan ook een aantal overeenkomsten tussen TOC en LEAN op te noemen. Volgens LeanSixSigma (2018) gaat LEAN, net als TOC om het constant verbeteren van processen. LEAN heeft ook een vijfstappenplan welke steeds worden herhaald (Figuur 8).



Figuur 8: De vijf fasen van LEAN

Noot. Herdrukt van "Wat is Lean?", door LeanSixSigma, (2018). Geraadpleegd van <https://www.sixsigma.nl/wat-is-lean>

De definities van de vijf fasen van LEAN zijn volgens Procesverbeteren.nl (2017):

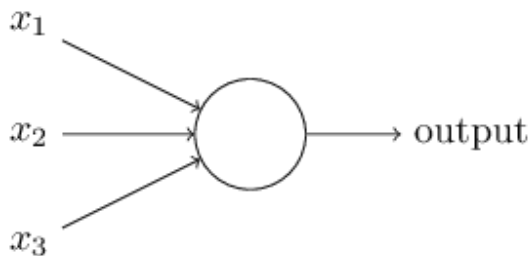
1. 'Value', het identificeren welke producten en diensten de klant van waarde vindt
2. 'Value stream', het identificeren van bedrijfsprocessen met toegevoegde waarde en het elimineren van verliezen die van invloed zijn op de doorlooptijd
3. 'Flow', het zorgen voor stroming en betere doorlooptijden. Hiervoor kan TOC worden toegepast.
4. 'Pull', het vraaggestuurd maken van de productie. Bij TOC heet dit de 'drum-buffer-rope' (Goldratt & Cox, 2007).
5. 'Perfection', net als bij TOC wordt er steeds teruggekeerd naar fase 1 met het doel om perfectie te bereiken.

## 4.4 Machine Learning

ML is een tak van Artificial Intelligence die gebruikt is om TOC te implementeren in EMI. ML wordt al sinds de jaren vijftig gebruikt en is sindsdien uitgegroeid tot een onmisbare technologie voor techgiganten als Google, Microsoft en Apple.

### 4.4.1 De perceptron

Aan de basis van ML staan zogeheten Neural Networks. Dit zijn simpele netwerken die outputs berekenen door het vermenigvuldigen van inputs. De meest simpele vorm hiervan is een perceptron. Volgens Nielsen (2017) kan een perceptron een output berekenen van 0 (uit) of 1 (aan) door een aantal berekeningen te doen op de inputs. In Figuur 9 wordt een perceptron weergegeven.  $x_1$ ,  $x_2$  en  $x_3$  zijn de drie inputs van deze perceptron.



Figuur 9: Perceptron

Noot. Herdrukt van "Using neural nets to recognize handwritten digits", door Nielsen, M., (2017, december). Geraadpleegd van <http://neuralnetworksanddeeplearning.com/chap1.html>

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

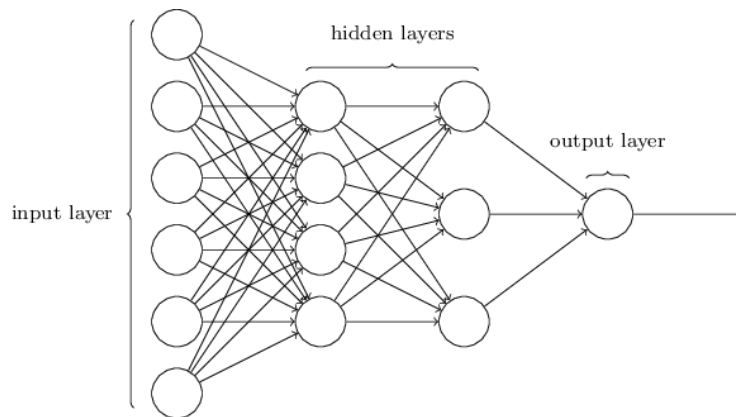
Figuur 10: Perceptron output formule

Noot. Herdrukt van "Using neural nets to recognize handwritten digits", door Nielsen, M., (2017, december). Geraadpleegd van <http://neuralnetworksanddeeplearning.com/chap1.html>

De output van de perceptron is te berekenen met de formule van Figuur 10. Elke lijn van de inputs van Figuur 9 is een 'weight' ( $w$  in Figuur 10). Dit is in essentie een getal, meestal tussen de -5 en 5 waarmee de input vermenigvuldigd wordt. Alle drie inputs worden vermenigvuldigd met de bijbehorende weight. Deze drie uitkomsten worden bij elkaar opgeteld. Als de uitkomst hiervan boven de threshold ligt, is de output van de perceptron 1 en anders 0. De threshold is een nummer die per perceptron anders kan zijn en is een parameter van de perceptron. De 'j' in de formule van Figuur 10 is het nummer van de input  $x$ .

### 4.4.2 Neural Networks

Een neural network is een netwerk bestaande uit lagen van perceptrons. In Figuur 11 is te zien dat een neural network bestaat uit een input laag, een of meerdere verborgen lagen en een output laag van perceptrons. De perceptrons van de input laag krijgen hun inputs van de buitenwereld. Deze inputs kunnen dus worden gezien als een vector die als input van het neural network dient. Om de output van het neural network te berekenen worden per laag de outputs van de betreffende perceptrons berekend. De outputs van de eerste laag worden de inputs van de perceptrons in de tweede laag enzovoort. Uiteindelijk resulteert dit in een of meerdere outputs van het neural network, afhankelijk van het aantal perceptrons in de laatste laag.



Figuur 11: Neural Network

Noot. Herdrukt van "Using neural nets to recognize handwritten digits", door Nielsen, M., (2017, december). Geraadpleegd van <http://neuralnetworksanddeeplearning.com/chap1.html>

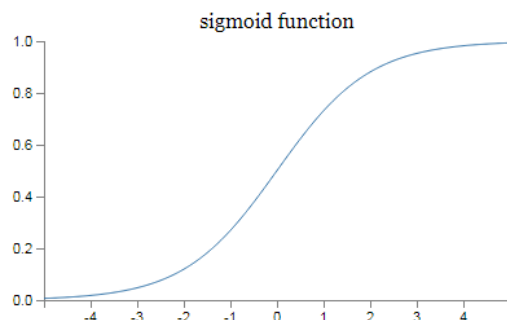
In het geval van neural networks wordt echter vaak de term '*neuron*' gebruikt in plaats van perceptron. Naast de waarde ( $x$ ) en de bijbehorende weights ( $w_n$ ) heeft een neuron ook een *bias*. Dit is een extra variabele die wordt opgeteld bij de output van de neuron. Hierna wordt nog een extra berekening gedaan, de *activation function*.

#### 4.4.3 Activation Functions

In plaats van een threshold heeft een neuron een activation function om de output te berekenen. Een veelvoorkomende activation function is de *sigmoid* (Figuur 12). Het resultaat van de standaard sigmoid formule ligt altijd tussen de 0 en 1 zoals te zien is in de plot van Figuur 13. Een neuron kan door de activation function ook gedeeltelijk aan staan, wanneer de output niet exact 0 of 1 is.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Figuur 12: Standaard sigmoid formule

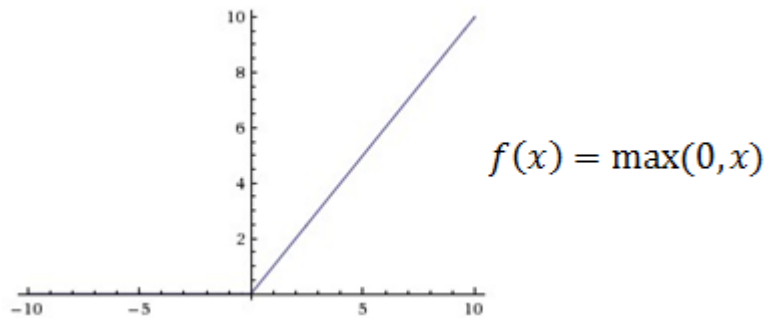


Figuur 13: Plot van de sigmoid formule

Noot. Herdrukt van "Using neural nets to recognize handwritten digits", door Nielsen, M., (2017, december). Geraadpleegd van <http://neuralnetworksanddeeplearning.com/chap1.html>

Volgens Rojas (1996) is het noodzakelijk om een functie te gebruiken met een gelijkmatige helling om te zorgen dat het ML algoritme de weights van de neurons ook gelijkmatig kan veranderen en zo langzaam bij het gewenste resultaat kan komen.

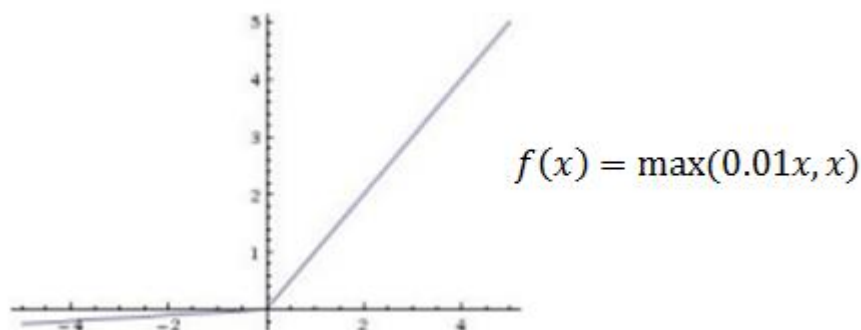
Een andere activation function die steeds vaker gebruikt wordt is de *Rectified Linear Unit (ReLU)*. Het resultaat van de ReLU formule is 0 voor alle negatieve waarden en verandert niets aan de positieve waarden (Figuur 14).



Figuur 14: Plot van de standaard ReLU

Noot. Aangepast van "Rectified-Linear unit Layer", door Santos, L., (2018). Geraadpleegd van [https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/relu\\_layer.html](https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/relu_layer.html)

In 2017 onderzocht Sharma de voor- en nadelen van de ReLU formule. Een belangrijk voordeel is dat de negatieve waarden niet doorberekend hoeven te worden, omdat ze altijd 0 worden. Dit is van positieve invloed op de snelheid waarmee het neural network berekend wordt. Een nadeel van de ReLU formule is dat de formule geen helling heeft voor negatieve waarden. De helling van de activation function is de reden dat neural networks langzaam bij het beoogde doel komen. Zodra een neuron een negatieve output heeft, is de helling van de ReLU formule 0 en verandert de output van de neuron niet meer. Volgens Sharma (2017) heet dit het '*stervende ReLU probleem*'. De neurons die dit probleem ervaren zijn niet meer van nut voor het neural network. Door een aangepaste ReLU formule te gebruiken kan dit worden opgelost. Deze formule heet de *leaky ReLU* (Figuur 15) en is voor positieve waarden hetzelfde als de standaard ReLU. Voor negatieve waarden is de helling niet 0 maar 0,01. Hiermee is het stervende ReLU probleem verholpen.



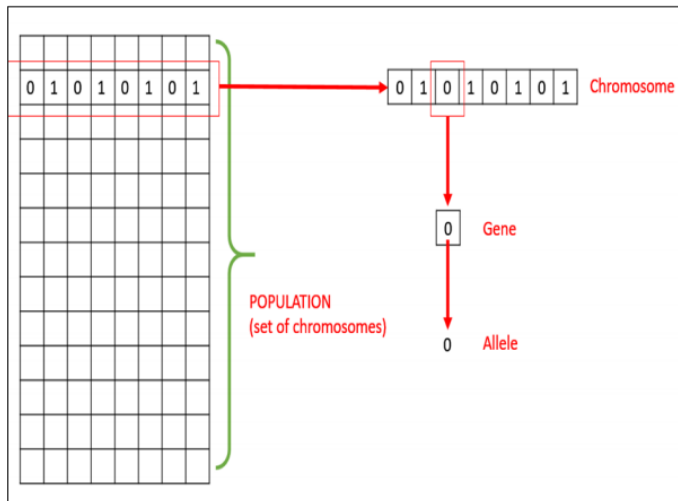
Figuur 15: Plot van de leaky ReLU

Noot. Aangepast van "Deep Learning Class #1 – Go Deep or Go Home", door Monier, L., (2016). Geraadpleegd van <https://www.slideshare.net/holbertonschool/deep-learning-keynote-1-by-louis-monier>

Nog een andere activation function is de hyperbolische tangens  $f(x) = \tanh(x)$  wat eigenlijk een variatie is op de sigmoïd formule, want  $\tanh(x) = 2\sigma(2x) - 1$ . Het verschil is dat de resultaten van deze formule tussen de -1 en 1 liggen en dat de helling twee keer steiler is.

#### 4.4.4 Genetic algorithms

Volgens Tutorials Point (I) Pvt. Ltd.. (2016) is een Genetic Algorithm (GA) een algoritme dat gebaseerd is op natuurlijke selectie in de natuur en is nog steeds een van de beste ML algoritmes. Een GA heeft net als in de natuur een bevolking of '*population*'. Deze population ondergaat recombinaties en mutaties van genen. Deze genetische veranderingen resulteren in een nieuwe generatie van de bevolking. In de terminologie van GA's heten de leden van de bevolking '*chromosomen*'. Deze chromosomen worden ieder op basis van hun resultaten gescoord. Deze score heet de '*fitness*' en wordt gebruikt om te bepalen welke chromosomen reproduceren voor de volgende generatie. In Figuur 16 wordt een overzicht van een GA weergegeven.



Figuur 16: Basis structuur en terminologie van GA's

Noot. Herdrukt van "Genetic Algorithms", door Tutorials Point (I) Pvt. Ltd., (2016). Geraadpleegd van [https://www.tutorialspoint.com/genetic\\_algorithms/genetic\\_algorithms\\_tutorial.pdf](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf)

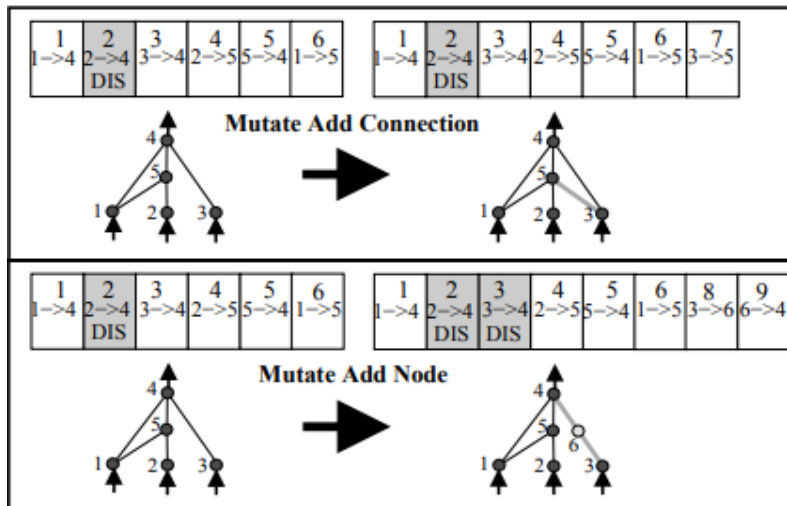
Een voordeel van GA's is dat de hele population parallel kan worden gescoord, omdat het probleem wat opgelost moet worden voor de hele population hetzelfde is. Ook is het eindresultaat van een GA niet een enkele oplossing, maar een hele verzameling van oplossingen die allemaal net iets anders zijn dan de rest. Een GA is echter niet geschikt voor problemen die te simpel zijn of waarvan de oplossing al bekend is. Voor dit soort problemen is een neural network een betere oplossing.

#### 4.4.5 Neuro Evolution of Augmenting Topologies

In 2004 ontwierp Stanley een combinatie tussen een neural network en GA's en noemde dit nieuwe algoritme Neuro Evolution of Augmenting Topologies (NEAT). De recombinaties en mutaties zoals bij GA's gebeurt wordt bij NEAT op dezelfde manier toegepast op de weights van de neural networks. Bij NEAT bestaat de population uit neural networks die kunnen worden gezien als de chromosomen. Naast de recombinaties en mutaties van weights zijn er nog twee manieren van mutatie (Figuur 17).

Ten eerste kan er een nieuwe weight-lijn worden aangemaakt tussen twee neurons die nog niet met een weight-lijn verbonden waren. Dit is mogelijk omdat de neurons van de neural networks in de eerste generatie niet allemaal met elkaar verbonden zijn. Ten tweede kan er een nieuwe neuron worden aangemaakt die een bestaande weight-lijn in tweeën splitst. Deze neuron kan dan in een latere generatie door mutatie weer met andere neurons verbonden worden. Zo worden de neural networks langzaam groter en kunnen ze steeds complexere problemen oplossen.

Stanley (2004) heeft naast de bestaande inspiraties uit de natuur voor GA's nog een nieuwe toepassing. Om ervoor te zorgen dat de variëteit van de neural networks hoog blijft wordt de population gegroepeerd op een vergelijkbare manier als diersoorten. De groepering vindt plaats op basis van het verschil in structuur en weights van de neural networks. De groepen neural networks evolueren onafhankelijk van elkaar waardoor elke groep compleet anders kan zijn. Hierdoor wordt er met NEAT vaak sneller een oplossing gevonden dan met een standaard GA. Dit blijkt ook uit de experimenten van NEAT door K. O. Stanley.



Figuur 17: Aanvullende mutatiemogelijkheden van het NEAT algoritme

Noot. Herdrukt van "Efficient Evolution of Neural Networks through Complexification", door Stanley, K., (2004). Geraadpleegd van <http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf>

## 5 ONDERZOEK

In dit hoofdstuk worden de deelvragen onderzocht en beantwoord. Deze antwoorden dragen bij aan de algemene conclusie van dit onderzoek en resulteren in het beantwoorden van de hoofdvraag:

***Hoe kunnen machine learning algoritmes, gericht op Theory of Constraints, in EMI worden geïmplementeerd om de buffervoorraden van EKB klanten te verminderen?***

### 5.1 Data

Om te kunnen simuleren is er allereerst data over de rollenfabriek van TN nodig. Deze data zou uit EMI, maar ook uit externe bronnen kunnen komen. De eerste deelvraag van dit onderzoek luidt:

*Welke data uit EMI en externe data zijn er nodig om simulaties uit te kunnen voeren?*

#### 5.1.1 Basis gegevens

Om de rollenfabriek van TN te kunnen simuleren zijn een aantal basis gegevens nodig. Dit zijn de:

- Logistiek van de fabriek
- Geproduceerde producten per productielijn met daarbij behorende productiesnelheden
- Omstellingen per productielijn
- Stilstanden per productielijn
- Uitgevallen of afgekeurde producten per productielijn
- Buffervoorraden aan het begin van de simulatie periode

De logistiek van de fabriek is nodig om in de simulaties vast te kunnen leggen welke productielijnen er allemaal zijn en wat de relatie tussen deze productielijnen is. Welke routes kunnen producten allemaal afleggen van het magazijn tot aan de opslag aan het eind van de fabriek?

Als een productielijn meerdere verschillende producten kan produceren kan het voor komen dat deze productielijn moet worden omgesteld voordat er een ander product geproduceerd kan worden. In dat geval zijn er data nodig over deze omstelling zoals wanneer deze omstelling heeft plaats gevonden en hoe lang de omstelling heeft geduurd. Hetzelfde geldt ook voor eventuele stilstanden van een productielijn.

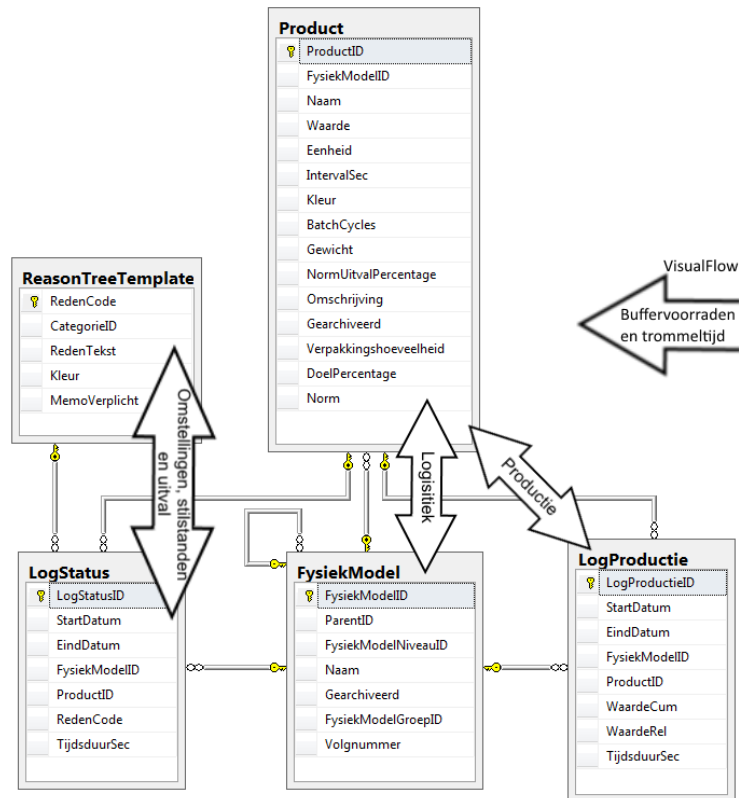
Tenzij de simulaties gestart worden vanaf de in gebruik neming van de fabriek is er ook data nodig over de buffervoorraden aan het begin van de simulatie periode.

#### 5.1.2 Data uit EKB Manufacturing Intelligence

Om de programmatuur toe te kunnen passen bij meerdere klanten van EKB is het van belang zo veel mogelijk gebruik te maken van data uit de database van EMI. Deze database heeft namelijk voor elke klant van EKB dezelfde architectuur. In Figuur 18 zijn de tabellen en velden uit het EMI database diagram weergegeven die data bevatten voor de simulaties.

De tabel 'FysiekModel' bevat de namen en ID's van de productielijnen. Aangezien namen kunnen veranderen en ID's niet, zullen de ID's worden gebruikt. De 'Product' tabel is eigenlijk een koppeltabel tussen de producten en productielijnen. Uit een gesprek met een software engineer van EKB is geconcludeerd dat de producten in de 'Product' tabel alleen te traceren zijn op basis van de naam (M. Kok, persoonlijke communicatie, 27 april 2018). Op basis van de productnamen kunnen in de 'Product' tabel de productielijnen worden opgezocht waar de producten op kunnen worden geproduceerd. De productiesnelheid heet in de database 'norm' en is een kolom van de 'Product' tabel. De productie van de fabriek wordt gelogd in de 'LogProductie' tabel.

De omstellingen, stilstanden en uitval kunnen worden teruggevonden in de 'LogStatus' tabel op basis van de 'RedenCode'. Elke 'RedenCode' heeft een 'CategorieID' in de 'ReasonTreeTemplate' tabel. Voor de omstellingen wordt 'CategorieID' 400 gebruikt, voor stilstanden 300 en voor uitval 700.



Figuur 18: Data uit de EMI database en externe bronnen  
 Noot. Aangepast van “EMI database diagram”, door EKB, 2018, 1 februari. Geraadpleegd van  
 (local)\SQLEXPRESS.EMI\_NN\dbo.DatabaseSchema<sup>3</sup>

### 5.1.3 Uitzonderingen

Uit een gesprek met de LEAN manager en tevens contactpersoon bij TN is gebleken dat niet alle producten door de harderij van de rollenfabriek gaan, maar dat sommigen bij een extern bedrijf worden gehard (G. Bargeman, persoonlijke communicatie, 17 mei 2018). Bij producten waarvan de naam begint met “RT” en later in de naam “HA” of “HN” bevat duurt dit extern harden gemiddeld 4,5 week. Alle andere producten die extern worden gehard doen er een week over.

In het geval dat producten extern worden gehard zijn er in de EMI- en VisualFlow database geen data beschikbaar over het harden van deze producten, dus zal er in de simulaties uit moeten worden gegaan van deze gemiddelde tijden.

Ook zijn er producten die niet door de slijperij van de rollenfabriek hoeven, maar direct na het harden verkocht kunnen worden. De namen van deze producten beginnen allemaal met “RQ”.

### 5.1.4 Externe data

De buffervoorraden worden echter niet in de EMI database bijgehouden. Om de buffervoorraden aan het begin van de simulatieperiode te kunnen simuleren is er dus externe data nodig. In het geval van TN is deze data beschikbaar in een softwarepakket genaamd ‘VisualFlow’ (G. Bargeman, persoonlijke communicatie, 19 februari 2018). Deze software houdt onder andere de locaties van alle containers bij. Een container heeft een ordernummer welke correspondeert met een order van een bepaald product die overeenkomt met een van de productnamen in de EMI database. Hierdoor kan voor elke productielijn aan het begin van de simulatie periode de buffervoorraden opgehaald en berekend worden uit VisualFlow.

Ook de trommels van de rollenfabriek van TN staan niet geregistreerd in de EMI database. Zoals aangegeven in hoofdstuk 4.1.1 zijn de trommels niet van groot belang voor de simulatie. Wel kan de gemiddelde productiesnelheid van de trommels uit de data van VisualFlow berekend worden.

<sup>3</sup> Bron afkomstig van het intranet (niet publiekelijk toegankelijk) van EKB.



### 5.1.5 Conclusie

Deze paragraaf geeft een antwoord op de eerste deelvraag van dit onderzoek:

*Welke data uit EMI en externe data zijn er nodig om simulaties uit te kunnen voeren?*

De EMI database bevat grotendeels de data die nodig zijn voor de simulaties. De relevante tabellen uit deze database zijn weergegeven in Figuur 18 met daarbij welke tabellen welke basis gegevens bevatten. De overige basis gegevens zijn terug te vinden in de database van VisualFlow die in beheer is van TN.

## 5.2 Simulatie

Nu de data bekend zijn kan er worden nagedacht over het simuleren zelf. De resultaten van de ML algoritmes zijn afhankelijk van de kwaliteit van de simulatie. De tweede deelvraag is:

*Welk simulatie-softwarepakket is het meest geschikt om in EMI te implementeren?*

### 5.2.1 Simulatiesoftware keuze

Gezien er beperkte tijd is voor de implementatie fase is er besloten om de simulaties uit te voeren met bestaande software. Om een goede keuze te kunnen maken voor een software pakket dat geïmplementeerd kan worden in EMI zijn er een aantal functionele eisen vastgesteld. De software moet de in ieder geval de basis gegevens uit de vorige deelvraag kunnen verwerken en de volgende onderdelen kunnen simuleren:

- Externe data
- Stilstanden
- Omstellingen
- Uitval of afkeur van producten
- Verschillende producten op verschillende productielijnen
- Buffervoorraden

Het inladen van externe data is nodig om de data uit de EMI- en VisualFlow database te kunnen gebruiken.

In overleg met de bedrijfsbegeleider is vastgelegd dat de simulatiesoftware gratis commercieel te gebruiken moet zijn om in EMI te kunnen implementeren. Daarnaast moet de software open source zijn om toekomstig onderhoud te versimpelen voor EKB (A. Roelofsen, persoonlijke communicatie, 13 april 2018).

Na het vaststellen van deze eisen zijn er verschillende softwarepakketten vergeleken. De gedetailleerde bevindingen hiervan zijn terug te vinden in Bijlage B. Tabel 4 geeft de resultaten hiervan weer.

Tabel 4: Simulatiesoftware afweging

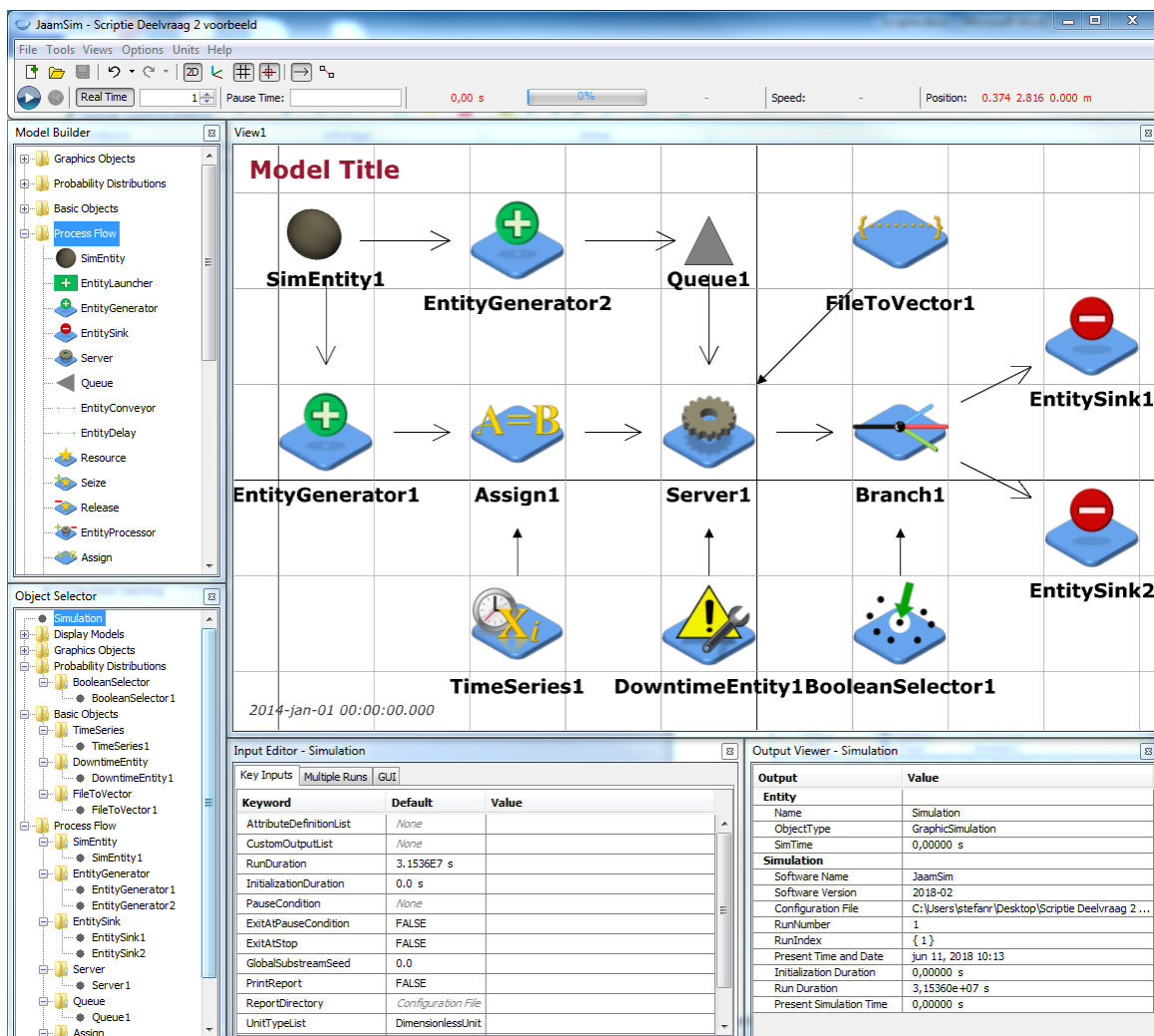
NIET-FUNCTIONELE REQUIREMENTS	ANYLOGIC (2018)	ARENA (2018)	FLEXSIM (2018)	JAAMSIM (2018)	SIMUL8 (2018)
Open source	✗	✗	✗	✓	✗
Commercieel gebruik	✗	✗	✗	✓	✗
FUNCTIONELE REQUIREMENTS					
Externe data	✓	✓	✓	✓	✓
Stilstanden	✓	✓	✓	✓	✓
Omstellingen	✓	✓	✓	✓	✓
Uitval of afkeur van producten	✓	✓	✓	✓	✓
Verschillende producten op verschillende productielijnen	✓	✓	✓	✓	✓
Buffervoorraden	✓	✓	✓	✓	✓

Opvallend is dat er maar een enkel softwarepakket zowel open source als voor commercieel gebruik beschikbaar is. Ook na verder onderzoek van andere softwarepakketten via onder andere capterra.com zijn er geen open source softwarepakketten gevonden die hieraan voldoen. De meeste softwarepakketten hebben echter wel een demo- of academische versie die gratis te gebruiken is, maar zijn niet beschikbaar voor commercieel gebruik.

Softwarepakket JaamSim (2018) voldoet echter aan alle requirements en na overleg met de bedrijfsbegeleider en de product owner is besloten om dit softwarepakket te gebruiken, ondanks dat er geen mogelijkheid geweest is om JaamSim te vergelijken met andere open source software.

## 5.2.2 JaamSim

JaamSim is een gratis open source 'discrete event simulation' softwarepakket. Volgens Allen, Spencer, Gibson (2015) is een discrete event simulation een manier om het gedrag en de prestatie van processen, faciliteiten of systemen te simuleren. JaamSim begon in 2002 en de eerste release verscheen op 6 juli 2016 door H. Harrison en is geschreven in Java. Voor dit onderzoek is de nieuwste versie van 3 mei 2018 gebruikt. JaamSim heeft uitgebreide documentatie, echter is de gebruikershandleiding sinds 7 september 2017 niet meer bijgewerkt.



Figuur 19: JaamSim GUI

Noot. Rang, S. A. (2018, 11 juni). JaamSim GUI [Foto], Geraadpleegd van 'JaamSim (2018-03)'

In Figuur 19 is de GUI van JaamSim (2018) te zien. Het bovenste venster bevat standaard opties zoals het openen en opslaan van simulaties. Deze simulaties worden door JaamSim omgezet tot een config-bestand met de ".cfg"-extensie. Ook is er de optie om de simulatiesnelheid aan te passen of te pauzeren na een bepaalde tijd.

De volgende paragrafen beschrijven hoe de functionele requirements in JaamSim gesimuleerd kunnen worden aan de hand van Figuur 19 en de gebruikershandleiding van JaamSim (JaamSim Development Team, 2017).

#### 5.2.2.1 Productie

In het middelste en grootste venster is een voorbeeldsimulatie te zien waarin de objecten staan waarmee de requirements uit paragraaf 5.2.1 gesimuleerd kunnen worden.

Ten eerste kan de productie worden gesimuleerd door 'Server1'. Dit is een machine die vanaf, in dit geval, 'EntityGenerator1' 'entities' (entiteiten) ontvangt om te produceren. Als de machine al bezig is terwijl er een entity ontvangen wordt, wordt deze entity in de buffervoorraad ('Queue1') van de machine geplaatst totdat de machine klaar is. Hoe lang de machine doet over een entity is een invoerveld. De meeste invoervelden in JaamSim kunnen ook worden gevuld met een expressie. Hiermee kan externe data opgezocht worden zoals ook de troomeltijden uit VisualFlow. Het object 'FileToVector1' kan van een tekstbestand een vector maken met waarden. Dit tekstbestand zou bijvoorbeeld informatie kunnen bevatten over de productiesnelheid van de entities. Deze productiesnelheden kunnen dan worden ingelezen in het invoerveld van de productiesnelheid van object 'Server1' met deze expressie: "[FileToVector1].Value(this.obj.ID) [s]". Waarbij "[s]" de tijdsaanduiding in seconden is en "this.obj.ID" het ID-attribuut van de huidige entity is die op dit moment door de 'Server1' wordt geproduceerd. Dit ID-attribuut wordt door het 'Assign1'-object toegewezen aan de entities die vanaf de EntityGenerator1 komen. Deze ID's veranderen gedurende de simulatie door het 'TimeSeries1'-object waarin een lijst van tijdsaanduidingen staat met daarbij product-ID's. Deze ID's en tijdsaanduidingen kunnen worden ingeladen door een ander 'FileToVector'-object die de EMI-data bevat van de geproduceerde producten. In de GUI staan geen extra FileToVector-objecten om het voorbeeld overzichtelijk te houden.

#### 5.2.2.2 Buffervoorraden aan het begin van de simulatieperiode

Het simuleren van de buffervoorraden die aan het begin van de simulatieperiode aanwezig waren kan simpelweg door een extra 'EntityGenerator'-object toe te voegen en de entities naar de bijbehorende buffer te sturen. Elke EntityGenerator heeft invoervelden voor de hoeveelheid entities die gegenereerd moeten worden en hoe vaak. Deze data kan via een extra 'FileToVector'-object worden ingeladen vanuit een tekstbestand die door EMI gegenereerd kan worden door de data uit VisualFlow te gebruiken.

#### 5.2.2.3 Stilstanden en omstellingen

De stilstanden en omstellingen kunnen worden gesimuleerd door het 'DowntimeEntity1'-object. Hierin kan worden aangegeven hoeveel tijd er tussen een stilstand of omstelling zit en hoe lang deze stilstand of omstelling duurt. In de praktijk is dit niet van tevoren duidelijk. JaamSim heeft ook een aantal 'Probability Distribution'-objecten zoals te zien in het 'Model Builder'-venster. Dit zijn objecten die willekeurige waarden kunnen genereren aan de hand van een waarschijnlijkheidsverdeling. Dit is een formule van kansberekening. Deze 'Probability Distribution'-objecten hebben de invoerwaarden minimum, gemiddelde en maximum nodig om de formule te berekenen. Ook deze waarden kunnen weer worden ingeladen met een 'FileToVector'-object gevuld met OEE-data uit de EMI database.

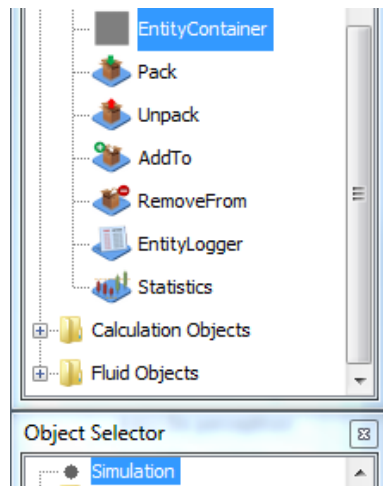
#### 5.2.2.4 Uitval

Tot slot kan de uitval van producten worden gesimuleerd met een 'Branch'-object en een 'BooleanSelector'-object. Het Branch-object kan entities naar andere objecten sturen. Dit kunnen er ook meer dan twee zijn. Hiermee is het Branch-object ook geschikt om de verschillende productieroutes te simuleren, maar in dit voorbeeld wordt het gebruikt voor de uitval. Het 'BooleanSelector'-object kan op basis van een bepaalde kans een 1.0 of een 0.0 waarde genereren. Deze kans kan ook weer extern worden ingeladen door een tekstbestand uit de EMI database. Vervolgens kan het Branch-object deze waarde gebruiken om te bepalen naar welk object de

huidige entity gaat. Dit zijn in dit geval twee 'EntitySink'-objecten. Dit zijn objecten die entities verwijderen van de simulatie. De ene kan gebruikt worden als uitval en de ander als output van de fabriek. Het aantal entities dat bij JaamSim-objecten zijn aangekomen worden door JaamSim gelogd in een log-bestand. Dit log-bestand kan dan weer door EMI worden uitgelezen aan het einde van de simulatie om de resultaten te interpreteren.

### 5.2.2.5 Containers

JaamSim heeft ook een aantal objecten om de containers van TN te simuleren zoals te zien is in Figuur 20.



Figuur 20: JaamSim container objecten

Noot. Rang, S. A. (2018, 11 juni). JaamSim container objecten [Foto], Geraadpleegd van 'JaamSim (2018-03)'

'EntityContainer'-objecten werken net als gewone entities in JaamSim, alleen kan een EntityContainer meerdere entities tegelijk bevatten. Entities kunnen worden toegevoegd en verwijderd van containers door de 'AddTo'- en 'RemoveFrom'-objecten. De containers zelf kunnen net als entities worden aangemaakt met een EntityGenerator-object. Hiermee kan het maximum van 475 containers worden aangehouden door bij het invoerveld van de EntityGenerator een maximum van 475 aan te geven. In de simulatie kan er na de perserij een 'AddTo'-object worden gebruikt om de rollen per container vervolgens door de trommels en harderij te verwerken. Daarna kan een 'RemoveFrom'-object worden gebruikt om, net als in de echte rollenfabriek, de containers te legen in de slijperij. Hierna kunnen de containers worden teruggestuurd naar de perserij om opnieuw gevuld te worden.

### 5.2.3 Conclusie

Deze paragraaf geeft een antwoord op de tweede deelvraag van dit onderzoek:

*Welk simulatie-softwarepakket is het meest geschikt om in EMI te implementeren?*

Het gebruik van bestaande software is noodzakelijk gezien de beperkte tijd die is gereserveerd voor het programmeren van het eindproduct. Na onderzoek naar verschillende simulatie-softwarepakketten is gebleken dat het lastig is om software te vinden die open source is en beschikbaar is voor commercieel gebruik. Na overleg met de stakeholders van EKB is besloten om JaamSim (2018) te gaan gebruiken, omdat deze software aan alle functionele en niet-functionele requirements voldoet.

Met JaamSim kunnen alle aspecten van de rollenfabriek van TN gesimuleerd worden. Dit is gebleken uit onderzoek naar de werking van JaamSim. De externe data moet echter wel ingeladen worden in de vorm van tekstbestanden, die het onoverzichtelijker maken. Ook het feit dat JaamSim simulaties worden opgeslagen in een config-bestand zonder opmaak maakt dat aanpassingen en het oplossen van problemen lastiger zijn dan wanneer simulaties bijvoorbeeld in XML-formaat zouden worden opgeslagen.

### 5.3 Machine learning

Dit hoofdstuk is bedoeld om een keuze te kunnen maken tussen verschillende soorten machine learning algoritmes:

*Welke machine learning algoritmes zijn het meest geschikt?*

#### 5.3.1 Algoritme types

ML kan worden onderverdeeld in vijf types:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Genetic algorithms (GA's)

Bij supervised learning worden verwachte outputs vergeleken met de daadwerkelijke outputs om het algoritme te trainen. Aangezien de optimale buffervoorraadgroottes onbekend zijn, zijn er geen verwachte outputs en kan supervised learning niet toegepast worden.

Unsupervised learning wordt gebruikt om data te clusteren, een histogram van de data te maken of overbodige data te vinden. Geen van deze toepassingen kan worden gebruikt om de buffergroottes te verbeteren.

Semi-supervised learning is een hybride tussen supervised- en unsupervised learning. Aangezien beiden niet geschikt zijn voor dit onderzoek is ook semi-supervised learning niet geschikt.

Volgens Van Otterlo en Wiering (2009) is reinforcement learning een algoritme die in een simulatie beloningen krijgt voor goede- en straffen krijgt voor slechte acties. Echter is reinforcement learning alleen geschikt voor simulaties waarin het algoritme meerdere acties uitvoert voordat de simulatie stopt. In dit onderzoek hoeft het algoritme maar een enkele actie uit te voeren, namelijk het vastleggen van buffervoorraadgroottes.

De output van een GA is een lijst van waarden. Deze waarden kunnen worden gebruikt om de buffervoorraadgroottes mee vast te leggen voor de simulaties. Een voordeel van GA's ten opzichte van reinforcement learning is dat er meerdere algoritmes tegelijk op zoek gaan naar een oplossing (de population). Dit vergroot de diversiteit van de oplossingen. Aangezien er voor complexe problemen (bijna) altijd meerdere oplossingen zijn, kan er geconcludeerd worden dat GA's van deze vijf types van ML de beste is om voor dit onderzoek te implementeren.

#### 5.3.2 Genetic algorithm frameworks

Volgens een software engineer van EKB komt het vaak voor dat de EKB-server stilgezet wordt of door een probleem crasht (M. Kok, persoonlijke communicatie, 9 mei 2018). Hieruit kan opgemaakt worden dat het mogelijk moet zijn om de population van het GA-framework op te kunnen slaan als back-up.

Om de implementatie binnen de beperkte tijd te kunnen implementeren in EMI is het ook noodzakelijk dat het GA-framework goede documentatie heeft.

Een andere functionele requirement is dat het mogelijk moet zijn om het NEAT-framework te gebruiken. Uit onderzoek van Stanley (2004) is gebleken dat het NEAT-framework in verschillende scenario's significant beter presteert dan andere GA's. Het aantal generaties die nodig waren om tot een oplossing te komen en het aantal berekende neural networks waren aanzienlijk lager.

Aangezien er in dit onderzoek wordt gewerkt met grote hoeveelheden data is de snelheid van NEAT een belangrijk punt.

Net als de simulatiesoftware moet het GA-framework gratis commercieel te gebruiken zijn en open source zijn. Daarnaast is de programmeertaal van het framework belangrijk om het eenvoudig in EMI te implementeren. Tabel 5 toont de resultaten van de vergelijkingen tussen een aantal GA-frameworks. De oorsprong van deze resultaten staat in Bijlage C.

Tabel 5: GA-framework afweging

NIET-FUNCTIONELE REQUIREMENTS	ACCORD (2017)	AForge (2013)	NUML (2017)	ENCOG (2018)	SHARPNEAT (2018)
Open source	1	1	1	1	1
Commercieel gebruik	1	1	1	1	1
Taal	1	1	1	1	1
FUNCTIONELE REQUIREMENTS					
Back-up	0,5	0	0	0	1
Documentatie	1	1	0,5	1	0,5
NEAT	0	0	0	1	1
<b>TOTAAL</b>	<b>4,5</b>	<b>4</b>	<b>3,5</b>	<b>5</b>	<b>5,5</b>

Het puntensysteem van Tabel 5 werkt als volgt:

- 1 punt: Volledige ondersteuning van het requirement
- 0,5 punt: Gedeeltelijke ondersteuning van het requirement. In het geval van de niet-functionele requirement 'Taal' wordt een halve punt toegekend als de taal van het GA-framework eenvoudig te combineren is met C#, zoals Java, Virtual Basic of C++.
- 0 punten: Geen ondersteuning van het requirement

### 5.3.3 Conclusie

Deze paragraaf geeft een antwoord op de derde deelvraag van dit onderzoek:

*Welke machine learning algoritmes zijn het meest geschikt?*

Van de vijf verschillende ML-types zijn GA's gekozen boven Reinforcement learning aangezien Reinforcement learning niet geschikt is voor het genereren van buffervoorraadgroottes. Daarnaast kunnen GA's gebruikt worden om een oplossing te vinden die nog niet bekend is, in tegenstelling tot de andere ML-types.

In Tabel 5 staat het resultaat van het vergelijkend onderzoek naar verschillende GA-frameworks. Aan de hand van deze requirements kan gesteld worden dat SharpNEAT het meest geschikte GA-framework is.

## 5.4 Experimenten

Dit hoofdstuk bevat het experimenteel onderzoek op de gerealiseerde software om uiteindelijk de laatste deelvraag te kunnen beantwoorden:

*Voldoen de machine learning algoritmes aan de verwachtingen van EKB?*

### 5.4.1 Benchmarking

Volgens de bedrijfsbegeleider mag het trainen van de algoritmes niet langer duren dan twee weken (A. Roelofsen, persoonlijke communicatie, 3 mei 2018). De looptijd van het trainen van de algoritmes is afhankelijk van de looptijd van de simulatie, de grootte van de population en het aantal generaties. De grootte van de population is een parameter die bij GA's vaak na het observeren van de eerste resultaten aangepast wordt. De initiële grootte van de population is vaak honderd. Het aantal generaties die nodig zijn om een oplossing te vinden is pas bekend wanneer er daadwerkelijk een oplossing wordt gevonden. Daarnaast is het aantal benodigde generaties door de willekeurigheid van GA's elke keer anders. Ook voor het aantal generaties is honderd een standaard waarde. Uitgaande van deze waarden mag de looptijd van een simulatie niet meer zijn dan:

$$\frac{2 \text{ weken}}{100 \cdot 100} = 120,96 \text{ seconden.}$$

Tabel 6: JaamSim Benchmarks over februari 2018

STUKS PER ENTITY	BUFFERS (ENTITIES)	TIJD
1	1	>15 min.
1000	1	~30,2 sec.
1 kg	1	~176,2 sec.
500 kg	1	~3,57 sec.

In Tabel 6 staan de resultaten van de uitgevoerde benchmarks van JaamSim gebruik makende van de EMI-data van februari 2018. JaamSim is in dit geval op de achtergrond en zonder GUI uitgevoerd, omdat de server waar EMI draait geen grafische kaart heeft. De benchmarks zijn elk met een andere hoeveelheid rollen per JaamSim entity gesimuleerd.

Uit de benchmarks is gebleken dat het aantal rollen per entity van grote invloed is op de looptijd van de simulaties en dat 500 kg per entity de snelste simulatie was. Een nadeel van het simuleren van entiteiten van 500 kg is dat volgens de data uit VisualFlow de containers van TN lang niet altijd het maximum van 500 kg aan rollen bevatten, maar ook de helft of ergens tussen de helft en het maximum. Dit zou resulteren in een onrealistische hoeveelheid rollen aangezien een enkele rol slechts een aantal gram weegt.

Aan de hand van de benchmarks is gekozen om in de experimenten entiteiten van 1000 rollen te simuleren. De geschatte tijd om honderd generaties met een population van honderd te simuleren wordt dan:  $30,2 \text{ seconden} \cdot 100 \cdot 100 \approx 3,5 \text{ dag}$ .

### 5.4.2 Simulaties zonder VisualFlow buffer-data

In verband met de looptijd van de simulaties zoals beschreven in de vorige paragraaf is gekozen om de eerste simulaties zonder buffer-data uit VisualFlow uit te voeren. De gemiddelde troommeltijd is wel uit de data van Visual Flow berekend en is ongeveer 7 uur en 23,5 minuut.

SharpNEAT bevat een aantal verschillende activation functions zoals de sigmoïd, ReLU, leaky ReLU, hyperbolische tangens en sinus. Zoals beschreven in hoofdstuk 4.4.3 wordt de ReLU activation function steeds meer gebruikt. Een nadeel is dat de ReLU niet goed om kan gaan met grote getallen die in dit geval nodig zijn voor de buffervoorraadgroottes. De weights van het neural network moeten hierdoor ook hoog zijn en er zijn meer weights nodig. Hierdoor wordt het neural

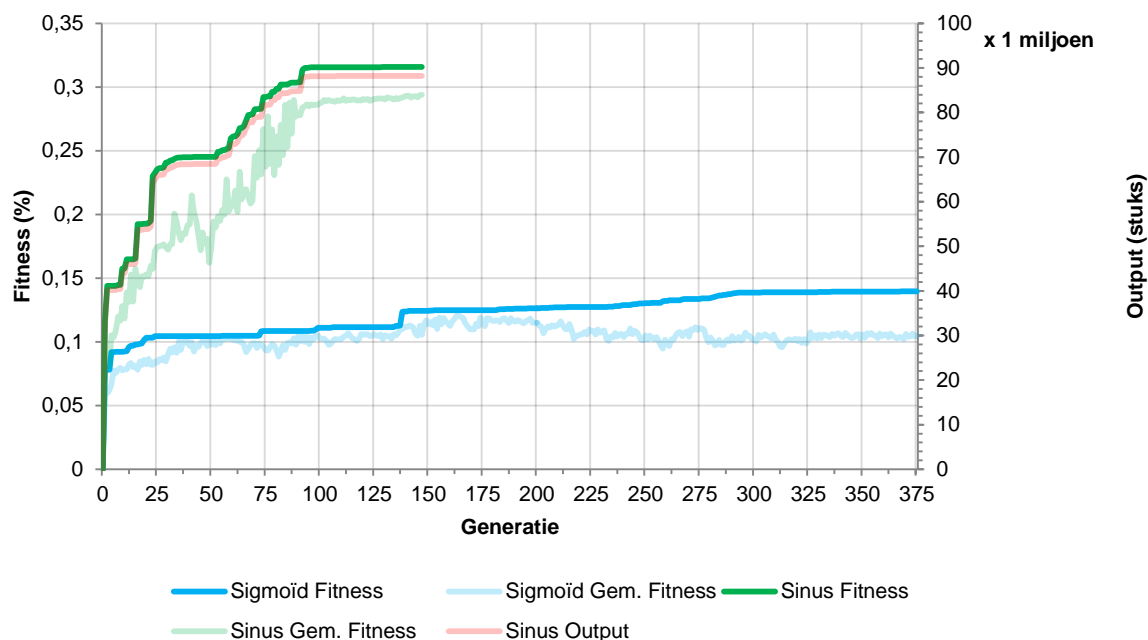


network overbodig complex. Hierom zijn voor de eerste experimenten de sigmoïd en sinus activation functions gekozen. Deze uitkomsten zijn vermenigvuldigd met de maximum buffervoorraadgrootte om zo de uitkomst van het neural network te berekenen. De maximum buffervoorraadgrootte is als volgt vastgesteld:  $\text{max. buffervoorraadgrootte} = \frac{\text{max. totale output}}{\text{productielijnen}} \cdot \text{OEE}$ . Waarbij de maximale totale output afhankelijk is van het aantal 'EntityGenerators' (zie hoofdstuk 5.2.2.1) in de simulatie en het aantal productielijnen in het geval van TN 44 is (exclusief de trommels).

In Tabel 7 staan de parameters van de simulaties weergegeven. De sigmoïd activation function is aangepast naar de versie die volgens Stanley (2004) het beste werkte. Verder zijn de termen '+13' en '+1' toegevoegd om 1 als startgetal te hebben in plaats van 0,5 voor de sigmoïd en 0 voor de sinus. Dit is gekozen om te beginnen met simuleren volgens de stelling van Goldratt en Cox (2007) dat de buffervoorraadgroottes zo laag mogelijk moeten zijn. Het algoritme kan dan in latere generaties langzaam de buffervoorraadgroottes verhogen waar dit nodig is.

Tabel 7: Simulatie parameters

ACTIVATION FUNCTION NAAM	ACTIVATION FUNCTION	BEGIN BUFFERS (ENTITIES)	SIMULATIE STARTDATUM	SIMULATIE EINDDATUM
sigmoïd (aangepast)	$\frac{1}{1 + e^{-4.9x+13}} + 1$	1	1 feb. 2018	28 feb. 2018
sinus (aangepast)	$\sin(2x) + 1$	1	1 feb. 2018	28 feb. 2018



Figuur 21: Simulatie resultaten sigmoïd- en sinus activation functions

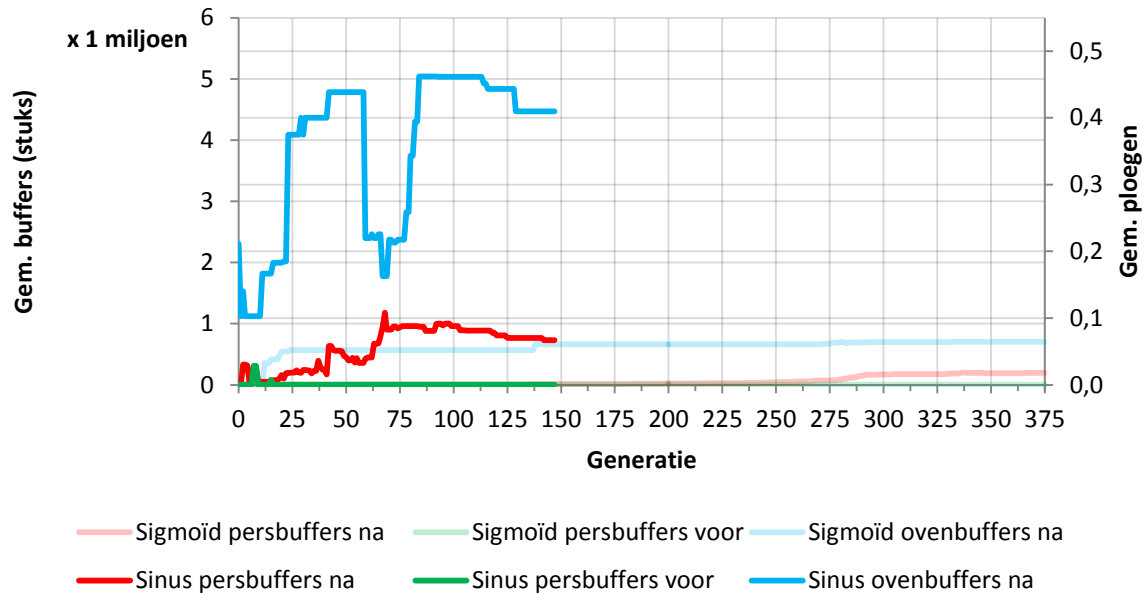
De fitness is berekend volgens de formule:

$$\text{fitness}(\%) = \frac{\text{output}}{\text{max.output}} \cdot 50\% + \left(1 - \frac{\text{buffers}}{\text{max.buffers}}\right) \cdot 50\%.$$

In Figuur 21 is te zien dat de sinus- een stuk beter presteert dan de sigmoïd activation function. De output van de simulatie met de sigmoïd activation function werd op het moment van simuleren nog niet bijgehouden, maar ligt waarschijnlijk net als bij de sinus activation function ongeveer even hoog als de fitness (op de rechter as). De simulatie met de sigmoïd activation function heeft veel meer generaties gesimuleerd dan de andere simulatie. Dit in verband met de looptijd van de simulatie. De buffervoorraadgroottes waren bij deze simulatie een stuk lager, hierdoor waren er minder entiteiten om te simuleren. Dit resulteert in kortere looptijden van de simulaties zoals ook bevonden in de benchmarks. Door deze kortere looptijden konden er in dezelfde tijd meer generaties gesimuleerd worden.

De daadwerkelijke fabrieksooutput van TN in februari was ongeveer 95,9 miljoen stuks. De simulatie met de sinus activation function haalde een fabrieksooutput van ongeveer 88,2 miljoen stuks. Ondanks dat de buffers tussen de productielijnen aan het begin van de simulatie leeg waren, terwijl ze in het echt gevuld zijn, is de fabrieksooutput van de simulatie ongeveer 91,9% van de daadwerkelijke fabrieksooutput.

In Figuur 22 is te zien dat de bijbehorende buffervoorraadgroottes net als de fitness en fabrieksooutput sterk van elkaar verschillen per activation function. Opvallend is de dip van de buffers na de ovens (tussen de ovens en de slijperij) bij de sinus activation function tussen generatie 57 en 75. Tussen deze generaties steeg zowel de fitness als de output van het algoritme terwijl de buffers daalde met ongeveer 2,5 miljoen stuks.



Figuur 22: Simulatie buffervoorraadgroottes sigmoïd- en sinus activation functions

## LITERATUUR

- Accord. (2017, 2 december). License. Geraadpleegd van <http://accord-framework.net/license.html>
- Accord. (2017, 7 juli). DoubleArrayChromosome. Geraadpleegd van <https://github.com/accord-net/framework/blob/development/Sources/Accord.Genetic/Chromosomes/DoubleArrayChromosome.cs>
- Accord. (2017). Accord.NET Framework. Geraadpleegd van [http://accord-framework.net/docs/html/R\\_Project\\_Accord\\_NET.htm](http://accord-framework.net/docs/html/R_Project_Accord_NET.htm)
- Accord. (2018, 27 maart). Accord GitHub code. Geraadpleegd van <https://github.com/accord-net/framework>
- AForge. (2012). License. Geraadpleegd van <http://www.aforgenet.com/framework/license.html>
- AForge. (2013). AForge.NET docs. Geraadpleegd van <http://www.aforgenet.com/framework/docs/>
- Allen, M., Spencer, A., & Gibson, A. (2015). Right cot, right place, right time: improving the design and organisation of neonatal care networks – a computer simulation study.. *Health Service and Delivery Research*, 3(20), 9-10. Geraadpleegd van <https://www.ncbi.nlm.nih.gov/books/NBK293948/>
- AnyLogic. (2018, 17 mei). Software Licensing Agreement for AnyLogic 8.x.x. Geraadpleegd van [https://www.anylogic.com/upload/license\\_agreements/software-licensing-agreement-for-anylogic.pdf](https://www.anylogic.com/upload/license_agreements/software-licensing-agreement-for-anylogic.pdf)
- AnyLogic. (2018). Simulation Software for Every Business Challenge. Geraadpleegd van <https://www.anylogic.com/features/>
- AnyLogic. (2018). AnyLogic downloads. Geraadpleegd van <https://www.anylogic.com/downloads/#todownload>
- Apache. (2018). APACHE LICENSE, VERSION 2.0 (CURRENT). Geraadpleegd van <http://www.apache.org/licenses/>
- Arena. (2018). Manufacturing Simulation Software. Geraadpleegd van <https://www.arenasimulation.com/industry-solutions/industry/manufacturing-simulation-software>
- Arena. (2018). Arena Compare Products. Geraadpleegd van <https://www.arenasimulation.com/academic/compare-products>
- Arena. (2018). Arena Entities\_Attributes [Video]. Geraadpleegd van <http://embed.vidyard.com/share/XZeGJynckuAXAy08SLXFyQ>
- AnyLogic. (2018). AnyLogic Manufacturing [Video]. Geraadpleegd van <https://youtu.be/75GCZqmNQLE>
- AnyLogic. (2018). AnyLogic Help. Geraadpleegd van <https://help.anylogic.com/index.jsp>
- Capterra. (z.d.). Simulation Software. Geraadpleegd van <https://www.capterra.com/simulation-software/>
- Costas, J., Ponte, B., De la Fuente, D., Pino, R., & Puche, J. (2015). Applying Golratt's Theory of Constraints to reduce the Bullwhip Effect through agent-based modeling. *Elsevier*, 42(4), 2049-2060. Geraadpleegd van <https://paperdownload.me/wp-content/uploads/2017/11/5693-applying-goldratts-theory-constraints-reduce-bullwhip-effect-agent-based-modeling.pdf>

- EKB. (z.d.). EMI Database diagram [Database]. Geraadpleegd van (local)\SQLEXPRESS.EMI\_NN\dbo.DatabaseSchema
- EKB. (2017, 21 februari). EKB Groep (Totaal) [Intranet]. Geraadpleegd van <http://intranet.ekb.nl/Documenten%20Personeelszaken/Organogrammen%20EKB%20Groep.pdf>
- FlexSim. (2018). Factory Simulation. Geraadpleegd van <https://www.flexsim.com/factory-simulation/>
- FlexSim. (2018). FlexSim problem solved.. Geraadpleegd van <https://www.flexsim.com/flexsim/>
- GNU. (2016, 18 november). GNU Lesser General Public License. Geraadpleegd van <http://www.gnu.org/licenses/lgpl.html>
- Goldratt, E. M. (2000). *Het Doel*. Utrecht, Nederland: Het Spectrum B.V..
- Goldratt, E. M., & Cox, J. (2017, 2 april). The Goal, A Process of Ongoing Improvement. Geraadpleegd van <http://www.2ndbn5thmar.com/lean/Notes%20on%20The%20Goal.pdf>
- Green, C. (2016). Licensing. Geraadpleegd van <http://sharpneat.sourceforge.net/licensing.html>
- Green, C. (2016, 30 september). NeatGenomeXmlIO.cs. Geraadpleegd van <https://github.com/colgreen/sharpneat/blob/master/src/SharpNeatLib/Genomes/Neat/NeatGenomeXmlIO.cs>
- Green, C. (2018, 13 juni). SharpNeat GitHub code. Geraadpleegd van <https://github.com/colgreen/sharpneat>
- Heaton Research. (2017, 3 september). Encog GitHub code. Geraadpleegd van <http://numl.net/api/index.html>
- Heaton, J. (2011, oktober). Programming Neural Networks with Encog3 in C#. Geraadpleegd van <https://s3.amazonaws.com/heatonresearch-books/free/Encog3CS-User.pdf>
- Heaton, J. (2018). Encog Copyright. Geraadpleegd van <https://www.heatonresearch.com/legal/copyright.html>
- Hussung, T. (2016, 10 maart). What Is the Software Development Life Cycle? [Blogpost]. Geraadpleegd van <https://online.husson.edu/software-development-cycle/>
- JaamSim Development Team. (2015, 25 november). JaamSim License. Geraadpleegd van <https://github.com/jaamsim/jaamsim/blob/master/LICENSE>
- JaamSim Development Team. (2017, 7 september). JaamSim User Manual. Geraadpleegd van <https://jaamsim.com/docs/JaamSim%20User%20Manual%202017-10.pdf>
- JaamSim Development Team. (2018). JaamSim (2018-03) [Software]. Geraadpleegd van <https://jaamsim.com/index.html>
- Juarez, S. (2017, 7 september). NUML GitHub code. Geraadpleegd van <https://github.com/sethjuarez/numl>
- Juarez, S. (2017). API Documentation. Geraadpleegd van <http://numl.net/api/index.html>
- LeanSixSigma. (2018). Wat is Lean? Geraadpleegd van <https://www.sixsigma.nl/wat-is-lean>
- Managementmodellensite. (2018). Theory of Constraints: Goldratt. Geraadpleegd van <https://managementmodellensite.nl/theory-constraints-goldratt/#.WyJBuNUzbRa>

- Monier, L. (2016, 30 april). Deep Learning Class #1 - Go Deep or Go Home [Slide 23]. Geraadpleegd van <https://www.slideshare.net/holbertonschool/deep-learning-keynote-1-by-louis-monier>
- Nielsen, M. (2017, december). Neural Networks and Deep Learning. Geraadpleegd van <http://neuralnetworksanddeeplearning.com/index.html>
- Nordgren, B. (2007, 10 september). FlexSim Simulation Software [Video]. Geraadpleegd van <https://www.youtube.com/watch?v=t620IkFkw28>
- Open Source Initiative. (z.d.). The MIT License. Geraadpleegd van <https://opensource.org/licenses/MIT>
- Procesverbeteren.nl. (2017, 24 augustus). Introductie Lean: de slanke organisatie. Geraadpleegd van <https://www.procesverbeteren.nl/LEAN/leanmanufacturing.php#definitie>
- Procesverbeteren.nl. (2017, 6 september). Introductie TOC: de ongelimiteerde organisatie. Geraadpleegd van <http://www.procesverbeteren.nl/TOC/ToC.php>
- Rang, S. A. (2018, 3 april). EMI modules van de rollenfabriek van Tsubaki Nakashima [Foto]. Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/Default.aspx](http://emi-demo.ekb.nl/EMI_NNN/Default.aspx)
- Rang, S. A. (2018, 3 april). EMI OEE analyse van de rollenfabriek van Tsubaki Nakashima [Foto]. Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/OEE/OEEAnalyse.aspx](http://emi-demo.ekb.nl/EMI_NNN/OEE/OEEAnalyse.aspx)
- Rang, S. A. (2018, 11 juni). JaamSim GUI [Foto]. Geraadpleegd van JaamSim (2018-03)
- Rang, S. A. (2018, 3 april). EMI OEE status categorieën van de rollenfabriek van Tsubaki Nakashima [Foto]. Geraadpleegd van [http://emi-demo.ekb.nl/EMI\\_NNN/OEE/OEEAnalyse.aspx](http://emi-demo.ekb.nl/EMI_NNN/OEE/OEEAnalyse.aspx)
- Rang, S. A. (2018, 11 juni). JaamSim container objecten [Foto]. Geraadpleegd van JaamSim (2018-03)
- Rojas, R. (1996). The Backpropagation Algorithm. Geraadpleegd van <https://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf>
- Santos, L. (2018). Rectified-Linear unit Layer. Geraadpleegd van [https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/relu\\_layer.html](https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/relu_layer.html)
- Sharma, A. (2017, 30 maart). Rectified-Linear unit Layer. Geraadpleegd van <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>
- SIMUL8. (2018). SIMUL8 customer success case studies. Geraadpleegd van <https://www.simul8.com/case-studies/>
- SIMUL8. (2018). International & Educational Prices. Geraadpleegd van <https://www.simul8.com/shop/intprices>
- SIMUL8. (2018). Assembly Line [Video]. Geraadpleegd van <https://www.simul8.com/videos/assembly-line>
- Stanley, K. O. (2004, augustus). Efficient Evolution of Neural Networks through Complexification. Geraadpleegd van <http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf>
- Tutorials Point. (2016). Genetic Algorithms. Geraadpleegd van [https://www.tutorialspoint.com/genetic\\_algorithms/genetic\\_algorithms\\_tutorial.pdf](https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_tutorial.pdf)
- Van Otterlo, M., & Wiering, M. (2009). Reinforcement Learning and Markov Decision Processes. Geraadpleegd van [http://www.ai.rug.nl/~mwiering/Intro\\_RLBOOK.pdf](http://www.ai.rug.nl/~mwiering/Intro_RLBOOK.pdf)



## BIJLAGEN

### Bijlage A: Plan van Aanpak

Pagina's **Fout! Bladwijzer niet gedefinieerd.**-**Fout! Bladwijzer niet gedefinieerd.**





























































## Bijlage B: Simulatiesoftware requirements onderzoek

### AnyLogic

AnyLogic is een simulatie-softwarepakket voor 'supply chains', 'transportation', 'manufacturing' en nog een aantal industriële toepassingen (Bijlage B, Figuur 1).

AnyLogic is used in these industries:






[MORE INDUSTRIES](#)

Bijlage B, Figuur 1: AnyLogic industriële toepassingen

Noot. Herdrukt van "Simulation Software for Every Business Challenge", door AnyLogic, (2018). Geraadpleegd van <https://www.anylogic.com>

### Open Source en commercieel gebruik

Volgens hoofdstuk 2, artikel b7 van de license agreement voor AnyLogic (AnyLogic, 2018) is het verboden om de software te kopiëren, gebruiken of meeleveren met een ander softwarepakket zoals EMI tenzij een ander artikel in de software agreement dit toestaat. Artikel d van hetzelfde hoofdstuk gaat over de gratis universiteitseditie van de software. Ook volgens dit artikel is het niet toegestaan de software voor commerciële doeleinden te gebruiken. Andere edities van de AnyLogic software hebben alleen een gratis 'probeerversie' (Bijlage B, Figuur 2).

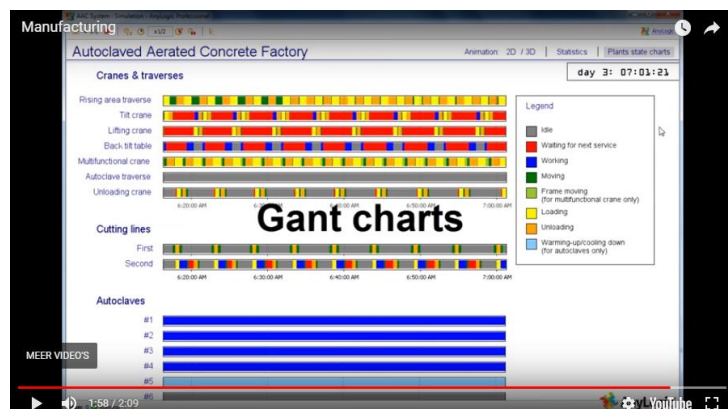
			
	FREE VERSION DOWNLOAD	DOWNLOAD	DOWNLOAD
		ASK FOR A QUOTE	ASK FOR A QUOTE
availability	free	free 60-day trial	free 60-day trial

Bijlage B, Figuur 2: AnyLogic edities

Noot. Herdrukt van "AnyLogic downloads", door AnyLogic, (2018). Geraadpleegd van <https://www.anylogic.com/downloads/#todownload>

### Stilstanden, omstellingen en uitval

In een demovideo over de manufacturing toepassing van AnyLogic is te zien dat stilstanden, omstellingen en uitval gesimuleerd en statistisch getoond kunnen worden (Bijlage B, Figuur 3).



Bijlage B, Figuur 3: AnyLogic stilstanden, omstellingen en uitval

Noot. Herdrukt van "AnyLogic Manufacturing", door AnyLogic [Video], (2018). Geraadpleegd van <https://youtu.be/75GCZqmNQLE>

*Buffervoorraden en verschillende producten op verschillende productielijnen*

In dezelfde demovideo wordt ook duidelijk gemaakt dat buffervoorraden en verschillende producten op verschillende productielijnen te simuleren zijn (Bijlage B, Figuur 4).



*Bijlage B, Figuur 4: AnyLogic buffervoorraden en verschillende producten op verschillende productielijnen*  
Noot. Herdrukt van "AnyLogic Manufacturing" [Video], door AnyLogic (2018). Geraadpleegd van <https://youtu.be/75GCZqmNQLE>

*Externe data*

Volgens hoofdstuk "Advanced Modeling with Java" paragraaf "Integrating Models with External Applications" van de documentatie van AnyLogic is het mogelijk om met Java data uit andere applicaties en ook databases in te laden (AnyLogic, 2018).

## Arena

Arena is een simulatie-softwarepakket voor onder andere 'manufacturing', 'supply chains' en 'logistics' (<https://www.arenasimulation.com/>).

### Open source en commercieel gebruik

Ook Arena heeft zowel academische versies als betaalde versies. Een overzicht van deze academische versies is weergegeven in Bijlage B, Figuur 5. Echter is geen enkele versie van Arena beschikbaar voor commercieel gebruik, hierdoor is ook duidelijk dat Arena niet open source is.

#### Arena Academic Offerings

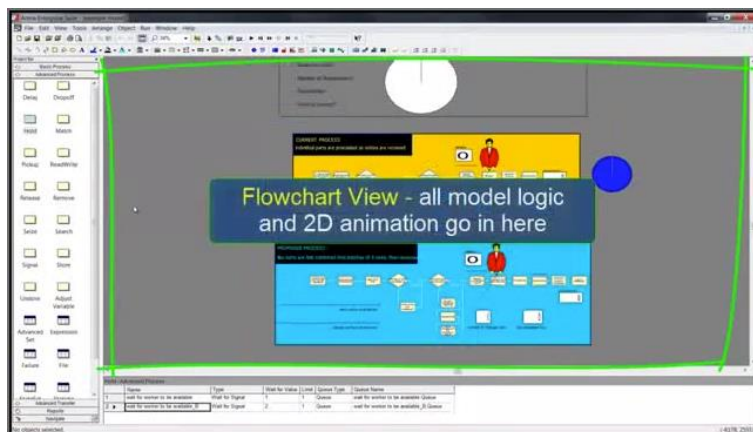
	Student Version	Academic Laboratory	Academic Research
Number of Users	1	30	1
OptQuest for Arena		✓	✓
Arena Professional Edition	✓	✓	✓
Network Capable		✓	✓
Model Size Limitations	✓		
Access to Selected Online Help	✓	✓	✓
Access to Arena Tech Support		✓	✓
Access to Teaching Materials		✓	
Commercial Use Restrictions	✓	✓	✓
Price	Free	Contact Us	Contact Us

Bijlage B, Figuur 5: Arena academische versies

Noot. Herdrukt van "Arena Compare Products", door Arena, (2018). Geraadpleegd van <https://www.arenasimulation.com/academic/compare-products>

### Functionele requirements

In een demovideo van Arena worden de functionele requirements stilstanden, omstellingen, uitval, buffervoorraden, verschillende producten op verschillende productielijnen en externe data getoond (Bijlage B, Figuur 6).



Bijlage B, Figuur 6: Arena functionele requirements

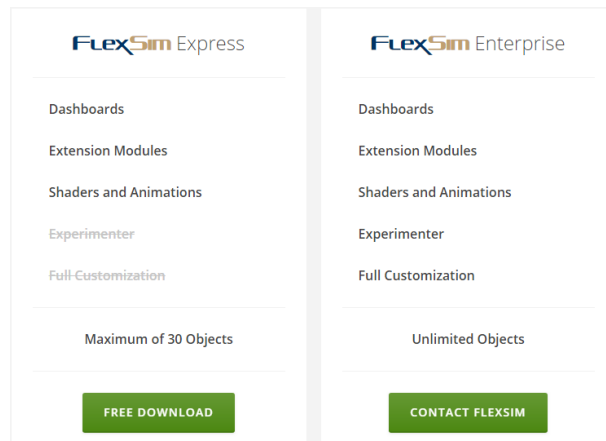
Noot. Herdrukt van "Arena Entities\_Attributes" [Video], door Arena, (2018). Geraadpleegd van <http://embed.vidyard.com/share/XZeGJynckuAXAy08SLXfYQ>

## FlexSim

FlexSim is een simulatie-softwarepakket waarmee 'manufacturing', 'healthcare', 'supply chain', 'discrete event' en nog vele andere industriële toepassingen gesimuleerd kunnen worden (<https://www.flexsim.com/factory-simulation/>).

### *Open source en commercieel gebruik*

FlexSim heeft de versies 'Express' en 'Enterprise' (Bijlage B, Figuur 7). De Express versie is gratis te gebruiken, maar niet voor commercieel gebruik en dus ook niet open source. FlexSim heeft ook een academische versie, maar ook deze is niet voor commercieel gebruik.

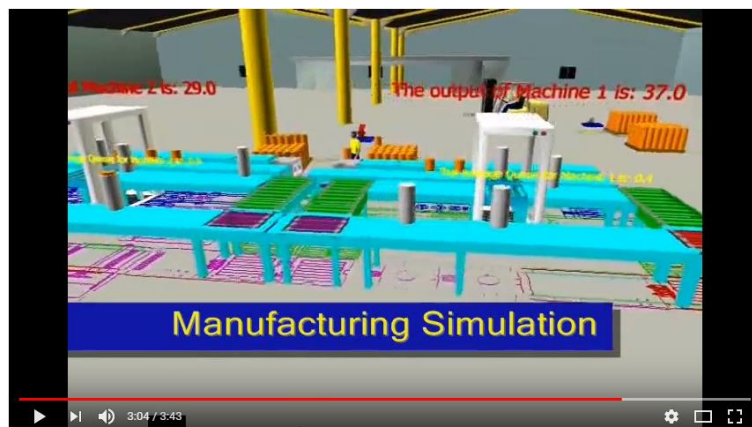


Bijlage B, Figuur 7: FlexSim versies

Noot. Herdrukt van "FlexSim problem solved.", door FlexSim, (2018). Geraadpleegd van <https://www.flexsim.com/flexsim/>

### *Functionele requirements*

In een video van FlexSim is te zien dat de basis functionaliteiten gesimuleerd kunnen worden in 3D. Er wordt onder andere een 'manufacturing' voorbeeldsimulatie getoond waarin producten worden geproduceerd terwijl de output wordt bijgehouden (Bijlage B, Figuur 8).



Bijlage B, Figuur 8: FlexSim Simulation Software

Noot. Herdrukt van "FlexSim Simulation Software" [Video], door Bill Nordgren, (10 september 2007). Geraadpleegd van <https://youtube.com/watch?v=t620lkFkw28>

### **JaamSim**

JaamSim is een 'discrete event' simulatie-softwarepakket geschreven in Java en wordt sinds 2016 op GitHub bijgewerkt (<https://jaamsim.com/>).

#### *Open source en commercieel gebruik*

Volgens artikel 2 en 3 van de license agreement van JaamSim (JaamSim Development Team, 2015) mag de software gekopieerd, aangepast en geredistribueerd worden zolang de aanpassingen worden vermeld en een kopie van de license agreement meegeleverd wordt. Ook zijn alle versies die vanaf 2016 op GitHub zijn uitgebracht gratis te downloaden. Dit betekent dat de software zowel open source is, als beschikbaar voor commercieel gebruik.

#### *Functionele requirements*

De functionele requirements worden beschreven in hoofdstuk 5.2.2 aangezien JaamSim een belangrijk onderdeel is van dit onderzoek en de realisatie van het eindproduct.

## SIMUL8

SIMUL8 is een simulatie-softwarepakket waarmee 'healthcare', 'manufacturing', 'supply chain & logistics' en andere industriële toepassingen gesimuleerd kunnen worden (<https://www.simul8.com/>).

### Open source en commercieel gebruik

SIMUL8 heeft vijf verschillende versies. Deze versie zijn echter allemaal betaald (Bijlage B, Figuur 9). Dit betekent dat SIMUL8 geen open source is en ook niet beschikbaar voor commercieel gebruik.

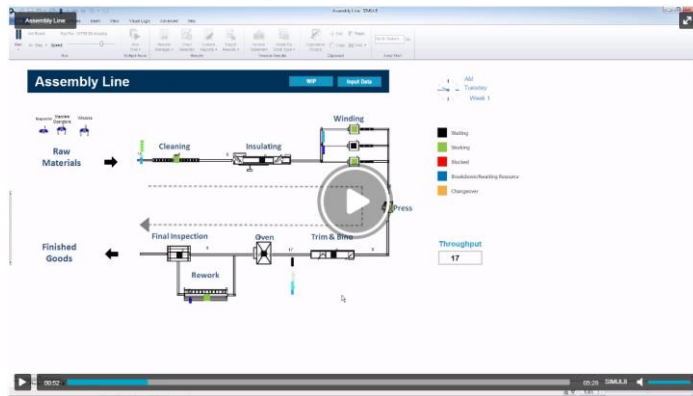
Product	\$ US Dollars	£ UK Sterling	€ EU Euros
Main Products			
<a href="#">SIMUL8 Basic 2018</a>	\$1995	£1295	€1995
<a href="#">SIMUL8 Professional 2018</a>	\$4995	£2995	€4995
<a href="#">SIMUL8 Professional Plus 2018</a>	\$7495	£4495	€7495
<a href="#">SIMUL8 Team 2018</a>	\$19995	£12995	€19995
<a href="#">SIMUL8 Educational Site License</a>	\$1995	£1250	€1995
<a href="#">Learning SIMUL8: The Complete Guide</a>	\$225	£160	€225
<a href="#">Learning SIMUL8: The Complete Guide (Academic Edition)</a>	\$115	£85	€115
Upgrades			
<a href="#">SIMUL8 Basic 2016 to SIMUL8 Basic 2018</a>	\$695	£495	€695
<a href="#">SIMUL8 Basic 2016 to SIMUL8 Professional Plus 2018</a>	\$5995	£3995	€5995
<a href="#">SIMUL8 Basic to SIMUL8 Professional 2018</a>	\$3295	£2095	€3295
<a href="#">SIMUL8 Standard to SIMUL8 Professional 2018</a>	Price on request	Price on request	Price on request
<a href="#">SIMUL8 Professional 2016 to SIMUL8 Professional 2018</a>	\$899	£540	€899
<a href="#">Annual Maintenance - SIMUL8 Basic</a>	\$595	£395	€595
<a href="#">Annual Maintenance - SIMUL8 Professional</a>	\$899	£540	€899
SIMUL8 Plug Ins			
<a href="#">OptQuest for SIMUL8</a>	\$495	£329	€495
<a href="#">OptQuest for SIMUL8 Educational Site License</a>	\$743	£530	€743
<a href="#">Stat::Fit for SIMUL8</a>	\$245	£149	€245
<a href="#">Stat::Fit for SIMUL8 Educational Site License</a>	\$357	£255	€357

Bijlage B, Figuur 9: SIMUL8 versies

Noot. Herdrukt van "International & Educational Prices", door SIMUL8, (2018). Geraadpleegd van <https://www.simul8.com/shop/intprices>

### Functionele requirements

In een voorbeeldvideo van SIMUL8 zijn de verschillende functionele requirements te zien (SIMUL8, 2018). Ook kan volgens SIMUL8 (2018) externe data ingeladen worden via Microsoft Excel, VB en C++.



Bijlage B, Figuur 10: SIMUL8 functionele requirements  
Noot. Herdruckt van "Assembly Line" [Video], door SIMUL8, (2018). Geraadpleegd van <https://www.simul8.com/videos/assembly-line>



## Bijlage C: Genetic algorithm framework requirements onderzoek

### Accord

Het Accord .NET Framework is geschreven in C# en kan niet alleen voor GA's worden gebruikt, maar ook voor supervised- en unsupervised learning (<http://accord-framework.net/>).

#### *Open source en commercieel gebruik*

Volgens de license agreement van Accord (Accord, 2017) mag de code aangepast worden en gratis commercieel gebruikt worden. De code is vanaf GitHub te downloaden (Accord, 2018).

#### *Documentatie*

De documentatie van Accord (Accord, 2017) is te vinden op de website en is zeer uitgebreid. Alle modules van de software staan in detail beschreven.

#### *Back-up*

In de documentatie van Accord staan geen aanwijzingen voor het kunnen opslaan en inladen van de population. Echter is er in de code van Accord (Accord, 2017) een optie om een 'chromosome' op te slaan als tekst. De population bestaat uit leden die chromosomes heten. Als alle chromosomes opgeslagen kunnen worden als tekst kan er eigen code worden geschreven om de volledige population op te slaan in een tekstbestand en deze later in te laden in Accord. Aangezien er eigen code moet worden geschreven om een back-up te kunnen maken krijgt Accord voor dit onderdeel maar een half punt.

#### *NEAT*

In de documentatie van Accord is een 'Genetic'-module te zien. Deze module bevat echter geen functionaliteiten voor het NEAT-framework. Ook elders in de documentatie en code zijn hier geen aanwijzingen voor gevonden.

### AForge

Het AForge.NET Framework is een open source framework geschreven in C# en kan worden gebruikt voor GA's, het verwerken van afbeeldingen en neural networks (<http://www.aforgenet.com/framework/>).

#### *Commercieel gebruik*

Volgens de license-pagina van AForge (AForge, 2012) is de 'LGPL v3 license' van kracht. Volgens deze license (GNU, 2016) mag de code worden aangepast zolang de license meegeleverd wordt en mag de software gratis commercieel gebruikt worden.

#### *Documentatie*

De documentatie van AForge (AForge, 2013) is te vinden op de website en is zeer uitgebreid. Alle modules van de software staan in detail beschreven.

#### *Back-up*

In de documentatie staan geen aanwijzingen voor het kunnen opslaan of inladen van de population.

#### *NEAT*

In de documentatie van AForge is een 'Genetic'-module te zien. Deze module bevat echter geen functionaliteiten voor het NEAT-framework. Ook elders in de documentatie en code zijn hier geen aanwijzingen voor gevonden.

### NUML

Het NUML-framework is een machine learning framework voor GA's, supervised learning en unsupervised learning door Seth Juarez ([numl.net/index/html](http://numl.net/index/html)).

#### *Open source en commercieel gebruik*

De code van het NUML-framework is beschikbaar op GitHub (Juarez, 2017). Verder gebruikt het NUML-framework de 'MIT License' ([numl.net/index/html](http://numl.net/index/html)). Volgens deze license (Open Source Initiative, z.d.) mag de code gratis worden aangepast en commercieel gebruikt worden zolang de license meegeleverd wordt.

#### *Documentatie en taal*

Hoewel de documentatie pagina van de website leeg is, is er wel documentatie op de 'API reference'-pagina van de site (Juarez, 2017). Echter is deze documentatie niet zo uitgebreid als de documentatie van andere frameworks. Hoewel ook deze documentatie alle modules van de code bevat, is de uitleg bij de verschillende methodes, classes en velden minimaal. De documentatie van NUML is hierdoor een halve punt waard. Wel is uit de documentatie op te maken dat het framework is geschreven in C#.

#### *Back-up*

In de documentatie staan geen aanwijzingen voor het kunnen opslaan of inladen van de population.

#### *NEAT*

In de documentatie van het NUML-framework is een 'Genetic'-module te zien. Deze module bevat echter geen functionaliteiten voor het NEAT-framework. Ook elders in de documentatie en code zijn hier geen aanwijzingen voor gevonden.

#### **Encog**

Het Encog Machine Learning Framework is een framework voor neural networks, GA's, NEAT en 'support vector machines' geschreven voor Java en C# door Jeff Heaton (<https://www.heatonresearch.com/encog/>).

#### *Open source en commercieel gebruik*

De code van Encog is beschikbaar op GitHub (Heaton Research, 2017). Verder gebruikt het Encog Machine Learning Framework de 'Apache License 2.0' (Heaton, 2018). Volgens deze license (Apache, 2018) mag de code gratis worden aangepast en commercieel gebruikt worden zolang de license meegeleverd wordt.

#### *Documentatie*

De documentatie van Encog (<https://www.heatonresearch.com/encog/>) bevat documentatie over de installatie, het verder programmeren van het framework en het gebruik van het framework voor zowel Java als C#. De C# documentatie (Heaton, 2011) bevat ook veel informatie over onder andere activation functions, het verzamelen en normaliseren van goede data en de werking van Encog GA's.

#### *Back-up*

In de documentatie staan geen aanwijzingen voor het kunnen opslaan of inladen van de population.

#### *NEAT*

In de documentatie staat een korte uitleg van het NEAT-framework en een beschrijving hoe NEAT gebruikt kan worden in het Encog Machine Learning Framework.

#### **SharpNEAT**

SharpNEAT is een framework speciaal gemaakt voor het NEAT-framework in C# door Colin Green (<http://sharpneat.sourceforge.net/>).

#### *Open source en commercieel gebruik*

De code van SharpNEAT is beschikbaar op GitHub (Green, 2018). Verder gebruikt SharpNEAT, net als het NUML-framework, de 'MIT License' (Green, 2016). Volgens deze license (Open Source Initiative, z.d.) mag de code gratis worden aangepast en commercieel gebruikt worden zolang de license meegeleverd wordt.

#### *Documentatie*

Hoewel er op de website en GitHub pagina's van SharpNEAT geen documentatie te vinden is, bevat de code van SharpNEAT zeer gedetailleerd commentaar. Hierdoor is de documentatie van SharpNEAT een halve punt waard.

#### *Back-up*

In de 'SharpNeatLib'-module van SharpNEAT zitten functionaliteiten voor het opslaan en inladen van de NEAT-population van- en naar een XML-bestand (Green, 2016). Hierdoor kan de population ingeladen worden mochten er problemen optreden of de simulaties gepauzeerd worden.